Table of Contents (Soft Version) :

Sudan University of Science and Technology

College of Engineering

Electronics Engineering

Development of
Smart Glove: Gesture Translator

A Research Submitted in Partial Fulfillment for the Requirements of the
The degree of B.Sc. (Honors) in Electronics Engineering

Prepared By:

1. Abdulrahman Babiker Fadlulmaula Hussain**.**
2. Basil Abul-Qasim Ahmad Mohammad.
3. Moez Babiker Hassan Mohammad-Zain.
4. Widaa'atullah Siddiq Abbas Fadlulmaula.

Supervised By:

Dr. Abu-Agla Babiker Mohammad

November 2017

الاسْتِهْلال

بِسْمِ اللهِ الرَّحْمَٰنِ الرَّحِيمِ

﴿ قَالَ رَبِّ ٱشْرَحْ لِى صَدْرِى ٢٥ وَيَسِّرْ لِىٓ أَمْرِى ٢٦ وَٱحْلُلْ عُقْدَةً مِّن لِّسَانِى ٢٧ يَفْقَهُواْ قَوْلِى ٢٨ ﴾

سورة طه (25 , 26 , 27 , 28)

DEDICATION

Dedicated, in thankful appreciation for support, encouragement and understandings to our beloved mothers, fathers, brothers, sisters, teachers and our friends.

ACKNOWLEDGEMENT

First of all, our gratitude's goes for Allah for blessing us with the patience, the strength and the ability to complete this study.

We thank all our mothers and fathers for the trust and the support that have been guiding our way. from the very start, till this finishing line.

We also thank our families and friends, for pushing us beyond the limits we have set for ourselves, and for changing every "we cannot"

To "we can".

Our sincere gratitude goes for our supervisor Dr. Abu-Agla for his sufficient advices, understanding, and reassuring.

We truly thank our friend Ahmed Mohammad Ali; this work would have never seen light without your consideration.

ABSTRACT

Generally, people with hearing impaired and speech disability use sign language based on hand gestures with specific motion to represent the “language” they are communicating. A gesture in a sign language is a particular movement of the hands with a specific shape from the fingers and hole hand. This project aims to convert the hand gestures based on electronic device that translate sign language into speech to make the communication take place between the mute communities with the general public. In this research, a practical implementation of a system which enables mute people to use gloves so as to communicate with normal people via their smart phone has been designed and implemented practically. Mute people can use the glove to perform hand gesture and it will be converted into speech to allow normal people to understand their expression. An Android Application designed will be used that will convert the gestures into text depending on hand shape detected and produce a voice. The developed project will potentially reduce the communication gap between mute and normal people.

المستخلص

بصفة عامة يستخدم الأشخاص المعاقين كلامياً لغة الإشارة استناداً إلى إيماءات اليد مع حركات معينة لتمثل اللغة التي يتواصلون بها. الإيماءة في لغة الإشارة عبارة عن حركة محددة عبر الأيدي، مع شكل محدد مصنوع عبر الأصابع. يهدف هذا المشروع إلى تحويل الإيماءات اليدوية عبر أجهزة إلكترونية إلى حديث من أجل جعل التواصل يجري بين ذوي الاحتياجات الخاصة مع عامة الناس. في هذا البحث، تم تطوير نظام عملي يسمح للأشخاص البكم باستخدام قفاز للحديث مع الناس العاديين عبر هواتفهم الذكية وتم تصميمه وتجربته عمليا. يمكن للبكم استخدام القفاز لعمل إيماءات اليد والتي سيتم تحويلها إلى حديث وحروف للسماح للأشخاص العاديين بفهم تلك التعابير. تم تصميم تطبيق يعمل على نظام آندرويد لاستخدامه لتحويل الإيماءات إلى نصوص حسب شكل اليد المكتشف ويقوم بعدها بإخراج الصوت. من شأن هذا المشروع تقليل فجوة الاتصال بين الأشخاص البكم والأشخاص العاديين.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AC	-	Alternating Current
ADC	-	Analog to Digital Converter
ASL	-	American Sign Language
CMOS	-	Complementary Metal Oxide Semiconductor
CPU	-	Central Processing Unit
DC	-	Direct Current
DOF	-	Depth Of Field
GPIO	-	General Purpose Input Output
HDMI	-	High Definition Multimedia Interface
ICSP	-	In Circuit Serial Programming
IDE	-	Integrated Development Environment
IR	-	Infrared
LCD	-	liquid Crystal Display
MEMS	-	Microelectromechanical Systems
NTSC	-	National Transportation safety committee
OS	-	Operating System
PAL	-	Phase Alternating Line
PCB	-	Printed Circuit Board
PSE	-	Pidgin Sign English
PWM	-	Pulse Width Modulation
RX	-	Receiver
SEE	-	Signed Exact English

SOC	-	System On a Chip
TCP/IP	-	Transmission Control Protocol / Internet Protocol
TX	-	Transmitter
UI	-	User Interface
USB	-	Universal Serial Bus
VT	-	Virtual Terminal
XML	-	Extensible Markup Language

CHAPTER 1
INTRODUCTION

1.1 Preface

"Sign language can be defined as combination of hand shape, body and arms movements and facial expressions without a voice. Sign language is main communication medium for the deaf and dumb people to communicate with each other. Normally, normal people do not understand the sign language used by deaf people. Therefore, the study of sign language is needed to improve the communication barrier between the verbal and non-verbal community." (signsoflifeasl, February 27, 2013)

1.2 Problem Statement

Sign language is the only communication tool used by deaf people to communicate to each other. However, normal people do not understand sign language, and this will create a large communication barrier between deaf people and normal people.

In addition, the sign language is also not easy to learn due to its natural differences in sentence structure and grammar. Therefore, there is a need to develop a system which can help in translating the sign language into text and voice to ensure the effective communication can be easily take place in this community.

1.3 Objectives

The aim of Smart Glove: Gesture Translator was to design a user-friendly glove that would translate sign language gestures into speech and text with high level of accuracy for recognizing gestures. The specific objectives are:

- To increase the efficiency of the past projects by increasing the letters and words which got out by the glove.

- To minimize the glove size so that it become easier to handle.

1.4 State of the Art

This section will explain briefly about some common methods of gesture recognizing and the comparison between these methods. Details of these approaches are added to chapter 2 of this thesis.

1.4.1 Data Glove Approach

The data-glove approach uses a unique assembled electronic glove, which has sensors that give us the hand shape. Most commercial sign language translation systems use the data-glove method, as it simple to acquire data on the bending of finger and 3D orientation of the hand using gloves (Ahmad Zaki Shukor, 2015).

1.4.2 Visual-Based Approach

With the current advancement in computer technology and software, there has been an increase in the use of visual-based methodology. Images of the signer (the one uses the sign language) are captured by a camera and video processing is done to perform detection of the sign language. Different from the data glove approach, the fundamental advantage of visual-based methodology is the adaptability of this approach. The recognition of facial expression and head movements additionally can be incorporated to the framework and also perform lip-perusing (Ahmad Zaki Shukor, 2015).

1.4.3 Virtual-based Button Approach

Function of a virtual button is to generate button events, a press and discharge, by perceiving hand motions of holding and discharging individually. This virtual button also can identify different sorts of gesture and generate proper command. The virtual button method uses patterns of the

wrist shape. It can perceive a squeeze movement. By using small sized IR optic sensors, the patterns of finger flexor muscles on wrist can be recognized by moving fingers. These patterns are used to perceive finger or hand movement. The IR emitter and IR optic sensor are attached on the bottom of the wrist because the area comprises of finger flexor tendons that delicately respond to finger movements. These sensors produce voltage values according to the amount of IR radiation. In the system, this sensor is used to monitor different patterns of the wrist shape resulting from the movement of finger flexor tendons in the wrist when fingers are moving (Ahmad Zaki Shukor, 2015).

The following table summarizes previously approaches.

Table 1-1 : Comparison of different approaches for sign language detection

Method	**Device/ components**	**Gestures**	**Environment**	**Accuracy**
Data-glove approach	Flex sensor, Accelerometer, Microcontroller.	Fingerspelling (alphabets and numbers), sign gestures.	Practical, no environmental concern.	Higher accuracy if use more sensors.
Visual-based approach	Custom-made glove, computer's webcam.	Effective for fingerspelling only (alphabets and numbers)	position of the camera, background condition and lighting sensitivity	can be misinterpreted
Virtual button	IR optic sensor	Finger and hand movement, not suitable for gestures	no environmental concern.	Overall correctness 88.82%

1.5 Proposed Solution:

In this project we propose a Sign Language Glove which will assist those people who are suffering for any kind of speech defect to communicate through gestures i.e. with the help of single handed sign language the user will make gestures of alphabets. The glove will record all the gestures made

by the user and then it will translate these gestures into visual form as well as in audio form.

1.6 Methodology

In this project, gloves are implemented to capture the hand gesture made by disabled person and converting it into speech as well as text. A pair of gloves with flex sensors along each finger, thumb and arm, contact sensors and accelerometer are used to capture the movement of user. With the help of flex sensors degree of fingers thumb and arm are calculated in voltage terms using voltage divider rule enters the comparator circuit to change it from analog to digital. Contact sensors are used to recognize between some letters which have similar finger shape and differ between another letter with little finger bending degree, and the accelerometer is to differ between letters with same hand shape but differ at gesture of the hand. Arduino is used for the processing of the entered data and decides the outputted word or letter. The letter or word that the Arduino produced get transferred to the phone that the other person (listener) is using. The phone has an android application designed by the team that present our outputted word or letter in the shape of voice and text.

Either for the design process we took to come with this design first we studied the ASL (American sign language) and then we decided the sensors that will give us all the letters and most of the words of this language with respect to the cost ,then we chose the process unit which is the Arduino UNO and then we decided the tool that we’re going to use to transfer the signal from the transmitter part to the receiver part and we chose the Wi-Fi to connect between the Arduino and the phone .

We build our project by using a transmitter and receiver parts to decrease the weight of the glove on the hand and to make it easily for the hand to move. But in return the cost will increase because we use two processors than one.

1.7 Expected Results

The expected results of the project are that it will express all the sign language letters and some of the words in both text form at the android phone and voice form at the speaker. Next Figure represents all sign language gestures.

Figure 1-1 Sign Language Letters

1.8 Thesis Outline

The thesis consists of five main chapters. The first chapter, Chapter One explains on the background of the project, problem statement, objectives, scopes, and summary of the work. Chapter Two comprises of

detail information on the theory and literature review related to the project. In Chapter Three explains about method used in construction and designing the project. The process includes are circuit design and construct and development of software and algorithm. In Chapter Four discuss about the results of the project. The data collected from the project are analyzed for discussion. Lastly, Chapter Five covers the conclusions, problems, recommendations and further improvement for the project.

CHAPTER 2

LITERATURE REVIEW & RELATED WORKS

2.1 Basic concepts of the projects

In this part we will discuss the main design component and the way to connection between them.

2.1.1 Sign language background

“There are three major forms of Sign Language currently used in the United States: American Sign (ASL), Pidgin Signed English (PSE), and Signed Exact English (SEE). ASL is used by many deaf in the United States, thus its use promotes assimilation into the Deaf Community. ASL is a visual language, and speech-reading or listening skills are not needed to learn ASL fluently.” (signsoflifeasl, February 27, 2013).

2.1.2 Gesture Recognition Methods

Referring to chapter 1, Nowadays, automatic sign language translation systems generally use two different sensor-based approaches, which are data glove and visual-based approaches. However, a new hand gesture recognition method has been introduced, called virtual button. This section will explain the detail these approaches. (Ahmad Zaki Shukor, 2015)

2.1.2.1 Data Glove Approach

The data-glove approach uses a unique assembled electronic glove, which has sensors that give us the hand shape. Most commercial sign language translation systems use the data-glove method, as it simple to acquire data on the bending of finger and 3D orientation of the hand using gloves. This approach requires less computational force and continuous analysis is much simpler to achieve. Some of the sensors that we use in this approach is the flex sensors and contact sensors which give us the finger shape and the accelerometer for hand gesture. The data glove is highly

suitable in perceiving both fingerspelling and sign motions, which include static and movement signs. However, these data glove can be costly. While it is possible to create less expensive data glove, they are much more vulnerable to noise. If the number of the sensors used is reduced, it will bring about loss of important data about the hand shape. This will result in the loss of accuracy in sign language interpretation. The data glove could also be less comfortable to be worn by the signer (Ahmad Zaki Shukor, 2015).

2.1.2.2 Visual-Based Approach

With the current advancement in computer technology and software, there has been an increase in the use of visual-based methodology. Images of the signer (the one uses the sign language) are captured by a camera and video processing is done to perform detection of the sign language. Different from the data glove approach, the fundamental advantage of visual-based methodology is the adaptability of this approach. The recognition of facial expression and head movements additionally can be incorporated to the framework and perform lip-perusing. This system can be separated into two strategies, which are utilization hand crafted shading gloves and in light of skin-color recognition. For the recognition based on skin-color, the approach hard requires just a camera to capture the moving pictures of the signer with no additional gadgets needed. It turns out to be more common and helpful for constant applications. This system utilizes an uncovered hand to concentrate information required for recognition, and it is simple, where the user directly communicates with the system. In order to track the position of hand, the skin color region will be fragmented using color threshold technique, and then the region of interest can be determined. The image acquisition runs constantly until the signer demonstrates a stop sign. After the threshold, the segmented

images are then analyzed to obtain the unique features of each sign. These visual-based approaches are minimizing the equipment cost. However, these systems are just suitable and viable for decoding alphabets and numbers, as opposed to perceiving sign gestures. Signs with comparable stance to another sign can be confused reducing the precision of the system. Moreover, the image acquisition process is subject to numerous natural concerns, for example, position of the camera, background condition and lighting effects. The different height of the signer must also be considered. Adequate lighting is additionally required to have enough brightness to be seen and analyzed (Ahmad Zaki Shukor, 2015).

