Hand Talk: Assistive Technology for the Speech Impaired

A Project Report

submitted by

ANISH TAMSE

(EE09B006)

in partial fulfilment of the requirements
for the award of the degrees of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.

MAY 2013

PROJECT CERTIFICATE

This is to certify that the project titled **Hand Talk: Assistive Technology for the Speech Impaired**, submitted by **Anish Tamse (EE09B006)**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the project work done by him under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Nagendra Krishnapura
Project Guide
Associate Professor
Dept. of Electrical Engineering
IIT Madras, Chennai 600 036

Prof. Enakshi Bhattacharya
Head
Dept. of Electrical Engineering
IIT Madras, Chennai 600 036

Place: Chennai

Date: June 7, 2013

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to all those who helped me in one way or the other with regard to the project. I would especially like to extend my appreciation to the following.

I thank my guide Dr. Nagendra Krishnapura, for his support and guidance during the course of the project

I would like to thank my friends Girish, Celestine and Sujan for their help with developing program on Android platform.

Last but not the least I would like to thank my family for their immense support and motivation during the four years and before that.

ABSTRACT

KEYWORDS: Flex sensors; Assistive technology; Data glove.

People who are speech impaired face great difficulty in daily life while communicating with others who do not understand sign language. This project proposes a low cost solution in the form of a glove. The glove is fitted with flex sensors which sense the bending of the finger joints. This data is sent wirelessly over Bluetooth. It is received and processed to understand the gesture being represented. Finally text to speech on an Android phone speaks out the gesture.

The flex sensors used in this project are not the commercially available ones. They are fabricated in lab using a piezoresistive material called velostat. The sensors made using velostat are much less expensive which helps in keeping the cost of the device low. The glove prototype is built and tested and various algorithms are discussed. Also the sensor construction and behaviour are dealt in detail.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS **i**

ABSTRACT **ii**

LIST OF TABLES **v**

LIST OF FIGURES **vii**

ABBREVIATIONS **viii**

1 INTRODUCTION **1**

1.1 American Sign Language (ASL) 2

2 IMPLEMENTATION **4**

2.1 Visual Sensing 4

2.2 Hardware Sensing 4

2.3 Hardware Implementation 6

2.3.1 Glove 6

2.3.2 Multiplexer 8

2.3.3 Analog to Digital Converter (ADC) 8

2.3.4 Microcontroller 8

2.3.5 Bluetooth Module 9

2.4 Software Implementation 9

2.4.1 Microcontroller Program 9

2.4.2 MATLAB Code 9

2.4.3 Mobile Software 10

3 CIRCUIT DETAILS **11**

3.1 Analog Multiplexer - MPC506A 12

3.2 Analog to Digital Converter – TLC549C 12

3.3 Microcontroller – MSP430G2231 13

3.4 Bluetooth module – EZ430RF2560 13

4 FLEX SENSORS 14

4.1 Construction 16

4.2 Modification 17

5 ALGORITHMS 19

5.1 Orthogonal Symbol Algorithm 19

5.2 Least Mean Square Algorithm 20

5.3 ASL (American Sign Language) Algorithm 21

6 RESULTS AND CONCLUSION 27

6.1 Scope for further development 28

6.2 Other Applications 28

A Codes 30

A.1 Microcontroller Code 30

A.2 MATLAB Codes 33

A.2.1 Mean Square Error Algorithm 33

A.2.2 ASL Algorithm 35

LIST OF TABLES

5.1 The ten finger joints for all the gestures of ASL for English letters classified as being fully open (1), fully closed (0) or indeterminate (x). The joints are numbered from 1 to 10 according to Figure 1.1. 22

LIST OF FIGURES

1.1 The gestures for all the English alphabets in ASL [8]. 3

2.1 The fifteen enumerated finger joints of our hand [7]. 5

2.2 The flow diagram briefing the high level implementation of the device. 5

2.3 A nitrile glove. 7

2.4 A cotton glove. This glove is placed within the nitrile glove for comfortable wearing. 7

3.1 Schematic of the system. 11

3.2 Pin diagram of the multiplexer MPC506A [9]. 12

3.3 Pin diagram of the ADC TLC549C [10]. 13

3.4 Pin diagram of the microcontroller used MSP430G2231 [11]. . 13

3.5 Bluetooth transceiver EZ430RF2560 [12]. 13

4.1 Typical resistance versus bend of the flex sensor fabricated for this project. The resistance change is very steep for bending angle between $40°$ and $80°$. 15

4.2 Output voltage of the resistive divider circuit versus bending with $V_{cc} = 3.5V$ and $R_{fixed} = 56k\Omega$. 16

4.3 Exploded view showing the internal structure of the sensor. . 16

4.4 The velostat layer is sandwiched between two layers of aluminium foil. Contact wires are yet to be placed. 17

4.5 A complete flex sensor. The setup of Figure 4.4 is now enclosed between layers of acetate sheet. 17

4.6 The modified flex sensor. This is the equivalent of two flex sensors. The gap between two different pieces of velostat is visible. One of the wires runs common to the two sensors. 18

4.7 Comparision between the gloves implemented with separate and combined sensors. 18

5.1 The ten finger joints are assigned numbers. This assignment is used in Table 5.1 [7]. 21

5.2 Decision chart for the first group of characters, wherein the four middle joints are closed. 24

5.3 Decision chart for the second group of characters, wherein three of the middle joints are closed. 25

5.4 Decision chart for the third group of characters. 26

6.1 The sensor data plotted in MATLAB for the glove being opened and closed repeatedly. The number of the plot refers to the joint according to Figure 5.1. The maximum value it can take is 255 ($2^8 - 1$) which comes about due to the 8 bit quantization of the ADC. 27

ABBREVIATIONS

IITM	Indian Institute of Technology, Madras
ASL	American Sign Language
ESD	Electrostatic-Sensitive Device
IC	Integrated Circuit
UART	Universal Asynchronous Receiver/Transmitter
SPI	Serial Peripheral Interface

CHAPTER 1

INTRODUCTION

Devices and gadgets that aid the differently-abled to lead normal and convenient lives has always been an area that has attracted innovation. Recent advancements in technology, like low-power electronics and wireless devices and ability to design both the analog front-end and digital processing back-end as integrated circuits has inspired a new range of wearable micro-devices. This project aims at developing a device with a motive to provide a low-cost solution to enable speech impaired persons to communicate using an artificial voice.

People who are speech impaired have difficulty to communicate with others who do not understand sign language. If there is an interface to convert the already existing sign language to speech, the ease of communication for speech impaired with others is increased significantly. The idea proposed in this project is to develop a glove with required sensors placed appropriately to capture the finger bends when worn. This data is then processed to recognize the gesture being shown. The glove would also have the additional capability of being able to learn new gestures from users, so that it can convert more gestures into speech across a wide pool of users and even in different languages.

The sensors being used on the glove are called flex sensors. These are passive sensors, resistive in nature. The resistance of the sensor decreases when the sensor is bent. The type of sensors used is limited only to flex sensors, i.e. other form of sensors like accelerometers and gyrometers are not used keeping in mind the cost effectiveness of the complete device. For this reason, it is difficult to capture the dynamic motion of the entire hand required for certain gestures. Dynamic gestures are the gestures involving motion of hand. This imposes certain limitations on the device.

The glove as a hardware is generic in nature, i.e. it can be used to learn any required gesture. But for implementing a practical sign language conversion on the device, many factors need to be looked at for choosing the sign language. Following are some of the important factors

- Popularity of the language
- Number of gestures in the language
- Number of dynamic gestures (involving complete hand movements)
- Ease of use of the language (whether the language communicates letter by letter or word by word)

Considering these factors into consideration, American Sign Language (ASL) is used in the project as it satisfies majority of the requirements.

1.1 American Sign Language (ASL)

American Sign Language is a very widely used sign language among the deaf communities of the world. It is the national sign language in United States, Canada, parts of Africa and some countries in South East Asia. Apart from these countries it is significant use in many other countries as well including India. The number of users for ASL in United States is estimated to be around 2,50,000 to 5,00,000 [3].

