Gesture to Speech Aid for Dumb People

*Project report submitted in partial fulfillment of the requirement for the degree of*

Bachelor of Technology

By

Kranthi Kumar Madhavaram (1510110197)

Rishabh Bhandari (1510110301)

Under supervision

of

Prof. R N Biswas

[SNU](http://snu.edu.in/)

DEPARTMENT OF ELECTRICAL ENGINEERING

SCHOOL OF ENGINEERING

SHIV NADAR UNIVERSITY

(November, 2018)

Candidate Declaration

I hereby declare that the thesis entitled “Gesture to Speech aid for Dumb people” submitted for the B Tech Degree program. This thesis has written in my own words. I have adequately cited and referenced the original sources.

(Signature)

Kranthi Kumar Madhavaram

(1510110197)

(Signature)

Rishabh Bhandari

(1510110301)

Date: November 28, 2018

CERTIFICATE

It is certified that the work contained in the project report titled “Gesture to Speech Aid for Dumb People” by Kranthi Kumar Madhavaram and Rishabh Bhandari has been carried out under my/our supervision and that this work has not been submitted elsewhere for a degree.

Signature of Supervisor(s)

Prof. R N Biswas

Electrical and Electronics Department

School of Engineering

Shiv Nadar University

November, 2018

ACKNOWLEDGEMENT

We are highly indebted to Prof. R N Biswas for his guidance and constant supervision as well as for providing necessary information regarding the project and also for his support in completing the project.

We would like to express our gratitude towards to Mr. Mohan for his kind co-operation and encouragement which helped us in the completion of the project.

We would like to express our special gratitude to the entire Electrical and Electronics Department for giving us their attention and time.

Abstract

India is home to a population of 1.3 billion people of which approximately 65 million are deaf and mute from birth. Our project aims at creating an aid to bridge the communication gap that is faced by these individuals with the gifted ones on a day to day basis. Starting with an initial plan to develop a simple pair of gloves that convert the sign language of the deaf and mute community, we had gone through understanding the problems and technical differences in sign languages. With these problems in mind we had limited our scope to working on specifically the Indian Sign Language (ISL), dictionary of which was recently launched in March 2018. Furthermore, after realising the increasing complexity in decoding the gestures with increasing database of signs we had further zeroed our scope to work on the English alphabets in the ISL. We had, therefore, developed a pair of gloves mounted with 10 flex sensors detecting the bent of particular joints, forming a binary pattern of the input, analysing the pattern, mapping with the alphabets and finally an algorithm to print the alphabets with minimum errors onto a mobile application. Although the plan was to extend the text to be output as speech limited knowledge of Android development set us back on this extension.

Contents

[1. Introduction 7](#_Toc531145852)

[1.1 Need and Motivation 7](#_Toc531145853)

[2 Basic Concept and Literature Review 8](#_Toc531145854)

[2.1 Components Used 8](#_Toc531145855)

[2.1.1 Arduino Uno 8](#_Toc531145856)

[2.1.2 Comparator Circuit 9](#_Toc531145857)

[2.1.2.1 Flex Sensors 9](#_Toc531145858)

[2.1.2.2 Potentiometer 10](#_Toc531145859)

[2.1.2.3 HC-05 Bluetooth Module 12](#_Toc531145860)

[2.2 Existing Prototypes: 13](#_Toc531145861)

[2.2.2 Sign Language Glove 14](#_Toc531145862)

[3 Methodology 16](#_Toc531145863)

[3.1 Algorithm 18](#_Toc531145864)

[4 Results and Discussion 21](#_Toc531145865)

[5 Conclusion 27](#_Toc531145866)

[6 References 28](#_Toc531145867)

[7 Appendix 30](#_Toc531145868)

# Introduction

India is home to a population of 1.3 billion people of which approximately 65 million are deaf and mute from birth. Our project aims at creating an aid to bridge the communication gap that is faced by these individuals with the gifted ones on a day to day basis.

