Towards a Cross Platform Solution to Share Small Scale Data Across Smart–Devices Using Audio Waves

A prototype development in Android Environment

Submitted in partial fulfillment of
the requirements for the award of the degree of

Bachelor of Science
in
Electronic and Communication Engineering
&
Electrical and Electronic Engineering

Submitted by

Roll No	Names of Students
13110028	S. M. Ferdous Hasan
13110001	Adnan Bin Murshed
13321086	Ahrar Hayat

Under the supervision of
Rachaen Mahfuz Huq
Lecturer, Department of Electrical and Electronic Engineering,
BRAC University, Bangladesh

Department of Electrical and Electronic Engineering
BRAC University
66 Mohakhali, Dhaka – 1212

Department of Electrical and Electronic Engineering

BRAC University

Certificate

This is to certify that this is a bonafide record of the thesis presented by the students whose names are given below from January, 2016 to December, 2016 in partial fulfilment of the requirements of the degree of Bachelor of Science in Electronic and Communication Engineering & Electrical and Electronic Engineering.

Roll No	Names of Students
13110028	S. M. Ferdous Hasan
13110001	Adnan Bin Murshed
13321086	Ahrar Hayat

Rachaen Mahfuz Huq
(Thesis Supervisor)

Date:

Acknowledgments

We would like to seize the opportunity to express a big thanks to those who helped us conduct this graduation project. Special thanks to our supervisor, Rachean Mahfuz Huq, Lecturer , Department of Electrical and Electronic Engineering who made it possible by supervising us from the start to finish. Lastly, thanks to BRAC University for all their logistic assistance with it's resources, such as workstations which helped the smooth finish of the project.

Abstract

In the present era of research on Internet of Things (IoT) and Machine-type-Communications (MTC), there is a growing need of cross platform solutions for short range communication techniques between machines. The short range communication standards of present day (Wi-Fi, Bluetooth etc) were mainly developed for large data streams, but machine type communications data are usually smaller in size, less complex in nature, but more frequent and more congestive due to the deemed large volume of participatory devices at a particular area. In this project we explored the potential of audio waves as a interoperable cross-platform solution for small scale data and short range Machine-type-Communications (MTC).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

In modern day telecommunication there are several techniques of transmitting and receiving data, each uses different methods and media. Radio frequency electromagnetic waves play a major role in day to day wireless communication, such as with GSM, Bluetooth, Wi-Fi along with other electromagnetic waves such as infrared [1]. Machine to machine (M2M) communication is the direct communication between devices with little or no human intervention[2]. By 2020, it is predicted that there will be 50 billion[3] connected devices in the world, those devices will have hundreds of different manufacturers and the manufacturers will have hundreds of different platforms. Therefore, these devices will require a cross platform communication solution between them in order to have interoperability. Interoperability between devices will have a key role in M2M communication. The sound waves can be used for such short range M2M communications. Using sound waves for M2M communication will require only a set of working speaker and microphone, without the need of a separate embedded antenna pair which any existing short range radio frequency communication protocol would require for transmission and reception. Therefore, the sound wave can be used as an alternative media for M2M communications. Thus, our motive is to develop a technique which uses the sound wave as a possible alternative to the various conventional existing technologies that uses the electromagnetic spectrum for transmitting and receiving small data in close proximity.

1.2 Problem Area

Internet of things is the internetworking of physical devices to exchange information between them [3]. In the era of Internet of things, sound waves could play a major role. Development of smart cities have been rising rapidly and is in desperate need of machine to machine communication. An ideal smart city would take advantage of smart devices to communicate in day to day activities. Smart devices and autonomous machines can communicate with each other using sound waves for numerous applications. Autonomous machines used at home and in the commercial industry such as printers, computers, refrigerators, TV remotes, air conditioners, vehicles, electronic equipment in smart grid such as transformers, factory equipment may develop the use of ultra-sonic sound waves in order to communicate.Refer to the Figure 1.1 for a visual representation of a smart city using machine to machine communications.Therefore, for developing a smart city, companies may use this technology to develop autonomous machines that would be compatible for using sound spectrum as the transmission media.

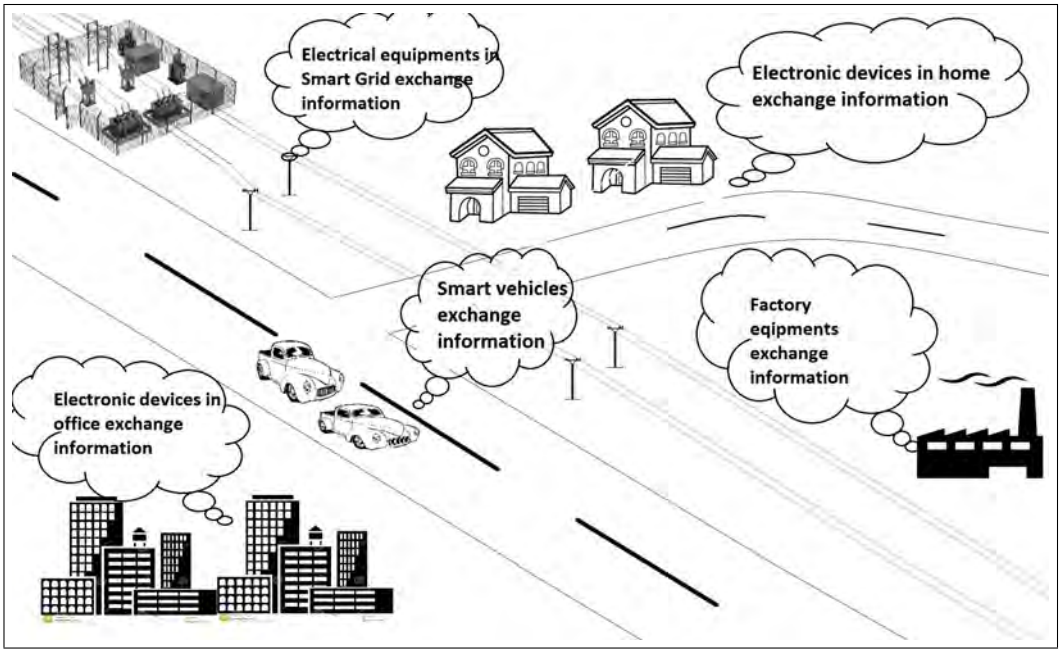

Figure 1.1: Devices in a smart city capable of M2M communictions

1.3 Previous Works

The idea of using sound waves as a medium of transmission was used to implement certain applications before. Using public switch telephone network to connect to the internet via a dial-up modem was a popular design in the past [4]. In order to connect to the internet, the regular telephone land line cable would be connected to a dial-up modem and an output cable from the modem was connected to the computer, the data was encoded and transmitted to be decoded at the receiving side. All the data transfer was completed using the sound spectrum which used transmission methods such as Frequency Shift Keying and Quadrature Amplitude Modulation [4]. Previous works on the development of the Internet of Things have been done. Tan et al.[5] proposes the development of the Internet of Things using Radio Frequency Identification (RFID) techniques. The paper discusses how human to human communication evolves into human to machine communication and eventually transforms into M2M. Therefore, previous works used sound waves, but for a different purpose, also previous works were on proposed development on the Internet of Things but did not use sound waves to do so.

1.4 Research Gap and Contribution of the Thesis

Previous works on the quest to achieve a cross platform solution have left gaps in the research that are yet to achieve closure. The research gap lies due to unavailability of a cross platform between devices. In this thesis, we focus on the development of using sound waves as a communication media to achieve interoperability between devices, so that we can finally close the gap in research. As mentioned before in section 1.1 there will be 50 billion connected devices across the globe by 2020 [3]. However, it is unlikely for the devices to use the same technology. The 50 billion devices will have different manufacturers, that is why they will have different operating systems and hardware. A device that uses Wi-Fi or NFC may not be capable of communicating with a device that does not use Wi-Fi or NFC. Even if the devices have the hardware to use such technologies, they might not be able to communicate due to nonconformity in the software. Therefore, the need for a cross platform solution is crucial for the development of machine to machine communication. Specifically, the use of sound spectrum may be vital for the development of a cross platform standard. Hence, the gap in the research may finally achieve closure with a cross platform solution. This

thesis contributes to that development by providing a model and an android application, capable of communicating using sound waves, only requiring the device to have working speakers and a microphone. The basic algorithm of the technology is, to input bit streams containing the data, and transmitting them by sound waves with frequency in the audible range, at the receiver, the sound is received by the microphone and the original information transmitted is extracted. Therefore, the sound waves may play a pivotal role in the future developments of Internet of Things.

