

EMERGENCY ALERT MOBILE APPLICATION FOR DEAF

By

Nur Fatimah Binti Hairuddin

Dissertation submitted in partial fulfillment of
the requirements for the
Bachelor of Technology (Hons)
(Information & Communication Technology)

MAY 2011

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Emergency Alert Mobile Application for Deaf

By

Nur-Fatihah Binti Hairuddin

A project dissertation submitted to

Information and Communication Technology Programme

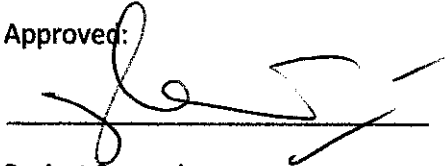
Universiti Teknologi PETRONAS

In partial fulfillment of the requirements for the

BACHELOR OF TECHNOLOGY (Hons)

(INFORMATION AND COMMUNICATION TECHNOLOGY)

Approved:



Project Supervisor

(NAZLEENI SAMIHA HARON)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

MAY 2011

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



Nur Fatimah Binti Hairuddin

ACKNOWLEDGEMENT

First and foremost, I would like to put all my gratitude to Allah s.w.t. for giving me strength and pink of health to complete this project on the specific time. Not to excluded, thank you to my parents, family and friends for being supportive and giving advices to help on making this project successful.

Personally, I would like to thank my supervisor, Ms Nazleeni Samiha Haron for her encouraging me to complete this project and all the support and guidance throughout this project. Added, I also want to thank the entire lecturer participating during this project timeline.

Next I also like to put my appreciation to Universiti Teknologi PETRONAS in particular Information and Communication Technology Department for providing me with important training classes and resources to accomplish my task and project. This also could enrich my knowledge for future use.

Finally, as a student and a person, I thanks to all individual who contribute towards my project either directly or indirectly.

ABSTRACT

Emergencies happen around us are unexpectedly in various forms which is fire, flood, earthquake or accidents. In emergencies, up-to-date information is life-saving. However, this information is not accessible to deaf and hard of hearing people. The current problem now is deaf and hard of hearing people face some difficulties on how to detect any alert sound and how to responds to the emergency. The study is comprehensive by focusing and highlighting on core and basic process of voice and sound recognition which covered from the root operations to the final production. Prior to the requirement of Emergency Alert Mobile Application for Deaf, the study also covered the description of tool development that will be used which is Eclipse IDE and also the other elements and parts of programming that could contribute directly to the project such as Android Development Tools (ADT) and Android API (Application Programming Interface). The tool is well recommended by experts and professionals as project development tool due to its good compliance and compatibility with most universal common current operation system on mobile which is Android Operating System. As you can see, an emergency alert device is not only useful for protecting the safety of our loved ones, but can also be economical. After all, one false alarm resulting in a visit to an emergency room can cost thousands of dollars. That alone makes an emergency alert device completely worth it. This project is to help the deaf people to alert with any emergency alarm or alarm trigger.

TABLE OF CONTENTS

| | PAGE |
|---|------|
| CERTIFICATION OF APPROVAL | ii |
| CERTIFICATION OF ORIGINALITY | iii |
| ACKNOWLEDGEMENT | iv |
| ABSTRACT | v |
| LIST OF FIGURES | ix |
| LIST OF TABLES | ix |

CHAPTER 1 : INTRODUCTION

| | | |
|------------|---------------------------------------|---|
| 1.1 | Background of study..... | 1 |
| 1.2 | Problem statements..... | 2 |
| 1.3 | Project Significant..... | 3 |
| 1.4 | Objectives..... | 3 |
| 1.5 | Scope of study..... | 4 |
| 1.6 | Relevancy of project..... | 4 |
| 1.7 | Feasibility analysis..... | 5 |
| | 1.7.1 Technical Feasibility..... | 5 |
| | 1.7.2 Scope Feasibility..... | 5 |
| | 1.7.3 Organizational Feasibility..... | 6 |

CHAPTER 2 : LITERATURE REVIEW

| | | |
|------------|---|----|
| 2.1 | Sound Capture and Processing..... | 7 |
| | 2.1.1 Definition of sound capture and processing..... | 7 |
| | 2.1.2 The operations of sound capture and processing..... | 9 |
| 2.2 | Microphones..... | 10 |
| 2.3 | Microphones as a sensor..... | 11 |
| 2.4 | Eclipse IDE..... | 12 |
| | 2.4.1 Definition of Eclipse IDE..... | 12 |
| | 2.4.2 Android Development Tools (ADT)..... | 13 |
| | 2.4.2.1 Definition of ADT..... | 13 |
| | 2.4.3 Android Emulator..... | 14 |

| | | |
|-------|---|-------|
| 2.4.4 | Android Debug Bridge..... | 15 |
| 2.4.5 | Android Interface Description Language..... | 15 |
| 2.5 | Media API..... | 16 |
| 2.6 | Fast Fourier Transform (FFT)..... | 16 |
| 2.7 | Existing system..... | 18 |
| 2.8 | Main concept of project..... | 19 |
| 2.9 | Latest Research..... | 19 |
| 2.10 | Applications of speech recognition..... | 20-22 |

CHAPTER 3 : RESEARCH METHODOLOGY

| | | |
|-------|--------------------------------|-------|
| 3.1 | Methodology..... | 23-24 |
| 3.2 | Project Activities..... | 24-25 |
| 3.2.1 | Description of phase..... | 25-27 |
| 3.3 | Project main interface..... | 27 |
| 3.3.1 | Capturing sound..... | 28-29 |
| 3.3.2 | Coding – Layout..... | 29-30 |
| 3.3.3 | Coding – Manifest file..... | 30 |
| 3.3.4 | Coding – Sound Processing..... | 30 |
| 3.3.5 | Coding – Alert Dialog..... | 31 |
| 3.3.6 | Coding – Vibration..... | 31 |
| 3.4 | Interface Prototype..... | 31 |
| 3.5 | System Architecture..... | 32 |
| 3.6 | Flow Chart..... | 33 |
| 3.7 | Tools Required..... | 34 |

CHAPTER 4 : RESULT AND DISCUSSION

| | | |
|-----|-----------------------------|-------|
| 4.1 | Result..... | 35-36 |
| 4.2 | Discussion..... | 36 |
| 4.3 | Fast Fourier Transform..... | 37 |
| 4.4 | Signal Frequency..... | 37-38 |

CHAPTER 5 : CONCLUSION AND RECOMMENDATION

| | | |
|------------|----------------------------|----------------|
| 5.1 | Conclusion..... |39 |
| 5.2 | Recommendation..... |40 |

| | |
|---|--------------------|
| References & Appendices..... |iv-vii |
|---|--------------------|

LIST OF TABLES

| | | |
|-------------------|---|-----------|
| Table 2.10 | Typical applications in terms of speech recognition capabilities..... | 21 |
| Table 3.2a | Main Task..... | 24 |
| Table 3.2b | Sub Task..... | 25 |
| Table 4.1 | Result..... | 26 |

LIST OF FIGURES

| | | |
|--------------------|--|-----------|
| Figure 2.1 | Example of sound processing..... | 8 |
| Figure 2.3 | Processing Pipeline..... | 12 |
| Figure 2.6 | Samples of FFT..... | 17 |
| Figure 2.7 | Deaf device..... | 18 |
| Figure 3.1 | RAD Model..... | 23 |
| Figure 3.2a | Sekolah Pendidikan Khas,Ipoh, Perak..... | 26 |
| Figure 3.2b | Survey Result..... | 26 |
| Figure 3.3a | Main interface..... | 27 |
| Figure 3.3b | Capturing sound..... | 28 |
| Figure 3.4 | Interface prototype..... | 31 |
| Figure 3.5 | System Architecture..... | 32 |
| Figure 3.6 | Flow chart..... | 33 |
| Figure 4.1a | Eclipse emulator error..... | 35 |
| Figure 4.1b | Result running on mobile..... | 36 |
| Figure 4.3 | Sample signal..... | 37 |
| Figure 4.4a | Ambulance sound waves..... | 37 |
| Figure 4.4b | Police siren sound wave..... | 38 |

CHAPTER 1

INTRODUCTION

1.1 Background of study

Deep study focuses and emphasizes on the main fields that relate to the project which are software engineering. Prior to its root algorithm process in 'sound capture and processing' with the adequate support from software engineering knowledge and expertise that heavily rely on programming language tool application, Eclipse IDE may produce the series of brilliance and excellence results of project development. Today, masses of mobile devices are being used as digital assistants, for communication, networking, and business purposes or simply for daily usage. Examples are PDAs, MP3 players, mobile phones, digital cameras, GPS devices and the like. Mobile devices are characterized as having limited computational power, memory size and battery life, whereas state-of-the-art sound capture and processing systems are computationally rigorous. Over the past decade, these areas have undergone substantial development. The crucial part of the technology on sound capture and processing system is to process the output at real time basis and in background process. This is to ensure the output will be synchronizing with the input in real time system. The purpose of this paper is to focus on sound capture and processing system implemented on mobile phones especially Android platform for emergency alert focusing for deaf people or hard of hearing needs. The current technology is using an external device which causes some difficulties towards the deaf. The main concept of "Emergency Alert Mobile Application for Deaf" is to capture an emergency sound using the build in microphone in the mobile phone and alert the end user by displaying text output and vibrate the phone. Therefore, the end user whom is deaf or hard of hearing will be able to react by noticing the emergency alert.

1.2 Problem statements

The existing problem now to the author's knowledge is there are no mobile applications to help the deaf or hard of hearing people to be alert on any emergency sound surround them.

In this ubiquitous computing environment, the use of keypad, stylus and small screen is inconvenient and speech-centric user interface is foreseen to be a desirable interaction paradigm where sound capture and processing is the enabling technology. This has led to the growing interest in deploying sound capture and processing on mobile devices.

Nowadays, the existing technology could support the sound capture and processing system to operate on mobile devices. However, the research is still being done in order to improve the usage of the sound capture and processing system on mobile devices.

