A real-time speech recognition system on smart glasses for people with hearing difficulties

A Thesis Submitted to the
Graduate School of Engineering
In Partial Fulfillment of the Requirements
For the Degree of Master of Engineering
Under the
Department of Computer and Information Sciences
Tokyo University of Agriculture and Technology

By

Satjapong Meeklai

15646146

Kaneko Laboratory

January, 2017

ABSTRACT

Speech recognition technologies are tools that are used in several environments to assist people or students with difficulties by automatically converting oral conversation or lectures into text. In this study, we have created a real-time speech recognition system on smart glasses for facilitating and assisting people with hearing difficulties to be able to live in normal life situation better and easier. The created system uses an open source speech recognition library called python speech recognition which additionally and manually optimized to suit system requirements and be able to create captions simultaneously while a speaker is speaking. The python speech recognition implements Google speech recognition service that has significant performance over the other speech recognition services to deliver faster and better accuracy of transcripts for creating comprehensible captions and present them on smart glasses. Our speech recognition system is implemented on microcontroller (Raspberry Pi 2 Model B), which connects a microphone as an input device and smart glasses as output devices through a local network that is set up for providing connectivity between devices. Multiple smart glasses are able to connect the same microcontroller as long as they are in the same network. Our system was tested in specific targeting environments such as meetings and classrooms where only one speaker utters at a time. The experiments show that the optimized version of the library achieves better response time in delivering understandable captions without needs of human editors and the new way of captions presentation which is handier and more portable.

Keywords: Speech Recognition System / Microcontroller / Smart Glasses / People with Hearing Difficulties / Deaf

ACKNOWLEDGEMENT

First of all, I would like to sincerely show my deepest gratitude toward my academic supervisor **Professor Keiichi Kaneko**. Without his efforts, assists and advices, I would not be in this point where I'm now going to complete master degree at Tokyo University of Agriculture and Technology. He always provides finest environments for his students to study. I feel absolutely lucky and fortuned that I belong to his lab because of his carefulness, calmness and open-mindness never make me feel uncomfortable, unhappy or discouraged to work in the lab.

Moreover, I would like to thank to **Japanese Ministry of Education, Culture, Sports, Science and Technology** who support my school fees and daily live expenses through Monbukagakusho (MEXT) so that I can focus on my research without any worries.

I also feel thankful to **Assistant Professor Yuki Hirai** and **Associate Professor Tomoko Hongo** who supervised me during the time I'm in Tokyo University of Agriculture and Technology. With their kindness and great supports, my student life here was totally smooth and well-managed.

Last but not least, I would love to show my gratefulness to all of my beloved **friends** and **family** who always motivate, give me strength and accompany with me in this long journey far away from home. Without them, my life here in Japan would be totally boring, exhausted, silly and incomplete. I'll never forget my precious times with my friends here in Japan.

Finally, I would like to absolutely thank **myself** for not giving up, not losing myself and always be optimistic to everything that happened around me. Don't forget all of the accomplishments you have done so far and always go higher! Nothing is possible until it is done.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE **PAGES**

CHAPTER 1

INTRODUCTION

1.1 Background

Speech recognition technologies have been widely known and used for long time in a wide range of application from captioning video and television for the hearing-impaired, voice-controlled computer operation, and dictation [1]. Education is also another application that makes use of speech recognition technologies to assist both normal students, and students with any kind of difficulties increase the efficiency of study in the classroom. In the past, most commercial speech recognition software applications were developed for dictation with punctuation, not for transcribing extemporaneous speech, [2], [3]. Moreover, real-time captioning or transcription of speech provided by previous educational applications based on speech recognition technologies are mostly displayed on either large screen or individual portable screen devices [31] [33] [34]. With the current power of technologies, the speech recognition technology weakness has been solved, and new technologies like smart glasses that utilize the concept of virtual reality create new possibilities of developing a reliable, portable, and user-friendly real-time captioning system.

1.2 Purposes

Regarding competency of students with hearing difficulties, joining and studying in the same classes with normal students is uneasy. Students with hearing impairments cannot similarly process the audio of oral lectures compared to normal students resulting in difficulty of comprehending the lectures and acquiring lecture notes. Previous studies showed that students without disabilities recorded up to 70 percent more lecture information than students with learning disabilities [4] [32].

In this research, our purpose is to implement a real-time speech recognition system based on Google speech recognition that shows significant performance over the other speech recognition services [5]. It is employed in the classroom where students with hearing difficulties wearing smart glasses connected to the system. The system automatically and simultaneously transcribes the instructor's lecture and sends the transcripts to display on smart glasses. The benefits of producing lecture transcripts is to enhance both learning and teaching in students with hearing difficulty.

Not only helping students, the system is also expected to assist adults and elderly people with hearing difficulties from several real life situations, which are mainly designed for normal people.

1.3 Scope

The scope of the research is to create a system with the following guidelines. Using a microcontroller as a central device to process voices of a speaker, the system creates captions and sends them to smart glasses which are in the same network. The system uses Raspberry PI 2 Model B as a microcontroller. The operating system of Raspberry PI 2 Model B is called Raspbian. It is a version of Linux operating system that is modified to work with Raspberry PI. The system has a function to execute real-time speech recognition. But, the available speech recognition libraries of Raspbian are not capable of performing real-time speech recognition. Therefore, an open source library called Python speech recognition is chosen and optimized to be able to perform speech recognition in real-time. A network is also necessary to provide connectivity between smart glasses and the microcontroller. Smart glasses called Epson Moverio BT-200 are used in the system. Their operating system is Android version 4.1. So, an Android application was developed using Java in order to continuously receive captions from the microcontroller and seamlessly show received captions to users.

1.4 Expected Benefits

The expected benefits of the research are as follows:

- The system can help people with hearing difficulties to be able to communicate with normal people.
- The system can help people with hearing difficulties to understand the points and meaning of conversation without the need of sign language interpreters.
- The system can help people with hearing difficulties to participate in the situations in which they cannot participate with normal people.

CHAPTER 2

FEASIBILITY STUDY

This chapter discusses the feasibility of the research. The chapter starts with the problem statement that focuses on the main problems and their causes. Then it discusses related research and projects that were investigated for knowledge and techniques to help support the project. Lastly, it presents the requirements specifications of the research.

2.1 Problem Statement

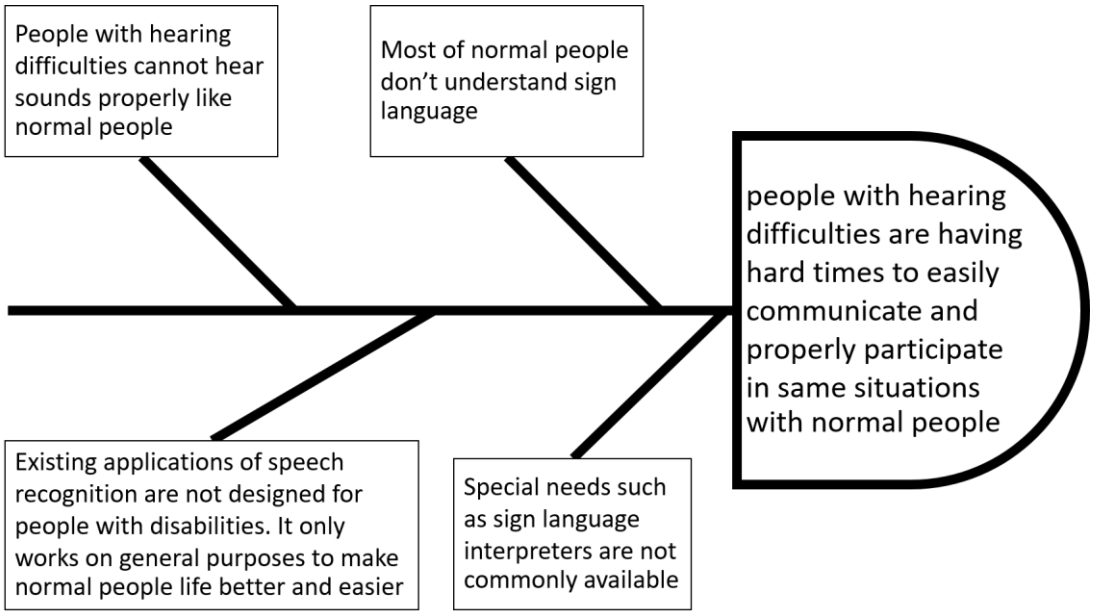

Figure 2.1: Fish Bone Diagram

In Figure 2.1, the fish bone diagram shows the analysis of the causes of the problems of why people with hearing difficulties are having hard times to easily communicate and properly participate in same situations with normal people.

The problem occurs from following causes:

- People with hearing difficulties cannot hear sounds properly like normal people.
- Most of normal people don't understand sign language.
- Special needs such as sign language interpreters are not commonly available.
- Existing applications of speech recognition are not designed for people with disabilities. It only works on general purposes to make normal people life better and easier.

2.2 Related Research

2.2.1 Enhancing the Usability of Real-Time Speech Recognition Captioning Through Personalised Displays and Real-Time Multiple Speaker Editing and Annotation

Text transcriptions of the spoken words can benefit deaf people and also anyone who needs to review what has been said (e.g. at lectures, presentations, meetings etc.).

This research describes the development of a system that can provide an automatic text transcription of multiple speakers using speech recognition, with the names of speakers identified in the transcription and corrections of speech recognition errors made in real-time by a human editor as shown in Figures 2.2, 2.3 and 2.4.

This research uses software called ViaScribe [4] [6]. ViaScribe is a speech recognition application that automatically formats real-time text captions from live speech with a visual indication of pauses. ViaScribe was chosen for real-time captioning because it has a proven track record by language learning members for reliable captioning and had a client-server platform for streaming live transcription to students' laptop PCs during lectures.

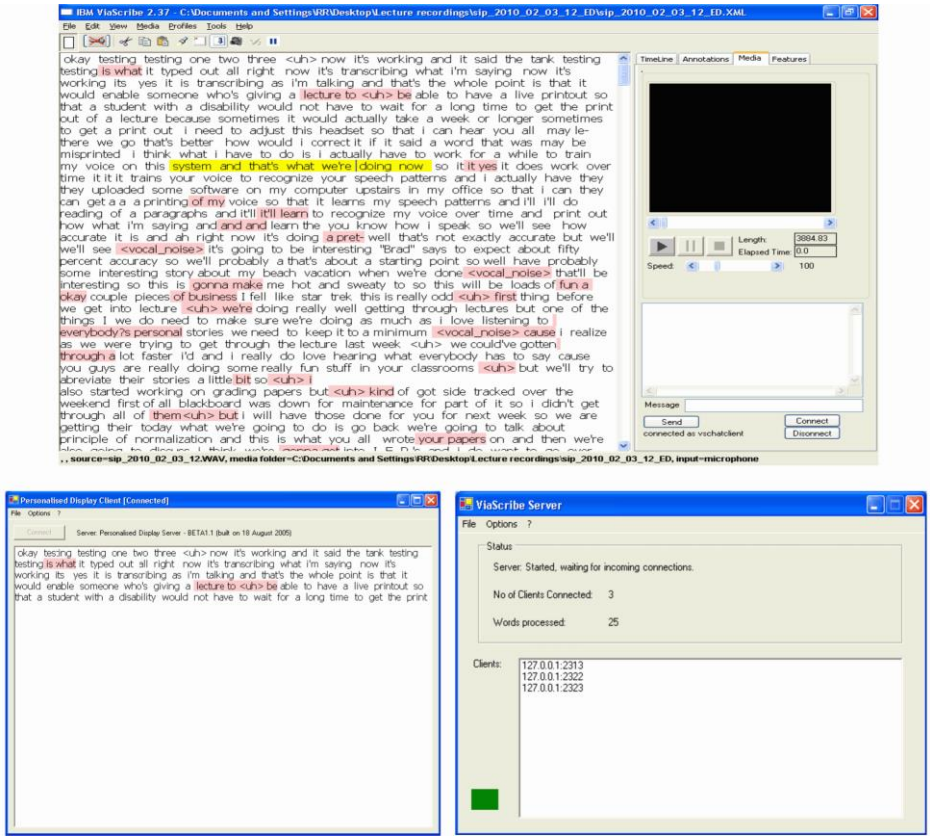

Figure 2.2: Interfaces of ViaScribe

The research claimed that using an individual personalized and customizable display is better than projecting the text onto a large screen in many situations because users can individually adjust and modify things such as font size, and format font color by themselves.

Also, this research introduced the concept of real-time editing. Speech recognition accuracy may be reduced where the original speech is not of sufficient volume/quality (e.g. poor microphone position, telephone, internet, television, and indistinct speaker) or when the system is not trained (e.g. multiple speakers, meetings, panels, and audience questions). Therefore, to improve accuracy of verbatim captions created directly from the voice of the original speaker the application RealTimeEdit (RTE) was developed to enable corrections to automatic speech recognition (ASR) captions to be made in real-time [7]. One editor can find and correct errors or the task of finding and correcting errors can be shared between two editors, one using the mouse and the other the keyboard.

In this way a real-time editor can be used in situations where high accuracy captions are required and a real-time stenographer is not available. Up to eleven corrections per minute were achieved by untrained users of an initial prototype of RTE. Analysis of an ASR transcript with a 22% error rate also suggested that correction of less than 20% of the 'critical' errors may be required to understanding the meaning of all the captions [8].

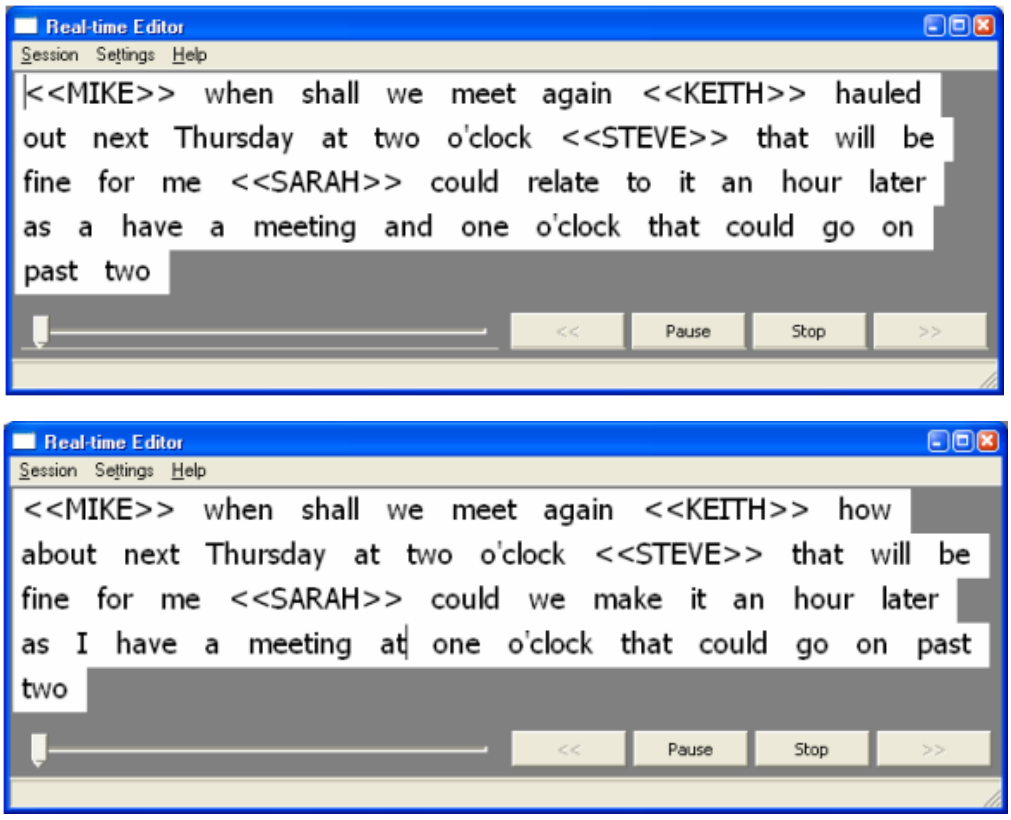

Figure 2.3: RTE displays a caption before and after correction

Moreover, the system also supports multiple speakers at the same time. In situations where there is more than one person speaking, using multiple instances of ViaScribe creates captions in multiple windows making it difficult to follow the sequence of the utterances. To produce a transcript of the session with speakers identified, the application RealTimeMerge (RTM) was developed to add the speaker's name to the text captions and merge the streams from the instances of ViaScribe.