1.5 Research Question

"How to achieve a cross platform interoperable solution to small scale data transfer for machine to machine communication application ?"

1.6 Outcome of the Thesis

Our main goal was to transmit a data or key via sound waves. The entire process was done in several unique steps which are mentioned above and elaborated in the later parts. All these steps together form an algorithm which is not only applicable for the android platform but also almost all other machines and devices that might require doing the same task on similar conditions. Hence, the algorithm itself is one of the basic and tested outcome of our thesis work. Secondly, the application which is developed is also unique in its nature in sensor based android application platform and is also an important outcome of the thesis. Besides, the research paper can be used for further research work whenever any research group would want to carry on their thesis on this field. It is one of the most vital outcome since not much of any research work has been previously done on this topic. Lastly, the test results by varying various parameters such as the high and low frequency, sampling rate, noise canceling Root Mean Square (RMS) value used in the application, distance, error rate and effect of hardware on the overall performance of the system is also an outcome to understand the efficiency and applicability of sound as an alternative form of energy other than the conventional electromagnetic wave to send and receive small scale data on a short distance between devices running on different platform.

1.7 Methods Adopted

The android application is developed using *Android Studio* and a number of algorithm compiled in *Java*. The basic idea of transmitting data using a audio wave is to convert a digital data into analog signals with two different frequencies so that a binary 1 or a 0 can be distinguished at the receiver in order to successfully decode the encoded data. As sound wave is analog and out data is digital, binary frequency shift keying is used for digital to analog conversion. Then to transmit the data, an audio track which represents the data is played by Android media player. The audio track of PCM format was created by *AudioTrack* class and played through the speaker via android music player function as audio sample.
On the receiver's end the audio signal is recorded via android audio recorder and after several evaluation steps to separate noise from original signal and the frequencies from the recorded audio sample is figured out using zero-crossing detection method. Then the final output is generated from the detected frequencies in reverse process. Figure 1.2 shows the overview of the working mechanism of the application.

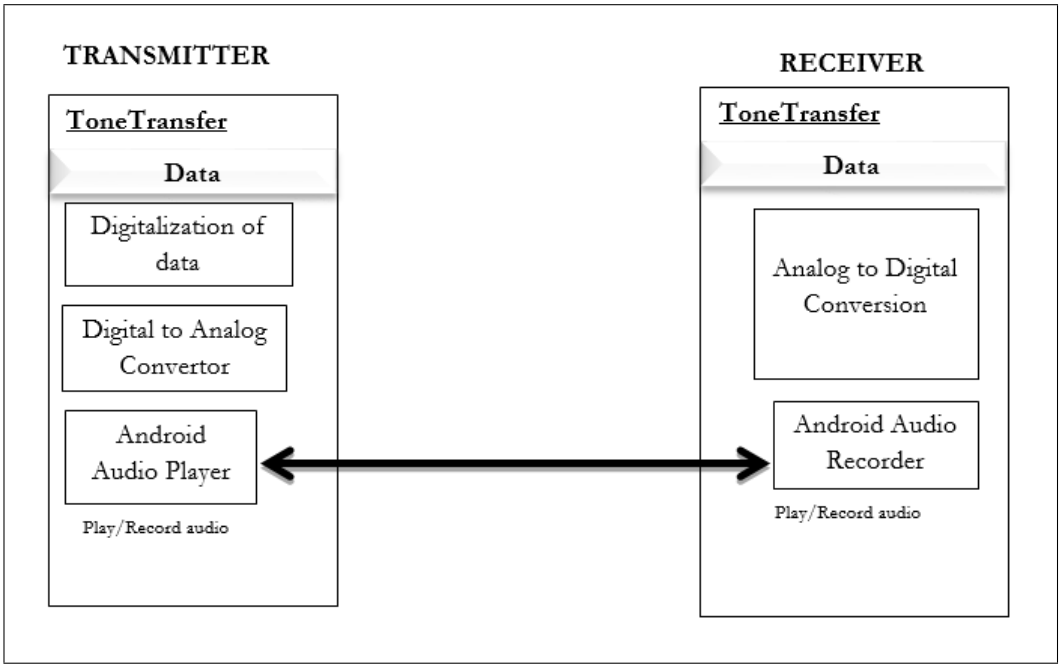

Figure 1.2: Overview of the mechanism of the application

1.8 Summary of the Following Chapters

The rest of the thesis is organized as follows. In Chapter 2, major definitions of the methods adopted on the thesis and coding schemes is illustrated. This chapter also points out the background on Android Sensor Programming. Chapter 3 illustrates the step by step coding techniques for the development of an Android application with public APIs. Design challenges and considerations in developing the application is also pointed out here. Through the experimental results, the performance of the application is evaluated in chapter 4. Finally, we conclude the paper with discussion on the future work that can be done in chapter 5.

Chapter 2

Theoretical Background

2.1 Data Communication [6] [7]

2.1.1 Data Communication System and its Components

Data communication is the exchange of information between two or more devices via transmission medium such as wired or wireless. For data communication to occur, the communicating devices are also the part of the communication system and those devices are the combination of hardware and software. The data communication system consists of several components depending on types of communication. The four primary components are data, transmitter, receiver and transmission medium. The data to transfer can be in different form such as text, number, images, audio, and video. Besides devices may need to share code or key (combination of alphabet and number) to exchange information between them. Throughout this paper, the data will be considered as a key which will identify the file.

2.1.2 Character Encoding

In data communication, data to transfer is represented as a bit sequence (0s and 1s). Different set of bit patterns have been designed to represent a character. One of the character encoding techniques is UTF-8. UTF-8 is a character encoding capable of encoding all possible characters, or code points, defined by Unicode. UTF stands for Unicode Transformation Format. The '8' means it uses 8-bit blocks to represent a character. A symbol or alphabet in UTF-8 format usually contains 8-32 binary bits since it follows a variable length encoding system.

2.1.3 Data flow

The communication between two devices can be of three types such as simplex, full-duplex and half-duplex. The flow is unidirectional in simplex mode and bidirectional in both half and full-duplex mode. The only difference between half and full-duplex is that in full-duplex mode, the communicating devices can act like both transmitter and receiver simultaneously and in half-duplex mode, the communicating devices can both transmit and receive but in different time.

2.2 Transmission of Data

There are two different approaches to transmit a digital data from one device to another [7].The approaches are baseband transmission and broad band transmission (using modulation). Baseband transmission means sending a digital signal over a channel without changing the signal to a analog signal. Whereas, broadband transmission means changing the digital signal to an analog signal before transmission. Generally, the data usable to an application are not in a form that can be transmitted to a wireless medium. To be transmitted in wireless medium data must be transformed to an analog signal.

2.2.1 Analog and Digital Signal

Signal to represent a data, can be both analog and digital. The term analog signal refers to a simple sine wave which have three characteristics: amplitude, frequency, and phase. The term digital data refers to the data in form of 0s and 1s that can be converted to an analog signal for transmission in wireless medium. This analog signal can be captured by a microphone and sampled and converted to a digital signal again. This how the transmission of data occurs between two devices where one functions like a transmitter and another as a receiver.

2.2.2 Composite Analog signal

Composite signal means a combination of simple sine waves with different frequencies, amplitude and phases. For two devices to communicate needs a composite signal created from a digital data. Modulation is a well known process to convert digital data to a composite analog signal.