2.1.2.3 Virtual-based Button Approach

Function of a virtual button is to generate button events, a press and discharge, by perceiving hand motions of holding and discharging individually. This virtual button also can identify different sorts of gesture and generate proper command. The virtual button method uses patterns of the wrist shape. It can perceive a squeeze movement. By using small sized IR optic sensors, the patterns of finger flexor muscles on wrist can be recognized by moving fingers. These patterns are used to perceive finger or hand movement. The IR emitter and IR optic sensor are attached on the bottom of the wrist because the area comprises of finger flexor tendons that delicately respond to finger movements. These sensors produce voltage values according to the amount of IR radiation. In the system, this sensor is used to monitor different patterns of the wrist shape resulting from the movement of finger flexor tendons in the wrist when fingers are moving. However, this method is not effective for interpreting sign language because it requires more intricate stance of fingers. Also, sign language gestures utilize more

than a finger movement and wrist shape. Thus, this method is not practical for perceiving communication via gestures motions (Ahmad Zaki Shukor, 2015).

2.2 Communication between Normal People and Deaf People

"Research projects have aimed to develop technology to facilitate communication between hearing and non-hearing people. In 1999, the first attempt to solve the problem described a simple glove system that can learn to decipher sign language gestures and output the associated words. Since then, numerous teams, mainly undergraduate or graduate university projects, have created their own prototypes, each with its own advantages and design decisions." (Princesa Cloutier 2016).

2.2.1 Related Work

Using the concept of gestures, few attempts have been made in the past to recognize the gestures made using hands but with limitations of recognition rate and time which include:

1. Using CMOS camera.

2. Leaf switches based glove.

3. Copper plate based glove.

4. Flex sensor based glove (Kanika Rastogi May 2016) .

2.2.1.1 Using CMOS Camera

CMOS camera transmits image data via UART serial port. The UART performs serial-to-parallel conversions on data received from a peripheral device (CMOS camera in this case) and parallel-to-serial conversion on data received from the CPU (Microcontroller in this case).

Hand gestures were detected using CMOS camera by 3 steps:

1. Capturing the image of the gesture
2. Edge detection of that image
3. Peak detection of that image

Disadvantage: Highly expensive, latency and each image occupies 50KB of memories (Kanika Rastogi May 2016).

2.2.1.2 Leaf switches based glove

These are like normal switches, but these are designed in such a way that when pressure is applied on the switch, the two ends come into contact and the switch will be closed. These leaf switches are placed on the fingers of the glove such that the two terminals of the switch come into contact when the finger is bent.

Disadvantage: After prolonged usage, the switch instead of being open when the finger is straight, it will be closed resulting in improper transmission of gesture (Kanika Rastogi May 2016).

2.2.1.3 Copper Plate Based Glove

In this prototype, a copper plate is fixed on the palm as ground. The copper strips indicate a voltage level of logic 1 in rest position. But when copper strips meet the ground plate, the voltage associated with them is drained and they indicate a voltage level of logic 0.

Disadvantage: The use of copper plate makes the glove bulky which makes it unsuitable to use it for a long time (Kanika Rastogi May 2016).

2.2.1.4 Flex sensor based glove

Flex means “bend” or “curve”. Sensor refers to a transducer which converts physical energy into electrical energy. Flex Sensor is a resistive

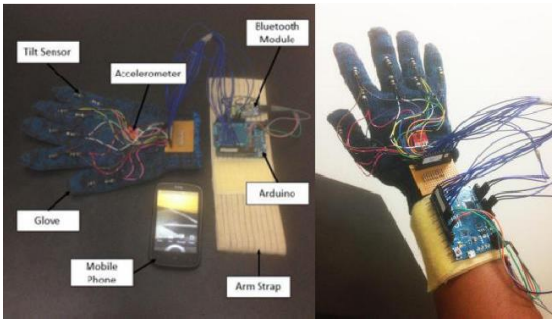
sensor which changes its resistance as per the change in bend or curvature of it into analog voltage. This is a haptic based approach which consists of using flex sensors to take in physical values for processing (Kanika Rastogi May 2016).

2.2.2 Some Models of Sign Language Translator

Mohammed *et al*. (2015) accomplished a model uses a glove fitted with sensors that can interpret the (ASL) alphabet. The glove uses flex sensors, contact sensors and 6 DOF accelerometer/Gyroscope on a single chip. All these sensors are mounted on the hand to gather the data on every finger's position and the orientation of the hand to differentiate the letters. Sensory data is sent to a computing unit (an Arduino microcontroller) to be translated and then displayed. The Accelerometer/Gyroscope are used for the movement and orientation detection and it is mounted at the upper side of the hand. The contact or force sensors determine which fingers are touching and how the fingers are relative to each other. To enable wireless transmission between the controller and a smart phone, a Bluetooth module is chosen. The selection of Bluetooth module is based on its capability of being easy to use, and its wide compatibility with today's gadgets. A serial Bluetooth module device is utilized in this work. In some cases, similarity between measurements pertained to different letters is very high. This misleading similarity causes wrong detection as seen in the table above, we call it ambiguity. This issue is known as identification problem.

Ahmad Zaki Shukor *et al.* (2015) developed a model (Figure 2-1) that described it as "This is the first application of the tilt sensor in sign language detection, to the best of the authors' knowledge", They specified it to translate Malaysian Sign Language. The microcontroller used is Arduino,

which interprets the information received from the sensors (tilt and accelerometer) to be sent to the Bluetooth module, and then the module sends the interpreted sign/gesture to the Bluetooth-enabled mobile phone. They said that constructing the data glove with flex sensors can be quite costly because the price for flex sensor is quite expensive. Also, the reading of flex sensor is not very stable and sensitive to noise. the accuracy for all the tests is quite high. In the number of 10 attempts tested for each sign, at least 7 trials were successful. The alphabet/number has higher accuracy due to the usage of tilt sensors. The words have lower accuracy because the words involve motion which needs to be detected by the accelerometer.



(a) (b)

Figure 2-1 Ahmad Zaki Shukor's Model – (a) Overall System Components and (b) the device worn on user's hand

Mohammad Taye *et al.* (2016) constructed a model named it as : ANY ONE CAN TALK TOOL. Based on this project they decided to divide it into three Layers, which are: The application Layer, The Software System Layer and The Hardware Layer. The first layer is Application Layer, which contained two applications: Signal Generator Application, which responsible on generating signal and sending it, and Interpreter Application which responsible on interpreting signals and generating voice. Second layer is

System Software Layer, Their project contains two-system software, first system software is RASPBIAN, which is a free operating system based on Debian optimized for the Raspberry Pi hardware. Second one is ARDUINO, it is some open-source electronics prototyping platform based on flexible easy to use hardware and software. Third layer is concerned about Hardware . This layer consists three main components, Main components are Gloves, each glove includes five flex sensors each sensor placed on glove finger, and microprocessor connected with these flex sensors. Hardware Used in their project are: Arduino fio v3, Flex Sensor 2.2, Raspberry Pi Model B as Interpreter, Video Out via Composite (PAL and NTSC), HDMI or Raw LCD, Audio Out via 3.5mm Jack or Audio over HDMI. Interpreter Can Be Connected to gloves in Two ways: Wireless or Serial communication.

K.V.Fale *et al.* (2016) Proposed a system named : Gesture Vocalizer for Deaf and Dumb People. In this system at the transmitter side they used a glove which must be worn by the user. This glove is mounted with 4 flex sensors each on the 4 fingers of the glove namely thumb, index, middle, ring. The flex sensors give their output in the form of change in resistance according to the bend angle. The output from the flex sensors is given to the ADC channels of the microcontroller. The processed ADC values from the microcontroller are compared with the threshold values for the recognition of a particular gesture. The particular gesture is recognized and is given to the microcontroller which transmits them through the RF module in a serial manner (as in Figure 2-2) . For each value received at RF receiver, the microcontroller gives corresponding commands to the LCD and the Voice

Module. Thus, they acquired the voice output for each gesture and displayed of each gesture in form of text on the LCD display.

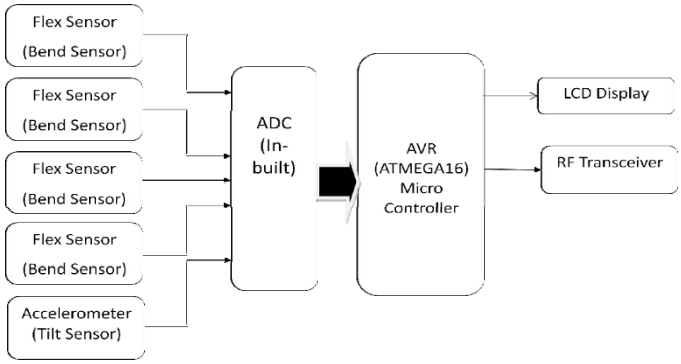


Figure 2-2 Gesture Vocalizer for Deaf and Dumb People - Block Diagram of Transmitter Section

2.3 Summary

In this project, ASL is chosen because it is widely used by deaf people around the world. Also, its gestures are like normal people motions and expressions.

After deep reading literature and related work, most models preferred data glove approach. It seems the reasons of that is cheapness of its components, less efforts in implementing it and its accuracy is relatively high (with some insufficiency that can be overcomes by some additional work). Most previous models discussed in this chapter used flex sensors as finger gesture recognition, Flex sensor is the most suitable sensor to measure and capture the movement of the fingers, accelerometer as hand motion recognition, ARDUINO as a microcontroller that interprets the gesture of the hand, and a smart phone with android application to outputs results.

Next chapter will discuss our proposed solution, its components and the methodology of constructing this model.

CHAPTER 3

RESEARCH METHODOLOGY

This chapter explains the development methodology and guidelines in designing and construction the project. There are two major parts developing the project: The circuit design and construct and the develop of software and algorithm.

3.1 Research Phases

It was apparent that the completion of this project would require meeting several Phases as described below:

1. Develop a sensing network that incorporates and differentiates the different types of hand movement used in American Sign Language (ASL):
   - Flexion.
   - Contact.
   - Rotation.
   - Translation.
2. Interface the sensor network with an Arduino.
3. Develop a program that recognizes sensor signals and stores those that correspond to a designated ASL sign into a code library.
4. Successfully output ASL signs in a manner that can be understood by a non ASL speaker.

It should be noted that the phases can be categorized into two main categories, hardware and software. Phases 1 and 3 are hardware objectives, as they relate to the physical, tangible parts of the glove, while Phase 3 is a software objective and is instead focused on the glove's code development and signal processing. Phase 2 relates to both the hardware and software components, can be considered embedded systems objectives.

3.2 Overview about System components

In this section, all components used in this research will be clarified, as shown in the next Figure.

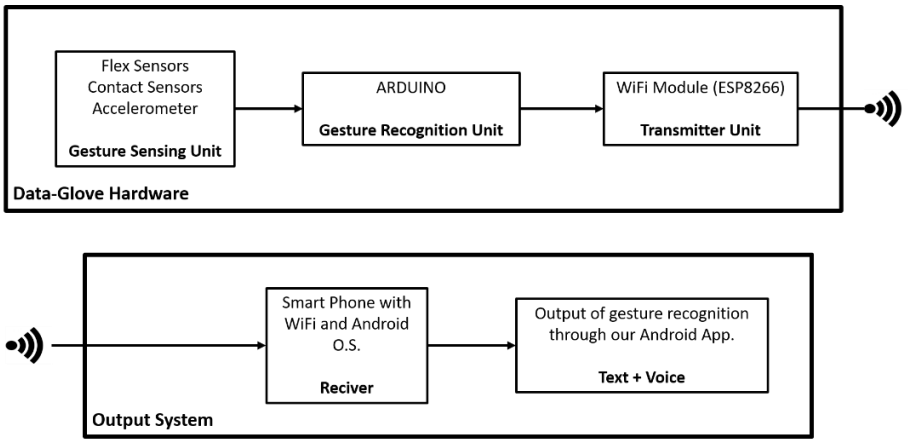


Figure 3-1 Project Block Diagram

The implementation of this project in real life is shown in the next Figure.

Figure 3-2 The Project in Real Life Action

3.2.1 Flex Sensors as fingers Gesture Recognition

A flex sensor (Figure 3-3) is a device that has very versatile compatibilities. When the sensor is flexed, its resistance increases significantly. Using a voltage divider circuit, comprising of the flex sensor

and a carefully chosen fixed resistor, we can know when the sensor is being flexed and when it is not by connecting it with the comparator circuit. Flex sensor is the most suitable sensor to measure and capture the movement of the fingers. When sensor placed in gloves is bent, it produces a resistance output correlated to the bend radius- the smaller the radius, the higher the resistance value. The bending resistance range for flex sensor varies approximately from 30Kohm to 125Kohms. The flex sensors are the most critical sensors because most letters can be distinguished based on fingers' flexes. (Eelctronics)

Figure 3-3 2.2 inch Flex Sensor

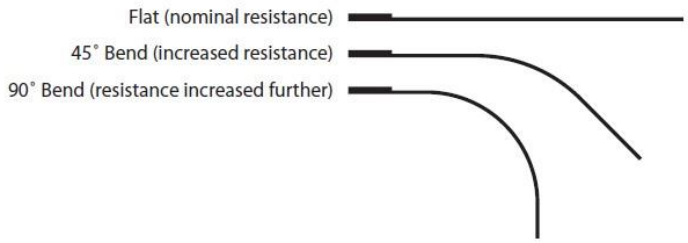


Figure 3-4 Flex Bend Levels

3.2.2 Contact Sensors or tactile sensors as fingers touching Recognition

Contact sensor uses transducer for the sensing operation. Some mostly used sensors are potentiometer, strain gauge etc. Contact or touch sensors are

one of the most common sensors in robotics. These are generally used to detect a change in position, velocity, acceleration, force, or torque at the manipulator joints and the end-effecter. There are two main types, bumper and tactile. Bumper type detect whether they are touching anything, the information is either Yes or No. They cannot give information about how hard is the contact or what they are touching. Tactile sensor are more complex and provide information on how hard the sensor is touched, or what is the direction and rate of relative movement.

