Year 2 Project

Speech Recognition Calculator based on MATLAB

By: Ziheng Zhang (201220030), Yajie Mao (201219375),
Jiawen Han (201219008), Libo Ren (201219448),
Tom Beckingham (201104297)
Supervised by: Dr Kirsty McKay

Department of Electrical Engineering and Electronics

23 March 2017

Abstract

This document serves as the report of Year 2 project: Speech Recognition Calculator based on MATLAB. This project is intended to design and construct a digital calculator with voice input and speech recognition is accomplished in MATLAB. The full-adder, decoder and 7-segment are employed to achieve digital calculation, all of which are connected to one another consecutively followed by Arduino UNO. Software in MALTAB mainly adapts three algorithms, Voice Activity Detection (VAD), Mel-Frequency Cepstrum Coefficients (MFCC) and Dynamic Time Warping (DTW), for speech recognition. The final deliverable is an integrated circuit and Arduino UNO connected with a computer where graphical user interface is developed for overall control of the system. The MATLAB GUI guides user to reset the system, to input his/her voice into computer microphone that then transfers recognition results to Arduino UNO and to display the addition result on 7-segment.

Declaration

I confirm that I have read and understood the University's definitions of plagiarism and collusion from the Code of Practice on Assessment. I confirm that I have neither committed plagiarism in the completion of this work nor have I colluded with any other party in the preparation and production of this work. The work presented here is my own and in my own words except where I have clearly indicated and acknowledged that I have quoted or used figures from published or unpublished sources (including the web). I understand the consequences of engaging in plagiarism and collusion as described in the Code of Practice on Assessment (Appendix L).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The introduction part will cover some basic principles of this project, including characteristics of voice signal, signal processing within MATLAB, Voice Activity Detection (VAD), Mel-Frequency Cepstrum Coefficients (MFCC), Dynamic Time Warping (DTW) and other electronic components. All these algorithms will be adapted in the following software design and implementation.

1.1 Voice Signal & MATLAB

Human voice is a basic but important tool to pass information to one another, with which human can communication easily and accurately [1]. With the development of computer science and electronics, the study on voice signal analysis has made a considerable progress. It is currently possible to generate, transmit, store and access voice signal through electronical devices, such as telephones and computers [1].

Voice has two definitions that voice is either pressure changes in the air (or other medium) or the experience we have when we hear [2]. Voice is basically in the form of wave, so a specific voice signal can be analysed in term of three crucial wave-characteristics, amplitude, frequency and phase. As figure 1.1 shows, amplitude of signal represents the volume of voice, which can be perceived as loud or quiet. Figures 1.2 display that frequency of signal represents the pitch of voice and, for example, doubling the frequency increases the pitch by one octave [3]. Phase of signal has no significant effect on voice, but the phase difference, which might result from phase shifting, will transmute voice into stereo for human hearing. As Figure 1.3 displays, human hearing system is sensitive to amplitude and frequency, and only perceived the phase difference when the travel distances to right ear and left ear are not the same.

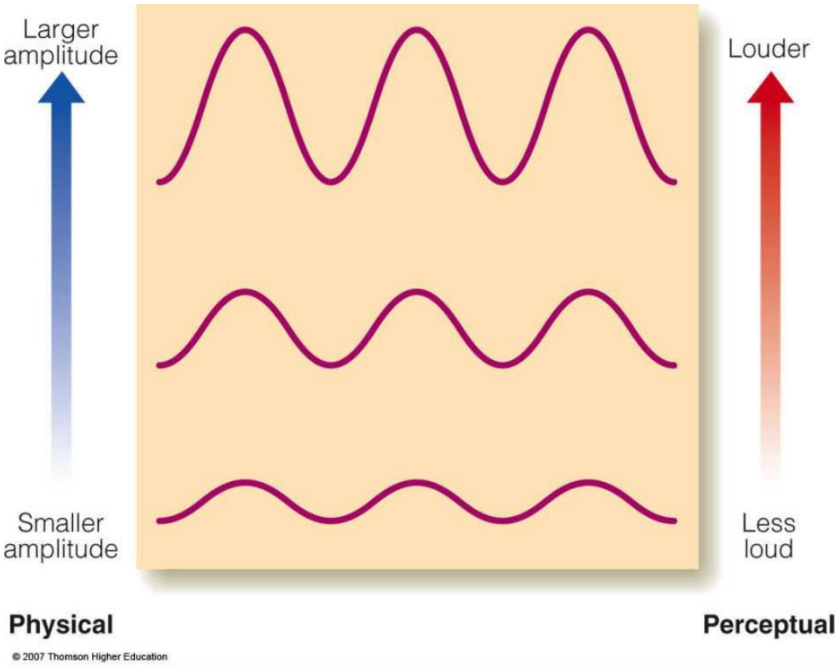

Figure 1.1 Amplitude and volume (taken from [3])

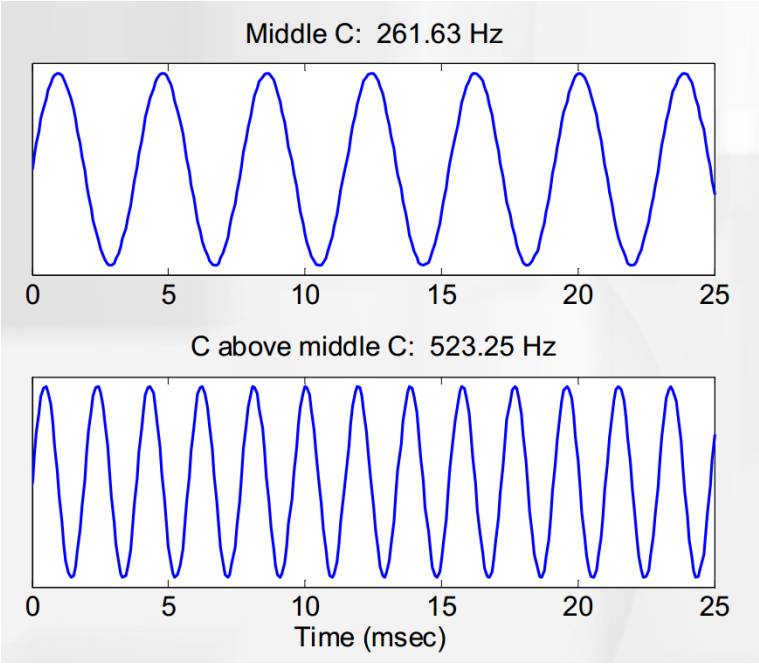

Figure 1.2 Frequency and pitch (taken from [3])

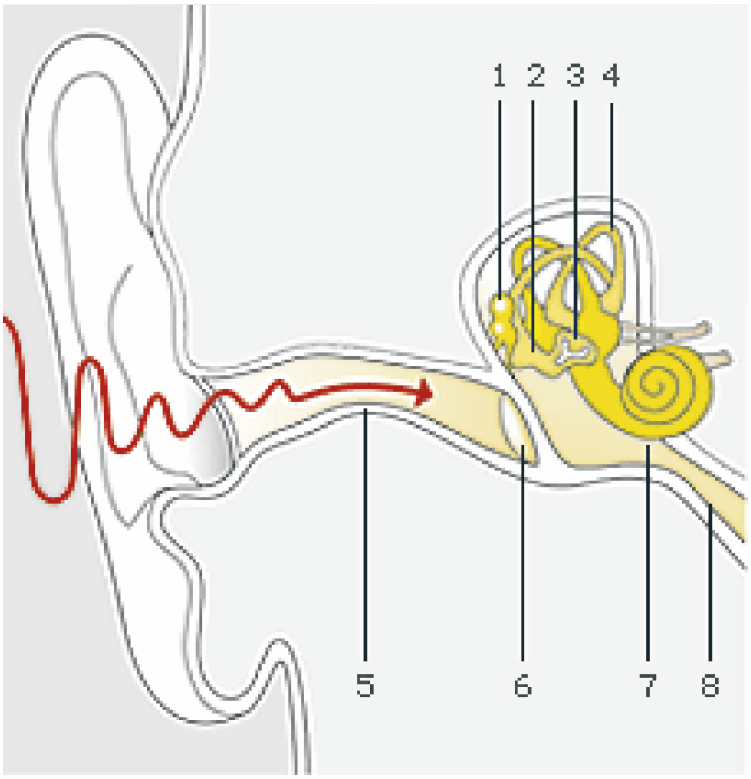

Figure 1.3 Human hearing system (taken from [3])

MATLAB, according to MathWorks (Makers of MALTAB and Simulink), is a multi-paradigm numerical computing environment, which helps engineers and scientists to analyse and design systems. Its applications cover several aspects, machine learning, signal processing, computer vision, communications, computational finance, control design and robotics [4]. As built-in Signal Processing Toolbox provides functions and apps to generate, measure, transform, filter and visualize signals [5], MATLAB was chosen to accomplish voice recognition for this project. Also, MATLAB Support Package for Arduino Hardware enables MATLAB to communicate with Arduino board and immediately feedback developers with results.

1.2 Voice Activity Detection (VAD)

Endpoint detection is very important in speech recognition. Its purpose is to detect useful speech signals from the speech signal and to remove the mute part of the speech signal. Endpoint detection will improve the system's speech recognition rate and reduce the amount of calculations of speech recognition [6].

At present, there are many kinds of endpoint detection algorithms, such as voice endpoint detection with information entropy, endpoint detection of frequency variance, endpoint detection of fractal technology, edge detection of sliding window and endpoint detection algorithm of double threshold comparison method [7]. The endpoint detection method used in this project is two-threshold comparison method. It finds the starting and ending points of the voice signal based on the short-term energy and zero-crossing rate of the speech signal [6].

The short-term energy of the speech signal is calculated as follows [7]:

$$E_{subframe} = \sum_{index=1}^{4} x^2((subframe-1)\times 4 + index) \quad \ldots (1)$$

Sub-frame takes values from 1 to the total number of sub-frames in the sample. Index denotes each sample in a given sub-frame.

The short-term zero-crossing rate of the speech signal is calculated as follows [7]:

$$Z_n = \sum_{m=-\infty}^{\infty} \left|\mathrm{sgn}\left[x(n)\right] - \mathrm{sgn}\left[x(n-1)\right]\right| \cdot w(n-m) \quad \ldots (2)$$

Where sgn[ ] is the sign function:

$$\mathrm{sgn}(n) = \begin{cases} 1, & x(n) \geq 0 \\ 0, & x(n) < 0 \end{cases} \quad \ldots (3)$$

And w(n) is the window function:

$$w(n) = \begin{cases} \dfrac{1}{2N}, & 0 \leq n \leq N-1 \\ 0, & \mathrm{Other} \end{cases} \quad \ldots (4)$$

N is the length of the window function.