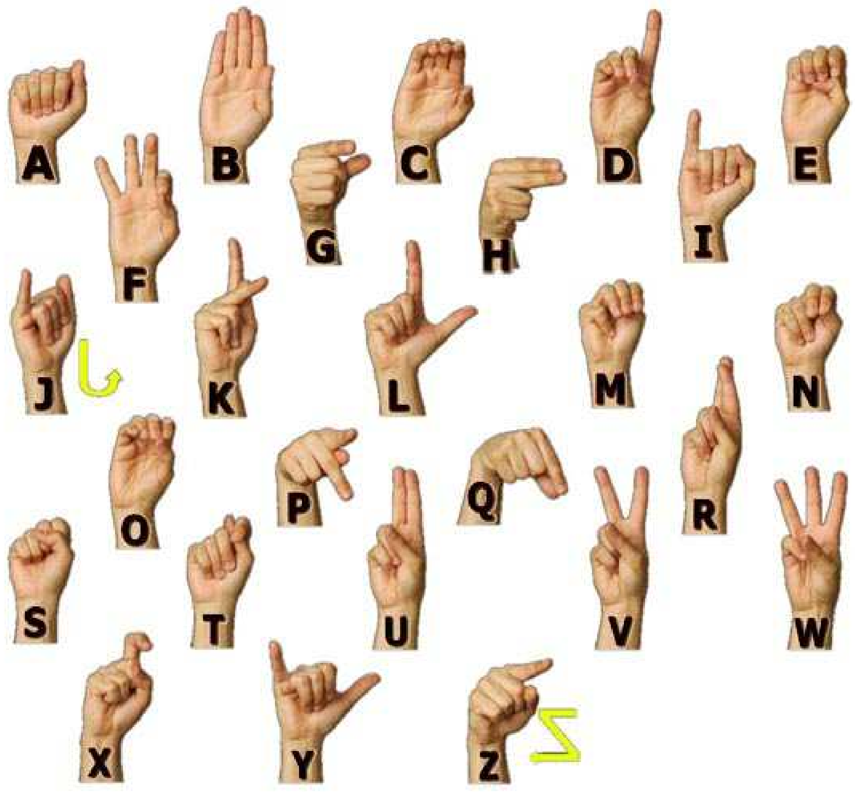

Figure 1.1: The gestures for all the English alphabets in ASL [8].

ASL has a gesture for each letter of the alphabet including gestures for common words as well. This project aims at converting the gestures for letters of this language. Also among the gestures for the letters only two involve hand movements. This helps reduce the complexity of the device significantly and makes the use of just flex sensors a feasible idea.

The disadvantage of using ASL in this manner is that the user needs to communicate letter by letter, instead of word by word. This makes the communication slower but possible nevertheless.

CHAPTER 2

IMPLEMENTATION

The previous implementations of the similar concept can be categorized into two types.

- Visual sensing
- Hardware sensing

2.1 Visual Sensing

In this method a camera is used to capture the image of the hand making the gesture. Various techniques in image processing are then used to decide which gesture is being represented by the hand. The advantage of this method is the accuracy of the gesture recognized is very high. But the problem associated with this method is that the system becomes bulky as the camera needs to be carried around. Also since image processing is involved, the processing is fairly intensive. Various aspects of this method are addressed in detail in [1].

2.2 Hardware Sensing

This method makes use of sensors such as flex sensors, accelerometers, gyrometers etc. to recognize the gesture being represented. The major advantage of this implementation is the whole product can be made very compact.

In this project, the implementation is done using only flex sensors. Also these flex sensors are fabricated from a material called velostat in the lab and are much less expensive than commercially available ones. The fabrication process is described in detail in Section 4.1. We will also look at the limitations of constraining ourselves to using just flex sensors.

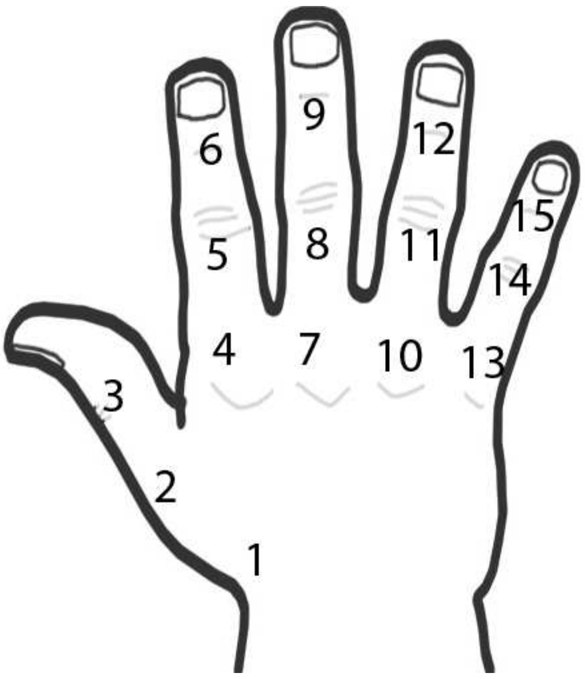

Figure 2.1: The fifteen enumerated finger joints of our hand [7].

Our hand has fifteen finger joints, three joints on each finger. But the joints near the finger tips except for the thumb (joints 6, 9, 12 and 15) are not totally independent, i.e. these joints move only when the middle joint (joints 5, 8, 11 and 14) of the corresponding finger is completely closed. Hence for recognizing the gestures, these joints are not very significant. Hence in this project only the remaining ten joints are considered for guessing the gesture. Most of the signs in American Sign Language can be interpreted in this manner, by using these ten joints.

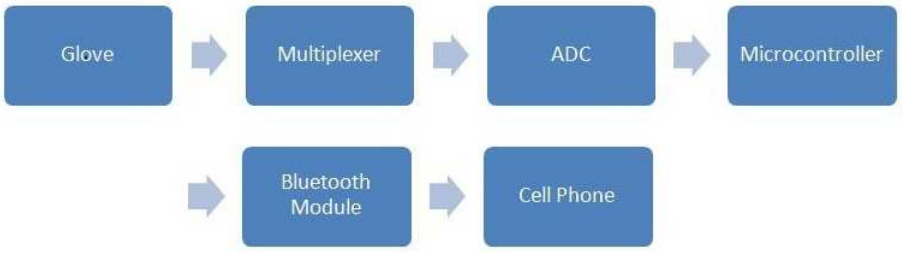

Figure 2.2: The flow diagram briefing the high level implementation of the device.

The implementation is summarized in Figure.2.2. The glove is designed with sensors to capture bending of all the fingers. This bending is converted to a change in the voltage level. Thus for ten sensors, we monitor ten voltage levels. These voltage levels are then time multiplexed into a single signal which is read and decoded by a microcontroller. The microcontroller is interfaced with a Bluetooth module which sends data to a processor. In this project the processing is demonstrated using MATLAB. The processor then receives the data and processes it to understand which symbol is being represented by the hand. Once a whole word is received, the text to speech algorithm available on the phone converts it to voice. The actual components used and the circuit level implementation are shown in Chapter 3.

2.3 Hardware Implementation

This section describes the different parts of the hardware.

2.3.1 Glove

The glove on which the sensors are attached is a nitrile glove. This glove is elastic in nature. This property helps the glove follow the contours of the hand closely. Thus the sensors attached on top of the glove give a close estimate of the bending. The glove when used over a long period of time becomes sticky on the inside and isn't very comfortable to use. Hence a cotton glove is provided on the inside for comfort of the user.

Figure 2.3: A nitrile glove.

Figure 2.4: A cotton glove. This glove is placed within the nitrile glove for comfortable wearing.

The glove is implemented with ten flex sensors, two on each finger. The construction and properties of the flex sensors are dealt with in detail in Chapter 4. The change in the resistance of the flex sensors is significant (over ten times). Hence the variation in the resistance of the flex sensor is converted to voltage variation with a simple resistive divider. Thus in total, we have ten voltage dividers and ten voltage levels to be monitored.

The voltages are sampled and then multiplexed. The sampling rate is not critical as hand movements are much slower than the working frequencies of the ICs used. Choosing a high sampling rate unnecessarily increases the processing required by the microcontroller. Hence we chose the lowest sampling rate which serves our purpose. A sampling rate of 20 Hz was chosen as it seemed sufficient to capture quick hand movements.

2.3.2 Multiplexer

These ten voltage variations are time multiplexed using an analog multiplexer (MPC506A). The multiplexer is controlled by the microcontroller. The multiplexer reads each sensor data for 5ms. Hence for once cycle of readings, the time taken is 50ms. Thus we get 20 readings every second.

2.3.3 Analog to Digital Converter (ADC)

The signal from the multiplexer is then fed into an Analog to Digital Converter (TLC549C). The ADC used has an 8 bit resolution. The ability of this ADC to serialize the digital output code and the ability to use an external clock control to tap out the digital data at our own convenience makes it adaptable and increases the ease of interfacing it with the microcontroller.

