
**BRNO UNIVERSITY OF TECHNOLOGY**
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**

**FACULTY OF INFORMATION TECHNOLOGY**
**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**

**DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA**
**ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ**


# TRANSMISSION OF DIGITAL INFORMATION OVER AUDIO

**PŘENOS DIGITÁLNÍ INFORMACE POMOCÍ ZVUKU**

**BACHELOR'S THESIS**
**BAKALÁŘSKÁ PRÁCE**


**AUTHOR** — **MICHAEL BUJNOVSKÝ**
**AUTOR PRÁCE**

**SUPERVISOR** — **Doc. Dr. Ing. JAN ČERNOCKÝ**
**VEDOUCÍ PRÁCE**



**BRNO 2020**

**Brno University of Technology**
Faculty of Information Technology

Department of Computer Graphics and Multimedia (DCGM) Academic year 2019/2020

# Bachelor's Thesis Specification


Student: **Bujnovský Michael**
Programme: Information Technology
Title: **Transmission of Digital Information over Audio**
Category: Signal Processing

Assignment:

1. Get acquainted with the principles of signal processing and digital communications (modulations, synchronization, error correcting codes).
2. Perform a survey of existing methods for transmission of information over audio.
3. Elaborate the algorithms for such transmission and test them on clean simulated signals.
4. Take into account the characteristics of a real audio channel (loudspeaker, microphone, noise, reverberation).
5. Implement a demonstration system allowing for a transfer of a file between two computers.
6. Perform and evaluate tests in different environments.
7. Create a poster and/or short video demonstrating your work.

Recommended literature:

- Simon Haykin: Digital Communications, Wiley 1988.
- according to supervisor's advice

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor: **Černocký Jan, doc. Dr. Ing.**
Head of Department: Černocký Jan, doc. Dr. Ing.
Beginning of work: November 1, 2019
Submission deadline: May 28, 2020
Approval date: November 1, 2019

# Abstract

The aim of this work is to transfer binary information between two devices just by usage of sound. The work starts with the analysis of existing solutions. Following by descriptions of different modulation techniques, synchronization and own protocol used in application. Significant part of the work is a test of transmission success rate in different setting of frequencies and environments. Based on results, I programmed user-friendly application showing system.



# Abstrakt

Cílem této práce je umožnit přenést binární informace mezi dvěma zařízeními jen za pomocí zvuku. Práce začíná analýzou existujících řešení. Dále popisuje různé techniky digitální modulace, věnuje se synchronizaci a vlastnímu protokolu zapouzdřující data. Velkou částí práce jsou testy úspěšnosti přenosu za užití různých nastavení frekvencí a prostředí. Na základě výsledků testů je implementovaná uživatelská aplikace, umožňující demonstraci systému.



# Keywords

audio qr, communication, sound, modulation, synchronization, microphone, QPSK, BER, digital



# Klíčová slova

audio qr, komunikace, zvuk, modulace, synchronizace, mikrofon, QPSK, BER, digitální


# Reference

BUJNOVSKÝ, Michael. *Transmission of Digital Information over Audio.* Brno, 2020. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Doc. Dr. Ing. Jan Černocký

# Rozšířený abstrakt

Cílem této práce je umožnit přenést binární informace mezi dvěma zařízeními jen za pomocí zvuku. Práce začíná analýzou existujících řešení. Pro samotný přenos digitální informace bylo nalezeno vícero projektů. V oblasti se angažuje Google, další společnosti vyvíjejí SDK a knihovy pro využití technologie např. při autentizaci. Zvlášť povedenou knihovnou je projekt Quit Modem, opensource dostupný na GitHubu. Má i rozšíření pro JavaScript. Původním plánem bylo tato řešení přehledně srovnat v tabulce s parametry jako spolehlivost, rychlost, použitá modulace. Vzhledem k uzavřenosti většiny projektů a velmi vágnímu nebo žádnému popisu těchto srovnatelných parametrů toto nebylo možné.

Kapitola Teorie uvádí do procesu modulace. Je zde popsaná modulace a demodulace BPSK, která využívá pro vyjádření 0 a 1 funkci signum. Dále je vysvětlena modulace QPSK, která slouží jako základ pro QAM (kvadraturně-amplitudová modulace). QPSK je popsána v komplexní rovině, kde jednotlivé složky I a Q vytvářejí komplexní exponenciálu do které je pak možné vkládat a číst symboly. Tyto symboly jsou přeloženy na digitální informaci. Pro optimalizaci vysílaného signálu byla použita dolní propust.

V podkapitole synchronizace je popsána implementace v knihovně 'AudioTransceiver' (která je výstupem této práce). 'AudioTransceiver' využívá symbolové synchronizace, kdy je na počátku sekvence dat zakódován předem známý symbol. Symbol je zvlášť vybrán tak, aby měl první část jednoduchou a teprve druhou část jedinečnou. K detekci dojde při splnění podmínky podobnosti a následné podobnosti po demodulaci počátečního symbolu. K symbolu je také přiřazena počáteční podpůrná vlna, která má za úkol zahřát (rozhýbat membránu) zařízení, které vysílá. K této optimalizaci došlo až po vyhodnocení testů.

Kapitola implementace mluví o architektuře 'AudioTransceiver' a o rozdělení kódu v projektu. Projekt se skládá z několika částí, tou úhlavní je jádro - knihovna. Ta definuje rozhraní (např. zvukového API), které je pak implementované v platformově specifických projektech. Knihovna obsahuje několik abstrakcí, části věnující se modulaci, synchronizaci, čtení a zapisování byte streamů, vizualizace, testování, vytváření komplexní exponenciály, byte-bit operací, vytváření vlastního protokolu pro přenos. Jako uživatelský program je použit 'Terminal' a jde o program pro platformu Windows.

Poslední kapitolou je testování. Knihovna byla napsána s ohledem na možnost měnit parametry frekvence, amplitudy, počet period na jeden symbol a rychlost. Pokud je nějaký parametr neznámý, je možné ho za použití těch známých dopočítat. Vždy bylo provedeno mnohonásobné měření a výsledky zaneseny do grafu. Testy se prováděly v těchto třech kategoriích.

1. frekvence
2. úhel
3. prostředí

Kategorii frekvence vykázala neočekávané výsledky, když došlo k úspěšnému přenosu u frekvencí vyšších než je polovina vzorkovací frekvence. Tento úkaz lze ale vysvětlit tím, že konečná frekvence byla kvůli nedostatku vzorků ve skutečnosti nižší, než co daná knihovní funkce parametricky počítala.

V přílohách je zaznamenán použitý software pro vytvoření této práce (a všeho okolo ní) a návod na použití uživatelského programu 'Terminal'.

# Transmission of Digital Information over Audio

## Declaration

Hereby I declare that this bachelor's thesis was prepared as an original author's work under the supervision of Mr. Doc. Dr. Ing. Jan Černocký, All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Michael Bujnovský
May 28, 2020


## Acknowledgements

I would like to thank my supervisor Doc. Dr. Ing. Jan Černocký for both expert insight into the issue and human support.

# Contents

# Chapter 1

# Introduction

Idea is to create an application, which would enable digital communication over sound. In a similar way as existing solutions, which use electromagnetic waves to transfer information. We can compare protocol to Morse code, in sense it is transferred over the air. It is implemented similarly as existing electromagnetic systems. The chapter Existing solutions shows how was the field of audio transmission processed by different people and companies. Theoretical part describes principles used by those systems. Implementation describes architecture of created program. Testing chapter is about trying the application under different conditions and evaluating results.

# Chapter 2

# Existing solutions

### 2.0.1 Audio Watermarking

Watermarking is about embedding additional information to the signal which is supposed to be perceived by with noise. Typical usage is to mark copyright material. Advanced versions of watermarking are trying to hide identification to not interfere with how humans are receiving the signal and both staying resistant to noise or different manipulation with the original signal. The main difference against the subject of this thesis is being static. The watermarked piece is final, it lacks the dynamic of live communication, there is much more robustness which takes space to speed. Still, the knowledge can be used in parts, where needed as pairing (hello process) or so.

### 2.0.2 Transferring non-digital data over audio

#### Morse code

Morse code comprise the alphabet where each character is represented by short and long (in this case sound) loud intervals split by short intervals of silence. Using Morse code each character is split by a longer interval of silence. In a sense of modulation the code use amplitude (sound or silence) modulation and phase (time intervals) modulation, however it is not really defined which modulation should the code use. Morse code as usually understand as ineffective, slow way of communication, a limit is in encoder and coder which was human.

#### Spoken language

Speech is the standard way of human to human communication over the sound. From a view of Information technology it is slow, but effective in a certain way. Beside actual words it contains also 'meta' information which speak about emotions of transmitter, about their intensity and other like age, gender, nationality, personal identification. It is a complex system and can't be compared to technical meaning of communication.

### 2.0.3 Existing solutions for transmitting digital information over audio

It is surprising how many programs do exist in this area. There is no standard, no universal naming for transferring bits over audio which makes it difficult to find such. At first attempt it was possible to find few commercial libraries or SDK. Later during implementation process, after accumulating new keywords, more (even open-source) were discovered.

Original idea was to take those solutions and compare individual parameters (such a speed, reliability) in table form. That was not carried out as most of them were either not publicly accessible or closed to testing or just working on incomparable principles and achieving different goals. Even incorporating reverse-engineering and rewriting libraries to be testable in communal manner would still leave blank cells in a table with results. Because the table is not a goal of the theses, it is presented a simple list with short description and highlights of the individual project.

### Google

Google has already at least 3 solutions in the field. 1st is Google Tone[5], it is an extension for Chrome enabling to share URL of open webpage by using dedicated audio signal. Shared is just the token which is translated into full URL after transmission.

Another one is Google Nearby[3]. It's API for communication based on proximity and peer-to-peer networking. One of the method of initiating connection is based of ultrasonic messages.

3rd is Google Tez App[4]. Its application for Android dedicated to Indian market. It enables a user to pay and receive money using UPI (local Indian unified interface for transferring money) by using ultrasound.

### Signal360[8]

Company that has patented inaudible-sound mobile notification technologies [13] and along with other solutions based on BLE (Bluetooth low energy) provides it to customers (marketing field, retail).

### Chirp[1]

'Sonic barcode' uses either audible or inaudible tones. Available SDK.

### Prontoly (Sonarax)[10]

Cross-OS SDK enabling authentication, data transfer. It offers demo-app with sending emoticons via ultrasound.

### LISNR[6]

Ultrasonic data transfer start-up. Speed up to 1kbps. Core technology is written in C library.

### ToneTag[11]

Audible authentication solution. It doesn't offer SDK on their website.

### CopSonic[2]

Dedicated solution with both audible and inaudible frequencies. Advertised is speed up to 15kbps, detection under 300ms and Full-duplex.

**pied-piper**[7]

Speed of 80 bps. It was built during the VeloCity Residence Hacknight for Spring 2015. Available as open-source.

**SinVoice**[9]

Chinese commercial project, uses a variety of frequencies with an option of superimposing into user-defined sounds. It has 98% recognition rate and 8 metres distance.

**Quiet Modem Project**[12]

Comprehensive open-source library with lots of filters, modulation methods, different wrappers (for JS, TCP/IP layer extension). It is written in C and it contains also JavaScript wrapper for usage on the Web.

**AudioNetwork**[18]

Poland hobby project with goal to build simple network that uses sound waves to transmit data. Open-source implementation using JavaScript.

# Chapter 3

# Theory

## 3.1 Modulation

"Modulation is a process by means of which one or more parameters of a sinusoidal carrier are varied in accordance with a message signal so as to facilitate transmission of that signal over a communication channel." [Haykin (2014), p.59][14]

In a meaning of this thesis it is a process of encoding binary data into the form of a sound wave. It is a core process of this project.

We can divide modulation methods into two basic categories: baseband and passband. In passband modulations we change a parameter of high-frequency signal. It may be amplitude (Amplitude Modulation), frequency (Frequency Modulation) or phase (Phase Modulation). It may be over one parameter.

Baseband modulations uses none high-frequency wave. They use only base bandwidth (therefore baseband). Similarly, as in passband modulations, different parameters of signal are changed, but without missing carrier the result are pulses. It may be again amplitude - PAM (Pulse Amplitude Modulation), frequency - PFM (Pulse Frequency modulation) or shift from a nominal value - PPM (Pulse Position modulation).

Another way how to divide modulation types are analog and digital. Prerequisite for analog modulation is access to analog parts of device, which is not the case here. Therefore, the focus is on digital modulation.