The needs of new applications to alert deaf or hard of hearing using sound capture and processing system is very importance for current and future needs in order to ease and alert the deaf or hard of hearing to notice any emergency call such as fire alarm or ambulance and able for them to respond towards the alarm trigger.

Current external device for deaf uses for emergency alert is quite expensive to purchase. Not all deaf affordable to buy it especially for students and it is hard to bring everywhere since it should be charge with the required charger.

1.3 Project Significant

Therefore, by having a new application development that consist of sound capture and processing system process on mobile phone will ease and solve many existing and current obstacles, constraints and problems which might produce and provide a thread of consequence output for deaf or hard of hearing people.

Benefits of project:

- The end user will be able to alert with emergency call
- The end user will be able to respond with the situation and environment that needs them to cooperate
- Enable the end user to use it anywhere and anytime as long as the application is installed.

1.4 Objectives

- To develop a mobile application on Android platform for deaf to be alert with emergency alarm in surrounding area
- To design a mobile application using sound capture and processing for deaf to alert with emergency alarm such as fire alarm

1.5 Scope of study

Sound capture and processing system will cover the main aspects of scope of study. With the core focus is about sound capture and processing process which the method is applicable in mobile devices (Android platform). This process can be declared as the core foundation and backbone for this application development. The study will go detail and specific about the basic operation of sound capture and processing system on mobile devices and combines knowledge and familiarity with the Java programming language that been used in mobile devices.

Another scope of study is about programming tool which is Eclipse IDE. The study will highlight on the tool functions and abilities that might can be used to develop a mobile device application that could contain sound capture and processing system in one single application. If the previous 'Progress Report' emphasized on the general description of the tool, thus this 'Dissertation' will go beyond than that and focus more on functionalities of Eclipse IDE itself in order to develop and complete the project.

Major scope of study is deaf people centric by studying the behaviour of deaf regarding emergency matters through surveys and questionnaires. Through the result, this project could benefit them by reducing burden for emergency purposes.

1.6 Relevancy of the Project

Although the terms sound capture and processing system are totally new for the scope of the author's field in degree programme study which is Information Communication and Technology but they are classified in software engineering and related to Information Communication and Technology (ICT) area.

The relevancy points of this project are:

- This project is about mobile application using sound capture and processing system in mobile devices (Android platform).
- Sound capture and processing system is under area of software engineering (in ICT expertise area).
- The knowledge of sound capture and processing system with the Android platform are related and complement to each other.

1.7 Feasibility Analysis

1.7.1 Technical Feasibility

- Familiarity with Application (Eclipse IDE): More familiar
- Familiarity with Technology (Sound Capture and Processing System):
Less familiar
- Project Size
 - ◆ People: Nur Fatimah Binti Hairuddin
 - ◆ Timeframe: 28 weeks (February 2011 - August 2011)

1.7.2 Scope Feasibility

- This project development study focuses on the aspects of mobile application and software engineering.
- The study will be conducted and covered within the topics of sound capture and processing system, technology of Eclipse IDE, JAVA and Android Development Tools (ADT) development (both are the elements of Eclipse IDE).
- The project would be operated and run in both phases of 'Final Year Project 1' and 'Final Year Project 2', the planning, analysis and design phases would be done in 'Final Year Project 1'.

- The testing and implementation phases are following in 'Final Year Project 2'. These indicate that the progression of development and deep studies including researches of the project must be done during 'Final Year Project 1' and the software must be delivered during 'Final Year Project 2'.

1.7.3 Organizational Feasibility

- Project coordinator and supervisor: Ms. Nazleeni Samiha Binti Haron, Coordinator for Final Year Project from Department of Computer Science and Information of Universiti Teknologi PETRONAS are responsible to monitor the progress of this project by providing consultation and expertise in software engineering area.
- The end users of the system, deaf or hard of hearing people are expected to appreciate the future benefits of Emergency Alert Mobile Application for Deaf.

CHAPTER 2

LITERATURE REVIEW

The literature review is focused and described on sound capture and processing system by going deep research to the basic and root operations of it. Another focus lie on Eclipse IDE that would be served as the major attention of study focus, work development flow and work spine. With these main two contents as the backbone of this report, it would be the strong points in Final Year Project 2 as part of the project development phase before the project move on to the next phase.

2.1 Sound Capture and Processing

2.1.1 Definition of Sound Capture and Processing

Sound is a mechanical wave that is a fluctuation of pressure transmitted through a solid, liquid, or gas, composed of frequencies within the range of hearing and of a level sufficiently strong to be heard, or the sensation stimulated in organs of hearing by such vibrations. Sound capture and audio processing in general stayed in the analog signal processing domain. The first programmable digital computers were designed and researchers started to work on digital signal processing algorithms. Initially communications were the major consumer of signal processing algorithms such as echo cancellation and digital speech compression (M. R. Schroede, 1999). In the mean time, digital computers become more powerful with more memory and faster processors. They invaded offices and homes and far exceeded their initial role as a tool for increased productivity for information workers. Modern computers are communications and entertainment centres many of them having attached or integrated loudspeakers, microphones and web camera. They are used for storing and playing music and videos. Programs for audio and video chat are widely used. Sound capture and processing algorithms today are an integral part of every personal gadget.

The behavior of sound propagation is generally affected by three things:

- A relationship between density and pressure. The relationship, affected by temperature, determines the speed of sound within the medium.
- The propagation is also affected by the motion of the medium itself. For example, sound moving through wind. Independent of the motion of sound through the medium, if the medium is moving, the sound is further transported.
- The thickness of the medium also affects the motion of sound waves. It determines the rate at which sound is attenuated. For many media, such as air or water, attenuation due to viscosity is negligible. Figure 2.1 show example of sound processing.

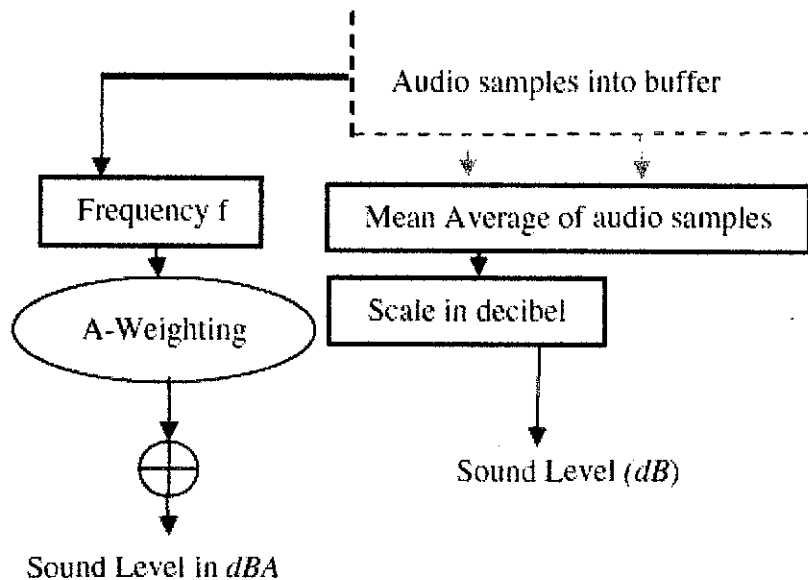


Figure 2.1 Example of sound processing

Mobile phones for the first time took the phone out of quiet rooms and exposed the microphones to substantially higher noise levels. This increased the demand for real time implementations of noise suppression and speech enhancement algorithms running on inexpensive processors (Nicholas D. Lane, 2009). Processing methods and application areas include storage, level compression, data compression, transmission, enhancement (equalization, filtering, noise cancellation, echo or reverb removal or addition).

2.1.2 The Operations of Sound Capture and Processing

Capturing sounds starts with converting the acoustic wave in the air to an electrical signal by one or more microphones. These microphones can have very different characteristic and if properly designed, selected and positioned can provide a substantially better sound for the next processing steps. The microphone signals are amplified, filtered and pre-processed. An analog-to-digital converter performs discretization and quantization and converts the sound into a stream of numbers (M. Snover, 2006). This is where the digital signal processing starts. Below is the example list of some of its functionalities:

- **Microphone** – capture sounds.
- **Sound** – data input.

Bandwidth (Tanzeem Choudhury and Andrew T. Campbell, 2009) is the width of the range of the frequencies that the signal occupies. It makes use of the SC value and shows the spectrum is concentrated around the centroid or spread out over the whole spectrum and express by:

$$BW_f = \frac{\sum_{i=1}^n (i - SC_f)^2 \cdot p(i)^2}{\sum_{i=1}^n p(i)^2}$$

Most ambient sound consists of a limited range of frequencies, having a small value. Music often consists of a broader mixture of frequencies than voice and ambient sound. The word “frequency” for signals was defined in both the continuous and the discrete-time domain. To define a Fourier transform for functions of a discrete variable. Here we can re-express such definition, as a function of frequency, for discrete-variable functions obtained by sampling continuous-time signals with sampling interval (Davide Rocchess, 2003). This transform is called the Discrete-Time Fourier Transform (DTFT) and is expressed by

$$Y(f) = \sum_{n=-\infty}^{+\infty} y(nT) e^{-j2\pi \frac{f}{f_s} n}$$

2.2 Microphones

It is crucial to consider the built-in microphone of mobile phones as a generic sensor to be used for mobile performance (Wong, G.S. and Embleton, T.F., 1995). One reason for using microphones is that they are integral to any mobile phone, no matter how basic. It seems natural to integrate microphones into mobile phone performance as well. There are several types of Microphones existing nowadays. The types are:

- Carbon Microphones
- Electrodynamics Microphones
- Miniature electro-mechanical system microphones (MEMS microphones)
- Condenser microphones

Recently, miniature electro-mechanical system microphones (MEMS microphones) have become popular. They are manufactured from a silicon crystal with the same technology (Andrew T. Campbell, 2009) used for manufacturing integrated circuits. This technology allows the formation of a diaphragm and holding it at a certain distance from the base.

