

Sign Language Conversion and Training by Machine Learning with an IOT Approach



Rezoan Ahmed Nazib | 13201004

Tanvir Ahmed | 12121131

Niloy Siddique | 12121155

Jahid Hasan Tusher | 12301038

**Department of Computer Science and Engineering
BRAC University**

Supervised by Dr. Md. Khalilur Rhaman

Submitted on: 21 August 2017

Declaration

We, hereby declare that this thesis is based on the results found by ourselves. Materials of work found by other researcher are mentioned by reference. This Thesis, neither in whole or in part, has been previously submitted for any degree.

Signature of Supervisor

.....

Dr. Md. Khalilur Rhaman

Signature of Authors

.....

Rezoan Ahmed Nazib

.....

Tanvir Ahmed

.....

Niloy Siddique

.....

Jahid Hasan Tusher

Acknowledgements

Firstly, we would like to thank the almighty to empowering us to start our exploration and giving us enough passion to successfully conclude it.

Furthermore we offer our earnest and heartiest appreciation to our respected Supervisor Dr. Md. Khalilur Rahman for his commitment, direction and support in leading the exploration and arrangement of the report. His contribution and direction has been of enormous incentive all throughout our thesis.

It would be ungrateful if we don't thank the Centre for Disability in Development (CDD) for their support and time. Their approach for sign language research and development and moreover their sign language trainers really helped us to understand the history, grammar and application of the language.

Last yet not the least, we are grateful to the faculties, seniors, friends and our families who has inspired us all through this trip. We also want to recognize the help we received from various resources over the Internet.

Contents

Declaration.....	i
Acknowledgements	ii
List of Figures.....	v
List of Abbreviation.....	vi
Abstract.....	vii
Chapter 1: Introduction.....	1
1.1 Motivation.....	1
1.2 Our Proposed System.....	2
1.3 ASL Learning Application Procedure.....	2
1.4 Message Passing Application Procedure	3
Chapter 2: Literature Review	4
2.1 Similar Research	4
2.2 Comparing Our Proposed System with Existing Technologies	8
Chapter 3: Research Methodology	9
3.1 First Phase.....	9
3.2 Second Phase	13
3.3 Third Phase	16
Chapter 4: Components	20
4.1 Hand Glove	20
4.2 Firebase.....	20
4.3 IDE (Integrated Development Environment).....	21
4.4 Electronics.....	23
Chapter 5: Communication	27
5.1 Block Diagram	27
5.2 Flex Sensor and Arduino	27
5.3 Force Sensor and Arduino	28
5.4 Arduino and SD Card Module	29
5.5 Arduino and NodeMcu	30
5.6 NodeMcu to Firebase.....	31
5.7 Firebase to Android.....	32
Chapter 6: Algorithm.....	33
6.1 Reading CSV	33
6.2 Node MCU ASL	34
6.3 Node MCU Message Passing.....	35
6.4 Only Callibration	36
6.5 Saving Creating CSV	37

Chapter 7:	Experiment	38
7.1	Taking raw values from flex sensors	38
7.2	Calibrating the flex sensors.....	38
7.3	Gyro integration and calibration	38
7.4	HMM implementation	38
7.5	Tensor Flow Implementations	39
7.6	Weka Classifier and JavaML	39
7.7	KNN algorithm implementation	39
Chapter 8:	Result.....	40
8.1	Calibration.....	40
8.2	Making of the data set.....	40
8.3	Finding Result from the dataset	42
Chapter 9:	Problem Analysisand Discussion	55
Chapter 10:	Conclusion	57
10.1	Concluding Remarks.....	57
10.2	Future Works	57
References.....	Error! Bookmark not defined.	
Appendix.....	61

List of Figures

- Fig. 3.1.1 Initial model10
- Fig. 3.1.2 Initial connection of the gloves with Arduino AT Mega11
- Fig. 3.1.3 Gyro MPU6050 and its schematic diagram12
- Fig. 3.1.4 Angular Rotation12
- Fig. 3.2.1 New glove vs old glove17
- Fig. 3.2.2 The bending points where the plastic tubes are attached with glue17
- Fig. 4.1.1 Hand gloves with plastic tube20
- Fig. 4.2.1 Firebase Interface21
- Fig. 4.3.1 Arduino IDE22
- Fig. 4.3.2 Android Studio23
- Fig. 4.4.1 Arduino Mega 256024
- Fig. 4.4.2 Schematic Diagram of Arduino Mega 256024
- Fig. 4.4.3 Flex sensor25
- Fig. 4.4.4 Force Sensor25
- Fig. 4.4.5 SD Card Module26
- Fig. 4.4.6 NodeMCU26
- Fig. 5.1.1 Block diagram of the system27
- Fig. 5.2.1 Basic Flex Censor Circuit.28
- Fig. 5.2.2 Flex Sensor Circuit Connection with Arduino28
- Fig. 5.3.1 Force Sensor Circuit Connection with Arduino29
- Fig. 5.4.1 SD Card Module Circuit Connection with Arduino30
- Fig. 5.5.1 NodeMcu Circuit connection with Arduino31
- Fig. 5.6.1 Firebase31
- Fig. 6.3.1 User interface of the application86
- Fig. 6.3.2 Launcher activity of learning application.86
- Fig. 6.3.3 Application using87
- Fig. 6.3.4 User interface of the application94
- Fig. 6.1.1 The first window of the Message Passing Application95
- Fig. 6.1.2 ListActivity view96
- Fig. 6.1.3 Message Activity98
- Fig. 8.2.1 Dataset41
- Fig. 8.3.1 Data Returned From Algorithm For A43
- Fig. 8.3.2 Displaying “B” and the corresponding result in console45
- Fig. 8.3.3 Showing some datasets and the lowest distance here is of “B”46
- Fig. 8.3.4 Displaying “C” and the corresponding result in console47
- Fig. 8.3.5 Showing some datasets and the lowest distance here is of “C”48
- Fig. 8.3.6 Displaying “D” and the corresponding result in console49
- Fig. 8.3.7 Showing the values of the predicted outcomes50
- Fig. 8.3.8 Displaying “E” and the corresponding result in console52
- Fig. 8.3.9 Showing the values of the predicted outcomes53

List of Abbreviation

ASL	American Sign Language
UART	Universal Asynchronous Receiver/Transmitter
1D	One Dimensional
2D	Two Dimensional
TX/RX	Transmit/Receive
SD	Secure Digital
SPI	Serial Peripheral Interface
ANN	Artificial Neural Network
I2C	Interface to Communicate
KNN	K Nearest Neighbor

Abstract

For the people, who are living without sound due to disabilities, sign language can be a handy tool to make their life easy and comfortable. Sign language is actually a medium of communication for the people who are either deaf or deaf and mute. Besides, normal people can also use sign language to communicate with these people whose viewpoints always remain unspoken just because of their ill-fate. Generally, mute or deaf people use manual communication and body language to communicate with others. However, there is no such device by which a person can learn sign languages and express their thoughts easily. To make the things easier and simple we decided to create a device that could help to learn sign language via “Learning Application” and a mechanism to translate sign language into text. For translating “Sign Languages” into text we have developed one special glove attached with necessary sensors to take the relative hand position of the user. We have used one Machine-Learning classifier to classify the raw data of the sensors to have better accuracy. Then through the help of one IOT device we have send the data to mobile application where the actual meaning of sign is expressed into text to the receiver. As a part of the whole process, we have developed two different mobile applications. Of these two applications one can be used to train sign languages and other one simply carries the information that the user want to provide to the receiver. However, in some cases deaf and mute people might know the Sign-Language but the normal person might not. Therefore, along with helping the deaf and mute people this training application will also help normal people to understand the language to communicate disabled personnel. We have used American Sign Languages (ASL) as our standard medium for both of the application. The other application has a great usability just after the training session. When a mute and deaf person is already trained how to use sign language with our first application then he/she may simply use our system to express their words with the second application.

We have chosen the smartphone because of the availability and functionality of the device. As Majority number of people owns smartphone it will be easier for anyone to learn and use the sign languages through our device. On the other hand we have used IOT devices for sending words to a longer distance. Nowadays almost every smartphone user is using internet in their handheld device. So it can be easily said that the device is user friendly as well. Overall, this device can help to eliminate the obstacles and communication barrier faced by a deaf or mute person when it comes to communicate with others.

Chapter 1: Introduction

1.1 Motivation

In a society where we are trying to ensure equal right for everyone, there are 360 million people or 5% of the total population of the whole world are deaf and mute and they are being deprived for their disabilities [1]. These people are used to communicate with sign language, which creates a communication gap between the mainstream people and these disable people. In this age of science and technology, there are ways that can remove this gap and improve their social and personal life. With this goal in mind, we have developed a “Sign Language Converter Glove” and “Learning Application” through which they can communicate with the people who do not have any idea about the sign language. This device contains an Arduino Mega, Flex Sensors, Force Sensors, Node MCU, rubber-woolen glove, and a Communication application. Communication via gesture is a route in which the signals made by the client is utilize in a way so that other person can picture it in his mind. On the other hand, human signals are a productive and intense method for connection. This paper is concentrated on building up an assistance for impaired individuals utilizing this signal acknowledgment system. In this framework, the motions are changed into instant messages for correspondence. The essential idea includes the utilization of information gloves worn by crippled individuals. These gloves are planned to utilize Flex sensors and contact sensors but no accelerometer is utilized for tilt location. The four flex sensors are regularly connected to the glove. Flex sensors are simple resistors that have the capacity like simple voltage dividers. If any point of a flex sensor is twisted, there is a change in resistance which is parsed by the ADC of the microcontroller. The greater part of the physical amount around us are constant and we imply that the amount can take any esteem between two extremes. Presently we have brought a physical amount into the electrical area. The electrical amount is just voltage. To bring this amount into computerized area we need to change this into computerized shape. For this, two advanced converter is required.

In the market there are several products using this kind of technology. However, in these technologies most of the cases there are many sensors, which have not been properly utilized. There are many other Sign language converting applications and devices which uses 3D camera to track the movement of a person’s hands and body, image processing, Microsoft’s Kinect motion-sensing equipment, lip-readers. As these techniques need lots, of time and memory, these are inefficient and these are not helping these people.

1.2 Our Proposed System

Our whole system is actually divided into two parts. One part teaches the sign language and the other part actually express the message that the user want to pass. I will go through both of the systems one after another.

1.2.1 Common Steps for both Approach (Message passing and learning)

Sensor Calibration

First, we have calibrated the sensor according to the size of different hands. To use our glove this is the first procedure that a user should do. In this phase, the user will have to squeeze and relax his hand for several time. Inside the program it will simply take the highest and lowest value of the sensor. In addition, it will simply return the highest and lowest values of the sensors in the Serial monitor.

1.2.2 Teaching the Classifier

Firstly, we have taken specific known hand gesture for multiple time. For example, we have taken the sign of “A” for five times. By pressing the controller button, the Arduino program asks to enter the label for the real time hand position. By entering the Label that means the letter, the person has to press enter to save it in the SD card.

1.3 ASL Learning Application Procedure

The first steps of detecting the correct letter for the specific hand gesture is same for both of the case. For this approach, we have different Arduino, NodeMcu and android program. To classify one specific hand movement a person will have to press the controller button once again. It will simply retrieve all the values from the SD card one after another and check for euclidian distance (K-nearest neighbors’ algorithm). Then we sort the distance and take first k values of the nearest label. After that, we check for the unique characters and then we send it to the NodeMcu. The NodeMcu actually does nothing but forward the data to the server and the server forwards the data to our mobile application. If the learning application is asking for a sign “V” and the chunk of unique characters has the value “V” inside it then it will simply tell you that you have shown the correct move and the application will ask you to show another move for a new letter. But if the data that is been send from the gloves does not have the exact

value not even for a single occurrence then the mobile application will show that the user is in wrong gesture and will ask to make the move for the same letter again. This is how the ASL learning application works. In addition, it works fully in real-time.

1.4 Message Passing Application Procedure

In this procedure, the user has to give a correct hand gesture and then he/she has to press the controller button to justify which letter sign he is actually showing. Now after retrieving and comparing with all the values in our dataset the algorithm will check for most occurrence letter in the first k 'th values. Then the algorithm will simply save the letter and then by pressing the same controller it will guess another letter by the hand movement of the user and concatenate the new letter with the previous letter. After completing a full word, we have kept another controller in the upper side of the hand which will push the word to the NodeMcu. Then the NodeMcu will push the word to the server and the server will send the word to our message passing mobile application. Here the application has two different phase. After showing welcome screen, one tap on the screen will carry the user to the next screen where the user will be able to see all of the words he has been pushed throughout the whole time. Long press on any item will bring the user to another activity where the user will be able to see only the current word he is sending to the mobile application. One button is kept there to clear all the words. Otherwise, if two words is received in same window then it will show both of the word and this is the way of expressing the whole sentences. All of this procedure has done in real time. Therefore, the user will be able to send his message to the receiver instantly. Now the classification of the word is under maintenance so the accuracy is low. However, we are working to develop the accuracy of the predicted word and expecting to have a good accuracy soon.

Chapter 2: Literature Review

2.1 Similar Research

The main purpose of our project is making a way to learn ASL for the deaf, mute and the people who do not know ASL. As a result they will be able to communicate through this sign language. We believe with this approach we will be able to remove the barriers to communicate with the deaf and mute people in our society. At the beginning of the project we have researched on some secondary resources based on this. From this sources, we have been able to know the existing functions and determined our work outline.

According to Kumar, Gurjar and Singh [2], the glove has four flex sensors each sits on each finger. The microcontroller consistently checks the bowing of flex sensor. At the point when the signal of the letters make particular word based on the sequence appeared in the LCD. The glove includes a few contact sensors, which help in recognizing couple of comparable motions like of "U" and "V". The precision of each flex sensor is constrained past a specific point. Smaller hands will bring about a bigger level of twist. Therefore, the contrast is very high. Since all correspondence are done through links, our gadget does not meddle with different plans. Any individual who fits into it can utilize the glove; they would just need to prepare on it and create new datasets on the off chance that they wish for a higher forecast precision than the standard or to consolidate new signs.

From Arsan and Ulgen [3] we can understand, this framework can be utilized for changing over gesture based communication to voice and furthermore voice to communication via gestures. A movement catch framework is utilized for communication via gestures transformation and a voice acknowledgment framework for voice change. It catches the signs and directs on the screen as composing. It additionally catches the voice and shows the gesture based communication significance on the screen as motioned picture or video. Microsoft Kinect Sensor XBOX 360 is chosen to use for catching capacities and specialized elements to the movement catch of sign to voice change. Google Voice Recognition is utilized for the voice to sign change. Google Voice Recognition is accessible just on android based projects. Inevitably, the voice acknowledgment program CMU Sphinx is picked. This enables us to join the two segments in Java. Change program is likewise outlined and written in Java. At last, Java based program is created which can make voice acknowledgment, movement catch and change over them two to each other. So a hard of hearing individual effortlessly addresses in

gesture based communication before movement sensor, the individual behind the screen can see effectively without capacity to talk communication through signing and the other way around.

DVI frameworks are normally arranged for little to medium measured vocabularies and might utilize word or expression spotting methods. In the two cases, the basic innovation is pretty much the same. Discourse Recognition Techniques and Template Based Approaches, Statistical Based Approaches and a Knowledge Based Approaches Matching Unknown discourse, which is looked at against an arrangement of pre-recorded words (formats) to locate the best match. A specialist information about varieties in discourse is hand coded into a framework. In which varieties in discourse are demonstrated measurably, utilizing programmed, factual learning methodology, ordinarily the Hidden Markov Models, or HMM. To beat the impediment of the HMMs machine learning techniques could be presented, for example, neural systems and hereditary calculation/programming. The computerized reasoning methodology endeavors to automate the acknowledgment strategy as per the way a man applies its insight in imagining, examining, lastly settling on a choice on the deliberate acoustic elements.

Lin and Villalba [4] demonstrated us, Machine Learning (ML) calculation to make an interpretation of gesture based communication into communicated in English. Each individual's hand is a novel size and shape, and we intended to make a gadget that could give dependable interpretations paying little mind to those distinctions. Our gadget utilizes five Spectra Symbol Flex-Sensors that we use to evaluate how much each finger is bowed, and the MPU-6050 (a three-hub accelerometer and whirligig) can recognize the introduction and rotational development of the hand. These sensors are perused, arrived at the midpoint of, and orchestrated into bundles utilizing an ATmega1284p microcontroller. These bundles are then sent serially to a client's PC to be keep running in conjunction with a Python content. The client makes informational indexes of data from the glove for each motion that ought to the end be deciphered, and the calculation prepares over these datasets to anticipate later at runtime what a client is marking. The point of the investigation is to give a total discourse without knowing gesture based communication. The program has two sections. Right off the bat, the voice acknowledgment part utilizes discourse handling techniques. It takes the acoustic voice flag and changes over it to a computerized motion in PC and afterward show to the client the .gif pictures as result. In addition, the movement acknowledgment part utilizes picture handling

strategies. It utilizes Microsoft Kinect sensor and after that provide for the client the result as voice.

Again, from the source [5], we have known that communication via gestures acknowledgment device and strategy is accommodated making an interpretation of hand signals into discourse or composed content. The contraption incorporates various sensors on the hand, arm and shoulder to quantify dynamic and static motions. The sensors are associated with a microchip to look through a library of motions and create yield flags that would then be able to be utilized to deliver an orchestrated voice or composed content. The mechanical assembly incorporates sensors, for example, accelerometers on the fingers and thumb and two accelerometers on the back of the hand to identify movement and introduction of the hand. Sensors are additionally given on the back of the hand or wrist to distinguish lower arm revolution, a point sensor to identify flexing of the elbow, two sensors on the upper arm to recognize arm rise and pivot, and a sensor on the upper arm to distinguish arm bend. The sensors transmit the information to the chip to decide the shape, position and introduction of the hand with respect to the body of the client.

This venture tries to connect the correspondence hole by planning a convenient glove that catches the client's ASL signals and yields the interpreted content on a cell phone [6]. The glove is outfitted with flex sensors, contact sensors, and a spinner to quantify the flexion of the fingers, the contact amongst fingers, and the turn of the hand. The glove's Arduino UNO microcontroller breaks down the sensor readings to distinguish the signal from a library of educated motions. The Bluetooth module transmits the motion to a cell phone. Utilizing this gadget, one day speakers of ASL might have the capacity to speak with others in a moderate and advantageous way.

According to the journal [7], this framework depicts talk capable hand glove framework which goes for interpretation of gesture based communication to dissect content info and voice. This framework comprises of a discussion capable glove that can be worn by a hard of hearing/moronic individual to encourage the correspondence progressively with other individuals. The framework deciphers the hand finger movement to relating letters utilizing Contact switch sensors and an Arduino Board. Our primary objective is to distinguish 26 letters in order and show message on the LCD. Once the content is gotten on the LCD then content to discourse change operation is completed lastly a voice yield is acquired. Further, the content pick up can likewise be seen on a LCD or any convenient hand held gadget. Our fundamental point is to set an interface between the Deaf or Dumb and typical individuals to enhance the

correspondence abilities so they can discuss conveniently with others. We mount contact switch sensor on the discussion capable hand glove and propose and productive strategy to change over these communications through signing with the assistance of Arduino UNO. This framework will disentangle the correspondence of hard of hearing or imbecilic individuals with individuals ready to ordinary interchanges without the need of a human interpreter.

We can see a project utilizes a sensor glove to catch the indications of American Sign Language performed by a client and makes an interpretation of them into sentences of English dialect [8]. Fake neural systems are utilized to perceive the sensor esteems originating from the sensor glove. These qualities are then ordered in 24 letters in order of English dialect and two accentuation images presented by the creator. Along these lines, quiet individuals can compose finish sentences utilizing this application.

The glove translates the sign language gestures into speech according to the American Sign Language Standard. The glove contained flex and contact sensors to detect the movements of the fingers and bending of the palm. In addition, an accelerometer was built in the glove to measure the acceleration produced by the changing positions of the hand. Principal Component Analysis (PCA) was used to train the glove into recognizing various gestures, and later classify the gestures into alphabets in real time [9]. The glove then established a Bluetooth link with an Android phone, which was used to display the received letters and words and convert the text into speech. The glove was found to have an accuracy of 92%. The systems used a skin color matching algorithm for tracking the hand.

An automatic American Sign Language recognition system is developed using artificial neural network (ANN) and to translate the ASL alphabets into text and sound. A glove circuit is designed with flex sensors, 3- axis accelerometer and sEMG sensors to capture the gestures. The finger bending data is obtained from the flex sensors on each finger whereas the accelerometer provides the trajectories of the hand motion. Some local features are extracted from the ASL alphabets which are then classified using neural network [10]. The proposed system is evaluated for both user-dependent and user-independent conditions successfully for isolated ASL recognition.

2.2 Comparing Our Proposed System with Existing Technologies

2.2.1 For Learning Application

After reviewing the papers and journals, we have come to a decision that there is no such application where a person can learn ASL in real time experience using hand glove. Therefore, from this point of view our system is totally new to the research field with a satisfactory accuracy level. Because we are taking k'nearest values as candidate keys where some garbage values may be there. However, 90% time the algorithm returns the string having the exact letter.