2.3 Digital to Analog signal

Digital to analog conversion (D/A) is a process of changing one characteristics of an analog signal based on the digital data. When any characteristic of a simple analog signal is varied, a different version of that signal is created. So, by changing one characteristic of an analog signal, it can be used to represent a digital data. Any of the three characteristics can be varied in this way, give rise to three mechanisms for modulating digital data to an analog signal [8]. These mechanisms are:

- Amplitude Shift keying (ASK)
- Frequecy Shift Keying (FSK)
- Phase Shift Keying (PSK)

2.3.1 Amplitude Shift Keying

Amplitude Shift Keying is a modulation scheme used to transmit data by changing amplitude of the carrier signal. Both frequency and phase remain constant. In Data communication, ASK can be implemented using two levels and this method is referred as binary amplitude shift keying or on-off keying (OOK). As illustrated in Figure 2.1, let the binary data stream **101010** is to be transmitted, the carrier signal will only appear for the time period only when there is a high bit, otherwise it will be zero. Therefore, on the receiving side, detecting any signal would mean a 1 and lack thereof would mean a 0 for a successful decoding of the modulated signal.

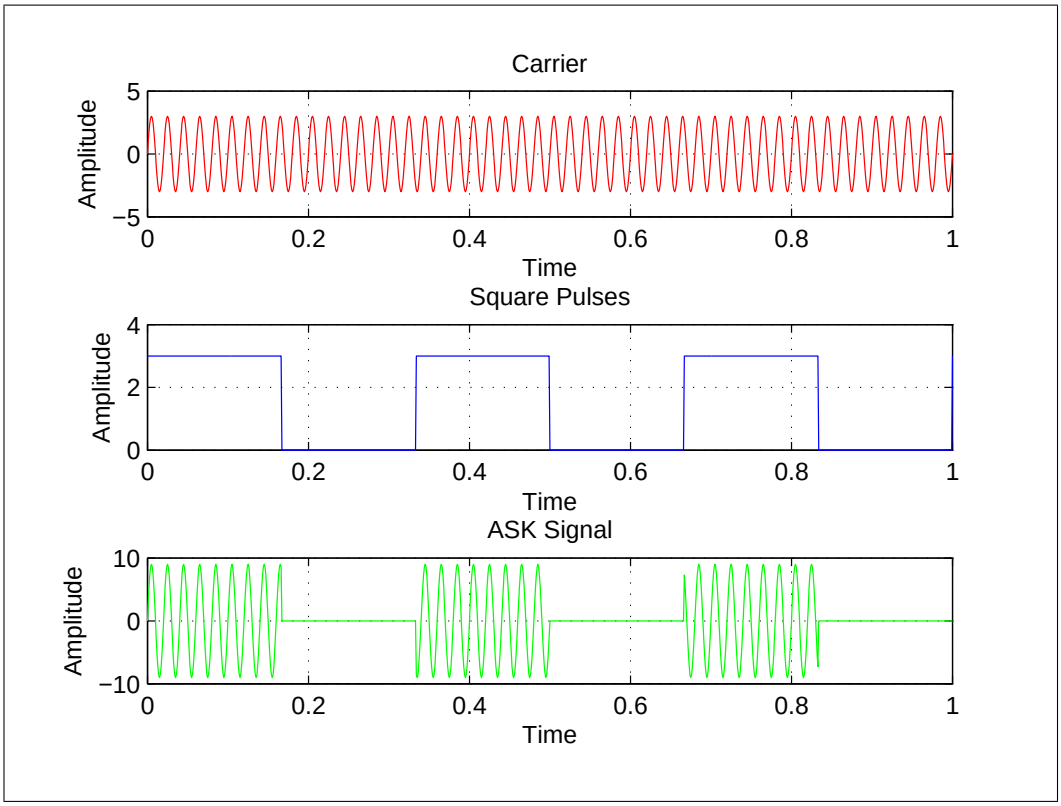

Figure 2.1: Waveform of a binary ASK signal

2.3.2 Phase Shift Keying

In phase shift keying, the phase of the carrier is varied to represent two or more different signal elements. Both peak amplitude and frequency remain constant as the phase changes. The simplest phase shift keying (PSK) is binary PSK, in which we have two signal element to represent the digital data, one with a phase of 0^0, and the other with a phase of 180^0. As illustrated in Figure 2.2 a digital data 101010 is to be transmitted. The phase of the carrier signal will be changed for the first time for the 0 next to the most significant bit and change again for the 1 next to it, and at last the phase will change again for the least significant bit. The phase changes will be detected on the receiving side to decode the information on the data stream.

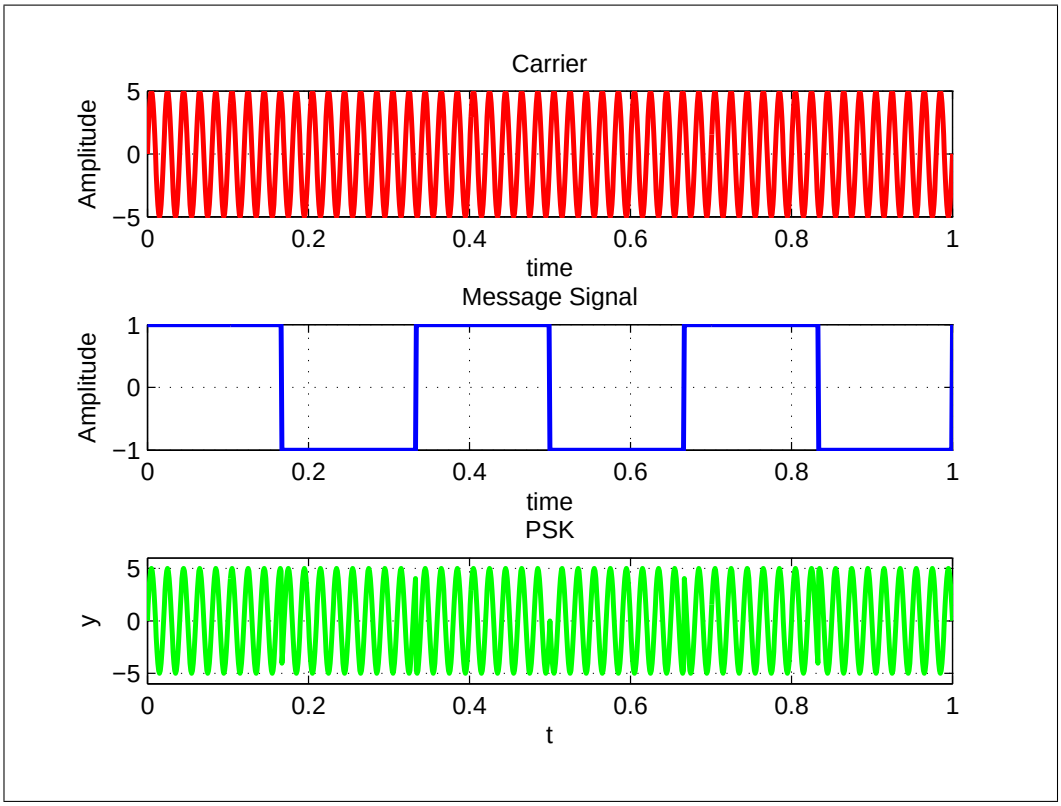

Figure 2.2: Waveform of a binary PSK signal

2.3.3 Frequency Shift Keying

In frequency shift keying, the frequency of the analog signal is varied to represent data. The frequency of the analog signal is constant for the duration of one signal element, but changes for the next signal element if the data element changes. Both peak amplitude and phase remain constant for all signal elements. The simplest FSK is binary FSK (BFSK). BFSK uses two frequencies to represent binary 0 and 1 as illustrated in Figure 2.3.