### 3.1.1 2PSK (BPSK)

#### Modulation of 2PSK

Binary Phase Shift Keying is possibly the simplest modulation in digital passband category. Let's suppose there are Bipolar NRZ (No Return to zero) code which represents bits sequence. Bipolar NRZ is basic link code for transmitting binary signal.

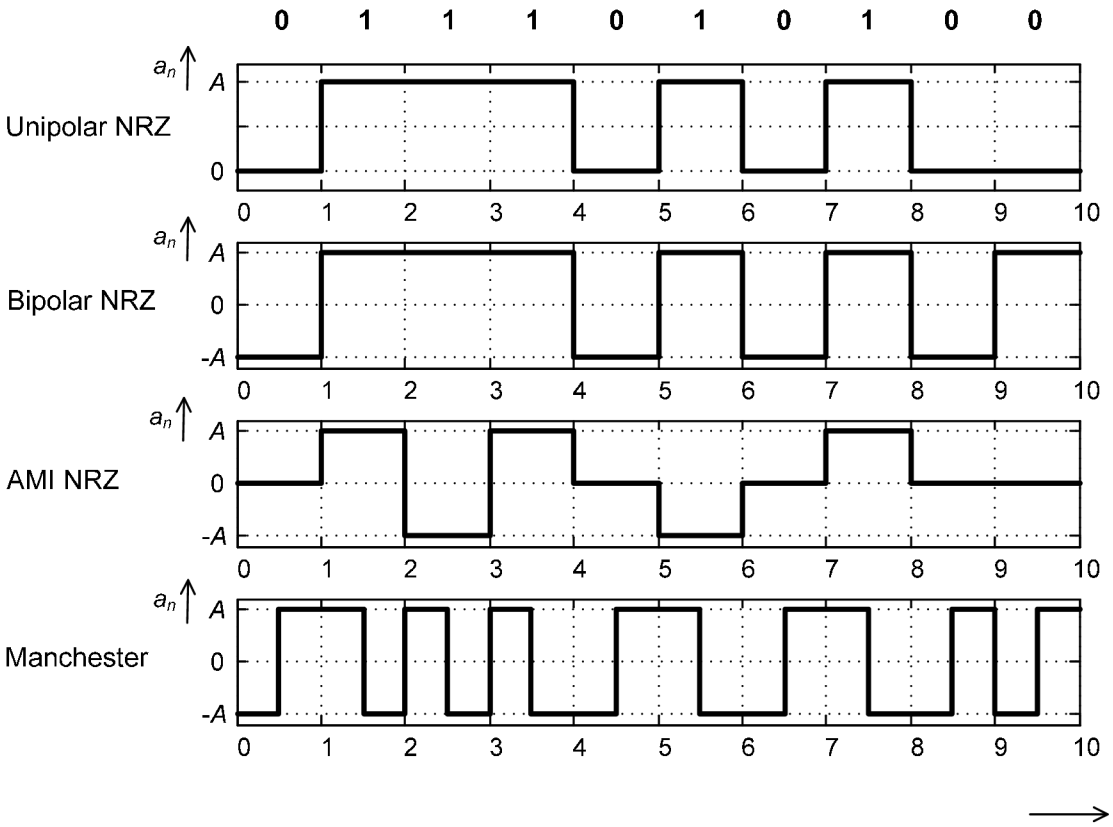


**Figure 3.1:** Basic link codes. From [16].

Amplitude of the signal is represented by

$$A_n = \{-A, A\},$$

where

$$-A$$

is constant value of amplitude representation of bit '0' and

$$A$$

is constant value of amplitude representation of bit '1'.

And full BPSK modulation is achieved by multiplication with the Carrier:

$$S_{BPSK}(t) = A_c \cos(\omega_c t)$$

for bit '1',

$$S_{BPSK}(t) = -A_c \cos(\omega_c t) = A_c \cos(\omega_c t \pm \pi)$$

for bit '0'.

Changing the sign before $A$ can be also achieved by shifting phase by $\pm\pi$.

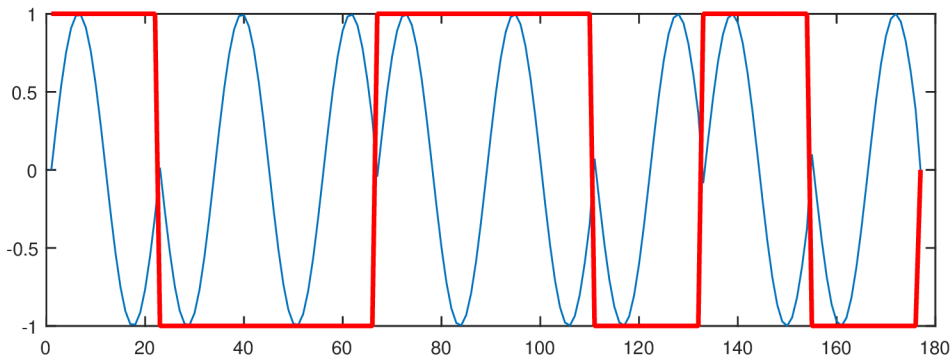


**Figure 3.2:** Experimental modulating of sequence '1,0,0,1,1,0,1,0'. Red-colored line is NRZ. Blue is modulated form.

### Demodulation of 2PSK

For demodulation it is necessary to get use synchronized Carrier. It is done using synchronization process section 3.3. Synchronization is determining $\phi$ shift of the Carrier. After multiplication of modulated data with the Carrier and integration, we can gain binary data.

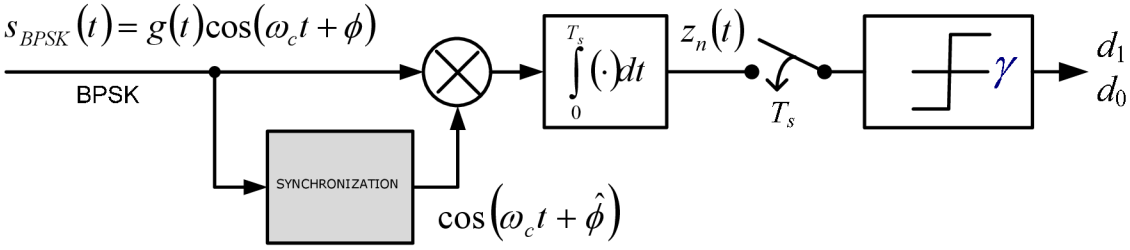


**Figure 3.3:** 2PSK demodulator scheme. Edited from [16].

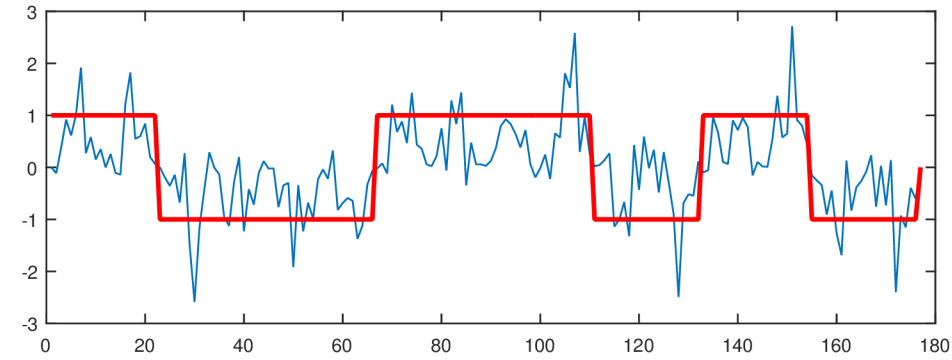


**Figure 3.4:** Experimental demodulating of sequence '1,0,0,1,1,0,1,0'. Red-colored line is original NRZ. Blue is demodulated form (before integration). White noise of $-1$ dB is added to simulate reality.

### 3.1.2 4PSK (QPSK) or also 4QAM

Quadrature Phase Shift Keying is a logical extension of BPSK, where a number of shifts is 4. Instead of shifting signal by $\{0, \pi\}$ as in BPSK, modulated signal is shifted by $\{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$.

Each state is represented by dibits (two bits). It is possible to extend number of those state to $m$ and get m-PSK modulation (e.g. 8PSK).

### Signal-Space Diagram

Another form of QPSK can be obtained by using IQ (In-Phase and Quadrature) components. For following values $\{\frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4}\}$ we may define QPSK signal as

$$s_i(t) = \left\{ \sqrt{2E} \cos\left[\, 2\pi f_c t + (2i-1)\frac{\pi}{4}\right], \right\{ i = 1, 2, 3, 4$$

where $E$ is signal energy and $f_c$ is Carrier frequency. We may redefine it using trigonometry in canonical form:

$$s_i(t) = \sqrt{2E} \cos\left[(2i-1)\frac{\pi}{4}\right] \cos(2\pi f_c t) - \sqrt{2E} \sin\left[(2i-1)\frac{\pi}{4}\right] \sin(2\pi f_c t), \left\{ i = 1, 2, 3, 4$$

Now we can observe two orthonormal basis functions, defined by a pair of quadrature carriers:

$$\phi_1(t) = \sqrt{2} \cos(2\pi f_c t)$$
$$\phi_2(t) = \sqrt{2} \sin(2\pi f_c t)$$

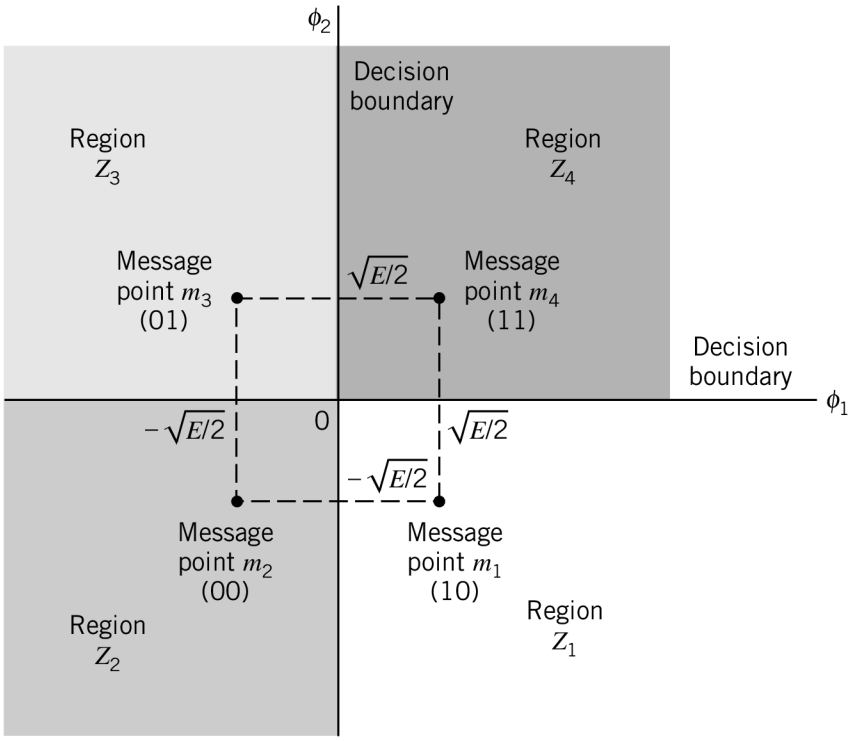


**Figure 3.5:** Signal-space diagram of QPSK system. During demodulation is an average or mean point taken from regions $Z_x$ and translated into bits based on Message point. Diagram from [14].