2.2.2 For Message Passing Application

At present, the accuracy of our predicted word is not good but we are working on it and expecting to have an accuracy over 80% in a near future. That means the system will be able to carry the right meaning that the user wanted to express. We have used IOT device so message can be send to anyone who uses a smartphone. We have used machine learning algorithm and a prediction through the algorithm over 90% accuracy will be saved in the dataset. So the more the user will use our system the accuracy will keep increasing.

Chapter 3: Research Methodology

3.1 First Phase

3.1.1 Making of the initial gloves

Materials

The initial gloves were a woolen one. We choose this fabric or material as it is flexible and it fixes with the hand tightly. We attached rubber pads and plastic tubes over the gloves to hold the flex sensors. The rubber pads and tubes were attached to the gloves with the help of the super glue.

As the gloves was made of wool so it was comparatively easier to stick the rubber pad over the gloves. If we have chosen plastic or rubber gloves the strength of the glue may have reduced due to less friction. However, we could have used the plastic tube directly over the gloves but the tubes do not bend smoothly and while bending it breaks at the point where pressure is applied. That is why we chose to put a layer of rubber padding in between the gloves and the tubes holding the flexes. Due to the layer of rubber padding the flexes could bend smoothly.

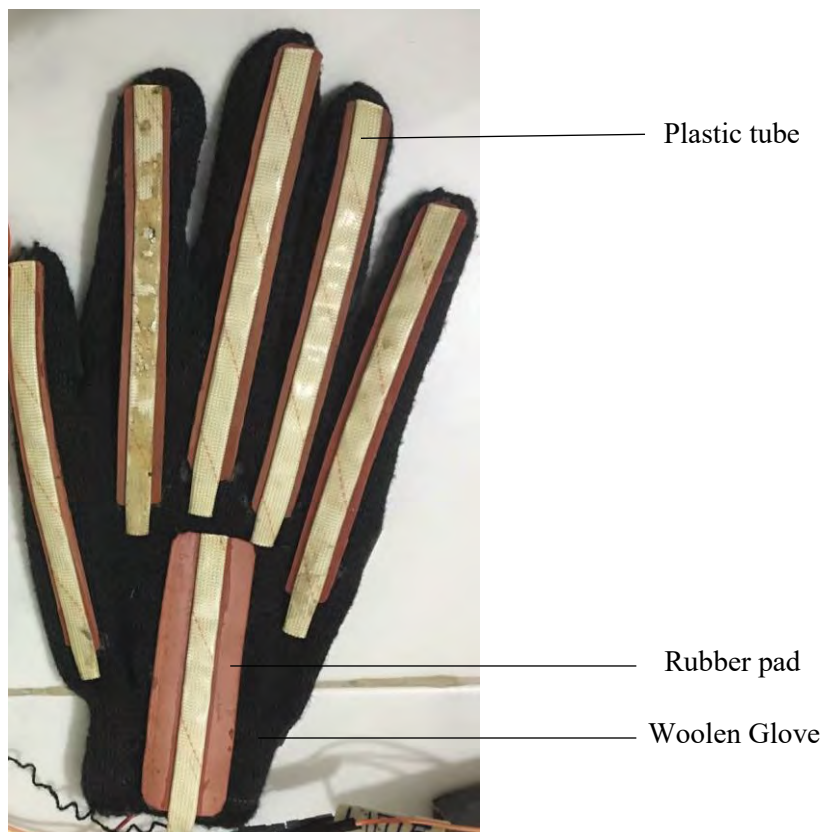


Fig. 3.1 Initial model

Design of Gloves

Throughout the entire project we used eight flex sensors in a single glove. Five flex sensors were used in five fingers- one in each finger, two sensors were used on the wrist -one above the wrist and another one on the lateral part of the wrist along the side of little finger. The last one was used along with the plum of the hand. The flex was attached with wires by help of metal soldering.

The flex sensors can detect the amount of bending. That is why we have used the flex sensors where the hand is mostly bent while showing the signs. After few analysis and observation, we came to the conclusion that the places where the hand is mostly bend are the fingers. But integrating flex sensors in the fingers only were not enough as it would not detect the accurate posture of the hand. To get the details of the posture we integrated flex in the plum of the hand. This sensor position can well detect the fist. Moreover, the sensors residing along the lateral side of the wrist can detect the bend towards the outer side of the hand. Again, the sensor over the top of the wrist can detect forward and backward bending. We size of the flex sensors along the fingers were 4.5 inch and the rest of the three were 2.2 inch. Longer flexes were used in the fingers because they cover more area with greater range of bending.

3.1.2 Initial connection with the Arduino Microcontroller

The microcontroller used of making this project is Arduino-Mega. The connection was established by jumper wires soldered by metal with the flex sensors. The wires were then inserted into a breadboard and from there further connection was established with the Arduino.

Each of the flex sensors needed individual resistors which worked as voltage divider in the circuit. The microcontroller was connected to the computer for programming and burning the program inside it. Throughout the entire project we used Arduino IDE for writing and compiling codes.

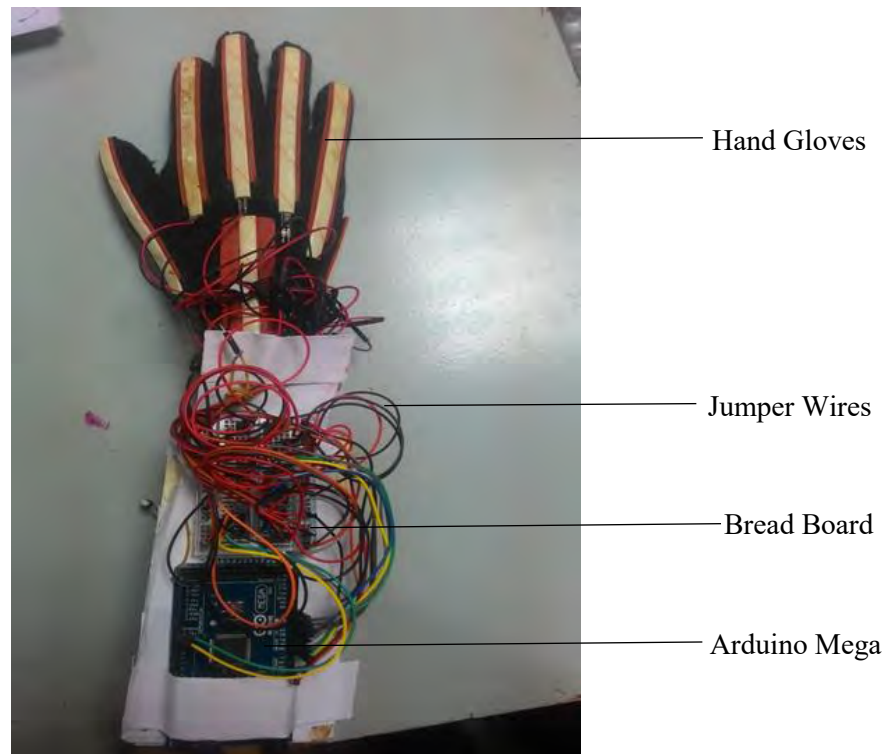


Fig. 3.2 Initial connection of the gloves with Arduino AT Mega

3.1.3 Taking Values from Flex Sensors

During the first stage of the project we worked with the raw values of the flex sensors. We measured the amount of bending of each flex while showing sign of a single alphabet and recorded those values. The raw values received from the sensors were the reference values initially. We detected the posture of the hand from these combination of raw sensor values. This was the first technique to detect the signs at initial phase.

3.1.4 Working on Orientation of Hand

Introducing Gyro Sensor

During the end of our first phase of the project, we decided to work with gyro sensor.

Gyro sensors, also known as angular rate sensors or angular velocity sensors, are devices that sense angular velocity. In simple terms, angular velocity is the change in rotational angle per unit of time. The Gyro we used **MPU6050** which has fusion of accelerometer. **Accelerometers** are electromechanical devices that sense either static or dynamic forces of acceleration. Static forces include gravity, while dynamic forces can include vibrations and movement.

Our first aim was to work with BSL (Bangla Sign Language). The sign language of Bangla comprises of several hand twisting and rotation. Thus, it was not possible to detect the Bengali signs from the flex values only. This is why we decided to integrate gyro sensor to our device. The gyro sensor was used to detect the rotation and angular position of the hand. The angular axes detected by gyro are yaw axis, roll axis and pitch axis. Pitch is the X-axis, Roll is Y-axis and Yaw is Z-axis. So, no matter in which direction the hand remains gyro can detect the exact posture.

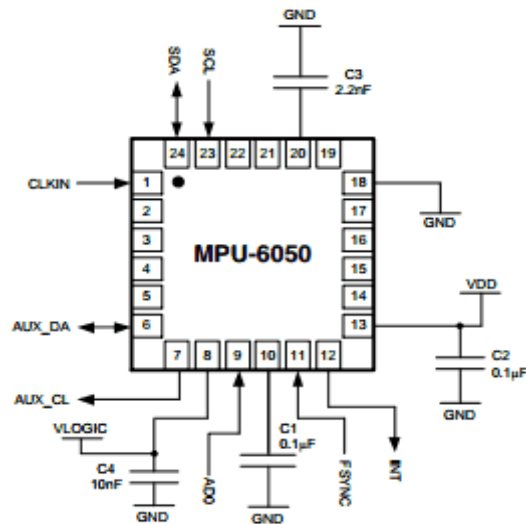


Fig. 3.3 Gyro MPU6050 and its schematic diagram

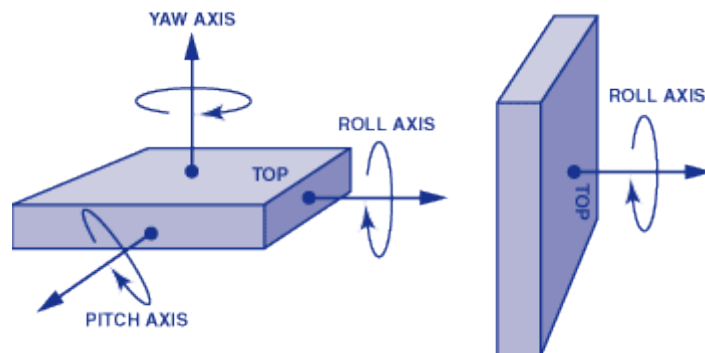


Fig. 3.4 Angular Rotation

Calibration of Gyro

The algorithm and library that we used to take the values from Gyro MPU-6050 is given by Jeff Rowberg [11]. Taking the reading from gyro as raw value is easy but it is a bit complex to take the value from DMP (“Digital motion Processor”) which is in the gyro. This DMP can be programmed with firmware and is able to do complex calculations with the sensor values. Jeff Rowberg has used the raw values from DMP and did reverse engineering to get an optimal value.

For calibrating (i.e. marking a standard scale of reading) the values we got by implementing the algorithm, we considered a position as zero and used it as reference position of all other position.

This is the initial or primary phase of our project. The flex values were the major concern of this part. In the later parts we have worked with hand orientation but did not integrate the orientation in this phase. The orientation with the gyro was an experimental approach in the fundamental stage.

3.2 Second Phase

3.2.1 Flex Calibration

In the primary stage of our project we used the raw value from the flex sensors. But this procedure was not very effective because raw value has a huge range and can fluctuate very frequently. So, we decided to calibrate the sensor values from each flex in the similar way as the gyro was calibrated. We took the raw value from each sensors and marked the highest and lowest values of it. Then we made a scaling from 1-20. That is the lowest value of each sensor while in bending condition is 1 and the highest value from relaxing position is 20. This scaling helped us greatly for making the dataset and classifying it in the right order. All the data after calibration falls in the fixed range. This also helped us to measure and classify the value of different people's hand. As we know that every people has different hand types and shape so the bending posture of all person's hand will not be always same. In this case the flex calibration comes handy. Because flex calibration bounds the value in a given scale and no matter what is the size of the hand the limit of the scaling will not be crossed. Thereby, all the classification will be same for all sort of hands. And due to this calibration technique any hand will show the wright classified answer.

3.2.2 Classification of sensor values According to Range

Firstly, for classification of the sensor values to know the desired sign we fixed a range for each sensor for every sign. Whenever we portrayed a sign then if the sensor values of the current sign fall within the range of the classified sign then we came to the decision that is this our required sign.

But this process is very long and extremely inefficient so we migrated to machine learning approach.

3.2.3 Getting Started with Machine Learning

After classification with range we moved to machine learning. Therefore, in the beginning of the second phase of building the system we opt to integrate machine learning algorithms to classify the sensor data we got from flex sensors and gyro sensor. Among many promising algorithms we chose to work with few of them. The first algorithm that we worked with was Hidden Markov Model (HMM).

A hidden Markov model (HMM) is a statistical model that can be used to describe the evolution of observable events that depend on internal factors, which are not directly observable. The hidden states form a Markov chain, and the probability distribution of the observed symbol depends on the underlying state.

In our project, we wanted to use HMM because while moving our hand from one position to another we considered each position as single state of the markov model. But as these states are not target states they are hidden from the eye of the observer. But these states are sequential and one state will lead to another. From each state i.e. hand position there is a fixed or measured probability to go to the next state. While moving from one state to another the probability to get the desired outcome of the sign increases gradually. Thus, a network is formed which can be represented as the simplest **dynamic Bayesian network**. Lastly, the chain of states in the Markov Model lead to the optimum possible output. We used python programming language for HMM algorithm.

The Machine Learning Library that we worked with next is **TensorFlow** . This is an open source library for numerical computation, specializing in machine learning applications which assists in transfer learning mechanism of Inception. **Inception** is a pre-trained Convolutional neural network (CNN) model [12]. We were going to use the Inception v3 network. We trained

the last layer of inception to learn about the dataset of the signs we have provided. After making the inception learn about our dataset it could classify the data according to given signs. We used python programming language for working with tensor flow. The dataset was made in csv (Comma Separated Value) format. In the cvs format all the placeholders are for the data and the last one is for label.

At this stage we experimented with another type of machine learning approach. This is known as **Weka Classifier**. Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from our own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes.

In weka there are many machine learning algorithms which was able to classify our dataset. But we could also implement the library from own java code. For working with weka we used java programming language.

Lastly, we also tried to classify our dataset with **Java ML**. Java has got a collection of machine learning algorithm which can classify the dataset. But a major issue was that the accuracy level or java ML was not satisfactory. For this we used java programming language.

With the machine learning approach, we tried to classify both the calibrated flex sensor values and gyro sensor values.

3.2.4 Working on Mobile Application Making

At the end of our second phase we started to work on Mobile Application. The purpose of our project was to show the signs in a mobile application and communicate via that application.

So, we started to work on an android based mobile application which can receive the data and show the desired sign. We needed a server where the classified data will retain so we chose **Firestore**. Which can be easily and effectively associated with the mobile application. The Firestore Real Time Database is a cloud-hosted database. Data is stored as JSON and synchronized in real time to every connected client.

With is much progress we moved to the third phase which is the last and concluding phase of our project.

3.3 Third Phase

3.3.1 Making New Gloves

Materials

At the beginning of this phase we made anew gloves as the previous gloves was not up to the mark. From a number of gloves type that were made from rubber, cotton, plastic and medical gloves we chose one which is a combination of rubber and cotton. The upper part is cotton and the lower past is made from rubber. The reason for choosing this gloves is that is more flexible and it capable of attaching with different hand sizes. The combination of cotton and rubber material makes it perfectly stretchable as well as gives the capability of perfect stretching.

Design

In the last phase, we made some changes in the design. We removed the rubber padding which was in between the flex sensors and the gloves. In place of that we cut the plastic tube in such a way that it will cover only the bending area of the hand. Thus we need not to use the rubber layer. The flex sensors were now more stable as they were shouldered by metal coating and also covered with glue gum so that the soldering does not break down.

The small pieces of plastic tube were attached with the gloves with the help of super glue. And the end of the flex sensors was sewed with the plastic tube and glue gum.

Due to these change the gloves became more flexible and the value we got from the gloves were more accurate.

Another notable change is that we removed the 2.2-inch flex sensors from the wrist and added two 4.5 inches' flex sensors at those places. We did this because the longer flex sensors covered more area and gave better result while classifying.



Fig. 3.3.1 The range of gloves from where one is selected

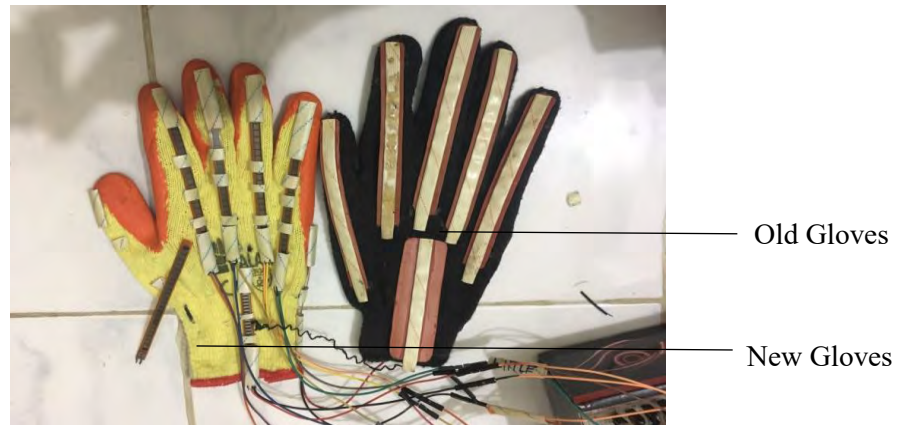


Fig. 3.5 New glove vs old glove



Fig. 3.6 The bending points where the plastic tubes are attached with glue

Integration of Force Sensors

Force sensors were integrated at this stage. The force sensors were used for various purpose. Here 3 force sensors were integrated in total. One force sensor is attached in the finger which was used to differentiate some alphabets mainly it was to differentiate between “U” and “V”. The sign of these two alphabets are almost same and they cannot be differentiated with the flex sensors. So, the threshold value from the force sensor is deciding where the alphabet is a “U” or a “V”. In “U” there is no force applied on the sensor and in “V” force is applied on the sensor.

The force sensor is placed in the upper side of the ring finger.

Force sensors as Switches

The rest of the two force sensors are used as switches. One is used to concat the letters to form a string of letters and another one is used to send the classified data to the wifi module. Both of the force sensors worked as switch for the threshold value over 600.

Removal of Gyro from the System

We decided to remove the gyro from the project as we switched from BSL (Bangla Sign language) to ASL (American Sign Language). If we could correctly classify ASL we need not use gyro sensor as ASL has lot more simple signs than BSL.

New machine Learning Algorithm

At this last phase, we chose the right and most suitable algorithm for our project. The algorithm is **KNN (K Nearest Neighbors)**. The algorithm chooses best k set of values by calculating the nearest distance from the reference datasets. Now from the k set of results the result which occur the most number of time is the most probable result. So we show that result as the outcome.

We considered the number of $k=5$ and calculated from the most occurred value. The dataset was made in csv for this case.

Making the Reference dataset and saving in SD card

The reference dataset is made in csv format and is saved in SD card. For this purpose, we integrated a SD card module with our project. The first eight values are the values from flex sensors and the last one is the label. We made an Arduino program to build the dataset which is the reference for classification.

For each alphabet we took five values. We tried to maintain difference in hand orientation and posture so that the values can get a good range and classify data in various posture. The values were saved in a text file in csv format. And during classification with KNN the values are matched with these reference values kept in the text file as csv format. From this format we can extract the label and decide the portrayed current value.

Wifi module Node MCU ESP8086 and Firebase Server

The result we got from the classification was sent to the node mcu ESP8086 which is a Wi-Fi module. This works wirelesses and can send the data to server. First we connected the node

mcu ESP8086 with the Arduino AT mega via RX TX where node mcu works as slave and Arduino works as master.

The classified data is sent to node mcu ESP8086 8086 and it send the data to firebase server in real-time. The fire base server of Google can save real time data. The classified data that is the labels we got are push to the database and then are retried later by the mobile application.

3.3.2 Final Mobile Applications

Finally, two mobile applications are made. One for learning the sign languages and another for messaging and communication.

The Messaging Application

The messaging app show the message a deaf person wants to show via the gloves. If he shows a sign of alphabet, he need to show the consecutive alphabets to make a word. The word is passed if a force sensor switch is pressed and the word goes to the node mcu and the node mcu sends it to the firebase server. The mobile application can detect that a new word is available so it shows the word in the application as a list and also as a text in another activity. If the person shows some consecutive words then the application will show the sentence the person wants to show. This is how the message passing application works.

The Learning Application

The learning application is a little bit different than the messaging application. It show the picture of an English Alphabet's ASL sign. And if the person who is learning the sign does the same posture by his hand while wearing the gloves and then presses the force sensor switch to send the value to the server then the application will show if the posture is correct or not. If the sign he made is correct then a "right" sign will be shown but if the sign is wrong then "cross" symbol will be shown. If the sign is correct then the application will automatically move to the next sign if tapped on the screen and the tutorial part will be repeated. But if the person shows wrong sign then the application will show the same alphabet until he learns and shows the correct sign.

This is how the application teaches a person the ASL sign language from A-Z.

Chapter 4: Components

4.1 Hand Glove

One of the most basic and essential component in our system is the hand gloves. The hand gloves which we have selected is made from woolen fabric and rubber. It has a standard size so that most of the people's hands having average shape and size fits inside it.