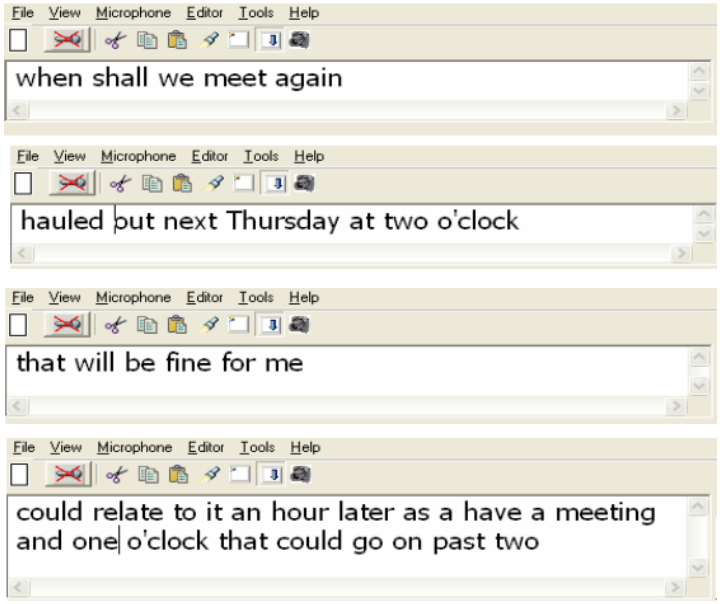

Figure 2.4: Four Instances of ViaScribe showing the ASR text captions with errors for four separate speakers

2.2.2 Using Speech Recognition for Real-Time Captioning and Lecture Transcription in the Classroom

This research conducts a study of using two distinct methods of speech recognition mediated lecture acquisition in different classroom environments. First method is real-time captioning, similar to above related research. Second method is called postlecture transcription. Each method uses different technique and tools to perform. Both methods were compared according to technical feasibility and reliability of classroom implementation, instructors' experiences, word recognition accuracy, and student class performance.

We will mainly focus on postlecture transcription method of this research since real-time captioning method was already explained in above related research. Real-time captioning of this research and above related research are identical because both real-time captioning methods use the same software (ViaScribe).

Figure 2.5 gives you an overview understanding of speech recognition mediated lecture acquisition meaning.

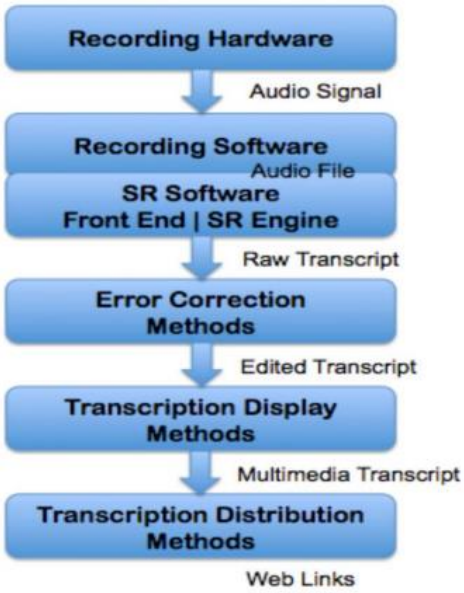

Figure 2.5: Overview of speech recognition mediated lecture acquisition

Features	ViaScribe (RTC)	HTS (PLT)
Recording method	ViaScribe has in-built recording capability through classroom PC that receives audio from wireless headset microphone worn by lecturers.	This system runs software on classroom PC to record lecture audio in a SR-compatible format.
User-dependent/ independent SR engine	Initial user-specific profile training is necessary to accustom the SR software to the speaker's voice to maximize word recognition accuracy.	No user training is required. The speaker wears a high-quality microphone to record a lecture audio file.
Error correction	The generated raw transcripts can be corrected offline after the lecture in order to continually update the user's voice profile.	Recorded audio file is uploaded to a HTS website for transcription and correcting errors via user interface.
Display method	Classroom PC runs server software during class to transmit captioning in real-time to student laptops PCs as clients or to a classroom projection screen.	Lecture transcripts can be synchronized with audio and PowerPoint slides and put online as multimedia notes.

Figure 2.6: Comparison of major Functionalities between real-time captioning and postlecture transcription methods

Figure 2.6 shows you comparison of major Functionalities between real-time captioning and postlecture transcription methods.

Postlecture transcription method employed a user-independent speech recognition algorithm to optimally generate multimedia class notes with synchronized lecture transcripts, instructor audio, and class PowerPoint slides for students to access online after class. This method primarily uses a service from IBM called Hosted Transcription Service to create postlecture transcription because of its higher word recognition accuracy rates compared to other systems [35]. It automatically transcribes a variety of standard audio or video file formats through a cloud service, and its engine also employs a double-pass decoding technique, which dynamically adjusts to the speaker's voice, without requiring voice profile training or enrollment. Figure 2.7 describes an overview of postlecture transcription method using Hosted Transcription Service.

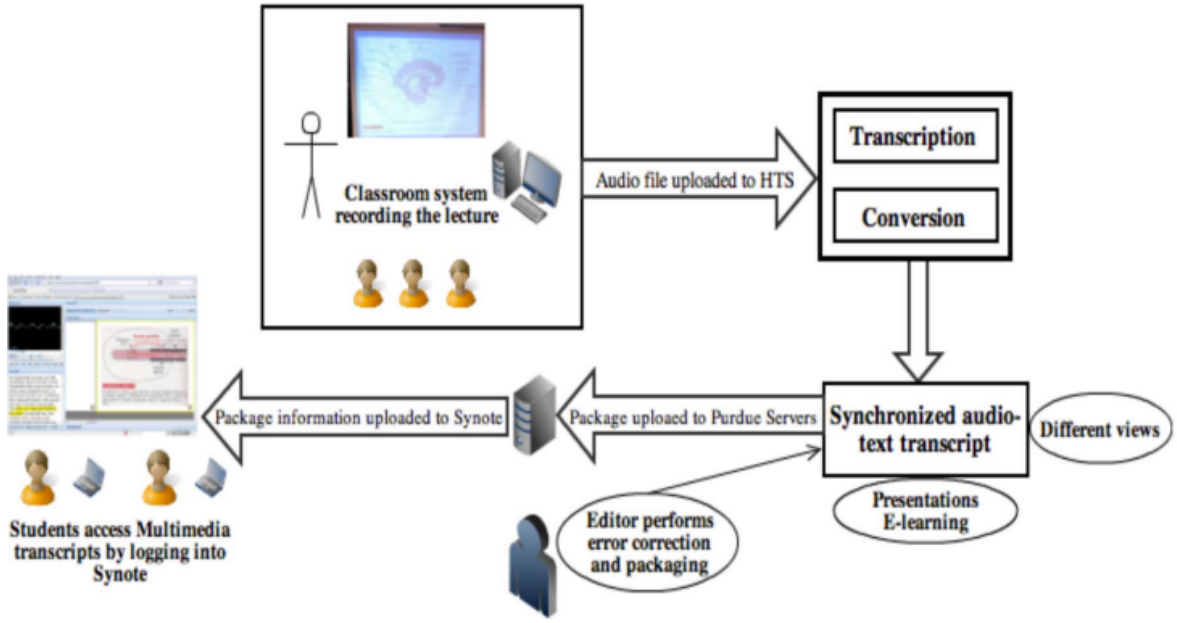

Figure 2.7: An overview of postlecture transcription method using Hosted Transcription Service

2.2.3 Automatic Classification of Usability of ASR Result for Real-time Captioning of Lectures

This research proposes a similar technique with our research to enhance automatic speech recognition to support hearing-impaired students in classrooms [36]. Correction of speech recognition errors often delays results to be shown on devices, however, not all results need to be corrected. This research creates an automatic classification system of automatic speech recognition result based on their usability.

The researchers also create a real-time captioning system by incorporating the automatic classification method and the presentation method.

The usability of captions is defined by syntactic correctness, errors and redundant spoken expressions in automatic speech recognition results. Each result is classified to be either “valid”, “invalid” or “to be checked” state, using hand-crafted rules and a created machine learning framework. The classification system is designed to work with Japanese lectures only.

In this work, by using morphological and acoustic features, we classify automatic speech recognition result into one of three categories; valid input, invalid input and to be checked. We adopt a grammatical chunk as a unit for this classification. First, we automatically segment the result into chunks, then apply rule-based classification using syntactic information. As the rule does not consider result errors and redundant spoken expressions, we further apply a framework of conditional random fields to make a final decision.

Thus, instead of making editors to check every single result, the classification system which is using machine learning framework helps editors to do the first screen of results produced from automatic speech recognition. As a result, when the work load of editors decreases, the speed of caption representation is faster. The outcome of this research claims to reduce the unnecessary correction of captions approximately 66.1%. Additionally, regarding responses from subjects of the trial which was conducted by the research team, the proposed system gives faster feeling than conventional system.

2.3 Requirement Specifications

The requirements of the system are as follows:

- The system can automatically adjust ambient noise.
- The system can tolerate to surrounding noise.
- The system can perform speech recognition and captioning in real time without significant delay and inaccuracy of results.
- The system can show interim results.
- The system can correct, improve and replace captions overtime.
- The system can support multiple smart glasses at the same time.
- The system can continuously stream captions to smart glasses without delay.
- The captions can be shown on smart glasses with appropriate font size and color.
- The system provide a local area network to connect devices altogether

CHAPTER 3

SYSTEM ANALYSIS AND DESIGN

This chapter describes the overall design concept of the research and includes requirement analysis, implementation techniques and system design. The requirement analysis describes what need to be fulfilled in the system. The implementation techniques describe all the techniques and technologies that are used to create the system. The system design is illustrated in the forms of Context diagram and Flow Chart. User interface of the system is also described as well.

3.1 Requirement Analysis

3.1.1 Microcontroller

The requirements of a microcontroller in our system are as following:

- The microcontroller can support a microphone (both wired and wireless).
- The microcontroller listens to one speaker at a time.
- The microcontroller can listen to a speaker continuously.
- The microcontroller can define the end of a phase.
- The microcontroller can calculate ambient noise of the room and set a threshold of the noise level in the room.
- The microcontroller is connected to the internet.
- The microcontroller can connect with Google speech recognition service.
- The microcontroller can record sounds and keep them in form of FLAC.
- The microcontroller can send http requests with a FLAC file (voice data) to Google speech recognition service to do speech recognition.
- The microcontroller can acquire interim and final results of a speech.
- The microcontroller can create a caption from text results of a speech.
- The microcontroller can connect and send captions to smart glasses via TCP.

3.1.2 Smart Glasses

The requirements of smart glasses in our system are as following:

- The smart glasses can listen and receive captions from microcontroller continuously.
- The smart glasses represent captions with appropriate font size, font color and font place.
- The smart glasses have a dim backgrounds.

- The smart glasses provide a mechanism to show previous caption.

3.2 Implementation Techniques

The following languages, libraries and tools are used:

- **Programming Languages**
  - Python 2.7
  - Java SDK 1.8
- **Libraries**
  - PyAudio
  - Python Speech Recognition 3.4.6
- **Tools**
  - Atom
  - Android Studio

When the system is started, the microcontroller is also started to listen to surrounding noise and set the energy of ambient noise of the room. Next, when a speaker starts to speak, the microcontroller starts to record the sound in form of bytes and saves it as a FLAC [9] file every two seconds. These FLAC files will be responsible by sub threads for doing http request [10] to send them to Google speech recognition service [11] to perform speech recognition. After Google finishes transcribing the speech, it returns transcripts (both interim and final ones) of the speech back to the microcontroller in form of JSON [12]. The microcontroller creates a caption from these transcripts and sends it to smart glasses via TCP/IP [13] for representing to users.

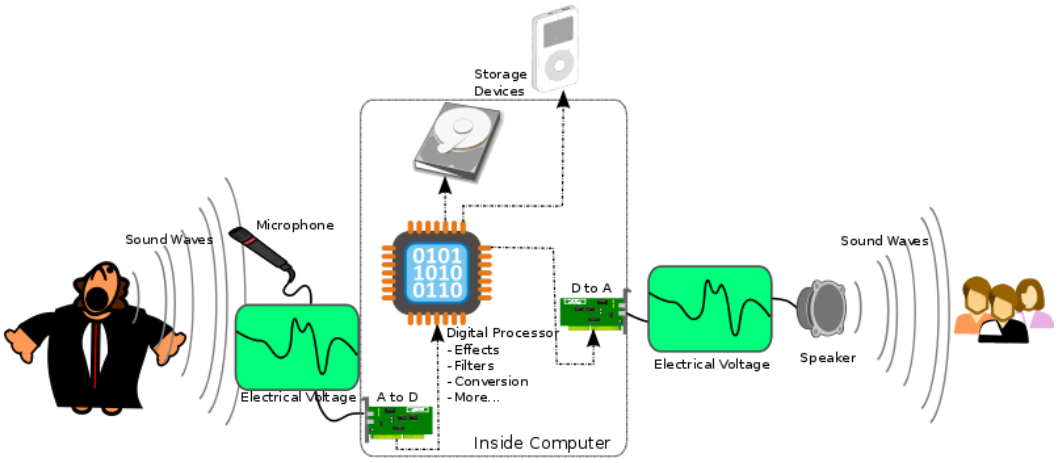

Figure 3.1: Pictorial summary of the process of storing sounds into a computer

The research uses a technique of multithreaded programming [14] to optimize Python Speech Recognition library so that the library can achieve real-time speech recognition so that a speech is being transcribed while a user is speaking to the microphone. Figure 3.1 gives an overview of how sounds are stored into a computer.

3.3 System Design

3.3.1 Context Diagram

From Figure 3.2, there are two external entities interacting with the system in a situation which are speakers and people with hearing difficulties. Firstly, the microphone of the system needs to be set up close to the speaker because accuracy of the captions essentially depends on the clearness of the speaker's voice. The closer the microphone is, the better output will be. When the speaker starts speaking, the system listens and transcribes the speech continuously while simultaneously creating captions from transcripts and sending them to represent on smart glasses which are being worn by people with hearing difficulties. As a result, people with hearing difficulties are able to understand the meaning and points of speeches or conversation easier without intolerable delay or special needs such as sign language interpreters or stenographer.