3.2.2.1 Tactile sensors:

That measure the touch pressure rely on strain gauges or pressure sensitive resistances. Variations of the pressure sensitive resistor principle include carbon fibers, conductive rubber, piezoelectric crystals, and piezo-diodes. These resistances can operate in two different modes: The material itself may conduct better when placed under pressure, or the pressure may increase some area of electrical contact with the material, allowing increased current flow. Pressure sensitive resistors are usually connected in series with fixed resistances across a DC voltage supply to form a voltage divider. The fixed resistor limits the current through the circuit should the variable resistance become very small. The voltage across the pressure variable resistor is the output of the sensor and is proportional to the pressure on the resistor. The relationship is usually non-linear, except for the piezo-diode, which has a linear output over a range of pressures. An analog to digital converter is necessary to interface these sensors with a computer. (Shailendra Kumar Bohidar, 2015)

3.2.3 MPU-6050 3 Axis Gyroscope and Accelerometer Module as a Hand Motion Recognition

The InvenSense MPU-6050 sensor (Figure 3-5) contains a MEMS accelerometer and a MEMS gyro in a single chip. It is very accurate, as it contains 16-bits analog to digital conversion hardware for each channel. Therefor it captures the x, y, and z channel at the same time. The sensor uses the I2C-bus to interface with the Arduino. The MPU-6050 is not expensive, especially given the fact that it combines both an accelerometer and a gyro. (ARDUINOPlayground)

Figure 3-5 InvenSense MPU-6050 sensor

3.2.4 ARDUINO UNO as A Gesture Recognition Unit

ARDUINO is an open-source computer hardware and software company, project and user community that designs and manufactures microcontroller-based kits for building digital devices and interactive objects that can sense and control the physical world.

The project is based on a family of microcontroller board designs manufactured primarily by Smart Projects in Italy, and also by several other vendors, using various 8-bit Atmel AVR microcontrollers or 32-bit Atmel ARM processors. These systems provide sets of digital and analog I/O pins

that can be interfaced to various expansion boards ("shields") and other circuits. The boards feature serial communications interfaces, including USB on some models, for loading programs from personal computers. For programming the microcontrollers, the Arduino platform provides an integrated development environment (IDE) based on the Processing project, which includes support for C, C++ and Java programming languages. (ARDUINOWebsite)

Arduino Uno (Figure 3-6) is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again. "Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index of boards. (ARDUINOWebsite)

The Arduino IDE (Integrated Development Environment) is a cross platform application written in java, and is derived from the IDE for processing programming language and writing this prototype. It includes a code editor with features such as syntax highlighting, brace matching and

automation indentation and is also capable of compiling and uploading programs to the board with a single click. (Kanika Rastogi May 2016)

Figure 3-6 ARDUINO UNO

3.2.5 WiFi Module (ESP8266) as a transmitter

The ESP8266 WiFi Module (Figure 3-7) is a self-contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to your WiFi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor. Each ESP8266 module comes pre-programmed with an AT command set firmware, meaning, you can simply hook this up to your Arduino device and get about as much WiFi-ability as a WiFi Shield offers (and that's just out of the box)! The ESP8266 module is an extremely cost-effective board with a huge, and ever growing, community. This module has a powerful enough on-board processing and storage capability that allows it to be integrated with the sensors and other application specific devices through its GPIOs with minimal development up-front and minimal loading during runtime. Its high degree of on-chip integration allows for minimal

external circuitry, including the front-end module, is designed to occupy minimal PCB area. (SparkFun)

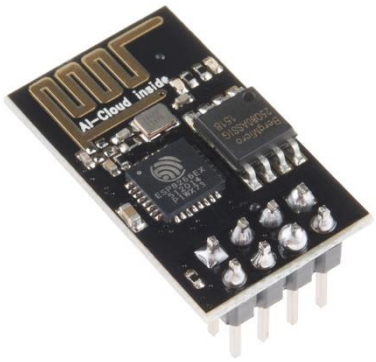


Figure 3-7 WiFi Module (ESP8266)

3.2.6 Android O.S. with Specific Application as Output System

Android is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touchscreen devices, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on game consoles, digital cameras, PCs and other electronics. (Kekana)

The following Figure represents the Block diagram of the Project.

3.3 System Flow Chart

Next Figure represents the flow Chart of the System.

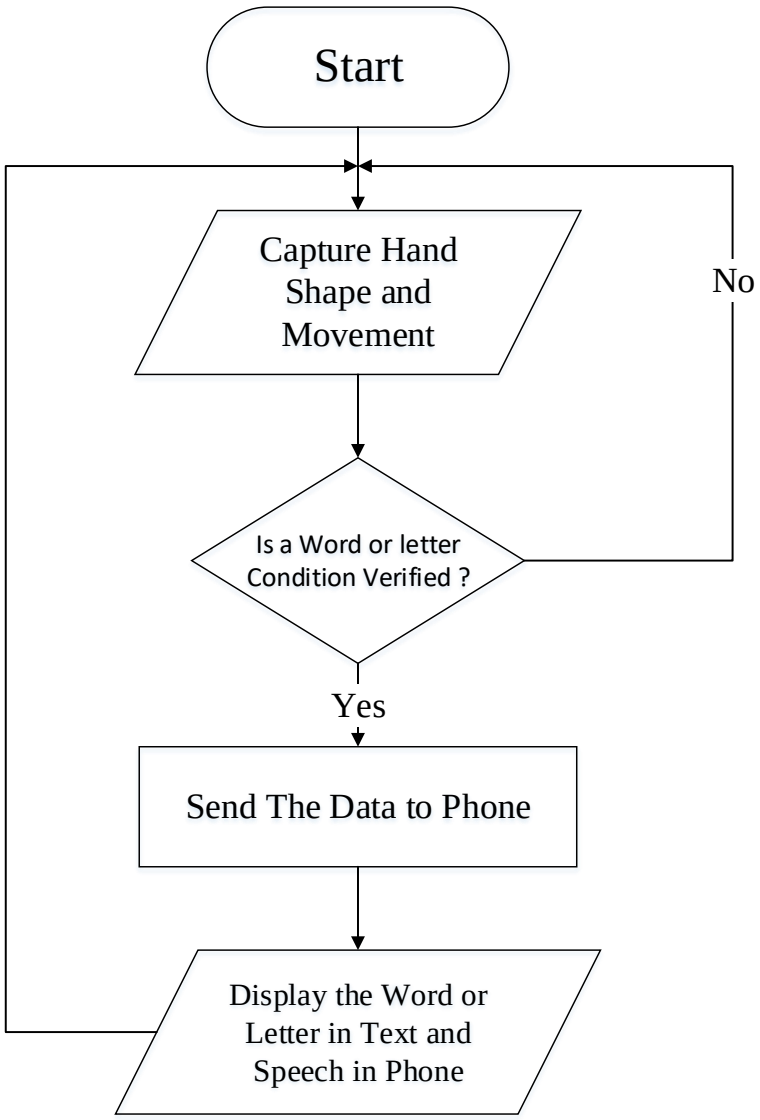


Figure 3-8 System Flow Chart

3.4 Hardware Development

The hardware in our system is basically consist of the flex sensor partition, the contact sensor partition, the accelerometer partition, the glove material, the Arduino and the WIFI module. The hardware also can be divided into three parts: the detection unit, hand position sensing and base station unit.

3.4.1 Detection Unit

The detection unit is the transmitter part of the project where the detection of the hand shape and the processing happen. The core of the detection unit is the glove circuit. There is a small board on the glove which receives all the outputs from the flex sensors and contact sensors. Wires is used to connect sensors with board and jumpers for connection with Arduino pins. The wires from the contacts part enters the Arduino and the other end connected with the Vcc. The wires from the flexes enter the comparator circuit like shown in the next Figure.

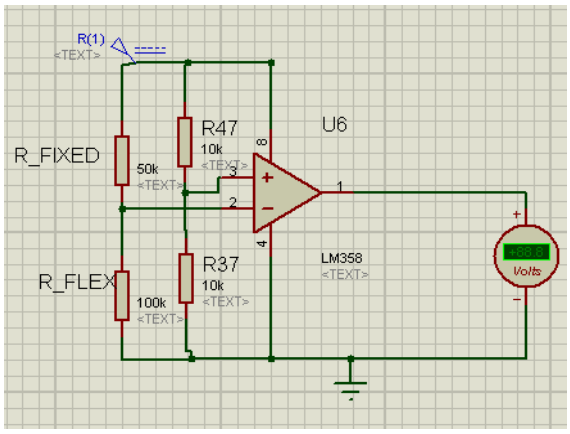


Figure 3-9 Comparator Circuit

The detection unit's transceiver (Wi-Fi Module ESP8266) is used only for transmissions of the translated letters. The coding process for the Wi-Fi was in two steps , first the code that will be inside Wi-Fi transported to the Wi-Fi by the Arduino which has the following configuration: Vcc pin of the module to the Arduino 3.3v, GND to the Arduino GND, ESP TX and RX to Arduino TX and RX in the same order. ESP CH-PD pin to the Vcc source (3.3v). When uploading the code, VGIO-0 is connected to the GND.

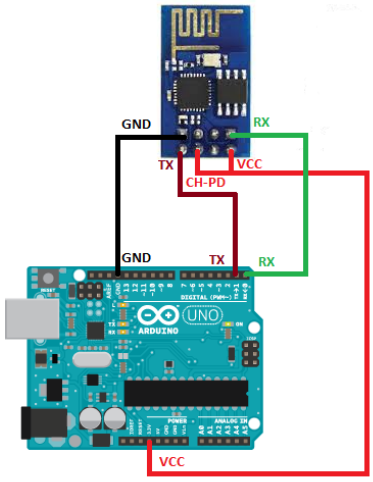


Figure 3-10 Wi-Fi Module ESP8266 Connected with ARDUINO UNO

After the programming of the Wi-Fi then we must program the Arduino to interact with the Wi-Fi and after the first programing finishes the connection differs to the follows : Vcc pin of the module to the Arduino 3.3v, GND to the Arduino GND, ESP TX and RX to Arduino RX and TX in the same order. ESP CH-PD pin to the Vcc source (3.3v). This time VGIO-0 doesn't have to be connected to the GND. See Appendix – B : Wi-Fi Module ESP8266 Code.

3.4.2 Hand Position Sensing:

We will use a lot of sensors to do the hand position sensing. To make the finger sensing it was determined that the best sensor to use to measure finger flexion was a flex sensor with the help of contact sensors. The flex sensor system is the system most dependent on analog components, since the motion of a finger gives us a continuous range of voltages. Due to this, we decided to simulate the flex system to ensure linearity and adequate sensitivity. "Linearity is important, because the microprocessor algorithm assumes that similar ranges of finger flexion will have similar outputs from

the flex sensor system" (Princesa Cloutier 2016). The flex sensors are the most important sensors because most letters can be distinguished based on fingers' flexes. All the fingers except the thumb have two flex sensors, one over the knuckle and the other over the lower joint. This provides two degrees of flexes for these fingers. For the thumb, there is one flex sensors over the lower joint. The contact sensors are used for the letters which have the same flex Figure but differs at the shape, for example letters R, U and V have the same flex but the relative position between their index and middle finger differ, as in Figure 3-11 .

Figure 3- 11 Letters R, U, and V

Contact sensors show which fingers are touching each other. They do not give us any information of the relative position between fingers except for whether they are together or not.

Each of the sensors are connected with a comparator instead of the ARDUINO UNO ports. The ARDUINO UNO has 2 types of Ports: Digital & Analog. Flex Sensors are connected to comparators. Comparators are connected to Digital Ports of ARDUINO UNO which also are connected to contact sensors.

The Accelerometer (Which has 3 output Pins) is connected to Analog Ports of ARDUINO. Processing will be in ARDUINO then its output will be

at TXD Pin (PD1). The output is connected to the input pin of the Wi-Fi Module ESP8266 then a signal will be sent to the Phone to show the letter. The Arduino has internal ADC's which could have been used for the flex sensors. However, each flex sensor only needs to be converted to one bit: flexed or not flexed. A one-bit ADC is not difficult to set up: a Schmitt Trigger with a proper threshold can serve as a ADC. Additionally, there are nine flex sensors, while the Arduino only has 4 pins with internal ADCs. Finally, the accelerometers need to be converted, and they need more than one bit in the digital output.

For these reasons, nine external comparator circuits convert the flex sensor inputs to digital signals. The disadvantage is that the comparator circuits take up a significant amount of space on the detection unit.