Conversion of the movements of the diaphragm to an electrical signal is based on varying the capacitance between the diaphragm and the base (Hong Lu, Wei Pan, 2009). That is, MEMS microphones are condenser microphones. The advantages of MEMS microphones are their small size (3 x 3 x 1 mm), the same packaging as surface mounted components, and lower manufacturing tolerances (important for their use in microphones array). In addition, the accompanying electronic circuitry can be integrated into the microphone chip using the same manufacturing process. The main disadvantages of MEMS microphones are still a higher price and higher self noise. The last is due to the smaller size of the diaphragm.

** Please refer to the Appendix 1 for the picture of Microphones*

2.3 Microphones as a sensor

Mobile Operating Systems are still in a process of maturation, which adds some complications to the development of applications for android platform. The complete architecture can be seen in Figure 2.3. The core to make this possible is allowing recording audio from the microphone. Then the microphone data is processed. The processed data can either be used directly as output or as input for parametric synthesis algorithms.

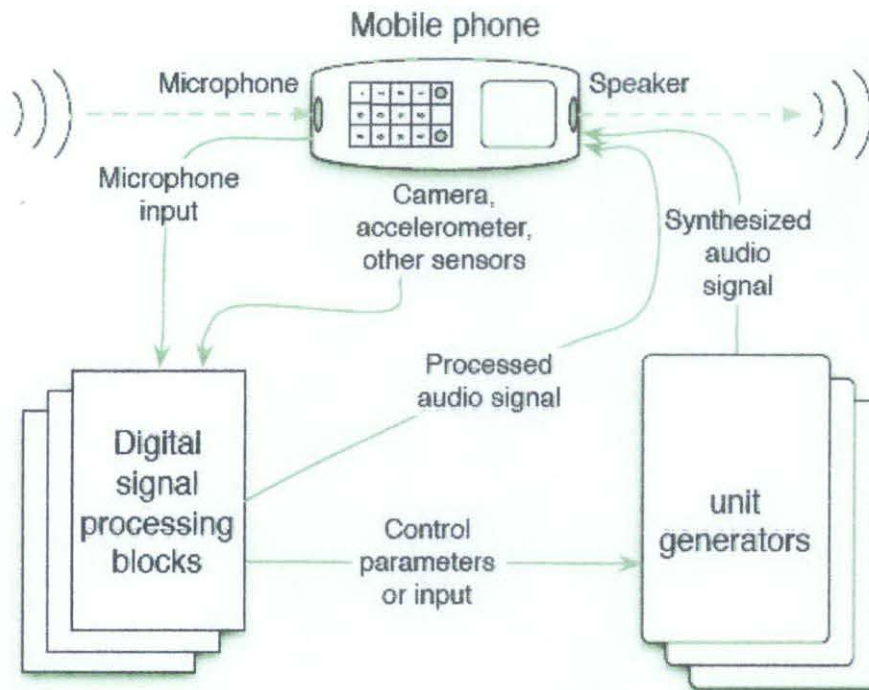


Figure 2.3 Processing pipeline: Audio received by the microphone passes through digital signal processing units. The output of these can be sent directly to the speaker for playback, or act as control parameters or input to unit generators.

2.4 Eclipse IDE

2.4.1 Definition of Eclipse IDE

Eclipse IDE is an open source community, whose projects are focused on building an extensible development platform, runtimes and application frameworks for building, deploying and managing software across the entire software lifecycle (Reto Meier, 2009). Many people know Eclipse IDE, as a Java IDE but Eclipse is much more than a Java IDE.

The Eclipse open source community has over 200 open source projects. These projects can be conceptually organized into seven different "pillars" or categories:

1. Enterprise Development
2. Application Frameworks
3. Service Oriented Architecture (SOA)
4. Embedded and Device Development
5. Rich Client Platform
6. Rich Internet Applications
7. Application Lifecycle Management (ALM)

2.4.2 Android Development Tools (ADT)

2.4.2.1 Definition of Android Development Tools (ADT)

Android Development Tools (ADT) is a plugin for the Eclipse IDE that is designed to give developer a powerful, integrated environment (Reto Meier, 2009) in which to build Android applications. ADT extends the capabilities of Eclipse to let the user quickly set up new Android projects, create an application user interface (UI), add components based on the Android Framework API, debug the applications using the Android SDK tools, and even export signed (or unsigned) *.apk* files in order to distribute the application.

Eclipse is most likely one of the best IDE development tools for Java based development. As an open source project, there are various plugins available to make Eclipse work with a variety of platforms and languages (William Bridges, 2009). The Android SDK includes a plugin for the Eclipse environment for developing Android applications.

The Android Development Tools Plugin for Eclipse (ADT) provides a number of shortcuts to other tools that make any Android application development move much more smoothly. When developer creates a new project in Eclipse, the ADT will generate the necessary stub files and layout the source files to build the application. Running or debugging in Eclipse will also automatically launch the emulator to allow the user to test the application on an emulated hardware profile (Daniel Ulery, 2009). The ADT even includes tools for packaging and distributing the Android application. Developer can literally do it all from within Eclipse by using this plugin.

2.4.3 Android Emulator

The Android Emulator is probably one of the important tools included in the Android SDK. This is a QEMU-based application that emulates an Android mobile device. It allows the user to test the application on various hardware and system configurations without owning multiple Android mobile devices. The emulator even allows the user to run different versions of the Android system (Chris Webb, 2009) so the user can verify that the application will be compatible between differing Android versions.

Another really useful feature of the emulator is that developer can run more than one instance of it on the development machine. This allows the developer to simulate calling another Android phone or sending SMS to another Android phone. To emulate a voice call from one emulator to another the user simply enter the port number of the emulator the user want to call in the dialer application and press send. To find the port number of the emulator for the user to call, look in the title bar of the emulator window. The title bar will read "Android Emulator" and the port number will follow in parentheses.

2.4.4 Android Debug Bridge

Android Debug Bridge is used to debug the applications. This is a client-server tool which interfaces with a debugging daemon on each emulator instance. The ADB provides a number of commands for collecting and analyzing debug output from the emulator instances.

Another useful tool in the Android Debug Bridge is the logcat tool. Logcat allows the user to view and filter the various logs collected by the Android emulator. There are a number of additional options and filters the user can familiarize with the complete syntax (Jack Lewis, 2009). Logging output can become pretty rambling so the user will save a lot of time if the user knows what to look for and how to filter for it.

2.4.5 Android Interface Description Language

The Android Interface Description Language is a type of Interface Definition Language used on Android to marshall objects and pass them between two processes. This is how Android manages Interprocess Communication (IPC). It has a number of similarities to remote procedure calls used in Com and Corba. While it is definitely lighter weight than either Com or Corba, the marshalling code is still difficult to write (James D. Kramer and Happenstance Type-O-Rama, 2009). This is where the Android Interface Description Language is introduced. It allows the user to create an *.aidl* file in which the user creates an interface as a subclass of the Stub abstract class. The compiler then will generate the necessary Java code to create the interface. This greatly simplifies IPC programming for Android.

Though the software development kit (SDK) contains a number of other very constructive Android developer tools, these are the best Android developer tools in terms of streamlining development and creating value for the SDK itself. The ADT, Android Emulator, ADB, and AIDL simplify the development of Android applications.

2.5 Media API

Media API allows Mobile Processing sketches to play sounds on supported mobile phones. The Android platform offers built-in encoding/decoding for a variety of common media types (Lynsey Stanford, 2009), so that user can easily integrate audio, video, and images into the applications.

Audio capture from the device is a bit more complicated than audio/video playback, but still fairly simple:

1. Create a new instance of android.media.MediaRecorder using new
2. Set the audio source using MediaRecorder.setAudioSource(). You will probably want to use `MediaRecorder.AudioSource.MIC`
3. Set output file format using MediaRecorder.setOutputFormat()
4. Set output file name using MediaRecorder.setOutputFile()
5. Set the audio encoder using MediaRecorder.setAudioEncoder()
6. Call MediaRecorder.prepare() on the `MediaRecorder` instance.
7. To start audio capture, call MediaRecorder.start().
8. To stop audio capture, call MediaRecorder.stop().
9. When you are done with the `MediaRecorder` instance, call MediaRecorder.release() on it. Calling MediaRecorder.release() is always recommended to free the resource immediately.

2.6 Fast Fourier Transform (FFT)

Fourier Transform converts signals from a time domain to a frequency domain and is the basis for many sound analysis and visualization algorithms (X. Chen and A. L. Yuille, 2008). It converts a signal into magnitudes and phases of the various sine and cosine frequencies making up the signal. For example, taking the Fourier Transform of these 32 real valued points results in 32 complex valued points shown in Figure 1

from the formula also shown in Figure 1, and sample it at the 32 equally spaced points shown. Plotting only the magnitudes of the first 16 gives the result in Figure 2; the other 16 points are the mirror image due to the symmetry.