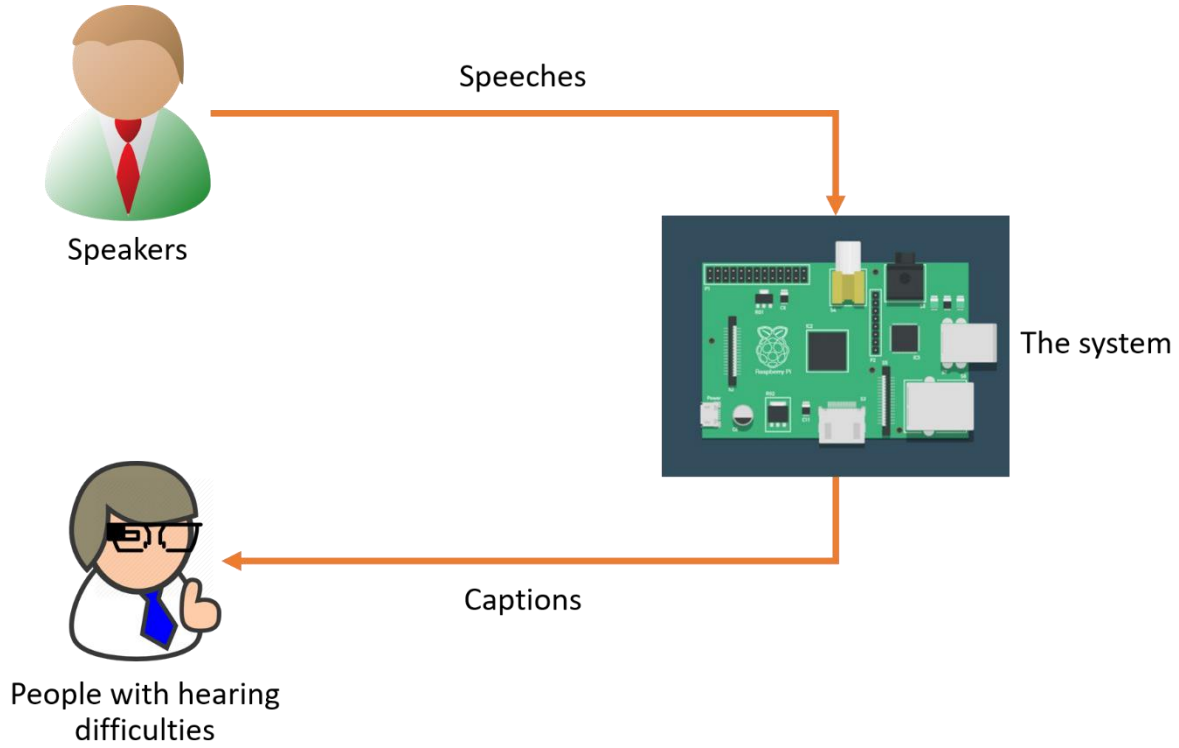

Figure 3.2: Context diagram of the system

3.3.2 System Flow Chart

Figure 3.3 shows how the system works after it has been started. First of all, the system listens to surrounding sounds for a second, calculates the energy of the ambient noise and sets it as a threshold. After the system defines the threshold of the surroundings, it continues listening to the surroundings and calculating the energy of the surrounding sounds in real time. After a speaker begins to speak, or gives a speech or communicate with someone, the energy of surrounding sounds of a period of time rises significantly. Therefore, the system is able to define that now a speaker is speaking, and starts to record the sounds. The system continuously compares the current energy of surrounding sounds and the ambient noise in order to define when a phase begins and ends. One speaker might speak for a quite long period of time without a pause. Consequently, making many speech recognition software that are not designed to work in real-time do not give any transcripts back until they found the end of the phase. On the contrary, the optimized version of Python speech recognition library which is used in the system is designed to work in real time so that the system can perform speech recognition and stream transcripts back while a speaker is speaking. The answer of how the system achieved this is multithreaded programming. While a speaker continues speaking, the system cuts the speech every 2 seconds. Then it creates an instance of the audio data of every 2 seconds and sends it to Google speech recognition service. After Google speech recognition service finishes processing the audio data and sends results back, the system creates a caption of the speaker's speech and sends it to illustrate on smart glasses. The technique of how system creates a caption will be precisely explained later in Chapter 5, software architecture.

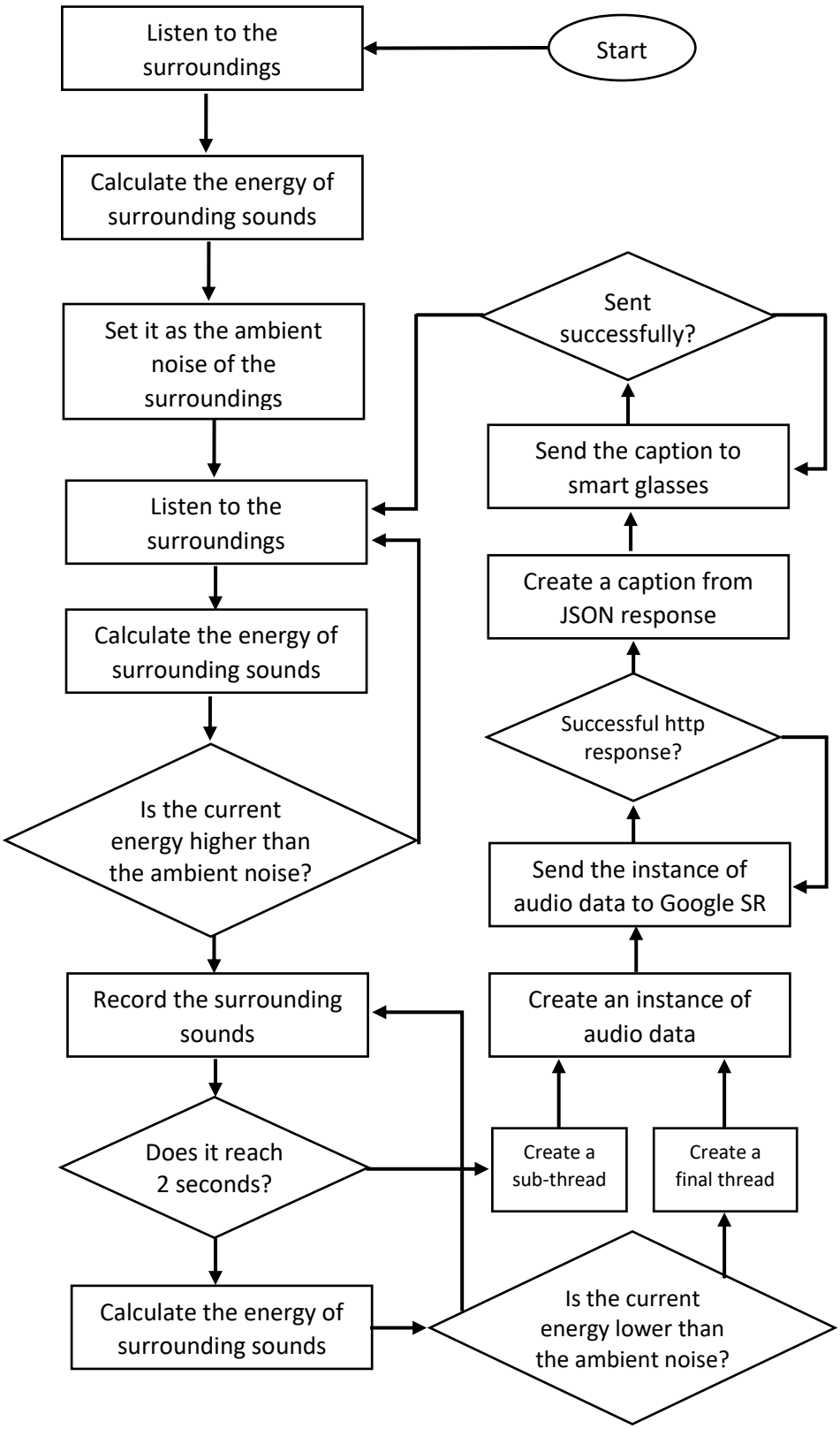

Figure 3.3: Flow chart of the system (main thread)

3.4 Graphical User Interface

Figure 3.4 shows a picture of the developed application when it is just started. No captions are being shown. It seems to be a dark screen in the picture but it's actually a dim screen which users can see through it. It does not block or interfere user's vision at all.

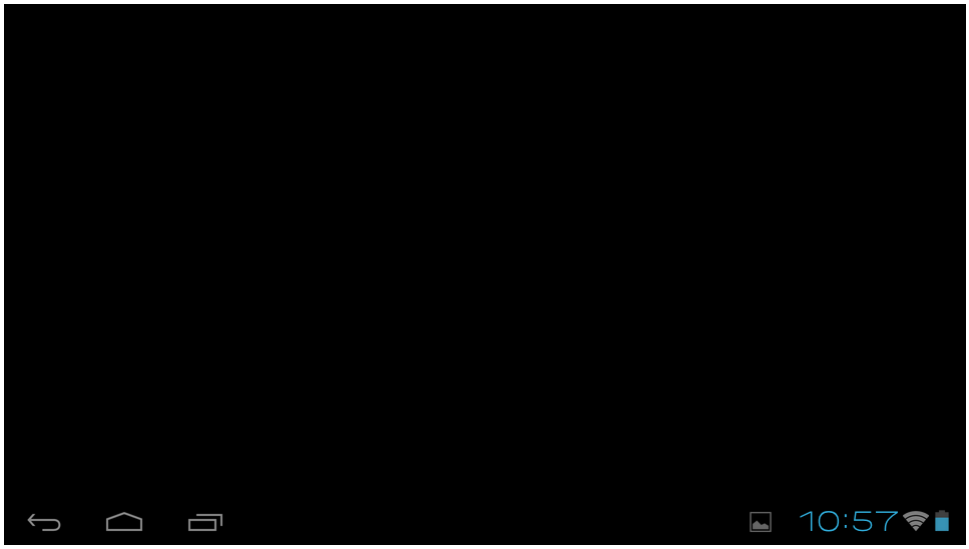

Figure 3.4: Standby screen

When a speaker utters a short phase to the system, the system does not need to show interim results of the speech to user at all because Google speech recognition service takes just short amount of time to perform speech recognition and send a list of transcripts back to the system. The system ignores showing all the interim results and just shows the final transcript to the users like what is represented in Figure 3.5.

Figure 3.5: A final caption is being shown on the screen

As it can be seen in Figure 3.6, when a phase is too long, the system does not need to wait until the phase has finished before performing speech recognition. Regarding Chapter 5, the optimized of Python speech recognition library, it can perform speech recognition in real time and show a caption that is made from interim results to the users.

Figure 3.6: The application shows a caption of interim results

Figure 3.7 illustrates the updated version of the same caption that is shown in Figure 3.6 when Google speech recognition service streams newer interim results back to the system. Note that Google speech recognition service does not need to process the whole audio data before it can send a list of transcripts back. It can transcribe the audio data in real time and send results back concurrently.

Figure 3.7: Updating the caption with the new interim results

Figure 3.8 demonstrates the completed caption. When a phase finishes, it is transcribed by Google speech recognition service, and the final result of the phase is sent back to the system. Regarding Chapter 5, caption creating process, the system always replace the final transcripts of speeches because they are the result of completed audio data. In other words, they are the most accurate and trustworthy results. Whether they are completely matched to the speeches or not, they are the most perfect results that Google speech recognition service can give to us.

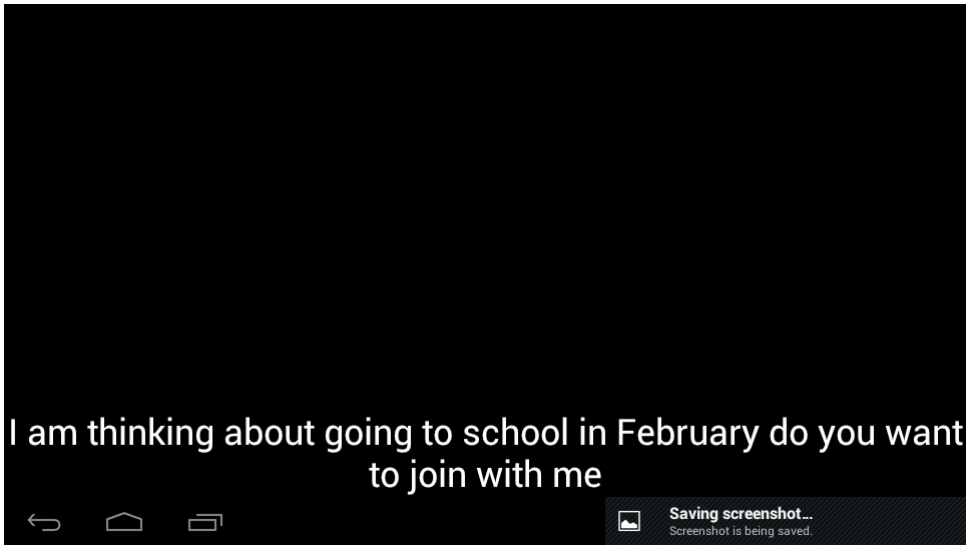

Figure 3.8: The final result of the caption

Figure 3.9 shows a sample of how it looks like on the microcontroller side. Usually, the microcontroller is not necessary to connect to a monitor in order to run the system. These results on the microcontroller side are only for debugging and testing purposes.

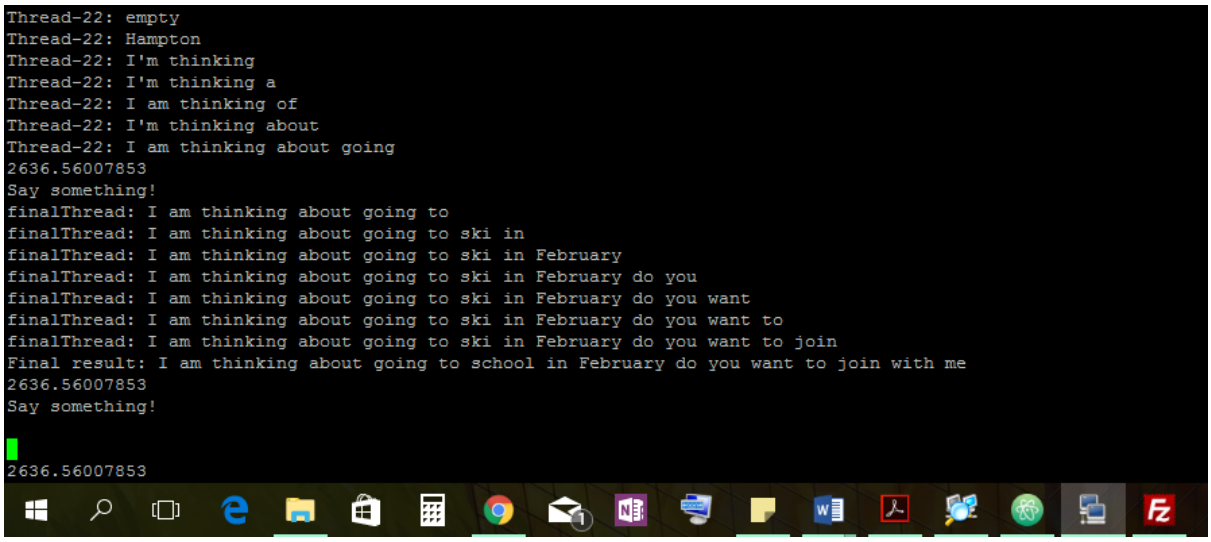

Figure 3.9: Results on the microcontroller side

CHAPTER 4

SYSTEM ARCHITECTURE

This chapter describes an overview and each components of the system. System overview gives a simple explanation about what technologies, devices and models are used in the system. System components explain in detail about each component that is using in the system.

4.1 System Overview

The system is developed on a microcontroller (Raspberry Pi 2 Model B) and it is designed to be a centralized system so that multiple smart glasses can be connected at a time. The performance of a microcontroller is not a main issue in this case since it is only used to be middleman to receive inputs (speeches), pre-process inputs, send them to Google and handle responses from Google speech recognition service to create readable captions.

One big advantage of this design despite from being portable and easy installation is low power consumption. Because the real-time captioning process is done by Raspberry Pi, which is connected to external source of power, the smart glasses which act like clients are just used to display the captions sending from Raspberry Pi.