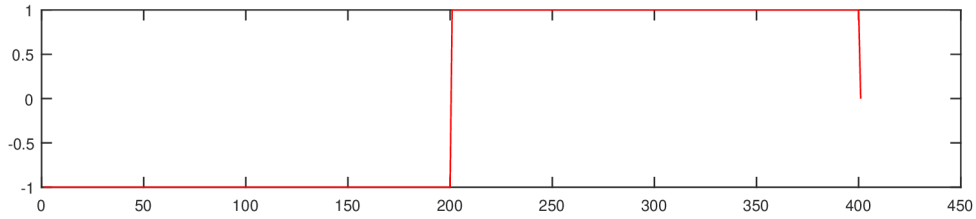


**Figure 3.6:** NRZ for $I(t)$ component.

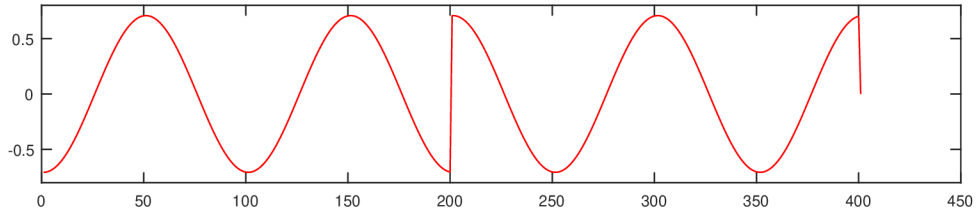


**Figure 3.7:** Modulated $I(t)$ component.

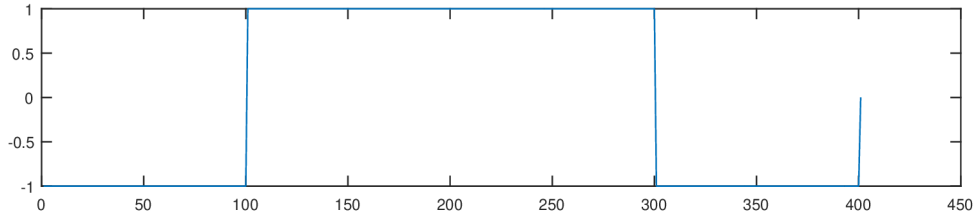


**Figure 3.8:** NRZ for $Q(t)$ component.

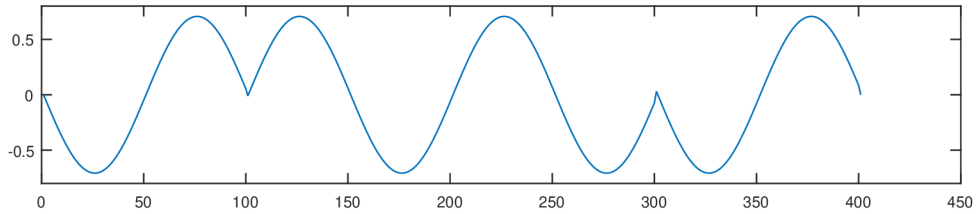


**Figure 3.9:** Modulated $Q(t)$ component.

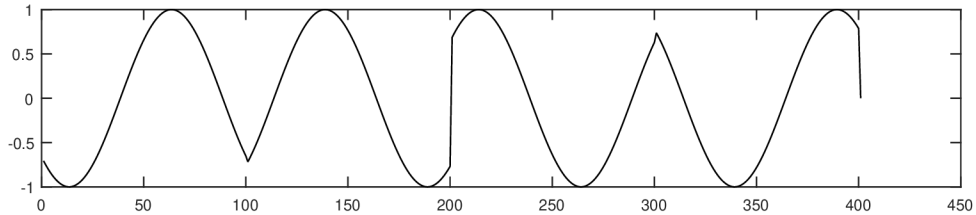


**Figure 3.10:** $Q(t) + I(t)$: final product.

**Figure 3.11:** Assembly process of QPSK modulation for sequence '0,0,0,1,1,1,1,0'.

### Demodulation

After synchronizing (see section 3.3), a Carrier generator is properly set and demodulation process started. The both components ($I(t)$ and $Q(t)$) are multiplicated with corresponding shifted Carrier. Then the value is computed based on mean from all the samples from a section of one symbol. Finally, the bit sequence is defined.

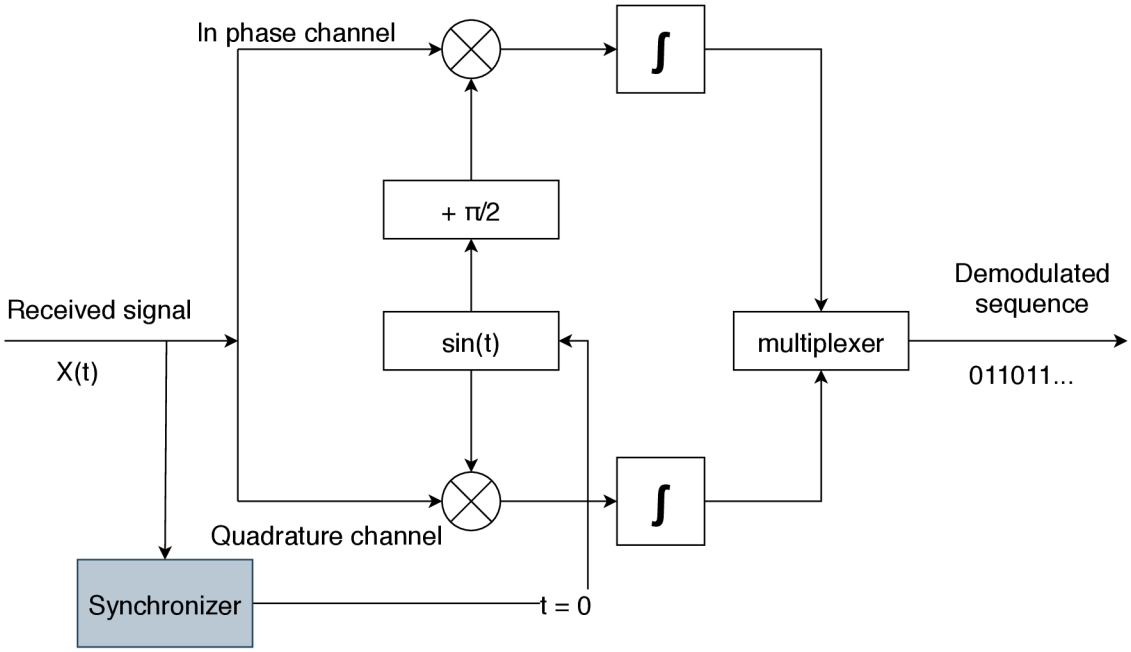


**Figure 3.12:** Demodulation diagram with $I(t)$ and $Q(t)$ components.

## 3.2 Anti-alias (low-pass) filter

During modulation are created sharp transitions among individual sections with same modulated phase. To avoid or rather reduce the negative impact of the high pitch transition, we can use a low-pass filter. The idea is to remove a section of a sudden change of phase to improve error-resistance of demodulation.

The simplest solution to apply a low-pass is to use Moving average filter. Moving average is averaging window of M samples of input signal. Equation:

$$y[i] = \frac{1}{M} \sum_{j=0}^{M-1} x[i+j]$$

**Figure 3.13:** $y(i)$ is an output signal, $x(i)$ is an input signal and M is a number of points (samples) used in the Moving average. [19]

It is crucial to keep $M$ parameter low enough not to disturb modulated signal. Another problem is that Moving average is lowering the energy of the signal.

It is a negligible optimization, especially in higher frequencies, where one sinus period may contain two samples.

**Figure 3.14:** Same signal with applied Moving average (black) and non-filtered (blue).

## 3.3 Synchronization

To recognize the beginning of the message, there is a need for a system to bring synchronization into place. Even a small inaccuracy in determined synchronization can produce a significant error in later processing. For now, lets suppose that frequency is known. What remains is to find out the exact moment when the symbol translated and modulated in a wave signal comes up on the side of the receiver.

### 3.3.1 Radio vs sound

Radio systems used for this purpose Carrier Recovery circuit. That is however relatively expensive on a number of periods of Carrier. Here comes the great difference between radio and sound. The radio uses frequencies between 2KHz to 300 GHz, typically for high-speed wireless communication, it is from 1GHz to 60 GHz (802.11 standards used in WiFi, Bluetooth). Sound (typically integrated sound systems) use range from 20 Hz to 20 kHz. Lets suppose for synchronization we need 1000 of periods. 1 GHz, translate that in 0.000 001 seconds. 1 kHz translate that in 1 second. And 1 second is already a considerably great amount of time. Another problem is that radios are getting data in real-time with real-time data processing instead of getting data in chunks from a sound card.

### 3.3.2 Symbol synchronization

Small frequency and data speed rate of sound offer the possibility of capturing it and processing in real time. Computer systems nowadays have enough power and memory to simulate or use different ways of solving the synchronization problem. The method used consists in continuous searching for synchronization symbol (that is a sequence of bytes encoded in sound). For further mentions, we call the synchronization symbol used in this solution 'preamble'. To maximize the chance of detection while minimizing false detection, the preamble is split into two parts:

- Part 1: Section of continuous clean signal (with no phase shifts). The section is longer to deny false positives.
- Part 2: Section with phase shifts distinguishing preamble. The section is shorter to minimize the risk of a missing preamble.

Algorithm of detection works in the following order:

1. Encoding and modulation of bytes which represent preamble into samples '0' and '1' (depending on sample energy).
2. Comparing individual samples to match preamble representation.

3. After similarity with preamble representation is greater than the threshold value (which is around 80%) it creates the match.
4. Try to find a better match in close samples trying one by one (to the maximal distance of the length of one modulated symbol from the first match).
5. Check if best match decoded into bytes is equal to preamble bytes. Likelihood of bits must be greater than a threshold (around 96%).
6. Return best match from close samples as a possible detected preamble.

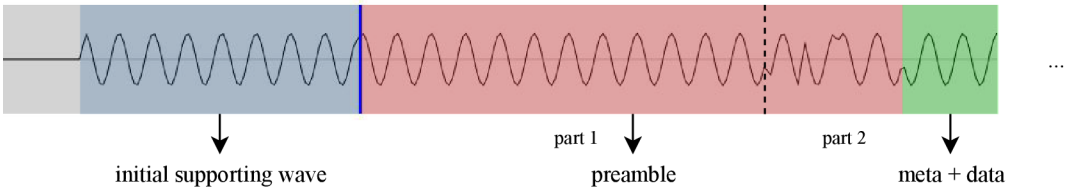


**Figure 3.15:** Preamble visualized

### Initial supportive wave

In the real world transmitting signal does not instantly achieve full energy but after the short build up. That is hurtful for a beginning sequence of data - preamble. Adding initial supportive wave - which is prolonging part 1 of preamble on transmitting side noticeably helps to improve synchronization.

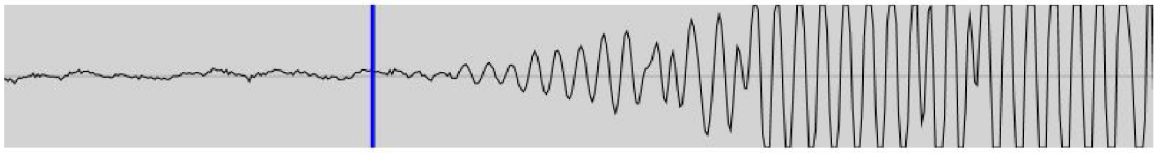

**Figure 3.16:** Preamble without initial supportive wave

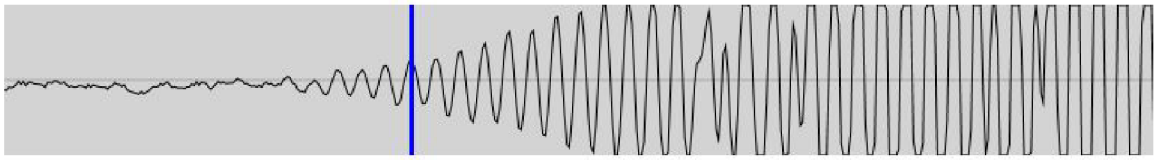

**Figure 3.17:** Preamble with initial supportive wave

Figures 3.16 and 3.17 shows real received signal in both scenarios. Vertical line signs start of preamble.