That all was for the finger sensing, either for the gesture there are one accelerometer x-y-z axis. the accelerometer is used for movement and orientation detection. Specific hand motions are the only way to detect the letters J and Z. For letters such as G and Q, the only way to distinguish between them is their orientation. while G has the palm facing sideways, Q has the palm facing downwards.

The accelerometer is a sensor that not only gives us the acceleration of the object but also the direction of the acceleration. It have three outputs AcX, AcY and AcZ, as shown in Figure 3-12, each show the acceleration of the three axis in both directions which represented by a positive and a negative value for each output .

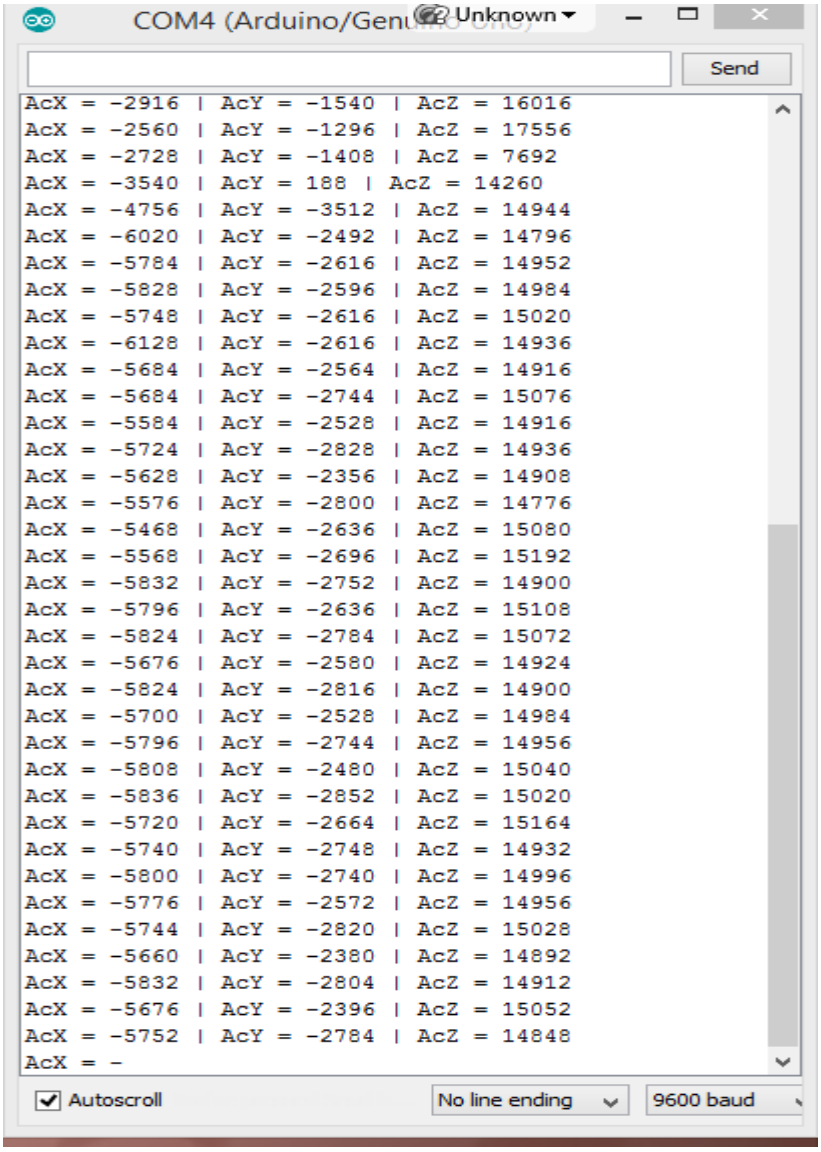
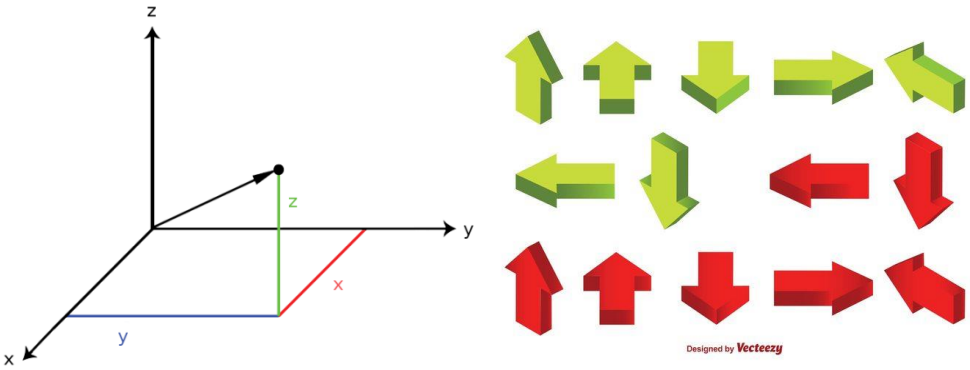


Figure 3-12 Outputs of Accelerometer through ARDUINO IDE

By getting the output, we can exactly determine the direction which the hand is moving at like shown in the next Figure.



(a) (b)

Figure 3-13 (a) Axis Movement, (b) Directions of Movement

By using this method, we can determine the direction of the hand at each second.

We know that the words at the sign language is a combination of the fingers shape and hand movement. We represented the movement of the hand in the program by using the concept of steps. A step is completed by either two ways:

1- The hand acceleration direction at the exact second is similar to the one at the code (like the phrase ‘we are’).

2- The difference in the acceleration in this step and the previous step is similar to the one at the code (like the phrase ‘thank you’).

For more accuracy we made more than root for the hand to take since not always the hand take the exact root it took the last time to present the word, for example at the phrase ‘We are’ the hand moves at the shape of a circle starting by going back, sometimes the move starts by going directly bake then to curve sometimes it stars by the curve to the back , and that’s what we meant by roots .

To know the best number of steps and the required value for the accelerometer parameters and the required delay for the step the team made several tests for the hand movement for the exact word and then noting the output represented and come out with the exact values and number of steps required to represent movement. Next Figure shows how all Detection Unit works. See Appendix A : Arduino Code for more details.

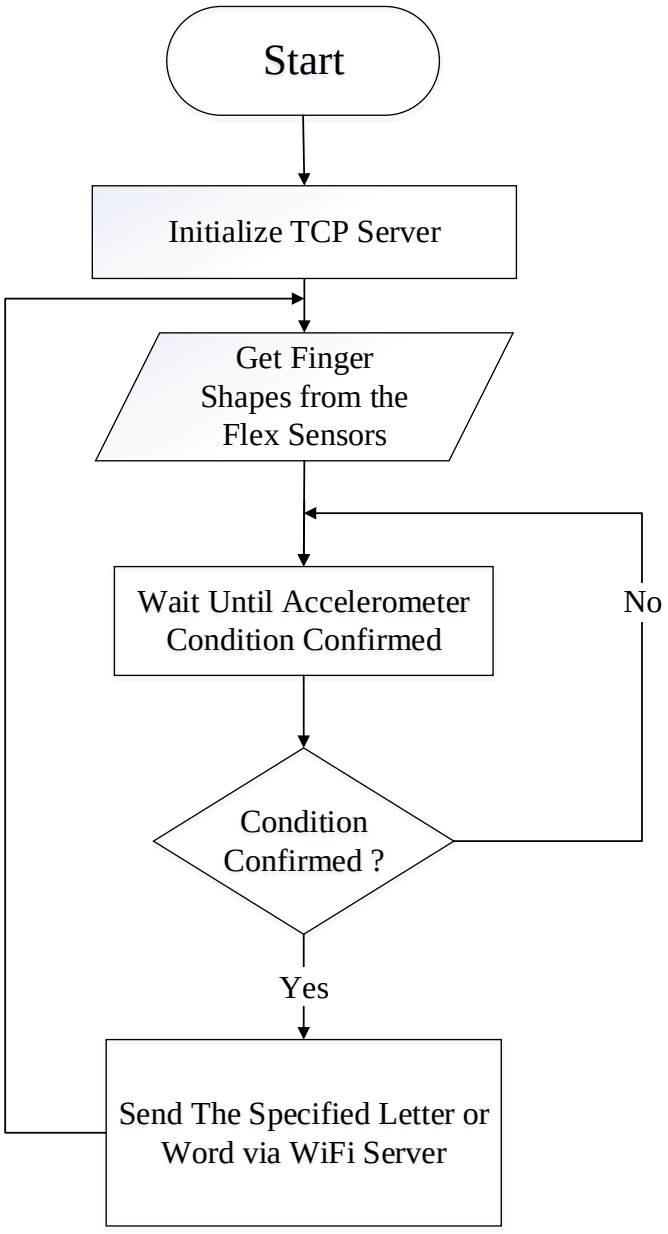


Figure 3-14 Detection Unit Flow Chart

3.4.3 Base station Unit:

3.4.3.1Android APP:

We made an Android app (Figure 3-15 represents the flow diagram of the application and Figure 3-16 shows the main interface of the application) to be the interface between the glove and the user and to be an easy and affordable way to communicate. The app is developed using Android Studio IDE. For more information about the code, see Appendix - C : Android Application Code.

Android Studio is an integrated development environment (IDE) from Google that provides developers with tools needed to build applications for the Android OS platform. Android Studio is available for download on Windows, Mac and Linux. The foundation for Android Studio is based on IntelliJ IDEA. The Android Studio IDE is free to download and use. It has a rich UI development environment with templates to give new developers a launching pad into Android development. Developers will find that Studio gives them the tools to build phone and tablet solutions as well as emerging technology solutions for Android TV, Android Wear, Android Auto, Glass and additional contextual models. (AndroidStudio)

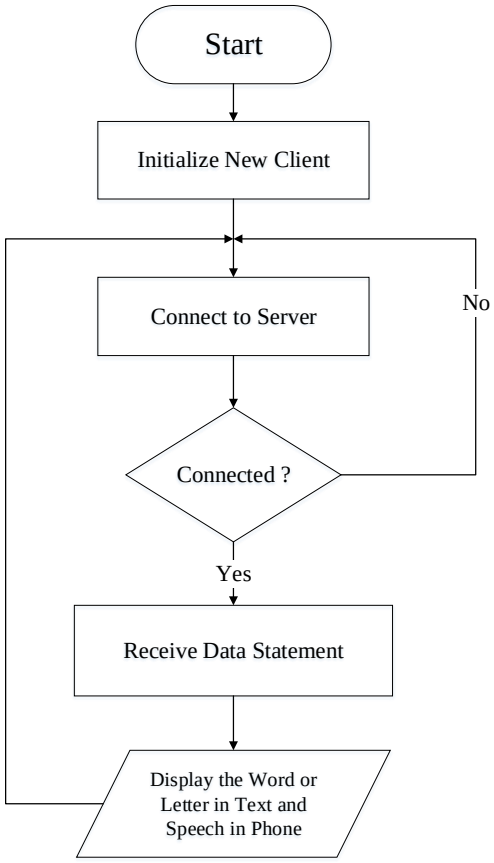


Figure 3-15 Base Station Unit Flow Chart

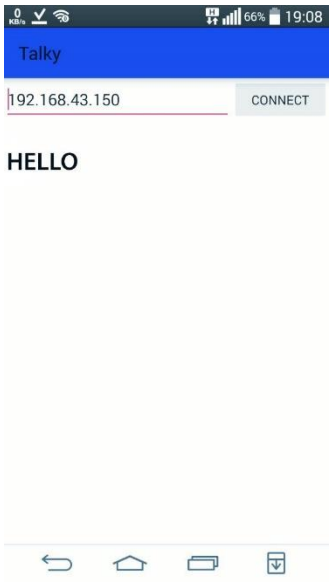


Figure 3-16 Android Application's Main interface

3.4.3.2 Text-to-Speech used

Google Text-to-Speech is a screen reader application developed by Android, Inc. for its Android operating system. It powers applications to read aloud (speak) the text on the screen.

It is a great piece of technology that was developed to help individuals with visual impairments. However, device manufacturers these days enable text-to-speech Android that allows books to be read out loud and new languages to be learned.

Android text to voice was introduced when Android 4.2.2 Jelly Bean was launched with a more conversational capability so that users can have a familiar human-like interaction. More recently, two high-quality digital voices were introduced for Google text-to-speech technology that further enhance Android app that reads text, which is uncommon for Android users.

Now, there are not many Android text to speech app available in the market that fully utilize Google text speech technology

The app is mainly constructed from two parts, the layout and the controlling code. Both of them are written in the android studio and that what makes it easier.

3.4.3.3 Application Layout

The layout (As seen in Figure 3-16) is written in XML programing language, and its controlling code is written in java.

The code itself is mainly containing two main parts. the first one is the part that act like server, a socket server that receive a transmitted data from the Arduino, to make that happen it need the IP number of the Wi-Fi module and a port number that can be conFigured from the Arduino.

The second part is integration to a built-in library called TextToSpeak, it’s a class that’s allow us to pronounce words, and we use it for that purpose.

3.4.3.4 Socket Server

Sockets allow communication between two different processes on the same or different machines. To be more precise, it's a way to talk to other computers using standard Unix file descriptors. In Unix, every I/O action is done by writing or reading a file descriptor. A file descriptor is just an integer associated with an open file and it can be a network connection, a text file, a terminal, or something else.

To a programmer, a socket looks and behaves much like a low-level file descriptor. This is because commands such as read() and write() work with sockets in the same way they do with files and pipes.

Sockets were first introduced in 2.1BSD and subsequently refined into their current form with 4.2BSD. The sockets feature is now available with most current UNIX system releases.