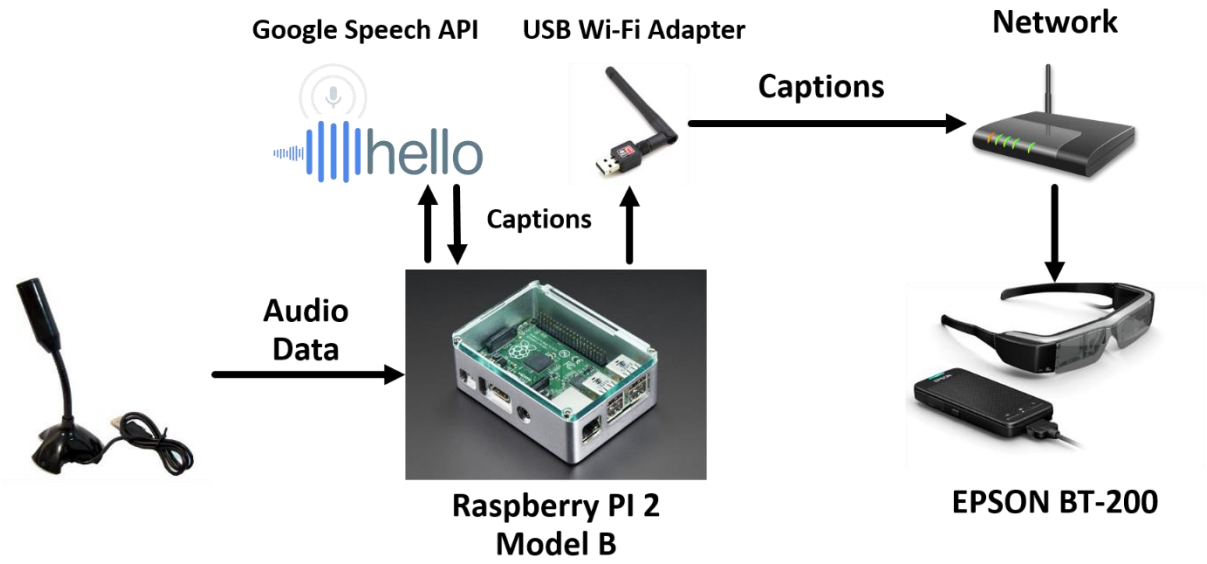

Figure 4.1: System overview

The microphone supported by the system for now is only a wired microphone (USB type). Although the first concept of the research was using wireless microphone such as Bluetooth headset with the microcontroller, the result did not come out as expectation because the Raspberry Pi operating system (Raspbian) does not well support wireless microphone.

Since all devices in the system are designed to be in a same network, a simple wireless router is necessary for creating a network to connect all the devices together. The router provides internet access and IP address to the devices so that the microcontroller can do http request to Google speech recognition service and send the captions to every smart glasses that are in the same network.

Finally, augmented reality plays an important role in this research. Smart glasses are used in the system to display the captions to people with hearing difficulties. This research is the very first one that applies augmented reality technology with the powerful speech recognition system from Google. Therefore, people with hearing difficulties can easily wear smart glasses, and they don't need to be afraid to go outside participating in real-life situations with normal people anymore.

There are some requirements that need to be prepared in order to run the system. Required hardware and software according to Figure 4.1 are as follow:

Hardware

- Raspberry PI 2 Model B (Microcontroller)
- Logicool HD Pro Webcam C910 (USB microphone module)
- Billion 5200W-TR2 (Wireless router)
- I-O DATA WN-G300UA (USB WiFi Adapter)
- EPSON Moverio BT-200 (Smart glasses)

Software

- Python 2.7
- PyAudio library
- Python speech recognition version 3.4.6 library
- Java sdk 1.8
- Android sdk (API level 15)

4.2 System Components

4.2.1 Microphone (Logicool HD Pro Webcam C910)

Figure 4.2 demonstrates a picture of the microphone that is used in the system. It is Logicool HD Pro Webcam C910. It is a web camera, but every web cameras nowadays have built-in microphones as well. Basically, any kinds of microphone can be used with the microcontroller as long as they are supported. No special configurations are required. However, only the USB microphone is currently supported due to the lack of supports from Raspbian. The HD Pro Webcam C910 has built-in stereo microphone which claims that it can record audio in high definition [15]. The reason that this web camera has been selected in the system is because it has a good built-in microphone, which is well supported by Raspberry Pi [16].

Figure 4.2: A picture of Logicool HD Pro Webcam C910

As it is described previously, the concept was using a wireless microphone, which is a Bluetooth headset. In other words, the microcontroller connects with a Bluetooth headset wirelessly so that a speaker can wear and use it like a wireless microphone. The situation is pretty same as people use their Bluetooth headsets with their phones to call someone else. Nevertheless, after we had tried to connect a Bluetooth headset with the microcontroller, the result came out that the currently official supported version of the library called PulseAudio by Raspbian is too old. As a result, Headset profile (HSP), one subset of Bluetooth profiles, has not been supported yet.

In order to use Bluetooth technology, a device must be compatible with the subset of Bluetooth profiles (often called services), which are necessary to use the

desired services [17]. A library called BlueZ is used to work with Bluetooth technology in Raspberry Pi. BlueZ provides support for the core Bluetooth layers and protocols. It is flexible, efficient, and it uses a modular implementation [18].

To set up Headset profile, BlueZ uses a library called PulseAudio to control Advanced Linux Sound Architecture (ALSA) [19] in order to set up HSP so that it can wirelessly receive audio input. PulseAudio is a sound system for POSIX OSes, meaning that it is a proxy for user sound applications. It allows the user to do advanced operations on user sound data as it passes between user application and user hardware. Things such that transferring the audio to a different machine, changing the sample format or channel count, and mixing several sounds into one are easily achieved using a sound server [20].

4.2.2 Microcontroller (Raspberry PI 2 Model B)

Figure 4.3 shows a picture of Raspberry Pi 2 Model B. When this research was started, the newest version of Raspberry Pi was Raspberry Pi 2. However, Raspberry Pi 2 has been now replaced by Raspberry Pi 3, which was launched in February, 2016. The followings are specification of Raspberry Pi 2 Model B:

- A 900MHz quad-core ARM Cortex-A7 CPU
- 1GB RAM
- 4 USB ports
- 40 GPIO pins
- Full HDMI port
- Ethernet port
- Combined 3.5mm audio jack and composite video
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot
- VideoCore IV 3D graphics core

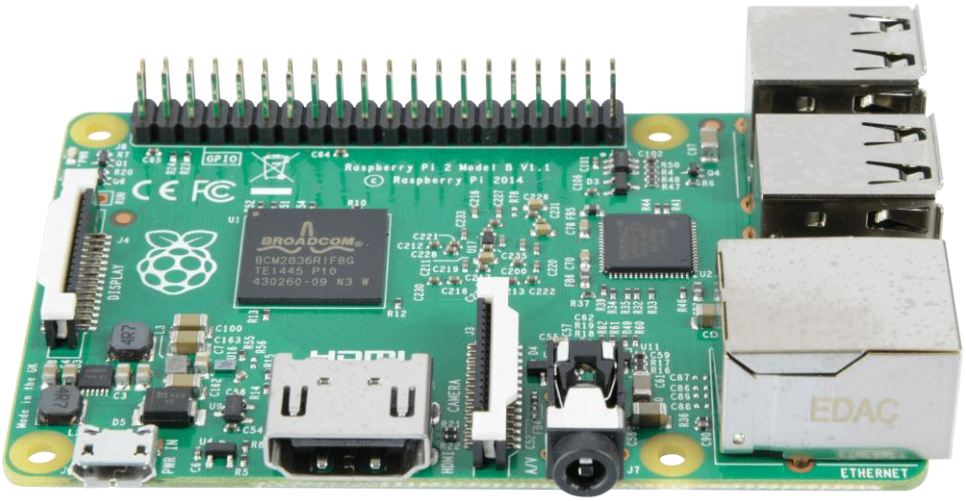

Figure 4.3: A picture of Raspberry Pi 2 Model B

It is able to run the full range of ARM GNU/Linux distributions, including Snappy Ubuntu Core, as well as Microsoft Windows 10 [21].

Raspbian is the operating system that runs in Raspberry Pi. It is a free operating system based on Debian (one of the Linux distributions) [22] optimized for the Raspberry Pi hardware. An operating system is the set of basic programs and utilities that make the Raspberry Pi run [23]. The initial build of over 35,000 Raspbian packages, it is optimized for best performance on the Raspberry Pi. This is very important point if a person ask himself/herself about what is the main difference between microcontrollers besides processing power. Because there are tons of hardware in this world, it is not easy to keep an operating system especially Linux that has many distributions to be always perfectly compatible with all the hardware. Strong community and popularity of Raspberry Pi can ensure that developers will not find their hard time to configure some hardware by themselves.

Someone might question about the processing power of the Raspberry Pi 2 Model B that it might be better to use a PC or other more powerful microcontrollers because they can process things faster. In fact, the assumption is absolutely correct. PC has higher processing power which results in the speed of delivering output. However, in this research, as mentioned, the microcontroller just acts like a middleman to send the audio data for speech recognition at Google speech recognition service, and then to receive a list of transcripts back to create a caption. Processing power does not play a substantial role to speed of delivering output. Thus, it is the speed of internet that plays an important role in this research. Faster speed means less time of uploading and downloading data from/to Google speech recognition service. Another main factor that affects the speed of delivering output is the size of inputs. Longer audio data would certainly take longer time to process by Google speech recognition service.

As a matter of fact which will discuss later at Chapter 5, Google speech recognition service proves that its speed is the best among other competitors.

4.2.3 Router (Billion 5200W-TR2)

All devices in the system must be connected. In other words, all the devices must be in the same network. After a caption has been created or updated, the microcontroller sends it to smart glasses. This cannot be accomplished if the devices are not reachable to each other. Hence, a router shown in Figure 4.4 is used in the system to create a network and provide internet accesses to all the connected devices. Consequently, all devices can communicate to each other via TCP/IP. TCP/IP (Transmission Control Protocol/Internet Protocol) is the basic communication protocol of the Internet. It can also be used as a communication protocol in a private network (either an intranet or an extranet). When a user sets up a computer with direct access to the Internet, the user's computer is provided with a copy of the TCP/IP program just as every other computer that the user may send messages to or get information from also has a copy of TCP/IP [13]. TCP/IP uses the client/server model of communication in which a computer user (a client) requests and is provided a service (such as sending a Web page) by another computer (a server) in the network. In our case, clients are people with hearing difficulties who are wearing smart glasses and server is the microcontroller.

Figure 4.4: A picture of Billion 5200W-TR2

4.3.4 Smart glasses (EPSON Moverio BT-200)

Figure 4.5 gives an image of Epson BT-200. Epson Moverio BT-200 is one of the smart glasses from Moverio product line of Epson. Currently, Epson Moverio BT-300 is the newest version of Moverio smart glasses.

Augmented reality and head-mounted displays are funky technologies that haven't quite found their place yet, even while many other forms of wearable technology have become widely adopted. Google Glass is the most well-known example that marries the two concepts, offering a display over the eye and the ability to take pictures, scan information, among other things, wherever you are [24]. When the research was started, it became impossible to buy a new Google Glass because Google decided to stop manufacturing it. So we decided to go with Epson Moverio BT-200 instead. Each lens of it has its own display. It is 960x540 QHD/60Hz dual display, and it comes with an OMAP4460 processor, a 24bit full color (16770k color), a built-in GPS, a compass, a gyroscope, an accelometer, a microphone, and a VGA camera.

Figure 4.5: A picture of Epson Moverio BT-200

The smart glasses run on Android OS 4.0.4. The device that looks like a phone in Figure 4.5 is a controller. It gives the user an idea of having an Android phone but instead of having normal screen display on the phone, it becomes displays on the smart glasses.

Since it operates on Android, an Android application was developed to use in the system. It is just a dark dim screen, which has a text shown in the middle bottom of the screen.

CHAPTER 5

SOFTWARE ARCHITECTURE

This chapter describes the overview of software architecture in the system. Software architecture overview provides basic ideas and summary of the system software architecture. The first part of Python speech recognition library explains about capabilities and limitations of the library, including comparison of speech recognition services from several companies. The optimized version of python speech recognition library explains about what techniques and concepts are used to optimize the library in order to meet the requirements of the research. Finally, how captions are generated is explained in caption creating process.

5.1 Software Architecture Overview

In Figure 5.1, this picture shows an overview of software architecture. The input of the system is audio data. An example is a speech of a person such as a lecturer in the classroom. Electronic devices usually store audio data in form of bytes. Few seconds of audio data contain a chuck of bytes. Therefore, the created real-time speech recognition system reads these chucks of bytes, converts them into FLAC files and sends them to Google speech recognition service. After Google speech recognition service finishes processing the audio data, it returns a list of transcripts back including both interim and final results. Next, the list of transcripts is passed into caption creating function to process a readable caption. Finally, the newly created caption is sent to smart glasses that are connected to the microcontroller in the same network via TCP/IP.

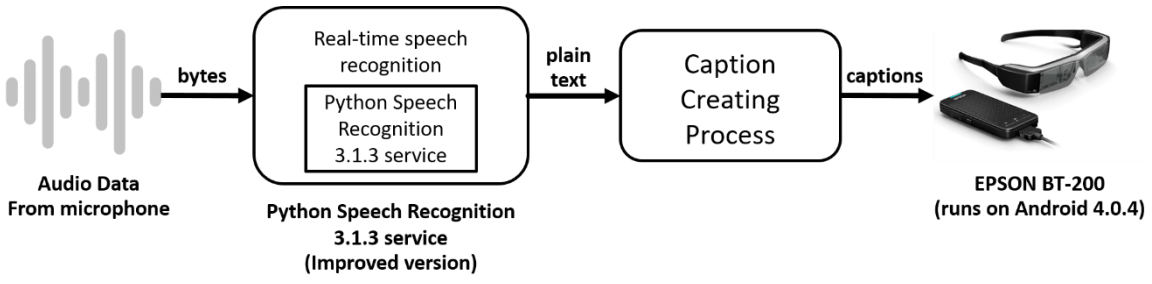

Figure 5.1: Software architecture overview

5.2 Python Speech Recognition Library (original version)

Since the system is developed using Python, only few libraries that are involved with speech recognition are available. Python speech recognition library is very efficient. It is an open source library for performing speech recognition, with support for several engines and APIs, online and offline [25]. It supports plenty of speech recognition engine/API such as:

- CMU Sphinx (works offline)
- Google Speech Recognition
- Wit.ai
- Microsoft Bing Voice Recognition
- Houndify API
- IBM Speech to Text

Currently, the newest version of the library is version 3.5.0. However, the version that is used in the system is version 3.4.6. The library supports Python 2.6, Python 3.3 and above.

Some libraries are required to be installed in order to use all of the functionality of Python speech recognition library. Prerequisites are as following:

- Python 2.6, 2.7, or 3.3+ (required)
- PyAudio 0.2.9+ (required only if microphone input is used)
- PocketSphinx (required the Sphinx recognizer is used)
- FLAC encoder (required only if the system is not x86-based Windows/Linux/OS X)

Figure 5.2 demonstrates the sample run of the original version of the library.

5.2.1 Functionalities

Python speech recognition library handles and provides almost all of the fundamental functions for developers so that they can do speech recognition on audio files and data instantly without writing additional lines of code.

There are four main classes in the library that handle all those fundamental things I mentioned before. Here are these four classes [26]:

- Microphone(device_index = None, sample_rate = 16000, chunk_size = 1024)
- AudioFile(filename_or_fileobject)
- Recognizer()
- AudioData(frame_data, sample_rate, sample_width)

Microphone class

What the Microphone class does is to create a new Microphone instance, which represents a physical microphone on the computer. The microphone audio is recorded in chunks of chunk_size samples, at a rate of sample_rate samples per second (Hertz).

Higher sample_rate values result in better audio quality, but also more bandwidth (and therefore, slower recognition). Additionally, some machines, such as some Raspberry Pi models, can't keep up if this value is too high.

Higher chunk_size values help avoid triggering on rapidly changing ambient noise, but also makes detection less sensitive. This value, generally, should be left at its default.

AuidioFile class

What the AudioFile class does is to create a new AudioFile instance given a WAV/AIFF/FLAC audio file filename_or_fileobject. If filename_or_fileobject is a string, then it is interpreted as a path to an audio file on the filesystem. Otherwise, filename_or_fileobject should be a file-like object such as io.BytesIO or similar.

WAV files must be in PCM/LPCM format [27]; WAVE_FORMAT_EXTENSIBLE and compressed WAV are not supported and may result in undefined behavior. Both AIFF and AIFF-C (compressed AIFF) [28] formats are supported. FLAC files must be in native FLAC [9] format; OGG-FLAC [29] is not supported and may result in undefined behavior.