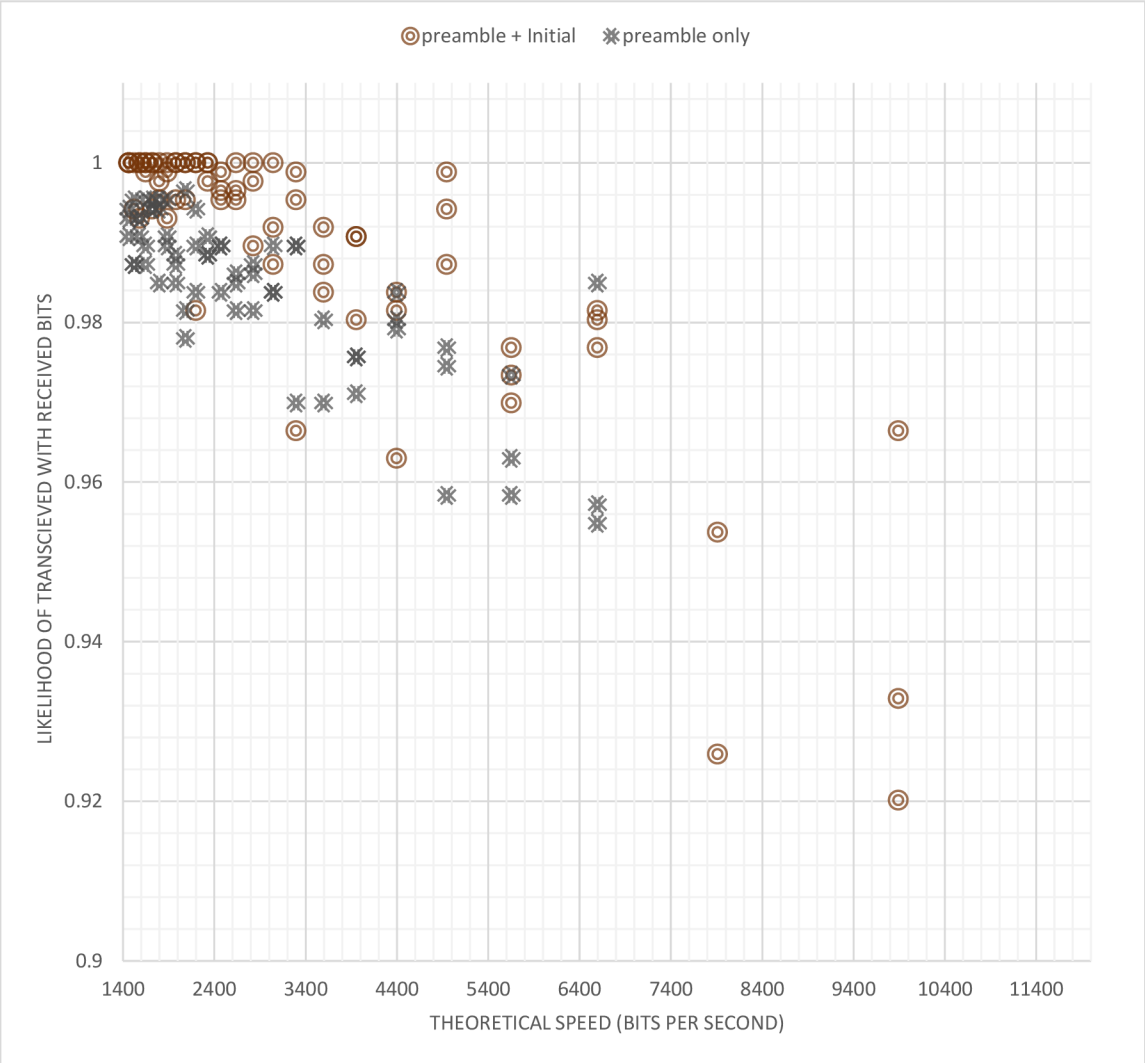


**Figure 3.18:** Initial supportive wave

Testing showed a clear advantage on an option with an initial supportive wave. The preamble is never 'correctly' matched with 'preamble only' option. Tested on 9890 Hz with device distance of 1 m. See more about testing in chapter 5

# Chapter 4

# Implementation

Library with name 'AudioTransceiver' was implemented in a programming language C#. It is implemented as a Portable library with .NETStandard. From the code architecture perspective it contains three parts (or projects - according to Visual Studio naming conventions [1]):

- Portable core
- WPF testing UI
- Windows sound API Implementation

### 4.0.1 Core library

It is a core part of the program. It contains everything expect platform specific tasks.

`class ATManager` is the highest class in library hierarchy. There are methods to run continuous receiving (`ContinuousReceiving`) and sending messages created from bytes (`SendMessage`). The class is putting everything from modulations to likelihood functions together and making it work. It reflects and binds UI functions.

Other parts of code are divided into folders:

#### Basic

It contains functions that handle operations as converting (bool array into a byte array, int into a byte array), extracting floats from byte Wav files and low-pass filter.

#### Interfaces

Folder with interfaces which is an abstract requirement for class. There are two interfaces one for work with files and second for work with sound API (such as getting data from a mic or playing them on speaker). Those interfaces are implemented in platform specific projects.

#### Modulation

Classes `Modulator` and `Demodulator`, inheriting from `ModulationBase`, are implementation modulation and demodulation circuit (see e.g. Figure 3.12).

[1] In this chapter the keyword 'project' is only used as Visual Studio project.

#### Synchronization

It contains algorithm for preamble detection as described in subsection 3.3.2.

#### Testing

Class `StatReport` as described in section 5.2. Class `ATTests` contains method to run single test (`TestRealLoopback`) or test session (`TestMultiple`).

#### Transceiver

Transceiver takes responsibility of creating frames from raw byte data.
Class `TransceiverProcessor` is folding and unfolding those raw data into the frames together with meta as size or identifier. It also justifies how many Frames to create depending on `MAX_FRAME_LENGTH`. The whole message sent by the program have following structure:

```
Preamble|FrameStartFlag|FrameId|FrameDataSize|... data ...
|FrameStartFlag|... data ...|FrameStartFlag|...|FrameStartFlag|
...|Postamble
```

#### Visuals

Those are classes that hold and bind values and data for Visual presentation. There are methods for rendering received signal and zooming implementation. Some methods are simplifying render method and optimizing its performance.

#### Wave

Useful structures and methods for generating Carrier.

### 4.0.2 WPF testing UI

MVVM (Model-View-ViewModel design pattern)[17] implementation, where Model is subsection 4.0.1. ViewModel and View are contained in the project. Folder UserControls contains components for rendering complex user controls. Table of binary/byte values is generated by `BinaryValues`. `SamplesControl` is used to generate received signal as is. And `SamplesRowControl` handles original signal overlay with custom defined preamble position and zooming function. It also generate blue preambles.

### 4.0.3 Windows sound API Implementation

Connecting to the microphone and speaker is difficult using raw Windows APIs. Therefore, it is used open-source library NAudio[15]. It offers not only simplified and wrapped access to Windows sound APIs but also ASIO - computer sound card driver protocol which might be interesting a in future development. Plus, it offers different conversions of .wav files. The project implements interface `ISoundApi` and it is further used in the core library.

# Chapter 5

# Tests

## 5.1 Introduction

There are two kinds of tests that were performed. One is focused on frequencies and ability of computer systems to obtain and push various waves of different frequencies throughout the air. The other type of test has frequencies already predefined and it is about testing speed of transmission in different environments.

## 5.2 Testing flow

Testing was from the beginning part of the program. To test different threshold and optimizations, the section of signal visualization was added.

For the report purposes is every test saved (after test execution) with following parameters:

```
public class StatReport
{
public int Frequency;
public double Amplitude;
public int SamplingRate;
public double PeriodsPer1Symbol;
public double FilteredSamplesPer1Period;
public bool Synchronized;
public double Success;
public int DataSizeInBytes;
public int TheoreticalSpeed;
}
```

`int Frequency` is frequency of the Carrier and so also of the signal. Default value is `4410`.

`double Amplitude` is amplitude of the Carrier and the signal. Default value is `0.5f`.

`int SamplingRate` defines a number of samples in 1 second. Default value is `44100`.

`double PeriodsPer1Symbol` defines a number of periods of Sinus signal that contain 1 QPSK symbol. Default value is `3`.

`double FilteredSamplesPer1Period` is a low-pass filter parameter. It is a ratio of filter length to period length. Default value is `0.15`.

`bool Synchronized` is indicator whether testing achieved successful synchronization (finding of correct Preamble ).

`double Success` is a likelihood of received bits with transmitted. If it was not successfully synchronized, then it is `0`.

`int DataSizeInBytes` is size of a data part of transmission (without meta). In tests we use value `100`.

`int TheoreticalSpeed` is theoretical speed of transmission, meta included in bps (bit per second). For the QPSK it is counted as:

```
TheoreticalSpeed => Frequency / PeriodsPer1Symbol * 2;
```

Program was oriented to always run multiple tests. Let's call running multiple tests session. There are 2 testing algorithms.

### Method 1

Method 1 was oriented to be faster and avoid multiple testings.

1. For each session are chosen frequencies.
2. Each frequency is tested starting with highest speed setting. That means that `PeriodsPer1Symbol` is set to the lowest value.
3. `PeriodsPer1Symbol` is rising each testing. If the program cannot synchronize on current settings, then it is rising faster otherwise in normal pace [1].
4. Tests are performed for every setting at least 3 times.
5. Each test gets random generated 100 bytes data to transfer. In reality, it is transferred more, because of meta.
6. After achieving 100% success for every test in same setting or operating too slow[2], next frequency is tested.

[1] Normal pace is around `1.5`, depending on frequency.
[2] Speeds lower than 1500bps.

### Method 2

Method 2 was designed to get better and more precise overview with slower execution.

1. For each session is chosen range of frequencies and speeds for testing and number on testing.
2. Each testing is performed with random generated speed and frequency. `PeriodsPer1Symbol` is computed using a reverse formula for speed.
3. Each test gets random generated 100 bytes data to transfer. In reality, it is transferred more, because of meta.

### Charts

Motivation is to get high speed while maintaining low bit error ratio (BER). Charts created from the measurements have therefore an axis of following choice:

- Axix y = 1 - BER - likelihood of sent and accepted bits.
- Axis x = Theoretical speed in bits per second. Raw bits without without any synchronization or protocol related data.

Let's look at what kind of chart will look like:

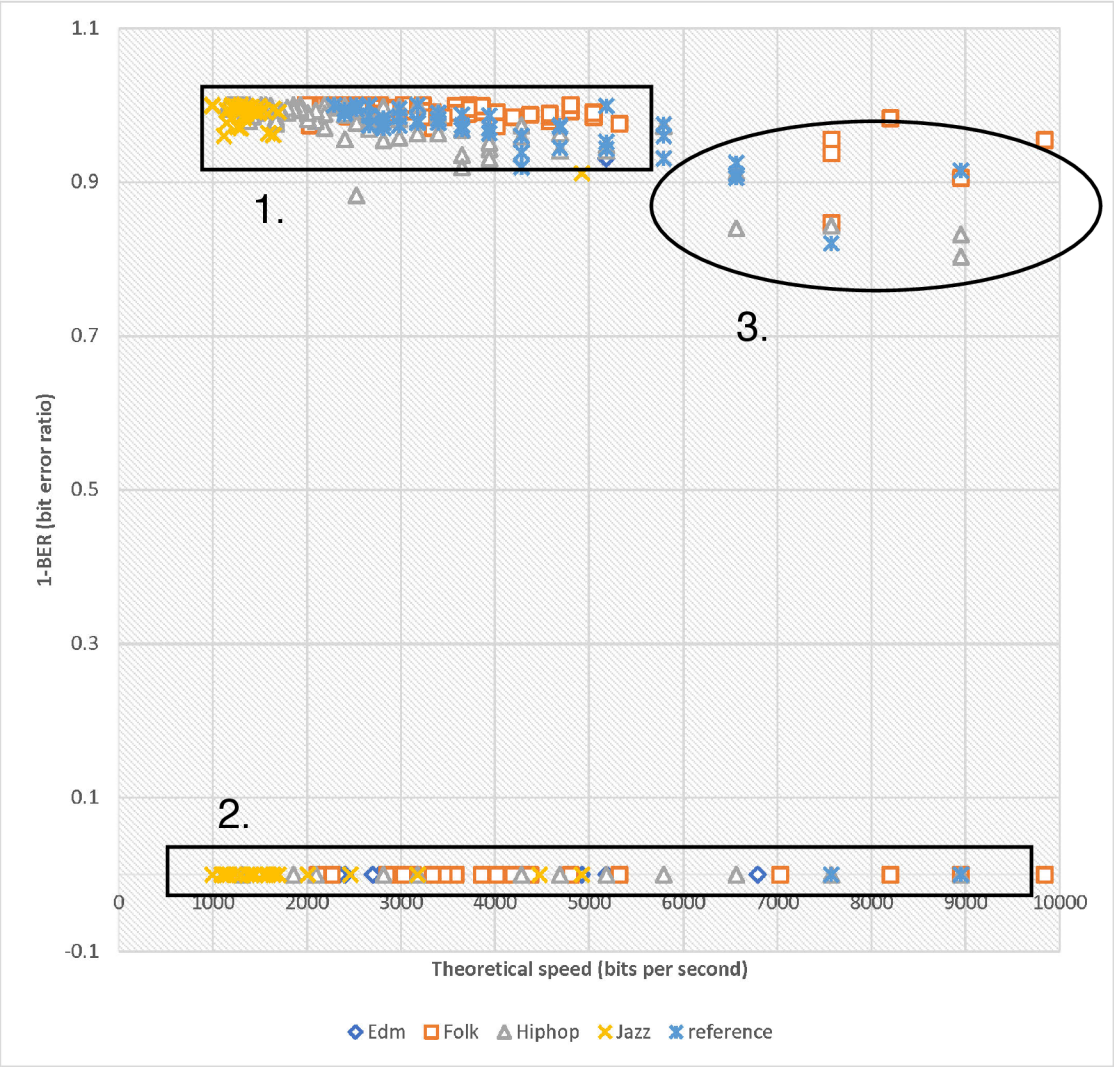


**Figure 5.1:** Full-size chart.

We can see 3 main areas in the chart.