At the beginning of the test, since the speech signal energy is relatively large, a higher threshold is set to confirm that the speech has started. And then set a lower threshold to confirm the true starting point and endpoint of the voice. Besides, set a lower zero-crossing rate threshold to determine the voice interval [6][7]. The result of detection example is shown as below:

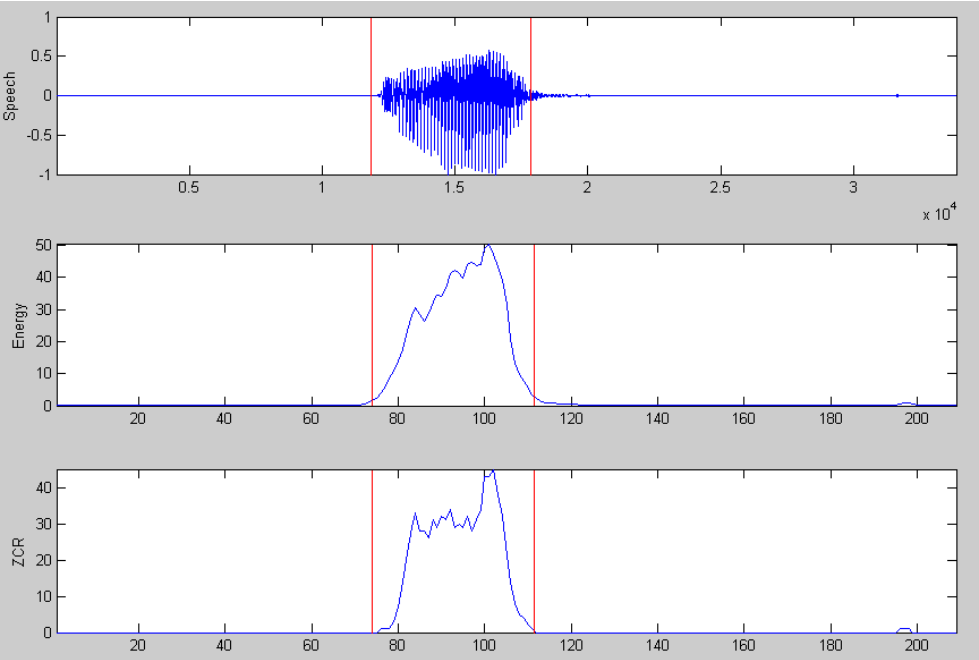

Figure 1.4 Result of the endpoint detection of the number '3'

1.3 Mel-Frequency Cepstrum Coefficient (MFCC)

In speaker recognition and speech recognition, the most commonly used speech feature parameter is the Mel-scale Cepstrum Coefficient (MFCC), which is consistent with the auditory characteristics of the human ear. The human auditory system is a non-linear system because it has different feelings for different frequency signals [8]. For a speech signal with a frequency less than 1 kHz, the human perception is linearly related to the frequency. And for more than 1kHz frequency of voice signals, perception and frequency are logarithmic relationship. The relationship between normal frequency f and Mel frequency are shown as below [9]:

$$f_{mel} = 2595 \log_{10}\left(1 + \frac{k}{700}\right) \quad \ldots (5)$$

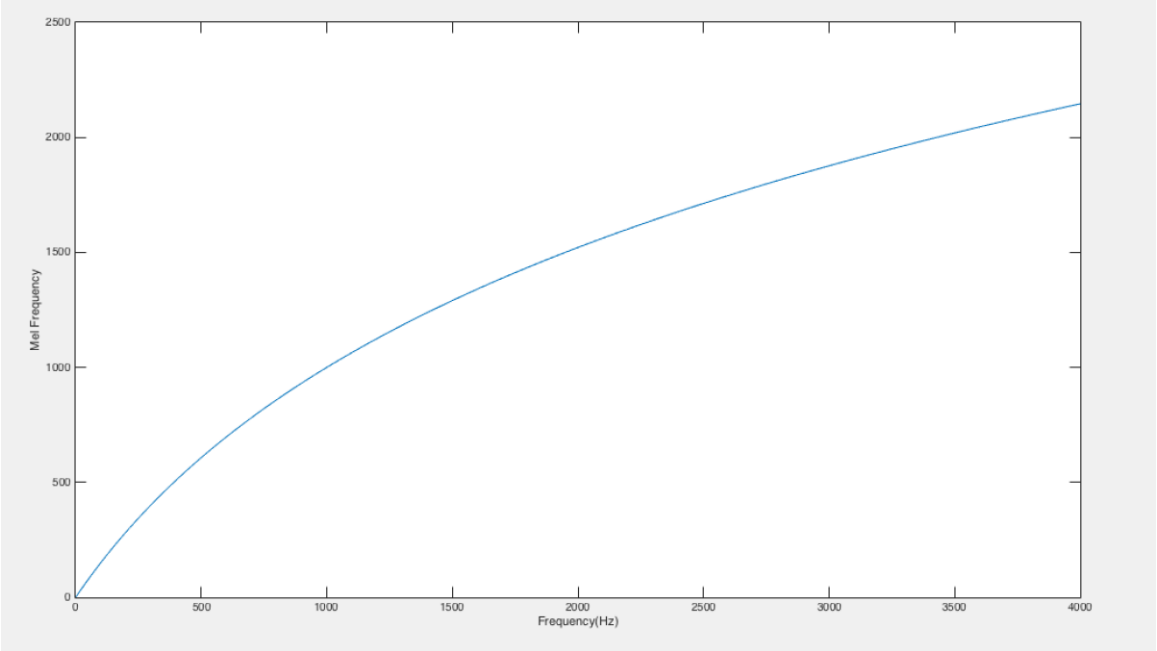

Figure 1.5 Corresponding relationship of frequency and Mel-frequency

People produce sound through the vocal organs, and he shape of the vocal organs determine the kind of the sounds. The vocal organ includes the tongue, teeth, vocal folds etc. If the shapes can be accurately known, then the produced phonemes can be accurately described. The shape of the vocal organ is displayed in the envelope of the speech short-time power spectrum, and MFCC is a kind of characteristic that can accurately describes this envelope [8].

The general steps for finding MFCC are as follows [8][10][11]:

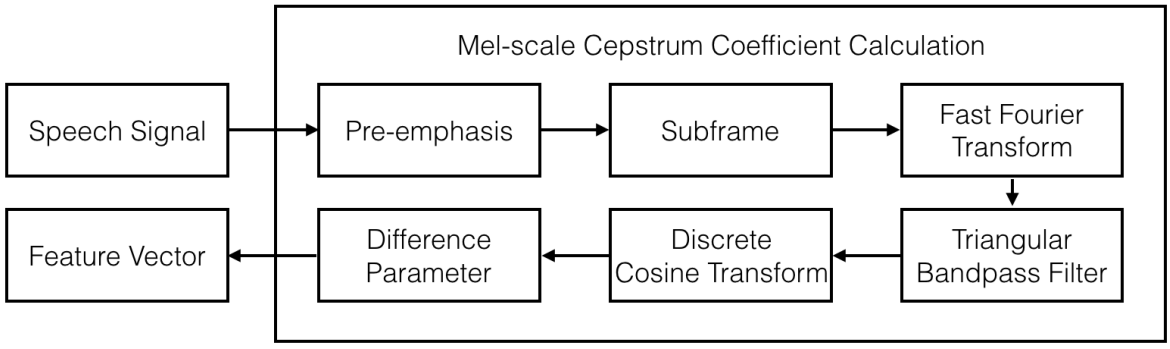

Figure 1.6 MFCC calculation process

a. Pre-emphasis: Let voice signal X(n) pass through a high-pass filter:

a) H(z) = 1 - a*z^(-1) … (6)

The coefficient a is between 0.9 to 1. In this project, a is equal to 0.9375. The purpose of this process is to eliminate the effects of the lips and vocal processes during the vocalization, and to compensate for the high frequency parts of the speech signal that are suppressed by the pronunciation system [9].

b. Sub-frame: N sampling points are aggregated into an observation unit, which called a frame. N = 256 in this project. Then multiply each frame by Hamming window to increase the continuity of the left and right ends of each frame [8].

c. Fast Fourier Transform: FFT for each frame, because the signal changing in the frequency domain is more clearly about the characteristics of the signal. In addition, during this process, the signals within each frame are assumed to be periodic. If this period does not exist, in order to conform to the discontinuities of the left and right ends, the fast Fourier transform will produce some energy distributions that do not exist in the original signal, and it will result in erroneous analysis [9][10].

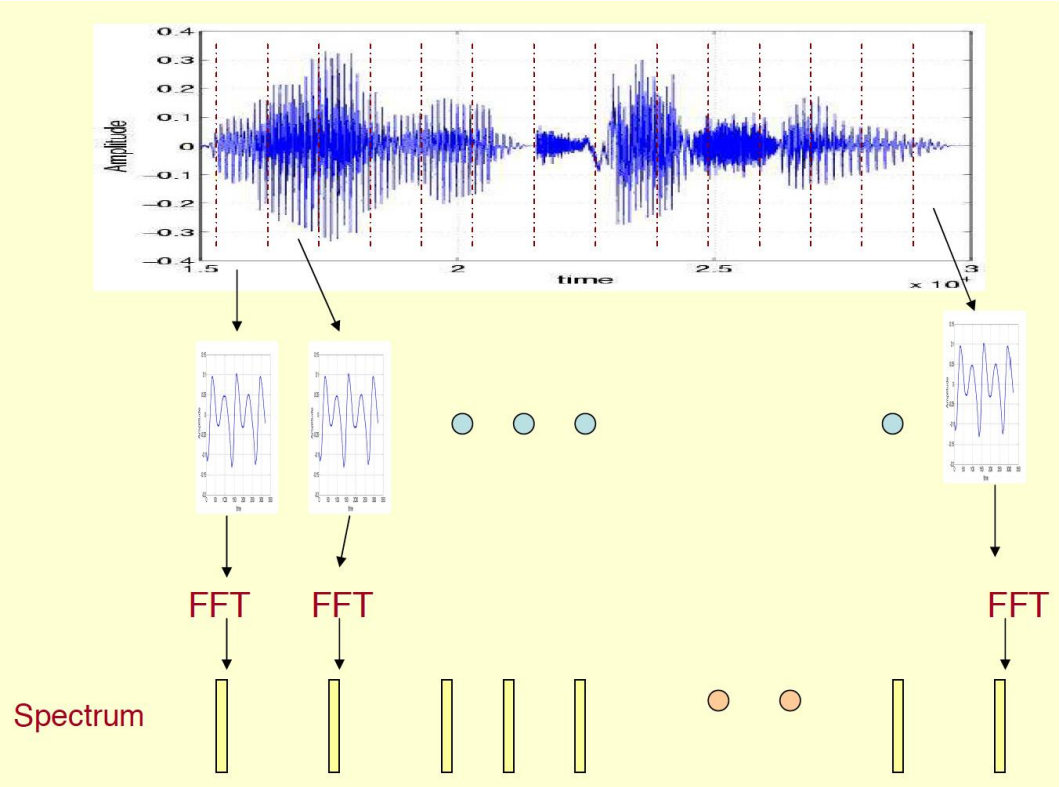

Figure 1.7 Fast Fourier Transform for each frame (taken from [10])

d. Add a Mel frequency filter: The Mel spectrum is obtained by passing the upper spectrum through the Mel filter bank [10].

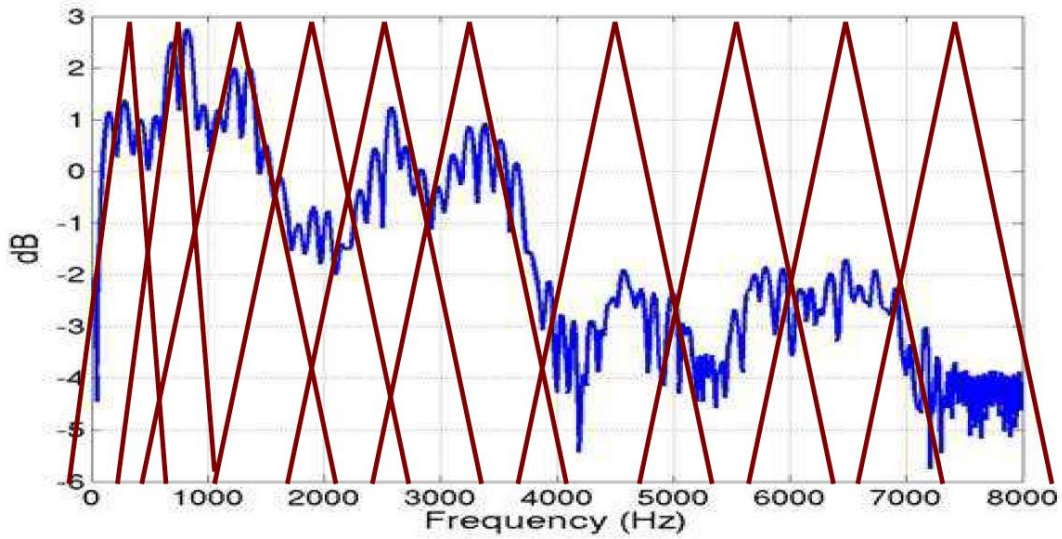

Figure 1.8 Mel-Frequency Filter (taken from [10])

e. Discrete cosine transform: The spectral cepstrum coefficients are obtained by discrete cosine transform. The formula is shown as below [9]:

$$c_m = \sum_{n=1}^{N} X_n \cos\left(\frac{\pi}{N} m(n-0.5)\right), m = 1, 2, \ldots, M, \quad \ldots (7)$$

Where M is the number of the MFCCs, and N is the number of Mel-scaled filters.

f. Extraction of Dynamic Difference Parameters: MFCC reflects the static characteristics of the voice signal. The dynamic characteristics of the speech signal can be described by these static feature differences [9]. Comparing with another feature value called Linear Predictive Coding(LCP), MFCC has two advantages on voice recognition [8]:

I. Most of the voice information is concentrated in the low frequency part, and the ambient noise is easy to interfere with the high frequency part of the speech information. MFCC emphasizes the low frequency of voice information, and highlights the information conducive to identification. It also inhibits the noise interference. The LPCC

coefficient does not have these features because it is based on linear frequency scales.