Recognizer class

This class is considered to be the most important class in this library. It creates a new Recognizer instance, which represents a collection of speech recognition settings and functionality. A lot of important settings are in this class. Here are examples:

- energy_threshold - Represents the energy level threshold for sounds. Values below this threshold are considered silence, and values above this threshold are considered speech. It can be changed.
- dynamic_energy_threshold - Represents whether the energy level threshold for sounds should be automatically adjusted based on the currently ambient noise level while listening.
- dynamic_energy_adjustment_damping - If the dynamic energy threshold setting is enabled, represents approximately the fraction of the current energy threshold that is retained after one second of dynamic threshold adjustment.
- dynamic_energy_adjustment_ratio - If the dynamic energy threshold setting is enabled (see recognizer_instance.dynamic_energy_threshold), represents the minimum factor by which speech is louder than ambient noise.

- pause_threshold - Represents the minimum length of silence (in seconds) that will register as the end of a phrase.

Furthermore, main functions such as listening to an audio, recording an audio, and etc. are also in this class as well. Here are examples:

- record(source, duration = None, offset = None) - Records up to duration seconds of audio from source (an AudioSource instance) starting at offset (or at the beginning if not specified) into an AudioData instance, which it returns. If duration is not specified, then it will record until there is no more audio input.
- adjust_for_ambient_noise(source, duration = 1) - Adjusts the energy threshold dynamically using audio from source (an AudioSource instance) to account for ambient noise. This value should be at least 0.5 in order to get a representative sample of the ambient noise.
- listen(source, timeout = None) - Records a single phrase from source (an AudioSource instance) into an AudioData instance, which it returns. The timeout parameter is the maximum number of seconds that it will wait for a phrase to start before giving up.
- listen_in_background(source, callback) - Spawns a thread to repeatedly record phrases from source (an AudioSource instance) into an AudioData instance and call callback with that AudioData instance as soon as each phrase are detected.
- recognize_google(audio_data, key = None, language = "en-US", show_all = False) - Performs speech recognition on audio_data (an AudioData instance), using the Google Speech Recognition API.

AudioData class

What the AudioData class does is to create a new AudioData instance, which represents mono audio data. The raw audio data is specified by frame_data, which is a sequence of bytes representing audio samples. This is the frame data structure used by the PCM WAV format. The width of each sample, in bytes, is specified by sample_width. Each group of sample_width bytes represents a single audio sample. The audio data is assumed to have a sample rate of sample_rate samples per second (Hertz).

```
do you want to go to school today
Google Response Time: 0.721739
I want to go to school today
Google Response Time: 0.538682
I want to go to school today do you know that what I mean right
Google Response Time: 1.516012
what do you mean
Google Response Time: 0.709241
I have been in Japan for one and a half year
Google Response Time: 1.076440
```

Figure 5.2: A sample run of Python speech recognition library

5.2.2 Comparison among speech recognition services

Experiments were conducted in two aspects, speed and accuracy.

Speed

Figure 5.3 represents times used by three speech recognition services for two seconds of audio data. When the experiment was conducted, these three speech recognition services which are from Google, Wit.ai, and IBM were available. The speed of the speech recognition services were measured by sending the same audio data for four times. As it can be seen from the graph, Google speech recognition service shows gratified results. It outperforms all the other services with around 75% lower response time than the other services. Hence, in order to minimize the delay of the system as much as possible, using Google speech recognition service is proved to be the best choice for the system.

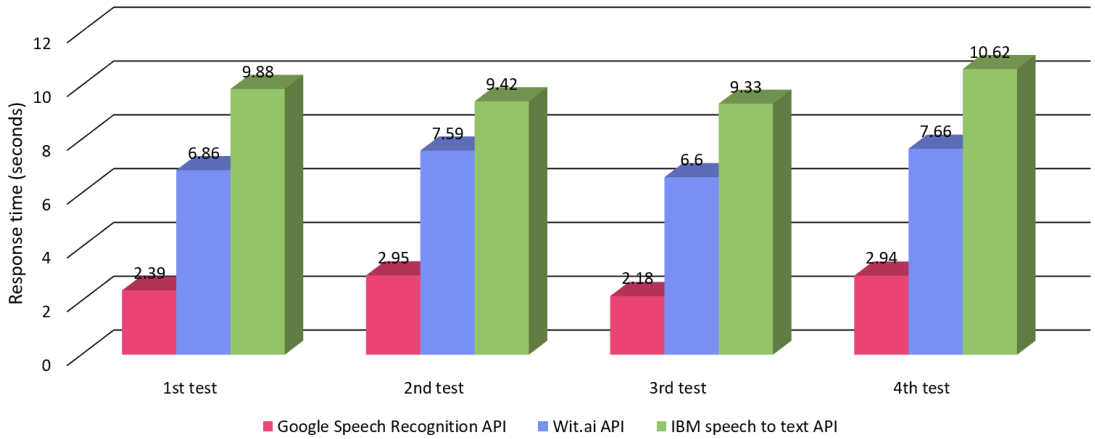

Figure 5.3: Times used by various speech recognition services for two seconds of audio data

Accuracy

Accuracy of the captions is only one factor that we can't neither improve nor guarantee 100 percent correctness of the result. Each speech recognition service from companies uses their own speech recognition technologies to carry the best result as good as they can. So far, this research is very first one, which implements speech recognition service from Google which is told to be the best speech recognition service among the other companies to perform speech recognition in classrooms and other targeting environments. An experiment were conducted to compare each speech recognition services from each company in two aspects, which are volume of exact recognized phrases and word error rate (WER). Results are displayed below in Figures 5.4 and 5.5

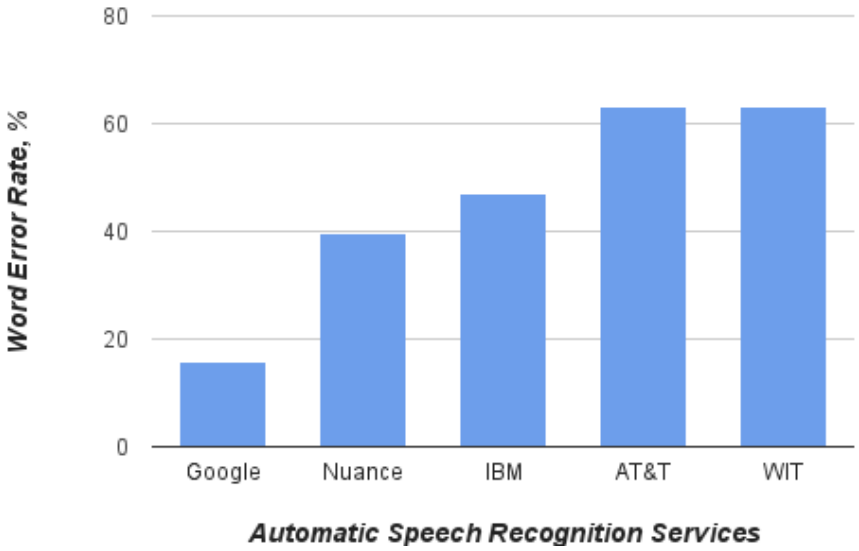

Figure 5.4: Word error rate result (lower is better)

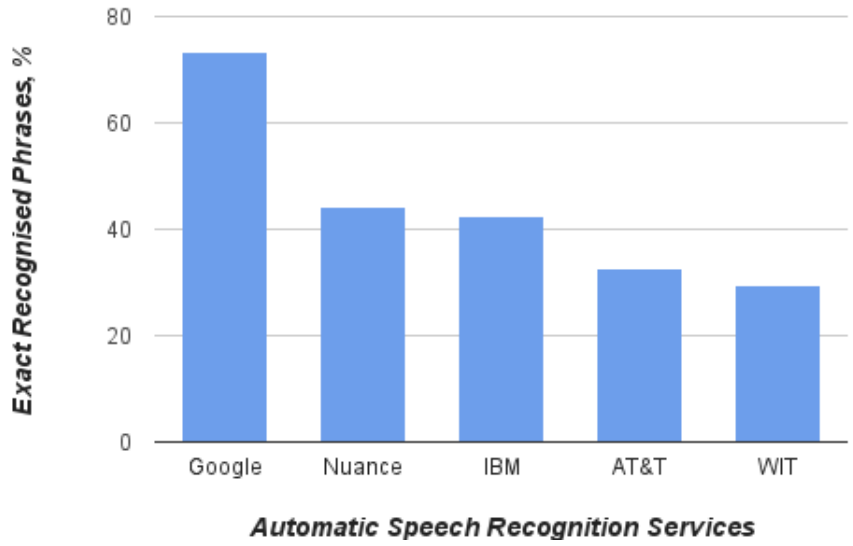

Figure 5.5: volume of exact recognized phrases

The speech recognition that achieved the best quality is provided by Google. The researcher who conducted the experiment could not reproduce the declared by Google 8% WER with their data, but the results are still impressive. Google achieved 73.3% of exact recognized phrases with a 15.8% WER [5].

5.2.3 Drawbacks of the original Python speech recognition library

Even though, original version of Python speech recognition provides almost all the basic functions that are necessary to perform speech recognition on a computer/microcontroller, it still has some drawbacks that make the original version of library do not match the requirements of our system. The drawbacks are as following:

- The library keeps listening to user's voice until there's no more sound and returns result after that.
- The response time (time which is used by a translation service to do a speech-to-text and send output back) is increased when the audio data keeps getting longer.
- Interim results are not available to be shown in real-time.

When a system wants to do something in real time, it means that the system processes inputs and delivers outputs simultaneously. The library is not capable of doing this. The main reason is an audio file is always sent to any speech recognition engine/API only after the library found the end of speech. If a speaker continuously keeps speaking without stops, no results are returned until the speaker finishes speaking. Moreover, interim transcripts are also not accessible by the original version of the library as well. Having interim transcripts of a speech is essential to create a feeling of real-time processing to users.

5.3 Python Speech Recognition Library (optimized version)

Regarding what stated in the purposes of the research, to perform speech recognition and provide captions in real-time and readable, the original Python speech recognition library cannot satisfy the purposes since it is not designed to work with the long and continuously speech. Therefore, we have developed an optimized version of the library.

Requirements that need to fulfill

- Interim result is streaming continuously.
- The correctness of interim results and final output must be acceptable.

- It does not take too long to get responses from speech recognition services and deliver a caption.
- It can be defined when a speech has been finished.
- It listens continuously in background.

The optimized version of the library uses the concepts of multithreaded programming. In other words, it has been modified to work in parallel. Figure 5.6 shows two groups of threads, which are the main thread and sub-threads.

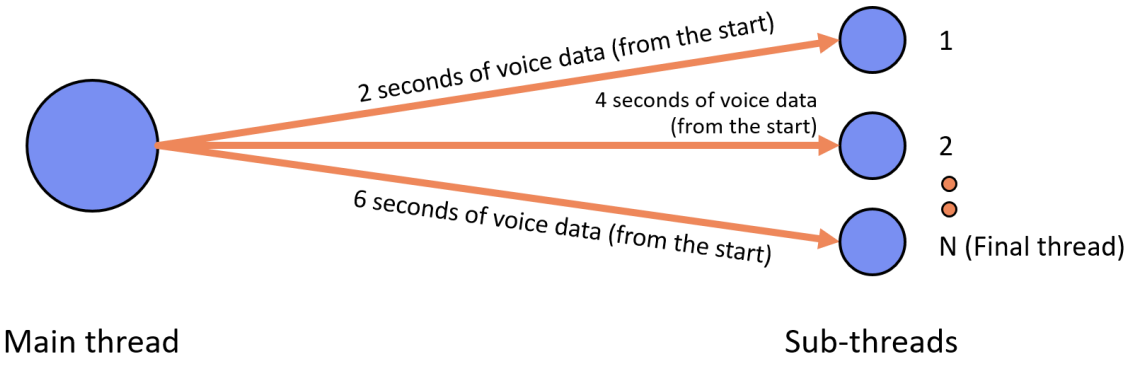

Figure 5.6: Overview of the optimized version of Python speech recognition library

These threads have their own responsibilities. Here are the responsibilities of both of the main thread and sub-threads:

Main thread:

- Listening to users continuously in background.
- Managing and preparing voice data for sub-threads.
- Creating captions.
- Managing connection between input (smartphone as a microphone) and output (smart glasses) devices.

Sub-threads:

- Sending voice data and handling received responses from Google.

When the system is started, the main thread begins to continuously listen to surrounding environment and sets the audio energy of surrounding environment as a threshold. When the energy of surrounding audio data goes higher than the threshold, the main thread starts recording audio data because somebody near to the microphone starts to speak.

While recording, the main thread creates a new instance of audio data every 2 seconds, creates a new sub-thread, and then sends that instance to it. After the new sub-thread receives an instance of audio data, it sends the instance, which is in the form of a FLAC file to Google speech recognition service to transcribe the audio and receives responses back as shown in Figure 5.7. Finally, all sub-threads send responses (raw data) back to the main thread for creating captions.

Figure 5.7: A work flow between sub-threads and Google speech recognition service

Table 5.1 points out the reason why the system creates the new instances every 2 seconds. It shows a comparison of system performance between three choices of time which are one, two and three seconds. According to the experimental result, the differences of several aspects are not significantly different such as accuracy, speech recognition used time, average delay between each captions, and first caption and last caption appear times. However, in term of resource consumption, one second result uses 25 sub-threads in total to achieve the similar result while two and three second results use only 12 and 7 sub-threads. So between two and three second results, two second result achieves slightly better performance over three second result as it can be seen in the table. Therefore, as the result, we decided to use two seconds as a separator of new instances of audio data.

Table 5.1: Experimental result of system performance between three periods of time

	Accuracy	Speech recognition used time	First caption appear time	Last caption appear time	Average delay between captions	Amount of responses	Amount of sub-threads used
One second	74%	6.06s	11.39s	83.28s	3.99s	19	25
Two seconds	77%	6.35s	12.83s	82.77s	4.11s	18	12
Three seconds	76%	5.08s	12.61s	83.87s	3.75s	20	7

5.4 Caption Creating Process

An overview of caption creating process is shown in Figure 5.8. Readability is another one of the two most important goals we focus on in this research beside from transcribe continuous speeches in real-time. After each response (raw data) from each sub-thread is received, the main thread starts caption creating process to produce an appropriate length of captions. Figure 5.9 illustrates an example of responses from Google speech recognition service (raw data).

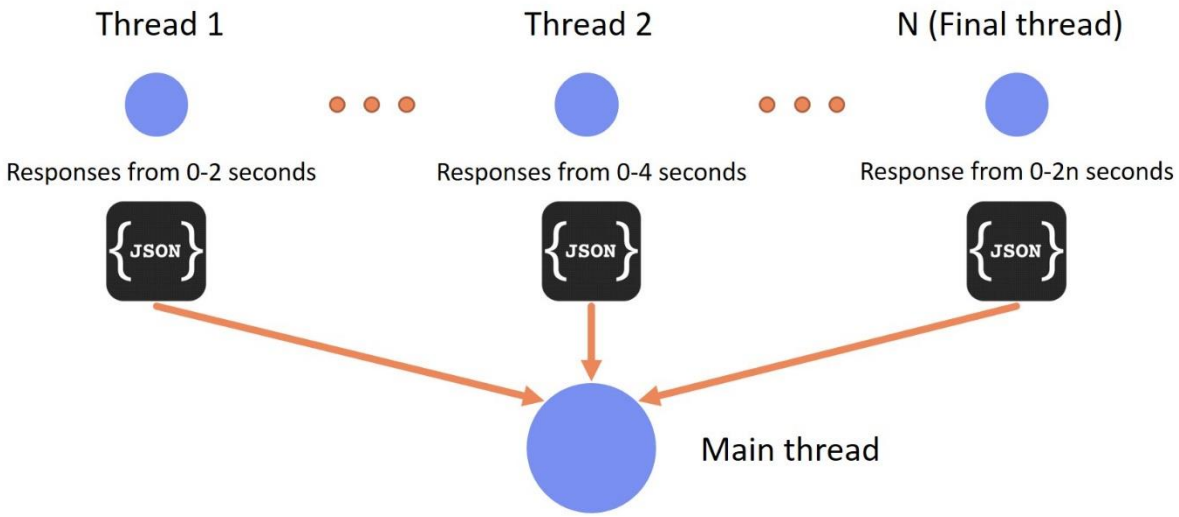

Figure 5.8: Overview of caption creating process

The system produces a caption by comparing and combining transcripts in responses. Each row (one row corresponds to an interim result) contains one or more possible transcripts which consist of plain text of what Google speech recognition service transcribed from a part of the whole audio data. One audio data is likely to contain more than 10 consecutive responses. One response is just a result from a very short amount of time in the audio data. The process of creating captions is explained below:

1. Getting JSON response from sub-threads.
2. Reading each line in responses (interim results).
3. Taking first transcript from the response.
4. If the length of the transcript is longer than the current caption.
5. Replace the current caption with the transcript.
6. Repeating this process until we found “final” = true”.
7. Replace the whole caption with the final transcript.