1. Area 1 is the most relevant. It shows values that were synchronized on a solid BER.
2. Area 2 might contain a lot of points. Lets remove them as they represent synchronization misses and does not carry any additional value. [3]
3. Area 3 contain few individual points, lets exclude them to get a better image.

Only upper-left sector of the chart is used. Synchronization fails are typical when the program cannot recognize Preamble, but sometimes it is because of audio API related error. Let's therefore concluded that those are not relevant.

[3] It should be noted, that although majority of them are fails on side of synchronization, sometimes it is a Windows sound API who does not load and behaves unpredictably.

#### Low frequencies excluded

Lower frequencies (<1000 Hz) are significantly slower and does not offer performance of higher ones. Therefore, were excluded from most of the testing.

#### Testing on loopback

Although final program is supposed to run on two devices, to speed up and simplify testing only one was used. As the response is about 150ms, it can be safely stated that there are no physical related issues with audio card.

#### Stereo vs Mono

Program generates mono (1 channel) sound. To avoid two sources of transmission only one speaker was connected or only one was enabled in Operation system setting.

## 5.3 Testing under different circumstances

### 5.3.1 Frequencies

Frequencies to test were chosen as a geometric progression.

$$\sum_{i=0}^{f<25\,000\,\text{Hz}} 50 * 1.2^k$$

After removing first few frequencies which are not possible to run on testing devices, the sequence follows from $124, 149, 179, \ldots$ up to 24 611 Hz. Even though the highest frequency exceeds the Nyquist–Shannon sampling theorem the transmission is still possible. Such a mathematically generated frequency does not have sufficient number of samples and therefore in reality it results into a lower frequency. Also, the individual symbols are distributed over many periods and therefore the demodulation can overcome insufficient number of samples by approximation.

Method 2 is ideal for finding best frequency based on speed and BER. Frequency range was chosen from 1000 Hz to 25 000 Hz and speed from 1800 bps to 15000 bps. All the sessions were performed with speaker 0.5 m from a microphone. Each session was lasted 300 testing. Results from each session were quantified into groups and inserted into a 6x6 Matrix. Each point of the Matrix is average `double Success` of every test in the quantum including zeros if not synchronized. See Figure 5.6.

Let's try higher resolution by combining all 4 sessions into a matrix of 12x12 (Figure 5.7). It is still quite vague which frequencies should be chosen into possible production. Let's use logarithmic scale and use more tests. It is possible to target even frequencies which are impossible to generate with 44100 sampling rate. If such a frequency is generated in the real environment it is produced as lower frequency (< 22 050 Hz) due to insufficient number of samples. 42x42 matrix (Figure 5.8) is showing that area around 5000 Hz has the most reliable transmission score.

It can be concluded that lower speed means better BER score. And it is not a case of Frequency - higher frequency does not automatically mean higher BER score. It is interesting that some frequencies appear to be incompatible with being used in transmission.

Those are frequencies seems to have value 22 151 Hz, 14 967 Hz and 11 375 Hz. As those are all random measurements it is not one exact frequency, but rather in frequency in the near neighborhood. Those 'deaf' frequencies are suspiciously close to factor number of the sampling rate of 44100.

$$44100/2 = 22500$$

$$44100/3 = 14700$$

$$44100/4 = 11025$$

Which are exactly frequencies that have lowest BER scores. The highest BER score reaching frequencies are in between those values.

Another influence is frequency characteristics of testing device. Device not capable and designed to play or capture high frequencies (>12 000 Hz) will not have high score in that area.

### 5.3.2 Angle

It is typical that speakers are directional. That means that it depends on the angle they are facing the capture device (microphone). Figure 5.9 shows results of transmission on different angles. Tilting more than 45° is undergoing significant degradation of the signal.

### 5.3.3 Environment

If we want an application to be successful and without bugs, we must test it in multiple environments. In get neutral and identical setting, external speakers with no connection to the test devices are set up. On a same volume are run recordings of the different environments. During this simulation test sessions are activated. Reference is testing with silence in the background.

We could use audio QR on different music occasions, lets test it with different music genres. It can be stated that current system cannot be used in bass heavy and loud environments.

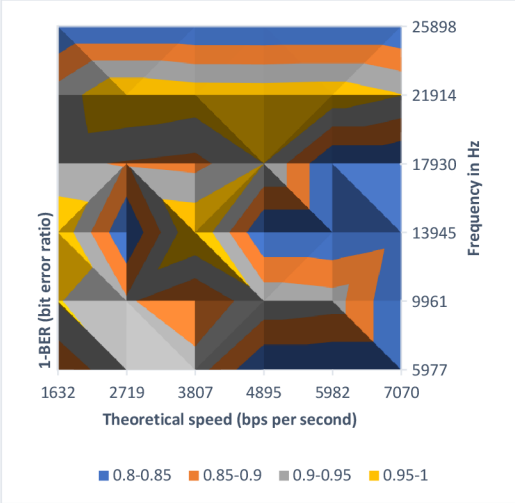


**Figure 5.2:** Session 1.

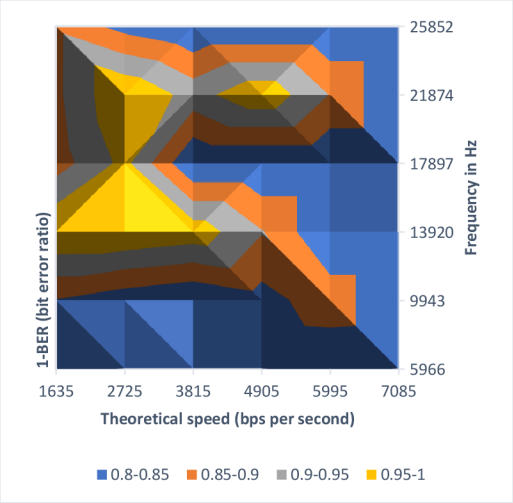


**Figure 5.3:** Session 2.

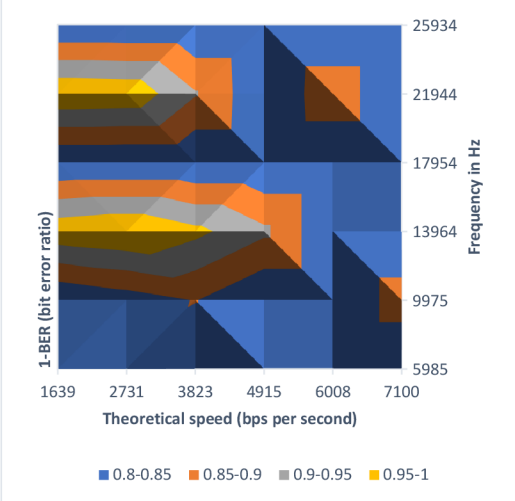


**Figure 5.4:** Session 3.

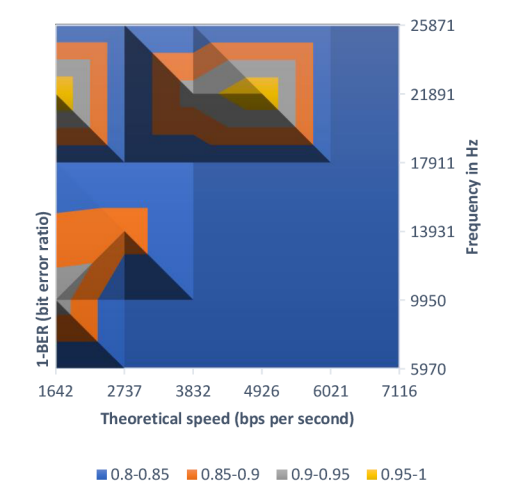


**Figure 5.5:** Session 4.

**Figure 5.6:** Each image is a quantification of the 300 test results. Pattern is visible even through small resolution.