Plastic tube strips were attached on the hand gloves with glue so that we could set the flex and force sensors in place.



Fig. 4.1 Hand gloves with plastic tube

4.2 Firebase

Firebase is a technology that permits you to make applications with no server-side programming so that development turns out to be quicker and easier. It works as realtime database as it stores and sync data between users and devices in realtime using a cloud-hosted, noSQL database. Updated data syncs across connected devices in milliseconds, and data remains available if your app goes offline, providing a great user experience regardless of network connectivity. It also has cloud storage and hosting functionalities.

In our project, the Node MCU sends data to firebase database and these data are retrieved via android application which is also connected to the same firebase database.

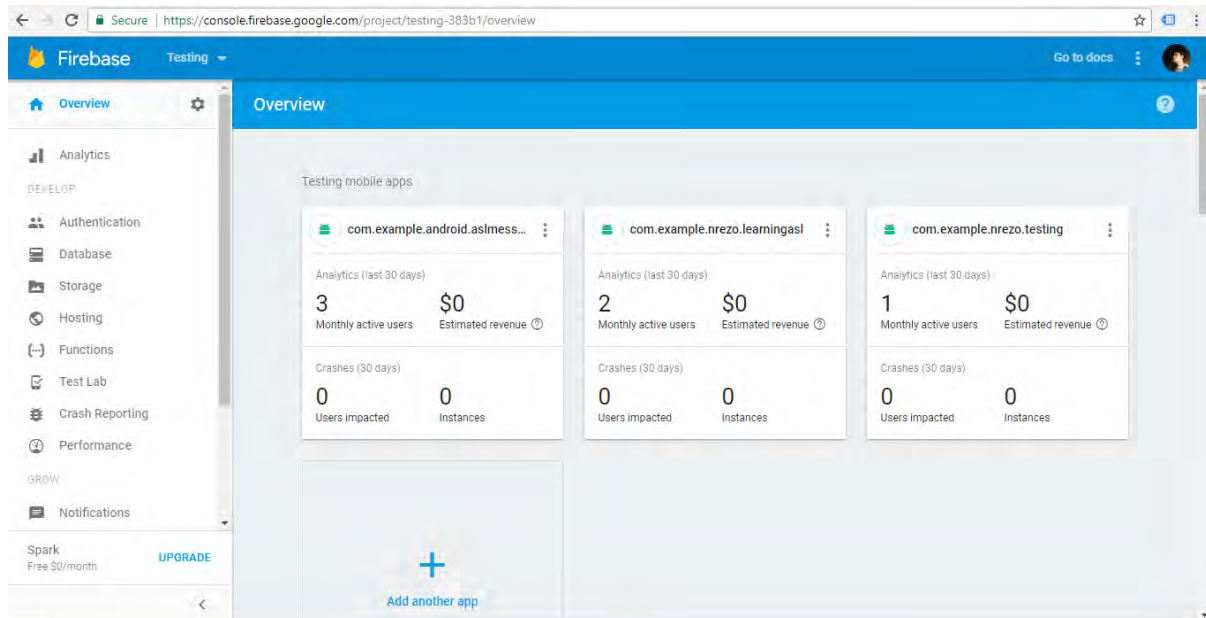


Fig. 4.2 Firebase Interface

4.3 IDE (Integrated Development Environment)

We have used multiple IDEs to write and compile our codes. Typically, an IDE is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger. The IDEs that we have used in our projects are described below:

4.3.1 Arduino IDE

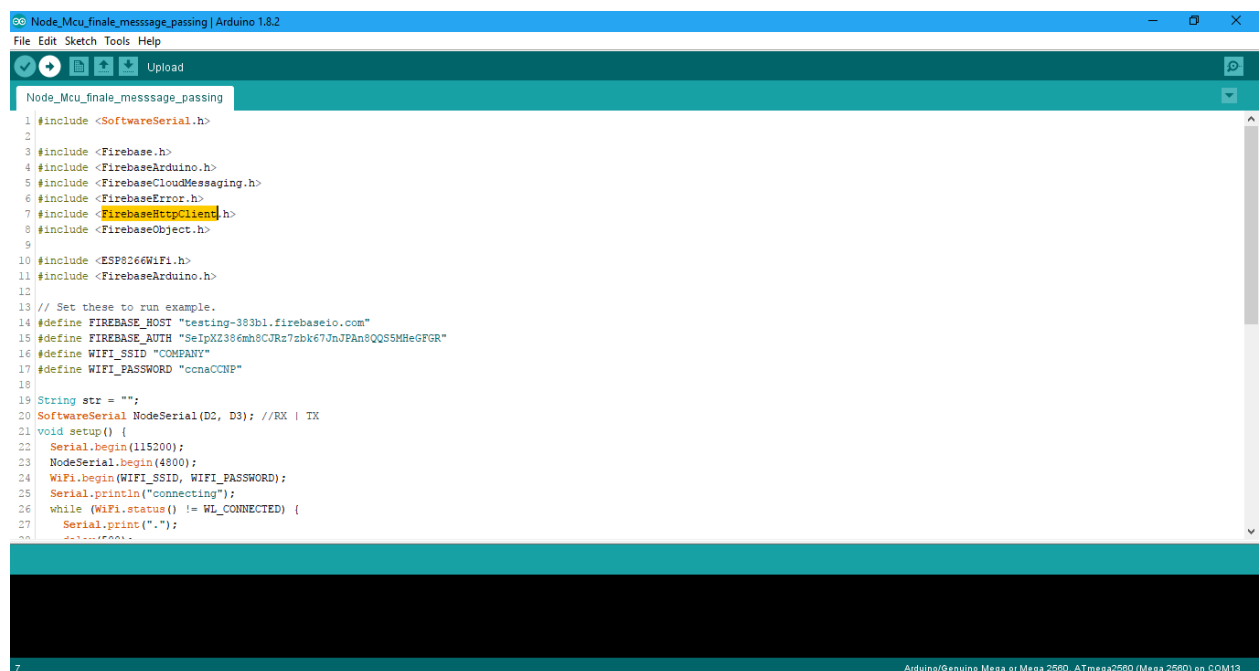
The **Arduino Software (IDE)** is a open-source IDE which comprises of many libraries. As it is open source, it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

For writing code on Arduino Mega board, we have used several libraries. The SD library is used for using the methods from the buildin SD library of Arduino. The SDI library is used for communicating with the SD card module. With the help of these libraries we can read and write

in sd card. The SoftwareSerial provides some extra libraries which helps to communicate between arduino board and node mcu. The serial communication is established by TX-RX pins.

There are three ways to write in Node MCU ESP8266. The main format is to code with lua based scripting language. But we have used Arduino to write on Node MCU, which is not the conventional procedure. But we have used this for our convenience. In case of Node MCU we have also used many libraries for various purposes. In the connection between Arduino Mega and Node MCU the Arduino Mega acts as master and Node MCU acts as slave.

The SoftwareSerial library is also used here same like Arduino Mega. This is used for serial communication between Node MCU and the Arduino Board. The ESP8266WiFi library is the basic library of Node MCU. It helps in the routing process. The Wi-Fi library for ESP8266 has been developed based on ESP8266 SDK, using naming convention and overall functionality philosophy of Arduino WiFi library. For communication with firebase server there are certain libraries. These are Firebase, FirebaseArduino, FirebaseCloudMessaging, FirebaseError, FirebaseHttpClient, FirebaseObject. Through these libraries Node MCU send data to firebase in real time. These libraries also help to detect errors generated while sending data. Firebase Cloud Messaging (FCM) is a cross-platform messaging solution that reliably deliver messages at no cost.



```
98 Node_Mcu_finale_message_passing | Arduino 1.8.2
File Edit Sketch Tools Help
Upload
Node_Mcu_finale_message_passing
1 #include <SoftwareSerial.h>
2
3 #include <Firebase.h>
4 #include <FirebaseArduino.h>
5 #include <FirebaseCloudMessaging.h>
6 #include <FirebaseError.h>
7 #include <FirebaseHttpClient.h>
8 #include <FirebaseObject.h>
9
10 #include <ESP8266WiFi.h>
11 #include <FirebaseArduino.h>
12
13 // Set these to run example.
14 #define FIREBASE_HOST "testing-383b1.firebaseio.com"
15 #define FIREBASE_AUTH "SeIpXZ38mh8CJRz7zbr67JnJPan8QQ5SMHeGFGR"
16 #define WIFI_SSID "COMPANY"
17 #define WIFI_PASSWORD "ccnaCCNP"
18
19 String str = "";
20 SoftwareSerial NodeSerial(D2, D3); //RX | TX
21
22 void setup() {
23   Serial.begin(115200);
24   NodeSerial.begin(4800);
25   WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
26   Serial.println("connecting");
27   while (WiFi.status() != WL_CONNECTED) {
28     Serial.print(".");
29     delay(500);
30   }
31   Serial.println();
32 }
```

Fig. 4.3 Arduino IDE

4.3.2 Android Studio IDE

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built based on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) which was used as primary IDE for native Android application development.

Both of the Mobile applications made for this project are created in Android Studio.

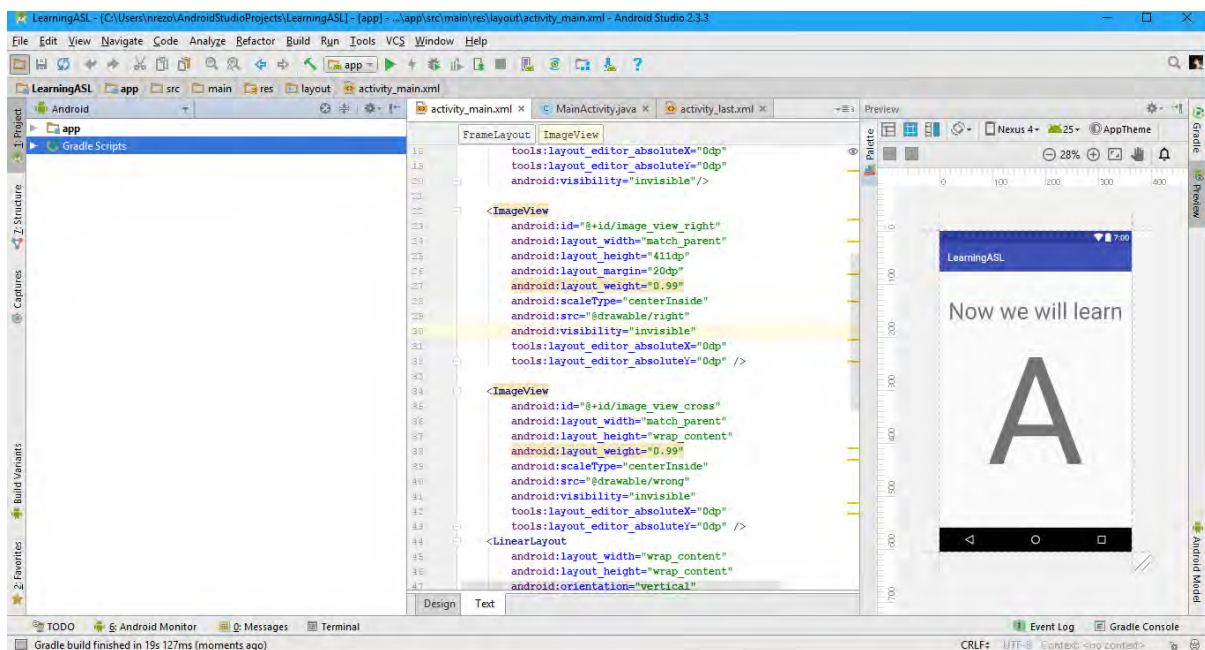


Fig. 4.4 Android Studio

4.4 Electronics

4.4.1 Arduino Mega

The Arduino Mega we used is a microcontroller development board based on the ATmega1280. It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It has everything, which is needed to support the microcontroller. It can be powered by simply connect it to a computer with a USB cable or an AC-to-DC adapter. The Mega is compatible with most shields designed for the Arduino. It's operating voltage is 5V, input voltage recommended 7-12V, input voltage limit is 6-20V, DC current per I/O pin is 40 mA and for 3.3V pin 50 mA, flash memory is 128

KB of which 4 KB used by bootloader, SRAM is 8KB, EEPROM is 4 KB and clock speed of 16 MHz [11].



Fig. 4.5 Arduino Mega 2560

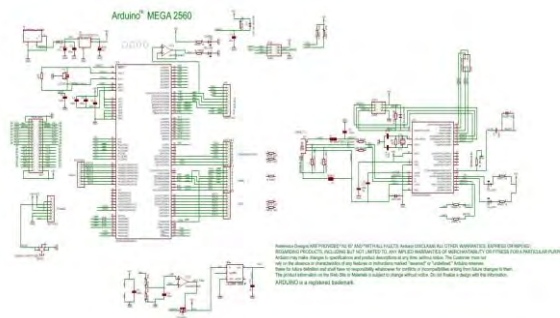


Fig. 4.6 Schematic Diagram of Arduino Mega 2560

4.4.2 Flex sensor

Flex Sensor is one of the most important components we have used in our device. It takes Angle Displacement Measurement from the flex whenever it bends. It is commonly used in fields like Robotics, Gaming (Virtual Motion), Medical Devices, Computer Peripherals, Musical Instruments, Physical Therapy, Simple Construction, Low Profile etc. It also has two types of shape. One is 2.2 inches and other one is 4.5 inches. It has a temperature limit, which is in between -35°C to $+80^{\circ}\text{C}$. Flat resistance is approximately 25K Ohms and the resistance tolerance is $\pm 30\%$. It also has Bend Resistance Range of 45K to 125K Ohms (depending on bend radius). The Life Cycle of these sensor is greater than 1 million [13].



Fig. 4.7 Flex sensor

4.4.3 Force Sensor Resistor

In our project, we have used FSR 400 Series force sensor. Exhibits a decrease in resistance with increase in force applied these robust polymer thick film (PTF) devices to the surface of the sensor. This force sensitivity is optimized for using in human touch control of electronic devices such as automotive electronics, robotics applications, industrial and medical systems. It has six different models and we used FSR® 4005mm Circle x 38mm. Its' actuation Force is ~0.2N min, Force Sensitivity Range ~0.2N – 20N, Force Resolution is Continuous, Force Repeatability Single Part +/- 2%, Non-Actuated Resistance is greater than 10M ohms, Hysteresis +10% Average $(RF+ - RF-)/RF+$, Device Rise Time is less than 3 Microseconds [14].



Fig. 4.8 Force Sensor

4.4.4 Micro SD Breakout Board

When it comes in need of a portable storage facility then micro SD card breakout board is the component we use. It helps to log data. They are strictly 3.3V devices and during writing to the card the power, drawing can be high, up to 100mA or more. That means we have to strictly provide 3.3V power supply for the card. Also, have 3.3V logic to interface the pins. SD cards are sensitive about the interface pins. There are two types of interfacing with SD cards. One is **SPI mode** and other one is **SDIO mode**. SDIO mode is faster than SPI but it is more complex and requires signing nondisclosure documents. There is an onboard ultra-low dropout regulator in the board that will convert voltages from 3.3V-6V down to ~3.3V. Also a level shifter can convert the interface logic from 3.3V-5V to 3.3V.

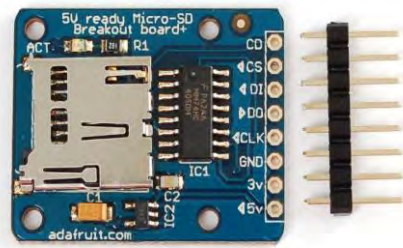


Fig. 4.9 SD Card Module

4.4.5 NodeMCU

Node MCU is a Low-power and highly-integrated Wi-Fi solution. A minimal of 7 external components with wide temperature range between -40°C to $+125^{\circ}\text{C}$. It is ESP8285 - ESP8266 embedded with 8 Mbit flash. Node MCU is an eLua based firmware for the ESP8266 WiFi SOC from Espressif. The NodeMCU firmware is a companion project to the popular Node MCU dev kits, ready-made open source development boards with ESP8266-12E chips. This component is Engineered for mobile devices, wearable electronics and the Internet of Things (IoT) applications, ESP8266EX achieves low power consumption with a combination of several proprietary technologies. The power saving architecture is featured with three modes of operation- active mode, sleep mode and deep sleep mode that allows battery powered designs to run longer [15].



Fig. 4.10 NodeMCU

Chapter 5: Communication

5.1 Block Diagram

The block diagram of our project is shown below-

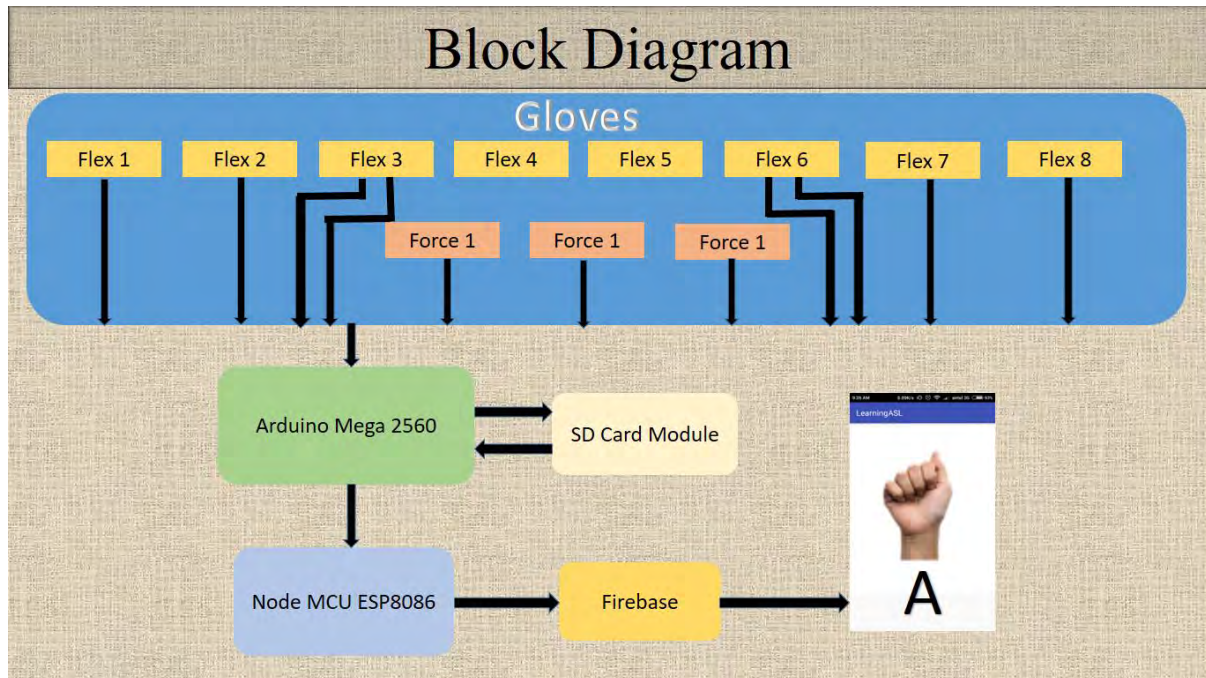


Fig. 5.1 Block diagram of the system

5.2 Flex Sensor and Arduino

Flex Sensor works as a voltage divider circuit with the Arduino. One leg of the sensor is connected to the 5v and another leg is connected with the resistor. The other leg of resistor is connected with the ground of the Arduino. One wire connects the common node of resistor and the Flex sensor to the Arduino Analog pin. Whenever the sensor is bend, its resistor varies. Depending on that, the divided voltage also varies. Arduino collects the amount of voltage it receives depending on the bending of flex sensor. In our system, we have used 8 flex sensors. Which are connected to the Arduino board from pin A0-A7. A0 is connected to the flex, which are set to the palm A1 in the thumb, A2 in the index, A3 in the middle, A4 in the ring, A5 in the little, A6 in the side and A7 in the upper side of the wrist. We have used 22k resistor for all flex sensor in our gloves.

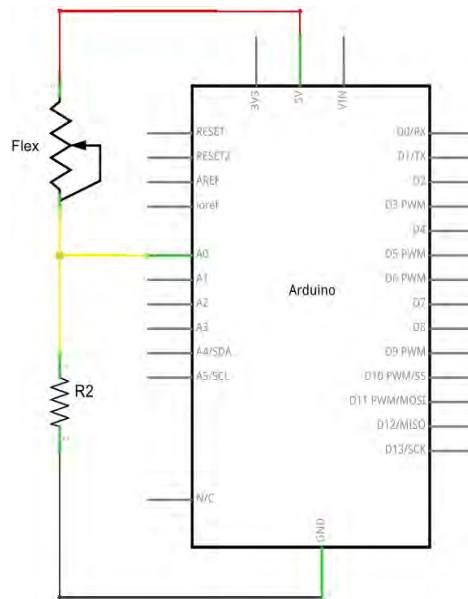
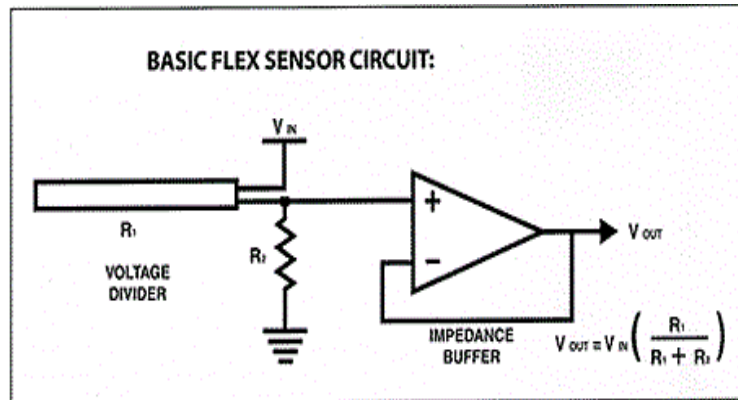


Fig. 5.2 Basic Flex Sensor Circuit.