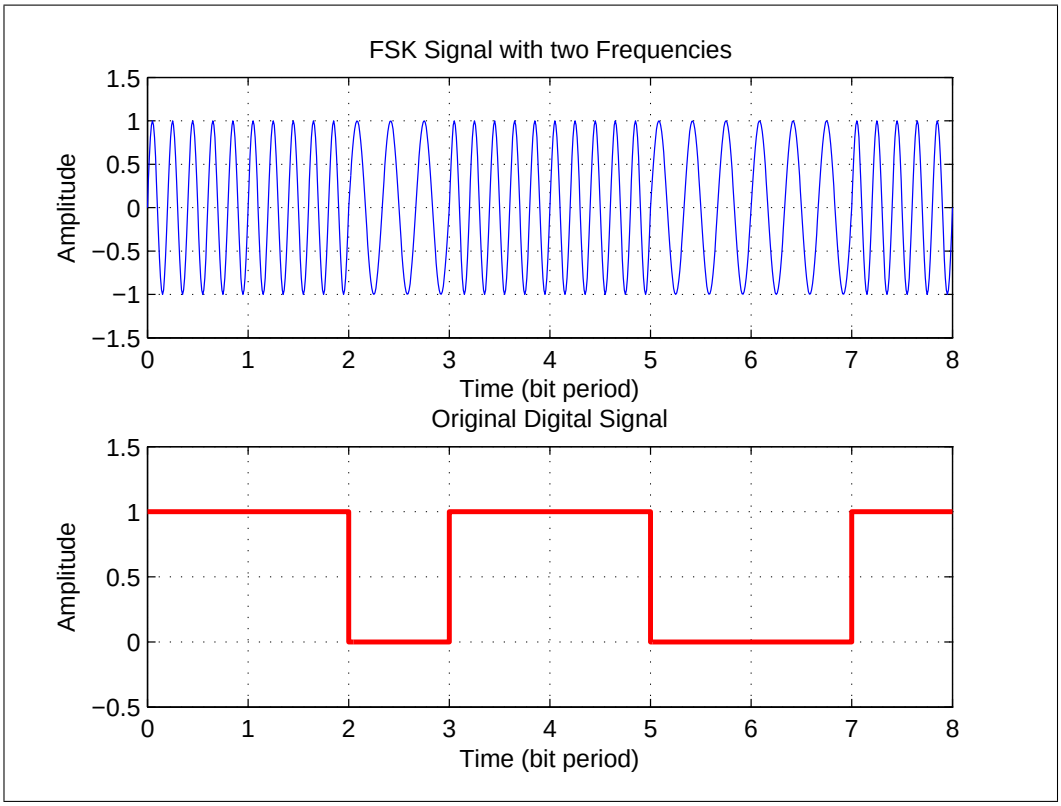

Figure 2.3: Waveform of a binary FSK signal

2.3.4 Implementation of Binary FSK

There are two implementations of BFSK: noncoherent and coherent. In noncoherent BFSK, there may be discontinuity in the phase when one signal element ends and the next begins. In coherent BFSK, the phase continues through the boundary of two signal elements. Noncoherent BFSK can be implemented by treating BFSK as two ASK modulations and using two carrier frequencies. Coherent BFSK can be implemented by changing the frequency of the analog signal according to the input bit. Figure 2.4 shows the simplified idea behind the implementation of coherent BFSK.

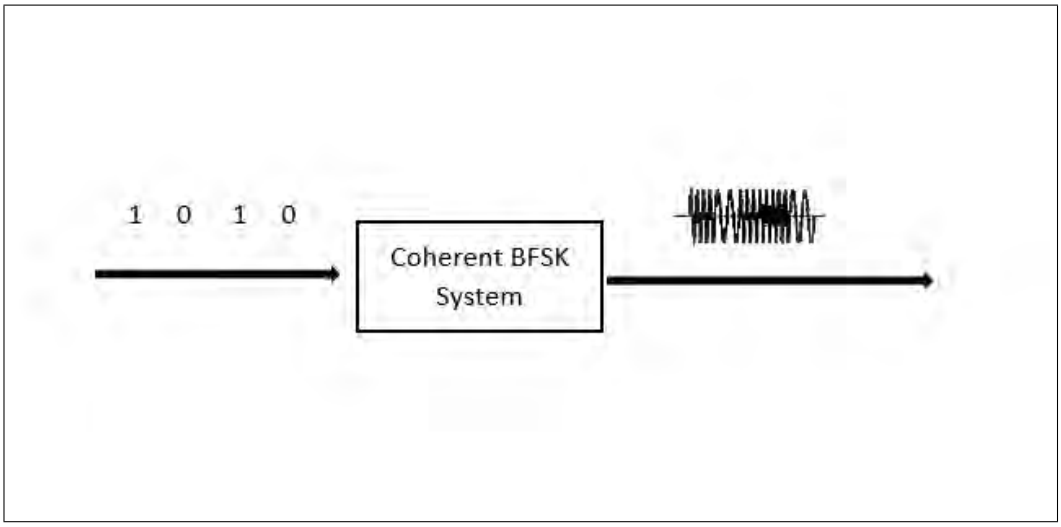

Figure 2.4: Implementation of binary FSK

2.4 Analog to Digital Conversion

Analog to Digital (A/D) conversion is a process of retrieving the digital data which the analog signal represents [9]. There are various process of A/D conversion. In this section only one technique, pulse code modulation is described. A PCM encoder has three steps as illustrated in Figure 2.5.

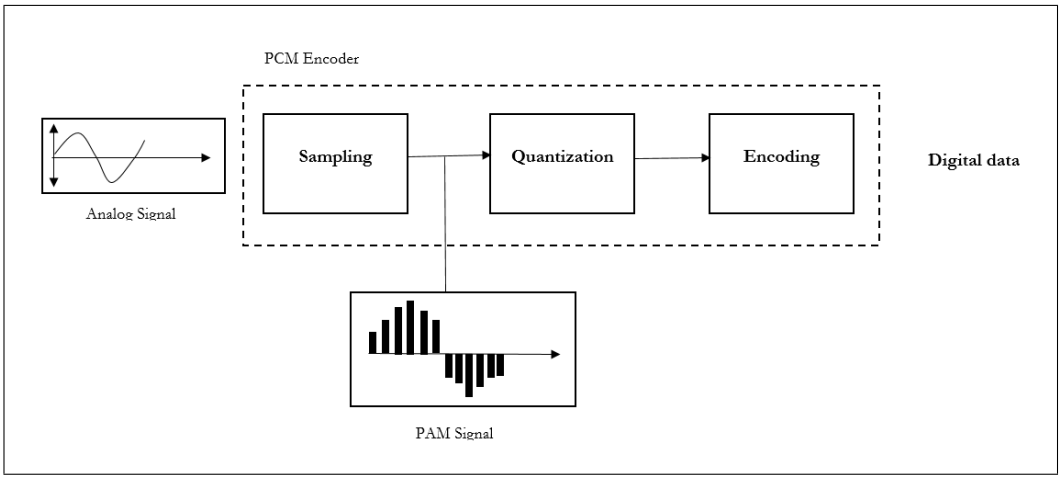

Figure 2.5: Block diagram of PCM encoder

2.4.1 Pulse Code Modulation

The most common technique to change an analog signal to digital data is named as pulse code modulation (PCM). As shown in Figure 2.5 the steps are

1. The analog signal is sampled.
2. The sample signal is quantized.
3. The quantized values are encoded as stream of bits.[13] [14]

PAM Signal

The sampling process produces the pulse amplitude modulated (PAM) signal, which contain the value of the amplitude of each samples. Figure 2.5 also points outs the PAM signal generated after sampling process which leads to the calculation of number of zero-crossing points.

2.5 Zero Crossing

A zero-crossing is a point where the sign of a mathematical function changes from positive to negative or vice-versa.In a sine waveform, the point where the sample is at zero amplitude is called zero-crosing point, and this point occurs twice during each cycle. As illustrated in Figure 2.6 x and y is the zero-crossing point.

By counting zero-crossings, the fundamental frequency and period of a periodic analog signal can be estimated.