## 1.1 Need and Motivation

Very recently (March, 2018) the government of India had launched an online dictionary specifically for the Indian Sign Language(ISL) [1,2]. This is counted as a huge step towards progress in this domain since up until December 2017, the Constitution of India did not even recognise such a thing as the ISL.

Just as much as vocal languages, the dialects of sign language vary geographically as well rising to at least 200 odd forms of sign language being used across the world[3]. There has a been a lot of research and prototypes to aid this gap across Europe and the US but never in India as there was no official language to work on. While a lot of solutions exist for the outside world none for us even though India harbours a significant portion of deaf and dumb around the world.

This recent introduction of the ISL dictionary gave us the motivation to work towards this humanitarian initiative.

# 2 Basic Concept and Literature Review

# 2.1 Components Used

## 2.1.1 Arduino Uno

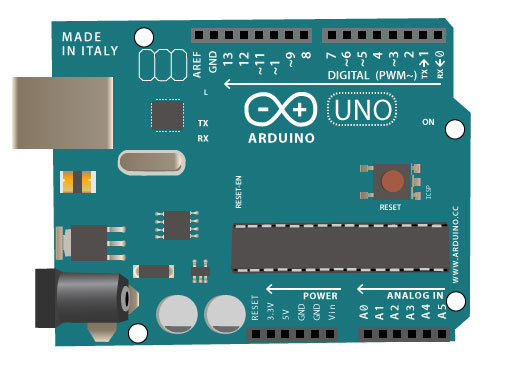


Figure 1 Arduino Uno

The Arduino UNO is an open-source microcontroller board based on the [Microchip](https://en.wikipedia.org/wiki/Microchip_Technology) [ATmega328P](https://en.wikipedia.org/wiki/ATmega328P) microcontroller and developed by [Arduino.cc](https://en.wikipedia.org/wiki/Arduino). The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits.[[1]](https://en.wikipedia.org/wiki/Arduino_Uno#cite_note-Makerspace-1) The board has 14 Digital pins, 6 Analog pins, and programmable with the [Arduino IDE](https://en.wikipedia.org/wiki/Arduino#Software) (Integrated Development Environment) via a type B USB cable.[[4]](https://en.wikipedia.org/wiki/Arduino_Uno#cite_note-priceton-4) It can be powered by a USB cable or by an external 9 volt battery, though it accepts voltages between 7 and 20 volts.

The Arduino Uno is priced around Rs. 750 and is the most widely used variant across all of the Arduino microcontrollers available. We are making use of two such boards in our project (one for each hand).

## 2.1.2 Comparator Circuit

## 2.1.2.1 Flex Sensors

Flex sensors are variable resistors in the shape of thin flexible strips whose resistance value changes with the degree of bending of the sensor. These sensors are manufactured in two sizes: 2.2” and 4.5”. We have used the 2.2” model for our project.

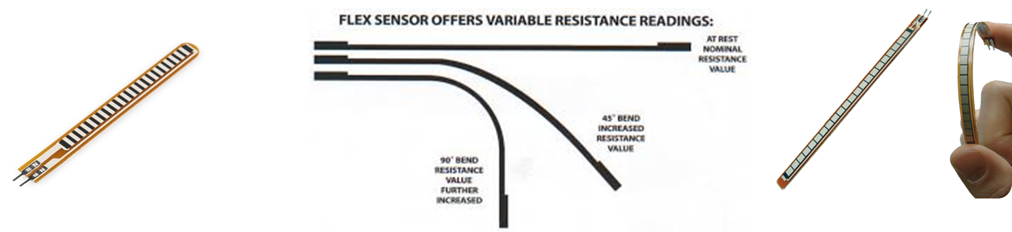


Figure 2 Flex sensor

Flex sensors are very commonly used with static resistors to create voltage dividers that can produce a variable voltage that can be easily read by a microcontroller’s analog-to-digital converter.