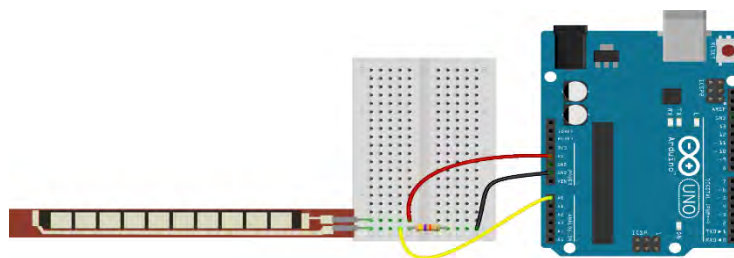


Fig. 5.3 Flex Sensor Circuit Connection with Arduino

5.3 Force Sensor and Arduino

Force sensor also works as a voltage divider circuit with the Arduino. The connections are also more or less similar to the flex sensor. When some force is exerted on the tip of force sensor the resistance of the sensor changes so the voltage that was working in the similar node with the fix resistor changes. Here we also collect the voltage from same joining node of fix

resistance and ground to the analog pin of the Arduino. We have used three force sensor in our system. All of them has their different purposes. One of them is used to differentiate between similar sign like “U” and “V”. Another of them is used to concatenate between letters. In addition, another the last force sensor is simply used to push the data to the NodeMcu.

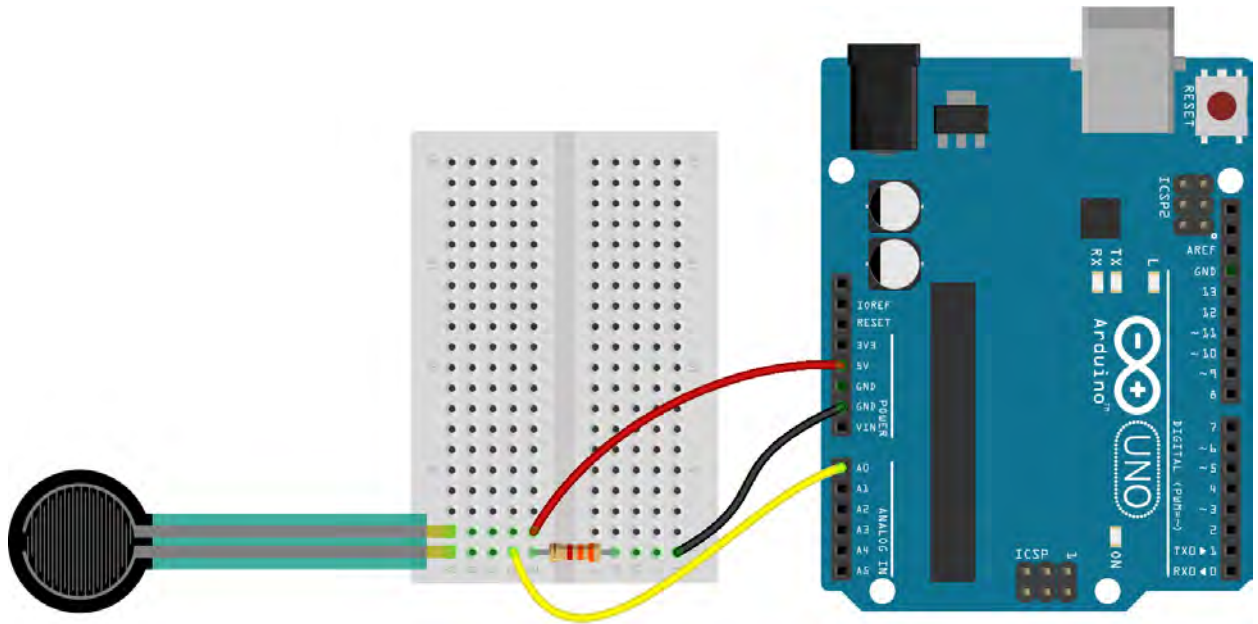


Fig. 5.4 Force Sensor Circuit Connection with Arduino

One leg of sensor is connected to the 5 volt of the Arduino. Another leg is connected with the analogue pin of Arduino and a fix resistor. Another part of the fix resistor is connected to the ground. Three of the force resistor is connected to A8, A9, and A10 pin respectively.

5.4 Arduino and SD Card Module

We have used one SD card module to save and retrieve our reference data. When we create data set we save data to the SD card and when we actually predict data through our classifier we need to retrieve all the data from the card and compare. The connection pin is as follows:

Digital pin 50 is hooked up with MISO pin of the SD card module then digital pin 51 to MOSI, pin 52 to SCK and pin 53 to CS of the SD card module. The ground pin shares the same ground with all other sensors and the 3.3v pin is attached to the 3.3V pin of Arduino. We have simply used the SD library of Arduino.

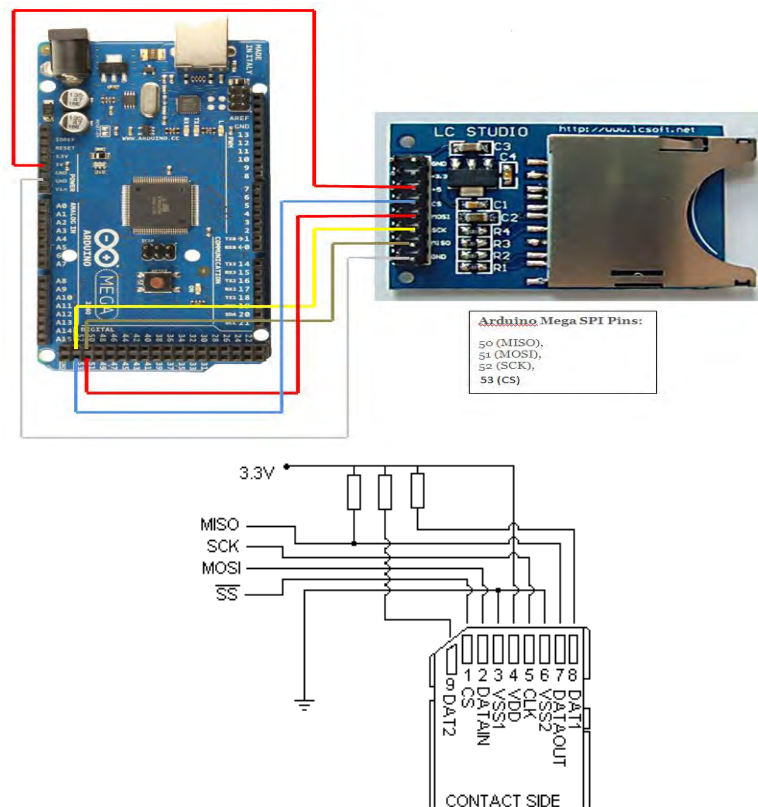


Fig. 5.5 SD Card Module Circuit Connection with Arduino

5.5 Arduino and NodeMcu

After classification of the projected data we send the calculated label/letter/word to the nodeMcu(ESP8266 12E). This is an IOT device which is used to send data to the firebase server. The communication between the Arduino and the NodeMcu is simply a serial communication. We have used Arduino's softwareSerial library to establish the connection. Then we have declared the D2, D3 pin of NodeMcu as RX,Tx pin. On the other hand, for Arduino we declared digital pin (2, 3) for this serial communication as Rx and Tx. Here the NodeMcu works as a slave, which only receives the data from Arduino and forward the data to firebase. We have used 48000 baud rate for the serial communication between the NodeMcu and the Arduino. Declared Rx pin of Arduino is connected to the declared Tx pin of the NodeMcu. And the declared Tx pin of the Arduino is connected to the declared Rx pin of the

NodeMcu. Apart from that Vin of the NodeMcu is Connected to the Vin Of theArduino.

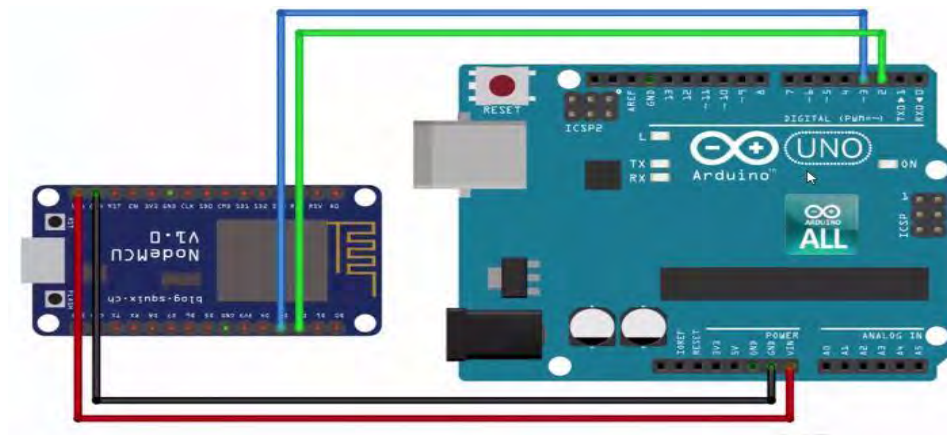


Fig. 5.6 NodeMcu Circuit connection with Arduino

5.6 NodeMcu to Firebase

For this communication part we have used the Arduino Firebase library. In total, there are three ways to use one NodeMcu. First, one is using the built in firmware by AT commands. Second, one is using lua scripting language and third one is using it through Arduino Ide. For the preparation of our NodeMcu we have chosen the third option because the Arduino environment was already familiar with us and this method is somewhat easier than other methods. Besides using this method, it was easier for us to use firebase. FireBase-Arduino library gives access to some useful push/get method, which helps to push the data to the database and retrieve the data from the database. For the training application, we send the predicted five letters from the Arduino to the nodeMcu and we set the value to the firebase using the token “Letter”. For the use of the message passing app we needed to use two different token in firebase one (word) to set the latest word (sentence) and another is to push the newly generated word.

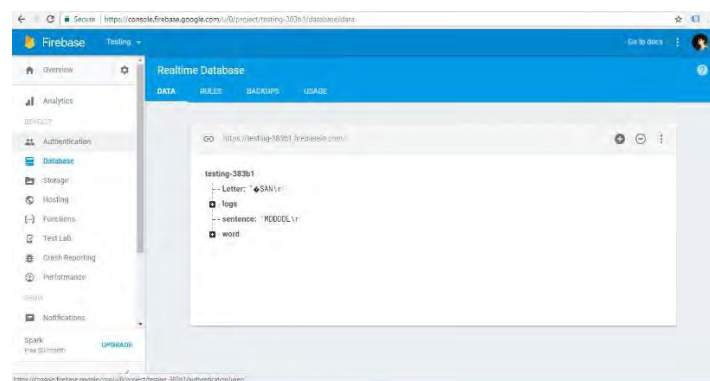


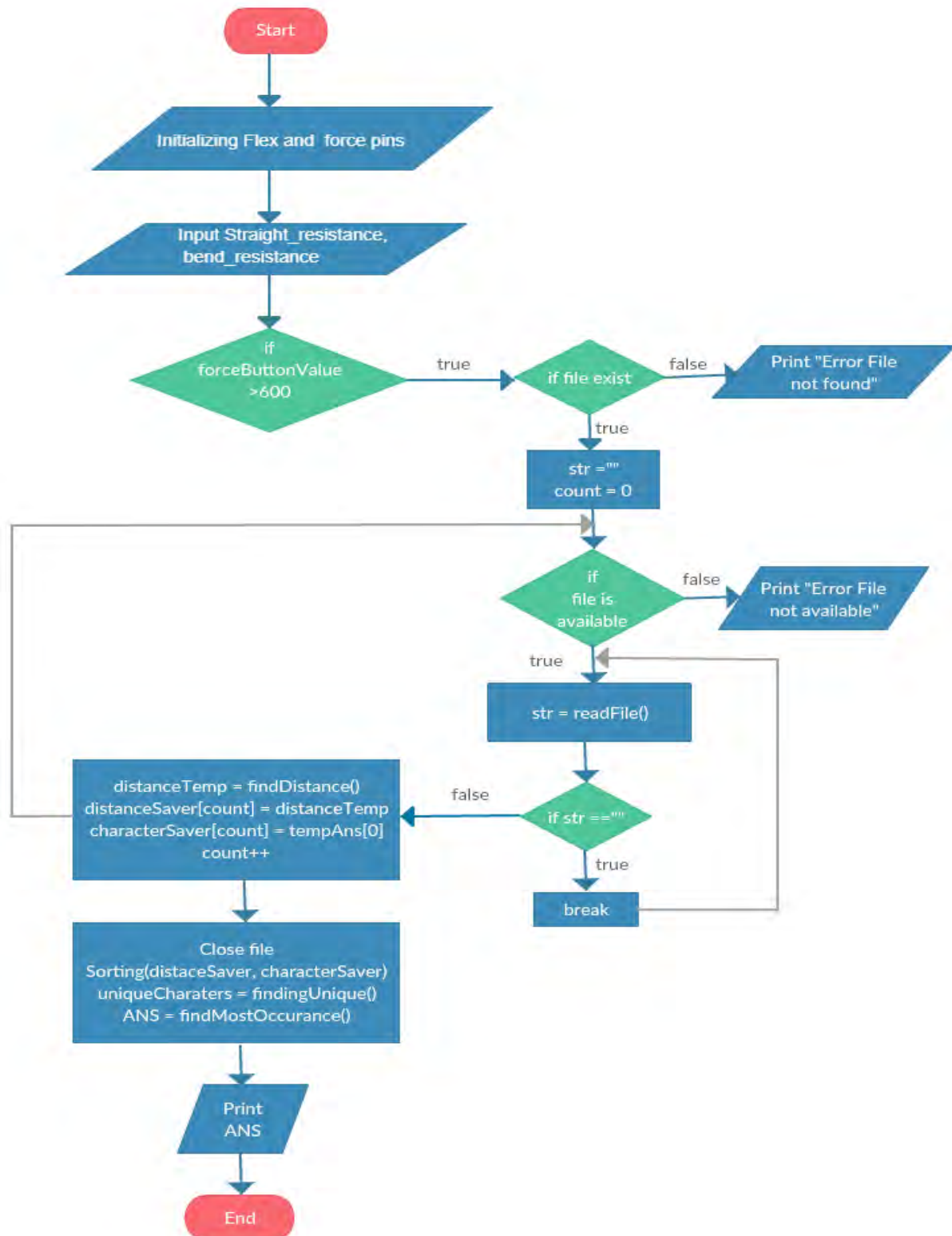
Fig. 5.7 Firebase

5.7 Firebase to Android

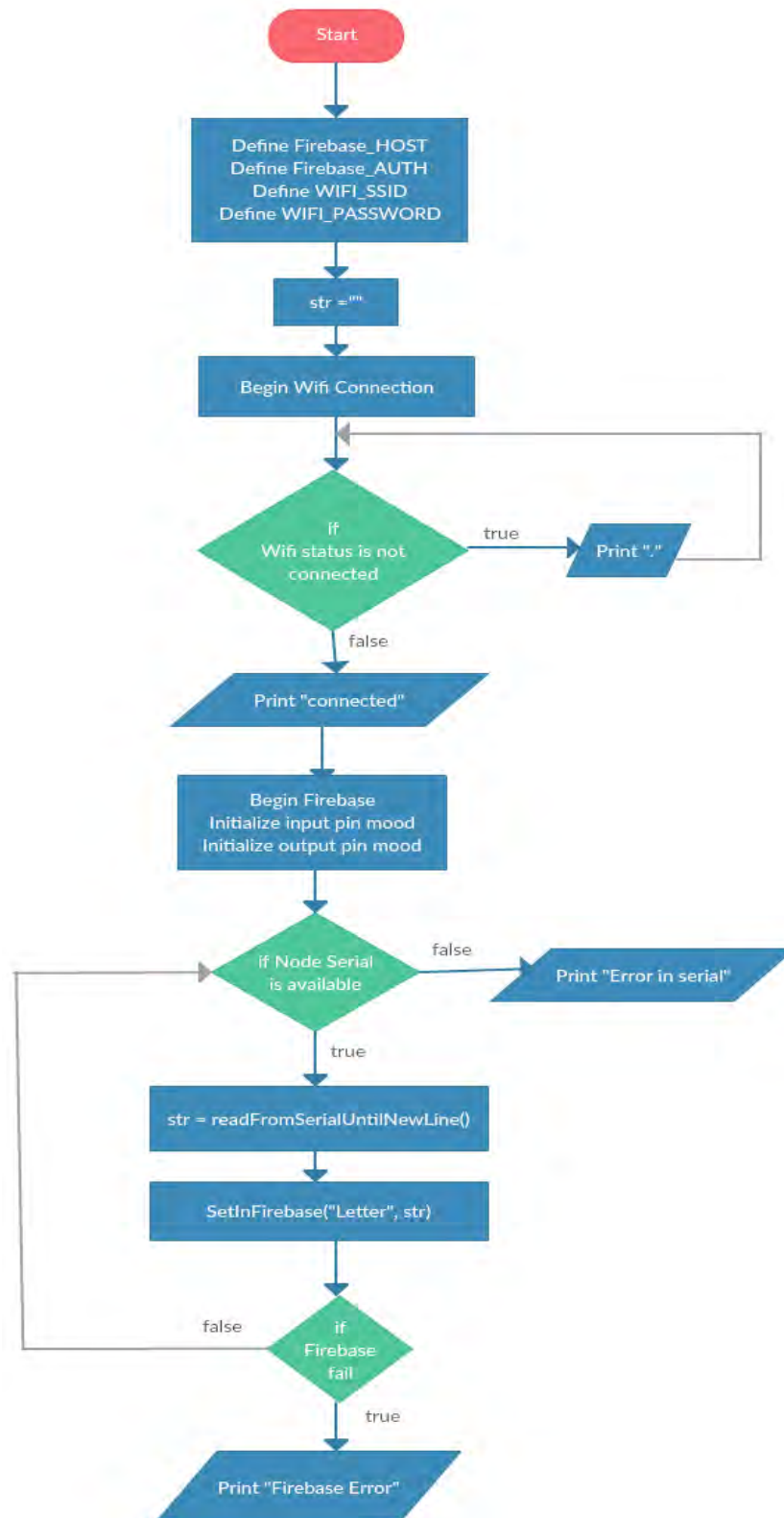
Here we have also used the built in firebase library for android to get the pushed or set data at real time. Then we used our logic to process the data in our app. whenever a new child is added to the firebase the `valueAddedListener` is called and the mobile application get to know about the new pushed data. Depending on the data, the mobile app takes his decision.