II. The signal is subjected to FFT during the extraction of the MFCC parameter.

Therefore, it can get all the information on the frequency domain of the voice signal. It will help endpoint detection, speech segmentation and other algorithms.

1.4 Dynamic Time Warping (DTW)

At present, the algorithm of speech recognition mainly has Dynamic Time Warping (DTW), Vector Quantization (VQ), Hidden Markov Model (HMM) and Artificial Neural Networks (ANN). Since the objective of the project is the recognition of simple isolated word, MFCC and DTW is enough and efficient [12].

Dynamic time warping is an algorithm used to measure the similarity between two sequences that may change over time or velocity. In speech recognition, even if a person said the same word twice, the two voice signals are different. The duration, peak, and peak time of the two signals may not be the same. Therefore, if compare the Euclidean Metric of two signal directly, the success rate of recognition will be very low. DTW will expand or compress the language, and then find the best possible match frame by frame [12]. The following figure is an example of a DTW algorithm:

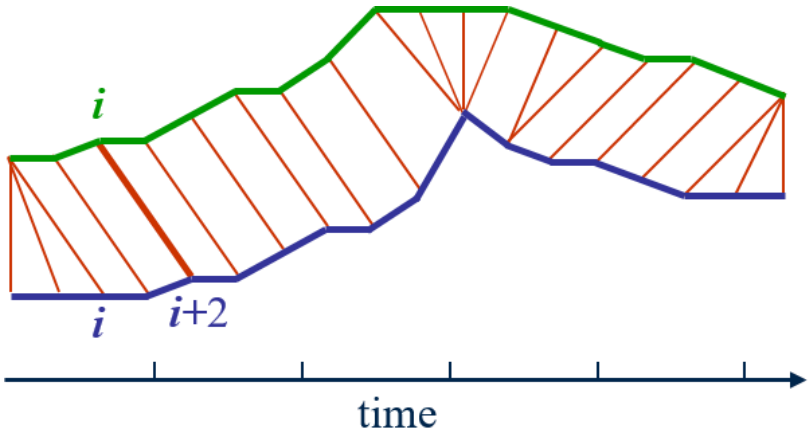

Figure 1.9 Example of DTW algorithm (taken from [12])

Meanwhile, the upper signal has a point i corresponding to the lower one. However, this point should actually correspond to (i+2). DTW can calculate the distance of the actual corresponding point, so that to get the best matching results [12].

In the voice training template phase and the speech recognition phase, the beginning and end of the voice should be determined by using the detection endpoint algorithm at the beginning. Each of the voice entry parameters stored in the template library is called a reference template. The following figure and paragraphs will introduce the basic procedures of DTW algorithm.

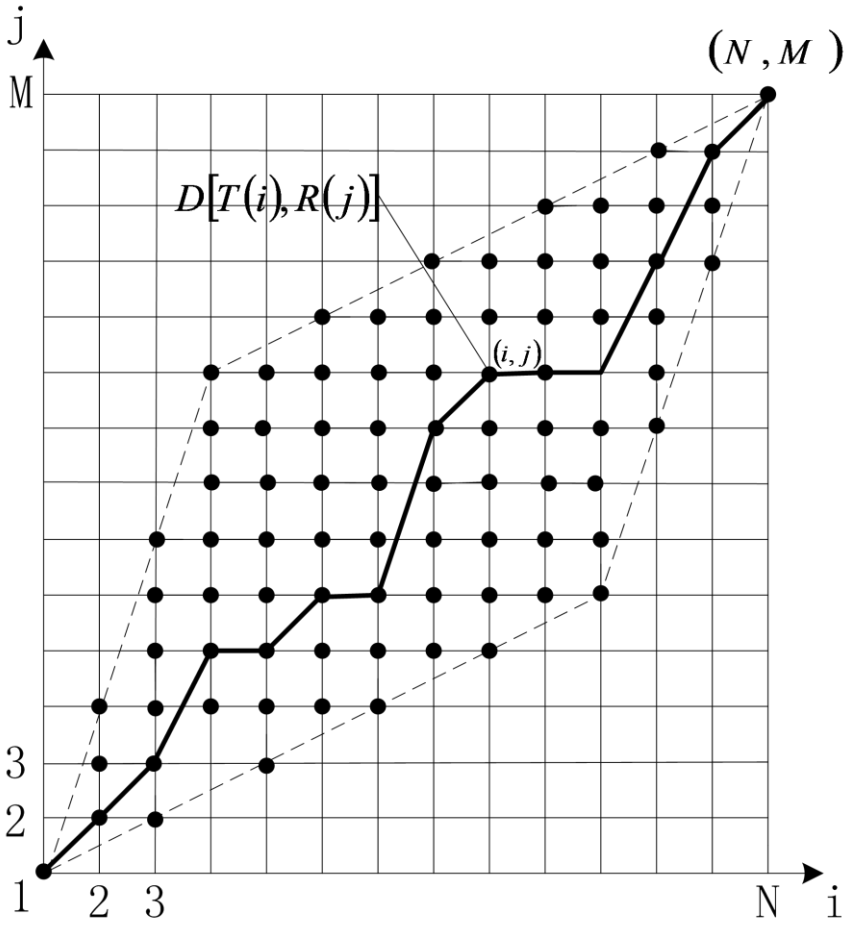

Figure 1.10 Dynamic programming algorithm schematic

One reference template is expressed as [12]:

R={R(1)，R(2)，…，R(j)，…，R(M)} … (8)

Where j is the timing label of the training speech frame, and M is the total number of speech frames contained in the template. Therefore, R(j) is the speech feature vector of j-th frame.

The input entry voice parameter that need to identify is called test template, which can be expressed as:

$$T=\{T(1),\ T(2),\ \cdots,\ T(i),\ \cdots,\ T(N)\} \quad \ldots (9)$$

Where i is the timing label of the test speech frame, and N is the total number of speech frames contained in the template. Therefore, T(i) is the speech feature vector of i-th frame. The reference template and the test template usually use the same type of feature vector (MFCC in this project), the same frame length, the same window function and the same frame shift.

In order to compare the similarity between the test template and the reference template, the distortion distance d [T, R] between them can be calculated. The smaller the distance means the higher the similarity. The distance between the corresponding frames in T and R is calculated to measure the distortion distance. Let i and j denote arbitrary selected frame numbers in T and R, and d [T (i), R (j)] represent the distortion distance between the two frame. There are many kinds of distance measurement, and Euclidean distances are usually used in the DTW algorithm.

The frame number i=1~N of the test template is used as the abscissa of the two-bit Cartesian coordinates. And then the reference template frame number j=1~M as the vertical coordinates. A two-dimensional network can be formed by these frame numbers. The intersection point (i, j) of the network represents the distance between the i-th frame number of the test template and the j-th frame of the reference template.

The dynamic programming algorithm is to find a minimum of the sum of the distances in the path through several intersections of the network. And then find the sum of the distance between the intersection of each path. The order of the

intersection of the paths does not change. Therefore, the selected path must start from the lower left corner and end to the upper right corner. The figure shows an example of dynamic programming algorithm [12].

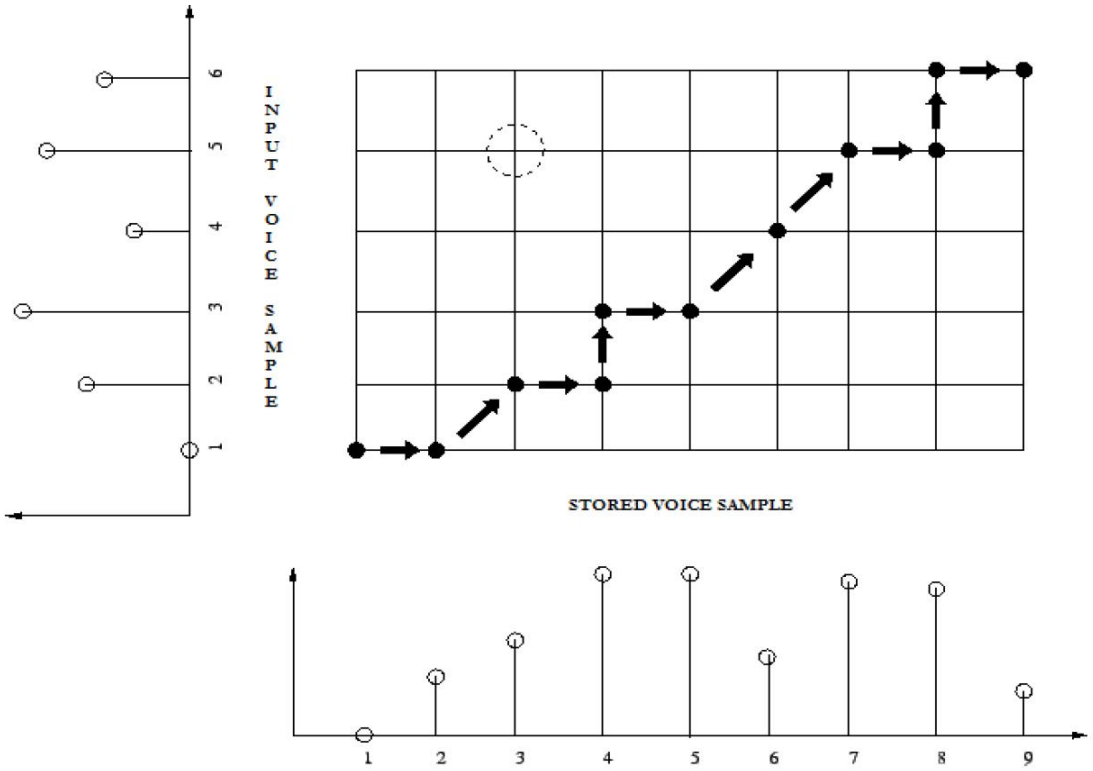

Figure 1.11 Example of dynamic programming algorithm (taken from [12])

1.5 Hardware Components

The hardware employed in this project includes Arduino (Microcontroller), full-adder ship (7483), breadboard, microphone, and 7-Segement display. Full-adder ship is used to build calculator circuit that accomplish the addition, and the function of 7-Segement is to build display circuit. The primary idea before the project is that the microphone receives voice signal, then MATLAB analyses it and send it to full-adder, and finally the result is displayed on 7-Segement. Arduino as an electronic prototype platform, is the most important components in hardware part, which is responsible to accept voice signal from microphone and to communicate with MATLAB. All the operations about hardware are based on breadboard.

a. Arduino (Microcontroller)

Arduino is a popular electronic interactive platform at present, developed by microcontroller system, with the advantages such as simple to use, versatile and inexpensive, it is widely used in the design of electronic system and development of interactive product. The Arduino type is shown in the Figure 1.12.