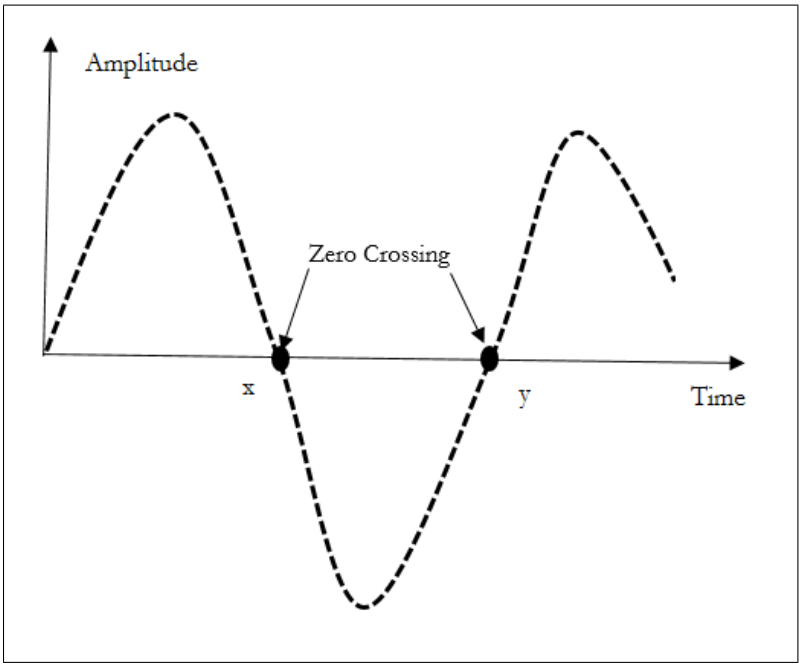

Figure 2.6: Zero-crossing in a waveform representing amplitude vs time

2.5.1 Zero-crossing Detection

As mentioned earlier, zero-crossing point in a period is the point where the amplitude is zero and at any other point the amplitude of the wave is rising towards its positive peak or sinking towards its negative peak. When the value of amplitude of a sample is positive and the value of the next sample amplitude is negative, we gets a zero-crossing point. Number of zero-crossing can be used in the calculation of frequency of a signal [10].

2.6 Root Mean Square Amplitude [15]

Calculating the average amplitude of a sine wave, it would unfortunately results to zero, since it rises and falls symmetrically above and below the zero reference. This would not tell us very much about its amplitude, since low-amplitude and high-amplitude sine waves would appear equivalent. A more meaningful reference has been developed to measure the average amplitude of a wave over time, called the root mean squared(RMS) method. The RMS value of a set of discrete values or a continuous function is the square root of the arithmetic mean of the squares of the discrete values, or the square of the function.

In the case of a set of n values $\{x_1, x_2, \ldots, x_n\}\{x_1, x_2, \ldots, x_n\}$,

$$x_{rms} = \sqrt{\frac{1}{n}(x_1^2 + x_2^2 + \cdots + x_n^2)}.$$

2.7 Android Sensor Programming Background and Android Environment

Unlike a personal computer or a laptop an android device is not a device that only has a central processing unit, bunch of permanent and volatile memory units and some graphics processors. What makes an android device so special and unique from others is the integration of a large number of sensors in such a small device that increases its capacity manifold. The first of the android devices that came into market in the earliest age of 21st century were not so great in their hardware and software specifications but quickly captured market due to some very specific factors and the right use of android sensor environment is a one of them. The most basic kinds of sensors for an android device are:

- Touch screen sensor
- Audio microphone sensor
- Photo sensors or camera
- Location sensors
- Magnetic field sensor
- Orientation sensor
- Proximity sensor
- Temperature sensor etc

The later versions of android phones are integrated with some more advanced bunch of sensors that can monitor the physical environment such as light, relative humidity, rotation vector, gravity, ambient temperature, Near Field Communication(NFC) scanner and so on [11]. Some other external sensors can be added via Android Open Accessory(AOA) system. Iris detector, voice recognition and fingerprint sensors have made these devices more secure. In addition to this, the basic kind of sensors are made more sharp and sophisticated that can sense even the slightest change in the physical environment.

Such an example can be low light camera which has given birth to a new term in the dictionary known as mobile photography that can be almost as good as a professional photographer's camera in some cases. Heart beat sensors, location trackers and many other features has been added to this family as an extra advantage for health-conscious population who can now monitor their physical condition and the amount of calorie they could burn while doing physical exercise measuring the distance they covered and number of steps taken while doing so. Addition of all these features in a common platform has made android a favorite platform for both the end customers and android developers who can now use this advanced integrated system to develop applications that can do amazing things.
While coding an application in android that uses the sensors a developer must keep in mind three things [12]

- What are the sensing capabilities of the target device
- Which sensors might be required to generate output in the application and
- How to call and interpret the data in the application obtained from the sensors.

There is a built-in framework in android development environment consisting of a wide range of methods and command through which we can call all types of sensor in the system. Later, we utilize the obtained sensor outputs as parameters and perform all types of logical operation using a high-level programming language such as java codes or codes for an android development kit like android studio. Once properly utilized these bunch of codes can generate the necessary output based on which an android application can run an interact with its user with meaningful data through the User Interface(UI) of the application.
Before starting to develop an application, an android developer must mention the minimum Application Program Interface(API) level which indicates the minimum version of operating system that the application can run on. An application with lower API level can serve more users since many users do not update their phone regularly. But as a tradeoff a lower API level will not allow the programmer to use many necessary built-in functions and sensors that comes with the updated version of devices and their operating systems. In addition to fixing the API level, a developer also creates a path to save the source codes along with few other necessary items. Next, the user interface is created which can be done both by coding or on the built-in graphical interface which is saved with a .XML file extension. The input-output boxes along with operational buttons, background and other elements

in the UI can also be developed on other graphics designer software and used in the UI. Each operational button and input-output box has their unique identification name that is used in the code to operate with them. Hence a combination of the right sensors controlled by the correct code which follows the right algorithm to solve a problem or serve a purpose, backed up by a user-friendly UI can bring the best out of an android device and meet the need of a user much efficiently.

Chapter 3

Design Method

This Chapter describes the design considerations taken into account during the development of the application. Specifically, the next section shows the general architecture of the system, in which the main blocks are presented: transmitter and receiver, as illustrated in Figure 3.1. These Transmitter and receiver mechanisms are explained in the following sections.

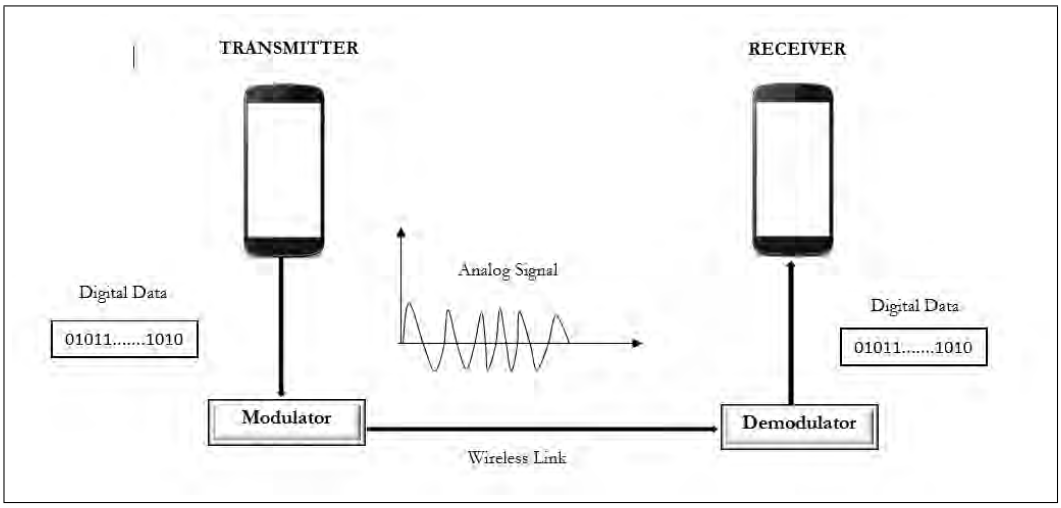

Figure 3.1: Data communication Block

3.1 Transmitter Architecture

The application hereby presented is unidirectional, that is, the receiver device needs to be in recording mode when a device is transmitting. When the app is initiated, user comes across two options named transmitter and

receiver, like all communication systems. Both transmitter and receiver has its own mechanism. The code or key to be sent is digital but transmitting signal has to be analog as the communication medium is sound wave and the channel is wireless. So, the transmitter got the mechanism to convert the digital signal to an analog signal to play. After conversion the analog signal is saved as an audio format and the app initiate android media player to play the audio. Now, the following described methods will complete the process. For example, the key to be sent be XYZ. This code needs to be converted to a digital domain which is the input of D/A converter and the generated output be played/transmitted by the transmitter side of the app, as illustrated in Figure 3.2. However, note that this is an unrealistic example used only for demonstration purpose.