Chapter 6: Algorithm

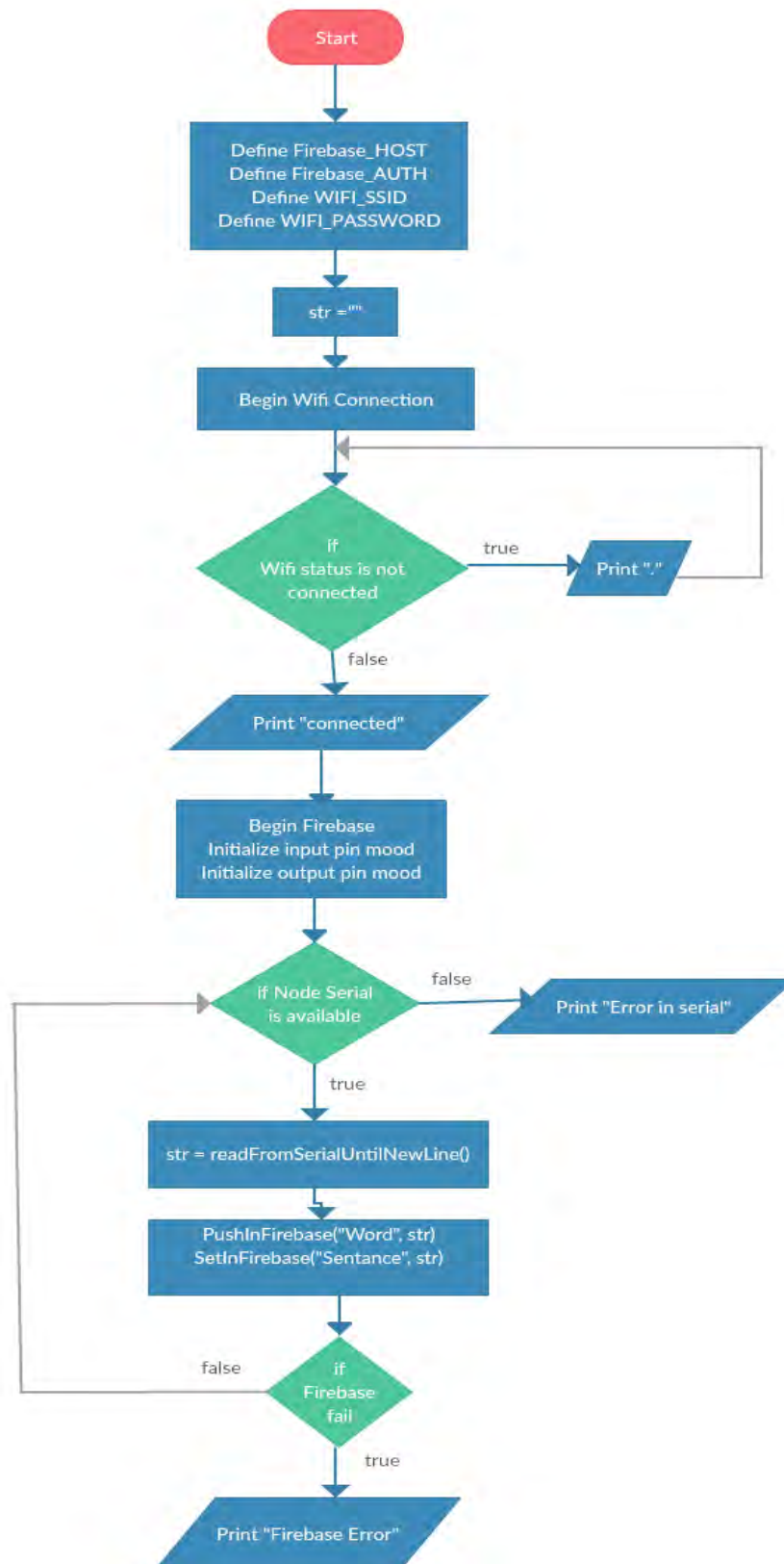
6.1 Reading CSV



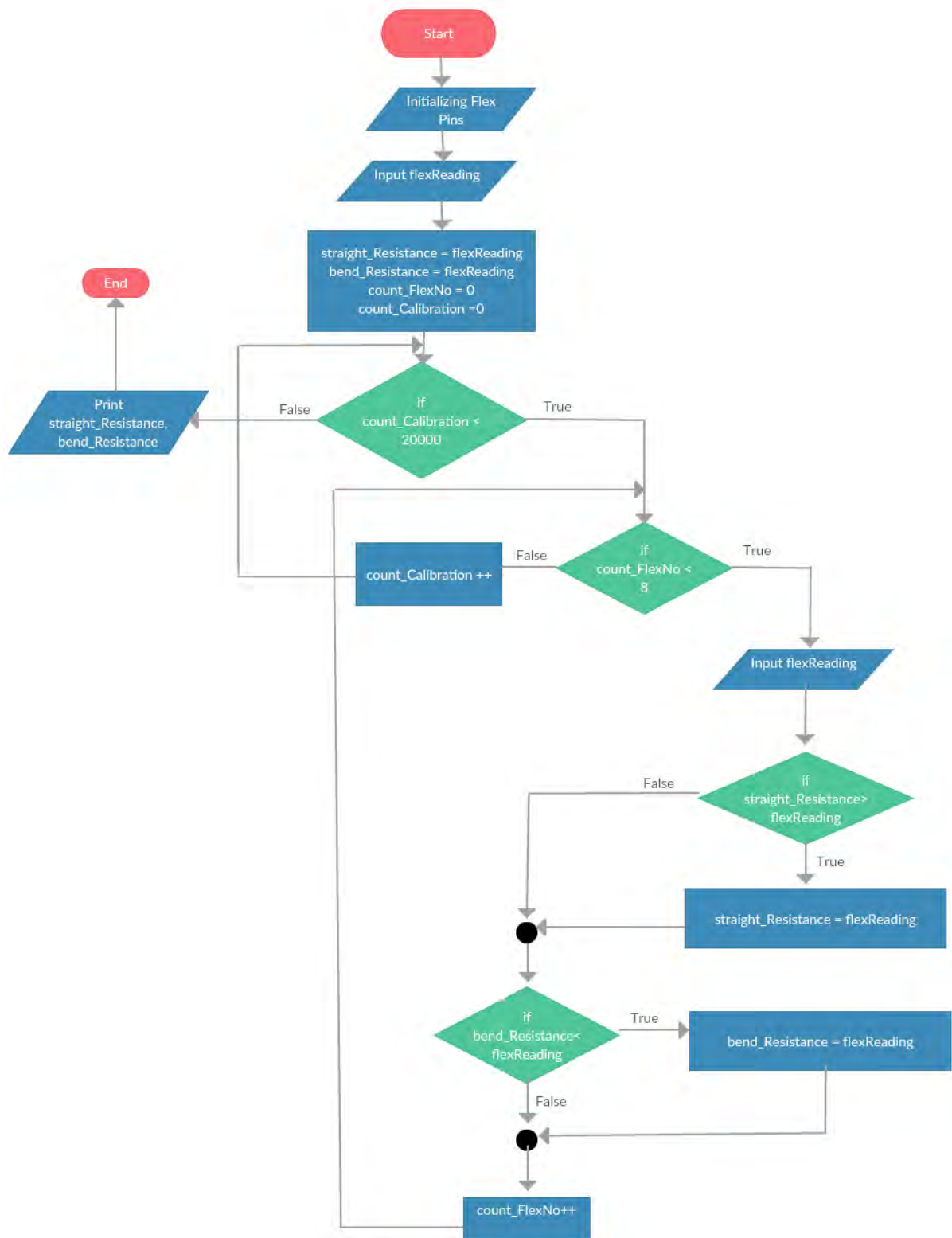
6.2 Node MCU ASL



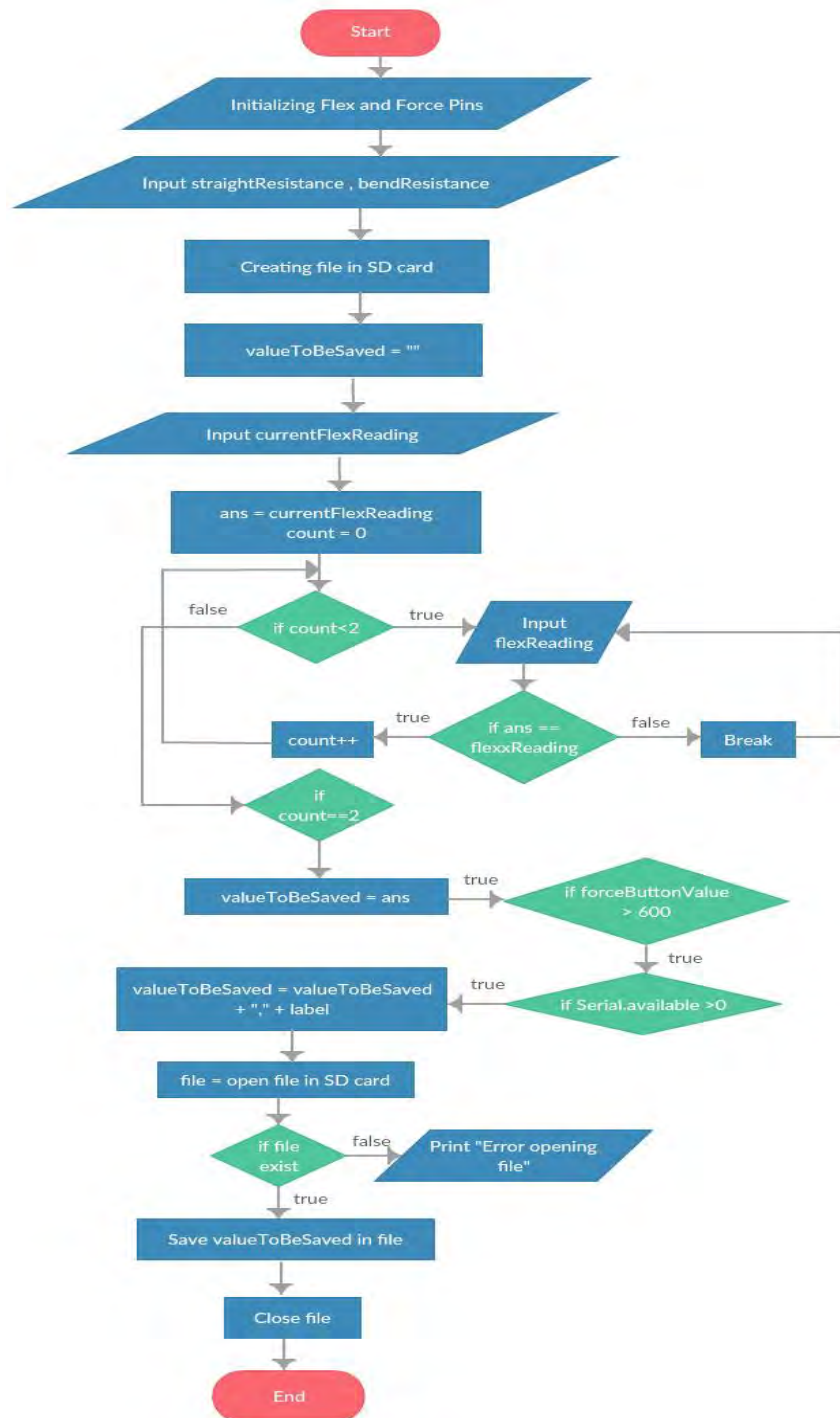
6.3 Node MCU Message Passing



6.4 Only Calibration



6.5 Saving Creating CSV



Chapter 7: Experiment

7.1 Taking raw values from flex sensors

In the initial stage of the project we took raw values from the sensors and tried to find out the values from each sensors in a given position showing the required sign. But the value difference was very unpredictable and unstable and also the range was very big. Thus working with the raw flex sensor value was not an effective process. To overcome this difficulty, we moved to fixing scale for every sensor. This is known as calibration. Taking raw value was an unsuccessful approach.

7.2 Calibrating the flex sensors

We migrated to flex calibration after facing difficulty with raw sensor values. We took a range from 1 to 20. For every flex sensor the highest bending point is 20 which is the position where the flex is not bent at all and the lowest bending point is 1 which is the position where the flex is maximum bent. Now whatever value a person shows will be in between 1-20. Through flex calibration the value became much more stable and classification became easier. The flex calibration does retain until the last of the project. Flex sensor calibration was hence successful.

7.3 Gyro integration and calibration

We used gyro to detect angular rotation and implement the BSL. We also thought that the value of gyro will be used to form the states of hidden markov model. But we then decided to switch to ASL so we did not need gyro sensor anymore as it made the classification much more complex. Though gyro worked pretty well but we did not continue with gyro till the end. So working with gyro was not completely successful.

7.4 HMM implementation

We worked with machine learning algorithm like HMM to classify the data received from Gyro sensors. This algorithm was used to detect states of the hand orientation and thus supposed to make a bayesian network of possible states and one state lead to another making it more probable of getting the desired value. But we omitted BSL and we did not need continuous

hand orientation movements. That is why we left HMM and gyro both at the same phase. Thus using HMM was not successful.

7.5 Tensor Flow Implementations

We used tensor flow to classify the datasets we made from flex movement. But we could not continue with that. At first we thought that we would write our own server scripts and the python code from tensor flow can be integrated with it. But then we started to work with firebase server. And we also made java based android application. So we could no more work with python codes. So we needed to leave the tensor flow library. Though tensor flow was an efficient one but it was not of our work. Thus using tensor flow was unsuccessful.

7.6 Weka Classifier and JavaML

We initially wanted to use weka and javaML library after being failed with python. The classifier was supposed to be useful but it was tough for the arduino to send an huge string and that would be classified in the mobile end. So we started classifying in the arduino end and did not need any java based machine learning classification algorithm. We can say that it was also a failed approach.

7.7 KNN algorithm implementation

Last of all machine learning algorithm we used the KNN algorithm at the arduino end. This algorithm was more successful than any other algorithms as it was implemented on the arduino side. But it had some problems because the accuracy level of the values was not satisfactory. It could not detect or differentiate many alphabets. For example: U and V both gave the result U. A, S and T gave the same value S. C, E, O values could not be differentiated precisely. Same goes for A and M. Other than that the KNN algorithm could detect the projected value. So it was 70% successful. And we continued to work with this algorithm.

Chapter 8: Result

8.1 Calibration

For making the dataset at first we needed to calibrate the raw values from flex sensors. Different sensors had different upper and lower bending limits according to the raw values of the flex sensors. So to maintain a fixed range of values and a fixed upper and lower bending limit we calibrated the flexes. The raw values were then converted into calibrated values. Below we have shown the BEND_RESISTANCE (Highest limit) and the STRAIGHT_RESISTANCE (lowest limit). These are the raw values from the flex which have been converted from 1-20, where 20 being the highest and 1 being the lowest. It is also important to mention that the calibration of the raw values differ from hand to hand i.e. different person have different value of calibration.

```
-----  
-----  
STRAIGHT_RESISTANCE[0] = 55.00;  
BEND_RESISTANCE[0] = 303.00;  
  
STRAIGHT_RESISTANCE[1] = 396.00;  
BEND_RESISTANCE[1] = 690.00;  
  
STRAIGHT_RESISTANCE[2] = 339.00;  
BEND_RESISTANCE[2] = 605.00;  
  
STRAIGHT_RESISTANCE[3] = 411.00;  
BEND_RESISTANCE[3] = 663.00;  
  
STRAIGHT_RESISTANCE[4] = 343.00;  
BEND_RESISTANCE[4] = 644.00;  
  
STRAIGHT_RESISTANCE[5] = 487.00;  
BEND_RESISTANCE[5] = 658.00;  
  
STRAIGHT_RESISTANCE[6] = 480.00;  
BEND_RESISTANCE[6] = 678.00;  
  
STRAIGHT_RESISTANCE[7] = 557.00;  
BEND_RESISTANCE[7] = 695.00;
```

8.2 Making of the data set

After the calibration was made the calibrated values from the sensor were put in the piece of code which is used to build the dataset. While displaying a sign, reading from each sensor are

taken as references and the collection of all eight calibrated sensor values during that sign is the cumulative reference value of that sign. This is how the reference data for each signs are taken. For one sign 5 datasets are recorded. Similarly, for 26 alphabets 5 datasets are taken. Altogether there are $(26 \times 5) = 130$ reference datasets to be compared with.

12	8	6	5	4	6	14	14	151	A
3	6	17	17	17	14	16	14	166	B
9	9	12	9	10	12	15	13	155	C
5	8	16	7	7	7	14	12	154	D
4	3	4	6	6	7	14	14	143	E
9	8	7	17	17	16	16	13	162	F
11	9	16	4	3	5	4	10	143	G
4	6	17	16	5	7	5	11	150	H
9	6	3	3	4	14	14	15	158	I
7	7	5	5	4	14	5	14	148	J
5	12	16	16	5	6	13	16	164	K
12	15	17	4	4	5	14	13	152	L
2	5	6	10	10	6	12	15	147	M
4	6	9	10	4	5	12	14	152	N
8	6	7	7	7	8	14	13	158	O
9	13	17	12	13	11	9	1	154	P
10	14	13	6	5	6	9	1	143	Q
3	7	16	14	5	6	13	13	155	R
11	4	2	3	3	4	12	14	120	S
6	9	6	7	5	5	14	13	113	T
3	6	16	16	4	5	14	12	120	U
3	7	17	16	6	7	13	11	120	V
2	7	17	17	16	6	14	12	122	W
5	2	11	5	6	7	12	12	114	X
15	17	3	4	4	14	11	14	121	Y
8	9	16	5	5	6	7	8	131	Z
12	8	1	1	1	2	13	16	131	A
4	5	16	16	17	14	16	17	153	B
10	8	13	9	10	12	16	18	144	C
5	7	16	5	5	5	15	17	137	D
4	2	3	5	4	5	14	17	127	E
10	8	5	17	17	15	16	17	138	F
8	7	16	6	5	5	3	14	135	G
5	6	16	16	5	7	5	15	139	H
9	8	4	3	4	13	14	16	144	I
11	6	6	5	5	14	4	14	139	J
6	10	16	15	5	4	14	16	147	K
11	14	16	6	5	5	15	13	142	L
3	4	6	9	9	5	13	13	140	M
5	6	9	9	4	5	12	14	150	N
8	6	7	6	7	7	16	14	137	O
8	12	17	11	11	8	8	1	151	P
9	14	13	5	4	4	9	2	143	Q
3	6	15	14	2	4	13	14	141	R
8	5	1	3	3	4	12	14	53	S
8	9	7	7	5	5	13	13	143	T
3	6	16	16	4	4	14	13	151	U
3	6	16	15	5	6	14	12	146	V
3	7	17	17	16	5	14	12	150	W
4	3	13	5	5	6	14	14	146	X
11	12	3	4	4	14	14	15	147	Y
6	7	16	4	4	5	10	11	145	Z

Fig. 8.1 Dataset

Some part of the dataset s shown here. The dataset is stored in the SD card as csv format. The First eight column shows the Flex sensor reading, the 9th column shows the value from the force sensor in the finger and the last column denotes the label of the data.

8.3 Finding Result from the dataset

The hardest part was to find the result of a displayed sign from the dataset references. While a person projected his hand to show a sign the classification algorithm (KNN) will find the nearest values that matches from reference dataset. As we have taken value of $K=5$ so there will be 5 most closest distances from the shown value to the all the reference values. Now, among those 5 nearest distances the one with the most common label will be selected as the result.

In the given section we will discuss about the first five alphabets and there corresponding results accuracy.

We, have used some colored marks which has symbolic meanings. The “Green” marks shows correct or expected outputs and their labels. The “Red” marks shows the output generated by classification algorithm but these are not the expected outputs. The “Yellow” highlights in the datasets show the reference’s labels and their respective distances with the projected signs. Furthermore, the “Yellow” highlights in the tables shows the first K (5) numbers of results in ascending order.

8.3.1 Displaying Alphabet A

When we displayed alphabet “A” the following outcome was seen in the console.



```

sorting is calling
findingUnique is calling
findingMostOccurance is calling
□
S
A
A
S
A
New Value Saved...
New Value Saved...
New Value Saved...

```

Fig. 8.2 Data Returned From Algorithm For A

Here we can see that the showing result for displaying “A” is also “A”. That means the classification is working well. But we can see that the algorithm also predicted “S”. So, let’s compare the distances between all the reference of “A” and “S” with the displayed sign.

```

12.00,8.00,6.00,5.00,4.00,6.00,14.00,14.00,151,A
givingDistance is calling
50.00
11.00,4.00,2.00,3.00,3.00,4.00,12.00,14.00,120,S
givingDistance is calling
42.00
12.00,8.00,1.00,1.00,1.00,2.00,13.00,16.00,131,A
givingDistance is calling
31.00
8.00,5.00,1.00,3.00,3.00,4.00,12.00,14.00,53,S
givingDistance is calling
37.00
12.00,9.00,4.00,5.00,4.00,5.00,14.00,13.00,106,A
givingDistance is calling
47.00
12.00,4.00,1.00,2.00,2.00,3.00,12.00,16.00,89,S
givingDistance is calling
37.00

```

```

13.00,8.00,2.00,2.00,2.00,3.00,12.00,15.00,91,A
givingDistance is calling
36.00

8.00,5.00,1.00,3.00,3.00,4.00,11.00,14.00,90,S
givingDistance is calling
36.00

11.00,9.00,2.00,2.00,2.00,4.00,11.00,16.00,94,A
givingDistance is calling
32.00

8.00,4.00,0.00,2.00,3.00,3.00,10.00,16.00,185,S
givingDistance is calling
31.00

```

Figure no **: Showing the values of the predicted outcomes

Alphabet	Distance (Ascending order)
A	31
S	31
A	32
A	36
S	36
S	37
S	37
S	42
A	47
A	50

Table No : 1

We observe from the above result that in the lowest 5 distances “A” occurred 3 times and “S” occurred 2 times thus the most occurred one is the result, which is “A”.