2.3.4 Microcontroller

This stream of data is then taken by the microcontroller (MSP430G2231) and de-multiplexed. The microcontroller packs the 8 bit values for each of the ten sensors into one large array. A marker byte is prefixed to this array. This array is sent out serially using the available UART interface. The microcontroller acts as an interface between the ADC, multiplexer and the Bluetooth module. It controls the multiplexer by giving it the address of the voltage level to be sent through. The microcontroller then reads the digital signal from ADC serially. The microcontroller is capable of serial data transmission and reception, which has been made use of for on-board communication. The processing on the microcontroller is minimal so that a more powerful processor can be used to do the processing off board. This is important especially keeping because the device is a communication device for the differently abled and lag in the device response becomes an inconvenience.

2.3.5 Bluetooth Module

A Bluetooth module (EZ430RF2560) receives this array using its UART interface. The module is coded to transmit the data received as it is over Bluetooth wirelessly. It transmits the data to MATLAB running on PC using SPI (Serial Peripheral Interface) protocol. The baud rate being used for this communication is 11520. The Bluetooth communication link is chosen because of the popularity of Bluetooth. Further, most modern day mobile phones support Bluetooth; hence the final text to speech can be implemented on any phone.

2.4 Software Implementation

This section deals with the software functionalities and how they are implemented.

2.4.1 Microcontroller Program

The code on the microcontroller receives the data from the ADC serially and de-multiplexes it. The data is then arranged in the form of a packet with a marker character and sent to the Bluetooth module for transmitting. The communication with the Bluetooth module is through serial interface using UART protocol. The code is provided in Appendix A.1.

2.4.2 MATLAB Code

MATLAB is chosen for processing because it offers a very good interface for testing and visualizing data and debugging different algorithms and approaches.

MATLAB receives the data over Bluetooth link using SPI protocol. The baud rate used is 11520. The code then searches for appropriate marker byte and maps the sensor values to the corresponding fingers. These sensor values are used as input data for the algorithm to be used.

For the orthogonal symbol algorithm, the sensor values are converted to 1s or 0s, which correspond to a particular joint being open or closed respectively. This is then compared against an available database of gestures to recognize the gesture being represented.

For the least mean square algorithm, a database of values for different gestures is maintained. The mean square error between the values which are read and each of the gestures in the database is found. The one with the least mean square error is given as the correct guess.

For the ASL (American Sign Language) algorithm, the code goes through a tree which is designed specifically for the purpose, and determines the closest alphabet being represented. The various algorithms for different purposes are described in detail in Chapter 5. The result is then sent to the mobile for converting the text to speech.

2.4.3 Mobile Software

The mobile program is written for Android platform. It receives the data character by character over Bluetooth link using SPI protocol. The baud rate used is 11520. Android provides a native speech synthesiser feature. This is made use of to convert the received text to speech.

CHAPTER 3

CIRCUIT DETAILS

The open state resistance of the sensors is more than $800k\Omega$ and the resistance when the sensors are fully bent is around $20k\Omega$. Since the change is quite significant, just using the sensor in a resistive divider configuration suffices to convert the resistance variation to voltage variation. No amplifier or signal conditioning circuitry is required which helps to maintain a low cost.

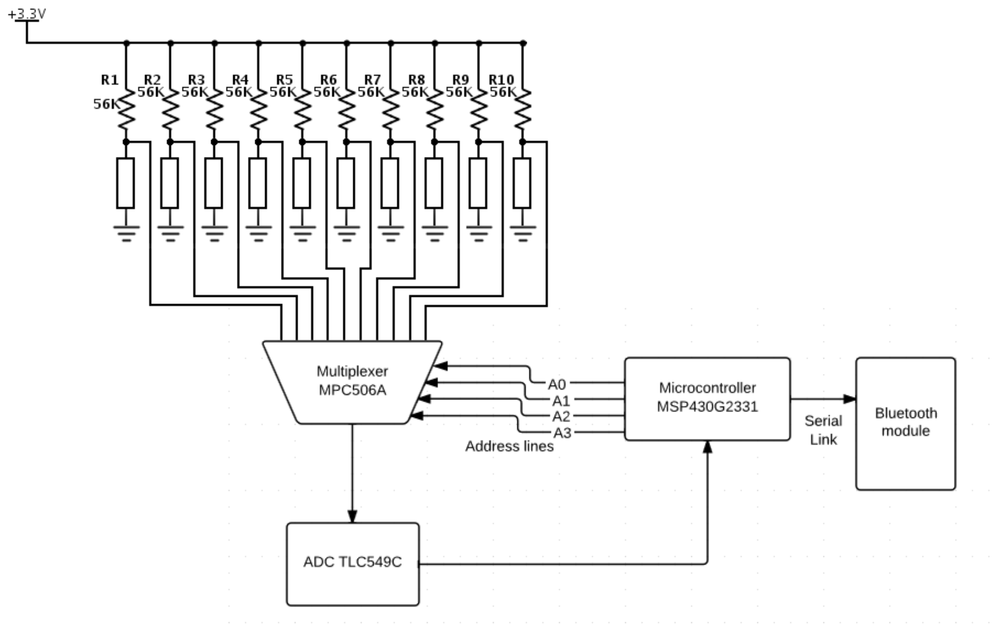

Figure 3.1: Schematic of the system.

Looking at the highest and lowest values for the flex sensor, i.e. $800k\Omega$ and $20k\Omega$, a series resistance of $56k\Omega$ is chosen to obtain a suitable change in detected voltage between the two states. The other components used in the circuit are described in detail below.

3.1 Analog Multiplexer - MPC506A

This is a single ended 16 channel CMOS analog multiplexer. It can operate for analog signals ranging between -15V and 15V, though in the circuit, it is being used only between 0V and 3.5V. It requires a minimum supply voltage of 3.5V. It consumes an idle state power of 7.5mW. The pin diagram for the multiplexer is given in Figure 3.2.

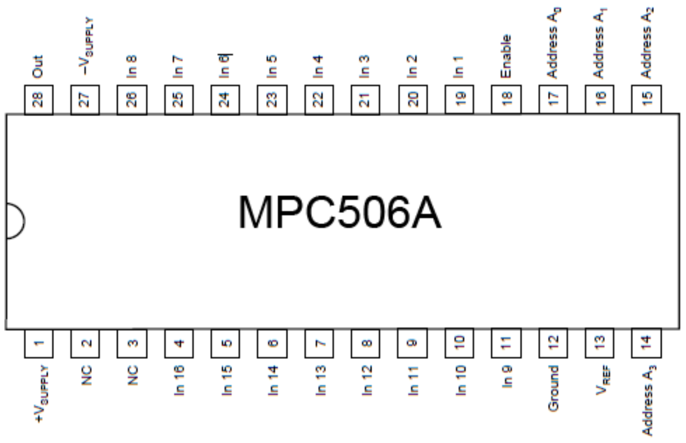

Figure 3.2: Pin diagram of the multiplexer MPC506A [9].

3.2 Analog to Digital Converter – TLC549C

This is an 8 bit resolution analog to digital converter. It takes input as a differential voltage. Since in the circuit, the voltage requiring conversion is not differential, the negative reference voltage for the ADC is grounded. It takes a maximum time of 17μs to convert the voltage to a digital value. The IC operates over a supply range of 3V to 6V. The maximum power consumption of the IC is 15mW. The digital output is given serially. The pin diagram for the ADC is given in Figure 3.3.

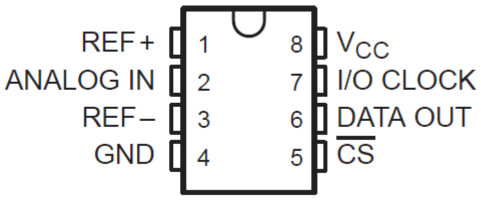

Figure 3.3: Pin diagram of the ADC TLC549C [10].