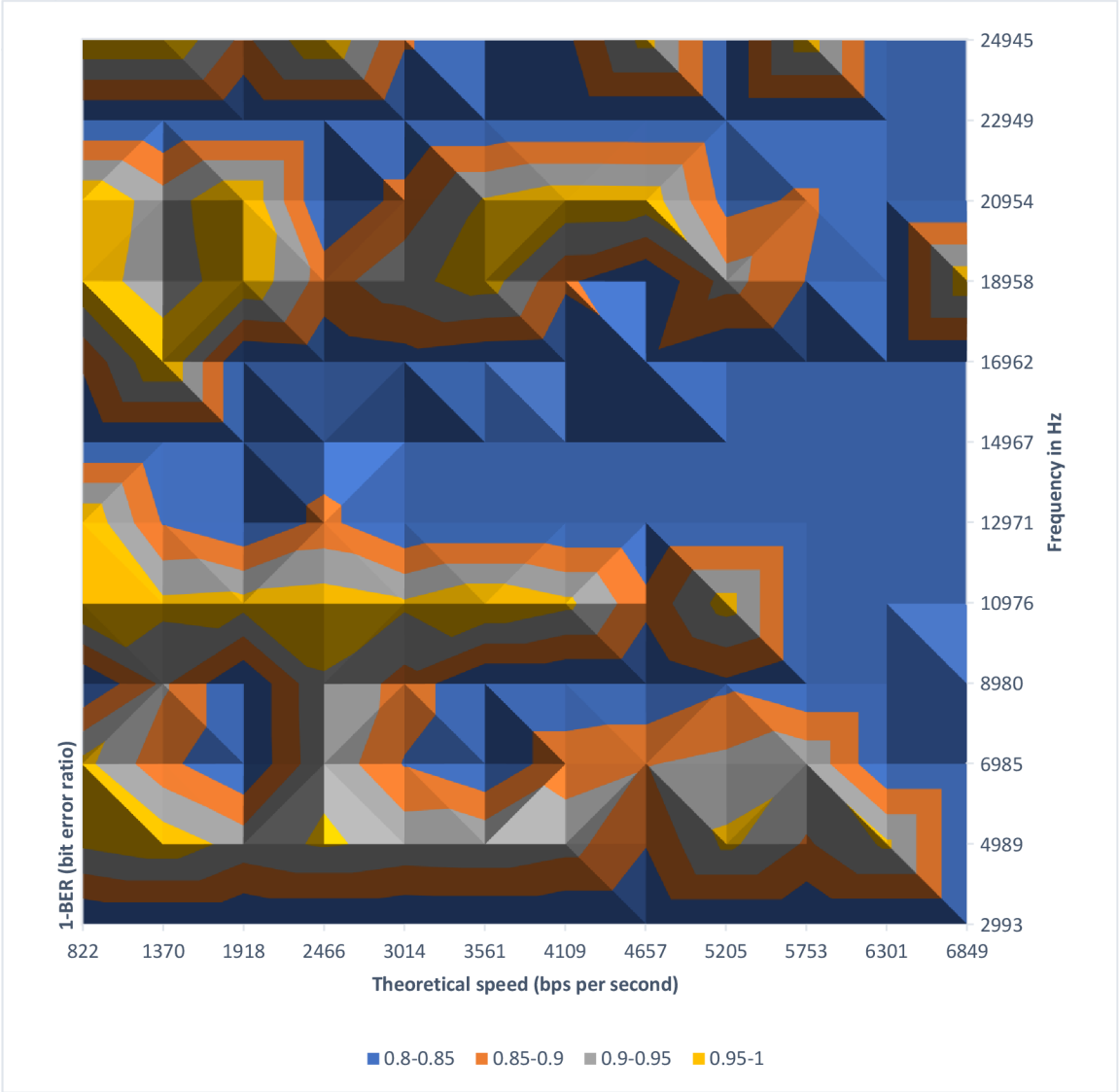


**Figure 5.7:** 12x12 quantification of 1200 tests.

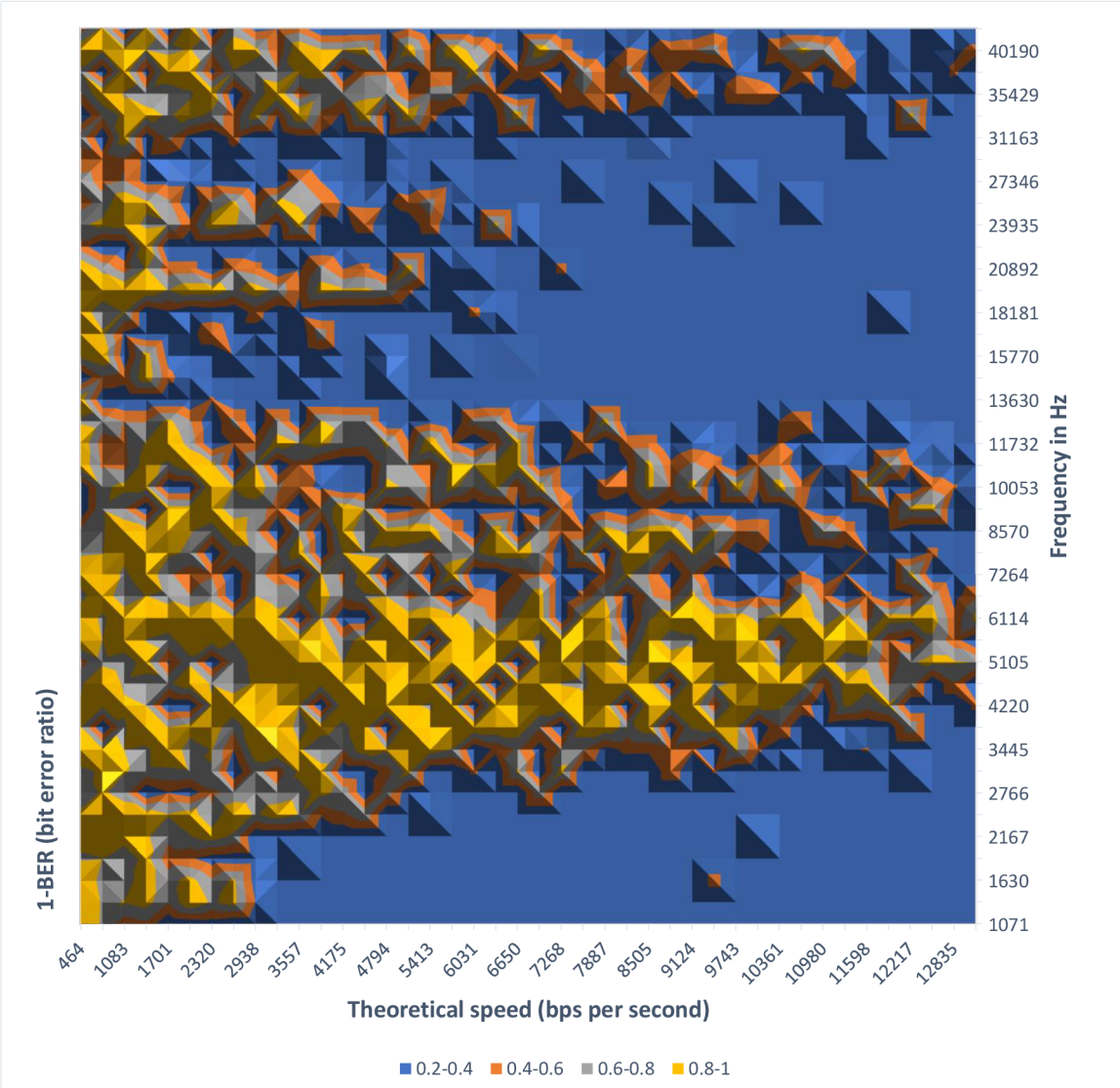


**Figure 5.8:** 42x42 quantification of 5200 tests from different test session. Frequency is in logarithmic scale.

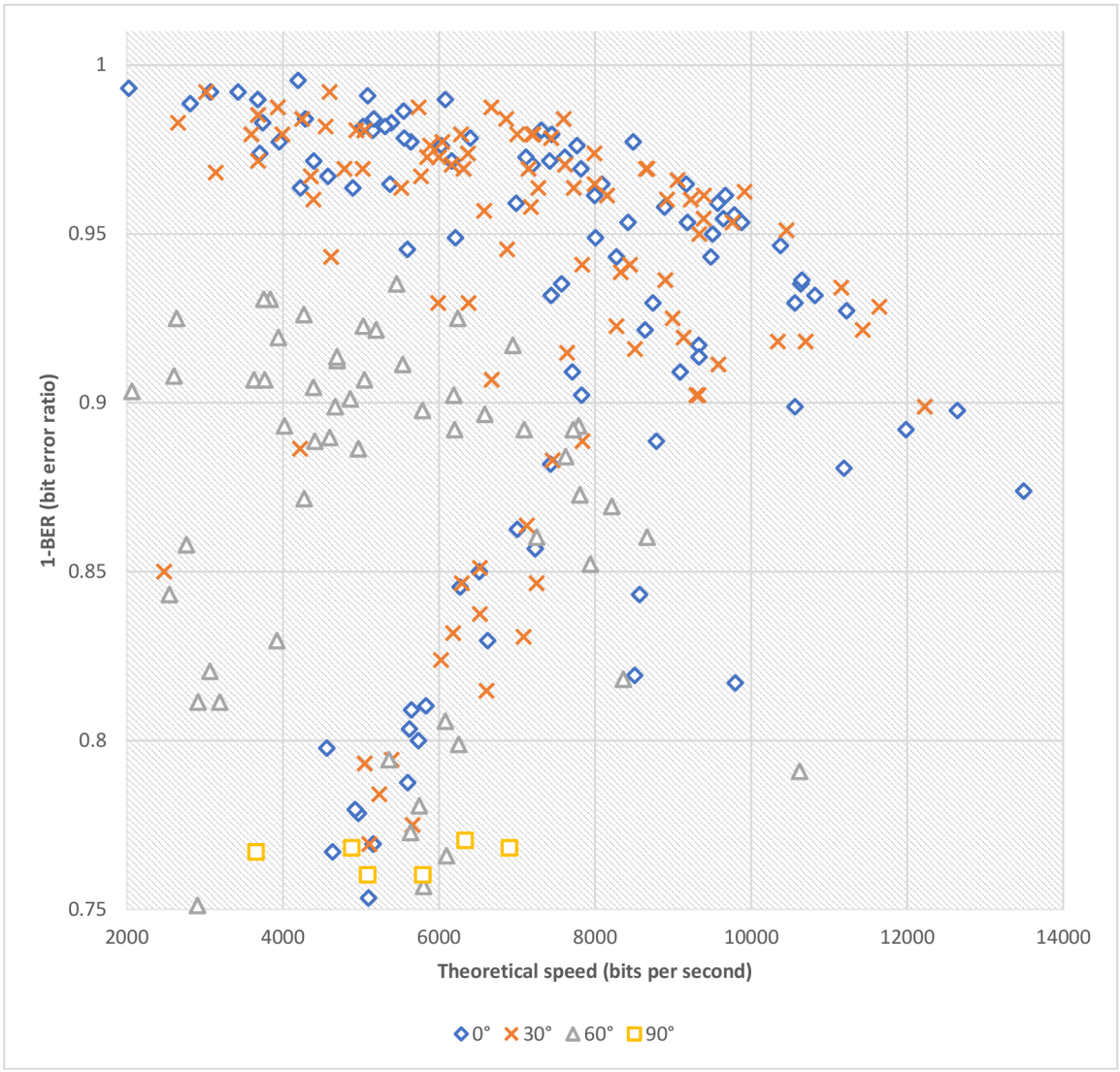


**Figure 5.9:** Test with different angles of the speaker facing the microphone. Distance between devices was 75 cm. 0° is the speaker directly facing the microphone.