3.5 Simulation

In the simulation for our project we used the software Protues to simulate our project (see Figure 3-17).

The Proteus Design Suite is a proprietary software tool suite used primarily for electronic design automation. The software is used mainly by electronic design engineers and technicians to create schematics and electronic prints for manufacturing printed circuit boards. It was developed in Yorkshire, England by Lab Center Electronics Ltd. (Wikipedia)

We used variable resisters to act like the flex sensors since we couldn’t find a library for them, we have 9 flex sensors (variable resisters) in the

simulation connected with comparators and the out of the comparators is entering the Arduino at pins 0and 2-9 at the digital pins , also we have 4 contact sensors which are represented by 4 normal switches which enters the Arduino at pins 10-13 also at the digital pins , the values of the sensors get processed inside the Arduino and get us the opposite letter or word or phrase as told in our words and letters table (as shown in table 3-1) which represents the hand shape to all letters and some words based on the ASL. Finally, in the hardware the out of the Arduino at pin1 enters the Wi-Fi and got sent to the phone to write and say the words or letters but in the simulation, we can't use the Wi-Fi module so we used what's called Virtual Terminal (Figure 3-18) which the output of the Arduino (from pin1) enters and get us the string and write it in the VT view.

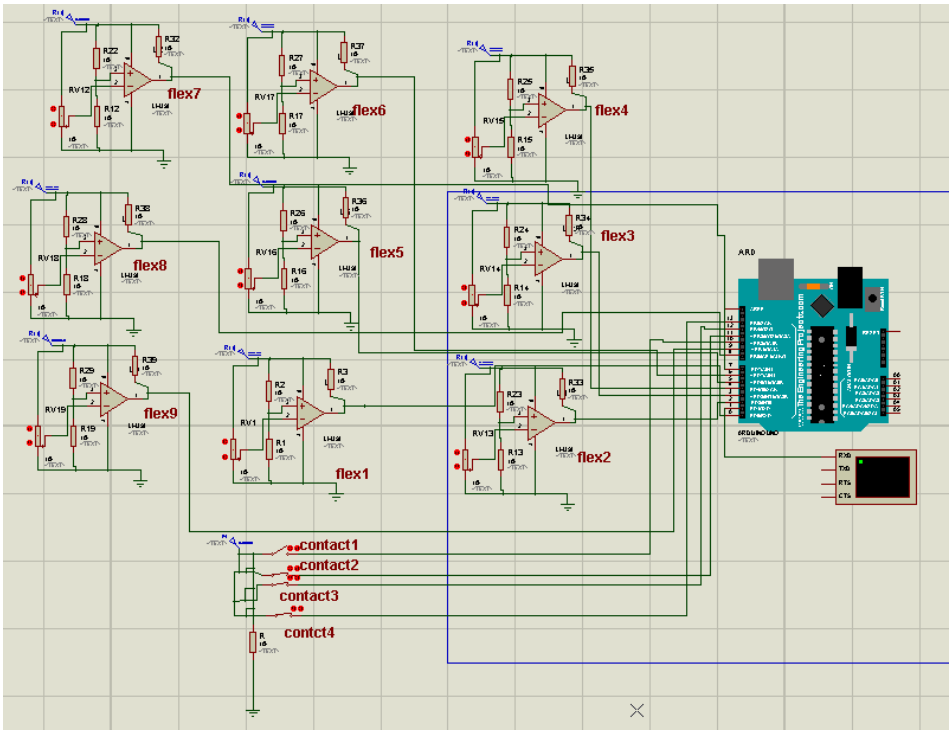


Figure 3-17 Proteus Simulation Circuit

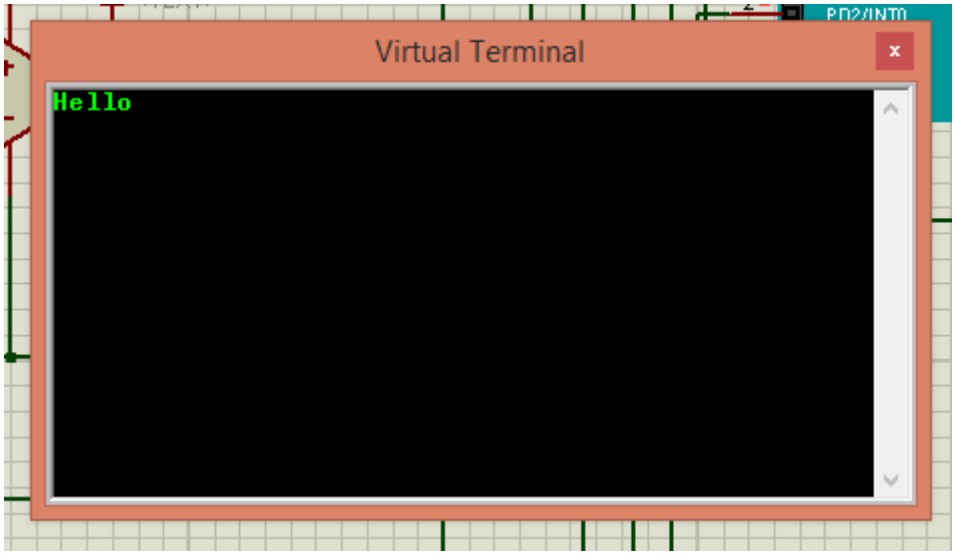


Figure 3-18 Virtual Terminal View

Table 3-1: Words and letters table

Thump Over the lower joint F1	Index Over the lower joint F2	Index over the knuckle F3	Middle finger Over the lower joint F4	Middle finger over the knuckle F5	Ring finger Over the lower joint F6	Ring over the knuckle F7	Pinky Over the lower joint F8	Pinky over the knuckle F9	Top Middle finger Cont1	Side Middle finger Cont2	side ring finger Cont3	Side pinky Cont4	WORD / LETTER
1	0	0	0	0	0	0	0	0	1	1	1	1	A
0	1	1	1	1	1	1	1	1	0	1	1	1	B
1	0	1	0	1	0	1	0	1	0	1	1	1	C
1	1	1	0	1	0	1	0	1	1	0	1	1	D
0	0	1	0	1	0	1	0	1	1	1	1	1	E
0	0	1	1	1	1	1	1	1	0	0	0	0	F
1	1	0	0	0	0	0	0	0	1	0	1	1	G
0	1	1	1	1	0	0	0	0	0	1	0	1	H
0	0	0	0	0	0	0	1	1	1	1	1	1	I
0	0	0	0	0	0	0	1	1	1	1	1	0	J
1	1	1	1	1	0	0	0	0	0	0	0	1	K
1	1	1	0	0	0	0	0	0	1	0	1	1	L
0	0	1	0	1	0	1	0	0	1	1	1	1	M
0	0	1	0	1	0	0	0	0	1	1	1	1	N
0	0	1	0	1	0	1	0	1	1	1	1	1	O
1	1	1	1	0	0	0	0	0	0	0	0	1	P
1	1	1	0	0	0	0	0	0	1	0	1	1	Q
0	1	1	1	1	0	0	0	0	1	0	0	1	R
0	0	0	0	0	0	0	0	0	1	1	1	1	S
0	0	1	0	0	0	0	0	0	1	1	1	1	T
0	1	1	1	1	0	0	0	0	0	1	0	1	U
0	1	1	1	1	0	0	0	0	0	0	0	1	V
0	1	1	1	1	1	1	0	1	0	0	0	1	W
0	0	1	0	0	0	0	0	0	1	0	1	1	X
1	0	0	0	0	0	0	1	1	1	1	1	0	Y
0	1	1	0	0	0	0	0	0	1	0	1	1	Z
1	1	1	1	1	1	1	1	1	0	1	1	1	HELLO

3.6 Summary

We began the design of this glove with its hardware. Flex sensors were tested for linearity and sensitivity, then placed in locations on the glove that the team determined to be the optimal sensing locations. An ADC was implemented to input the flex sensors' analog values to the Arduino so that they may be processed effectively while taking up as few Arduino pins as possible. Contact sensors were tested for responsiveness and then connected to digital pins, as they only sensed if there was contact or not. Accelerometer is used to recognize words and letters that have some hand motion plus fingers gestures. We used wifi module for connection between hand detection unit and base station unit. At base station unit, Android studio IDE is used for programming the application that installed to a smart android phone to receive the output of detection unit and converts it to voice as well as text. Proteus Design Suite was used to simulate the hardware circuit of the project.

The result of Smart Glove for using combination of four flex sensors, four contact sensors and an accelerometer will be discussed next chapter.

CHAPTER 4
RESULT AND DISCUSSION

In this chapter we are going to represent the outputs of the Simulation of the project and Hardware outputs.

4.1 Simulation Results

The simulation as shown in the Figures (3-17) and (3-18) contains of 9 flex sensors, 4 contact sensors represented by switches in addition to an Arduino and a virtual terminal to show to the output.

As represented in chapter 3, because of the lack of libraries in the Proteus simulation, we used a variable resistance instead of the flex sensors and we used switches instead of contact sensors. We noticed that we couldn't make words in the simulation for the lack of the accelerometer library which is essential to do this.

And now, next Figures will represent some samples of the letters in the table in chapter 3 as shown at the simulation virtual terminal.

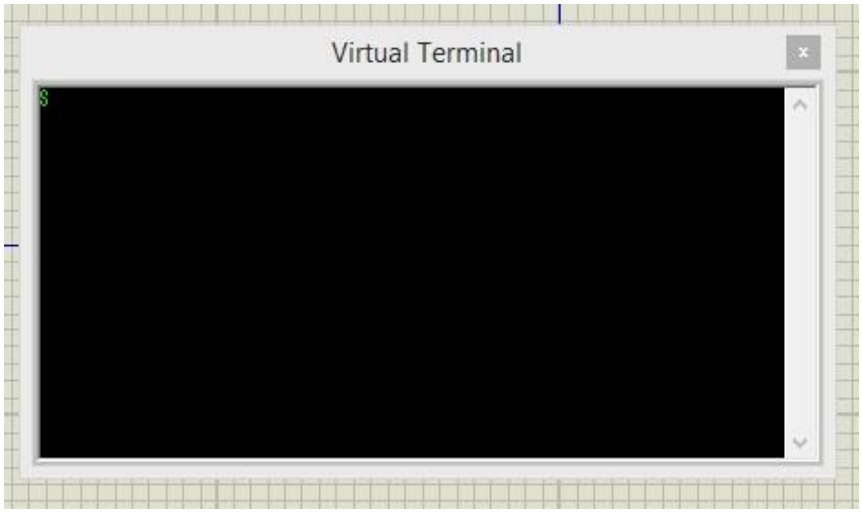


Figure 4-1 Letter 'S' at Simulation Virtual Terminal

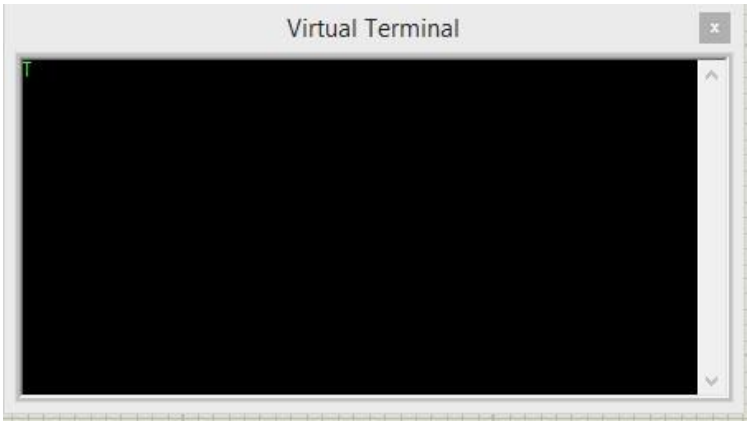


Figure 4-2 Letter ‘T’ at Simulation Virtual Terminal

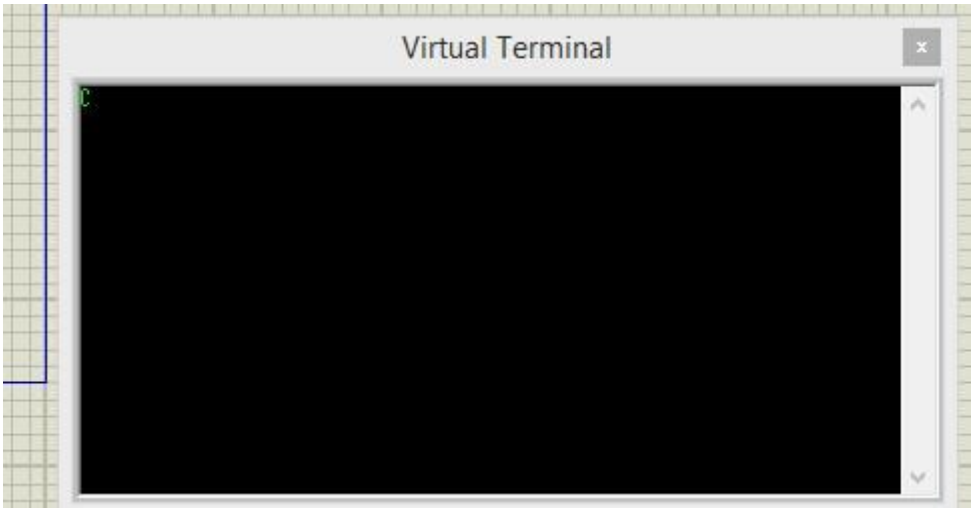


Figure 4-3 Letter ‘C’ at Simulation Virtual Terminal

4.2 Hardware Implementation

Our project prototype contains of 3 flex sensors, 4 contact sensors, Arduino and the ADC circuit which consist of the comparator and the resistors as shown in Figure below.