The percentage of accuracy of “A” is $(3/5) \times 100 = 60\%$

8.3.2 Displaying Alphabet B

Displaying “B” showed the following outcome in the console.



```
sorting is calling  
findingUnique is calling  
findingMostOccurance is calling  
B  
B  
B  
B  
B  
B  
B  
New Value Saved...  
New Value Saved...  
New Value Saved...
```

Fig. 8.3 Displaying “B” and the corresponding result in console

Here, all the resultant values are “B”. Thus, all the Bs has the minimum distance among all 130 reference values.

```
3.00,6.00,17.00,17.00,17.00,14.00,16.00,14.00,166.00  
givingDistance is calling  
9.00
```

```

4.00,5.00,16.00,16.00,17.00,14.00,16.00,17.00,153,B
givingDistance is calling
16.00

4.00,5.00,16.00,17.00,17.00,15.00,15.00,16.00,108,B
givingDistance is calling
14.00

3.00,5.00,16.00,16.00,16.00,14.00,15.00,15.00,107,B
givingDistance is calling
15.00

3.00,4.00,14.00,15.00,16.00,14.00,14.00,15.00,108,B
givingDistance is calling
20.00

```

Figure no: The lowest values are B

```

12.00,8.00,6.00,5.00,4.00,6.00,14.00,14.00,151,A
givingDistance is calling
64.00
3.00,6.00,17.00,17.00,17.00,14.00,16.00,14.00,166,B
givingDistance is calling
9.00
9.00,9.00,12.00,9.00,10.00,12.00,15.00,13.00,155,C
givingDistance is calling
40.00
5.00,8.00,16.00,7.00,7.00,7.00,14.00,12.00,154,D
givingDistance is calling
43.00
4.00,3.00,4.00,6.00,6.00,7.00,14.00,14.00,143,E
givingDistance is calling
57.00
9.00,8.00,7.00,17.00,17.00,16.00,16.00,13.00,162,F
givingDistance is calling
24.00
11.00,9.00,16.00,4.00,3.00,5.00,4.00,10.00,143,G
givingDistance is calling
71.00
4.00,6.00,17.00,16.00,5.00,7.00,5.00,11.00,150,H
givingDistance is calling
44.00
9.00,6.00,3.00,3.00,4.00,14.00,14.00,15.00,158,I
givingDistance is calling
59.00
7.00,7.00,5.00,5.00,4.00,14.00,5.00,14.00,148,J
givingDistance is calling
60.00
5.00,12.00,16.00,16.00,5.00,6.00,13.00,16.00,164,K
givingDistance is calling
42.00
12.00,15.00,17.00,4.00,4.00,5.00,14.00,13.00,152,L
givingDistance is calling
63.00

```

Fig. 8.4 Showing some datasets and the lowest distance here is of "B"

Since, the five lowest distances are B. So, the percentage of accuracy of "B" is $(5/5) \times 100 = 100\%$

8.3.3 Displaying Alphabet C

Displaying "B" showed the following outcome in the console.



```

| sorting is calling
| findingUnique is calling
| findingMostOccurance is calling
| C
| C
| C
| C
| C
| C
| C
| New Value Saved...
| New Value Saved...
| New Value Saved...

```

Fig. 8.5 Displaying “C” and the corresponding result in console

In case of “C” also all the resultant values are “C”. Thus all the Cs has the minimum distance among all 130 reference values

```

9.00,9.00,12.00,9.00,10.00,12.00,15.00,13.00,155,C
givingDistance is calling
20.00

10.00,8.00,13.00,9.00,10.00,12.00,16.00,18.00,144,C
givingDistance is calling
19.00

9.00,8.00,11.00,9.00,11.00,11.00,14.00,15.00,101,C
givingDistance is calling
19.00

```

9.00,9.00,12.00,10.00,12.00,12.00,14.00,14.00,103,C
givingDistance is calling
19.00

9.00,8.00,12.00,9.00,11.00,12.00,13.00,15.00,103,C
givingDistance is calling
20.00

Figure no: For this projection the lowest values are C

12.00,8.00,6.00,5.00,4.00,6.00,14.00,14.00,151,A
givingDistance is calling
44.00

3.00,6.00,17.00,17.00,17.00,14.00,16.00,14.00,166,B
givingDistance is calling
29.00

9.00,9.00,12.00,9.00,10.00,12.00,15.00,13.00,155,C
givingDistance is calling
20.00

5.00,8.00,16.00,7.00,7.00,7.00,14.00,12.00,154,D
givingDistance is calling
27.00

4.00,3.00,4.00,6.00,6.00,7.00,14.00,14.00,143,E
givingDistance is calling
39.00

9.00,8.00,7.00,17.00,17.00,16.00,16.00,13.00,162,F
givingDistance is calling
40.00

11.00,9.00,16.00,4.00,3.00,5.00,4.00,10.00,143,G
givingDistance is calling
53.00

4.00,6.00,17.00,16.00,5.00,7.00,5.00,11.00,150,H
givingDistance is calling
44.00

9.00,6.00,3.00,3.00,4.00,14.00,14.00,15.00,158,I
givingDistance is calling
45.00

7.00,7.00,5.00,5.00,4.00,14.00,5.00,14.00,148,J
givingDistance is calling
48.00

5.00,12.00,16.00,16.00,5.00,6.00,13.00,16.00,164,K
givingDistance is calling
30.00

12.00,15.00,17.00,4.00,4.00,5.00,14.00,13.00,152,L
givingDistance is calling
45.00

2.00,5.00,6.00,10.00,10.00,6.00,12.00,15.00,147,M

Fig. 8.6 Showing some datasets and the lowest distance here is of "C"

Since, the five lowest distances are C. So the percentage of accuracy of "C" is $(5/5) \times 100 = 100\%$

8.3.4 Displaying Alphabet D

When we displayed alphabet “D” the following outcome was seen in the console.



```
sorting is calling  
findingUnique is calling  
findingMostOccurance is calling  
D  
D  
V  
D  
R  
D  
New Value Saved...  
New Value Saved...  
New Value Saved...
```

Fig. 8.7 Displaying “D” and the corresponding result in console

Here, we can observe that the showing result for displaying “D” is “D”. But we can see that the algorithm also predicted “R” and “V”. The comparison of the distances between all the references of “D”, “R” and “V” with the displayed sign is shown below.

```
5.00,8.00,16.00,7.00,7.00,7.00,14.00,12.00,154.00  
givingDistance is calling  
15.00
```

4.00,6.00,10.00,13.00,3.00,4.00,11.00,16.00,198,R
givingDistance is calling
34.00

5.00,7.00,14.00,4.00,4.00,4.00,13.00,14.00,98,D
givingDistance is calling
26.00

3.00,7.00,14.00,16.00,5.00,7.00,12.00,13.00,102,V
givingDistance is calling
26.00

3.00,7.00,13.00,15.00,3.00,5.00,12.00,15.00,100,R
givingDistance is calling
28.00

6.00,7.00,16.00,5.00,5.00,4.00,13.00,13.00,98,D
givingDistance is calling
24.00

3.00,6.00,17.00,15.00,5.00,6.00,13.00,14.00,101,V
givingDistance is calling
22.00

3.00,6.00,16.00,14.00,3.00,5.00,13.00,15.00,10,R
givingDistance is calling
24.00

5.00,7.00,16.00,5.00,5.00,5.00,12.00,14.00,97,D
givingDistance is calling
22.00

3.00,6.00,16.00,15.00,5.00,6.00,14.00,12.00,146,V
givingDistance is calling
24.00

3.00,6.00,15.00,14.00,2.00,4.00,13.00,14.00,141,R
givingDistance is calling
28.00

5.00,7.00,16.00,5.00,5.00,5.00,15.00,17.00,137,D
givingDistance is calling
20.00

3.00,7.00,17.00,16.00,6.00,7.00,13.00,11.00,120,V
givingDistance is calling
23.00

3.00,7.00,16.00,14.00,5.00,6.00,13.00,13.00,155,R
givingDistance is calling
22.00

3.00,6.00,12.00,15.00,5.00,6.00,11.00,16.00,204,V
givingDistance is calling
29.00

Fig. 8.8 Showing the values of the predicted outcomes

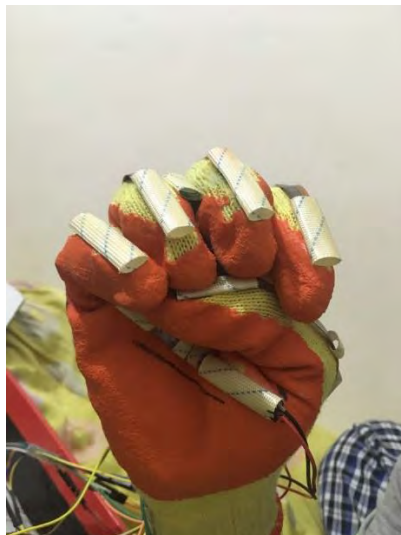
Alphabet	Distance (Ascending order)
D	15
D	20
V	22
D	22
R	22
V	23
D	24
V	24
R	24
D	26
V	26
R	28
R	28
V	29
R	34

We observe from the above result that in the lowest 5 distances “D” occurred 3 times, “R” occurred 1 time and “V” occurred 1 time. Thus, the most occurred one is the result, which is “D”.

The percentage of accuracy of “D” is $(3/5) \times 100 = 60\%$

8.3.5 Displaying Alphabet E

When we displayed alphabet “E” the following outcome was seen in the console.



```

sorting is calling
findingUnique is calling
findingMostOccurance is calling
E
E
E
O
X
E
New Value Saved...
New Value Saved...
New Value Saved...

```

Fig. 8.9 Displaying “E” and the corresponding result in console

Here, we can observe that the showing result for displaying “E” is “E”. But we can see that the algorithm also predicted “O” and “X”. The comparison of the distances between all the references of “E”, “O” and “X” with the displayed sign is shown below.

```

8.00,6.00,4.00,9.00,11.00,11.00,14.00,15.00,205 O
givingDistance is calling
30.00
4.00,2.00,2.00,4.00,7.00,6.00,12.00,15.00,92 E
givingDistance is calling
18.00
4.00,2.00,10.00,4.00,6.00,7.00,11.00,13.00,9 X
givingDistance is calling
19.00
8.00,6.00,8.00,6.00,9.00,9.00,12.00,16.00,97 O
givingDistance is calling
22.00
4.00,2.00,3.00,4.00,6.00,6.00,13.00,15.00,92 E
givingDistance is calling
15.00
4.00,2.00,11.00,3.00,6.00,7.00,12.00,14.00,94 X
givingDistance is calling
19.00
8.00,6.00,8.00,6.00,8.00,9.00,14.00,14.00,97 O
givingDistance is calling
21.00
4.00,3.00,4.00,6.00,6.00,6.00,13.00,14.00,92 E
givingDistance is calling
12.00
4.00,3.00,13.00,5.00,5.00,6.00,14.00,14.00,146 X
givingDistance is calling
16.00
8.00,6.00,7.00,6.00,7.00,7.00,16.00,14.00,137 O
givingDistance is calling
15.00

```

4.00,2.00,3.00,5.00,4.00,5.00,14.00,17.00,127 E
givingDistance is calling
16.00

5.00,2.00,11.00,5.00,6.00,7.00,12.00,12.00,114 X
givingDistance is calling
20.00

8.00,6.00,7.00,7.00,7.00,8.00,14.00,13.00,158 O
givingDistance is calling
20.00

4.00,3.00,4.00,6.00,6.00,7.00,14.00,14.00,143 E
givingDistance is calling
12.00

Alphabet	Distance (Ascending order)
E	12
E	12
E	15
O	15
X	15
E	16
X	16
E	16
X	18
X	19
X	19
O	20
O	20
O	21
O	22

4.00,2.00,9.00,4.00,5.00,7.00,11.00,16.00,190 X
givingDistance is calling
16.00

Fig. 8.10 Showing the values of the predicted outcomes

We observe from the above result that in the lowest 5 distances “E” occurred 3 times, “O” occurred 1 time and “X” occurred 1 time. Thus the most occurred one is the result, which is “E”.

The percentage of accuracy of “E” is $(3/5) \times 100 = 60\%$

8.3.6 Average Percentage of Accuracy

We have calculated the distance metrics according to the sum of the Euclidean formula

$$d = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

Average percentage of accuracy based on the first five alphabets: $(60+100+100+60+60)/5 = 76\%$

Chapter 9: Problem Analysis and Discussion

While going through the whole process of the project, we faced many difficulties. The major difficulties are discussed in this section. Initially while calibration we figured out that different hands have different values while calibration. So, for every time a new person wears the gloves he has to calibrate once again. We also noticed that, even in case of same hand the values of calibration are never the same. So no matter how many times a person calibrates the accuracy in value will not be satisfactory. This is a major drawback of our project. Moreover, every individual has different size and shape of hand. So we have to make separate gloves for separate persons as the flex will not be in the correct position and will not bend in the similar ways in case of different size of hands. Another problem that we faced is that, the value of the sensor is never completely stable. And again, the Gloves flexibility is also not stable. It is either more or less rigid while wearing depending on the hand. But if we attach glue with it its flexibility changes. These are the drawbacks related to gloves. While working with classification of the dataset, we tested with different classification algorithms but none of the algorithm gives entirely correct value. The last selected machine learning algorithm is KNN (K Nearest Neighbor). In the KNN algorithm the value of K cannot be decided for getting the best value. The most occurred unique label was supposed to be the result. But the required result is not always the most occurred. The accuracy of this process is not satisfactory. In case of applying the KNN algorithm, the distance we took from each saved sensor value to the projected sensor value was initially just normal subtraction. Then we approached Euclidean distance, Manhattan distance, etc. none of the distances could give exactly correct answer. We also observed that, SD card initialization failed frequently. Maybe as PCB was not integrated that is why the connection with the SD card was not always successful. In case of Node MCU, it does not work as a proper slave of Arduino Mega. It creates problem while serial communication with Arduino. Hence, the communication that we establish was not a reliable one. Moreover the value received at the Node MCU end from Arduino is not fast enough. Again, the library of Node MCU to Firebase is not a stable one. It created problem frequently. Now in case of sensors we faced some major difficulties. The sensors are not reliable. They used to get broken very easily. The metal shouldering of the flex sensors created immense problem as it broke down after getting slightest pressure. Moreover, the sensors are very expensive and the quality of the sensors are not up to the mark. We suppose, in Bangladesh good quality sensors and other devices are not available. Now in case of Android Applications, as they are real time application they consume

more battery life. However, the mobile application codes are not optimized so they take a lot more memory than they should. Lastly, in our project we did not send anything from mobile end to the arduino or the server. So the communication is one way. In future we will try to overcome this drawback. Also, the internet speed is not sufficient to transfer bulk amount of data at a time at a good frequency.

Chapter 10: Conclusion

10.1 Concluding Remarks

Every single country including Bangladesh has a great number of mute and deaf people. Although every single country tries to use its human resource efficiently but these mute and deaf people were never a part of the main stream society just because of the communication barrier. Researchers have been working on the problem for a long time and remarkable progress has already been done.

In this work, we have used modern technologies and efficient algorithm just not to give a solution proposal but to make a whole system, where the communication barrier will be actually bridged in reality. By the name of “OUR PROJECT NAME” we have successfully implemented the ASL learning approach and which is really a convenient way. The message passing system is also having a good structure which is about to be in its perfect condition after few modifications. And together two of its approach works as the most efficient system for closing the communication gap of the mute and deaf society which will not only have a good impact on the personal life of a mute and deaf person but the whole socio-economic factors will see a good progress.

10.2 Future Works

We need to collect a huge number of data from different size and shape of hand. The learning application gives us a remarkable result but when hand sizes changes drastically the system tends to give erroneous result.

Letter classification for message passing application needs to be modified and we already have an algorithm under development where each of the sensor will predict some of the letter it may show on that value of the sensor. After that, depending on the result of all sensor most occurred letter will be selected as the winner.

We will add word classification right after the change of letter classification of message learning approach. In future after creating a word it will compare its lexicographical values from the correct words with the same length. The words which will have least distance will be declared as the result of the whole system.

We will add multi user support as soon as possible. By enabling authentication on the mobile end, it will be able to recognize who is actually using the device and it will then collect the data from that specific user or hand glove.

Bibliography

- [1] World Health Organization, "Deafness and hearing loss," February 2017. [Online]. Available: <http://www.who.int/mediacentre/factsheets/fs300/en/>. [Accessed 19 August 2017].
- [2] M. G. Kumar, M. K. Gurjar and M. S. B. Singh, "American Sign Language Translating Glove using Flex Sensors," *Journal of Interdisciplinary Research (IJIR)*, vol. 2, no. 6, 2016.
- [3] T. Arsan and O. Ülgen, "SIGN LANGUAGE CONVERTER," *International Journal of Computer Science & Electronics (IJCES)*, vol. 6, no. 4, 2015.
- [4] M. Lin and R. Villalba, "Cornell University," [Online]. Available: http://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/f2014/rdv28_mjl256/webpage/. [Accessed 19 August 2017].
- [5] R. Krishna, S. Lee, S. P. Wang and J. Lang, "Cornell University," [Online]. Available: http://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2012/sl787_rak248_sw525_fl229/sl787_rak248_sw525_fl229/. [Accessed 19 August 2017].
- [6] P. Cloutier, A. Helderma and R. Mehili, "Prototyping a Portable, Affordable Sign," Worcester Polytechnic Institute, 2016.
- [7] S. N. Pawar, D. R. Shinde, G. S. Alai and S. S. Bodke, "Hand Glove To Translate Sign Language," *IJSTE - International Journal of Science Technology & Engineering*, vol. 2, no. 9, 2016.
- [8] Y. N. Khan and S. A. Mehdi, "Sign Language Recognition using Sensor Gloves," FAST-National University of Emerging Sciences, Lahore.
- [9] J. Bukhari, M. Rehman, S. I. Malik and A. M. Kamboh, "American Sign Language Translation through Sensor Gloves," *International Journal of u- and e- Service, Science and Technology*, vol. 8, no. 1, 2015.
- [10] A. K and R. P. J., "Hand Talk-A Sign Language Recognition," *International Journal of Innovative Research in Science, Technology & Management*, no. 3, July 2014.
- [11] "Arduino Mega," Arduino, [Online]. Available: <https://www.arduino.cc/en/Main/ArduinoBoardMega>. [Accessed 19 August 2017].
- [12] "Tensorflow," Tensorflow, [Online]. Available: <https://www.tensorflow.org/>. [Accessed 19 August 2017].
- [13] "FLEX SENSORS: spectrasymbol," spectrasymbol, [Online]. Available: <http://www.spectrasymbol.com/products/flex-sensors/>. [Accessed 18 August 2017].
- [14] "Force Sensitive Resistor 0.5": Sparkfun," Sparkfun, [Online]. Available: <https://www.sparkfun.com/products/11111>. [Accessed 19 August 2017].
- [15] "Espressif," Espressif, [Online]. Available: <http://espressif.com/products/hardware/esp8266ex/overview/>. [Accessed 19 August 2017].
- [16] "MPU-6050 Accelerometer + Gyro: Arduino," : Arduino , [Online]. Available: <https://playground.arduino.cc/Projects/Hardware/mpu6050/>. [Accessed 19 August 2017].