Figure 5.9: Responses from Google speech recognition service (raw data)

The system always tries to make a caption completed and meaningful as much as possible in terms of linguistic. Figure 5.10 shows a result after caption creating process.

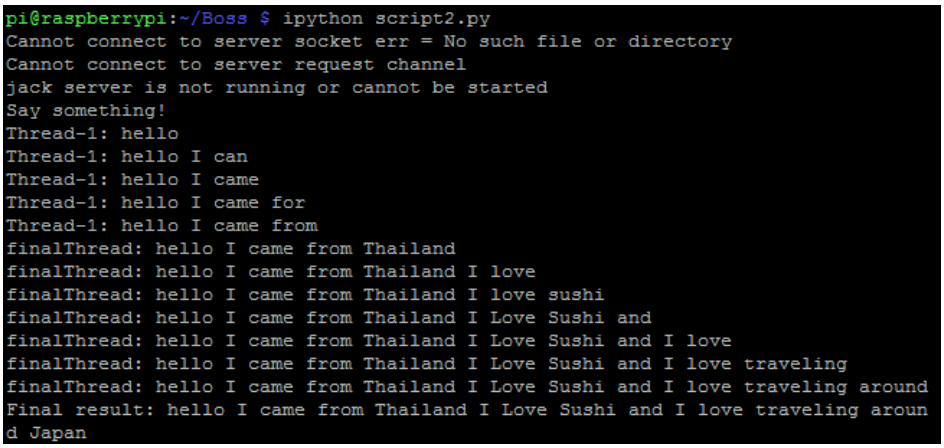

Figure 5.10: Result after caption creating process

CHAPTER 6

EXPERIMENTS & RESULT ANALYSIS

This chapter describes the experiments that were conducted to evaluate the system and their results. System performance shows the measurements of the system in several aspects, for example, string similarity of the transcripts and speech recognition used times. System effectiveness explains how well the system can help people with hearing difficulties to live with normal people in real life situations. Finally, system usability illustrates how much participants in the experiments are satisfied with the system.

6.1 System Performance

6.1.1 Introduction

In system performance experiments, two datasets were used to measure the performance of the system. These two datasets represent two conversation scripts in two different situations. In real life situation, conversation can be vary, regarding complexity of topics. Therefore, the system should not be test with only one conversation in order to reflex the true performance of the system. Several aspects such as speed, delay and accuracy of the captions were tested. These aspects are explained later on in this chapter.

However, because of the current limitation of speech recognition technology, the system is designed to be used in situations where only one speaker is speaking at a time. The accuracy of transcripts can be significantly affected by environmental noises that are caused by other speakers. Also, in this research, the concept is to use speech recognition as a service, not implement on our own. Hence, the only way to get most accurate transcripts is to provide clearest audio data with low environmental noise to Google speech recognition service.

6.1.2 Experimental method

There are, in total, 22 participants in the experiments Participants are divided into two groups which use different datasets. Each participant will be wearing the smart glasses and watching the prepared video of the dataset. However, instead of hearing the sound like normal people, the sound is mute and heard by the system only so that participants can act like people with hearing difficulties, and can read captions which are being shown on the smart glasses only. Finally, they are required to answer and finish the quiz of the dataset as fast as they can.

The experimental results of each participant such as captions are stored in the system so that, later on, the results will be evaluated to demonstrate the system performance.

6.1.3 Correctness of captions

Table 5.1 illustrates the results of caption accuracy of each participant compared to the original script of the dataset. The technique called Jaro–Winkler [30] distance is used to measure of similarity between two strings. As a result displayed in Figure 6.1, Google speech recognition service gives an average of 77.19% of accuracy among these two datasets.

Table 6.1: Each participant's result of caption accuracy

Dataset 1	Dataset 2
80.77%	76.27%
81.09%	77.03%
75.93%	76.55%
80.64%	76.34%
79.03%	78.53%
80.39%	80.61%
81.20%	72.22%
80.72%	76.10%
80.74%	72.25%
77.31%	63.67%
71.78%	78.93%

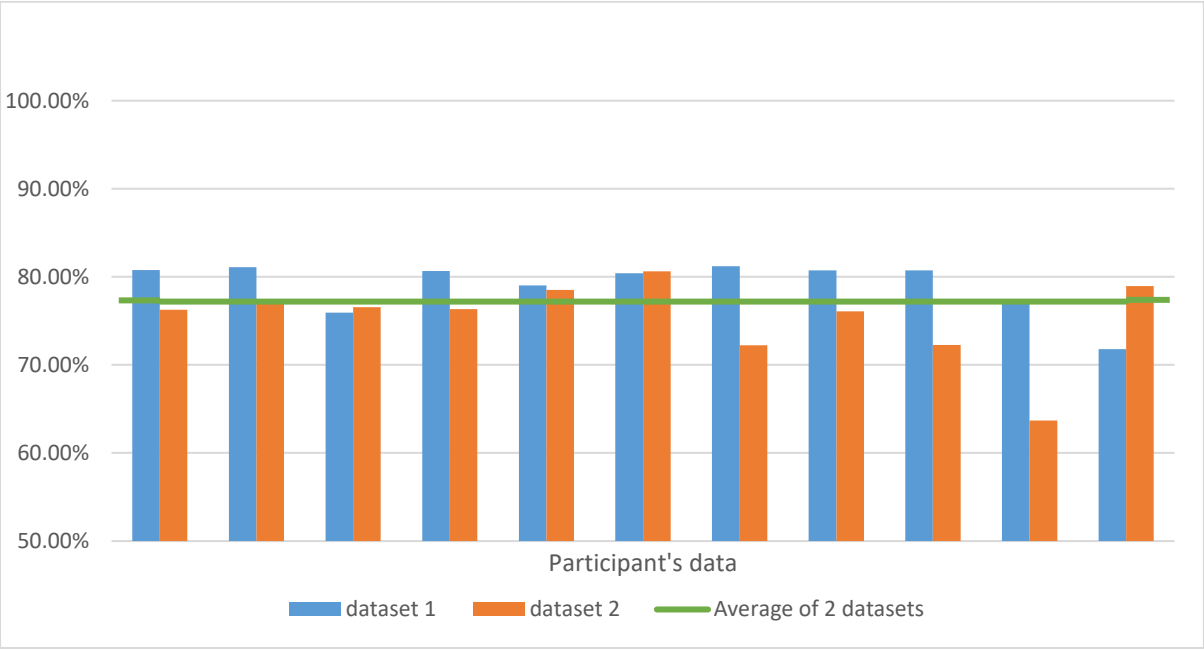

Figure 6.1: Correctness of captions

6.1.4 Average used time on audio data

Table 6.2 shows the system average used time on audio data. Noted that the audio data that were used to test are completed audio data which contained a whole speech, not only just a part of it. In other words, the result of this audio data is a final result, a fully completed caption which usually takes more time than getting a caption of interim results. Moreover, the times in the table represent the whole completed procedure which consists of doing http request and performing speech recognition on the audio data by Google. As a result shown in Figure 6.2, the system used approximately 4.88 seconds to completed the whole procedure with the average standard deviation of 0.72 seconds

Table 6.2: Results of the system average used time on audio data of each participant

Average used time of dataset 1 (second)	Average used time of dataset 2 (second)	Standard deviation of dataset 1 (second)	Standard deviation of dataset 2 (second)
3.71	5.79	0.64	0.69
3.61	6.02	0.51	0.79
5.64	5.99	0.68	0.64
3.88	5.92	0.84	0.64
3.69	6.74	0.78	1.37
3.59	6.29	0.55	0.98
6.11	3.15	0.96	0.33
5.74	5.81	0.68	0.75
5.76	5.72	0.76	0.75
3.30	3.25	0.43	0.49
3.53	4.08	0.71	0.81

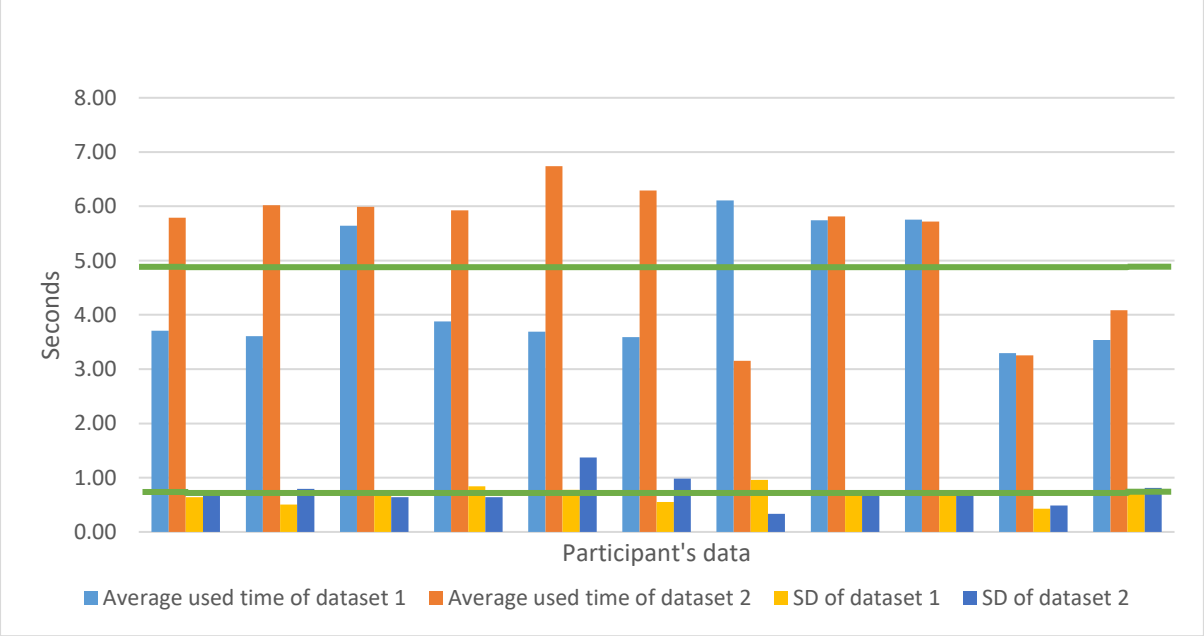

Figure 6.2: The system average used time on audio data

6.1.5 Average delay of the first and the last captions

Table 6.3 shows the result of first caption showed times that we got from each participant. Noted that the audio data of each participant that is sent to Google speech recognition service are not exactly need to be the same length of time since it totally depends on when the system defines the end of a speech. Therefore, the first caption of each participant could be different. The ones that used longer time tend to result in longer captions as well because generally, the longer the audio data is, the longer time the system uses to process and produce a result. As displayed in Figure 6.3, an average of first caption delays of two datasets is 15.63 seconds.

Table 6.3: Showed times and delays of the first caption of each participant

First speech of dataset 1 end at (second)	First caption of dataset 1 showed at (second)	Difference of dataset 1 (second)	First speech of dataset 2 end at (seconds)	First caption of dataset 2 showed at (second)	Difference of data set 2 (second)
12.00	24.79	12.79	13.00	25.75	12.75
12.00	31.25	19.25	13.00	27.82	14.82
12.00	24.88	12.88	13.00	28.39	15.39
12.00	32.22	20.22	13.00	25.85	12.85
12.00	45.73	33.73	13.00	31.87	18.87
12.00	37.52	25.52	13.00	25.47	12.47
12.00	20.46	8.46	13.00	25.27	12.27
12.00	25.16	13.16	13.00	30.14	17.14
12.00	21.16	9.16	13.00	25.76	12.76
12.00	30.11	18.11	13.00	27.29	14.29
12.00	25.65	13.65	13.00	26.23	13.23

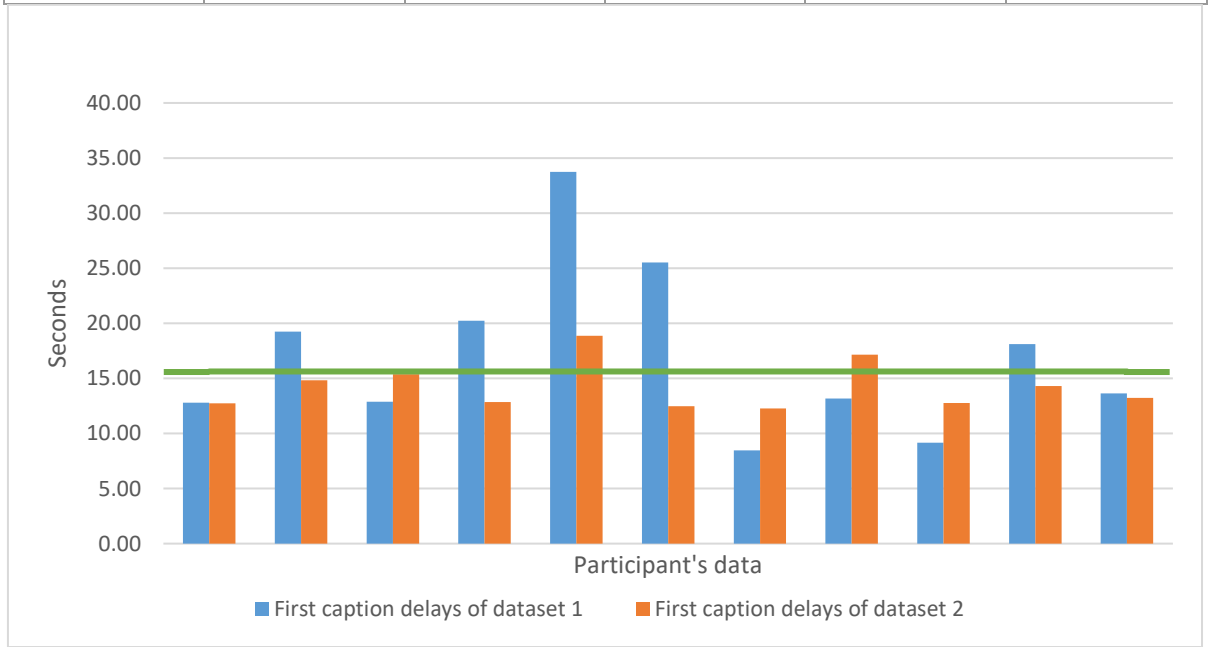

Figure 6.3: An average of first caption delays in both datasets

Likewise what describe above, Table 6.4 shows the result of last caption showed times and the differences of them. Figure 6.4 illustrates an average of last caption delays which is 15.99 seconds.

In summary, compared to normal people, people with hearing difficulties who are using the system will take an approximate of 15.30 seconds more to completely understand the same speech.