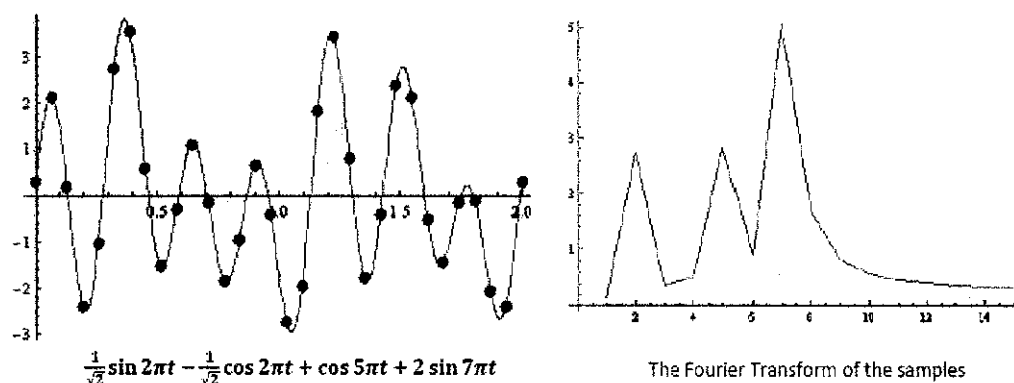


Figure 2.6 Samples for FFT

Given N input points, the Fast Fourier Transform (FFT) computes the Fourier Transform in $O(N \log N)$ steps. The majority of this note derives the FFT algorithm and shows how to implement it efficiently. These complex numbers represent the magnitude and phase of the various frequencies present in the x_k . A direct implementation of this requires adding N values for each of N array entries, for a complexity of $O(N^2)$. Given N (real or complex) samples $x_0, x_1, x_2, \dots, x_{N-1}$, the Fourier Transform of them are the N complex numbers $y_0, y_1, y_2, \dots, y_{N-1}$ given by

$$y_j = \sum_{k=0}^{N-1} e^{\frac{2\pi i}{N} jk} x_k, \text{ for } j = 0, 1, \dots, N-1$$

FFT algorithm consisting of the following steps:

1. Sort the inputs into bit reversed index order, which evaluates the eight $F_{j,1}$ transforms.
2. Evaluate four $F_{j,2}$ transforms on pairs from step 1.
3. Evaluate two $F_{j,4}$ transforms on data from step 2.
4. Evaluate one $F_{j,8}$ transform on data from step 3.

In many applications, such as sound processing, the x_k are real valued, which allows a performance and space improvement over the complex input to complex output FFT. When implementing the FFT in a many computer languages, complex numbers are stored as two floating point values, one for the real component and one for the imaginary component. To perform an FFT which expects complex inputs, the real valued input has to be augmented by an equal amount of space for the imaginary components, which takes additional time and space.

2.7 Existing System

Based on the deep research and finding, up to now, the Information Communication and Technology industry acquired the knowledge of sound capture and processes by using Android Development Tools (ADT) approach and method. This caused some drawback such as limited memory, power consumes of the mobile device and the most obvious one is the sound capture and processing is still under research. Current system that applicable for deaf or hard of hearing is on external devices which 'Emergency alert device for the deaf' which is costly and expensive refer to Figure 2.7. The most recent mobile application is the speech translation. However, the developer is not focusing on the needs of the deaf or hard of hearing based on mobile devices.



Figure 2.7 Deaf device

Another system that provide emergency alert to deaf is the Mobile Alerting Framework also provides a web interface allowing administrators to manually insert alerts into the system. It is internally stored in CAP format and published to a web server. In this form the alert is available for later retrieval by client software via mobile web access. After alerts have been matched with subscribers and a formatting transformation applied, they are ready for SMS delivery. Both the Alert Delivery and Subscription Management components use the Kannel open-source Wireless Application Protocol (WAP) and SMS gateway (<http://kannel.org>) to send and receive SMS messages.

2.8 Main concept of project

Referred to the mentioned weaknesses of the current technology of sound capture and processing for deaf or hard of hearing, the 'Emergency Alert Mobile Application for Deaf' is initiated to tackle and reduce the weaknesses. The main concept of this project is to capture an emergency sound using the build in microphone in the mobile phone and alert the end user by displaying text output and possible to vibrate the phone. Therefore, the end user whom is deaf or hard of hearing will be able to react by noticing the emergency alert.

2.9 Latest Research

There is rising interest in mobile phones research for pervasive computing and urban sensing applications (Brotherton, J. A, 2004). Although early work mentions the use of microphone as a clue to context, most work found in the literature focus on the camera, accelerometer, and GPS as sources of sensor data (X. Liu and J. Samarabandu, 2005). Existing work that considers problems such as sound recognition or sound recognition do not prove their techniques on resource limited hardware. One exception is which are focused on performing sound processing using wearable computers (J.G. Fiscus, 2007).

However, the authors collect data from wearable devices and perform offline analysis. Many developer uses a Hidden Markov Model (HMM) based strategy capable of classifying 10 environments with good performance using samples of only 3 second duration (G. Evermann and P. C. Woodland, 2000). This framework provides a way to build new classifiers. A different type of application also been considered. The ability to recognize a broad array of sound categories opens up interesting application spaces for example within the domain of participatory sensing (S. M. Lucas, 2005).

2.10 Applications of speech recognition

The increases applications developed for mobile phone has made the mobile phone more useful than the old time where mobile phone only was used for making calls or sending text messages. Some of applications regarding speech recognition are name dialing software, send messages using voice and message reader. Speech to text or dictation is fundamentally different in scope since the function is not basic to mobile phones originally. Dictation system exists for a long time for desktop and their performance is continuously improving.

However, these systems have been mostly successful in applications where dictation has already been established practice such as legal or medical domain automatic speech recognition (ASR) systems are more likely to succeed in the mobile environment, where there is a stronger motivation for users to adopt the new technology due to cumbersome traditional input mechanisms (A. Stolcke and E. Shriberg, 1996). The ASR for dictation may be fully implemented in the network with the speech transmitted over the wireless network by usual transmission techniques or be partly located in the mobile phone and partly in the network.

The basic technology relevant to the application mentioned above is isolated word recognition which means the capability of recognizing a single word to execute any process. Basically, isolated word recognition is useful for name dialing and command-and-control applications. Keyword spotting allow for a much more user friendly operation because the user is not required to speak isolated words. The speaker may speak natural phrases which contain dedicated keywords, the actual command. The speech recognizer separate the useful information from the non-useful information or as known as garbage. The vocabulary size can be kept still restricted as with isolated word recognition, up to 100 words.

Table 2.10 summarize the speaker independent (SI), speaker dependent (SD), and the mixer of both the speaker adaptive. Speaker dependent recognition is independent of languages, dialects and pronunciations. Some example of speaker dependent is name dialing by Dynamic Time Warping (DTW) and Hidden Markov Model (HMM) based (V. Levenshtein, 1998). The algorithm HMM-based speaker independent systems are studied to cope with various languages, dialects and speaker behavior. In order to achieve a good performance over a wide range of speaker samples taken from different speakers in different conditions are needed to pre-training (Schultz, Tanja & Kirchhoff Katrin, 2006).

The combinations of speaker independent and speaker dependent recognizers leverages the benefits of both system which is user-friendliness due to pre-trained vocabulary and high performance due to user-trained additions to the vocabulary. Furthermore, advanced speaker independent systems support on-the-fly adaptation of the acoustic models. These speaker adaptive systems maximize the accuracy of the system for user and environment variations while maintaining the low level of user interaction.

| | Isolated word | Keyword spotting | Continuous |
|--------------------------|--|---|--|
| Speaker independent (SI) | Basic digit or natural number dialing, basic command-and-control | Flexible command-and-control | SMS and/or Email dictation |
| Speaker dependent (SD) | Basic name dialing | High accuracy voice activation | - |
| Mix of SI and SD | Advanced digit and name (natural number and name) dialing | Flexible digit dialing, name dialing, and command-and-control | High accuracy SMS and/or email dictation |

Table 2.10 Typical applications in terms of speech recognition capabilities

In conclusion, developing sound capture and processing applications may be challenging because it rely on statistical techniques that usually require large training corpora for providing sufficient performance. This includes both textual and acoustic databases. For some sound the necessary sound resources are readily available. As in practical configurations, a set of sounds needs to be supported by a mobile device, several sound have to coexist in the limited memory space. Therefore, a suitable compact representation has to be developed. The vast advantage of microphone sensing in mobile devices is their wide availability. While accelerometers are only just emerging in contemporary high-end models of mobile devices (Nokia's 5500 and N95, Apple's, iPhone), microphones are available in any programmable mobile phone and offer signals of considerable quality.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Methodology

The 'Emergency Alert Mobile Application for Deaf' will implement 'Rapid Application Development –based' as the project methodology. The reason methodology is selected and chosen prior to the advantage of it is flexible and adaptable to change and it involves user participation thereby increasing chances of early user community acceptance.

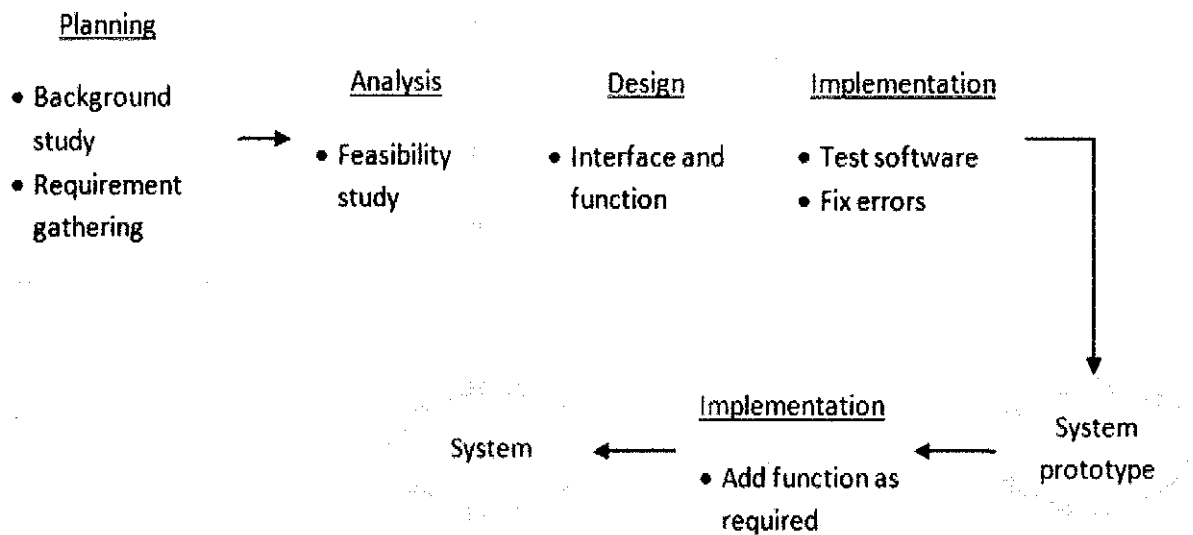


Figure 3.1 RAD Model

Besides that, while developing the project, the project risk can be realized and overall reduction. Due to users begin to work with the system faster than system completion, they are more likely to identify important additional requirements sooner than with sketch and structured design situations.