Appendix

Only Calibration

We will describe about the code snippet Only Calibration over here. The purpose of this code to take the maximum and minimum value of all flex sensor. Where all the maximum and minimum value will be printed in the serial monitor. Then we take the values and manually put the values wherever we need the code description line by line goes below:

```
int FLEX_PIN[8];

// Declarations of analogue pin by which we will be taking the sensors value from the circuit.

//these pins are mainly Arduino pin A0-A7

double STRAIGHT_RESISTANCE [8];

// these containers are declared to save the minimum or straight value of flex sensors.

double BEND_RESISTANCE [8];

//to store values coming from flex sensors when the sensors are in maximum bends

double flexADC [8];

//to store the raw values from sensors

//this method calibratingFlex takes analog values from sensors. After taking all the sensors values at
once //this method compares the value with its maximum and minimum value. If it is lower than
minimum then //this method save the value inside STRAIGHT_RESISTANCE[j]. if the raw value is
more than the maximum //then the value will be saved inside BEND_RESISTANCE;

//running the method 20,000 times with a delay of 1 millisecond gives the user enough time to
calibrate //his hands .

void calibratingFlex(){

    for(int i=0;i<8;i++){

        int flexADC=analogRead(FLEX_PIN[i]);

        STRAIGHT_RESISTANCE[i]=flexADC;

        BEND_RESISTANCE[i]=flexADC;

    }

    for(int i=0;i<20000;i++){

        for(int j=0;j<8;j++){

            int flexADC=analogRead(FLEX_PIN[j]);

            if(STRAIGHT_RESISTANCE[j]>flexADC){

                STRAIGHT_RESISTANCE[j]=flexADC;
```

```

    }
    else if(BEND_RESISTANCE[j]<flexADC){
        BEND_RESISTANCE[j]=flexADC;
    }
}
delay(1);
}
}

```

/* After determining the lowest and highest value for every sensor the purpose of this method is to print the values in a way that can be pasted into other method without any kind of editing.

```

void printingLimits(){
    Serial.println("-----");
    Serial.println("-----");
    for(int i=0;i<8;i++){
        String temp1 ="STRAIGHT_RESISTANCE[" + (String)i + "] = " +
        (String)STRAIGHT_RESISTANCE[i] + ",";
        Serial.print(temp1);
        Serial.println();
        String temp2 = "BEND_RESISTANCE[" + (String)i + "] = " + (String)BEND_RESISTANCE[i] +
        ",";
        Serial.print(temp2);
        Serial.println();
        Serial.println();
    }
}

```

/*this setup() will run just for once in its life time. We first initialize the variables with the exact analogue pin over here.then we run calibrate the flex and print those values in a way that it can be pasted anywhere without any hustle.

```

*/
void setup()
{
    FLEX_PIN[0]=A0;

```

```

FLEX_PIN[1]=A1;
FLEX_PIN[2]=A2;
FLEX_PIN[3]=A3;
FLEX_PIN[4]=A4;
FLEX_PIN[5]=A5;
FLEX_PIN[6]=A6;
FLEX_PIN[7]=A7;
Serial.begin(115200);
calibratingFlex();
printingLimits();
}

```

Saving and Creating Dataset (for both approach)

Here we will discuss about the code snippet “saving_creating_csv” . We have created this file to create reference data for our future classification. In this code, we wait for a data to appear similarly three times then if the user is sure to save the data he needs to press the controller button. Then program will ask the user to enter the label in the serial monitor. When the label/letter is being entered then it is saved to the SD card. The code elaboration given below:

```

//including the necessary file
#include <SD.h>
#include<SPI.h>

//this is the declaration of chip select pin of SD card module
const int CSpin = 53;

//Creating reference for File where the values will be saved
File sensorData;

// Declarations of analogue pin by which we will be taking the sensors value from the circuit.
//these pins are mainly Arduino pin A0-A7
int FLEX_PIN[8];

```

```
// Pin connected to voltage divider output. This force sensor is actually used as a controller to send the  
//from the Arduino to sd card
```

```
const int Force_Finger = A8;
```

```
//this pin is used to differentiate between u and v
```

```
const int Force_Switch = A9;
```

```
//This button was kept purposefully for debugging process
```

```
const int Force_Saved_Button = A10;
```

```
double STRAIGHT_RESISTANCE [8];
```

```
// these containers are declared to save the minimum or straight value of flex sensors.
```

```
double BEND_RESISTANCE [8];
```

```
//to store values coming from flex sensors when the sensors are in maximum bends
```

```
double flexADC [8];
```

```
//to store the raw values from sensors
```

```
//here value sensor value will be kept saved after getting from method compareFlexReading
```

```
String valueToBeSaved = "";
```

```
/*
```

this method ensures that a steady value is being saved to the SD card. This method takes one parameter name ans which is a current value of the flex sensors. Then the values are compared with the next two reading of all flex sensors. If it matches, then we save the examinee value to global variable “valueToBeSaved”. Then this method simply prints the sequence of steady data which indicates that this is a steady position and if you want to save this value as a reference you can save it.

```
*/
```

```
void compareFlexReading(String ans) {
```

```
    int count = 0;
```

```
    while (count < 2) {
```

```
        flag = false;
```



```

double tempFlexReading [8];
for (int i = 0; i < 8; i++) {
    double temp = analogRead(FLEX_PIN[i]);
    tempFlexReading[i] = map(temp, STRAIGHT_RESISTANCE[i], BEND_RESISTANCE[i], 1,
20);
}
for (int j = 0; j < 8; j++) {
    if (angle[j] != tempFlexReading[j]) {
        flag = true;
        return;
    }
}
if (flag == true) {
    break;
}
count++;
delay(50);
}
if (count == 2) {
    valueToBeSaved = ans;
    Serial.println(ans);
}
}

```

/*

After the data has been shoot to save in the SD card this method is called.It first opens the file from SD card and if everything works perfectly it simply saves the value to the SD card and closes the file reference

*/

```

void saveData() {
    sensorData = SD.open("data.csv", FILE_WRITE);

```

```

if (sensorData) { // check the card is still there
    // now append new data file
    sensorData.println(valueToBeSaved);
    sensorData.close(); // close the file
}
else {
    Serial.println("Error writing to file !");
}
}

//this method simply takes flex reading and saves into the flexADC global variable
void takingFlexReading() {
    for (int i = 0; i < 8; i++) {
        flexADC[i] = analogRead(FLEX_PIN[i]);
    }
}

```

/*

Inside the setup method we first begin the serial communication. Then initialize the flex pin assigning the pin values. Then we need to manually assign the values of the upper limit of lower limit for every single flex sensor. We have go the values from only calibration code snippet. Lastly we have set the pinMode for various analogue pin for reading and writing

*/

```

void setup()
{
    Serial.begin(115200);
    FLEX_PIN[0] = A0;
    FLEX_PIN[1] = A1;
    FLEX_PIN[2] = A2;
    FLEX_PIN[3] = A3;
    FLEX_PIN[4] = A4;
    FLEX_PIN[5] = A5;
    FLEX_PIN[6] = A6;
}

```

```
FLEX_PIN[7] = A7;
```

```
STRAIGHT_RESISTANCE[0] = 111.00;
```

```
BEND_RESISTANCE[0] = 427.00;
```

```
STRAIGHT_RESISTANCE[1] = 501.00;
```

```
BEND_RESISTANCE[1] = 776.00;
```

```
STRAIGHT_RESISTANCE[2] = 437.00;
```

```
BEND_RESISTANCE[2] = 680.00;
```

```
STRAIGHT_RESISTANCE[3] = 505.00;
```

```
BEND_RESISTANCE[3] = 724.00;
```

```
STRAIGHT_RESISTANCE[4] = 450.00;
```

```
BEND_RESISTANCE[4] = 708.00;
```

```
STRAIGHT_RESISTANCE[5] = 543.00;
```

```
BEND_RESISTANCE[5] = 710.00;
```

```
STRAIGHT_RESISTANCE[6] = 575.00;
```

```
BEND_RESISTANCE[6] = 724.00;
```

```
STRAIGHT_RESISTANCE[7] = 586.00;
```

```
BEND_RESISTANCE[7] = 710.00;
```

```
//File Code for initializing
```

```
Serial.print("Initializing SD card...");
```

```
pinMode(CSpin, OUTPUT);
```

```
//
```

```
// see if the card is present and can be initialized:
```

```
if (!SD.begin(CSpin)) {
```

```

Serial.println("Card failed, or not present");
// don't do anything more:
return;
}
Serial.println("card initialized.");
//File Code Ends

```

```

pinMode(FLEX_PIN, INPUT);
pinMode(Flex_Finger, INPUT); //Pin setup for force sensor
pinMode(Flex_Switch, INPUT); //Pin setup for force sensor
pinMode(Flex_Saved_Button, INPUT);

}

```

```

/*

```

This is the main body part of the whole program mainly we run all the method sequentially in this loop. First, we take the flex reading and save them to global variable then we map them into the from 1-20. And by concatenating we make a string called ans. After pressing the force sensor to shoot the value to the SD card. The answer is saved as comma separated value.

```

*/

```

```

void loop()
{

```

```

//this force sensor detect which mode should work right now? numerical or letter
int force_Button_ADC = analogRead(Flex_Switch); //Reading force values
if (force_Button_ADC > 600 && flag == false) {
    flag = true;
    delay(500);
}
else if (force_Button_ADC > 600 && flag == true) {

```

```

    flag = false;
    delay(500);
}

// Read the ADC, and calculate voltage and resistance from it
String ans = "";

//here we are taking reading from all the sensors and saving them to the flexADC array
takingFlexReading();

// Use the calculated resistance to estimate the sensor's
// bend angle:
for (int i = 0; i < 8; i++) {
    angle[i] = map(flexADC[i], STRAIGHT_RESISTANCE[i], BEND_RESISTANCE[i], 1, 20);
    if (i == 0) {
        ans = angle[i];
    } else {
        ans = ans + "," + angle[i];
    }
}

int force_finger_ADC = analogRead(Force_Finger);//Reading force values
ans = ans + "," + force_finger_ADC;

//Serial.println(ans);
compareFlexReading(ans);

int force_btn_value_saved = analogRead(Force_Saved_Button);
//Serial.println(force_btn_value_saved);

//after creating force on this sensor we save value to file
if (force_btn_value_saved > 600) {
    Serial.println("Please Give Us The Character");
    while (!Serial.available());
    if (Serial.available() > 0) {
        char label = Serial.read();
        valueToBeSaved = valueToBeSaved + "," + label;
    }
}

```

```

        saveData();
    }

    Serial.println(valueToBeSaved);
    delay(500);
}

delay(100);
}

```

Learning ASL

Reading csv and testing algorithm

Here we will talk about another code snippet reading csv and testing algorithm. This program mainly takes one specific hand gesture from glove. And with the help of controlling button it shoots the value to the Arduino. Whenever a new value is received by the Arduino the k-nearest neighbor routine runs and try to find out the neighbors. Then we find out the unique values of the string and send it to nodeMcu. Below the code description is given:

```

//includes sd card library to the
#include <SD.h>

//Includes spi communication protocol library support in the routine
#include<SPI.h>

//it is declared to build the serial communication between Arduino and nodemcu
#include <SoftwareSerial.h>

//Here we declare the Rx Tx pin to communicate with the Arduino
SoftwareSerial ArduinoSerial(3, 2); //RX , TX

const int CSpin = 53;

//Just declaring the file instance from where the value will be retrieved. The file is in the sd card
which //is inside the sd card module.

File sensorData;

//This variable is the most important variable. Here we actually define how many neighbor we will
take

const int k = 5;

```

```

//File Block Codes END

//the hardware CS pin (10 on most Arduino boards,
// 53 on the Mega) must be left as an output or the SD
// library functions will not work.
int FLEX_PIN[8];
//Container for saving the pin reference of flex sensors
// Pin connected to voltage divider output
int Force_Finger = A8;
// a force sensor pin which will be used to throw value to compare
const int Force_Switch = A9;
// a force sensor to differentiate between "u" and "v"
const int Force_Saved_Button = A10;
// Button for concatenation which is not applicable for this specific program
//will hold the ultimate answer which will be shoot to the NodeMcu
String ans = "";
//to save temporary guessed ans which is later saved into ans
String tempAns = "";
double STRAIGHT_RESISTANCE [8];
// these containers are declared to save the minimum or straight value of flex sensors.
double BEND_RESISTANCE [8];
//to store values coming from flex sensors when the sensors are in maximum bends
double flexADC [8];
//to store the raw values from sensors
//to store values coming from flex sensors
double flexADC [8];
//to store the angle calculated from map function
float angle [8];
//one boolean flag to change blocks
boolean flag = true;
//here value sensor value will be kept saved after getting from method compareFlexReading
String valueToBeSaved = "";

```

```
/*
```

this method ensures that a steady value is being saved to the SD card. This method takes one parameter name ans which is a current value of the flex sensors. Then the values are compared with the next two reading of all flex sensors. If it matches, then we save the examinee value to global variable “valueToBeSaved”. Then this method simply prints the sequence of steady data which indicates that this is a steady position and if you want to save this value as a reference you can save it.

```
*/
```

```
void compareFlexReading(String ans) {  
    int count = 0;  
    while (count < 1) {  
        flag = false;  
        double tempFlexReading [8];  
        for (int i = 0; i < 8; i++) {  
            double temp = analogRead(FLEX_PIN[i]);  
            tempFlexReading[i] = map(temp, STRAIGHT_RESISTANCE[i], BEND_RESISTANCE[i], 1,  
20);  
        }  
        for (int j = 0; j < 8; j++) {  
            if (angle[j] != tempFlexReading[j]) {  
                flag = true;  
                break;  
            }  
        }  
        if (flag == true) {  
            break;  
        }  
        count++;  
    }  
    if (count == 1) {  
        valueToBeSaved = ans;  
        Serial.println("New Value Saved...");  
        // Serial.println(ans);  
    }  
}
```



```

    }
}

```

```

/*

```

After the data has been shoot to save in the SD card this method is called. it first opens the file from sd card and if everything works perfectly it simply saves the value to the sd card and closes the file reference

```

*/

```

```

void saveData() {
    sensorData = SD.open("data.csv", FILE_WRITE);
    if (SD.exists("data.csv")) { // check the card is still there
        // now append new data file
        if (sensorData) {
            sensorData.println(valueToBeSaved);
            sensorData.close(); // close the file
        }
    }
    else {
        Serial.println("Error writing to file !");
    }
}

```

```

/*

```

This method first open the csv file then declare two array to store the distance between the projected value and the saved values in the SD card. Another character array saves the label where it will also be sorted when the distance array is sorted. This method takes one single line of reference value at a time and compare it with the projected value and save it then go for the next reference value. Then after sorting the array it takes Kth values and then within the kth values it find out unique characters that is send to the nodeMcu

```

*/

```

```

void readAndRetrieve() {
    Serial.println("readAndRerieve method is calling");
}

```

```

int lineSize = gettingHashSize();//Total number of lines in the given txt file
sensorData = SD.open("data.csv"); //Opening file
if (sensorData) {

    Serial.println(lineSize);
    double distanceSaver[lineSize]; //Distance saving array
    char characterSaver[lineSize]; //Character saving array corresponding to distance
    String str = "";           //string variable to save each line in file
    int count = 0;             //Counter of the array index
    //open the file here
    while (sensorData.available()) {
        //A inconsistent line length may lead to heap memory fragmentation
        str = sensorData.readStringUntil('\n'); //Reading and retrieving each line in file
        if (str == "") //no blank lines are anticipated
            break;
        //working on delimiting the retrieved value

        Serial.println(str);
        //call the method givingDistance

        double distanceTemp = givingDistance(str, valueToBeSaved); // Getting distance between file's
        one retrieved value and the current value to be checked

        distanceSaver[count] = distanceTemp;           // The distance is saved in the array

        //might return an error cause I am assuming the tempAns is having only one character written by
        givingDistance

        characterSaver[count] = tempAns[0];           //The corresponding character is saved in the
        character array, generated from givingDistance() method

        Serial.println(distanceTemp);
        count++;
    }
    sensorData.close();

    sorting(distanceSaver, characterSaver, lineSize); //When array computation are complete then
    they are sorted

```

String uniqueChars = findingUnique(characterSaver, lineSize); //Finding unique charaters-
(concatated in a string) from the sorted character array

String ANS = (String)findingMostOccurance(uniqueChars, characterSaver); //Finding most
occured character among the k number of characters

```
Serial.println(ANS);  
// String sendtoNodeMcu=(String)ANS;  
ArduinoSerial.println(uniqueChars);  
} else {  
    // if the file didn't open, print an error:  
    Serial.println("error opening data.csv");  
}  
}
```

/*

After finding the kth array this method simply returns the unique value from the Arduino.

*/

```
String findingUnique(char charArray [], int lengthOfArray) {  
    Serial.println("findingUnique is calling");  
    String unique = (String)charArray[0];  
    for (int i = 1; i <= k; i++) {  
        char c = charArray[i];  
        int j = 0;  
        for (; j < unique.length(); j++) {  
            if (unique[j] == c) {  
                break;  
            }  
        }  
        if (j == unique.length()) {  
            unique.concat(c);  
        }  
    }  
    return unique;  
}
```

```

}

/*
This method returns thee most occurrence letter
*/
char findingMostOccurance(String unique, char characterArray []) {
    Serial.println("findingMostOccurance is calling");
    int counter = 0;
    char ans;
    for (int i = 0; i < unique.length(); i++) {
        int tempCounter = 0;
        char tempChar = unique[i];
        for (int j = 0; j <= k; j++) {
            if (unique[i] == characterArray[j]) {
                tempCounter++;
            }
        }
        if (tempCounter > counter) {
            counter = tempCounter;
            ans = tempChar;
        }
    }
    for (int i = 0; i < k; i++) {
        Serial.println(characterArray[i]);
    }
    return ans;
}

/*

```

In this method, we sort the distance of the distance array. Whenever we change any value/distance position we change the corresponding character also.so as a result not the

neighboring character becomes in the top of the character array. So, after that we can simply take the first k'th character to find out the most occurrence character.

```
*/
void sorting(double value[], char characters[], int lengthOfArray) {
    Serial.println("sorting is calling");
    for (int i = 0; i < lengthOfArray; i++) {
        for (int j = 0; j < i; j++) {
            if (value[i] <= value[j]) {
                int temp = value[i];
                value[i] = value[j];
                value[j] = temp;
                char charTemp = characters[i];
                characters[i] = characters[j];
                characters[j] = charTemp;
            }
        }
    }
}
```

/*

This method takes two string as parameter. Those string were saved as csv format and here csv format values again parsed into double value and kept inside two temporary double array. Then distance have been calculated in between these two values (one of them is projected or testing and other is reference or from the dataset).

*/

```
double givingDistance(String str, String valueToCheck) {
    Serial.println("givingDistance is calling");

    double tempVal[9];
    double valueToCheckVal[9];
    //for keeping track of which sensor value we are getting right now
    int counter = 0;
```

```

String temp = "";
double ans = 0;
for (int i = 0; i < str.length(); i++) {
    if (str[i] != ',') {
        temp = temp + str[i];
    } else {
        tempVal[counter] = temp.toFloat();
        counter++;
        temp = "";
    }
}
tempAns = temp;
counter = 0;
for (int i = 0; i < valueToCheck.length() ; i++) {
    if (valueToCheck[i] != ',') {
        temp = temp + valueToCheck[i];
    } else {
        valueToCheckVal[counter] = temp.toFloat();
        counter++;
        temp = "";
    }
    if (i == valueToCheck.length() - 1) {
        valueToCheckVal[counter] = temp.toFloat();
        counter++;
        temp = "";
    }
}
//here i am printing the float array got from above code
for (int i = 0; i < 8; i++) {
    //ans = ans + sq(valueToCheckVal[i] - tempVal[i]);
    ans = ans + abs(valueToCheckVal[i] - tempVal[i]);
}

```

```

//calculating Euclidian distance sqrt(sq(sum(Xi-Yi)))
//ans = sqrt(ans);
return ans;
}

```

```

/*

```

This method simply returns the line numbers or reference values. It searches for the “\n” token which can only be found once in the end of the line. So, the number of “\n” we are using that much line or reference we have in the dataset

```

*/
int gettingHashSize() {
    Serial.println("gettingHashSize is printing");
    sensorData = SD.open("data.csv");
    int val = 0;
    if (sensorData) {
        Serial.println("data.csv:");

        // read from the file until there's nothing else in it:
        while (sensorData.available()) {
            sensorData.readStringUntil('\n');
            val++;
        }
        // close the file:
        sensorData.close();
    } else {
        // if the file didn't open, print an error:
        Serial.println("error opening test.txt");
    }
    return val;
}

/*

```

Default function of arduino. Firstly, we set up the baud rate then we begin another serial communication for communicating with nodeMcu. After that we have simply initialized pin numbers inside the Flex_Pin array. Then we have set some pinmode to input for functioning properly.

```
*/  
void setup()  
{  
  Serial.begin(115200);  
  ArduinoSerial.begin(4800);  
  FLEX_PIN[0] = A0;  
  FLEX_PIN[1] = A1;  
  FLEX_PIN[2] = A2;  
  FLEX_PIN[3] = A3;  
  FLEX_PIN[4] = A4;  
  FLEX_PIN[5] = A5;  
  FLEX_PIN[6] = A6;  
  FLEX_PIN[7] = A7;  
  
  STRAIGHT_RESISTANCE[0] = 111.00;  
  BEND_RESISTANCE[0] = 427.00;  
  
  STRAIGHT_RESISTANCE[1] = 501.00;  
  BEND_RESISTANCE[1] = 776.00;  
  
  STRAIGHT_RESISTANCE[2] = 437.00;  
  BEND_RESISTANCE[2] = 680.00;  
  
  STRAIGHT_RESISTANCE[3] = 505.00;  
  BEND_RESISTANCE[3] = 724.00;  
  
  STRAIGHT_RESISTANCE[4] = 450.00;  
  BEND_RESISTANCE[4] = 708.00;
```



```
STRAIGHT_RESISTANCE[5] = 543.00;
```

```
BEND_RESISTANCE[5] = 710.00;
```

```
STRAIGHT_RESISTANCE[6] = 575.00;
```

```
BEND_RESISTANCE[6] = 724.00;
```

```
STRAIGHT_RESISTANCE[7] = 586.00;
```

```
BEND_RESISTANCE[7] = 710.00;
```

```
//File Code for initializing
```

```
Serial.print("Initializing SD card...");
```

```
pinMode(CSpin, OUTPUT);
```

```
//
```

```
// see if the card is present and can be initialized:
```

```
if (!SD.begin(CSpin)) {
```

```
    Serial.println("Card failed, or not present");
```

```
    // don't do anything more:
```

```
    return;
```

```
}
```

```
Serial.println("card initialized.");
```

```
//File Code Ends
```

```
pinMode(FLEX_PIN, INPUT);
```

```
pinMode(Force_Finger, INPUT); //Pin setup for force sensor
```

```
pinMode(Force_Switch, INPUT); //Pin setup for force sensor
```

```
pinMode(Force_Saved_Button, INPUT);
```

```
}
```

```
/*
```

In this default Arduino loop we take values of flex sensors we use the map function and we map the values inside angle array. Which are then concatenated to one single string and created

a csv value. After that when the controller/force sensor is been pressed we simply calculated the distance and findout neighbor through k-nearest neighbor algorithm.

```

*/
void loop()
{
  //this force sensor detets which mode should work right now? numerical or letter
  int force_Button_ADC = analogRead(Force_Switch);//Reading force values
  if (force_Button_ADC > 600 && flag == false) {
    flag = true;
    delay(500);
  }
  else if (force_Button_ADC > 600 && flag == true) {
    flag = false;
    delay(500);
  }
  // Read the ADC, and calculate voltage and resistance from it
  String ans = "";
  //here we are taking reading from all the sensors and saving them to the flexADC array
  takingFlexReading();
  // Use the calculated resistance to estimate the sensor's
  // bend angle:
  for (int i = 0; i < 8; i++) {
    angle[i] = map(flexADC[i], STRAIGHT_RESISTANCE[i], BEND_RESISTANCE[i], 1, 20);
    if (i == 0) {
      ans = angle[i];
    } else {
      ans = ans + "," + angle[i];
    }
  }
  int force_finger_ADC = analogRead(Force_Finger);//Reading force values
  ans = ans + "," + force_finger_ADC;
  //Serial.println(ans);

```

```

compareFlexReading(ans);

int force_btn_value_saved = analogRead(Force_Saved_Button);
//Serial.println(force_btn_value_saved);
//after creating force on this sensor we save value to file
if (force_btn_value_saved > 600) {
    readAndRetrieve();
    delay(500);
}
delay(100);
}

```

NodeMcu Finale

In this document, we will show the elaboration of the NodeMcu Algorithm for ASLLearningApp. This code is relatively simple. Once NodeMcu gets values from Arduino it simply set the value to firebase under the token “Letter”. I will try elaborate the code below:

```

//including all the library which will be needed to communicate with firebase
#include <SoftwareSerial.h>
#include <Firebase.h>
#include <FirebaseArduino.h>
#include <FirebaseCloudMessaging.h>
#include <FirebaseError.h>
#include <FirebaseHttpClient.h>
#include <FirebaseObject.h>
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>

#define FIREBASE_HOST "testing-383b1.firebaseio.com"//name of the firebase database
#define FIREBASE_AUTH "SeIpXZ386mh8CJRz7zbk67JnJPAn8QQS5MHeGFGR"//the secret of the database

#define WIFI_SSID "#SSID"//you should put your wifi name over here
#define WIFI_PASSWORD "#####"//you should put your wifi name over here

String str = "";

SoftwareSerial NodeSerial(D2, D3); //RX | TX //defining the serial communication pin with android

```

```
/*
```

Here we initiate serial communication. Establish wifi connection. And set the pinmode of D2,D3 pin over here

```
*/
```

```
void setup() {  
  Serial.begin(115200);  
  NodeSerial.begin(4800);  
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);  
  Serial.println("connecting");  
  while (WiFi.status() != WL_CONNECTED) {  
    Serial.print(".");  
    delay(500);  
  }  
  Serial.println();  
  Serial.println("connected: ");  
  Serial.println(WiFi.localIP());  
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);  
  pinMode(D2, INPUT);  
  pinMode(D3, OUTPUT);  
}
```

```
/*
```

In this loop whenever some data is available in the serial we take the data as a whole line till the “\n”. then we set the string to firebase with token “Letter”

```
*/
```

```
void loop() {  
  if (NodeSerial.available()) {  
    String str = NodeSerial.readStringUntil('\n');  
    Serial.println(str);  
    Firebase.setString("Letter", str);  
  }  
}
```

```

if (Firebase.failed()) {
    Serial.print("pushing /logs failed:");
    Serial.println(Firebase.error());
    return;
}
}
}

```

Android application for Sign Language Converter.