Table 6.4: Showed times and delays of the last caption of each participant

Last speech of dataset 1 end at (second)	Last caption of dataset 1 showed at (second)	Difference of dataset 1 (second)	Last speech of dataset 2 end at (seconds)	Last caption of dataset 2 showed at (second)	Difference of data set 2 (second)
97.00	112.39	15.39	83.00	93.53	10.53
97.00	117.97	20.97	83.00	96.77	13.77
97.00	112.68	15.68	83.00	96.57	13.57
97.00	119.17	22.17	83.00	92.43	9.43
97.00	127.19	30.19	83.00	100.13	17.13
97.00	120.80	23.80	83.00	94.24	11.24
97.00	108.11	11.11	83.00	108.70	25.70
97.00	113.17	16.17	83.00	96.85	13.85
97.00	108.24	11.24	83.00	91.64	8.64
97.00	113.75	16.75	83.00	97.37	14.37
97.00	113.47	16.47	83.00	96.68	13.68

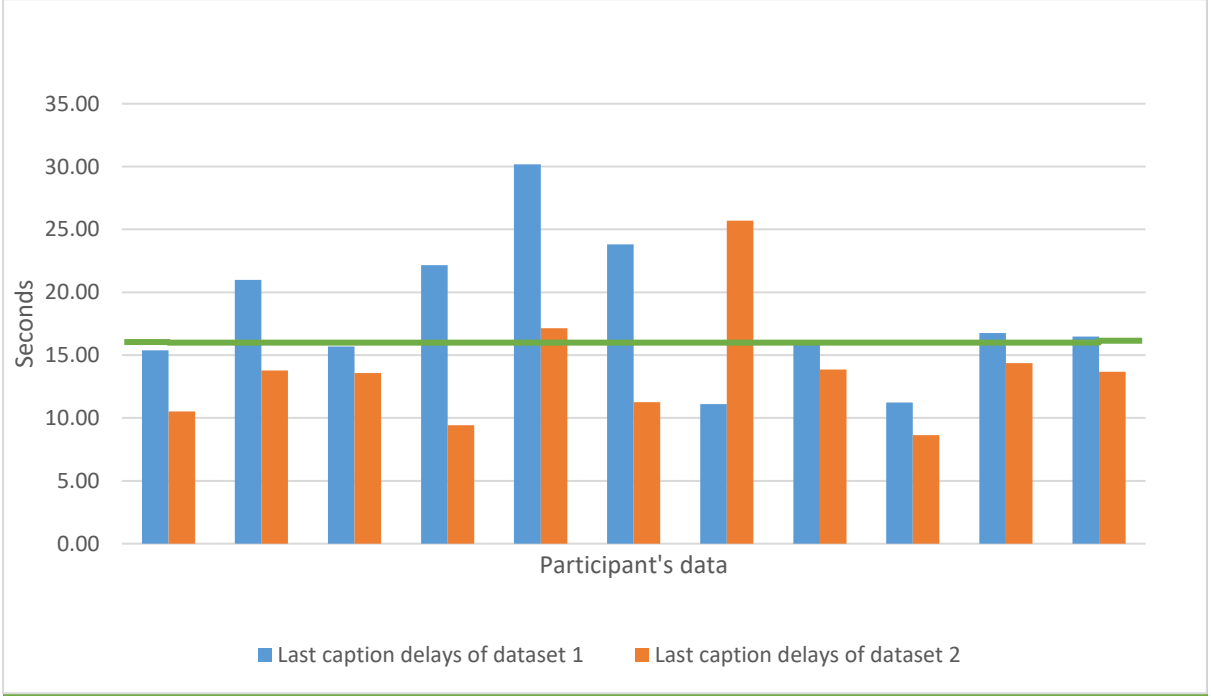

Figure 6.4: An average of last caption delays in both datasets

6.1.6 Average delay between captions

Table 6.5 shows the results of participant's average delays between captions. Figure 6.5 shows an average of two datasets which is 6.02 seconds with standard deviation of 2.88 seconds. In conclusion, after the first caption is shown on the smart glasses, the system shows another completed caption in every 6.02 seconds. Note that before a completed caption is shown, there are interim results of the completed caption being shown on smart glass in real time.

Table 6.5: Each participant's result of average delays between captions

Average delay between captions of dataset 1	Average delay between captions of dataset 2	Standard deviation of dataset 1	Standard deviation of dataset 2
5.84	6.64	2.79	3.02
5.78	6.90	2.67	2.47
5.85	6.82	2.47	2.20
5.80	6.66	3.51	3.11
5.82	7.58	3.16	3.31
5.55	6.88	2.48	2.13
6.26	4.17	2.83	2.70
5.87	5.56	2.41	2.94
5.81	6.59	2.72	3.93
4.92	5.01	2.34	3.52
5.85	6.40	3.31	3.27

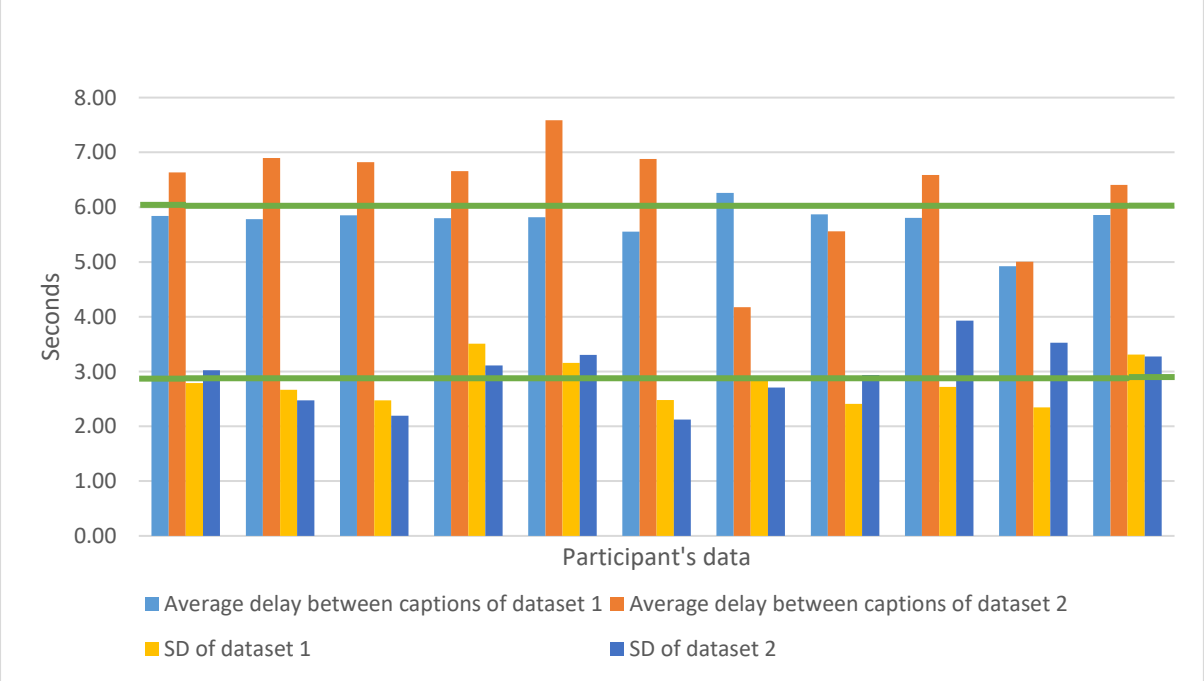

Figure 6.5: An average of participant's average delays between captions

6.1.7 Internal server error rate of Google speech recognition

Table 6.6 shows successful and failed responses from doing http requests to Google speech recognition service when conducted the experiment. In reality, failure from performing http request is unavoidable. Hence, it's important to know the rate of internal server error of Google speech recognition. As it can be seen from Figure 6.6, the system did 328 times of http requests in total, and there are 15 times of internal server error occurred. We can conclude that the internal server error rate of Google speech recognition is approximately 4.57%.

Table 6.6: Successful and failed responses returned from Google SR of participants

Successful responses of dataset 1	Successful responses of dataset 2	Error responses of dataset 1	Error responses of dataset 2
16	11	0	3
16	11	0	1
16	11	1	1
16	11	0	2
15	10	0	1
16	11	0	1
15	21	1	0
16	13	0	2
16	11	0	2
18	15	0	0
16	12	0	0

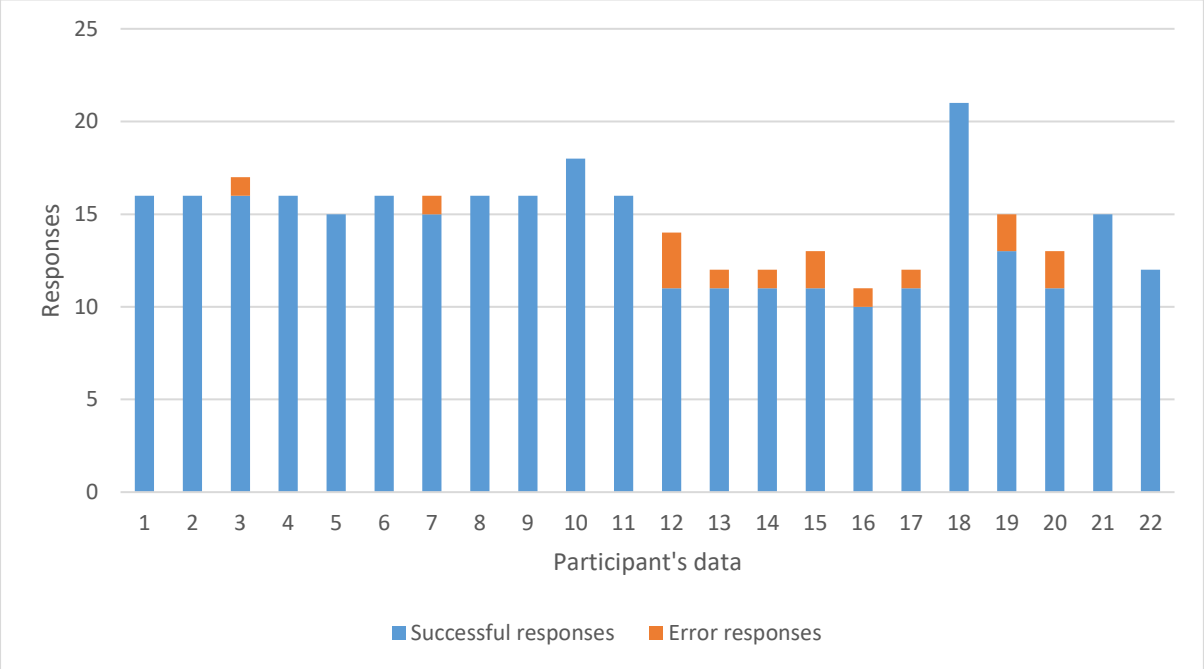

Figure 6.6: A graph visualized successful and failed responses from Google SR

6.2 System Effectiveness

This section demonstrates how much the system can help people with hearing difficulties to follow and understand conversation in the same situation comparing with normal people.

6.2.1 Experimental method

In the experiment, 22 non-native English speakers from variety of counties are assigned into two groups which are uncontrolled and controlled groups. They are all physically complete, and have no problems of hearing difficulties. Uncontrolled group represents a group of normal people, and controlled group represents a group of people with hearing difficulties.

Participants in uncontrolled group will be giving a quiz, and then watch a video with sounds. They are required to finish the quiz as fast as they can. Participants in controlled group will also be doing the same thing but the sounds of the video will be mute so that the participants in this group can be considered as people with hearing difficulties.

Regarding system performance section, there are two datasets in the experiment. Therefore, after finished conducting first round of the experiment, all participants will be switching the groups and doing the experiment again with the other dataset. As a consequence, each group has altogether 22 participants. Score and time used by participants are recorded and shown below in the table 6.7.

We also separated participants regarding their English level of competency. Orange, blue and green areas in table 6.7 display high, intermediate and low levels of English competency of the participants respectively. The criteria we used to split participant English competency levels is by using TOEIC score. Participants who get TOEIC score more than 700 are qualified to be in high level of English competency group. Participants who get TOEIC score more than 500 are qualified to be in intermediate level of English competency group. Lastly, participants who get TOEIC score lower than 500 are qualified to be in low level of English competency group.

6.2.2 Result

Data illustrated in Table 6.7, Figures 6.7, 6.8, and Table 6.8 show averages and differences of the data. In the same quiz, a non-native English speaker would get an average score of 4.86 and spend approximately 2 minutes to finish the quiz while a non-native English speaker with hearing difficulties who is being assisted by the system would get an average score of 4.14 and spend approximately 2 minutes and 23 seconds to finish the quiz.

Table 6.7: Quiz scores of three different groups of participants

Nationality	Uncontrolled group (score)	Time (second)	Controlled group (score)	Time (second)
Philippines	5	100	5	126
Philippines	6	100	5	138
Japan	6	100	3	131
Bangladesh	6	100	5	120
Malay	6	104	5	106
Malay	6	104	5	121
Indonesian	6	99	5	99
Vietnamese	6	100	3	152
Average	**5.88**	**100.88**	**4.50**	**124.13**
Thai	4	100	5	156
Japan	3	163	3	172
Japan	4	175	5	270
Japan	4	164	4	141
Thai	5	100	6	139
Korean	4	105	3	118
Japan	4	140	1	185
Japan	6	115	2	155
Japan	5	120	5	158
Japan	6	105	5	130
Japan	4	100	2	130
Japan	4	100	4	130
Average	**4.42**	**123.92**	**3.75**	**157.00**
Japan	4	178	4	103
Japan	3	155	6	172
Average	**3.50**	**166.50**	**5.00**	**137.50**

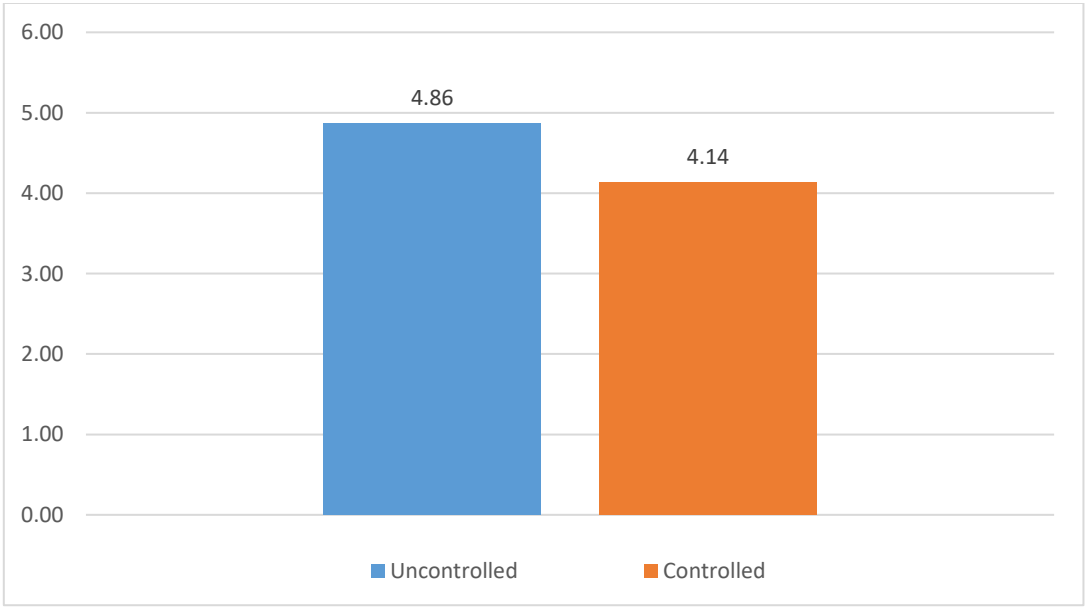

Figure 6.7: Average scores of uncontrolled and controlled groups

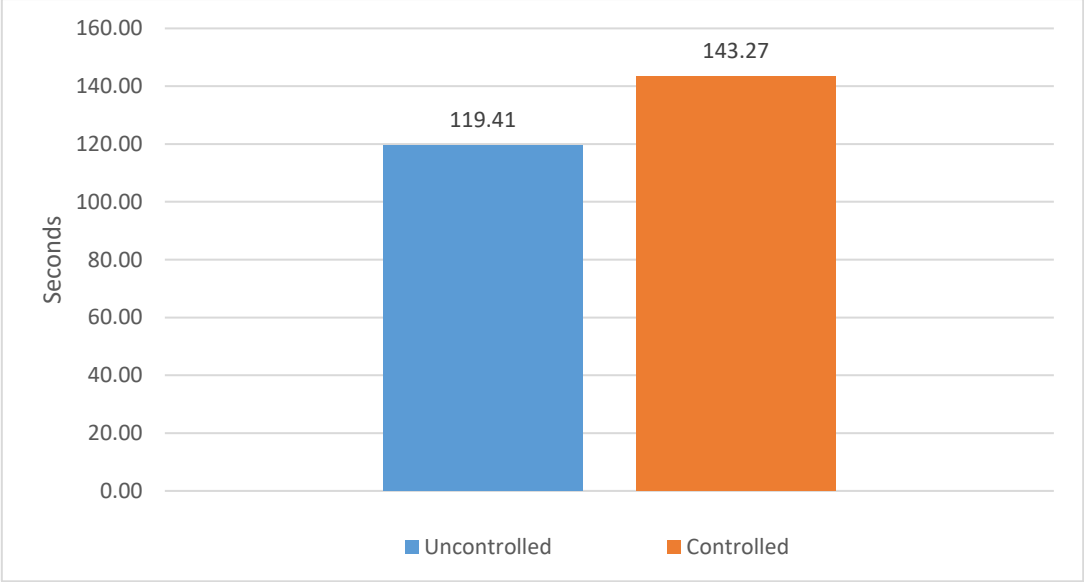

Figure 6.8: Average used times on the quiz of uncontrolled and controlled groups

Table 6.8: Differences between normal people and people with hearing difficulties

	Non-native English speaker without hearing difficulties	Non-native English speaker with hearing difficulties	Difference
Average level of understanding	4.86 (81%)	4.41 (73.5%)	0.45 (7.5% lower)
Average time used	119.41 seconds	143.27 seconds	23.86 seconds (19.98% slower)

In conclusion, the experiment proved that the system can assist people with hearing difficulties to achieve 73.5% of understandability with the speed of just 20% slower than normal people in the same situation. The difference of understandability between normal people and people with hearing difficulties shown in table 6.8 is only 7.5% which we believe it does not make any significant differences in understanding among two groups of people.