Figure 1.12 Arduino UNO

A 9 VDC input, to provide external power output for the Arduino board, to enable the Arduino board to drive servo power external equipment. A USB port, connected to the computer through the mouth, 14 digital input/output, six analogue input/output, 1 5 VDC output and 1 3.3 VDC output. It can quickly be combined with the software such as Arduino with Adobe Flash, Processing, Max/MSP, Pure Data [13]. Arduino can communicate with computer with its own IDE (Integrated Development Environment), whose syntax is similar to C and it is reasonably easy for developers to grasp.

b. Full-adder ship (7483)

These full adders perform the addition of two 4-bit binary numbers. The sum (Σ) outputs are provided for each bit and the resultant carry (C4) is obtained from the fourth bit. These adders feature full internal look ahead across all four bits. This provides the system designer with partial lookahead performance at the economy and reduced package count of a ripple-carry implementation [14]. The adder logic, including the carry, is implemented in its true form meaning that the end-around carry can be accomplished without the need for logic or level inversion. In this

project, we used it to build the calculator circuit. The connection diagram is shown in the Figure 1.13

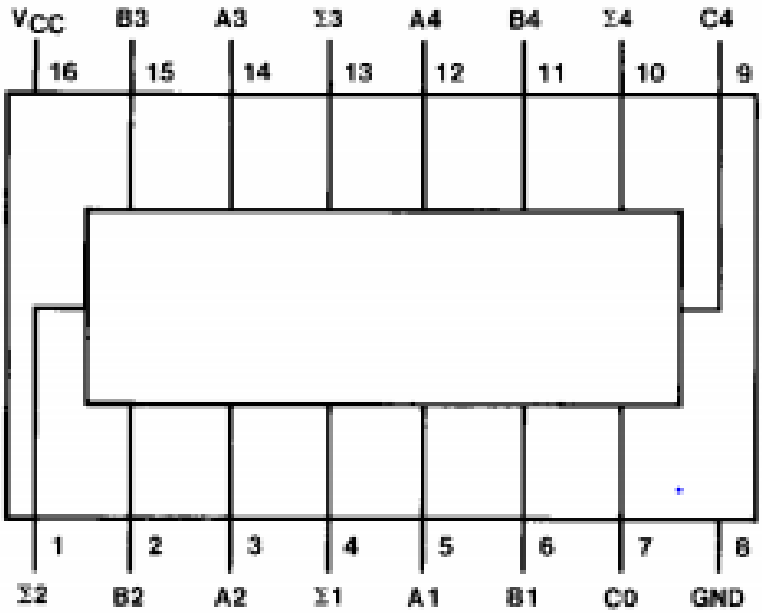

Figure 1.13 Full-adder (7483)

c. Breadboard

All the operations about hardware are based on breadboard, the type we used is shown below.

Figure 1.14 Breadboard

d. Microphone

Electret microphone consists of two parts, acoustoelectric conversion and impedance transformation, we used it to build the microphone circuit, the aim of it is to receive the voice input.

Figure 1.15 Microphone chip

e. 7-Segement display

The 7-segment display, consists of seven LEDs (hence its name) arranged in a rectangular fashion. Each of the seven LEDs is called a segment because when illuminated the segment forms part of a numerical digit (both Decimal and Hex) to be displayed. An additional 8th LED is sometimes used within the same package thus allowing the indication of a decimal point, (DP) when two or more 7-segment displays are connected together to display numbers greater than ten [14].

Depending upon the decimal digit to be displayed, the particular set of LEDs is forward biased. For instance, to display the numerical digit 0, we will need to light up six of the LED segments corresponding to a, b, c, d, e and f. Then the various digits from 0 through 9 can be displayed using a 7-segment display [15].

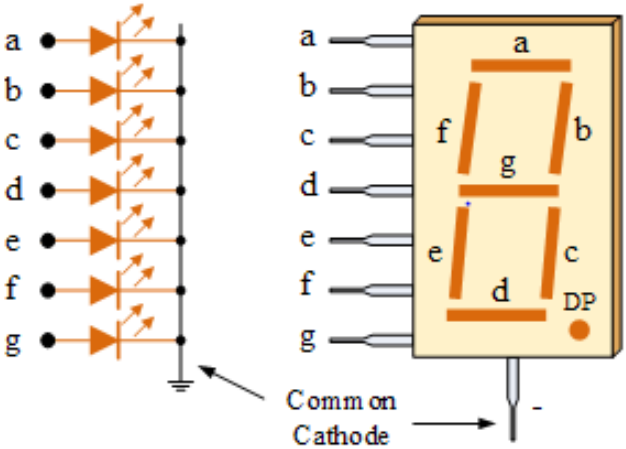

Figure 1.16 Common-anode 7-segment display

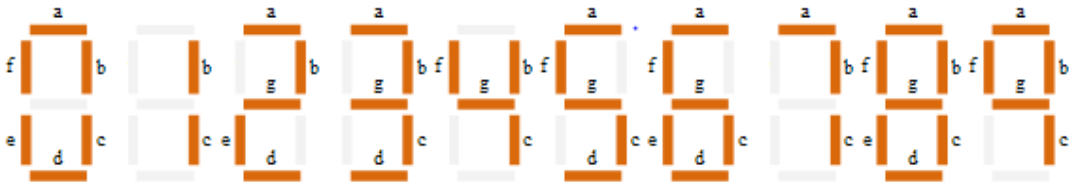

Figure 1.17 7-segment display for all numbers

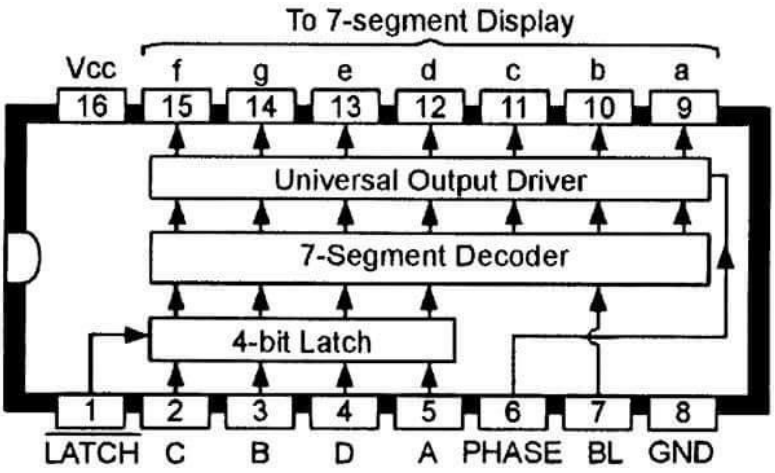

Figure 1.18 BCD-7-segment Decoder

1.6 Objectives

The main objective of this project is to design and construct a digital calculator with voice input and 7-segement output, and the final deliverable should be a speech recognition calculator based on MATLAB. More accurate objectives are listed below.

- Designing and constructing a digital circuit with a full-adder.
- Constructing a display circuit with 7-segment LED display.
- Researching and understanding the basic attributes of a voice signal.
- Studying how to process a voice signal, especially to compare it with others.
- Designing and implementing an MATLAB software with recognition range from number one to number nine.
- Investigating the communication between MATLAB and Arduino UNO.
- Studying how to locate and correct errors.

Since speech technology is unfamiliar to team members, it is suggested to study relevant research in advance to obtain a basic background. The final deliverable should be a combination of Arduino UNO and some other essential ICs, and it should accept user's voice and feedback with addition results. By the end of the project, the members are expected to attain a profound knowledge about integrated circuits and signal processing.

Chapter 2

Materials and Methods

2.1 Apparatus List

74LS08 AND gate

74LS32 OR gate

2x 74LS04 inverter

744LS283 4-bit full adder

74LS47 BCD to 7 segment decoder

Arduino UNO

2x SC56-11EWA 7 segment displays

2x SK10 prototype board

ISO tech LPS2250 DC voltage supply

TENMA 72-1016 multimeter

Tektronix 1012C-EDU digital oscilloscope

MATLAB R2016b

2.2 Circuit Design

Shown by Figure 2.1 is the functioning circuit that implements the aims and objectives. The red and black wires connecting the Arduino to the breadboard are the 5V DC supply and ground respectively. This 5V DC supply was connected as such so that the two breadboards each had a DC supply rail. From these supply rails the various chips/components are powered (VCC). The blue wires seen connecting the Arduino to the breadboard represent the two 2 bit numbers outputted from the voice recognition software. These wires are connected to the 4-bit full adder chip (744LS283) as such so to allow for addition of the two numbers. The result from this addition is then passed to the BCD to 7 segment decoder (74LS47). This decoder converts the binary output of the adder into several signals corresponding to individual segments of the 7-segment display. The outputs from the decoder are inverted by 74LS04 chips. As shown by the schematic, two separate 7 segment displays are used. One for showing the result of the calculator and the other for displaying an error message when the result is

out of the range (1-9). To determine when a result is out of range a logical operation is performed by the AND (74LS08) and OR (74LS32) chips which, if true, will display an ‘E’ on the second display.

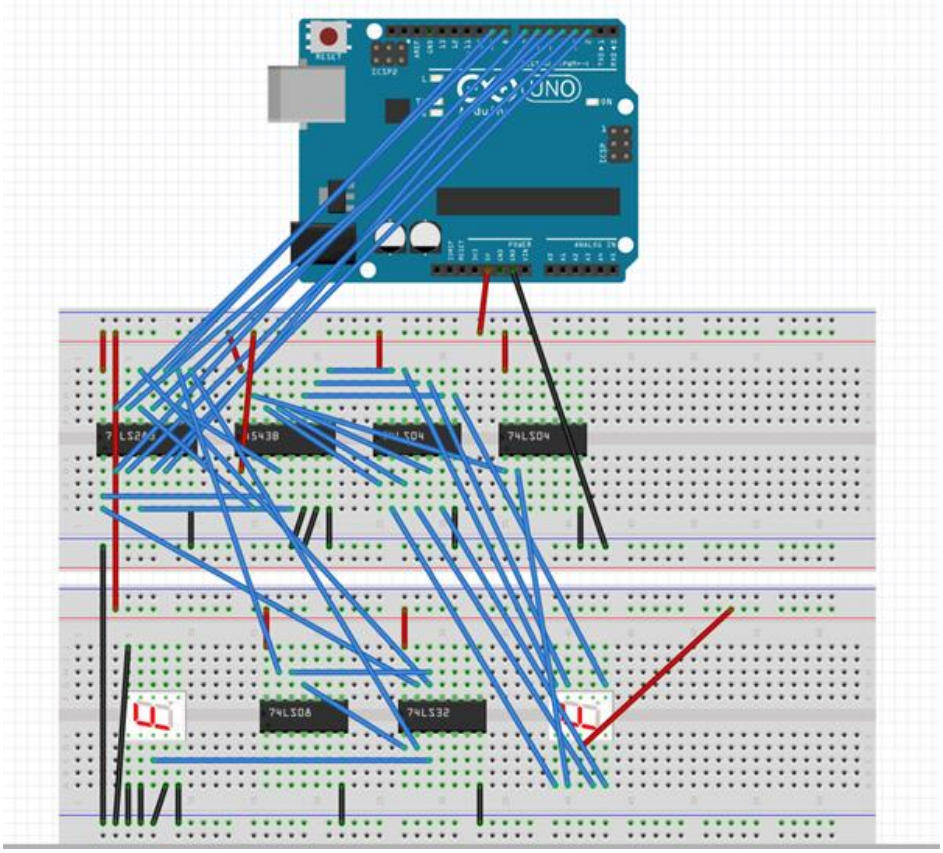

Figure 2.1 Final Circuit for calculator with voice input

2.3 MATLAB Coding

As the algorithm explained in the Introduction part, MATLAB code contains seven functions, which are GUI.m, recognition_1.m, recognition_2.m, vad.m, mfcc.m, dtw.m, disPlay.m. Since the calculator needs to accept two voice signal, there are two functions of recognition that process two voice input respectively. Although two recognition function share the same code that might slower computer processing, the programme achieves asynchrony which allows calculator to transfer the first digit to 7-segment display after first voice input and transfer the addition result after second voice input. Functions vad.m, mfcc.m, dtw.m correspond to three introduced algorithms, Voice Activity Detection (VAD), Mel-Frequency Cepstrum (MFCC) and Dynamic Time Warping (DTW).