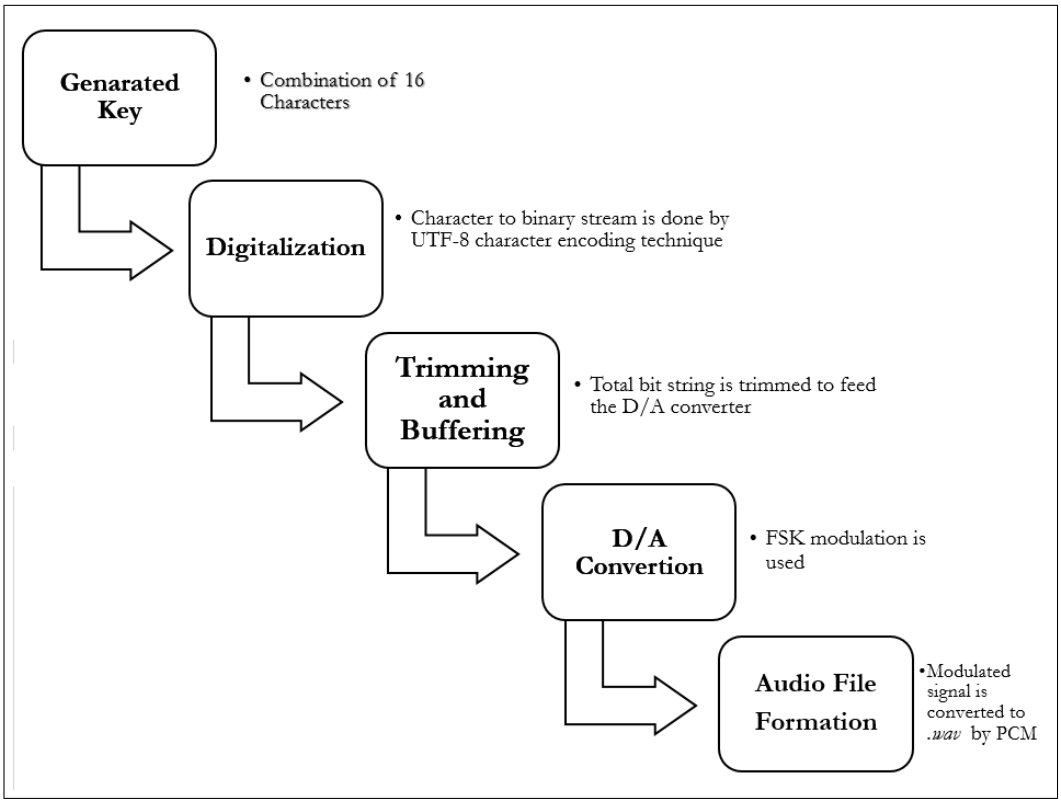

Figure 3.2: Block Diagram of Transmitter's mechanism

3.1.1 Encoding

Text to Binary bit

Java String **getBytes()** method encodes a string into a sequence of bytes using the platform's default charset and storing the result into a new byte array. The code or key that has to be transmitted is first digitalized by a character encoding technique, UTF-8.

So, here that XYZ is converted to

010110000101100101011010

Now this binary byte array is run through a trimming and buffering algorithm to feed the D/A converter.

Trimming and Buffering

First the byte array is compared with the pre-determined buffer size. If it is less than buffer size the encoder wait for the next input. In case, bit length is above buffer size, the size is trimmed to make it equal to buffer size.

Digital to Analog Signal Conversion

As discussed in Chapter 2, the project uses BFSK mechanism to represent the digital key. So, we select two frequencies, f_1 and f_2 to represent data element 0 & 1 respectively. Any two high frequencies with satisfactory deference between them will satisfy the purpose. For demonstration, let us consider $m(t)$ be the required analog signal where $f_1 = 4.9kHz$, $f_2 = 7.35kHz$. So, the formula to generate $m(t)$ is as follows

$$m(t) = 128 + A \sin [2\pi f t]$$

Where,A =amplitude,f =frequency, Sample rate = $44.1kHz$ and t = time period.

In case of frequency determination, if the bit if high then $f = f_2$ and if the bit is a 0 then $f = f_1$ frequency is used.

3.1.2 Playing Analog Signal

Android platform has a build-in class to write the analog data to a audio track. That is,
AudioTrack (int streamType, int sampleRateInHz, int channelConfig, int audioFormat, int bufferSizeInBytes, int mode)

To play the analog data we need this **AudioTrack** class. This class will build the track that needed to play by the android media player. Setting up this object requires an audio source, channel configuration, encoding and buffer size. The type of audio stream is selected as **STREAM_MUSIC** which is the audio stream for music playback. There is a convenient method called **getMinBufferSize()**, that can calculate the buffer size depending on the provided configuration of the devices' hardware. Furthermore, channel configuration is chosen as **CHANNEL_OUT_MONO** and to operate the **AudioTrack** instance among two modes, pointed out in Chapter 2, streaming mode is being selected.
Now we calling **play()** signals the **AudioTrack** to begin playing.

3.2 Receiver Architecture

The receiver is kept ready to listen and record the audio. Then the signal has to be converted to digital again from the analog signal. Figure 3.3 illustrates the basic blocks of the Receiver.

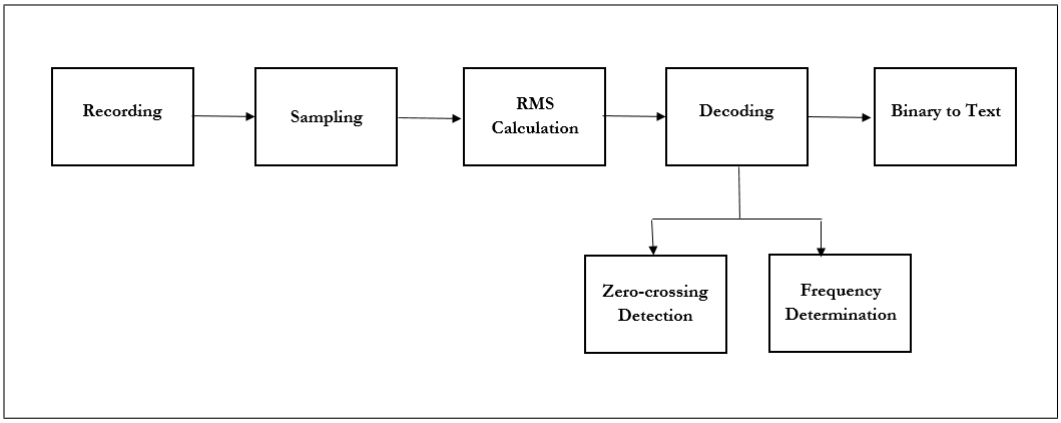

Figure 3.3: Block diagram of Receiver Architecture

3.2.1 Recording the Sound Played

The receiver is kept in recording mode. Recording the transmitted audio data is done through using a build-in class named **AudioRecord**. The syntax is,
AudioRecord (int audioSource, int sampleRateInHz, int channelConfig, int audioFormat, int bufferSizeInBytes)

All the parameters here are selected as same as the **AudioTrack** class. Only change here is audio source parameter and is selected as **MIC**. After the recording is finished the analog signal is sampled with the pre-determined sampling frequency and then fed to the RMS amplitude calculation box .

3.2.2 Sampling

The recorded audio is first sampled with sampling frequency 44.1 kHz to get a PAM signal. This PAM signal helps in following calculations.

3.2.3 RMS Amplitude Calculation

The RMS amplitude of the samples is calculated here. The equation to calculate RMS amplitude is as follows.

$$RMS = \sqrt{\left[\frac{1}{n}\sum_{n=0}^{n} x_n^2\right]}$$

Where n is the number of sample.
The data is only fed to the decoder if and only if the RMS value is greater or equal than the pre-determined optimum value.