3.2 Project Activities

In order to fully development will be made within 'Final Year Project 1' and 'Final Year Project 2', the project activities are divided into two major classes of tasks which are main task and sub-task. The main task is consisted of:

| Main Task | Status |
|------------------|--------------------|
| Planning | <i>Completed</i> |
| Analysis | <i>Completed</i> |
| Design | <i>Completed</i> |
| Testing | <i>In progress</i> |
| Implementation | - |

Table 3.2a: Main Task

The sub-task is consisted of:

| Sub-Task | Status |
|---------------------------------------|--------------------|
| Proposal reporting | <i>Completed</i> |
| Extended proposal reporting | <i>Completed</i> |
| Interim reporting | <i>Completed</i> |
| Perform proposal defence presentation | <i>Completed</i> |
| Progress reporting (FYP 2) | <i>Completed</i> |
| Pre-EDX (FYP 2) | <i>Completed</i> |
| Dissertation (FYP 2) | <i>In progress</i> |
| Viva (FYP 2) | - |
| Technical reporting (FYP 2) | - |

Table 3.2b: Sub Task

3.2.1 Description of phase

First phase on RAD model is planning phase. This is where research and background study on the current project which is sound capture and processing is been done. The basic step on developing a project is to plan what to be done at certain period of time. At the end of this phase, all the required data and information regarding to sound and capture processing on mobile will be gathered to obtain and proceed to the next step. Gantt chart is prepared during this stage to observe what to be done at certain time to achieve what is planned (*refer Appendix 2*).

Based on the chart given, the second stage would be analysis for the project. Two main points in feasibility study as known as technical feasibility and scope feasibility. Java programming knowledge is crucial to develop this system and several tools that important to be familiar with such that Eclipse IDE, Android Developer Tools, and Android Emulator.

Analysis and survey had been implemented to gather information on current system being provided by the android developer and how the deaf people expectations towards the new system. The survey had been accomplished by five students from Sekolah Pendidikan Khas, Ipoh, Perak which is consist of five deaf students.



Figure 3.2a Sekolah Pendidikan Khas, Ipoh, Perak

Survey result:

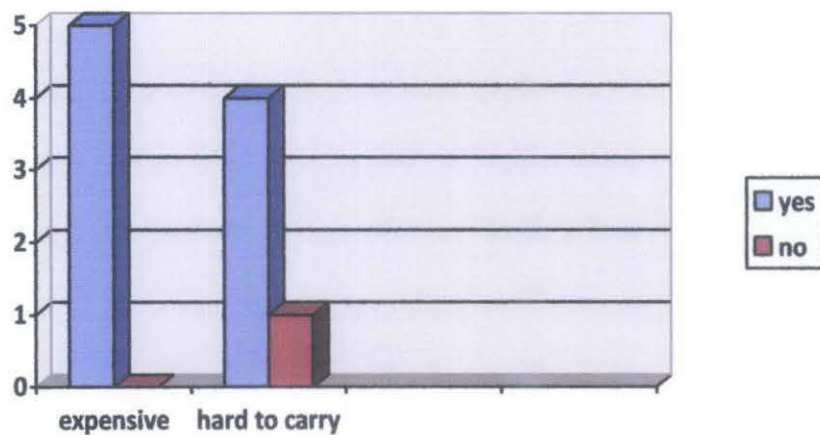


Figure 3.2b Survey Result

Due to suggestion and request by Ms. Nazleeni Samiha Binti Haron, the Final Year Project 2 will also include the design phase as it was planned to be accomplished in this semester. The design is about programming a simple prototype. The purpose of having the prototype is to give the clients about the ideas and opinions of further development since this project is based on from sketch.

In implementation phase, the list should include:

- ❖ The work is divided in modules or units and actual coding is started
- ❖ First developed in small programs called units
- ❖ Each unit is developed and tested for its functionality
- ❖ Unit testing verifies if the modules or units meet their specifications
- ❖ Units are then integrated into a complete system during Integration phase and tested

3.3 Project main interface

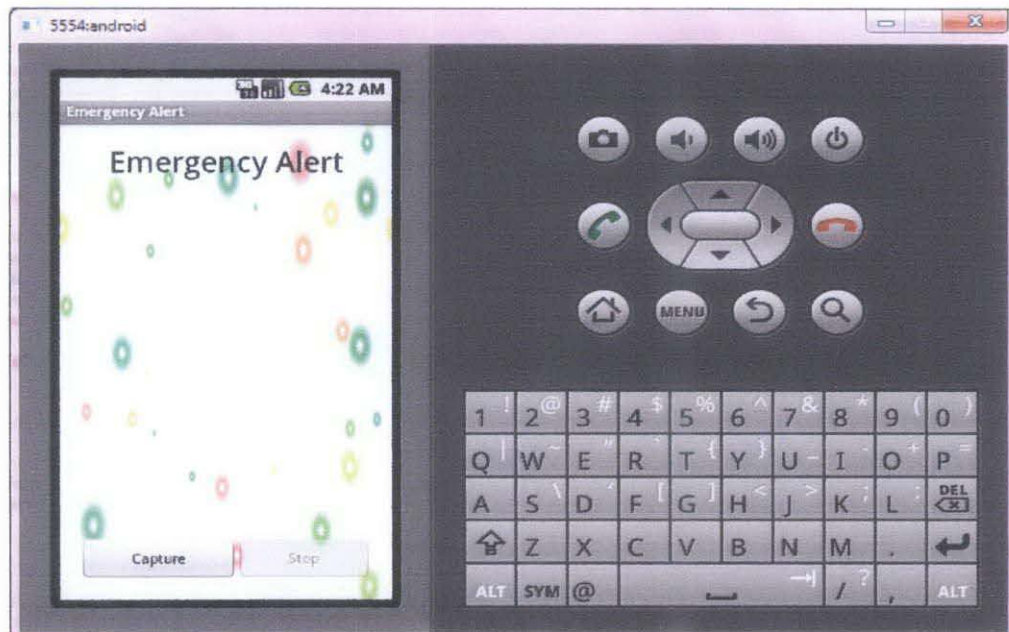


Figure 3.3a Main interface

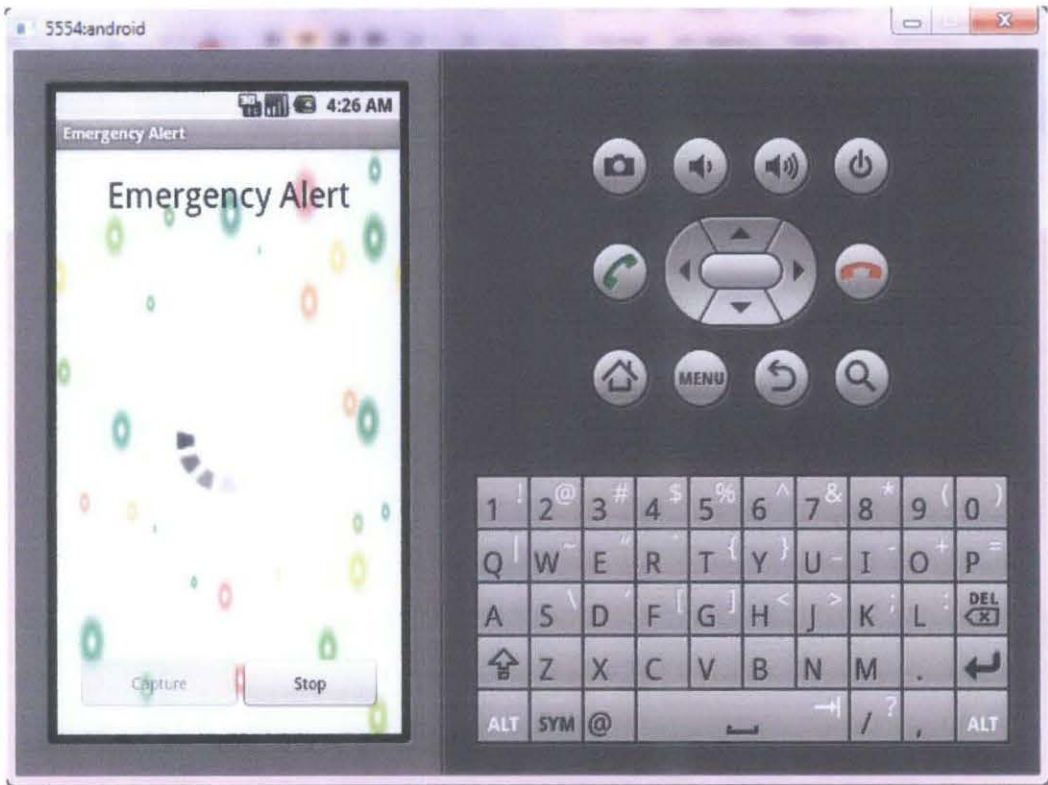


Figure 3.3b Capturing sound

3.3.1 Coding – Capturing sound

```
private void startRecording(){
    //Log.d("Start Recording","get input from mic");
    try{
        //bufferSize
        Log.d("bufferSize",Integer.toString(bufferSize));

        recorder = new AudioRecord(MediaRecorder.AudioSource.MIC,
            RECORDER_SAMPLERATE,
            RECORDER_CHANNELS,RECORDER_AUDIO_ENCODING, bufferSize );

        recorder.startRecording();
        Log.d("Start Recording","get input from mic");

        isRecording = true;

        recordingThread = new Thread(new Runnable() {

            // @Override
            public void run() {
                writeAudioDataToFile();
            }
        }, "AudioRecorder Thread");
    }
}
```

```

        recordingThread.start();
    }
    catch(Exception e){
        Log.e("error", "startRecording", e);
    }
}

private String getFilename(){
    String filepath = Environment.getExternalStorageDirectory().getPath();
    File file = new File(filepath,AUDIO_RECORDER_FOLDER);

    if(!file.exists()){
        file.mkdirs();
    }

    return (file.getAbsolutePath() + "/" + System.currentTimeMillis() +
    AUDIO_RECORDER_FILE_EXT_WAV);
}

private String getTempFilename(){
    String filepath = Environment.getExternalStorageDirectory().getPath();
    File file = new File(filepath,AUDIO_RECORDER_FOLDER);

    if(!file.exists()){
        file.mkdirs();
    }

    File tempFile = new File(filepath,AUDIO_RECORDER_TEMP_FILE);

    if(tempFile.exists())
        tempFile.delete();

    return (file.getAbsolutePath() + "/" + AUDIO_RECORDER_TEMP_FILE);
}
}