3.3 Microcontroller – MSP430G2231

This is a 16 bit microcontroller and works over a supply range of 1.8V to 3.6V. It has universal serial interface supporting SPI which is being used in the project. It consumes power of 0.5mW in active mode. Pin diagram for the microcontroller is given in Figure 3.4

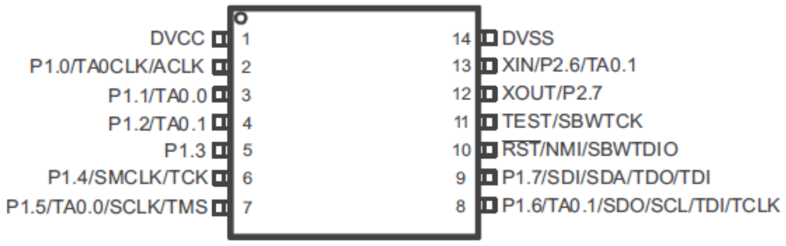

Figure 3.4: Pin diagram of the microcontroller used MSP430G2231 [11].

3.4 Bluetooth module – EZ430RF2560

This can send and receive data over Bluetooth. It works over a supply range of 3V to 5V. In the project it is being used to send data serially at a baud rate of 11520.

Figure 3.5: Bluetooth transceiver EZ430RF2560 [12].

CHAPTER 4

FLEX SENSORS

The sensors which are used for sensing the bends on the glove are called flex sensors or bend sensors. There are ten such sensors placed on the glove, two sensors for each finger. These sensors are available commercially, but keeping in mind different lengths of sensors required for each joint and overall cost effectiveness of the project, these sensors were handmade. The sensors are constructed from a material called velostat. Velostat is a packaging material used for transporting items or devices prone to damage from electrostatic discharge (ESD). It is a polymer which is impregnated with carbon black to make it conducting. It is available for a very low cost in the form of ESD bags [13].

Velostat has piezo-resistive properties, i.e. its resistance changes when pressure is applied on it. This property of velostat is leveraged in making of the flex sensor. The idea of using velostat for making flex sensors has been used by hobbyists for various projects as a low cost alternative [5], [6]. The idea of velostat flex sensors being used for interpreting sign language is original. The project also makes a few modifications to the construction to suit the purpose of glove based gesture recognition.

The flex sensor has two terminals. The resistance between these terminals varies according to the bend of the sensor, the more the bending the lower the resistance. The variation of resistance with the bend is linear in the commercially available sensors. The trade-off with using the sensor made using velostat is the change is non linear (resistance changes rather abruptly after particular degrees of bend) and the sensor is less robust physically.

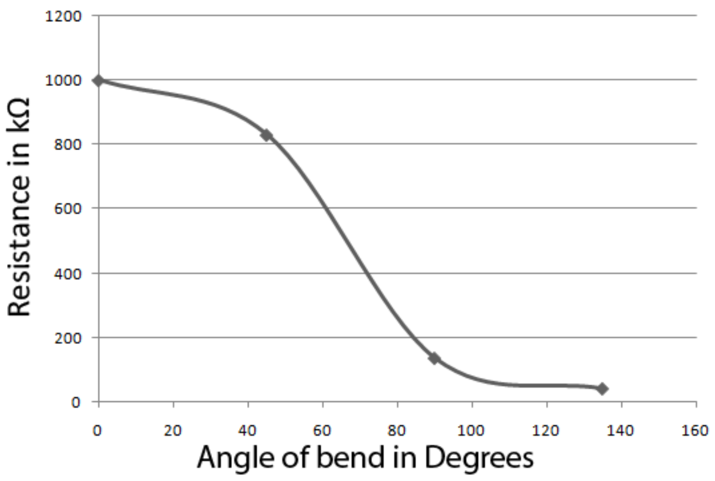

Figure 4.1: Typical resistance versus bend of the flex sensor fabricated for this project. The resistance change is very steep for bending angle between $40°$ and $80°$.

The resistance changes between $1M\Omega$ to $0.05M\Omega$. Since the change in resistance is significant, a simple resistive divider with one of the arms as the sensor suffices to convert the bending into voltage.

The output voltage of the voltage divider is given by $V_{CC}\frac{R_{sensor}}{R_{sensor}+R_{fixed}}$. Figure 4.2 shows how the voltage of the resistive divider varies with the bending. It can be seen that the plot is very noisy when the sensor is being bent. Once it is held stationary at a particular angle of bend, the noise is reduced. In the plot after reaching a bend of $135°$, the bending is stopped and reduction in noise can be seen.

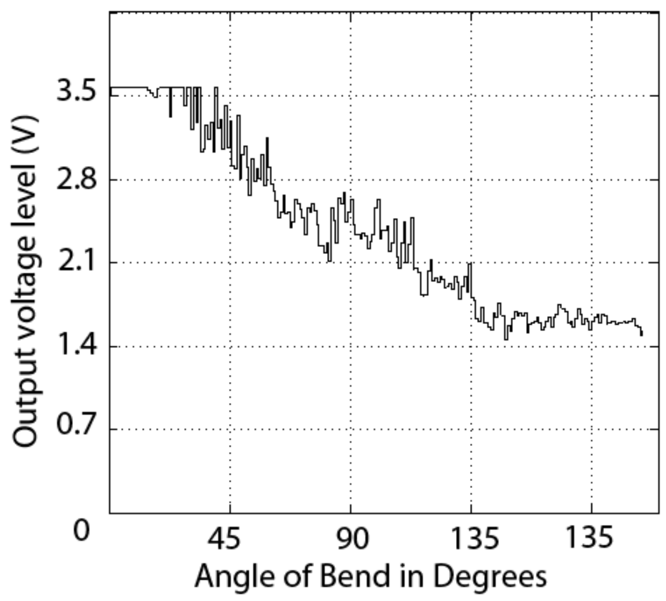

Figure 4.2: Output voltage of the resistive divider circuit versus bending with $V_{cc} = 3.5V$ and $R_{fixed} = 56k\Omega$.

4.1 Construction

The basic idea behind the construction of the velostat flex sensor is that the velostat layer should be pressed between two pieces of a flexible material when bent. The flexible material used here is acetate sheet. The sheet of velostat placed in the sensor is bi-layered. This was decided after experimenting with different number of layers. Using two layers gives the best variation in resistance.

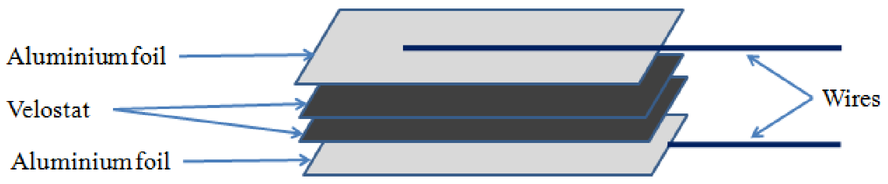

Figure 4.3: Exploded view showing the internal structure of the sensor.

Two pieces of required dimension of velostat is cut out and aluminium foil is placed on either sides of the two layered velostat as shown in Figure 4.3. The aluminium foil is used to make proper contact with the velostat. After this, two wires are placed touching the aluminium foil, one on either side. These wires act as the terminals of the sensors.

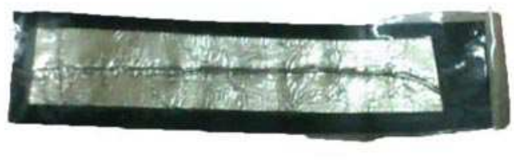

Figure 4.4: The velostat layer is sandwiched between two layers of aluminium foil. Contact wires are yet to be placed.

Now this whole setup is sandwiched between two layers of acetate sheet. Acetate sheet is a transparent plastic layer which is flexible. Doing this gives the required elasticity to the sensor while maintaining the flexibility, i.e. the sensor comes back to its original shape after bending. Also it serves the important purpose of applying pressure on the velostat when the sensor is bent, as the velostat is sandwiched between two layers of the sheet.

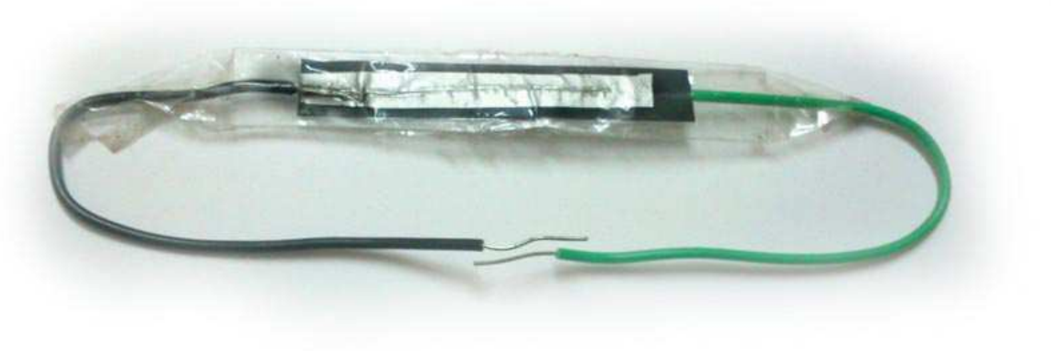

Figure 4.5: A complete flex sensor. The setup of Figure 4.4 is now enclosed between layers of acetate sheet.