2.1.2.1.1 Working:

One side of the sensor is printed with a polymer ink that has conductive particles embedded in it. When the sensor is straight, the particles give the ink a resistance of about 30k Ohms. When the sensor is bent away from the ink, the conductive particles move further apart, increasing this resistance (to about 50k-70K Ohms when the sensor is bent to 90°, as in the diagram below).

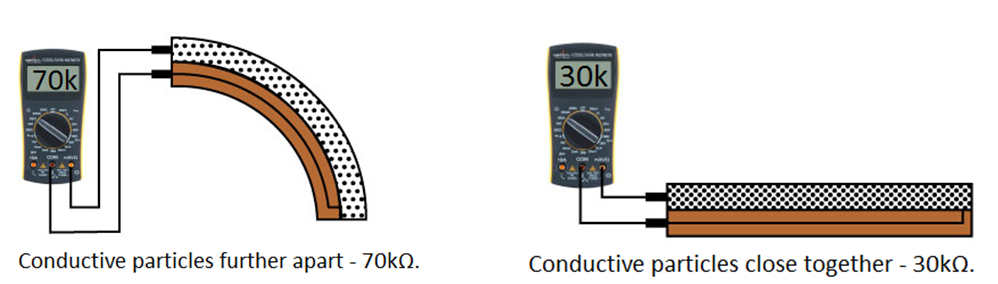


Figure 3 Variation in resistance of Flex sensor with bend

## 2.1.2.2 Potentiometer

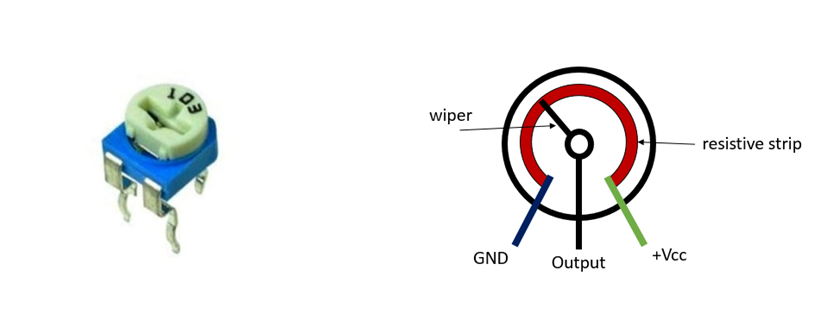


Figure 4 Potentiometer

A potentiometer is a three-terminal variable resistor with a rotatable contact which forms an adjustable voltage divider. When the ends are connected across a potential difference, the wiper terminal gives a voltage in between the voltages applied across its ends. The value of this voltage changes with the change in the value of resistance which can be adjusted by rotating the screw present on top of the potentiometer.

We have made use of a 100K ohm potentiometer for this project. The reason is so that it can be tuned according the base value of the flex sensor which is 30K ohms.

The presence of potentiometers allows us to set different thresholds (i.e. the point at which the output value changes from 1 to 0 or vice versa) for each of the joint upon which the flex sensor is present. This allows the user to tune the threshold point according to their preference. The comparator circuit can also be made without the use of a potentiometer but then the thresholds for the sensors would be fixed and would depend on the value of the static resistance in place on the potentiometer.

Working

A comparator circuit is present which compares the resistance value of the flex sensor to that of the potentiometer and gives an output of 1 or 0 depending on these values. The potentiometer is mounted with a screw on top which can be rotated to adjust the resistance of the potentiometer and hence the degree of bend of the flex sensor at which you want the output to change.

As different users will have slightly different gestures (in terms of bending along joints) for the same alphabets, the use of a separate comparator circuit with each of the flex sensors enables the user to configure the gloves accordingly.