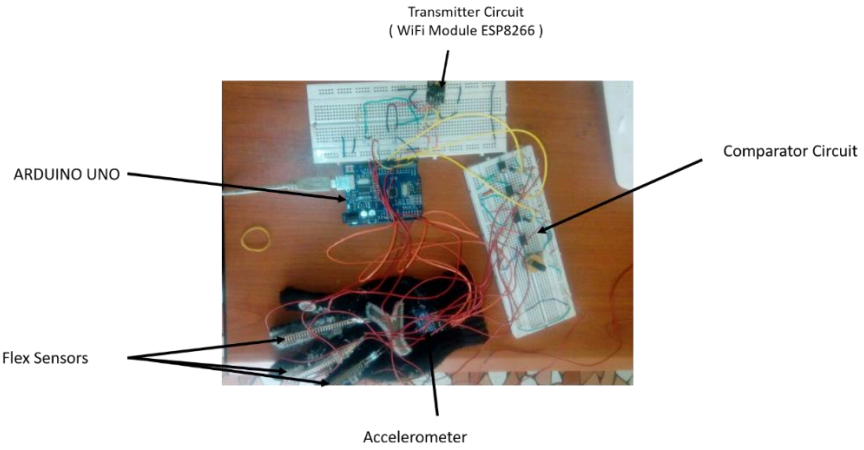


Figure 4-4 Glove Prototype

Keeping this in mind the following table was generated showing the hardware implementation outputs

Table 4-1: Hardware Words and Letters

Index Over the lower joint F2	Middle finger Over the lower joint F4	Ring finger Over the lower joint F6	Top Middle finger Cont1	Side Middle finger Cont2	side ring finger Cont3	Side pinky Cont4	Hand palm Cont5	WORD or LETTER
1	1	1	0	1	1	1	X	Thank you
0	0	0	1	1	1	0	X	I
1	1	0	0	0	0	1	0	K
1	0	0	1	0	1	1	1	L
1	1	0	1	0	0	1	X	R
0	0	0	1	1	1	1	0	S
0	0	0	1	1	1	1	1	T
1	1	0	0	1	0	1	1	U
1	1	0	0	0	0	1	1	V
1	1	1	0	0	0	1	X	W
1	0	0	1	0	1	1	0	At
1	1	1	0	0	0	0	X	HELLO
1	1	1	0	1	1	1	X	Student
1	1	0	0	1	0	1	0	we are

We notice that we have only three flex sensors, considering the high cost of the sensor and some fault occurred.

Due to all that, efficiency has decreased in the matter that we could not represent all the characters using this number of flex sensors.

According to that, we used a special push button in the palm of the hand to be able to differ between the letters and words used in the presentation.

For example, when we want to differ in the R and S letters in our implementation we can't because there is a similarity in the hand cannot be differentiate between them which cannot be done by using three flex sensors as we will show you later.

Next Figures will represent some results of hand gestures through mobile phone application.

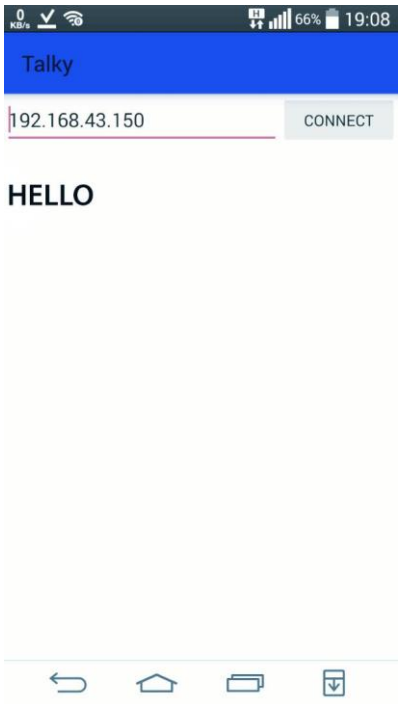


Figure 4-5 ‘HELLO’ gesture result at smart phone application

Figure 4-6 ‘WE ARE’ gesture result at smart phone application

Figure 4-7 ‘STUDENTS’ gesture result at smart phone application

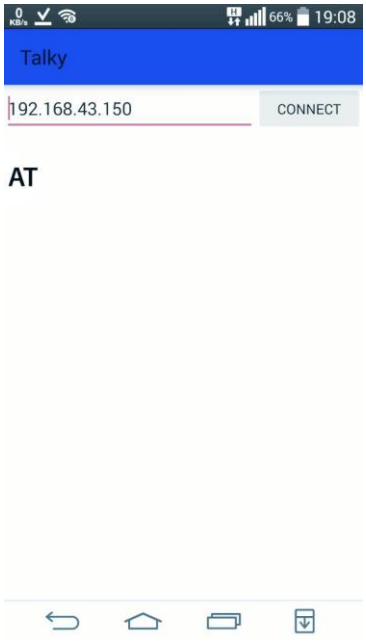


Figure 4-8 ‘AT’ gesture result at smart phone application

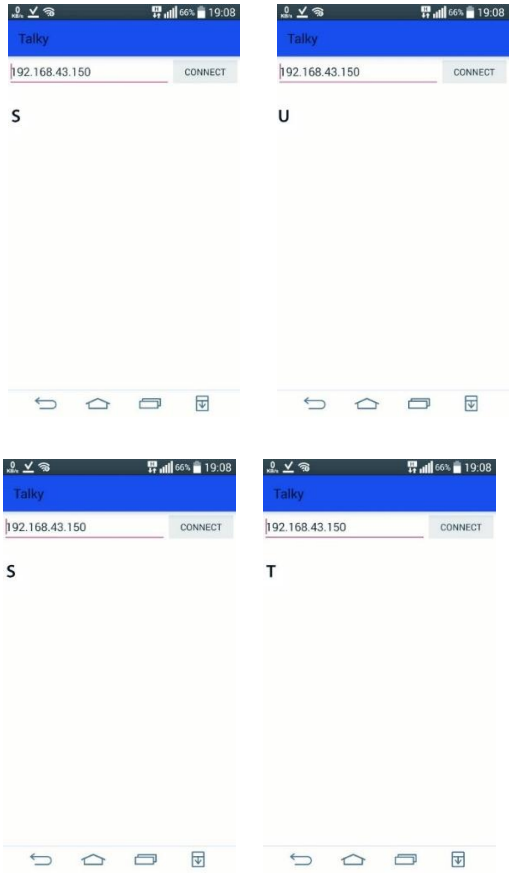


Figure 4-9 ‘S’ ‘U’ ‘T’ gestures result at smart phone application

4.3 Summary

At this chapter, simulation and hardware proof are shown.

Due to lack of sensors libraries, simulation of the project is deficient. Moreover, it is been concluded that the simulation isn’t good to present the project since it is a very real-time system project where the changes of the flexes and accelerometer done simultaneously. So, Hardware implementation became more important to show true results.

We observe that the accuracy of the results is very good, but the internal resistance of the flex is changing for various reasons that might be related to the low quality of the flex itself or wrong usage which causes the low accuracy.

Generally, when we find that when we use the camera the results will be enhanced comparing to the flex results but that in the cost of other things as, wearing a glove is much easier than holding a camera.

CHAPTER 5
CONCLUSION AND RECOMMENDATION

5.1 Conclusion

Through this work, a user-friendly glove was designed the glove was developed to be able to translate sign language gestures into words with high level of accuracy in recognizing gestures, Efficiency was later increased by increasing the number of letters and words that the glove translates.

Moreover, flex and contact sensor was used to collect the data form finger and convert this data to characters.

Furthermore, as proven in chapter 4 we were able to calculate the acceleration of the wrist using an accelerometer then convert it to words as will later on.

Finally processing and converting steps were performed by Arduino result were lastly presented using an android application.

5.2 Recommendation

In the future, there are several changes that can be made to improve the glove into a well-rounded product. To help shape the device into something more slim and comfortable, the team suggests a few improvements to the prototype. Primarily, the team suggests designing a smaller and more compact PCB, which will combine the components of the system together more efficiently. The team also recommends using conductive fabric to replace, the contact sensors, the flex sensors, and/or the wiring to allow for a more lightweight glove that is less bulky and more easily conforms to hand movement.

Another valued improvement to the prototype would be replacement batteries which are slimmer but can still provide the required voltage of 5V and enough power to last for at least 5 hours.

To add to that form a hardware perspective when increasing the number of device used to collect the data the accuracy increases and the number of words covered by the device increases too.

For a quick and easy usage two gloves must be used instead of one From a software perspective the best inherent that must be focused on is the language Arabic must be implemented for the convenience of use for largest number of Arab people.

REFERENCES

AHMAD ZAKI SHUKOR, M. F. M., MUHAMMAD HERMAN JAMALUDDIN, FARIZ BIN ALI@IBRAHIM, MOHD FAREED ASYRAF, MOHD BAZLI BIN BAHAR 2015. A New Data Glove Approach for Malaysian Sign Language Detection. *Elsevier B.V, Procedia Computer Science,* 1877-0509 60 – 67.

ANDROIDSTUDIO. *AndroidStudio - The Official IDE for Android* [Online]. Available: https://developer.android.com/studio/index.html [Accessed 23-10-2017 2017].

ARDUINOPLAYGROUND. *MPU-6050 Accelerometer + Gyro* [Online]. Available: https://playground.arduino.cc/Main/MPU-6050 [Accessed 23-10-2017 2017].

ARDUINOWEBSITE. *Arduino Uno Rev3* [Online]. Available: https://store.arduino.cc/usa/arduino-uno-rev3 [Accessed 22-10-2017 2017].

ARDUINOWEBSITE. *What is Arduino?* [Online]. Available: https://www.arduino.cc/en/Guide/Introduction [Accessed 20-10-2017 2017].

EELCTRONICS, L. A. *How to Build a Flex Sensor Circuit with a Voltage Comparator* [Online]. Available: http://www.learningaboutelectronics.com/Articles/Flex-sensor-circuit-with-a-voltage-comparator.php [Accessed 20-10-2017 2015].

K.V.FALE, A. P., PRATIK CHAUDHARI, PRADEEP JADHAV 2016. Smart Glove: Gesture Vocalizer for Deaf and Dumb People. *International Journal of Innovative Research in Computer and Communication Engineering,* 4**,** 7.

KANIKA RASTOGI , P. B. May 2016. A Review Paper on Smart Glove - Converts Gestures into Speech and Text. *International Journal on Recent and Innovation Trends in Computing and Communication* 4**,** 92 - 94.

KEKANA, I. L. *End User Computing*.

MOHAMMAD TAYE, M. A. S., MOYAD RAYYAN AND HUSAM YOUNIS February 2016. ANYONE CAN TALK TOOL. *Computer Applications: An International Journal (CAIJ),* 3.

MOHAMMED ELMAHGIUBI, M. E., NABIL DRAWIL, AND MOHAMED SAMIR ELBUNI June 2015. Sign Language Translator and Gesture Recognition. *IEEE*.

PRINCESA CLOUTIER , A. H. R. M. 2016. *Prototyping a Portable, Affordable Sign Language Glove.* Bachelor of Science, Faculty of Worcester Polytechnic Institute.

SHAILENDRA KUMAR BOHIDAR, K. P., PRAKASH KUMAR SEN 2015. ROBOTIC SENSOR: CONTACT AND NON-CONTACT SENSOR. *International Conference on Emerging Trends in Technology, Science and Upcoming Research in Computer Science***,** 67.

SIGNSOFLIFEASL. February 27, 2013. *3 Forms of Sign Language: ASL vs. PSE vs. SEE* [Online]. Available: https://signsoflifeasl.wordpress.com/2013/02/27/3-forms-of-sign-language-asl-vs-pse-vs-see/ [Accessed 5-4-2017 2017].

SPARKFUN. *WiFi Module - ESP8266* [Online]. Available: https://www.sparkfun.com/products/13678 [Accessed 23-10-2017 2017].

WIKIPEDIA. *Proteus Design Suite* [Online]. Available: https://en.wikipedia.org/wiki/Proteus_Design_Suite [Accessed 24-10-2017 2017].