4.2 Modification

We require two of these sensors on each finger to sense the bending in the two joints. But having two on each finger makes the glove very cluttered and more prone to damage due to exposed connections. Thus pair of these sensors is placed together and one terminal is made common for the (ground). Also all the three wires are brought out from the same side, making the glove tidier.

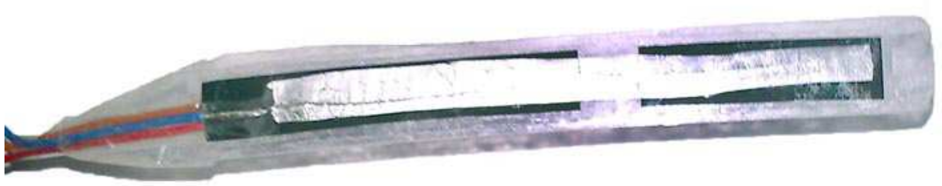

Figure 4.6: The modified flex sensor. This is the equivalent of two flex sensors. The gap between two different pieces of velostat is visible. One of the wires runs common to the two sensors.

Figure 4.6 shows the modified sensor. The advantage of using the modified sensor for neatness can be seen in Figure 4.7.

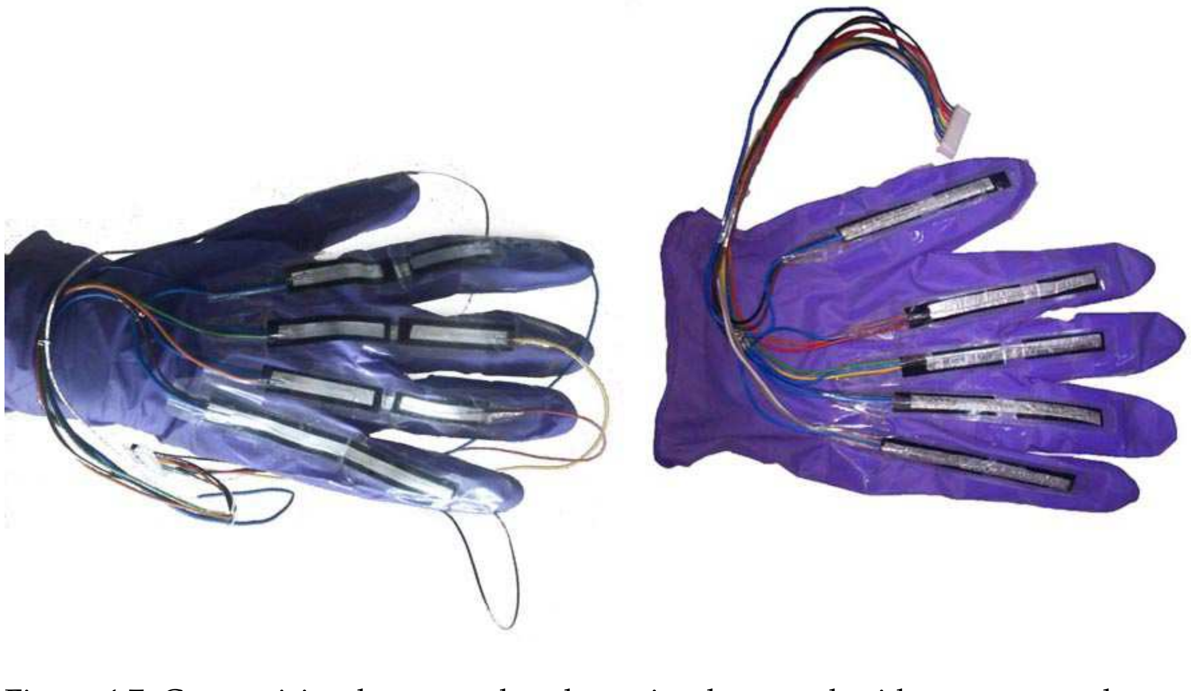

Figure 4.7: Comparision between the gloves implemented with separate and combined sensors.

After placing all the sensors on the glove, all the grounds are brought together and the ten sensor terminals are pulled out in the form of a ten pin connector.

CHAPTER 5

ALGORITHMS

Once we have received the raw data from the sensors, we need to have appropriate algorithms to make sense out of it and assign it to the closest gesture known. As we have seen in Chapter 4 the unbent and bent state values for the sensors are distinctly different. Hence we can easily detect at least 2^{10} different gestures with very high accuracy. But not all of the symbols of the American Sign Language (ASL) belong to these 1024 gesture set, i.e. most of the symbols have some joints which are half bent. Hence we need to rely on modified algorithms for better gesture recognition for various purposes.

5.1 Orthogonal Symbol Algorithm

This is the most rudimentary form of gesture recognition. This allows only for the gestures which have joints which are either fully bent or unbent. The joints cannot be bent halfway.

Given the constraints, there are 2^{10} distinct orthogonal gestures possible. Each of these gestures or a subset of the same is mapped to a letter/word. A look-up table is then created and any new symbol can be added to the database.

Next a threshold vector is stored, which essentially means the threshold value for each of the sensors above which they are deemed unbent and below that bent. When a gesture is represented, the sensor values are compared against the threshold and another binary vector is created, with 1 representing open and 0 representing closed. This is then looked up against the look-up table and the matching symbol is given as output.

Since this doesn't allow joints which are bent halfway, it means that none of the practical sign languages can be implemented using this method. But a major advantage is it offers very high accuracy for the gestures it recognizes.

Another advantage is the number of gestures possible is quite high (1024). This means that apart from just letters, we can also map complete words. Thus the large number of gestures combined with high accuracy still keeps this method as a contender for the choice of algorithms.

Thus if one is willing to map a new set of gestures to words/letters and learn them, this algorithm proves to be an effective method.

5.2 Least Mean Square Algorithm

Here we store the sensor values associated with each of the gestures unlike the previous method where only the binary values are stored. When a new gesture is made, the sensor values are compared with all the stored values and the closest match is given as output. This is done as follows.

- Read the sensor value vector from the glove.
- Subtract the sensor value vector from the stored vectors for each gesture to compute the error vector with respect to each gesture.
- Compute the mean square error for each gesture by squaring and adding the elements in each error vector.
- Find the lowest of these errors and the gesture corresponding to it.

This method can also be thought of as mapping the recorded gestures on a ten-dimensional vector space. This is the signal constellation of the database. When a new gesture is made, it is placed on this signal constellation and the gesture from database at the least distance from the new gesture is given as the result. Also we can always add a new symbol to the database

Naturally the accuracy of this method is only as good as the separation of the database gestures in the signal constellation. If the separation is large and uniform among the symbols, the accuracy will tend to be better.

The advantage of this method is that we can now have half bent joints, which in principle allows any kind of gesture. When we implement this method for alphabet of a language, there is a possibility of confusion between letters of similar gestures (example: for American Sign Language, it will most likely confuse between I and J, G and Q, and U and V). This can effectively be solved by implementing spell check and autocorrect (Example: if the complete word is being spelled as JCE in ASL, it could equally likely mean ICE which is most probably the correct word). The different possibilities for the autocorrect can be obtained by asking the code for two most likely output alphabets instead of a single output.

The disadvantage of this method is that it always gives a result no matter how different the represented gesture is. This can be countered by including a threshold value for the mean squared error such that the output is given only if it is below the threshold.

5.3 ASL (American Sign Language) Algorithm

Since we are designing the glove for ASL, we have all the gestures at our disposal beforehand. Therefore we can take advantage of this knowledge and algorithm for detecting gestures can be specifically tailored for these signs. Table 5.1 is developed after referring to the Figure 1.1, which shows the various gestures for English alphabets in ASL. The cells with 1 refer to corresponding joint being open. Cells with 0 refer to the joint being closed. If a cell is marked x, it means that the joint is neither completely open nor completely closed.