3.2.4 Decoding

The application uses Zero-crossing detection method to calculate the transmitted high and low frequency combinations. The following subsections describe the method of determining frequency step by step.

Zero-Crossing calculation

The Zero-crossing point is computed, focusing on the changes of sign of the signal, using a sample-by-sample sequential algorithm. When the value of amplitude of a sample is positive and the value of the next sample amplitude of the PAM signal is negative, we gets a zero-crossing point. Thus, when this condition is satisfied the number of zero-crossing is increased. The following block diagram in Figure 3.4 shows the process to calculate the number of zero crossing.

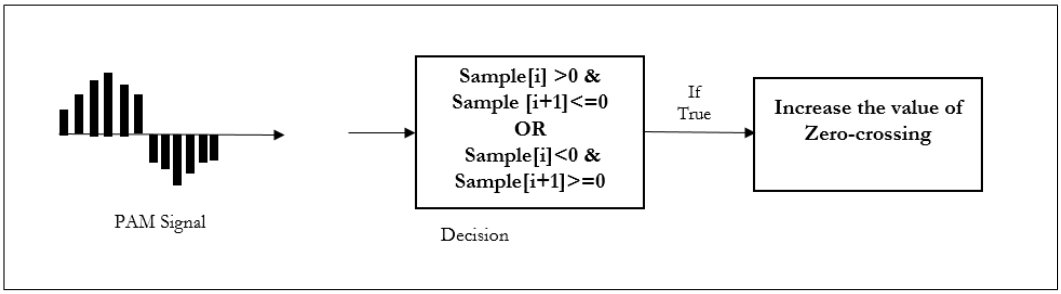

Figure 3.4: Block diagram for zero-crossing detection

Determining the Frequencies

The equation used to find the frequency is

$$f = \frac{number \quad of \quad cycle}{sample \quad rate}$$

$$number \quad of \quad cycle = \frac{number \quad of \quad zero - cross}{2}$$

The value of frequency calculated in receiver side will be different from that is transmitted. As, we have a significant difference between two frequencies f_1 and f_2, we can tell the decoder to consider 20% above or bellow the f_1 and f_2 value to detect as 0 and 1. So, the threshold for both the frequencies are,

$$Low \quad frequency \quad threshold \quad high = f_1 + \left[\frac{f_1 \times \tau}{100}\right]$$

$$High \quad frequency \quad theshold \quad low = f_2 - \left[\frac{f_2 \times \tau}{100}\right]$$

Here, τ is the amount of threshold percentage.

3.2.5 Binary to Data

After getting the frequency, the application maps these high and low frequencies to high and low bit which leads to an array of $0s$ and $1s$. This is the digital data which is then converted back to the character string.

3.3 Design Consideration

The sound generation and sensing capacity of an android mobile varies from device to device. Upon testing the capability of range of sound generation on various high end and medium capacity devices such as Samsung Galaxy $S7$, Samsung Galaxy Grand Prime, Samsung Galaxy ACE, Samsung $A5(2016)$, Motorola and few others it was found that most devices could generate sound wave starting from the bottom to fairly about 18 kHz to 22 kHz. However, the range varies more widely from device to device in case of detection. While some device hardware fabricated with good microphone can detect sound up to about 22 kHz, others can detect sound of roughly between 15 kHz to 20 kHz.
Hence, while selecting the frequencies for $0s$ and $1s$ the frequency range was kept within this range. In addition to this there is kept enough guard space between these frequencies so that the system does not make error while detecting these frequencies and generate wrong output.

Since the application might require working on a noisy environment. Hence, there is another parameter used known as the Root Mean Square(RMS) amplitude value which is taken from the RMS value of each sample of the received audio signal. The RMS value will determine the minimum level below which a sound will be considered as noise and will be ignored while staying in idle mode. It is kept 15 for 8-bit and 2000 for 16-bit PCM audio format. Some higher values above this will decrease the range of audio reception since most of the received sound will be considered as noise and ignored. But, it will decrease the probability of error while working in noisy environment. A lower value might increase the communication range but increases the probability of error as well since the application might consider an environmental noise as a valid signal.

3.4 Design challenges

The main challenge lies in how far the communicating devices can exchange data. Mostly two things affects the effective transmission range. Firstly, since the entire process is done in a fraction of second it is found that the processor, sound generator and microphone often becomes overwhelmed with the sudden burst of load and loses some data during the process. Secondly, the audio quality mostly depends on the quality of speaker and microphone of the devices. The transmission range can be increased by using quality speakers and microphone.

3.5 The Prototype Transmission Simulation

A prototype simulation was run on MATLAB to transmit data using the sound spectrum using ASCII as the coding scheme. The algorithm is very direct and primitive, each bit of data was represented by two frequencies to avoid errors transmitting two identical bits one after another. This method was similar to the coding scheme of DTMF. The detector would have to perform Fourier transform on the incoming sound signal and decode the information and hence there would be a successful transmission of the data using sound waves. Like the Morse code, but instead of a dot and a dash, each character was represented by a series of binary bits in ASCII language. For example, 1000001 is the ASCII code for the character `A`, therefore for transmitting the character each bit should produce sound waves with significantly different frequencies in order to avoid interference between them. If each 1s were given two different frequencies and each 0s were also given two separate frequencies, the detector would be able to decode the character with minimum error. However, this method would take a very long time to transmit a single character, transmitting a series of characters would take even longer. Therefore, this prototype was not developed further as in Android platform there is a faster alternative.

Chapter 4

Outcome of the Thesis

4.1 Test results and Performance of the Application

While putting the application in test several parameters are taken into consideration. The performance of the application is marked varying the values of the distance from the sender to receiver, Root Mean Square (RMS) value of the samples from the received audio signal which determines the minimum level of sound of within certain frequency below which it will be considered as environmental noise, various frequency range for $0s$ and $1s$ and their overall corresponding average error rate taken from a repetitive number of tests. The tests are repeated in both noisy and noiseless environment to find its performance in the presence of noise compared to a noiseless atmosphere. The test is performed between a Motorola G as sender and a Samsung Galaxy Grand Prime as receiver. The test results are shown in Table 4.1 and 4.2

Table 4.1: Application performance with minimal closed room noise

RMS Amplitude	Distance	% of accuracy
10	Point to point	100%
	2 inch	100%
	3 inch	95%
	4 inch	90%
	5 inch	60%
	6 inch	below 10%
15	Point to Point	100%
	2 inch	100%
	3 inch	95%
	4 inch	90%
	5 inch	50%
	6 inch	below 20%
30	Point to Point	95%
	2 inch	90%
	3 inch	80%
	Greater than 4 inch	below 10% or no signal received

Table 4.2: Application performance in presence of loud music as noise

RMS Amplitude	Distance	% of accuracy
10	Point to Point	100%
	2 inch	95%
	2.5 inch	90%
	3 inch	80%
	4 inch	Below 10%
15	Point to Point	100%
	2-2.5 inch	90%
	3 inch	80%
	4 inch	Below 10%
20	Point to Point	100%
	2-2.5 inch	90%
	3 inch	80%
	4 inch	50%
	4.5 inch	Below 20%
	5 inch	No signal received
30	Point to Point	95%
	2-2.5 inch	70%
	Less than 3 inch	Below 20% or no signal received

Furthermore, in order to understand the effect of sound quality and am-

plitude of the signal, the same test is carried out once more using a good quality Microlab speaker with higher gain in the presence of small noise. The results are shown in Table 4.3

Table 4.3: Application performance with a quality speaker

RMS Amplitude	Distance	% of accuracy
20	Point to point	100%
	2-5 inch	90-95%
	6 inch	Below 40%
	7 inch	Below 10%