GUI function is responsible for the building graphical user interface of the whole programme, which is also regarded as the ‘control centre’ of the calculator, as displayed in Figure 2.2.

The communication between MATLAB and Arduino is supported by the Arduino package developed by MathWorks. With commands `a = arduino()`, an Arduino object is created with several attributes (name, port, memory etc). After declaration, the MATLAB code can directly manipulate the pins on Arduino, and therefore the output digital pins can be configured in MATLAB, which is integrated in a separate function display.m.

There are some variables that maybe more than one function need to access to, such as recognition result (1-9) or Arduino object. It is necessary to declare these variables as global variables, whose syntax forces the global declaration for every single access. Moreover, the sequence of variables’ assignment is crucial, otherwise some overwritten errors might occur.

There are four bottoms on GUI, which allows user to control the programme through clicking the bottom. The bottom ‘Reset’ in the GUI enables user to reset the programme and meanwhile reset all the pins on Arduino UNO connected to the computer. The bottom ‘Number 1’ enables user to input the first number with his/her voice and the bottom ‘Number 2’ enables user to input the second number. After voice input, recognition part will operate to generate a result tabulate and display just under the bottom ‘Number 1’ and ‘Number 2’ (Figure 2.1). Meanwhile the addition part will display the recognized number and console part will display the recognition result to feedback user. The bottom ‘Display’ enables user to calculate the addition and transfer calculation result to Arduino UNO by the means of assigning the digit pins. Interaction process usually starts with ‘Reset’, which resets the program and Arduino UNO at the same time, and then user can input his/her voice through ‘Number 1’ and ‘Number 2’ bottoms. After recognition, user can choose to click ‘Display’ to display the result on both the GUI and the 7-segement. Therefore, the general process for input and output is displayed as Figure 2.3. Complete MATLAB code is appended in appendix A.2.

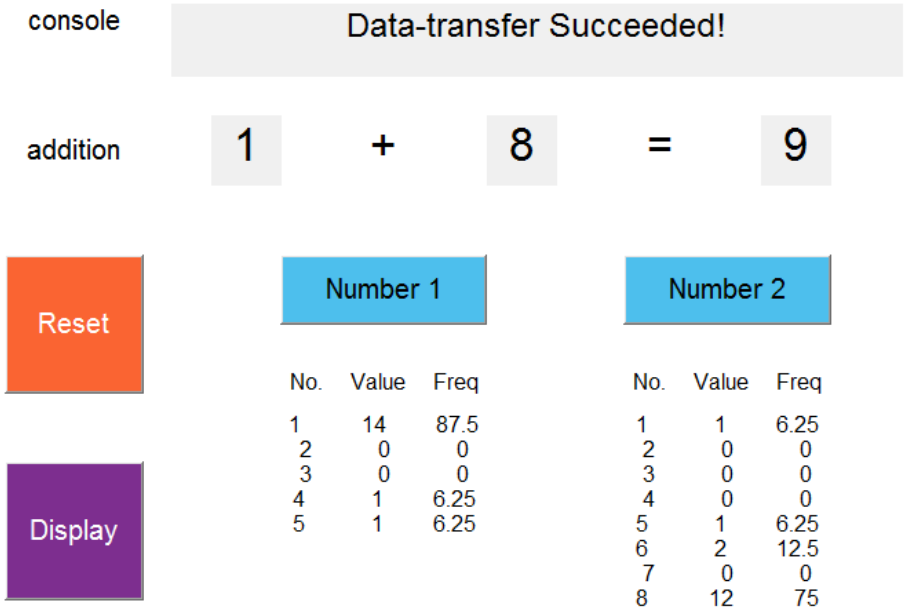

Figure 2.2 MATLAB GUI of programme

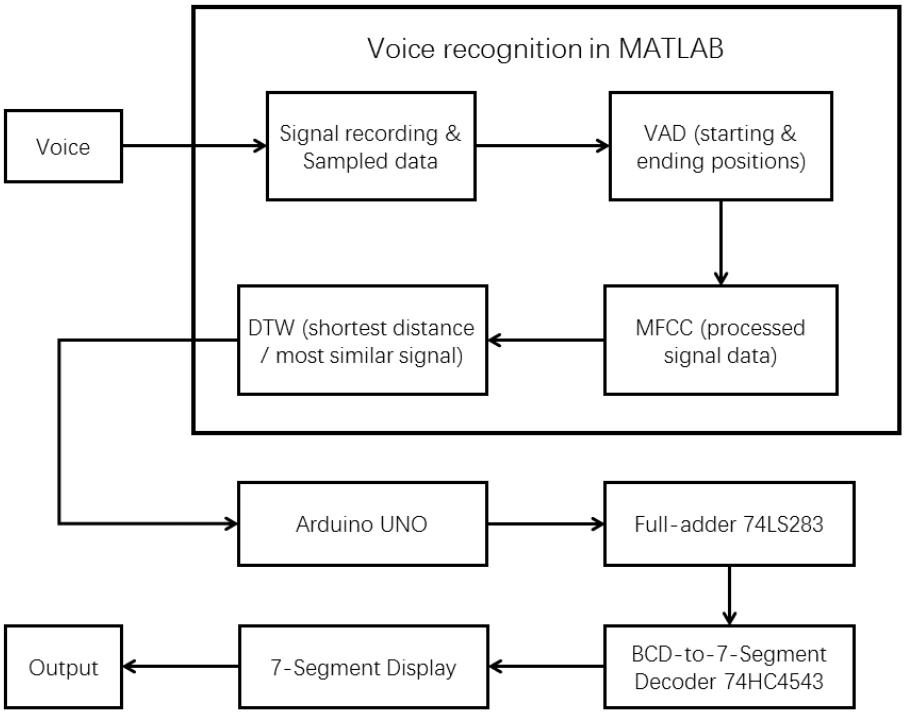

Figure 2.3 General process for input and output

Chapter 3

Results

The project is finished with the deliverable of a computer connected to some integrated circuits, and it is displayed as Figure A.2. The final deliverable contains both MATLAB code and electronic circuit, in which speech recognition matching rate and circuit output respectively matter. These two project results will be addressed separately.

3.1 Speech Recognition Matching Rate

Finally, in this project, the group recorded 8 sets of databases. Each set of databases consists of a set of numbers 1-9 eigenvalues, which are issued by the same person. In particular, the database number 1-4 is recorded by the same person, and 5-6 is recorded by another four others. The relationship between the database and the people is as follows：

Table 1 Relationship between person and database

Person	A	B	C	D	E
Database	1-4	5	6	7	8

During the test, each number was tested 10 times in each single test. For example, if the test results were successfully matched 7 times, then the match rate is 70%. The results of the speech recognition tests based on different databases will be shown as below:

Test 1: Person A with Database 1-4

Firstly, the group tested the matching rate of the person with his database. The database 1 to 4 was recorded from person A. This test is aim to find the basic matching rate of the same person with his own voice.

Table 2 Matching rate of a person with his own database

Number	Matching Rate (%)				Average (%)
	Database1	Database2	Database3	Database4	
1	100	100	90	100	97.5
2	100	80	90	90	90
3	90	100	90	80	90
4	100	100	100	90	97.5
5	90	100	100	100	97.5
6	100	100	100	100	100
7	90	100	90	100	95
8	90	100	100	100	97.5
9	100	90	100	100	97.5
Average	95.56	96.67	95.56	95.56	95.83

It can be seen from the statistics that the matching rates were extremely high, which is near to 100%. This result may be due to the characteristics of MFCC. Even when the length and volume of the speaker's voice changes, there is just a light change in the value of the MFCC. At the same time, the table also shows that even the sample and the database are from the same person, the matching rate will not always be 100%.

Test 2: Person A with Database 5-8

In test 2, the testing database was changed to 5-8, which was recorded from 4 different people. The objective of this test is to find the how the different people's database will influence the recognition result.

Table 3 Matching rate of a person with other's database

Sample	Matching Rate (%)				Average (%)
	Database5	Database6	Database7	Database8	
1	90	90	80	90	87.5
2	70	80	60	70	70
3	90	80	80	100	87.5
4	80	70	70	80	75
5	100	90	60	80	82.5
6	90	80	80	70	80
7	90	90	70	90	85
8	100	70	70	80	80
9	80	90	80	80	82.5
Average	87.78	82.22	72.22	82.22	81.11

The table shows the matching rate when compare the testing person's voice with another people's database. It is apparent from the statistics that the matching rate decreased from test 1. The average matching rate was about 81.11% in general. And the data shows that it the feature value of testing person's voice is most similar to person B's, but have the biggest difference with C's voice.

Test 3: Person BCDE with Combined Database 1-4

In this test, the testing person was change to B, C, D and E, and used the person A's voice as the sample database. The database was combined with 1-4 database. The recognition result will consider all database and select the most one as the result. For example, when person B said number "2", if 1, 2 and 4 databases show the result is "2", but database 3 consider it is "4", then the recognition result should

be 2, which is correct. This test result will show the relationship between different people's voice and the same database. Table recorded the testing result in test 3:

Table 4 Matching rate of a person with same person's combined database

Sample	Matching Rate (%)				Average (%)
	Person B	Person C	Person D	Person E	
1	90	80	80	90	85
2	80	90	60	70	75
3	80	70	70	80	75
4	90	90	70	80	82.5
5	80	80	80	90	82.5
6	70	80	70	80	75
7	90	70	70	90	80
8	70	80	60	80	72.5
9	90	80	80	70	80
Average	82.22	80.00	71.11	81.11	78.61

The statistics indicated that the matching rate has obvious difference for different people. The overall matching rate is 78.61%, which is still near to 80%.

Comparing with test 2, the results has similarity. For example, the average rate of B in is near to database 5's in test 2. And the same conclusion can be found by comparing other pairs. It is because comparing with A's sample and B's database should be the similar as comparing with B's sample and A's database.

Test 4: Person A and B with Combined Database 1-8

In the final test, all database will be considered, which included 4 of the same one's and 4 of other different people's. Only A and B took part in this test because

it is same for people B, C, D and E, who only has 1 database. Comparing with the result of this test and other test should obtain the relationship between single database and combined database.

Table 5 Matching rate of a person with combined database

Sample	Matching Rate (%)		Average (%)
	Person A	Person B	
1	100	90	95
2	90	70	80
3	100	80	90
4	100	90	95
5	90	80	85
6	100	90	95
7	100	80	90
8	100	90	95
9	100	80	90
Average	97.78	83.33	90.56

After finishing 4 test, there are several inferences can be obtained by comparing with all results:

I. A person will have high matching rate by using the database record from his voice. And if the tester uses the database which is not come from himself, the matching rate will be significantly reduced.

II. For different numbers, the matching rate is different. During the test, the number “2” is most likely to be identified incorrectly, but the overall match rate is still around 80%.

III. The combined database can increase the matching rate when the source of the sample and the database are the same person. However, for different people and database, the effect is not obvious.