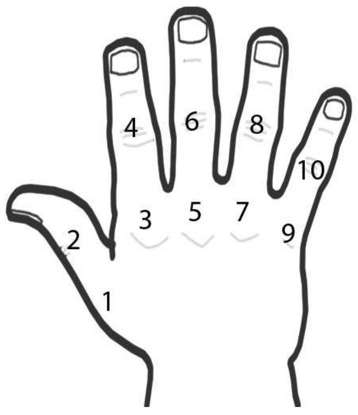

Figure 5.1: The ten finger joints are assigned numbers. This assignment is used in Table 5.1 [7].

Table 5.1: The ten finger joints for all the gestures of ASL for English letters classified as being fully open (1), fully closed (0) or indeterminate (x). The joints are numbered from 1 to 10 according to Figure 1.1.

Letter\Joint No.	1	2	3	4	5	6	7	8	9	10
A	1	1	0	0	0	0	0	0	0	0
B	1	1	1	1	1	1	1	1	1	1
C	X	0	1	0	1	0	1	0	1	0
D	X	X	1	1	1	0	1	0	1	0
E	0	0	1	0	1	0	1	0	1	0
F	X	X	X	0	1	1	1	1	1	1
G	1	1	0	1	0	0	0	0	0	0
H	X	X	0	1	0	1	0	0	0	0
I	X	X	0	0	0	0	0	0	1	1
J	X	X	0	0	0	0	0	0	1	1
K	1	1	1	1	0	1	0	0	0	0
L	1	1	1	1	0	0	0	0	0	0
M	0	0	1	0	1	0	1	0	0	0
N	0	0	1	0	1	0	0	0	0	0
O	1	X	1	0	1	0	1	0	1	0
P	1	1	1	1	0	1	0	0	0	0
Q	1	1	0	1	0	0	0	0	0	0
R	X	1	1	1	1	1	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0
T	X	1	0	X	0	0	0	0	0	0
U	0	1	1	1	1	1	0	0	0	0
V	0	1	1	1	1	1	0	0	0	0
W	0	1	1	1	1	1	1	1	X	0
X	X	0	1	0	0	0	0	0	0	0
Y	1	1	0	0	0	0	0	0	1	1
Z	X	X	0	1	0	0	0	0	0	0

After studying the signs for all the alphabets, they can be classified into three broad categories.

- Group 1 -The four middle joints 4,6,8 and 10 are closed
- Group 2 -Three of the four middle joints are closed
- Group 3 -Others

The reason for using the middle joints for the broad classification is that the middle joints are totally uncorrelated with the other joints. That is, the knuckle joints (3,5,7 and 9) are correlated with their corresponding neighbouring knuckle joints. (Example: If joint 5 is bent fully, the joints 3 and 7 are also bent to some extent because of their close placement on the glove.) Hence their values are not as reliable as the middle joints.

We have roughly equal number of letters in each category. Now, the letters in each category are further classified based on the differences. The differentiation for each of the categories is described further using flowcharts.

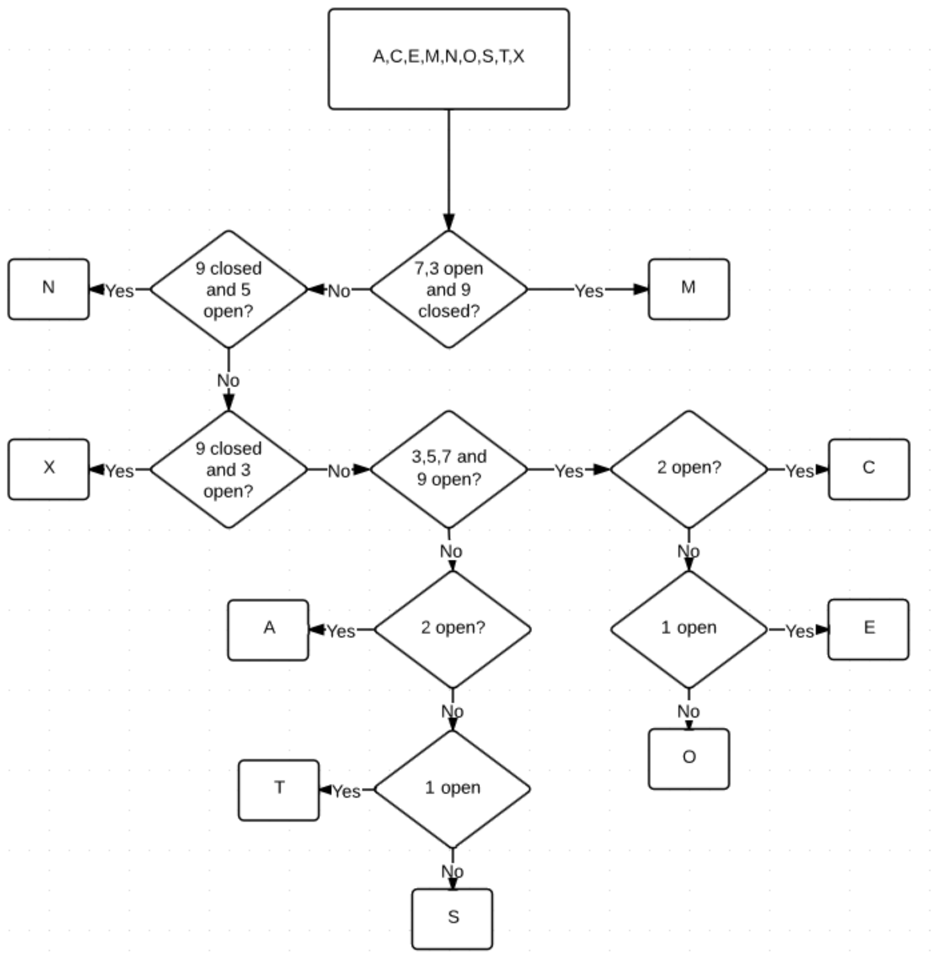

Figure 5.2: Decision chart for the first group of characters, wherein the four middle joints are closed.

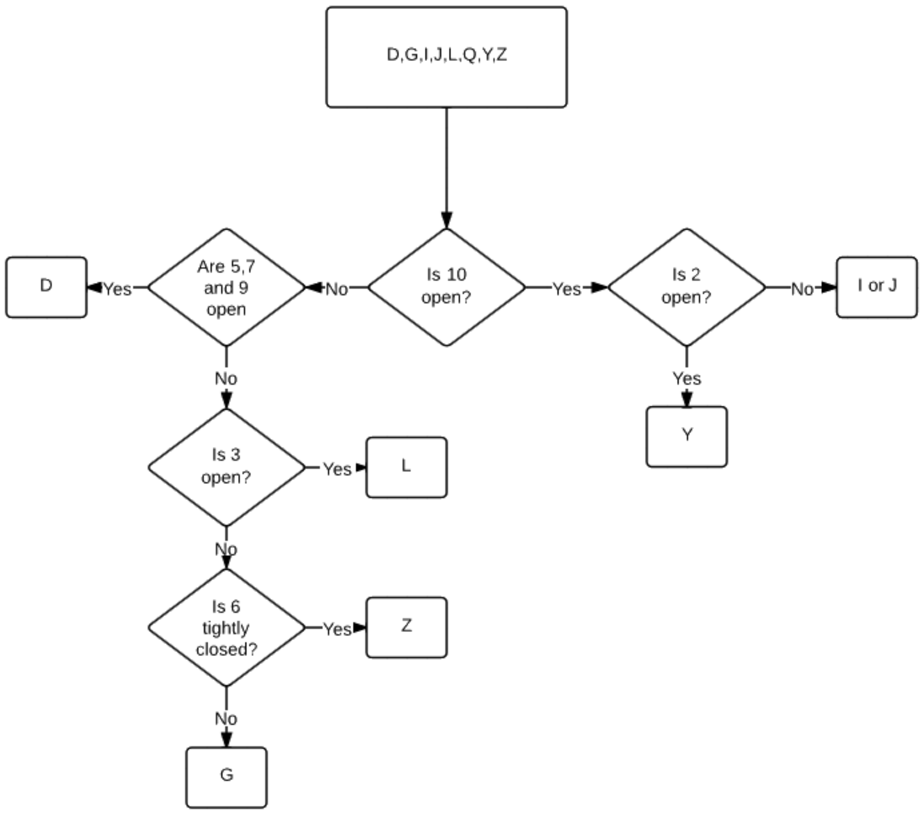

Figure 5.3: Decision chart for the second group of characters, wherein three of the middle joints are closed.