```

3.3.2 Coding – Layout

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@drawable/backgroundmain"
    android:layout_height="fill_parent"
    android:padding="20dip" android:orientation="vertical" android:layout_width="wrap_content">
    <RelativeLayout android:id="@+id/relativeLayout1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"

    android:gravity="center">
        <LinearLayout android:id="@+id/linearLayout1" android:orientation="horizontal"
        android:layout_width="fill_parent" android:layout_height="fill_parent">
            <Button android:layout_height="wrap_content" android:onClick="@string/start_recording"
            android:layout_weight="1.0" android:layout_gravity="bottom" android:layout_width="wrap_content"
            android:id="@+id/btnStart" android:text="@string/start_recording"></Button>
            <Button android:layout_height="wrap_content" android:layout_weight="1.0"
            android:layout_gravity="bottom" android:layout_width="wrap_content" android:id="@+id/btnStop"
            android:text="@string/stop_recording"></Button>
        </LinearLayout>
    </RelativeLayout>
</LinearLayout>

```

```

    <TextView android:text="Emergency Alert" android:id="@+id/textView1"
    android:layout_alignParentTop="true" android:textSize="30dip" android:layout_height="wrap_content"
    android:layout_width="fill_parent" android:gravity="center" android:textColor="#000000"></TextView>
    <RelativeLayout android:id="@+id/relativeLayout2" android:layout_height="wrap_content"
    android:gravity="center" android:layout_alignParentRight="true" android:layout_width="fill_parent">
        <ProgressBar android:id="@+id/progressBar1"
        android:layout_alignParentBottom="true"
        android:layout_height="100dip"
        android:layout_width="100dip"
        android:onClick="@string/start_recording"
        android:visibility="invisible"
        style="@android:style/Widget.ProgressBar.Inverse">
        </ProgressBar>
    </RelativeLayout>
</RelativeLayout>
</LinearLayout>

```

3.3.3 Coding – Manifest file

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.audio.fatihah"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name" android:debuggable="false">
        <activity android:name=".audioCapture.Activity"
            android:label="@string/app_name"><intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter></activity></application>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"></uses-permission>
    <uses-permission android:name="android.permission.RECORD_AUDIO"></uses-permission>
    <uses-permission android:name="android.permission.VIBRATE"></uses-permission>
    <uses-sdk android:minSdkVersion="7"></uses-sdk>
    <!-- uses-permission android:name="android.permission.INTERNET" --></manifest>

```

3.3.4 Coding – Sound Processing

```

public void onMarkerReachedSoftClip(short[] buffer) {
    double th=1.0/3.0;
    double multiplier = 1.0/0x7fff;//normalize input to double 1,1
    double out = 0.0;
    for(int i=0;i<buffer.length;i++){
        double in = multiplier*(double)buffer[i];
        double absIn = java.lang.Math.abs(in);
        if(absIn<th){
            out=(buffer[i]*2*multiplier);
        }
        else if(absIn<2*th){
            if(in>0)out= (3-(2-in*3)*(2-in*3))/3;
            else if(in<0)out=-(3-(2-absIn*3)*(2-absIn*3))/3;
        }
        else if(absIn>=2*th){
            if(in>0)out=1;

```

```

        else if(in<0)out--1;
    }
    buffer[i] = (short)(out/multiplier);
} }

```

3.3.5 Coding – Alert Dialog

```

private void showSimplePopUp() {

    AlertDialog.Builder helpBuilder = new AlertDialog.Builder(this);
    helpBuilder.setTitle("CAUTION!");
    helpBuilder.setMessage("Emergency Alarm Detected!");
    helpBuilder.setPositiveButton("Ok",
        new DialogInterface.OnClickListener() {

            public void onClick(DialogInterface dialog, int which) {
                // Do nothing but close the dialog
            }
        });

    // Remember, create doesn't show the dialog
    AlertDialog helpDialog = helpBuilder.create();
    helpDialog.show();

}

```

3.3.6 Coding – Vibration

```

final Vibrator v = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);

v.vibrate(3000);

```

3.4 Interface prototype



Figure 3.4 Interface prototype

3.5 System Architecture

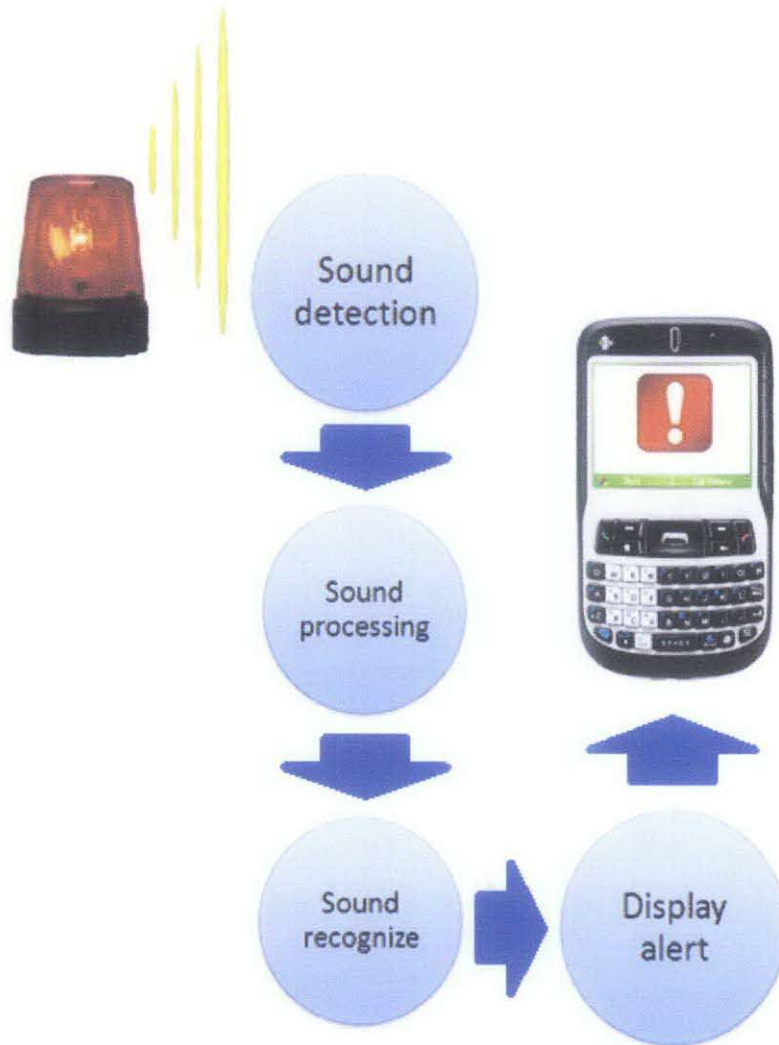


Figure 3.5 System Architecture

3.6 Flow Chart

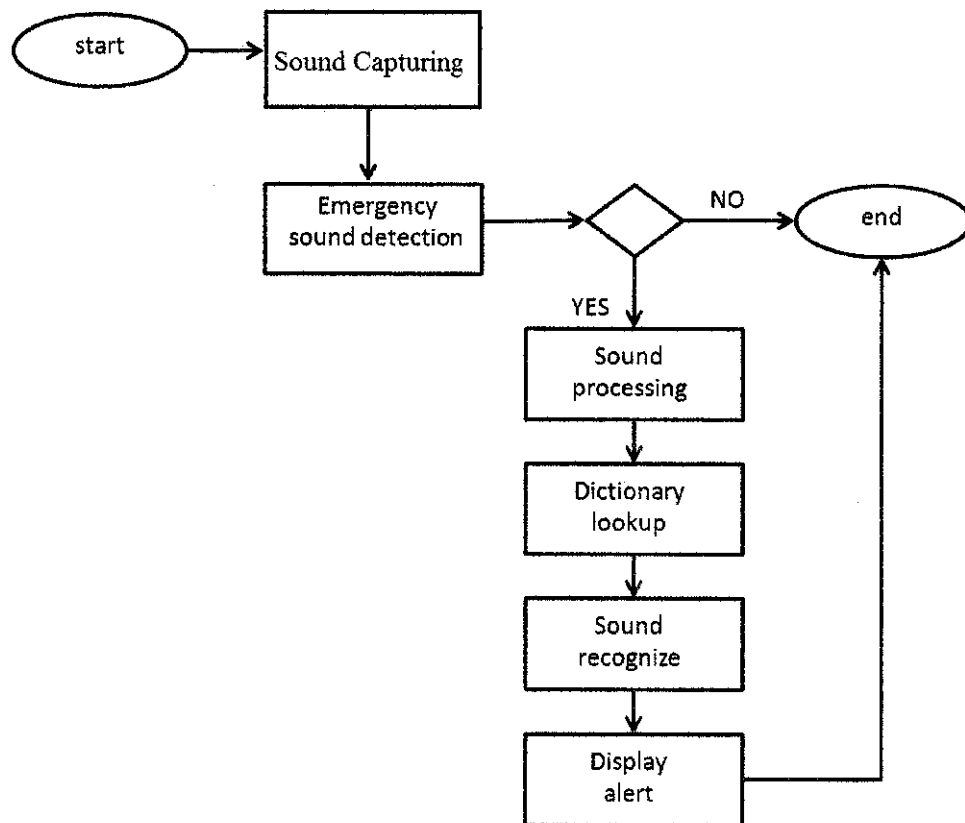


Figure 3.6 Flow Chart

Sound processing, dictionary lookup and sound recognize is a background process in this system. Therefore, the user will not be able to notice the process and only receive the output later after the execution done. This application is to run in the background once the user switches it on until the user switches it off.