3.2 Hardware Output

As a digital calculator, there are two 7-segment displays in the hardware, which represent the addition result and calculator status respectively.

First Digit (addition result display)

The hardware part is designed to read the data from Arduino UNO board, and then to display the addition result on the two 7-segment displays. The addition function is accomplished by 4-bit binary full adder with carry (model 74LS283). The outputs of full adder are decoded by connecting to the corresponding pins. There are 7 outputs from BCD to 7-segment decoder (model 4543B). Each of the outputs need to be inverted due to the common-anode 7-segment displays. Connected to pins, the first digit 7-segment display can successfully display the range of 0 to 9. Nevertheless, the outputs of decoder will all be forced to logic 0.

Second Digit (status display)

From 10 to 15, the first 7-segment display swill be disabled. To solve this problem, a combinational logic circuit is constructed to display “E” (represent error) in the second 7 segment display. The logic output is an enable button for the second display. The combinational logic circuit is designed as Carry ∪ [D ∩ (C ∪ B)]. When the binary output is greater than 1010 (carry bit will be discussed section), the enable button is pushed, and “E” will be displayed in the second display.

Carry Bit

When the addition result is greater than binary 1111, the carry bit will output logic 1. At the same time, the decoder starts decoding. Therefore, the first display will continue counting from 0 to 3. Whilst, the enable button of the second display is still closed, “E” is displayed in the second display.

Chapter 4

Discussion and Conclusions

4.1 Microphone Amplifier

During the initial design phase, to take voice input from the user, a microphone circuit was to be constructed in the lab, which is display as Figure A.1 in appendix. This circuit was required to amplify and then to offset the signal from the microphone, which is suitable for a level that Arduino could interpret. Unfortunately, such a design proved too difficult to implement. Some progress was made using the 741 operational-amplifier, however useful functionality was never achieved. Upon realising this, changes to the design were made so that the microphone built into a laptop could be substituted for receiving vocal input from a user. This alteration proved successful and was carried through to the final design.

4.2 Communication Between MATLAB and Arduino

Since the project needs to recognize human voice, it is necessary to accomplish through MATLAB, which can communicate with Arduino with support hardware package. According to MATLAB Arduino syntax, some functions are achievable, such as reading voltage from specific pin and writing logic status to digit pin. These two functions enable the signal, after analysed by MATLAB, to be transferred to Arduino and output correctly by writing logic statue to digital pin. However, the initial idea for the final deliverable is only Arduino UNO with MATLAB code pre-transferred into it. The reason why it is not accomplished it that Arduino cannot operate alone without connection to computer, which not only exchanges data with Arduino but also supplies power. Moreover, there is no memory for storing data on Arduino so the MATLAB code cannot even exist on Arduino, which can only accept the commands from computer to configure its digital pins. Finally, without any switch on Arduino, it is impossible to control the whole interaction through just an Arduino board and therefore, one computer must be used, which takes part in supplying power, operating MATLAB code, configuring Arduino pins and controlling the whole interaction.

4.3 Display Range & Logic Gate

Since there are some defects and shortage of components, the display part is re-designed to accomplish the project to a maximum extent. The display range is divided to three parts, which represents different situations of addition result, as Table 6 shows.

Table 6 Logic states of pins and logic gates in different ranges

Range	D	C	B	A	Carry	D ∩ (C ∪ B)
0-9	0	0	0	0	0	0
	0	0	0	1		
	0	0	1	0		
	0	0	1	1		
	0	1	0	0		
	0	1	0	1		
	0	1	1	0		
	0	1	1	1		
	1	0	0	0		
	1	0	0	1		
10-15	1	0	1	0		1
	1	0	1	1		
	1	1	0	0		
	1	1	0	1		
	1	1	1	0		
	1	1	1	1		
16-18	0	0	0	0	1	0
	0	0	0	1		
	0	0	1	0		

For the range from 0 to 9, the first digit displays addition result, and the second digit will be disabled by combinational logic circuits.

For the range from 10 to 15, the first digit will be disabled because the decoder does not work at this stage. And in accordance with logic circuit, the second digit is enabled, displaying “E”

For the range from 16 to 18, outputs DCBA is cleared, and start from 0000 with a logic 1 carry output. The carry output will enable the second digit display, presenting “E”.

Combinational Circuit design is showed in Fig 4.1.

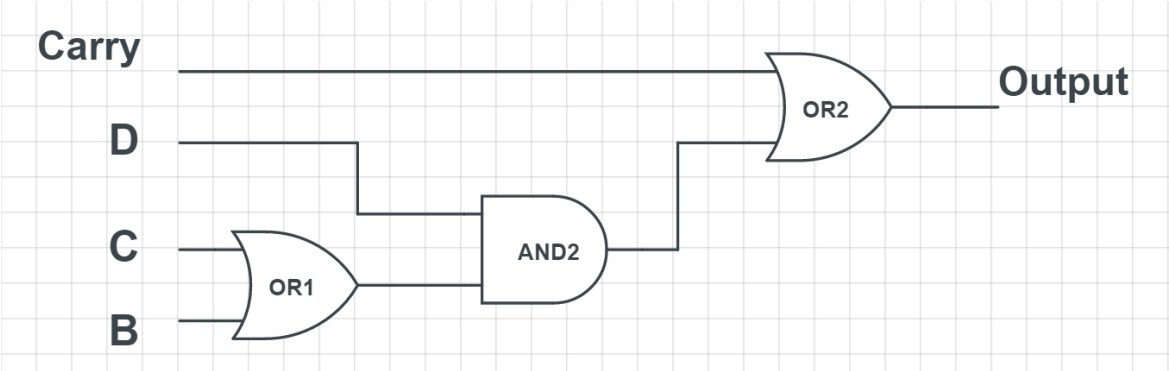

Figure 4.1 Logic circuit design

Lack of decoder limits the second digit display, the carry out bit cannot decode. The reason of using logic circuit is that, if only one decoder is applied, only one of the display can work properly. Considered the range of the result values, a wise solution came out. Due to that the result range is from 2 to 18, the second digit will only switch between 0 and 1.

In the case of 0, the addition values fall within 0 to 9. Convert it to binary, the values are not greater than 1001 (included). Thus, only the first digit display work normally, and the second digit display is disabled at this stage due to the carry out bit is logic 0.

In the case of 1, and the addition values fall within 10 to 15. Convert it to binary, the logic gates will active high, and then drive the second digit display, showing “E”. The first digit display will show void.

In the case of 1, and the addition values fall within 16 to 18. The logic gates will active high, because carry out bit is enabled. The second digit display will show “E”. Output pins of full adder are cleared (all logic 0). Therefore, the first digit display will continue counting from 0 to 2.

4.4 Applications

Applications of a voice input calculator are expected to be numerous. Possibilities range from implementation in lab environments where scientists or technicians may be using their hands to conduct other work, to the situation in home environments where simple functionality is ever important.

The key concept of this project is basically the voice recognition software. Voice recognition arguably has infinite applications, an example would be iPhone and voice recognition software, which is included on all latest operating systems developed by Apple. The ability to talk to nearby devices is a functionality that seems to be included onto hardware ubiquitously. Another intriguing area of development would be for those who have disabilities, particularly lack of vision. A system that requires vocal input is much more accessible to those who are visually impaired therefore further insight into this application could be highly beneficial to society. Consequently, for this particular project, 7-segment displays are used for outputting results which is not suited to those with lack of vision. However, the concept behind voice recognition remains. Regarding the calculator half of this project, calculators are already firmly established devices in current society. A few possible implementations of a voice recognition calculator are mentioned above however there is scope for far more. Mankind in the 21st century is ever searching for ways to make things easier for ourselves and perhaps a voice recognition calculator is one way to accomplish that.

4.5 Ethical Considerations

The voice database for recognition is formed by the group of developers. Each of group member recorded his voice of saying from number one to nine, and the voice might contain private information. Failure to protect privacy will cause

customers' mistrust and further the loss of clients and business. If personal information is leaked by product, privacy breach causes the developers to be prosecuted.

To prevent the advent of privacy leak, appropriate procedures should be adopted to keep personal identities in product development. Some feasible solutions are listed:

1. Folders that contains voice information should be marked as "confidential" and follow some confidential terms.
2. The voice documents should be protected via firewalls, passwords and even especial encryption [16].

4.6 Commercialisation and Intellectual Property

Through critical analysis, the project is deemed as potential to be commercialized with some adjustments. Basically, it has two promising directions: children early-education and smart furniture.

In the field of children early-education, this project is considerably suitable for mathematical education for young children. Children can learn the arithmetic even when they are unable to write, who are expected to find the math learning more interesting with voice recognition calculator. If the development is in this direction, there are some suggestions for the project. First, the project needs to add a safe outer packaging to protect children when they use it. Second, the calculated number needs to be expanded instead of the range from 1-9. Finally, it is recommended to add some external devices to make the product more interesting, such as installing a speaker to feedback the results.

The other direction is to develop on smart furniture, such as voice control air conditioning and voice control light, which can adjust the brightness and mode. This direction focuses on the voice recognition part. Since the algorithm of the program is focus on isolated words recognition, the furniture cannot accept too complicated operations. The operations should be composed by a single word or phrase, such as "Open" or "Turn down".

4.7 Future Improvement

In this project, the Mel frequency is selected as the characteristic of the sound, and then uses the dynamic time warping to identify the sounds. This method has been proved to have a very high success rate (greater than 80%) for the identification of isolated words. When the project needs to further improve the success rate, the method of multiple database filtering is used. However, the testing results showed that this method only has a slight increase in the matching rate when the source of the sample and the database is the same person.

In the future study, there are two possible research directions that can improve the correct match rate. The first is to replace the core calculation method. For example, replace the DTW with Vector Quantization (VQ) or another model. The other one is to optimize the collection method of the database. The databases used in this project was recorded from the group members in the laboratory. It may influence the quality of the databases because of 3 reason. Firstly, the lab is full of irregular noise, which will affect the recording signal. It is suggested to record in the quiet space, and add white noise after recording. Then, all the group members are male. Therefore, the database will save more male voice features. Finally, people cannot control his voice perfectly, so that the result of each recording may be different. Thus, the database need to be set standard values that can be a reference to identify the quality of recording.

The one of the objective of this project is to build a simple calculator. However, due to lack of basic components, the calculator can only calculate the addition of numbers 1-9, which is very limited. However, the future study may be not to build a better calculator, such as adding arithmetic and calculate fractions. It is suggested to use the voice recognition on other works. For example, build an air conditioning control system with voice instructions.

4.8 Conclusions

After five weeks' hardworking, the team has successfully accomplished voice recognition software, calculator circuit and 7-Segement display part, which can basically achieve the function of speech recognition calculator. Unfortunately, the team failed to build a microphone IC to record voice input as a result that there is no correct waveform of output on oscilloscope, so we used the computer-build-in microphone to replace it, so there is no microphone circuit in the final deliverable. The final deliverable was an MATLAB software and Arduino UNO connected with 7-segement display. Following the instruction, User just needs to say two number to the microphone of computer, then he/she can receive the result from both the 7-segement display and computer display. The group members have successfully obtained basic knowledge on how to analyse a signal with different attributes and how to locate and correct the error within the circuit. Furthermore, the cooperation and ability of team members were developed during the whole project.

References

[1] N. Wu and B. Wang, “Process and Analysis of Voice Signal by MATLAB”, Faculty of Engineering and Sustainable Development from University of GAVLE, June 2014.

[2] M. Torlak, “Voice Transmission (Basic Concepts)”, EE4367 Telecom. Switching & Transmission from University of Texas at Dallas

[3] G. Boynton, “Chapter 11: Sound, the Auditory System, and Pitch Perception”, Psychology 333: Sensation and Perception from University of Washington, 2008

[4] P.I. Kattan and G.Z. Voyiadjis, “MATLAB Guide to Finite Elements: An Interactive Approach 1st ed.”, New York: Springer Berlin Hiedelberg, 2010.