6.3 System Usability

This section describes evaluation and analyzation the system feedbacks from the participants as it can be seen in Table 6.9 and Figure 6.9.

- **Experimental method**

After participants finished the quizzes, they will be asked to complete a survey about the system.

- **Result**

Some explanatory statements are needed here.

Table 6.9: Result of system usability evaluated by participants

Question	Average score	Standard Deviation
System satisfaction	3.27	1.08
Eyewear comfortableness	2.68	1.09
Font size appropriateness	3.86	1.17
Font color appropriateness	4.05	1.33
Font place appropriateness	3.77	1.19
Caption delay	3.36	1.05
Caption lasting time	2.82	1.18
Caption changing speed	2.36	1.05
Caption understandability	3.14	1.13

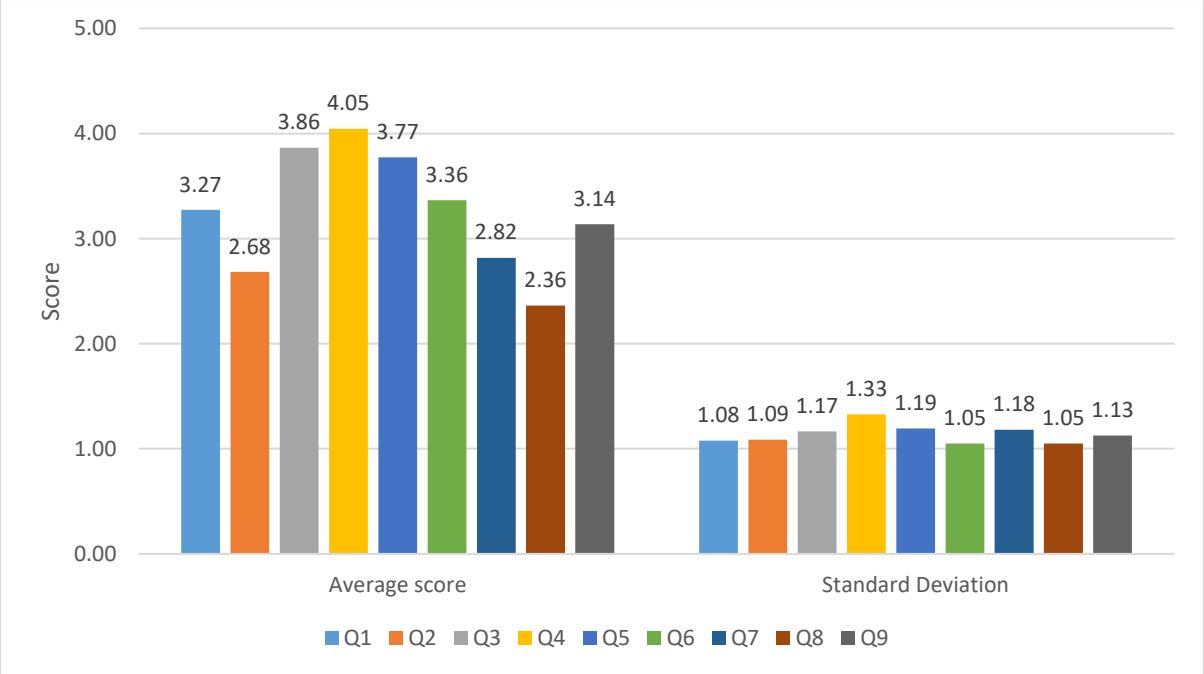

Figure 6.9: System usability result

CHAPTER 7

DISCUSSION & CONCLUSION

This chapter summarizes this research and concludes by recommending some future works.

7.1 Summary

The purpose of this research is to implement a speech recognition system on microcontroller which is connected to smart glasses as displays in order to support people with hearing difficulties to live, understand and communicate with normal people easier in certain situations where only one person speaks at a time.

Raspberry Pi 2 Model B is one of the most well-known and acceptable microcontroller by many developers all around the world. Therefore, it is chosen to be microcontroller of the system. The microcontroller connects with an USB microphone to record audio from users as inputs. The research selected Python to implement the system so the library called Python speech recognition is used to perform speech recognition. This library supports plenty of speech recognition engines from several companies. However, Google speech recognition service is picked by the system because of its outstanding performance that overwhelmed other speech recognition engines in both aspects of accuracy and speed. Furthermore, the research choses smart glasses from Epson called EPSON Moverio BT-200 to be a display for people with hearing difficulties. This is one of the big advantage in this research since the portability of smart glasses allows people with hearing difficulties to carry it around and use it anytime they want as long as the system is available in the area.

Nevertheless, Python speech recognition library still has some limitations that make it could not satisfy the requirements of the research. The biggest issue is it was not designed to perform speech recognition continuously. For this reason, the optimized version of Python speech recognition library that can handle continuous inputs, perform speech recognition, and able to create captions simultaneously was developed to fulfill the requirements. The optimized version of the library uses the technique of multithreaded programming to achieve the goal.

The experiment was conducted to evaluate performance, effectiveness and usability of the system. 22 non-native English speakers participated in the experiment. All of them did the experiment for two times under different conditions (with and without sound) in order to simulate the different groups of people (normal people and people with hearing difficulties). Two datasets were prepared to avoid problems of data duplication which may consequence in inconsistency of evaluation.

Experimental results revealed that people with hearing difficulties who are being assisted by the system have slightly inferior understanding of contents than normal people in the same situation. Besides, the difference between times taken by the people with hearing difficulties and normal people is not considerably different as well.

7.2 Future Works

This section demonstrates some useful functions which could be included in the future are as following:

- **Multilingual support**

Right now, the system only supports English as an input language. If a user speaks in a language which is not English, the system will absolutely misinterpret the input and give irrelevant results. Therefore, the system should be able to automatically define the input language by itself in the future.

- **Automatic translation**

There could be a situation where speakers and listeners are from different countries. It would be great if the system is able to know the native languages of a user, and then translate captions into user native language instantly.

- **Wireless microphone support**

Currently, the library that is working on managing and handling microphone in Raspberry Pi has not supported wireless modules of microphone yet. Speakers will surely be more comfortable with wireless microphone while using the system.

- **Bi-directional streaming support**

The biggest obstacle to optimize speed of the system right now is from doing http request. Because Python speech recognition library still does not support the new version of Google speech recognition called Google cloud speech API which is already designed to support bi-directional streaming, it is still unavoidable to do http request every times when there is new audio data. Technically, the cost of doing http request frequently and rapidly is very expensive. Uploading data, processing data and returning output cannot be done simultaneously with the old version of Google speech recognition service. Therefore, using the new version would increase the speed of the whole system by at least 10 times faster.

- **Individual settings**

Since users have different backgrounds of knowledge and physical conditions, the appropriate font size, color or caption changing speed can be vary to suit different users. So the system would be more friendly and convenient to users if they can set these settings by themselves.

- **Support multiple speakers**

Another drawback of the current version of the system is that the system cannot guarantee acceptable level of caption accurateness if the audio data is mixed with the voices of two or more speakers speaking at the same time. Moreover, it cannot differentiate the voice of speakers as well. For this reason, the system should be able to separate two or more distinct voices and return two or more results regarding the voices in the future.

REFERENCES

1. Ranchal, R., Taber-Doughty, T., Guo, Y., Bain, K., Martin, H., Robinson, J. P., & Duerstock, B. S. (2013). Using speech recognition for real-time captioning and lecture transcription in the classroom. *IEEE Transactions on Learning Technologies*, Vol. 6, No. 4, pp. 299-311.
2. Wald, M., & Bain, K. (2005, January). Using automatic speech recognition to assist communication and learning. In *Proceedings of HCI International*.
3. Bain, K., Basson, S. H., & Wald, M. (2002, July). Speech recognition in university classrooms: liberated learning project. In *Proceedings of the Fifth International ACM Conference on Assistive Technologies*, pp. 192-196.
4. Wald, M., & Bain, K. (2007, July). Enhancing the usability of real-time speech recognition captioning through personalised displays and real-time multiple speaker editing and annotation. In *Proceedings of the International Conference on Universal Access in Human-Computer Interaction*, pp. 446-452.
5. Kudryavtsev, A. (2016, Jan 11). Automatic Speech Recognition Services Comparison [Web blog post]. Retrieved Jan, 6, 2017, from http://blog-archive.griddynamics.com
6. IBM (2005) Retrieved Jan, 6, 2017, from http://www-306.ibm.com/able/solution_offerings/ViaScribe.html
7. Wald, M. (2006). Creating accessible educational multimedia through editing automatic speech recognition captioning in real time. *Interactive Technology and Smart Education*, Vol. 3, No. 2, pp. 131-141.
8. Wald, M. (2006). Research and development of client-server personal display of speech recognition generated text, real time editing and annotation systems: Speech Technologies-Accessibility Inroads: A special symposium on accessibility and speech recognition technology. *IBM Hursley Research Park 2006*, Retrieved February 7, 2007.
9. FLAC (2004), Retrieved Jan 6, 2017, from https://xiph.org/flac/index.html
10. Tutorialspoint (2016), Retrieved Jan 6, 2017, from http://www.tutorialspoint.com/http/http_requests
11. S. Perez, I. Lunden (2016, Mar 23), Retrieved Jan, 6, 2017, from https://techcrunch.com/2016/03/23/google-opens-access-to-its-speech-recognition-api-going-head-to-head-with-nuance/
12. JSON (2017), Retrieved Jan 6, 2017, from http://www.json.org/
13. Rouse, M. (2008, Oct), Retrieved Jan 6, 2017, from http://searchnetworking.techtarget.com/definition/TCP-IP
14. Multithreading (2017), Retrieved Jan 6, 2017, from https://en.wikipedia.org/wiki/Multithreading_(computer_architecture)

15. Logitech HD Pro C910 (2017), Retrieved Jan 6, 2017, from https://www.cnet.com/products/logitech-hd-pro-c910/specs/
16. Raspberry Pi Supported USB Webcams (2017), Retrieved Jan 6, 2017, from http://elinux.org/RPi_USB_Webcams
17. Bluetooth Profiles (2017), Retrieved Jan 6, 2017, from https://en.wikipedia.org/wiki/List_of_Bluetooth_profiles
18. BlueZ (2017), Retrieved Jan 6, 2017, from http://www.bluez.org/about/
19. ALSA (2017), Retrieved Jan 6, 2017, from https://en.wikipedia.org/wiki/Advanced_Linux_Sound_Architecture
20. PulseAudio (2017), Retrieved Jan 6, 2017, from https://www.freedesktop.org/wiki/Software/PulseAudio/
21. Raspberry Pi 2 Model B (2017), Retrieved Jan 6, 2017, from https://www.raspberrypi.org/products/raspberry-pi-2-model-b/
22. Debian (2017), Retrieved Jan 6, 2017, from https://www.debian.org/
23. Raspbian (2017), Retrieved Jan 6, 2017, from https://www.raspbian.org/
24. Greenwald, W. (2014, Aug), Epson Moverio BT-200, Retrieved Jan 6, 2017, from http://www.pcmag.com/article2/0,2817,2462138,00.asp
25. Zhang, A. (2016). Speech Recognition (Version 3.4.6) [Software]. Available from https://github.com/Uberi/speech_recognition#readme.
26. Python Speech Recognition Reference (2016), Retrieved Jan 6, 2017, from https://github.com/Uberi/speech_recognition/blob/master/reference/library-reference.rst
27. PCM (2017), Retrieved Jan 6, 2017, from https://en.wikipedia.org/wiki/Pulse-code_modulation
28. AIFF (2017), Retrieved Jan 6, 2017, from https://en.wikipedia.org/wiki/Audio_Interchange_File_Format
29. OGG Maping (2004), Retrieved Jan 6, 2017, from https://xiph.org/flac/ogg_mapping.html
30. Jaro-Winkler Distance (2017), Retrieved Jan 6, 2017, from https://en.wikipedia.org/wiki/Jaro%E2%80%93Winkler_distance
31. Kheir, R., & Way, T. (2007). Inclusion of deaf students in computer science classes using real-time speech transcription. *ACM SIGCSE Bulletin*, Vol. 39, No. 3, pp. 261-265.
32. Leitch, D., & MacMillan, T. (2003). Liberated learning initiative innovative technology and inclusion: current issues and future directions for liberated learning research. *Year III Report*.
33. Francis, P. M., & Stinson, M. (2003). The C-Print speech-to-text system for communication access and learning. In *Proceedings of CSUN conference technology and persons with disabilities*.

34. Lambourne, A., Hewitt, J., Lyon, C., & Warren, S. (2004). Speech-based real-time subtitling services. *International Journal of Speech Technology*, Vol. 7, No. 4, pp. 269-279.
35. Ranchal, R., Taber-Doughty, T., Guo, Y., Bain, K., Martin, H., Robinson, J. P., & Duerstock, B. S. (2013). Using speech recognition for real-time captioning and lecture transcription in the classroom. IEEE Transactions on Learning Technologies, 6(4), 299-311.
36. Akita, Y., Kuwahara, N., & Kawahara, T. (2015, December). Automatic classification of usability of ASR result for real-time captioning of lectures. In Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2015 Asia-Pacific (pp. 19-22). IEEE.

RELATED PUBLICATION

1. Meeklai Satjapong, Yuki Hirai, Keiichi Kaneko, Vinh Le, Keisuke Kumagai and Tetsuo Takahashi: "CROCOTILE: A Learning Environment for C Language Learners with Tile Programming," Proceedings of the 2015 4th International Student Project Conference, Tokyo, Japan, May 23-24, 2015.
2. Meeklai Satjapong, Yuki Hirai, Keiichi Kaneko, Obara Kaede, Sripraprutchai Kantaporn: “Development of a Programming Experience System By Using Operation Cards,” Proceedings of ISERD international conference, pp. 1-7, Oxford, UK, March 15, 2016
3. Meeklai, Satjapong, Kaede Obara, Kantaporn Sripraprutchai, Yuki Hirai, and Keiichi Kaneko: "Development of a Programming Experience System by Using Operation Cards," International Journal of Management and Applied Science, Vol. 2, No. 5, pp. 31-37, May 2016.

APPENDICES

Appendix A A sample of participant's raw data from both datasets that is used to analyze and evaluate system performance (Section 6.2)

Appendix B Samples of the scripts and quizzes of the two datasets that is used analyze and evaluate system effectiveness (Section 6.3)

Appendix C A sample of system usability survey (Section 6.4)