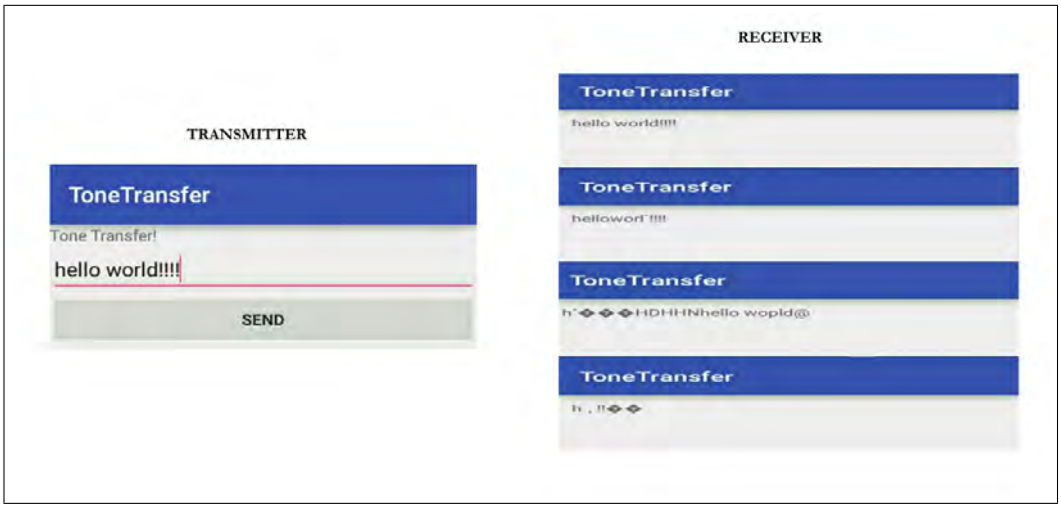

Figure 4.1: Various levels of accuracy of the received data

Figure 4.1 shows the different levels of accuracy of the received signal at different distances. Here, the first screenshot of the output on the receiver shows fully accurate output and the lower ones represent the degrading output with more error with increasing distance between the sender and the receiver.

The test is further carried out varying the frequencies for $0s$ and $1s$. The value of frequencies for 0s used are 882, 1575, 3150 and 4900 Hz. The value of frequencies for their corresponding 1s are 1764, 3150, 6300 and 7350 Hz respectively. These values of frequencies could not be exceeded to a much higher or lower value beyond human audibility range because of the

hardware limitations of the android mobile platform. However, it is found that there is no significant change in accuracy or range of communication due to the change of value of these frequencies. But in order to avoid error it is recommended to keep a good distance between the two frequencies. In Figure 4.2, the top red peaks indicate two frequencies for high and low bit in frequency domain from the sound generated to send a data. The first peak at approximately 4900 Hz is representing 0 or low bit and the peak at 7350 Hz represent 1 or high bit.

Figure 4.2: Received signal in frequency domain (Generated by Spectrum Analyze application)

4.2 Key Findings

There are several important key findings from the test results mentioned above. Firstly, we can find an inverse relation between the RMS value and the range of communication with percentage of accuracy. It is seen that on a noise less environment a lower value of RMS will accept more audio signal as valid one and give proper output on a larger distance without much of any error. However, when this value is excessively high (like 30), it also blocks the valid signal as noise and limits the range of communication. But while on a noisy environment the RMS value can filter out some good amount of environmental noise and decrease the percentage of error with respect to distance. However, just like before an excessive value of RMS will also limit the range of communication to a significantly lower distance (3 inch). Hence, an optimal value of 15 is recommended for the exact hardware and environment.

Secondly, the performance of this algorithm is highly dependent on the hardware of the platform that it is implemented upon. Comparing the second and third chart of the above-mentioned application performance test it can be seen that distance of communication has increased to almost 40% with significantly lower rate of error on the output. Hence it can be concluded that since the system is largely dependent on the quality of the hardware and amplitude of the signal, a good quality speaker and sensitive microphone can increase the system performance significantly. Also, with much larger sound from a bigger audio transmitter, the range of communication can be increased manifold.
Thirdly, the application completes generating an audio signal of a sample data such as **hello world!!** consisting 58 characters within one second. On the receiving end the processing takes about 1.2 seconds. So, the time taken to send and receive the entire data is not more than 2.2 seconds. Hence the combined data transmission and reception rate is 26 characters per second approximately. A better processor can decrease the processing time. Hence, the data transmission and reception with sound wave is fast enough for small scale data transmission.
Besides, since the frequency values do not affect much in the communication performance, this algorithm can be further implemented on a platform with audio generators and receivers which can operate beyond human audibility range to avoid any noisy situation where constant data transmission and reception is required to maintain communication between machine to machine for small scale data.

Chapter 5

Conclusion and Future Work

5.1 Concluding Remarks

Smart cities are an urban vision to integrate information and communication technology in order to control assets of a city in a secure fashion [16]. In this thesis, we developed an algorithm for the development of an Android application for small scale data transmission using audio waves, this is a step forward on the evolution of machine to machine communication and may act as a foundation for building smart cities.As we emerge into the future, the number of connected devices increases and the need for cross-platform solution for data communication becomes more crucial as assets of the smart cities such as smart cars, transformers, printers and several other devices need to be controlled by other machines, which can be achieved by an interoperable communication system.

5.2 Future works and Development Opportunities

Although the thesis was based on developing the algorithm and testing the performance on android platform, there are still several important opportunities for further development to get a full-scale output of the entire technology. Since machine to machine communication using sound waves might be required on a large scale in the future, it might create noisy situation. Hence, devices can be developed which are capable of operating beyond human audibility range. Another solution to the problem can be using the sound of musical instruments or birds chirping so that it does not sound annoying.
If in the future many smart devices and autonomous machines start using

this technology, they might create interference among themselves if operating under the same room and same frequency. Hence, a sensing algorithm can be added to sense on what frequency range other devices are operating on to avoid interference. Besides, a sharp filtering algorithm can be added to the existing one to lessen the chances of error in the output and hence increasing the effective communication range.

References

[1] Rappaport, T. S. (1996). *Wireless communications: principles and practice* (Vol.2). New Jersey: Prentice Hall PTR.

[2] Starsinic, Michael. (2010, May). System architecture challenges in the home M2M network. In*Application and Technology Conference (LISAT), 2010 Long Island Systems* (pp. 1–7). IEEE

[3] Chase, J. (2013). The evolution of the internet of things. *Texas Instruments.*

[4] Henderson, P. M. (1997, November). 56 Kbps data transmission across the PSTN. How does it work?. In *Wescon/97. Conference Proceedings* (pp. 352-365). IEEE.

[5] Tan, L., Wing, N. (2010, August). Future internet: The internet of things. In *2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)* (Vol. 5, pp. V5-376). IEEE

[6] Forouzan, B. A. (2006). Data Communications. In *Data communications networking* (pp. 3–7). New Delhi: Tata McGraw-Hill Education.

[7] Forouzan, B. A. (2006). Data and Signals. In *Data communications networking* (pp. 56–80). New Delhi: Tata McGraw-Hill Education.

[8] Rappaport, T. S. (1996). Modulation Techniques for Mobile Radio. In *Wireless communications: principles and practice* (Vol.2, pp. 255–346). New Jersey: Prentice Hall PTR.

[9] Proakis, J. G., Manolakis, D. G. (2006). Analog-to-Digital and Digital-to-Analog Conversion. In *Digital Signal Processing* (pp. 19–21). New Delhi: Prentice-Hall of India.

[10] Logan, B. F. (1977). Information in the zero crossings of bandpass signals. *Bell System Technical Journal*, 56(4), 487-510.

[11] Sensors Overview. (n.d.). Retrieved from `https://developer.android.com/guide/topics/sensors/sensors_overview.html`

[12] Milette, G., & Stroud, A. (2012). Inferring Information From Physical Sensors. In *Professional Android sensor programming* (pp. 65–189). John Wiley & Sons.

[13] Forouzan, B. A. (2006). Analog-to-Digital Conversion. In *Data communications networking* (pp. 120–122). New Delhi: Tata McGraw-Hill Education.

[14] Lathi, B. P. (2009). Pulse-Code Modulation. In *Modern digital and analog communication systems* (pp. 262–263). Oxford University Press, Inc..

[15] Rico, G. (2006). Tech Tip: Effective or RMS Voltage of a Sinusoid. *the Technology Interface*, 6(1)

[16] Jin, J., Gubbi, J., Marusic, S., & Palaniswami, M. (2014). An information framework for creating a smart city through internet of things. *IEEE Internet of Things Journal, 1(2)*, 112–121.