[5] MathWorks, “Signal Processing Toolbox: Perform signal processing and analysis”, [online] 2017, https://uk.mathworks.com/products/signal.html?s_tid=srchtitle

[6] Moattar, M. H. and M. M. Homayounpour, “A simple but efficient real-time voice activity detection algorithm,” in 17th European Signal Processing Conf. (EUSIPCO 2009), Glasgow, Scotland, Aug. 24-28, 2009

[7] Sangwan, V. et al. “VAD techniques for real-time speech transmission on the Internet.” High Speed Networks and Multimedia Communications 5th IEEE International Conference on. IEEE, 2002.

[8] Tiwari, V. “MFCC and its applications in speaker recognition.” International journal on emerging technologies 1.1 (2010): 19-22.

[9] Kelly, A. C., and C. Gobl, “A comparison of mel-frequency cepstral coefficient (MFCC) calculation techniques.” Journal of Computing 3.10 (2011): 62-66.

[10] Prahallad, K. “Speech Technology: A Practical Introduction, topic: Spectrogram, Cepstrum and Mel-Frequency Analysis.”

[11] Gaikwad, S. K., B. W. Gawali, and P. Yannawar. “A review on speech recognition technique.” International Journal of Computer Applications 10.3 (2010): 16-24.

[12] Mohan, B. J. “Speech recognition using MFCC and DTW.” Advances in Electrical Engineering (ICAEE), 2014 International Conference on. IEEE, 2014.

[13] Johh, bakingham, “The application of Arduino in electronic circuit”, Dave aaraon, http://www.edgefxkits.com/blog/arduino-technology-architecture-and-applications/, April 2011.

[14] V. Robert, “the principle of 7-segement display.” Science and technology of the circuit. https://learn.sparkfun.com/tutorials/using-the-serial-7-segment-display, May 2012

[15] M.F. Jessica, “The core theory about 7-segement driver.” http://www.electronics-tutorials.ws/blog/7-segment-display-tutorial.html, July 2009

[16] J. Halpern, "The Importance of Confidentiality in the Workplace", Jules Halpern Associates LLC, 2017. [Online]. Available: https://www.halpernadvisors.com/why-is-confidentiality-important/.

Appendices

Appendix A

A.1 Photographs

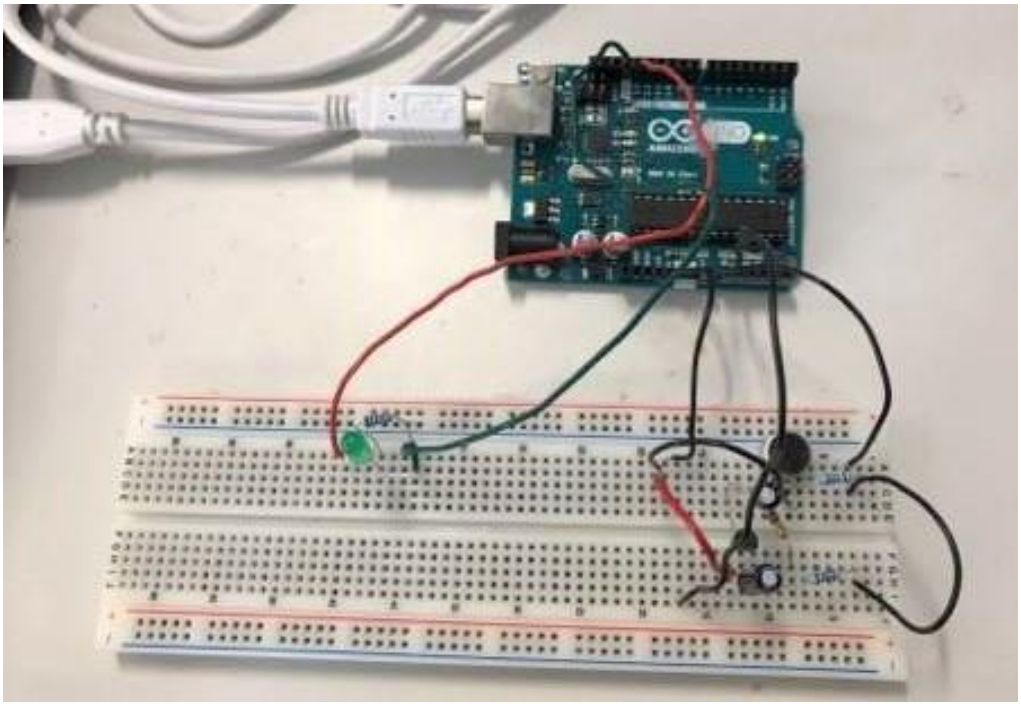

Figure A.1 Amplified microphone

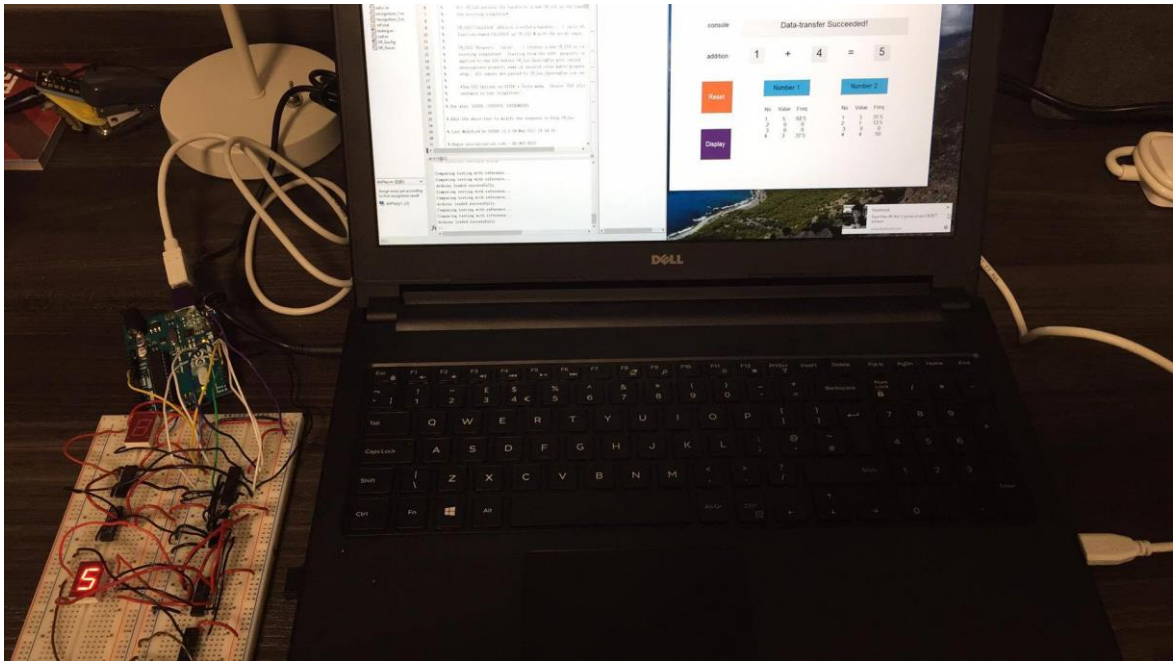

Figure A.2 Whole setup for deliverable

A.2 MATLAB Code and Programmes

GUI function as the interface with user:

```
function varargout = VR_Gui(varargin)
% VR_GUI MATLAB code for VR_Gui.fig
%      VR_GUI, by itself, creates a new VR_GUI or raises the existing
%      singleton*.
%
%      H = VR_GUI returns the handle to a new VR_GUI or the handle to
%      the existing singleton*.
%
%      VR_GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in VR_GUI.M with the given input arguments.
%
%      VR_GUI('Property','Value',...) creates a new VR_GUI or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before VR_Gui_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to VR_Gui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help VR_Gui

% Last Modified by GUIDE v2.5 09-Mar-2017 20:54:08

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @VR_Gui_OpeningFcn, ...
                   'gui_OutputFcn',  @VR_Gui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before VR_Gui is made visible.
function VR_Gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to VR_Gui (see VARARGIN)
```

```
% Choose default command line output for VR_Gui
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes VR_Gui wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = VR_Gui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
im = imread('liverpool.bmp');
axes(handles.axes1);
imshow(im);


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.text2,'String',' ');
set(handles.text3,'String',' ');
set(handles.text7,'String','Reset');
set(handles.text12,'String',' ');
set(handles.text13,'String',' ');
set(handles.text14,'String',' ');


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.text7,'String','Start speaking');
result1 = recognition_1;
set(handles.text7,'String','End speaking');
[~,y1]=max(result1);
global yD1
yD1 = y1(2);
set(handles.text2,'String',num2str(result1));
RR1 = 'The recognition result is ';
SC1 = [RR1, num2str(yD1)];
set(handles.text7,'String',SC1);
set(handles.text12,'String',num2str(yD1));


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
set(handles.text7,'String','Start speaking');
result2 = recognition_2;
set(handles.text7,'String','End speaking');
[~,y2]=max(result2);
global yD2;
yD2 = y2(2);
set(handles.text3,'String',num2str(result2));
RR2 = 'The recognition result is ';
SC2 = [RR2, num2str(yD2)];
set(handles.text7,'String',SC2);
set(handles.text13,'String',num2str(yD2));


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global yD1;
global yD2;
yD = yD1 + yD2;
set(handles.text14,'String',num2str(yD));
transFer = disPlay(yD1,yD2);
if (transFer)
    set(handles.text7,'String','Data-transfer Succeeded!');
else
    set(handles.text7,'String','Data-transfer Failed!');
end
```

Recognition function as the core programme to recognize the input voice (there are two identical recognition function, respectively responsible for two voice-input recognition):

```
function Rresult1 = recognition_1

% Loading the database
load('ref.mat');
% Recording new voice
R1 = audiorecorder(16000,8,1) ;
recordblocking(R1,2);

% Exact the voice
voice1=getaudiodata(R1);
[x1,x2]=vad(voice1);
m=mfcc(voice1);
m=m(x1:x2,:);

% Compare with database
disp('Comparing testing with reference...')
dist=zeros(1,9);
[~,b]=size(ref);
a=b/9;
result=zeros(1,a);
for i=1:a
    for j=i*9-8:i*9
    dist(1,j-9*i+9)=dtw(m,ref(j).m);
```

```
    [~,k]=min(dist(1,:));
    result(1,i)=k;
    end
end

% Output
Rresult1=tabulate(result);
[~,y1]=max(Rresult1);
[~,b1]=size(y1);
if b1==1
    y1(2)=Rresult1(1,1);
end
```

VAD function to extract meaning part of voice signal:

```
function [x1,x2] = vad(x)

% Unify amplitude to [-1,1]
x = double(x);
x = x / max(abs(x));

% assign Frame coefficients
FrameLen = 240;
FrameInc = 80;

amp1 = 10;
amp2 = 2;
zcr1 = 10;
zcr2 = 5;

maxsilence = 8;  % 6*10ms  = 30ms
minlen  = 15;    % 15*10ms = 150ms
status  = 0;
count   = 0;
silence = 0;

% Calculate over-zero-rate
tmp1  = enframe(x(1:end-1), FrameLen, FrameInc);
tmp2  = enframe(x(2:end)  , FrameLen, FrameInc);
signs = (tmp1.*tmp2)<0;
diffs = (tmp1 -tmp2)>0.02;
zcr   = sum(signs.*diffs, 2);

% Calculate instantaneous energy
amp = sum(abs(enframe(filter([1 -0.9375], 1, x), FrameLen, FrameInc)), 2);

% Adjust threshold value
amp1 = min(amp1, max(amp)/4);
amp2 = min(amp2, max(amp)/8);

% Terminal detection
x1 = 0;
x2 = 0;
for n=1:length(zcr)
   goto = 0;
   switch status
   case {0,1}                   % 0 = mute, 1 = maybe the beginning
```

```
      if amp(n) > amp1          % entering voice section
         x1 = max(n-count-1,1);
         status  = 2;
         silence = 0;
         count   = count + 1;
      elseif amp(n) > amp2 || ... % maybe voice section
             zcr(n) > zcr2
         status = 1;
         count  = count + 1;
      else                        % mute
         status  = 0;
         count   = 0;
      end
   case 2                        % 2 = voice section
      if amp(n) > amp2 || ...     % in voice section
         zcr(n) > zcr2
         count = count + 1;
      else                        % voice to end
         silence = silence+1;
         if silence < maxsilence % mute not enough
            count  = count + 1;
         elseif count < minlen   % voice too short, maybe noise
            status  = 0;
            silence = 0;
            count   = 0;
         else                    % voice ending
            status  = 3;
         end
      end
   case 3
      break;
   end
end

count = count-silence/2;
x2 = x1 + count -1;
```