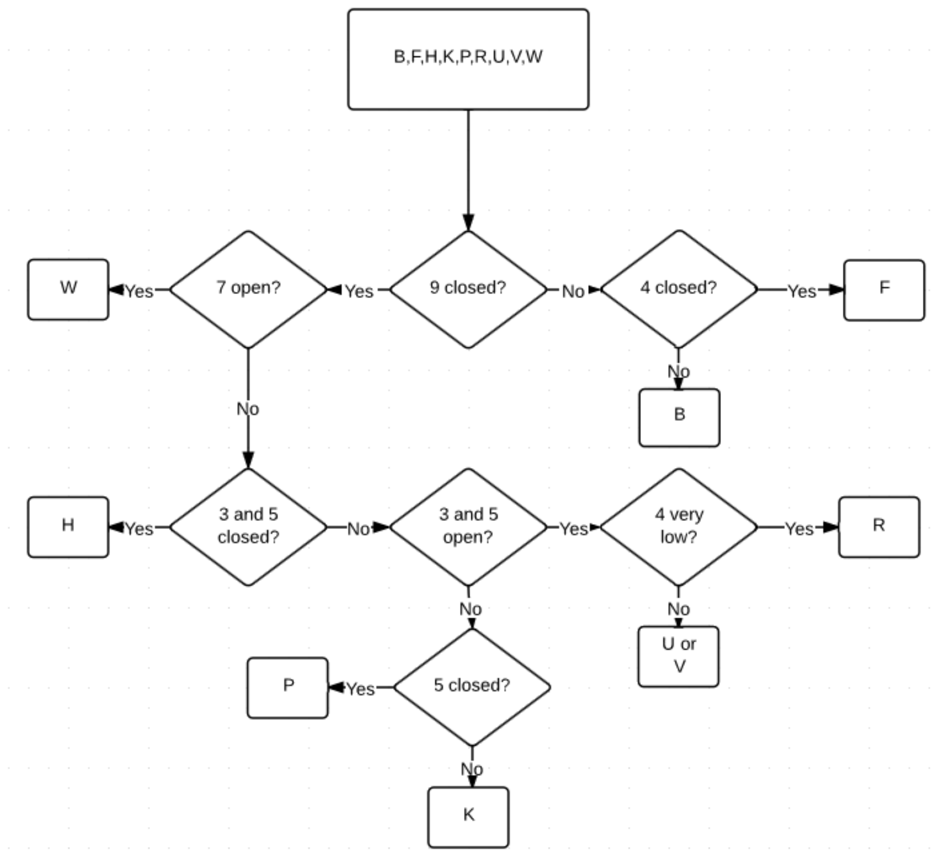

Figure 5.4: Decision chart for the third group of characters.

This method is an provides an original classification of the letters of the ASL for glove based gesture recognition. We can see that most of the alphabets can be classified, except three groups each of two characters, namely I and J, G and Q and U and V.

CHAPTER 6

RESULTS AND CONCLUSION

The data from the sensors can be visualized after having received it using MATLAB. Figure 6.1 shows the sensor data for the ten joints in a particular case. The glove was being closed and opened repeatedly, hence the wavy nature of the plot.

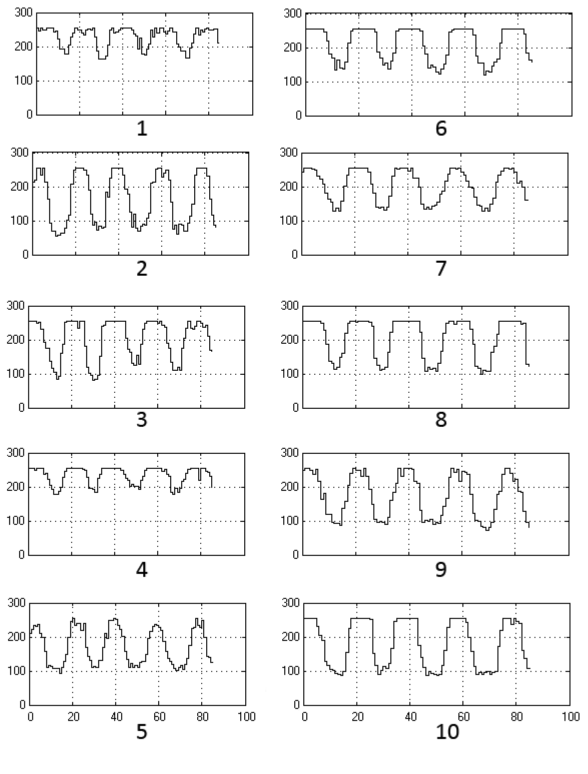

Figure 6.1: The sensor data plotted in MATLAB for the glove being opened and closed repeatedly. The number of the plot refers to the joint according to Figure 5.1. The maximum value it can take is 255 ($2^8 - 1$) which comes about due to the 8 bit quantization of the ADC.

When implemented using orthogonal symbol algorithm, the gesture recognition works with an almost 100 percent accuracy as expected. But it has its own limitation, that is half bent joints not being allowed.

The mean square error algorithm works quite well for some gesture set and not so well for others. The accuracy for this depends totally on the gesture set and cannot be quantified in general.

The ASL algorithm works with 80 to 90 percent accuracy for most of the symbols, the exception being confusion between gestures I and J, gestures G and Q and gestures U and V.

6.1 Scope for further development

Currently the glove is being implemented only using flex sensors. If additional sensors like accelerometers and gyrometers are added, the glove would be able to capture dynamic motions also. This will significantly improve the range of gestures which can be recognized by the device. Another area to develop further would be the algorithm for recognizing gestures. More robust machine learning algorithms can be used to improve gesture recognition accuracy.

6.2 Other Applications

The use of the glove need not be limited to just conversion of sign language to speech. It is a very generic hardware and finds many applications in today's world. Another bio-medical application for this glove could be to measure the developments in hand paralysis patients. The glove can quantify the strain a person can exert and hence quantify the improvement.

The glove could be used as a 3-D mouse for computers. The conventionally used mouse provides data for the pointer location along two axes. A 3-D mouse adds data along a third axis. This can be implemented based on the bending of any three fingers. The more a finger bends, the faster the movement in corresponding direction. 3-D mouse can be used for improving the ease to work with applications such as 3-D modelling, 3-D graphic designing or working with various CAD tools.

Another computer related application could be the use of the glove as a gaming device [14]. A similar glove was launched by Nintendo as an additional hardware for their gaming platform. This glove could prove a low cost alternative to the same.

For applications involving robotic arms where in a person's hand movements are needed to be mimicked, the glove can prove as a good sensory mechanism.

APPENDIX A

Codes

A.1 Microcontroller Code

The microcontroller receives time multiplexed data, demultiplexes it, packages it in the form of an array and forwards it to the bluetooth module using the serial port. Following is the code snippet used to implement this functionality.

The read function demultiplexes the input data.

```
int read(void) {  int i=0,fac=0,data=0,temp;
P1OUT=P1OUT|BIT5;
__delay_cycles(100);
P1OUT=P1OUT^BIT5;
__delay_cycles(100);
if (P1IN&BIT6)
{
   data+=128;
}
i=0;
  while(1)
    {
   P1OUT=P1OUT|BIT4; //P1.0 is clock also connected to red LED
         __delay_cycles(100);
          P1OUT=P1OUT&(~BIT4);
          __delay_cycles(100);
             if(i==0)      fac=64;
             else if(i==1) fac=32;
         else if(i==2) fac=16;
             else if(i==3) fac=8;
             else if(i==4) fac=4;
             else if(i==5) fac=2;
             else if(i==6) fac=1;
if (P1IN&BIT6)   //P1.7 takes input from ADC
data+=fac;
if(i==7){
return(data);}
    i++;
}}
```

The address function calculates the address to be sent to the multiplexer.

```
void address(int a)
{  if(a==0) P1OUT=P1OUT&(~(BIT0+BIT2+BIT3+BIT7));
else if(a==1)
P1OUT=(P1OUT&(~(BIT0+BIT2+BIT3+BIT7)))+BIT0;
else if(a==2)
P1OUT=(P1OUT&(~(BIT0+BIT2+BIT3+BIT7)))+BIT2;
else if(a==3)
P1OUT=(P1OUT&(~(BIT0+BIT2+BIT3+BIT7)))+BIT0+BIT2;
else if(a==4)
P1OUT=(P1OUT&(~(BIT0+BIT2+BIT3+BIT7)))+BIT3;
else if(a==5)
P1OUT=(P1OUT&(~(BIT0+BIT2+BIT3+BIT7)))+BIT0+BIT3;
else if(a==6)
P1OUT=(P1OUT&(~(BIT0+BIT2+BIT3+BIT7)))+BIT2+BIT3;
else if(a==7)
P1OUT=(P1OUT&(~(BIT0+BIT2+BIT3+BIT7)))+BIT0+BIT2+BIT3;
else if(a==8)
P1OUT=(P1OUT&(~(BIT0+BIT2+BIT3+BIT7)))+BIT7;
else if(a==9)
P1OUT=(P1OUT&(~(BIT0+BIT2+BIT3+BIT7)))+BIT0+BIT7;
    __delay_cycles(100);
}
```