3.7 Tools required

Software

- Eclipse IDE. (main development tool)

Eclipse IDE is an open source project and there are various plugins available to make Eclipse work with a variety of platforms and languages. It is most likely one of the best IDE development tools for Java based development. The Android SDK includes a plugin for the Eclipse environment for developing Android applications. Current version is *eclipse-java-galileo* and these editions will always remain free-of-charge.

- Android Developer Tools (ADT)

The Android Development Tools Plugin for Eclipse (ADT) is to let developer quickly set up new Android projects, create an application UI, add components based on the Android Framework API, debug your applications using the Android SDK tools, and even export signed (or unsigned).

Hardware

- Android mobile phones

CHAPTER 4

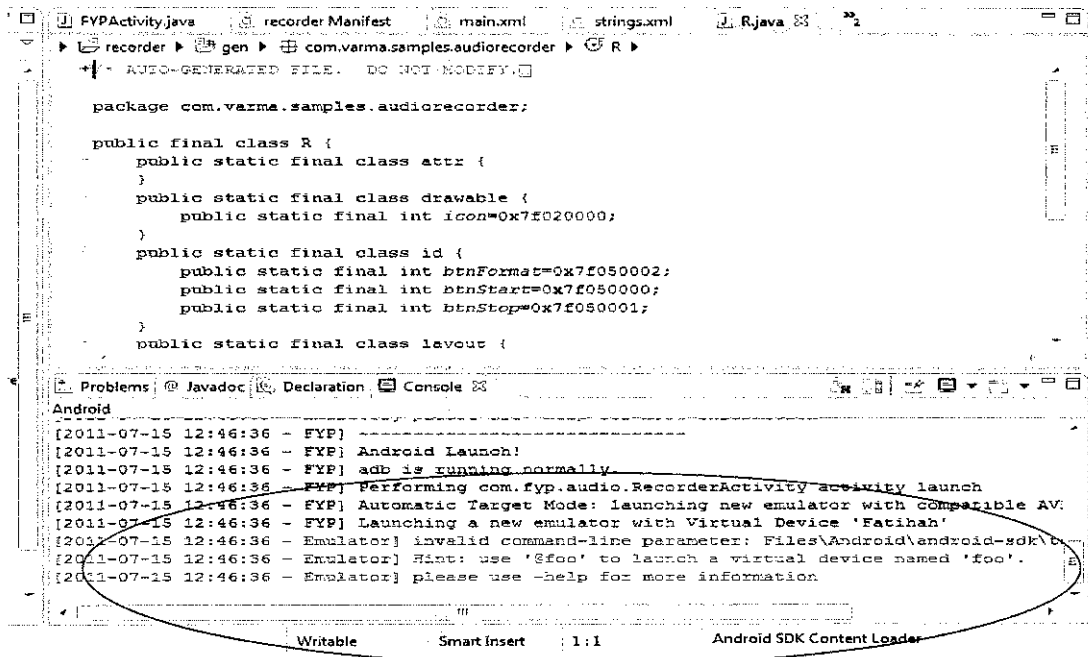
RESULT AND DISCUSSION

4.1 Result

According on the accomplishment analysis and survey, the result below are achieved:

| Area of Improvement | Benefit |
|--|--------------|
| The easy way to bring a device with application for deaf | Increase 40% |
| Time required for the program to run (eclipse emulator) | Decrease 60% |
| Error in Eclipse emulator | Increase 25% |
| Total number of bytes | 4096 |

Table 4.1 Result



```
package com.varma.samples.audiorecorder;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int btnFormat=0x7f050002;
        public static final int btnStart=0x7f050000;
        public static final int btnStop=0x7f050001;
    }
    public static final class layout {
    }
}
```

Android

```
[2011-07-15 12:46:36 - FYP]
[2011-07-15 12:46:36 - FYP] Android Launch!
[2011-07-15 12:46:36 - FYP] adb is running normally
[2011-07-15 12:46:36 - FYP] Performing com.fyp.audio.RecorderActivity activity launch
[2011-07-15 12:46:36 - FYP] Automatic Target Mode: launching new emulator with Comparable AV:
[2011-07-15 12:46:36 - FYP] Launching a new emulator with Virtual Device 'Fatihah'
[2011-07-15 12:46:36 - Emulator] invalid command-line parameter: Files\Android\android-sdk\platform-tools\adb.exe -s fatihah
[2011-07-15 12:46:36 - Emulator] Hint: use '@foo' to launch a virtual device named 'foo'.
[2011-07-15 12:46:36 - Emulator] please use -help for more information
```

Figure 4.1a Eclipse emulator error

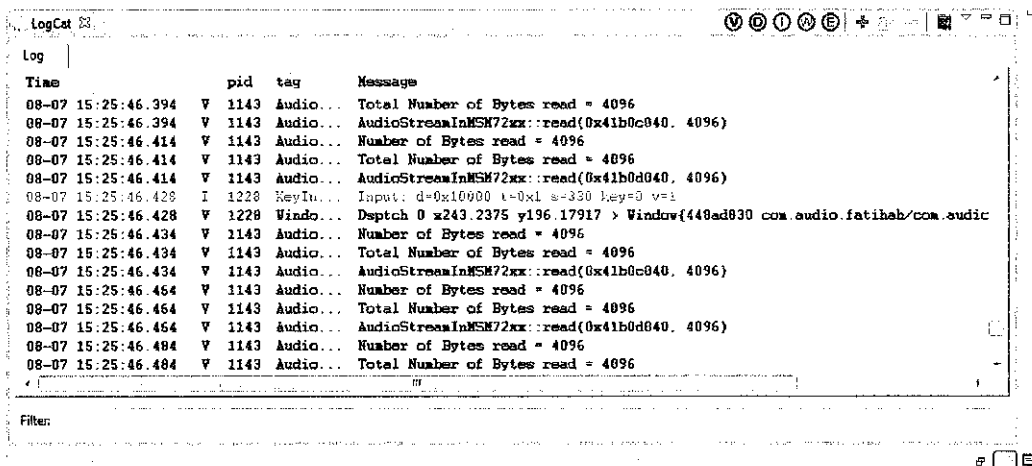


Figure 4.1b Result running on mobile

4.2 Discussion

Referring to the above table, it indicates that the Eclipse IDE have more disadvantage comparing to Netbeans IDE. The respond time for Eclipse IDE took more time to run especially when running the Android Emulator. This is because each time the emulator run, it will save the previous project cache for referring. To setting up the Android SDK on the Eclipse IDE process are more complicated and could result in error if the set up failed. Therefore, it required more memory space for it to run each time. For this project, it uses the audio recording process to run this project. The disadvantage of the emulator is that it does not run the audio recording and consequently the project has to be tested on the real Android device. On top of that, although this Emulator could give uneasiness to the developer, it also provides advantage such that developer does not have to have the real device to run the codes.

The above errors occur only in emulator since emulator does not have the ability to capture sound. Hence, this project should be run in the real phone and the result is shown in Figure 4.2.

4.3 Fast Fourier Transform

Fast Fourier Transform (FFT) converts signals from a time domain to a frequency domain and is the basis for many sound analysis and visualization algorithms. It converts a signal into magnitudes and phases of the various sine and cosine frequencies making up the signal. For example, take the signal shown in Figure 4.3.

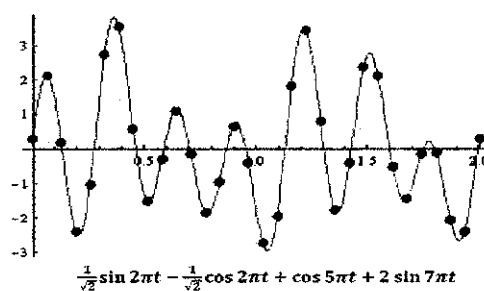


Figure 4.3 Sample signal

4.4 Signal Frequency

Frequency is the number of cycles per second. The frequency spectrum of a time-domain signal is a representation of that signal in the frequency domain. The frequency spectrum is generated via a Fourier transform of the signal, and the resulting values are usually presented as amplitude and phase, both plotted versus frequency. For example on this project is using signal wav of ambulance and police siren. The results had shown in Figure 4.4a and Figure 4.4b with frequency 11025 Hz and 44100 Hz.

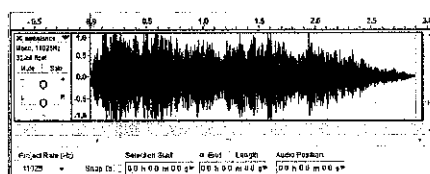


Figure 4.4a Ambulance sound waves

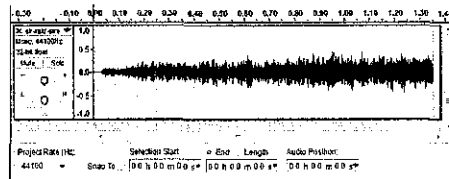


Figure 4.4b Police siren sound wave

In signal processing, sampling is the reduction of a continuous signal to a discrete signal. A common example is the conversion of a sound wave which is a continuous signal to a sequence of samples a discrete-time signal.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

In general, the level of progress of project development of sound capture and processes on mobile could be concluded as follow the planning with appropriate ways and approaches. By identifying the difficulty a significance of the project may lead to deep understanding of the main concept and principle of the project hence may contribute to the series of project achievements. Sound capture and processes might be the useful technology for deaf or hard of hearing to be alert with any emergency call and be able to respond towards it for their needs. With the support from the programming tool like Eclipse IDE to make the project development of sound capture and processes on mobile especially Android platform to be transformed into reality, many experts of Information Communication and Technology fields believe this applications that have been tested the functionality on the real person may make and produce impacts to help the deaf or hard of hearing people once it is accomplished and implemented. It is believed that the flexibility and scalability of this project makes it suitable for a wide range of deaf people-centric sensing applications and present two simple proof-of-concept applications in this paper.