Enframe function to detect voice signal activity:

```
function f=enframe(x,win,inc)

nx=length(x);
nwin=length(win);
if (nwin == 1)
   len = win;
else
   len = nwin;
end
if (nargin < 3)
   inc = len;
end
nf = fix((nx-len+inc)/inc);
f=zeros(nf,len);
indf= inc*(0:(nf-1)).';
inds = (1:len);
f(:) = x(indf(:,ones(1,len))+inds(ones(nf,1),:));
if (nwin > 1)
    w = win(:)';
```

```
    f = f .* w(ones(nf,1),:);
end
```

MFCC function to calculate Mel-coefficients:

```
function ccc = mfcc(x)
% Unify all the mel filters coefficient
bank=melbankm(24,256,8000,0,0.5,'m');
bank=full(bank);
bank=bank/max(bank(:));

% DCT coeffcient = 12*24
for k=1:12
  n=0:23;
  dctcoef(k,:)=cos((2*n+1)*k*pi/(2*24));
end

% Unify Cepstrum frame
w = 1 + 6 * sin(pi * [1:12] ./ 12);
w = w/max(w);

% Pre-enhance filter
xx=double(x);
xx=filter([1 -0.9375],1,xx);

% Voice signal frame
xx=enframe(xx,256,80);

% Calculate Mel coefficients for each frame
for i=1:size(xx,1)
  y = xx(i,:);
  s = y' .* hamming(256);
  t = abs(fft(s));
  t = t.^2;
  c1=dctcoef * log(bank * t(1:129));
  c2 = c1.*w';
  m(i,:)=c2';
end

% Differential coefficients
dtm = zeros(size(m));
for i=3:size(m,1)-2
  dtm(i,:) = -2*m(i-2,:) - m(i-1,:) + m(i+1,:) + 2*m(i+2,:);
end
dtm = dtm / 3;

% Combine differential coefficients and Mel coefficients
ccc = [m dtm];
% Delete beginning and ending frame
% Because these two have differential coefficient 0
ccc = ccc(3:size(m,1)-2,:);
```

Melbank function to determine matrix for a mel-spaced filterbank:

```
function [x,mn,mx]=melbankm(p,n,fs,fl,fh,w)
%MELBANKM determine matrix for a mel-spaced filterbank
```

```
[X,MN,MX]=(P,N,FS,FL,FH,W)
%
% Inputs:   p   number of filters in filterbank
%       n   length of fft
%       fs  sample rate in Hz
%       fl  low end of the lowest filter as a fraction of fs (default = 0)
%       fh  high end of highest filter as a fraction of fs (default = 0.5)
%       w   any sensible combination of the following:
%             't'  triangular shaped filters in mel domain (default)
%             'n'  hanning shaped filters in mel domain
%             'm'  hamming shaped filters in mel domain
%
%             'z'  highest and lowest filters taper down to zero (default)
%             'y'  lowest filter remains at 1 down to 0 frequency and
%              highest filter remains at 1 up to nyquist freqency
%
%              If 'ty' or 'ny' is specified, the total power in the fft
is preserved.
%
% Outputs:  x     a sparse matrix containing the filterbank amplitudes
%             If x is the only output argument then
size(x)=[p,1+floor(n/2)]
%             otherwise size(x)=[p,mx-mn+1]
%       mn   the lowest fft bin with a non-zero coefficient
%       mx   the highest fft bin with a non-zero coefficient
%
% Usage:    f=fft(s);           f=fft(s);
%       x=melbankm(p,n,fs);     [x,na,nb]=melbankm(p,n,fs);
%       n2=1+floor(n/2);        z=log(x*(f(na:nb)).*conj(f(na:nb)));
%       z=log(x*abs(f(1:n2)).^2);
%       c=dct(z); c(1)=[];
%
% To plot filterbanks e.g.  plot(melbankm(20,256,8000)')
%

if nargin < 6
  w='tz';
  if nargin < 5
    fh=0.5;
    if nargin < 4
      fl=0;
    end
  end
end
f0=700/fs;
fn2=floor(n/2);
lr=log((f0+fh)/(f0+fl))/(p+1);
% convert to fft bin numbers with 0 for DC term
bl=n*((f0+fl)*exp([0 1 p p+1]*lr)-f0);
b2=ceil(bl(2));
b3=floor(bl(3));
if any(w=='y')
  pf=log((f0+(b2:b3)/n)/(f0+fl))/lr;
  fp=floor(pf);
  r=[ones(1,b2) fp fp+1 p*ones(1,fn2-b3)];
  c=[1:b3+1 b2+1:fn2+1];
  v=2*[0.5 ones(1,b2-1) 1-pf+fp pf-fp ones(1,fn2-b3-1) 0.5];
  mn=1;
  mx=fn2+1;
else
  b1=floor(bl(1))+1;
```

```
  b4=min(fn2,ceil(bl(4)))-1;
  pf=log((f0+(b1:b4)/n)/(f0+fl))/lr;
  fp=floor(pf);
  pm=pf-fp;
  k2=b2-b1+1;
  k3=b3-b1+1;
  k4=b4-b1+1;
  r=[fp(k2:k4) 1+fp(1:k3)];
  c=[k2:k4 1:k3];
  v=2*[1-pm(k2:k4) pm(1:k3)];
  mn=b1+1;
  mx=b4+1;
end
if any(w=='n')
  v=1-cos(v*pi/2);
elseif any(w=='m')
  v=1-0.92/1.08*cos(v*pi/2);
end
if nargout > 1
  x=sparse(r,c,v);
else
  x=sparse(r,c+mn-1,v,p,1+fn2);
end
```

DTW function to find the smallest distance between two voice signal:

```
function dist = dtw(t,r)
n = size(t,1);
m = size(r,1);
d = zeros(n,m);
for i = 1:n
    for j = 1:m
        d(i,j) = sum((t(i,:) - r(j,:)).^2);
    end
end

D = ones(n,m) * realmax;
D(1,1) = d(1,1);
for i = 2:n
    for j = 1:m
        D1 = D(i-1,j);
        if j>1
            D2 = D(i-1,j-1);
        else
            D2 = realmax;
        end
        if j > 2
            D3 = D(i-1,j-2);
        else
            D3 = realmax;
        end
        D(i,j) = d(i,j) + min([D1,D2,D3]);
    end
end
dist = D(n,m);
end
```

Display function to communication with Arduino UNO and assign the pins' mode:

```
function Transfer = disPlay(y1,y2)

global ar;
transfer = false;

disp('Arduino loaded successfully');

% Assign every pin according to first recognition result
switch y1
    case 1
        writeDigitalPin(ar,'D2',1);
        writeDigitalPin(ar,'D3',0);
        writeDigitalPin(ar,'D4',0);
        writeDigitalPin(ar,'D5',0);
        transfer = true;
    case 2
        writeDigitalPin(ar,'D2',0);
        writeDigitalPin(ar,'D3',1);
        writeDigitalPin(ar,'D4',0);
        writeDigitalPin(ar,'D5',0);
        transfer = true;
    case 3
        writeDigitalPin(ar,'D2',1);
        writeDigitalPin(ar,'D3',1);
        writeDigitalPin(ar,'D4',0);
        writeDigitalPin(ar,'D5',0);
        transfer = true;
    case 4
        writeDigitalPin(ar,'D2',0);
        writeDigitalPin(ar,'D3',0);
        writeDigitalPin(ar,'D4',1);
        writeDigitalPin(ar,'D5',0);
        transfer = true;
    case 5
        writeDigitalPin(ar,'D2',1);
        writeDigitalPin(ar,'D3',0);
        writeDigitalPin(ar,'D4',1);
        writeDigitalPin(ar,'D5',0);
        transfer = true;
    case 6
        writeDigitalPin(ar,'D2',0);
        writeDigitalPin(ar,'D3',1);
        writeDigitalPin(ar,'D4',1);
        writeDigitalPin(ar,'D5',0);
        transfer = true;
    case 7
        writeDigitalPin(ar,'D2',1);
        writeDigitalPin(ar,'D3',1);
        writeDigitalPin(ar,'D4',1);
        writeDigitalPin(ar,'D5',0);
        transfer = true;
    case 8
        writeDigitalPin(ar,'D2',0);
        writeDigitalPin(ar,'D3',0);
        writeDigitalPin(ar,'D4',0);
        writeDigitalPin(ar,'D5',1);
        transfer = true;
    case 9
        writeDigitalPin(ar,'D2',1);
```

```
        writeDigitalPin(ar,'D3',0);
        writeDigitalPin(ar,'D4',0);
        writeDigitalPin(ar,'D5',1);
        transfer = true;
    otherwise
end

switch y2
    case 1
        writeDigitalPin(ar,'D6',1);
        writeDigitalPin(ar,'D7',0);
        writeDigitalPin(ar,'D8',0);
        writeDigitalPin(ar,'D9',0);
        transfer = true;
    case 2
        writeDigitalPin(ar,'D6',0);
        writeDigitalPin(ar,'D7',1);
        writeDigitalPin(ar,'D8',0);
        writeDigitalPin(ar,'D9',0);
        transfer = true;
    case 3
        writeDigitalPin(ar,'D6',1);
        writeDigitalPin(ar,'D7',1);
        writeDigitalPin(ar,'D8',0);
        writeDigitalPin(ar,'D9',0);
        transfer = true;
    case 4
        writeDigitalPin(ar,'D6',0);
        writeDigitalPin(ar,'D7',0);
        writeDigitalPin(ar,'D8',1);
        writeDigitalPin(ar,'D9',0);
        transfer = true;
    case 5
        writeDigitalPin(ar,'D6',1);
        writeDigitalPin(ar,'D7',0);
        writeDigitalPin(ar,'D8',1);
        writeDigitalPin(ar,'D9',0);
        transfer = true;
    case 6
        writeDigitalPin(ar,'D6',0);
        writeDigitalPin(ar,'D7',1);
        writeDigitalPin(ar,'D8',1);
        writeDigitalPin(ar,'D9',0);
        transfer = true;
    case 7
        writeDigitalPin(ar,'D6',1);
        writeDigitalPin(ar,'D7',1);
        writeDigitalPin(ar,'D8',1);
        writeDigitalPin(ar,'D9',0);
        transfer = true;
    case 8
        writeDigitalPin(ar,'D6',0);
        writeDigitalPin(ar,'D7',0);
        writeDigitalPin(ar,'D8',0);
        writeDigitalPin(ar,'D9',1);
        transfer = true;
    case 9
        writeDigitalPin(ar,'D6',1);
        writeDigitalPin(ar,'D7',0);
        writeDigitalPin(ar,'D8',0);
        writeDigitalPin(ar,'D9',1);
```

```
        transfer = true;
    otherwise
end

Transfer = transfer;
```