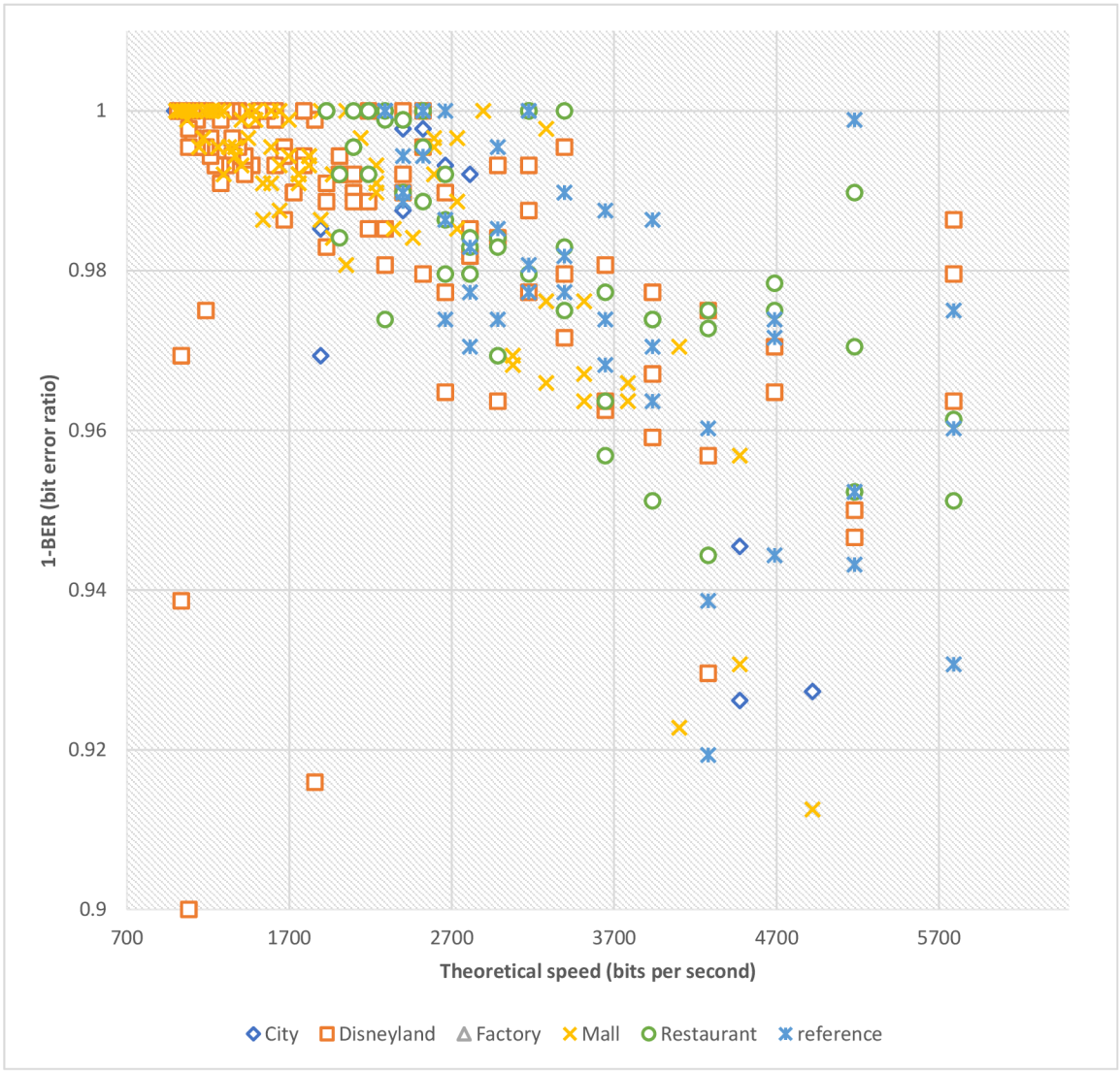


**Figure 5.10:** Tests performed in different environments with method 1. Frequency was set to 24 611 Hz.

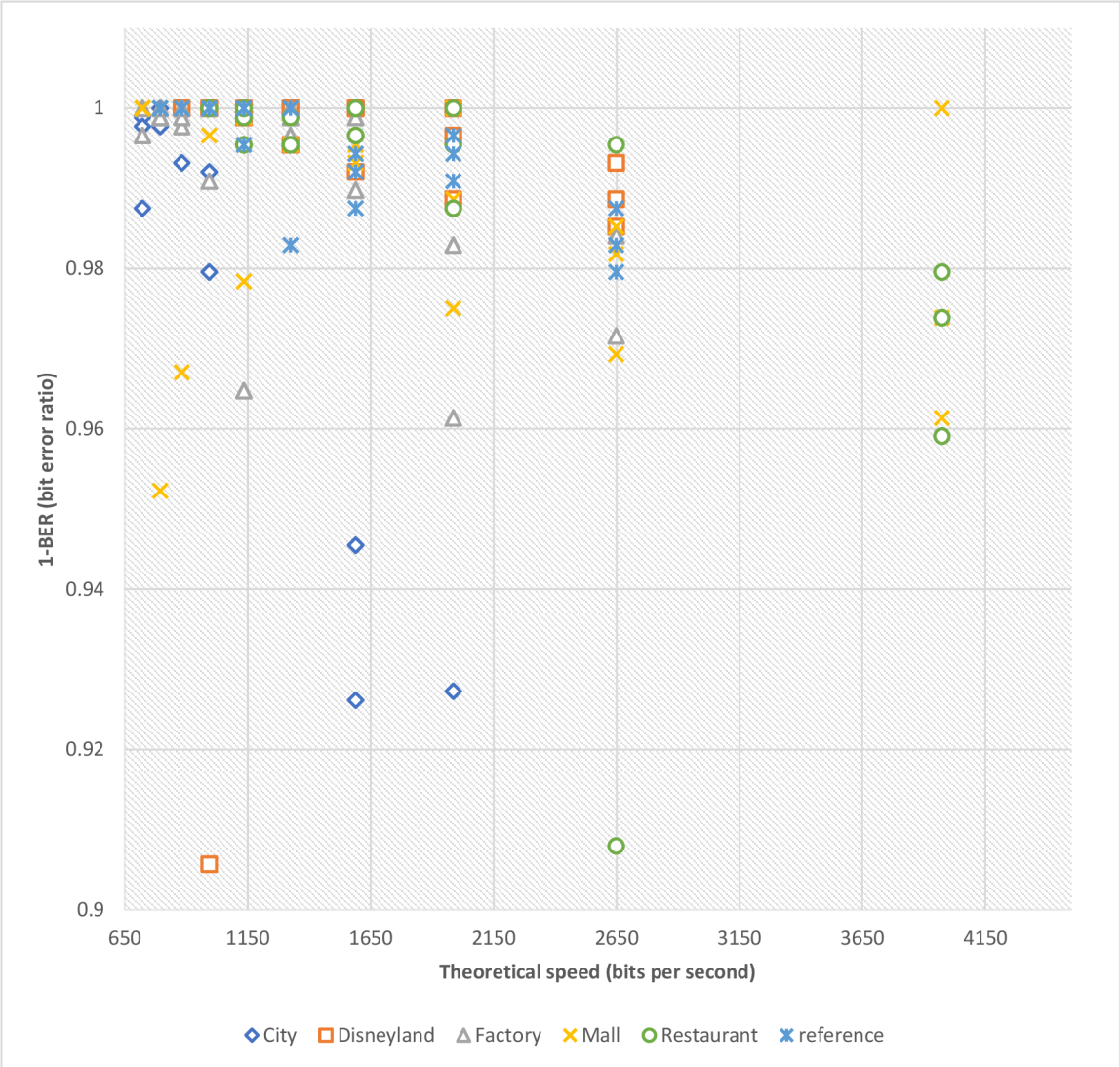


**Figure 5.11:** Tests performed in different environments. Frequency changed to 3974 Hz.

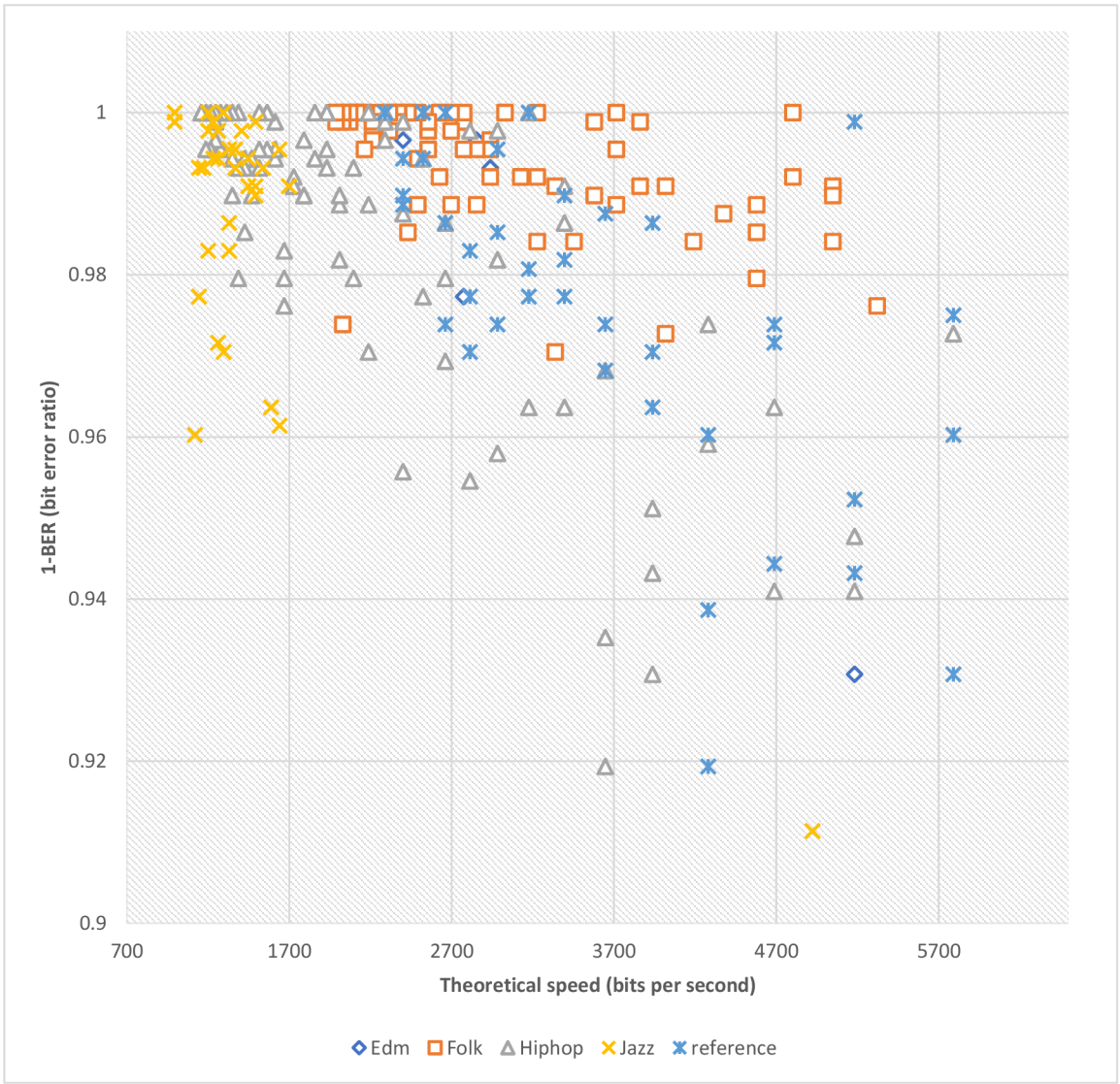


**Figure 5.12:** Testing with different music genres. Frequency was set to 24 611 Hz. Edm (electronic dance music) profile is conspicuously similar to the factory in Figure 5.10 - it is absent in the area of chart.

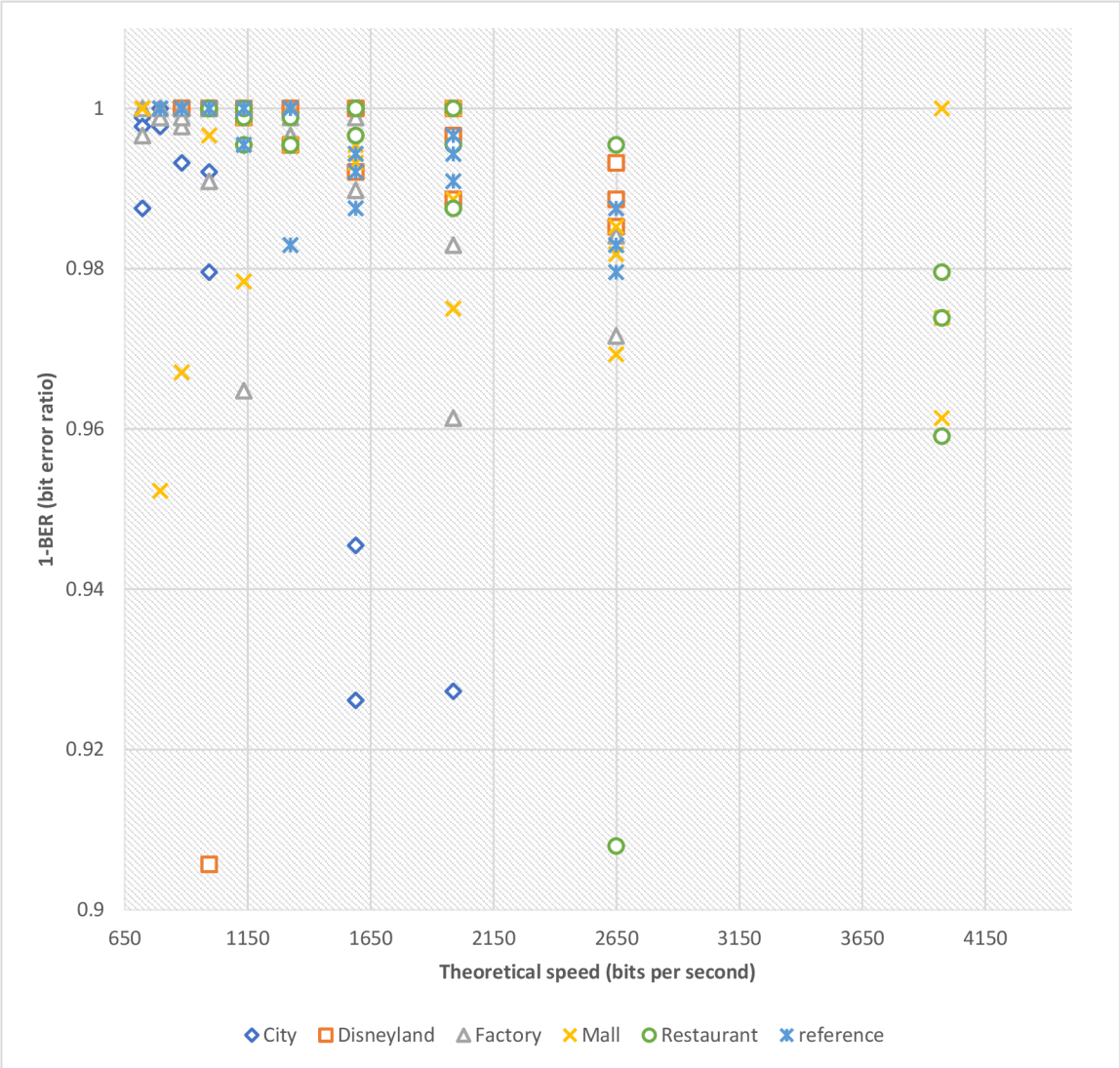


**Figure 5.13:** Testing with different music genres. Frequency is changed to 9890 Hz. Jazz appears to perform worse on this frequency when compared with Figure 5.12

**Figure 5.14:** One of the testing configurations. Similar one was used in generating Figure 5.8. Mono speaker connected to a laptop with integrated mic. The speaker is laid down to enable better directing of sound waves.

# Chapter 6

# Conclusion

This thesis discusses the possibility of transferring digital data through air using sound. It starts by research of existing solutions. There is about a dozen, most of them are commercial and a few of them are open-source. The work follows none, but it comes with its own solution.

The solution is based on knowledge and theory from digital communications which are meant to be used on radio-systems. Part of the research was taking theory into a simulated environment of Matlab. Successfully working version of algorithm was taken further. The theory part of the thesis describes different techniques of modulations, and the process of synchronization. Application developed along with the thesis uses QPSK modulation and symbol based synchronization.

Another part of describing implementation of the application. Core part is developed as an independent library with WPF front-end program. It is developed with the focus on testing of transmission and its components. There are extensions to enable visualization of what is happening. It was used to optimize and improve performance of the algorithm.

Last part of the thesis is dedicated to the testing of the application under different circumstances such as a different environment. During the testing it was found which frequencies are best to use. And for the future use - which environments can be expected open for the Transmission of Digital Information over Audio.