5.2 Recommendation

For future development of the project, it is recommended to use the most advance programming concept for sound capturing and processing and to make a deep research on sound and signal processing. The project is to be implemented to run in background so that the objective to detect emergency alarm anywhere can be accomplished. It is important to include more people in research team and expose the developer to more courses related to the project as for example sound and signal processing.

A few recommendations are dedicated to Computer and Science Information Department of Universiti Teknologi PETRONAS such as invite experts from industry to deliver talk and training that relate to students' project especially about Information Technology and Information System issues to provide a clear picture related to their project and also to provide more training on Android developer applications.

References

- Baecker, R. M., Wolf, P., Rankin, K. The ePresence Interactive Webcasting System: Technology Overview and Current Research Issues. *Proceedings of Elearn 2004* (2004) 2396-3069
- Brotherton, J. A., Abowd, G. D. Lessons Learned From eClass: Assessing Automated Capture and Access in the Classroom, *ACM Transactions on Computer-Human Interaction*, Vol. 11, No. 2. (2004)
- Tyre, P. Professor In Your Pocket, Newsweek MSNBC (2005) Retrieved December 8, 2005, from <http://www.msnbc.msn.com/id/10117475/site/newsweek>
- Rabiner, L., & Juang, B.-H. (1993). *Fundamentals of speech recognition*. Englewood Cliffs: Prentice-Hall.
- Bocchieri, E. (2008). In *Automatic speech recognition on mobile devices and over communication networks (advances in pattern recognition)*, *Fixed-point arithmetic* (pp. 255-274). Berlin: Springer.
- Schultz, Tanja & Kirchhoff Katrin (2006). In *Multilingual Speech Processing*. Academic Press
- A. Adams, N. Gelfand, and K. Pulli. Viewfinder alignment. *Computer Graphics Forum (Proc. Eurographics)*, pages 597–606, 2008.
- D. Arthur and S. Vassilvitskii. k-means++: the advantages of careful seeding. In *SODA '07: Proc. 18th annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.
- S. Benhimane and E. Malis. Real-time image-based tracking of planes using efficient second-order minimization. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 943–948, 2004.
- X. Chen and A. L. Yuille. Detecting and reading text in natural scenes. In *Proc. IEEE CVPR 2008*, March 2008.
- T. N. Dinh, J. Park, and G. Lee. Low-complexity text extraction in korean signboards for mobile applications. In *Computer and Information Technology, 2008. CIT 2008. 8th IEEE Intl. Conf. on*, pages 333 –337, 8-11 2008.
- M. R. Schroeder. *Computer Speech: Recognition, Compression, and Synthesis*. Springer Verlag, Berlin, Germany, 1999



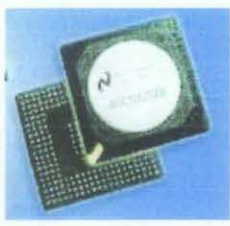

- Wong, G.S. and Embleton, T.F. (1995) *AIP Handbook of Condenser Microphones*, AIP Press, New York.
- A. Jain and B. Yu. Automatic text location in images and video frames. In Proc. 14th Intl. Conf. on Pattern Recognition, volume 2, pages 1497–1499, Aug 1998.
- G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In Proc. 8th IEEE and ACM Intl. Symposium on Mixed and Augmented Reality (ISMAR 2009), pages 83–86, Oct. 2009.
- V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, 10(8):707–710, 1998. Original in *Doklady Akademii Nauk SSSR* 163(4): 845–848 (1965).
- X. Liu and J. Samarabandu. An edge-based text region extraction algorithm for indoor mobile robot navigation. In Proc. IEEE Intl. Conf. Mechatronics and Automation, volume 2, pages 701–706 Vol. 2, July-1 Aug. 2005.
- S. M. Lucas. ICDAR 2005 text locating competition results. In *Document Analysis and Recognition, 2005. Proc.. 8th Intl. Conf. on*, pages 80 – 84 Vol. 1, 29 2005.
- J.G. Fiscus, “A post-processing system to yield reduced word error rates: Recogniser output voting error reduction (ROVER),” *Proc. IEEE ASRU Workshop*, pp. 347–352, 2007.
- G. Evermann and P. C. Woodland, “Posterior probability decoding, confidence estimation and system combination,” in *Proceedings Speech Transcription Workshop, College Park, MD, 2000*.
- M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J.Makhoul, “A study of translation edit rate with targeted human annotation,” in *Proceedings of Association for Machine Translation in the Americas, 2006*.
- K. Papineni, S. Roukos, T. Ward, and W. Zhu, “BLEU: a method for automatic evaluation of machine translation,” Tech. Rep. RC22176 (W0109-022), IBM Research Division, 2001.
- A. Stolcke and E. Shriberg, “Automatic linguistic segmentation of conversational speech,” in *ICASSP, 1996*.
- P. Koehn, F. Och, and D. Marcu, “Statistical phrase-based translation,” in *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference, 2003*.

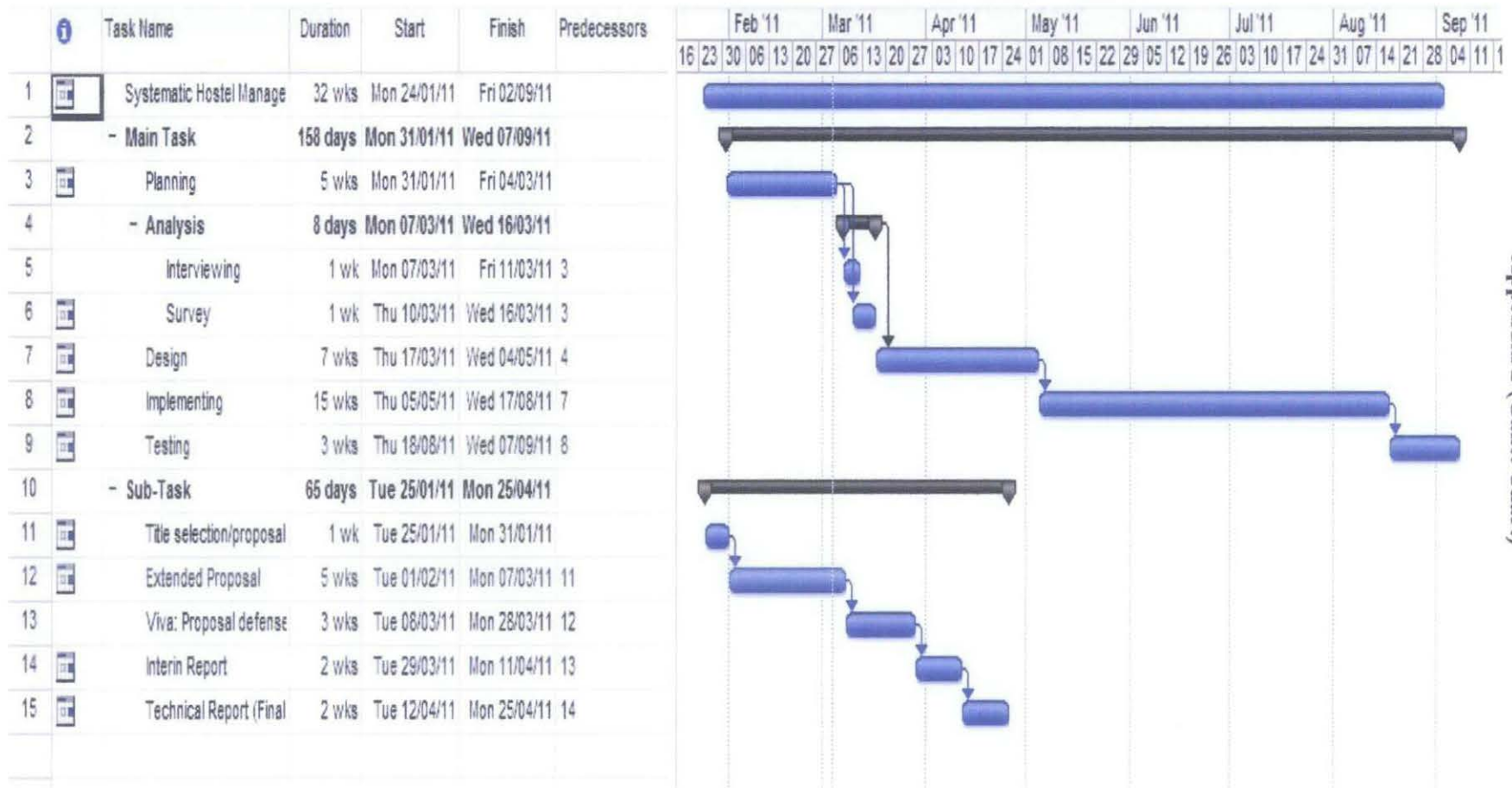
Websites

- <http://mandarsblog.wordpress.com/2010/07/16/android-sdk-eclipse-and-emulator-settings/>
- <http://developer.android.com/guide/developing/devices/index.html>
- <http://www.androiddevblog.net/android/android-audio-recording-part-1>

APPENDICES

Appendix 1

| | |
|--|--|
|  <p>Electrodynamics microphones</p> |  <p>Miniature electro-mechanical system microphones (MEMS microphones)</p> |
|  <p>Carbon microphones</p> |  <p>Condenser microphones</p> |



Appendix 2 (Gantt Chart)