In our particular system, we have used two different applications. One is for learning sign language and another one is for message passing. Here we will discuss about both the applications in details with line-by-line code.

LearningASL Android Application

When user wears the glove on his hand and start LearningASL application first screen will pop up and it will take the user to the application with a welcoming message. By tapping on the screen it will take the user through the learning process. On the second screen letter A will show up. When the user taps on that screen an image will be displayed which is the sign gesture of letter A. Now user have to make that particular gesture with his hand. If the sign matches with the gesture an image of tick sign will be displayed to ensure the user that he is correct. Otherwise a cross sign representing wrong sign will be displayed. If it is correct and user taps on the screen the next letter B will be shown. But if the user is in wrong screen he/she will have to try letter A again. And this process will continue till Z. After successfully completing all the letters user will be taken to the third and last screen with congratulation message on it.

Now we will discuss about the technology behind this application in details along with the code.

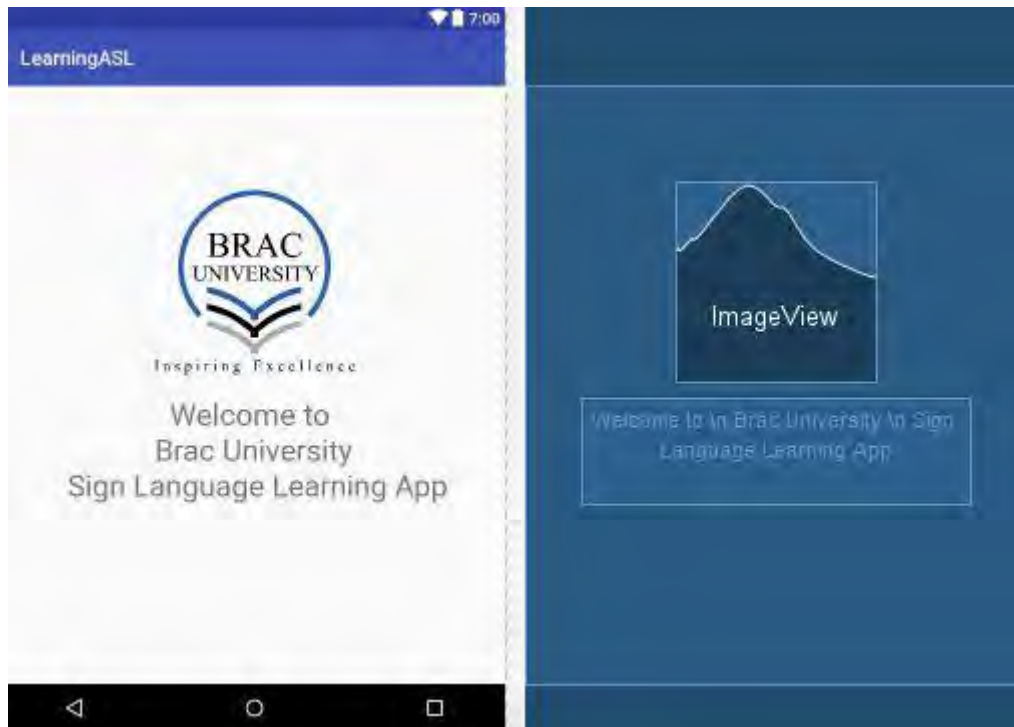


Fig. 10.1 User interface of the application

In this screen there is one image view and one text view. The image view is showing the BRAC University logo and text view is displaying welcome message.

```
public class LauncherActivity extends AppCompatActivity {
    RelativeLayout relativeLayout;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_launcher);
        relativeLayout = (RelativeLayout) findViewById(R.id.layout);

        relativeLayout.setOnClickListener((v) -> {
            Intent intent = new Intent(getBaseContext(), MainActivity.class);
            startActivity(intent);
        });
    }
}
```

Fig. 10.2 Launcher activity of learning application.

These are the codes for our launcher activity or welcome screen. Just an on click listener event is there which takes the user to next screen when the user taps on this screen.

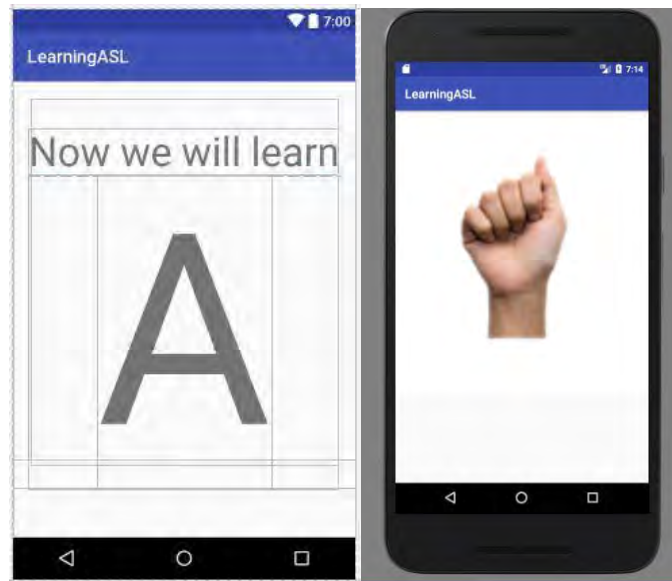


Fig. 10.3 Application using

Tapping on welcome screen user will be shifted to our main activity. Here within a text view with Letters appears. When the user taps on this screen, an on click listener event will execute and the text view will hide and an image view with the corresponding gesture of that letter will be displayed on the same screen.

Now the user will have to perform the given gesture with his hand. By performing it, the glove's sensor will collect the data and after classifying the data it will be sent to NodeMCU. With the help of NodeMCU the data will be sent to online server. After that user will get the data back in his android smartphone. This data will produce the result showing whether the gesture is right or wrong.

Based on the result user will be provided with a right or wrong screen. If the gesture is correct, the image view will be replaced by tick symbol on it. On the other hand, if the gesture is not correct the gesture image view will be replaced with cross symbol. For the right gesture screen, user will be shown the next letter. If it is wrong user needs to perform previous later again.

```
wrong.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        wrong.setVisibility(View.INVISIBLE);
        linearLayout.setVisibility(View.VISIBLE);
    }
});
```

The crucial part of this application is to reduce size of the application and make it faster. Therefore, instead of making 54 different activities for 26 English Letters and 26 images of gestures and 2 right and wrong symbol we have done it in one activity. Within this one activity, letters and images are changing according to user's activity. Now the question is how we have done it.

```
right.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        if(n==26){  
            Intent intent = new Intent(getApplicationContext(),LastActivity.class);  
            startActivity(intent);  
        }  
        switch (n) {  
            case 0:  
                letter.setText("A");  
                break;  
            case 1:  
                letter.setText("B");  
                break;  
            case 2:  
                letter.setText("C");  
                break;  
            case 3:  
                letter.setText("D");  
                break;  
            case 4:  
                letter.setText("E");  
                break;  
            case 5:  
                letter.setText("F");  
                break;  
            case 6:
```



```
        letter.setText("G");
        break;
case 7:
    letter.setText("H");
    break;
case 8:
    letter.setText("I");
    break;
case 9:
    letter.setText("J");
    break;
case 10:
    letter.setText("K");
    break;
case 11:
    letter.setText("L");
    break;
case 12:
    letter.setText("M");
    break;
case 13:
    letter.setText("N");
    break;
case 14:
    letter.setText("O");
    break;
case 15:
    letter.setText("P");
    break;
case 16:
    letter.setText("Q");
    break;
```

```

        case 17:
            letter.setText("R");
            break;
        case 18:
            letter.setText("S");
            break;
        case 19:
            letter.setText("T");
            break;
        case 20:
            letter.setText("U");
            break;
        case 21:
            letter.setText("V");
            break;
        case 22:
            letter.setText("W");
            break;
        case 23:
            letter.setText("X");
            break;
        case 24:
            letter.setText("Y");
            break;
        case 25:
            letter.setText("Z");
            break;
    }

    right.setVisibility(View.INVISIBLE);
    linearLayout.setVisibility(View.VISIBLE);
}

```

```

    });

    linearLayout.setOnClickListener(new View.OnClickListener() {

        @Override

        public void onClick(View v) {

            incomingStremFlag = true;

            switch (n) {

                case 0:

                    imageView.setImageResource(R.drawable.a);

                    break;

                case 1:

                    imageView.setImageResource(R.drawable.b);

                    break;

                case 2:

                    imageView.setImageResource(R.drawable.c);

                    break;

                case 3:

                    imageView.setImageResource(R.drawable.d);

                    break;

                case 4:

                    imageView.setImageResource(R.drawable.e);

                    break;

                case 5:

                    imageView.setImageResource(R.drawable.f);

                    break;

                case 6:

                    imageView.setImageResource(R.drawable.g);

                    break;

                case 7:

                    imageView.setImageResource(R.drawable.h);

                    break;

```

```
case 8:
    imageView.setImageResource(R.drawable.i);
    break;
case 9:
    imageView.setImageResource(R.drawable.j);
    break;
case 10:
    imageView.setImageResource(R.drawable.k);
    break;
case 11:
    imageView.setImageResource(R.drawable.l);
    break;
case 12:
    imageView.setImageResource(R.drawable.m);
    break;
case 13:
    imageView.setImageResource(R.drawable.n);
    break;
case 14:
    imageView.setImageResource(R.drawable.o);
    break;
case 15:
    imageView.setImageResource(R.drawable.p);
    break;
case 16:
    imageView.setImageResource(R.drawable.q);
    break;
case 17:
    imageView.setImageResource(R.drawable.r);
    break;
case 18:
    imageView.setImageResource(R.drawable.s);
```

```

        break;
    case 19:
        imageView.setImageResource(R.drawable.t);
        break;
    case 20:
        imageView.setImageResource(R.drawable.u);
        break;
    case 21:
        imageView.setImageResource(R.drawable.v);
        break;
    case 22:
        imageView.setImageResource(R.drawable.w);
        break;
    case 23:
        imageView.setImageResource(R.drawable.x);
        break;
    case 24:
        imageView.setImageResource(R.drawable.y);
        break;
    case 25:
        imageView.setImageResource(R.drawable.z);
        break;
    }
    linearLayout.setVisibility(View.INVISIBLE);
    imageView.setVisibility(View.VISIBLE);

}

});

}

```

With the help of the code given above the decision is taking place. Here we can see the code begins with N=0. In this case the screened letter will be A and its corresponding gesture will

be shown after that. If the user can perform the gesture correctly, the value of N will increase. With the new value of N, the next letter will be shown to user. In this case, the letter will be “B”. On the other hand, if the user unable to perform the gesture correctly the value of N will remain same and user will be given the previous letter again.



Fig. 10.4 User interface of the application

After performing all the gestures correctly user will be shifted to the last screen. Here with an image view alongside two text views user will be ensured that he has completed the learning of sign language.

Code for Message Passing Android Application

Message passing, reading and testing algorithm for Android

This code snippet works more or less similar of `reading_csv_testing_algorithm`. Only difference is whenever we test any data it simply concatenates the predicted letter with the previous letter. And this is how this algorithm makes a word. Another force sensor is used to send the data from the Arduino to the NodeMcu.

NodeMcu Code for Message Passing

This algorithm works more or less similarly with NodeMcu finale. The only difference is it will receive a valid word from the Arduino and it will push and set the values under two different

token in Firebase. One of the token is “word” and another token is “sentence”. These are used in message passing Android application.

Message Passing Android Application

The launching screen of this application is a page that has an image of BRAC University logo and a text view. If the layout is clicked once then the application will move to the next page. In this page there is a list view which shows the list of the words pushed in the firebase. If we long press on any list item, then the app will move to the next activity which shows the last added words. The sequence of words can represent a sentence. In firebase in the “word” tag the strings are being pushed and in the “sentence” tag the same strings are being set. The pushed values will be shown in the list view activity and in the message activity the strings retrieved from firebase is concat and the concat string is shown in the text view. While concatenating a space is also contacted in between the words. Thus, the sequential words look like sentence.

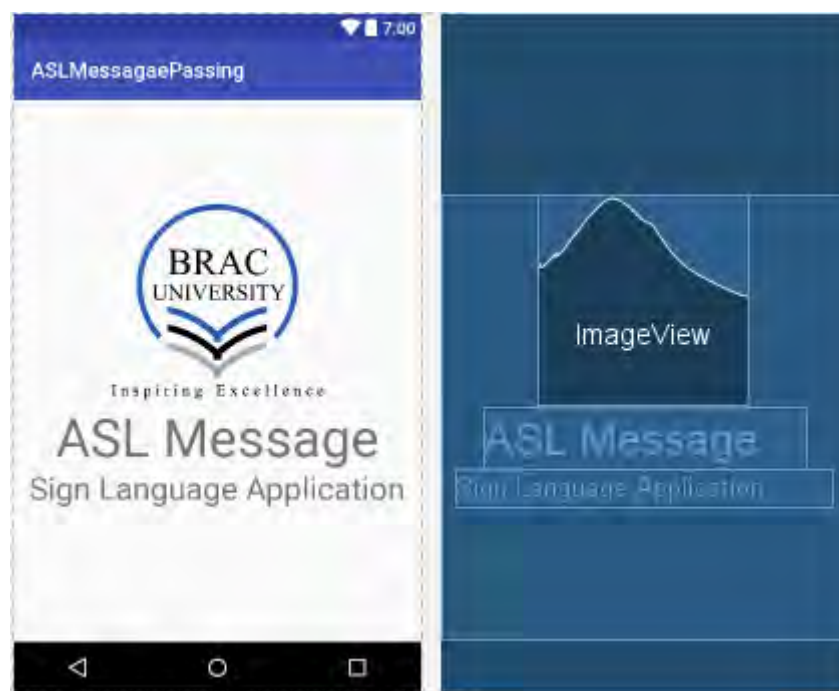


Fig. 10.5 The first window of the Message Passing Application

The main code of the main activity is given below:

```
RelativeLayout.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {
```

```

Intent intent = new Intent(getApplicationContext(),ListActivity.class);
startActivity(intent);
}
});

```

Here, clicking on the relative layout the app will move forward to the next activity with the intent call. Therefore, the application will move to the List Activity. Inside the list activity there is list view which contains all the “word” retrieved from firebase.



Fig. 10.6 ListActivity view

```

public class ListActivity extends AppCompatActivity {
    FirebaseDatabase database = FirebaseDatabase.getInstance();
    DatabaseReference myRef = database.getReference("word");
    ListView listView;
    ArrayList<String>messageList = new ArrayList<String>();
    ArrayAdapter<String>adapter;
    LinearLayout linearLayout;

```


Here, instance of the FirebaseDatabase and DatabaseReference is made. The DatabaseReference shows that it can retrieve and put value in firebase child “word”. We made an ArrayList and an ArrayAdapter. The ArrayList is put in the ArrayAdapter.

```
myRef.addChildEventListener(new ChildEventListener() {  
    @Override  
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {  
        String value = dataSnapshot.getValue(String.class);  
        adapter.add(value);  
    }  
})
```

OnChildAdded method adds the child that already exists in the child of “word” and all the future childs added will be also added in the arrayAdapter. The ListView while Long pressed goes to the next activity which is MessageActivity. This is done by intent call.

```
listView.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {  
    @Override  
    public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {  
        Intent intent = new Intent(getApplicationContext(), MessageActivity.class);  
        startActivity(intent);  
        return false;  
    }  
});
```

Then comes the MessageActivity. Which has a textView and a clear button. The textView shows the integrated message and the clear button clears the message.

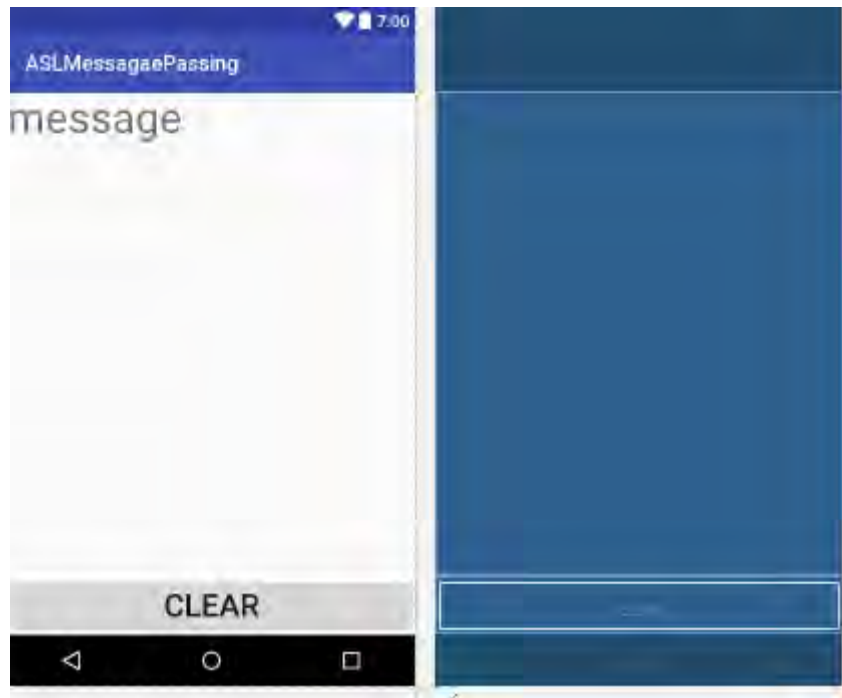


Fig. 10.7 Message Activity

```

public class MessageActivity extends AppCompatActivity {
    RelativeLayout relativeLayout;
    TextView textView;
    FirebaseDatabase database = FirebaseDatabase.getInstance();
    DatabaseReference myRef = database.getReference("sentence");
    Button clear;
    String msg;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_message);
        textView = (TextView)findViewById(R.id.txt);
        relativeLayout = (RelativeLayout) findViewById(R.id.message_activity_layout);
        clear = (Button) findViewById(R.id.clr_button);
        msg="";
        myRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                String value = dataSnapshot.getValue(String.class);
                msg= msg + " " +value;
            }
        });
    }
}

```

```

textView.setText(msg);
    }

```

The string msg is initialized to null string. Similar like previous activity the FirebaseDatabase and DatabaseReference is initialized. And here “sentence” child is used from firebase. In Value event listener the msg is caoncat which we get from the set value of “sentence” and msg is shown in the textview.

```

clear.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        msg = "";
        textView.setText(msg);
    }
});

```

The clear button erases the msg string as well as the textView. If we long press the layout we will return to the previous activity by intent call.

```

relativeLayout.setOnLongClickListener(new View.OnLongClickListener() {
    @Override
    public boolean onLongClick(View v) {
        Intent intent = new Intent(getApplicationContext(),ListActivity.class);
        startActivity(intent);
        return false;
    }
});

```