# Bibliography


[1] Chirp. [online, Accessed: 2019-05-11].
Retrieved from: https://chirp.io/

[2] CopSonic. [online, Accessed: 2019-05-11].
Retrieved from: https://www.copsonic.com/

[3] Google Nearby. [online, Accessed: 2019-05-11].
Retrieved from: https://developers.google.com/nearby/

[4] Google Tez. [online, Accessed: 2019-05-11].
Retrieved from: https://pay.google.com/intl/en_in/about/

[5] Google Tone. [online, Accessed: 2019-05-11].
Retrieved from: https://chrome.google.com/webstore/detail/google-tone/nnckehldicaciogcbchegobnafnjkcne?hl=en

[6] LISNR. [online, Accessed: 2019-05-11].
Retrieved from: https://lisnr.com/

[7] Pied-Piper. [online, Accessed: 2019-05-11].
Retrieved from: https://github.com/rraval/pied-piper

[8] Signal360. [online, Accessed: 2019-05-11].
Retrieved from: http://www.signal360.com/

[9] SinVoice. [online, Accessed: 2019-05-11].
Retrieved from: http://www.sinvoice.com/

[10] Sonarax. [online, Accessed: 2019-05-11].
Retrieved from: https://www.sonarax.com/

[11] ToneTag. [online, Accessed: 2019-05-11].
Retrieved from: https://www.tonetag.com/

[12] Armstrong, B.: Quiet Modem Project. [online, Accessed: 2019-05-11].
Retrieved from: https://github.com/quiet

[13] Bell, A.; Glanz, J.: System effective to demodulate a modulated code and provide content to a user. October 31 2013. uS Patent App. 13/845,500.
Retrieved from: https://www.google.com/patents/US20130288723

[14] Haykin, S.: *Digital communication systems*. Hoboken, NJ: J. Wiley & Sons. 2014. ISBN 978-0-471-64735-5.

[15] Heath, M.: NAudio. [online, Accessed: 2019-05-11].
Retrieved from: https://github.com/naudio/NAudio

[16] prof. Ing. Aleš Prokeš, P.: *RÁDIOVÉ KOMUNIKAČNÍ SYSTÉMY*. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky, Purkyňova 118, 612 00 Brno. 2013. iSBN: 978-xxxxxx.

[17] Kouraklis, J.: *MVVM as Design Pattern*. 10 2016. doi:10.1007/978-1-4842-2214-0_1.

[18] Rypuła, R.: Audio Network. [online, Accessed: 2019-05-11].
Retrieved from: https://audio-network.rypula.pl/

[19] Smith, S.: *The scientist and engineer's guide to digital signal processing*. San Diego (Calif: California Technical Pub. 1999. ISBN 0-9660176-6-8.

# Appendix A

# Software used

### A.0.1 Simulation

I simulated modulations in **Matlab**.

### A.0.2 Programming

Project named as **AudioTransceiver** was created in **Visual Studio** with the help of coding assistance of extension **ReSharper**. I backed source code up using **Git** in **GitHub** repository.

### A.0.3 Writing thesis

I wrote the thesis in **LaTeX**. I programmed the code using editor **Visual Studio Code** with Extension **LaTeX Workshop** and partially with online editor **Overleaf**. For creating diagrams was used **Draw.io**. I visualized charts in **Matlab**, **MS Excel** and some of them with help of **AudioTransceiver.Terminal**. For editing charts I also used **Adobe Acrobat DC**. I backed the work up using system **Git** in **GitHub** repository.

# Appendix B

# Program UI

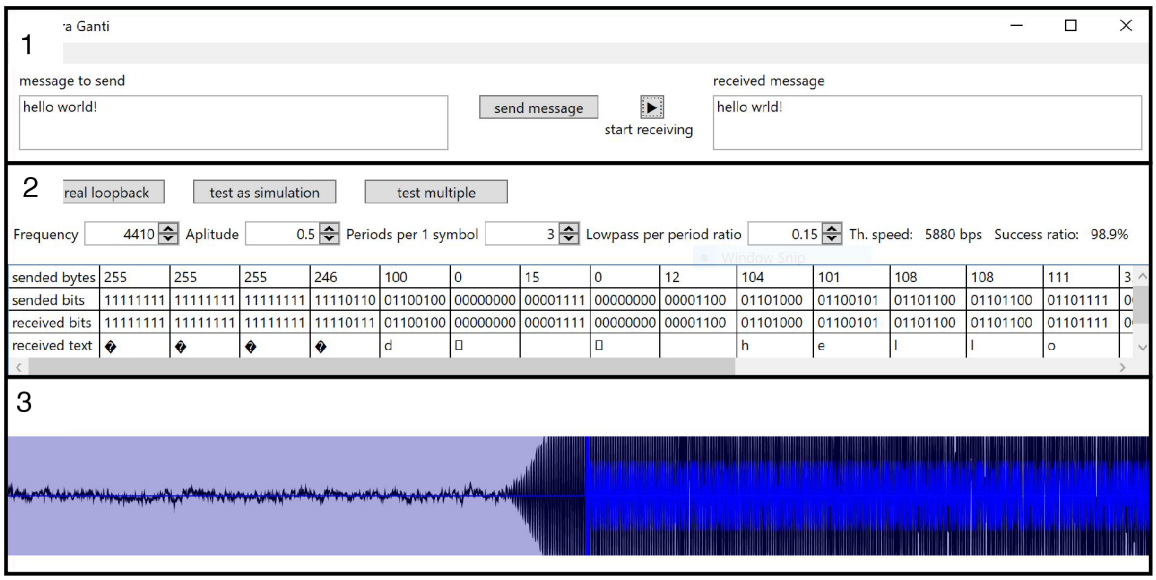


**Figure B.1:** UI divided into 3 parts.

Let's divide the program into 3 parts.

1. Sending of text messages.
2. Setting of parameters.
3. Signal visualization.

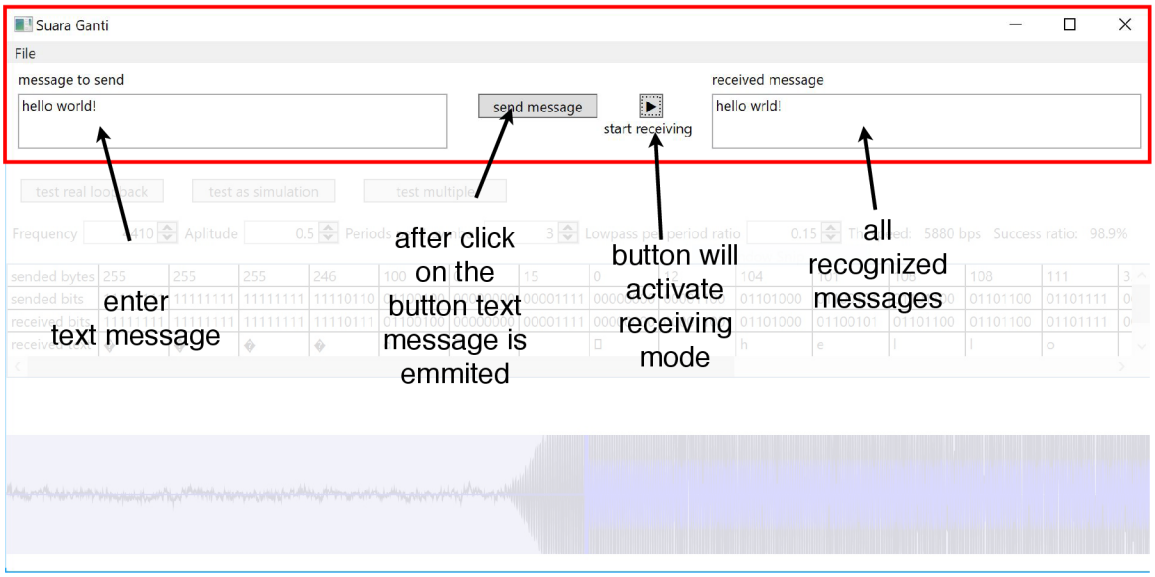


**Figure B.2:** Part 1: sending of a text message.

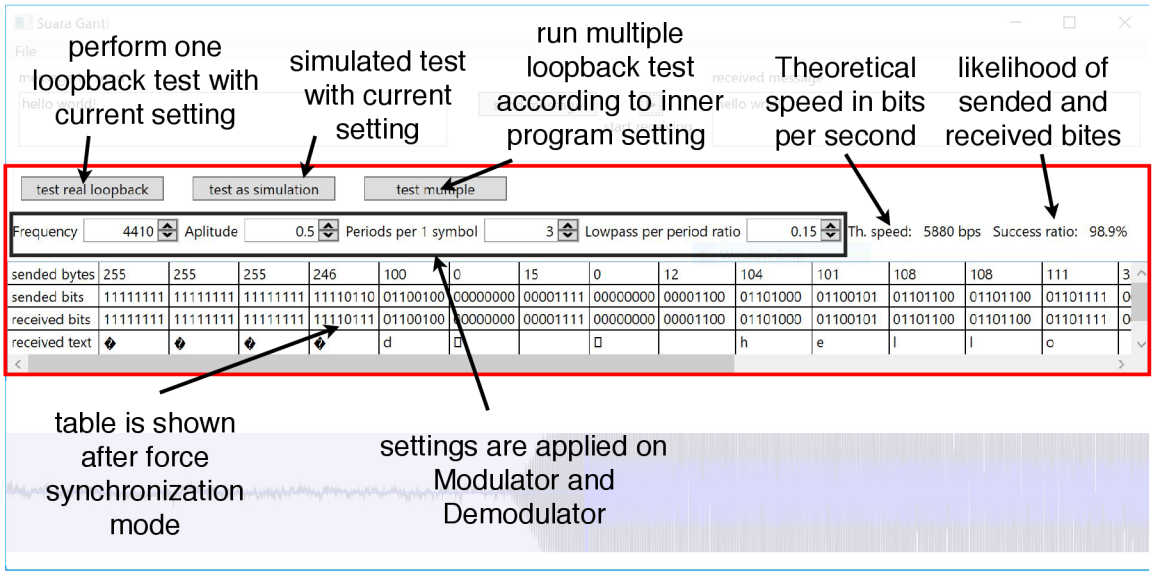


**Figure B.3:** Part 2: Setting of parameters, view of data.

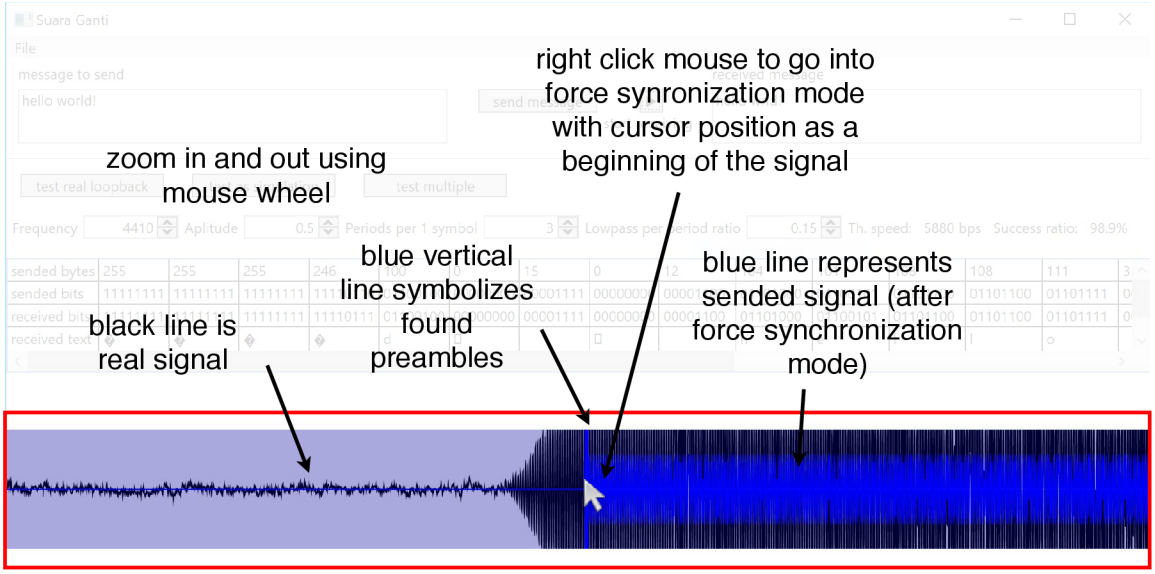


**Figure B.4:** Part 3: View of the accepted signal. After extending the window down, it will create multiple rows of the signal.

There are two general modes, how to test sending of data:

1. Receiving mode. It is activated after click on [play] button.
2. Force synchronization mode. It is activated after right click of a mouse into accepted signal area. We can generate signal after click on test buttons or from receiving mode.

   It is also possible to save your generated signal with preamble position into a file and later load it into the program.