APPENDICES

Appendix – A : Arduino code

```
#include<Wire.h>

const int MPU_addr=0x68;  // I2C address of the MPU-6050

int16_t
AcX,AcY,AcZ,Tmp,AcX1,AcY1,AcZ1,AcX2,AcY2,AcZ2,AcX0,AcY0,AcZ
0;

int flex1 = 7;

int flex2 = 8;

int flex3 = 9;

int contact4 = 13;

int contact1 = 10;

int contact2 = 11;

int contact3 = 12;

int contact5 = 2;

int counter = 0;

int con1=3;

int con2=4;

int con3=5;

int con4=6;

void setup() {

Wire.begin();

Wire.beginTransmission(MPU_addr);

Wire.write(0x6B);  // PWR_MGMT_1 register

Wire.write(0);     // set to zero (wakes up the MPU-6050)

Wire.endTransmission(true);

Serial.begin(9600);


pinMode(flex1, INPUT);

pinMode(flex2, INPUT);
```

```
pinMode(flex3, INPUT);
pinMode(contact1, INPUT);
pinMode(contact2, INPUT);
pinMode(contact3, INPUT);
pinMode(contact4, INPUT);

pinMode(contact5, INPUT);
pinMode(con1, OUTPUT);
pinMode(con2, OUTPUT);
pinMode(con3, OUTPUT);
pinMode(con4, OUTPUT);

}
int q=0, a1=0, a2=0,f=0, w=1,h=0,j=1 ;
void loop() {
// read the input pin:
int F2 = digitalRead(flex1);
int F4 = digitalRead(flex2);
int F6 = digitalRead(flex3);
int C1 = digitalRead(contact1);
int C2 = digitalRead(contact2);
int C3 = digitalRead(contact3);
int C4 = digitalRead(contact4);
int C5= digitalRead(contact5);
while ( F2 == HIGH  and F4 == HIGH  and F6 == HIGH and C1
== LOW and C2 == LOW and C3 ==        LOW and C4 == LOW  )
{
Wire.beginTransmission(MPU_addr);
Wire.write(0x3B);  // starting with register 0x3B
(ACCEL_XOUT_H)
```

```
Wire.endTransmission(false);
Wire.requestFrom(MPU_addr,14,true);  // request a total of
14 registers
AcX=Wire.read()<<8|Wire.read();  // 0x3B (ACCEL_XOUT_H) &
0x3C (ACCEL_XOUT_L)
AcY=Wire.read()<<8|Wire.read();  // 0x3D (ACCEL_YOUT_H) &
0x3E (ACCEL_YOUT_L)
AcZ=Wire.read()<<8|Wire.read();  // 0x3F (ACCEL_ZOUT_H) &
0x40 (ACCEL_ZOUT_L)
//  Serial.print("\nAcX = "); Serial.print(AcX);
//  Serial.print(" | AcY = "); Serial.print(AcY);
//  Serial.print(" | AcZ = "); Serial.print(AcZ);

if (q==0)
{

AcX1=AcX;
AcZ1=AcZ;
AcY1=AcY;
q++;
}
else
{
AcX2=AcX;
AcZ2=AcZ;
AcY2=AcY;
if (w==1)
{
if ( AcY2-AcY1 <-5000  )
{
```

```
if ( AcX2-AcX1 > 1500 )
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
w++;
f++;
}
else
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
w++;
}
}
}
else if ( w==2 )
{
if ( f==0 and AcX2-AcX1 > 1500 and AcY2-AcY1 > 5000 )
     {
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
w++;
}
else if ( AcX2-AcX1 < -1500 and AcY2-AcY1 > 2500  )
{
AcX1=AcX2;
```

```
AcZ1=AcZ2;
AcY1=AcY2;
w++;
}
else
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
w=1;
f=0;
}
}
else if ( w==3 )
{
if ( f==0 and AcX2-AcX1 < -1500 and AcY2-AcY1 > 5000 )
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
q=0;
w=1;
Serial.print("\nHello ");
//   digitalWrite(con1, HIGH);digitalWrite(con2,
LOW);digitalWrite(con3, HIGH);digitalWrite(con4, HIGH);
//              delay(200);
//              digitalWrite(con1, HIGH);digitalWrite(con2,
HIGH);digitalWrite(con3, HIGH);digitalWrite(con4, HIGH);
}
```

```
else if ( AcY2-AcY1 > 5000  )
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
q=0;
w=1;
Serial.print("\nHello ");
}
else
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
w=1;
f=0;
}
}
}
// Serial.print(" \n"); Serial.print(q);
Serial.print(w);Serial.print(f);
delay(200);}
// S , T are the same
while ( F2 == LOW and F4 == LOW and F6 == LOW and C1 ==
HIGH and C2 == HIGH and C3 == HIGH and C4 == HIGH  ) {
if (C5==1)
{
Wire.beginTransmission(MPU_addr);
Wire.write(0x3B);  // starting with register 0x3B
(ACCEL_XOUT_H)
```

```
Wire.endTransmission(false);
Wire.requestFrom(MPU_addr,14,true);  // request a total of
14 registers
AcX=Wire.read()<<8|Wire.read();  // 0x3B (ACCEL_XOUT_H) &
0x3C (ACCEL_XOUT_L)
AcY=Wire.read()<<8|Wire.read();  // 0x3D (ACCEL_YOUT_H) &
0x3E (ACCEL_YOUT_L)
AcZ=Wire.read()<<8|Wire.read();  // 0x3F (ACCEL_ZOUT_H) &
0x40 (ACCEL_ZOUT_L)
//  Serial.print("\nAcX = "); Serial.print(AcX);
//  Serial.print(" | AcY = "); Serial.print(AcY);
//  Serial.print(" | AcZ = "); Serial.print(AcZ);
if ( 2000<AcY>-2000 and AcX>13000 and AcZ>2000)
w++;
else
w=1;
if ( w==5 ){
Serial.print( "T" );
w=1;}
delay(500);
}
}
if (C5==0)
{
Wire.beginTransmission(MPU_addr);
Wire.write(0x3B);  // starting with register 0x3B
(ACCEL_XOUT_H)
Wire.endTransmission(false);
Wire.requestFrom(MPU_addr,14,true);  // request a total of
14 registers
```

```
AcX=Wire.read()<<8|Wire.read();  // 0x3B (ACCEL_XOUT_H) &
0x3C (ACCEL_XOUT_L)

AcY=Wire.read()<<8|Wire.read();  // 0x3D (ACCEL_YOUT_H) &
0x3E (ACCEL_YOUT_L)

AcZ=Wire.read()<<8|Wire.read();  // 0x3F (ACCEL_ZOUT_H) &
0x40 (ACCEL_ZOUT_L)

//  Serial.print("\nAcX = "); Serial.print(AcX);

//  Serial.print(" | AcY = "); Serial.print(AcY);

//  Serial.print(" | AcZ = "); Serial.print(AcZ);

if ( 2000<AcY>-2000 and AcX>13000 and AcZ>2000)

w++;

else

w=1;

if ( w==5 ){

Serial.print( "S" );

w=1;}

delay(500);

}

}

}

// 'student' and 'thank you' have the same hand shape

while ( F2 == LOW and F4 == LOW  and F6 == LOW and C1 ==
LOW and C2 == HIGH and C3 == HIGH and C4 == HIGH  ) {

Wire.beginTransmission(MPU_addr);

Wire.write(0x3B);  // starting with register 0x3B
(ACCEL_XOUT_H)

Wire.endTransmission(false);

Wire.requestFrom(MPU_addr,14,true);  // request a total of
14 registers

AcX=Wire.read()<<8|Wire.read();  // 0x3B (ACCEL_XOUT_H) &
0x3C (ACCEL_XOUT_L)
```

```
AcY=Wire.read()<<8|Wire.read();  // 0x3D (ACCEL_YOUT_H) &
0x3E (ACCEL_YOUT_L)
AcZ=Wire.read()<<8|Wire.read();  // 0x3F (ACCEL_ZOUT_H) &
0x40 (ACCEL_ZOUT_L)
//  Serial.print("\nAcX = "); Serial.print(AcX);
//  Serial.print(" | AcY = "); Serial.print(AcY);
//  Serial.print(" | AcZ = "); Serial.print(AcZ);
//
if (q==0)
{
AcX1=AcX;
AcZ1=AcZ;
AcY1=AcY;
q++;
}
else
{
AcX2=AcX;
AcZ2=AcZ;
AcY2=AcY;
if (w==1)
{
if ( AcY2-AcY1<-1000 and AcZ2-AcZ1>4000 )
{
AcX0=AcX;
AcZ0=AcZ;
AcY0=AcY;

AcX1=AcX2;
```

```
AcZ1=AcZ2;
AcY1=AcY2;
w++;
}
}
else if ( w==2 )
{
if ( AcY2-AcY1>1000 and AcZ2-AcZ1<-1000 )
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
Serial.print(" \n student");
w=1;
q=0;
}
else
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
w=1;
}
}
}
//    Serial.print("\n");Serial.print(w);Serial.print(q);
if (h==0)
{
AcX1=AcX;
```

```
AcZ1=AcZ;
AcY1=AcY;
h++;
}
else
{
AcX2=AcX;
AcZ2=AcZ;
AcY2=AcY;
if (j==1)
{
if (  AcZ2-AcZ1>5000  )
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
j++;
}
}
else if ( j == 2  )
{
if (AcZ2-AcZ1<-5000  )
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
j++;
}
else
```

```
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
j=1;
f=0;}}
else if ( j==3 )
{
if (AcZ2<-5000 )
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
h=0;
j=1;
f=0;
Serial.print("\nThanks ");
}
else
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
j=1;
f=0;
}
}
}
```

```
Serial.print(" \n"); Serial.print(q);
Serial.print(w);Serial.print(f);

delay(200);

}

while ( F2 == LOW and F4 == LOW and F6 == LOW and C1 ==
HIGH and C2 == HIGH and C3 == HIGH and C4 == HIGH  ) {

Wire.beginTransmission(MPU_addr);

Wire.write(0x3B);  // starting with register 0x3B
(ACCEL_XOUT_H)

Wire.endTransmission(false);

Wire.requestFrom(MPU_addr,14,true);  // request a total of
14 registers

AcX=Wire.read()<<8|Wire.read();  // 0x3B (ACCEL_XOUT_H) &
0x3C (ACCEL_XOUT_L)

AcY=Wire.read()<<8|Wire.read();  // 0x3D (ACCEL_YOUT_H) &
0x3E (ACCEL_YOUT_L)

AcZ=Wire.read()<<8|Wire.read();  // 0x3F (ACCEL_ZOUT_H) &
0x40 (ACCEL_ZOUT_L)

//  Serial.print("\nAcX = "); Serial.print(AcX);

//  Serial.print(" | AcY = "); Serial.print(AcY);

//  Serial.print(" | AcZ = "); Serial.print(AcZ);

if ( 2000<AcY>-2000 and AcX>13000 and AcZ>2000)

w++;

else

w=1;

if ( w==5 ){

Serial.print( "I" );

w=1;}

delay(500);

}

}
```

```
//K and V

while ( F2 == HIGH and F4 == HIGH and F6 == LOW  and C1 == LOW and C2 == LOW and C3 == LOW and C4 == HIGH and C5 == LOW ) {

if (C5==0)

{

Wire.beginTransmission(MPU_addr);

Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)

Wire.endTransmission(false);

Wire.requestFrom(MPU_addr,14,true);  // request a total of 14 registers

AcX=Wire.read()<<8|Wire.read();  // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)

AcY=Wire.read()<<8|Wire.read();  // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)

AcZ=Wire.read()<<8|Wire.read();  // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)

Serial.print("\nAcX = "); Serial.print(AcX);

Serial.print(" | AcY = "); Serial.print(AcY);

Serial.print(" | AcZ = "); Serial.print(AcZ);

if ( 2000<AcY>-2000 and AcX>13000 and AcZ>2000)

w++;

else

w=1;

if ( w==5 ){

Serial.print( "K" );

w=1;}

delay(500);

}}

if ( C5==1 )
```

```
{
Wire.beginTransmission(MPU_addr);
Wire.write(0x3B);  // starting with register 0x3B
(ACCEL_XOUT_H)
Wire.endTransmission(false);
Wire.requestFrom(MPU_addr,14,true);  // request a total of
14 registers
AcX=Wire.read()<<8|Wire.read();  // 0x3B (ACCEL_XOUT_H) &
0x3C (ACCEL_XOUT_L)
AcY=Wire.read()<<8|Wire.read();  // 0x3D (ACCEL_YOUT_H) &
0x3E (ACCEL_YOUT_L)
AcZ=Wire.read()<<8|Wire.read();  // 0x3F (ACCEL_ZOUT_H) &
0x40 (ACCEL_ZOUT_L)
Serial.print("\nAcX = "); Serial.print(AcX);
Serial.print(" | AcY = "); Serial.print(AcY);
Serial.print(" | AcZ = "); Serial.print(AcZ);
if ( 2000<AcY>-2000 and AcX>13000 and AcZ>2000)
w++;
else
w=1;
if ( w==5 ){
Serial.print( "V" );
w=1;}
delay(500);
}
}
}
//L and AT
while ( F2 == HIGH and F4 == LOW and  F6 == LOW and C1 ==
HIGH and C2 == LOW and C3 == HIGH and C4 == HIGH  ) {
Wire.beginTransmission(MPU_addr);
```

```
Wire.write(0x3B);  // starting with register 0x3B
(ACCEL_XOUT_H)

Wire.endTransmission(false);

Wire.requestFrom(MPU_addr,14,true);  // request a total of
14 registers

AcX=Wire.read()<<8|Wire.read();  // 0x3B (ACCEL_XOUT_H) &
0x3C (ACCEL_XOUT_L)

AcY=Wire.read()<<8|Wire.read();  // 0x3D (ACCEL_YOUT_H) &
0x3E (ACCEL_YOUT_L)

AcZ=Wire.read()<<8|Wire.read();  // 0x3F (ACCEL_ZOUT_H) &
0x40 (ACCEL_ZOUT_L)

//  Serial.print("\nAcX = "); Serial.print(AcX);

//  Serial.print(" | AcY = "); Serial.print(AcY);

//  Serial.print(" | AcZ = "); Serial.print(AcZ);

if (C5==1)

{

if ( 2000<AcY>-2000 and AcX>13000 and AcZ>2000)

w++;

else

w=1;

if ( w==5 ){

Serial.print( "L" );

w=1;}

delay(500);

}}

else if (c5==0)

{

if (q==0)

{

AcX1=AcX;
```

```
AcZ1=AcZ;
AcY1=AcY;
q++;
}
else
{
AcX2=AcX;
AcZ2=AcZ;
AcY2=AcY;
if (w==1)
{
if ( AcX2-AcX1>3000 and AcX2>16000 )
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
w++;
}
}
else if ( w==2 )
{
if (AcX2<3000 and AcZ2-AcZ1<-5000)
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
w++;
f++;
}
```

```
else if ( AcX2<3000 and AcZ2-AcZ1>3000  )
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
w++;
}
else
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
w=1;
f=0;
}
}
else if ( w==3 )
{
if ( f==0 and AcX2<-10000 )
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
Serial.print("\nAT");
q=0;
w=1;
f=0;
}
else if ( AcX2<-10000  )
```

```
{  AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
Serial.print("\nAT");
q=0;
w=1;
f=0;
}
else
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
w=1;
f=0;
}
}
}
//Serial.print("\n");Serial.print(q);Serial.print(w);Serial
.print(f);
delay(200);
}
}
}
while ( F2 == HIGH and F4 == HIGH  and F6 == LOW and C1 ==
HIGH and C2 == LOW and C3 == LOW and C4 == HIGH  ) {
Wire.beginTransmission(MPU_addr);
Wire.write(0x3B);  // starting with register 0x3B
(ACCEL_XOUT_H)
Wire.endTransmission(false);
```

```
Wire.requestFrom(MPU_addr,14,true);  // request a total of
14 registers

AcX=Wire.read()<<8|Wire.read();  // 0x3B (ACCEL_XOUT_H) &
0x3C (ACCEL_XOUT_L)

AcY=Wire.read()<<8|Wire.read();  // 0x3D (ACCEL_YOUT_H) &
0x3E (ACCEL_YOUT_L)

AcZ=Wire.read()<<8|Wire.read();  // 0x3F (ACCEL_ZOUT_H) &
0x40 (ACCEL_ZOUT_L)

Serial.print("\nAcX = "); Serial.print(AcX);

Serial.print(" | AcY = "); Serial.print(AcY);

Serial.print(" | AcZ = "); Serial.print(AcZ);

if ( 2000<AcY>-2000 and AcX>13000 and AcZ>2000)

w++;

else

w=1;

if ( w==5 ){

Serial.print( "R" );

w=1;}

delay(500);

}

}

//U and "we are" have the same gesture

while (F2 == HIGH and F4 == HIGH and F6 == LOW and C1 ==
LOW and C2 == HIGH and C3 == LOW and C4 == HIGH  ) {

Wire.beginTransmission(MPU_addr);

Wire.write(0x3B);  // starting with register 0x3B
(ACCEL_XOUT_H)

Wire.endTransmission(false);

Wire.requestFrom(MPU_addr,14,true);  // request a total of
14 registers
```

```
AcX=Wire.read()<<8|Wire.read();  // 0x3B (ACCEL_XOUT_H) &
0x3C (ACCEL_XOUT_L)

AcY=Wire.read()<<8|Wire.read();  // 0x3D (ACCEL_YOUT_H) &
0x3E (ACCEL_YOUT_L)

AcZ=Wire.read()<<8|Wire.read();  // 0x3F (ACCEL_ZOUT_H) &
0x40 (ACCEL_ZOUT_L)

Serial.print("\nAcX = "); Serial.print(AcX);

Serial.print(" | AcY = "); Serial.print(AcY);

Serial.print(" | AcZ = "); Serial.print(AcZ);

if (C5==0)

{

if (q==0)

{

AcX1=AcX;

AcZ1=AcZ;

AcY1=AcY;

q++;

}

else

{

AcX2=AcX;

AcZ2=AcZ;

AcY2=AcY;

if (((AcY1-AcY0)  <-3000 && (AcY2-AcY0) <-3000 ) ||( AcY1
<-1000 && AcY2 <-1000 && AcZ1 <-1000 && AcZ2 <-1000 )||(
AcY1 >1000 && AcY2 >1000 && AcZ1 <-1000 && AcZ2 <-1000 )||(
AcY1 >1000 && AcY2 >1000 && 1200>AcZ1<-1200 && 1200>AcZ2<-
1200 )||( AcY1 >1000 && AcY2 >1000 && AcZ1 >1000 &&
AcZ2>1000 )||( AcZ1 >1000 && AcZ2 >1000 && 1200>AcY1<-1200
&& 1200>AcY2<-1200 ))

goto MOEZ ;

if (w==0)
```

```
{
if ( AcY2-AcY0<-1300)
{
AcX0=AcX;
AcZ0=AcZ;
AcY0=AcY;
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
w++;
}
}
else if ( w==1 )
{
if ( AcY2<-1000 and AcZ2<-1000 )
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
w++;
}
else
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
w=1;
}
}
```

```
else if ( w==2 )
{
if ( AcY2>1000 and AcZ2<-1000 )
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
w++;
}
else
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
w=1;
}
}
else if ( w==3 )
{
if ( AcY2>1000 and 1200>AcZ2<-1200 )
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
w++;
}
else if ( AcY2>1000 and AcZ2>1000 )
{
AcX1=AcX2;
```

```
AcZ1=AcZ2;
AcY1=AcY2;
w++;
w++;
}
else
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
w=1;
}
}
else if ( w==4 )
{
if ( AcY2>1000 and AcZ2<-1000 )
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
w++;
}
else  if ( AcZ2>1000 and 1200>AcY2<-1200 )
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
w++;
}
```

```
else
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
w=1;
}
}
else if ( w==5 )
{
if ( AcY2<-1000 )
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
Serial.print("\nWe are");
q=0;
w=1;
}
else
{
AcX1=AcX2;
AcZ1=AcZ2;
AcY1=AcY2;
w=1;
}
}
}
MOEZ :
```

```
Serial.print("\n");
//Serial.print(w);Serial.print(q);
delay(170);
}
if (C5==1)
{
if ( 2000<AcY>-2000 and AcX>13000 and AcZ>2000)
w++;
else
w=1;
if ( w==5 ){
Serial.print( "letter" );
w=1;}
delay(500);
}
}
}
while ( F2 == HIGH and F4 == HIGH and F6 == LOW and C1 == LOW and C2 == LOW and C3 == LOW  and C4 == HIGH  ) {
delay(3000);
Serial.println("V ");
}
while (F2 == HIGH and F4 == HIGH and F6 == HIGH and C1 == LOW and C2 == LOW and C3 == LOW and C4 == HIGH  ) {
Wire.beginTransmission(MPU_addr);
Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
Wire.endTransmission(false);
Wire.requestFrom(MPU_addr,14,true);  // request a total of 14 registers
```

```
AcX=Wire.read()<<8|Wire.read();  // 0x3B (ACCEL_XOUT_H) &
0x3C (ACCEL_XOUT_L)

AcY=Wire.read()<<8|Wire.read();  // 0x3D (ACCEL_YOUT_H) &
0x3E (ACCEL_YOUT_L)

AcZ=Wire.read()<<8|Wire.read();  // 0x3F (ACCEL_ZOUT_H) &
0x40 (ACCEL_ZOUT_L)

Serial.print("\nAcX = "); Serial.print(AcX);

Serial.print(" | AcY = "); Serial.print(AcY);

Serial.print(" | AcZ = "); Serial.print(AcZ);

if ( 2000<AcY>-2000 and AcX>13000 and AcZ>2000)

w++;

else

w=1;

if ( w==5 ){

Serial.print( "W" );

w=1;}

delay(500);

}

}

}
```