Next is the main function.

```
int main(void)
{ int a,b,c,j=0,temp;
    if (CALBC1_1MHZ==0xFF)
{       while(1);// do not load, trap CPU!!     }
WDTCTL = WDTPW + WDTHOLD;
    DCOCTL = 0;
    BCSCTL1 = CALBC1_1MHZ;
    DCOCTL = CALDCO_1MHZ;
    P1OUT = 0x00; // Initialize all GPIO
    P1SEL = UART_TXD;
    P1DIR=BIT0+BIT2+BIT3+BIT4+BIT5+BIT7+BIT1;
    P2OUT = 0x00;
    P2SEL = 0x00;
    P2DIR = 0xFF;
    __enable_interrupt();
    TimerA_UART_init();
    TimerA_UART_print("G2xx1 TimerA UART\r\n");
    TimerA_UART_print("READY.\r\n");
    int ans[10]=0,i=0,x=0,y=0,sum[10]=0;
    P1OUT=BIT5;
    __delay_cycles(100);
    P1OUT=0;
    for (;;)     {
     //TimerA_UART_tx('U');
     address(i);
     ans[i]=read();
     i++;
     if(i==10)
     { i=0;
   TimerA_UART_tx((char)0);
TimerA_UART_tx((char)ans[0]-1);
TimerA_UART_tx((char)ans[1]-1);
TimerA_UART_tx((char)ans[2]-1);
TimerA_UART_tx((char)ans[3]-1);
TimerA_UART_tx((char)ans[4]-1);
TimerA_UART_tx((char)ans[5]-1);
TimerA_UART_tx((char)ans[6]-1);
TimerA_UART_tx((char)ans[7]-1);
TimerA_UART_tx((char)ans[8]-1);
TimerA_UART_tx((char)ans[9]-1);
}}}
```

A.2 MATLAB Codes

MATLAB was used for processing the sensor data and guessing the gesture. There are two codes in this section, each implementing a different algorithm. The .mat files used in the codes store the sensor values for various gestures.

A.2.1 Mean Square Error Algorithm

```
% initialization code
% clear all;
% global s1;
% s1=serial('COM11','BAUDRATE',115200);
fopen(s1);
while(1)
    reply = input('tRain/Test/eXit/Pause [R/T/X/P]: ', 's');
    if isempty(reply)
        reply = 'T';
        end;
       if reply=='R'
        trainLetter=input('Which letter?: ', 's');
        if isempty(reply)
            continue;
        else
```

```
            trainer(trainLetter)
        end
    elseif reply==’T’
        guess=testletter();
        disp(guess);
        %fwrite(s2,guess);
    elseif reply==’P’
        disp(’resume to continue’);
        pause;
    elseif reply==’X’
        fclose(s1);
        disp(’Exiting’);
        break;
    end;
end;
function trainer(Letter)
    disp(’Training... take your position!’);
    pause(1);
    sum=readbt();
    disp(’Training done!’);
    load(’data.mat’);
    load(’letters.mat’);
    data=[data, sum];
    letter=[letter, Letter];
    save(’letters.mat’,’letter’);
    save(’data.mat’,’data’);
end
function letterGuess=testletter()
    load(’data.mat’);
    load(’letters.mat’);
    sum=readbt();
    ssq=zeros(1,length(letter));
    for i=1:length(letter)
```

```
            ssq(i)=sumsqr(data(:,i)-sum);
        end
        index=find(ssq==min(ssq));
        letterGuess=letter(index);
    end
    function sum=readbt()
        global s1;
        i=1;
        N=25;
        sum=0;
    fopen(s1);
        while(1)
            A=fread(s1,1);
            if A==0
                b=fread(s1,10);
                b(find(b==0))=256;
                i=i+1;
                A=0;
                sum=sum+b;
            end
            if i==N+1
                break;
            end
        end;
        sum=sum/N;
    end
```

A.2.2 ASL Algorithm

For this algorithm, only the test function varies from the previous code. The new test fuction is given below.

```
    function data=test()
        load('openData.mat');
```

```
load('closeData.mat');
data=read();
dataCopy=data;
data1=(data-openData).^2;
data2=(data-closeData).^2;
for i=1:length(data)
    if(data1(i)>data2(i)) data(i)=0;
    else data(i)=1;
    end
end
guess='NotDefined';
nonBends=data(4)+data(6)+data(8)+data(10);
if nonBends==0
    disp('FirstGroup');
    if(data(9)==0&data(7)==1&data(3)==1)
        guess='M';
    elseif(data(9)==0&data(5)==1)
        guess='N';
    elseif(data(9)==0&data(3)==1)
        guess='X';
    elseif(data(3)==1&data(5)==1&data(7)==1&data(9)==1)
        if(data(2)==1)
            guess='C';
        elseif(data(1)==1)
            guess='O';
        else guess='E';
        end
    else
        if(data(2)==1)
            guess='A';
        elseif(data(1)==1)
            guess='T';
        else guess='S';
```

```
              end
         end
               elseif nonBends==1
          disp('SecondGroup');
          if(data(4)==1)
              if((data(5)==1)&(data(7)==1)&(data(9)==1))
                  guess='D';
              elseif(data(3)==1)
                  guess='L';
              elseif(dataCopy(6)<100)
                  guess='Z';
              else guess='GQ';
              end
         elseif(data(10)==1)
              if(data(2)==1)
                  guess='Y';
              else guess='IJ';
              end
          else guess='?';
          end
guess
end
```

REFERENCES

[1] D. J. Sturman and D. Zeltzer, "A survey of glove-based input," *IEEE Computer Graphics and Applications* vol. 14, pp. 30–39, 1994.

[2] A. Mulder. (1994). *How to build an instrumented glove based on the Powerglove flex sensors*. [Online]. Available: http://xspasm.com/x/sfu/vmi/PCVR.html

[3] R. Mitchell, T. Young, B. Bachleda, M. Karchmer, "How Many People Use ASL in the United States?: Why Estimates Need Updating," *Sign Language Studies*, Gallaudet University Press, 2006.

[4] D. Armstrong, M. Karchmer, "William C. Stokoe and the Study of Signed Languages," in *The Study of Signed Languages*, Gallaudet University, pp. xi–xix, 2002.

[5] *The Velostat sensor* (n.d.) [Online]. Available: http://www.pulsar.org/archive/int/timswork/Velostat.html

[6] *How to Make Bi-Directional Flx Sensors* (2009, July 25) [Online]. Available: http://www.instructables.com/id/How-to-Make-Bi-Directional-Flex-Sensors

[7] *Hand schematic* (2010, March 23) [Online]. Available: http://www.idrawdigital.com/wp-content/uploads/2010/03/hand01.jpg

[8] *Sign Language* (2007) [Online]. Available: http://lifeprint.com/asl101/fingerspelling/images/signlanguageabc02.jpg

[9] *Single-Ended 16-Channel/Differential 8-Channel CMOS Analog Multiplexers* (2003) [Online]. Available: http://www.ti.com/lit/ds/symlink/mpc506.pdf

[10] *8-Bit Analog-to-Digital Converters with Serial Control (1996)* [Online]. Available: http://www.ti.com/lit/ds/symlink/tlc549.pdf

[11] *Mixed Signal Microcontroller* (2010) [Online]. Available: http://www.ti.com/lit/ds/symlink/msp430g2231.pdf

[12] *EZ430-RF256x* (2013, April 18) [Online]. Available: http://processors.wiki.ti.com/index.php/EZ430-RF256x

[13] *Plastic and Glass Electromagnetic Field Shielding Material* (n.d.) [Online]. Available: http://www.lessemf.com/plastic.html

[14] T. Shiratori and J. K. Hodgins, "Accelerometer-based user interfaces for the control of a physically simulated character," *ACM Transactions on Graphics*, vol. 27, no. 5, pp. 1–9, 2008.