Appendix – B : Wi-Fi Module ESP8266 Code

```
#include <ESP8266WiFi.h>
const char* ssid = "ssid";
const char* password = "password";
WiFiServer server(4999);
WiFiClient client;
char data[1500];
int ind = 0;
long c = 0 ;
void setup() {
// put your setup code here, to run once:
Serial.begin(115200);
// Connect to WiFi network
Serial.println();
Serial.println();
Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");}
Serial.println("");
Serial.println("WiFi connected");
server.begin();
Serial.println("Server started");
Serial.setDebugOutput(true);}
void loop() {
// put your main code here, to run repeatedly:
if (client.connected())
```

```
{
client.print(" Thanks for the data   " );client.println(c);
delay(3000);
c++ ;}
if(!client.connected())
{
//try to connect to a new client
client = server.available();
//  Serial.println("A client connected !") ; // i thought that the line above means its offer it self to new client , seems thats not true}
else
{
if (client.available()> 0){
// Serial.println("A client connected !?") ; // this line make the server response everytime it get a data .
while(client.available())
{
data[ind] = client.read();
ind++;}
client.flush();
// Serial print data
for (int j = 0 ; j< ind ; j++)
{
Serial.print(data[j]);}
ind = 0;
// client.print(" Thank you fot the data   ");
} }}
```

Appendix – C : Android Application Code

```
package com.example.ahmedali.sakadik;

import android.content.Intent;

import android.net.Uri;

import android.os.Build;

import android.speech.tts.TextToSpeech;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import java.io.*;

import java.net.*;

import java.util.Locale;

import android.util.Log;

import android.view.View;

import android.widget.*;

import com.google.android.gms.appindexing.Action;

import com.google.android.gms.appindexing.AppIndex;

import com.google.android.gms.appindexing.Thing;

import com.google.android.gms.common.api.GoogleApiClient;


public class MainActivity extends AppCompatActivity
implements TextToSpeech.OnInitListener {

private Socket socket;

private static final int SERVER_PORT = 4999;

// private static final String SERVER_IP =
"192.168.43.150";

private EditText currentEditText, destinationEditText;

private TextView resultTextView;

private Button button;

private PrintWriter printWriter;
```

```
private BufferedReader bufferedReader;
private String currentLocationMessage,
destinationLocationMessage, messageReceived;
private TextToSpeech tts;
Button speackBtn ;
private int MY_DATA_CHECK_CODE = 0;
EditText ipEd ;
String SERVER_IP = "192.168.43.150";
/**
* ATTENTION: This was auto-generated to implement the App
Indexing API.
* See https://g.co/AppIndexing/AndroidStudio for more
information.
*/
private GoogleApiClient client;
//Called when the activity is first created
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
ipEd = (EditText) findViewById(R.id.ipEd);
tts = new TextToSpeech(getApplicationContext(), this);
button = (Button) findViewById(R.id.button);
resultTextView = (TextView) findViewById(R.id.textView);
SERVER_IP = ipEd.getText().toString();
Intent checkTTSIntent = new Intent();
checkTTSIntent.setAction(TextToSpeech.Engine.ACTION_CHECK_T
TS_DATA);
startActivityForResult(checkTTSIntent, MY_DATA_CHECK_CODE);
button.setOnClickListener(
```

```
new Button.OnClickListener() {

public void onClick(View v) {

new Thread(new SendMessage()).start();

}});

// ATTENTION: This was auto-generated to implement the App
Indexing API.

// See https://g.co/AppIndexing/AndroidStudio for more
information.

client = new
GoogleApiClient.Builder(this).addApi(AppIndex.API).build();

}

@Override

protected void onActivityResult(int requestCode, int
resultCode, Intent data) {

if (requestCode == MY_DATA_CHECK_CODE) {

if (resultCode ==
TextToSpeech.Engine.CHECK_VOICE_DATA_PASS) {

//the user has the necessary data - create the TTS

tts = new TextToSpeech(this, this);

}

else {

//no data - install it now

Intent installTTSIntent = new Intent();

installTTSIntent.setAction(TextToSpeech.Engine.ACTION_INSTA
LL_TTS_DATA);

startActivity(installTTSIntent);

}}}

/**

* ATTENTION: This was auto-generated to implement the App
Indexing API.
```

```
* See https://g.co/AppIndexing/AndroidStudio for more
information.
*/
private void speakWords(String speech) {
//speak straight away
tts.speak(speech, TextToSpeech.QUEUE_FLUSH, null);
}
@Override
public void onInit(int status) {
if (status == TextToSpeech.SUCCESS) {
int result = tts.setLanguage(Locale.US);
if (result == TextToSpeech.LANG_MISSING_DATA
|| result == TextToSpeech.LANG_NOT_SUPPORTED) {
Log.e("TTS", "This Language is not supported");}
else {
//   speakWords("Initialization completed "); //TODO
}
} else {
Log.e("TTS", "Initilization Failed!");
}}
//    private void speakOut() {
//
//        CharSequence text = "hey";//messageReceived ;
//
//      //  if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.LOLLIPOP) {
//            tts.speak(text, TextToSpeech.QUEUE_FLUSH,
null,"id1");
//
Toast.makeText(this,"sure",Toast.LENGTH_SHORT).show();
```

```
//        // }
//    }
private class SendMessage implements Runnable {
public void run() {
try {
String inputMessage = currentLocationMessage + "#" +
destinationLocationMessage;
socket = new Socket(InetAddress.getByName(SERVER_IP),
SERVER_PORT);
printWriter = new PrintWriter(socket.getOutputStream());
bufferedReader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
printWriter.println(inputMessage);
printWriter.flush();
currentLocationMessage = "";
destinationLocationMessage = "";
while(true){
messageReceived = bufferedReader.readLine();
runOnUiThread(new Runnable() {
@Override
public void run() {
resultTextView.append(messageReceived+"\n");
speakWords(messageReceived);
}});}
} catch (UnknownHostException e) {
System.out.println(e.getLocalizedMessage());
} catch (IOException e) {
System.out.println(e.getLocalizedMessage());
}}}
//Called when the activity is about to become visible
```

```
@Override
public void onDestroy() {
if (tts != null) {
tts.stop();
tts.shutdown();
}
printWriter.close();
try {
bufferedReader.close();
} catch (IOException e) {
e.printStackTrace();
}
try {
socket.close();
} catch (IOException e)
super.onDestroy();}}
```