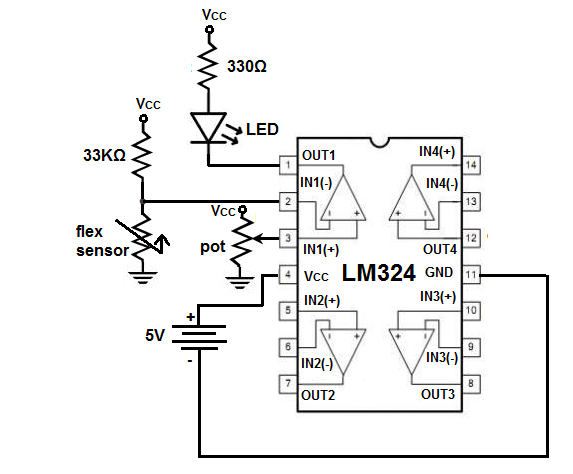


Figure 5 Comparator Circuit

We have made use of LM324 Voltage Comparator IC in this project. We have two inputs present in each voltage comparator circuit. The circuit compares the votlage at the  (+) input to that of the (-) input to give an output. We put the flex sensor in series with a 33k ohm resistor with the other end of the flex sensor grounded and the other end of the resistor  Vcc while the

The inverting input (-) has a 33k ohm resistance in series with a flex sensor with the other ends at Vcc and ground respectively.  This essentially acts as a potential divider circuit as the resistance value of the flex sensor is varied and we get a voltage between between 0-Vcc as an input to the comparator circuit.

The value of the flex sensor varies between ~ 30K ohm to 90K ohm when the flex sensor is bent from 0° to 90°. Thereby compelling us to use a resistance of value comparable to this in order for us to detect the bend with accuracy with the help of the potentiometer. Hence, we chose to use a 33K ohm resistor.

At the non-inverting input (+) we have connected a potentiometer and its other two ends between ground and Vcc with a screw on top to vary the resistance between 0-100K ohm. So, by turning the screw from 0-100K, we are increasing the voltage value at non-inverting input from 0 to Vcc.

If the voltage at the (-) input is higher than the (+) input voltage, the output of the circuit gets drawn to ground. Similarly, when the non-inverting input voltage is higher than the inverting input voltage, the output is drawn to Vcc.

The LM324 has four such op-amp circuits on each chip.

# 2.1.2.3 HC-05 Bluetooth Module

****HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup.

Serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. It uses CSR Bluecore 04-External single chip Bluetooth system with CMOS technology and with AFH(Adaptive Frequency Hopping Feature). It has the footprint as small as 12.7 mm x 27mm.

Figure 6 Bluetooth Module

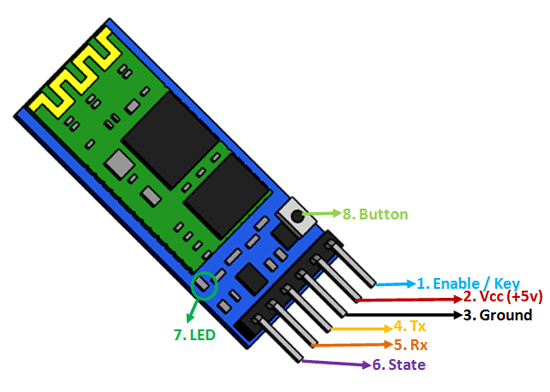


Figure 7 Bluetooth Module Pin Out

# 2.2 Existing Prototypes:

2.2.1 EnableTalk

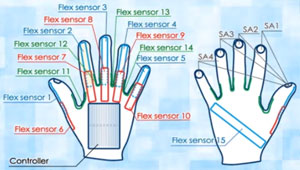


Figure 8 EnableTalk blueprint

EnableTalk is a pair of gloves mounted with 11 flex sensors on each hand along with accelerometer, gyroscope and a compass which together account for the motion of the glove through space and the degree of bending of their fingers. This product was made by team QuadSquad of Ukraine and were the winners of the Microsoft Imagine Cup Software Design Competition 2012[7].

Ten flex sensors account for the joints of the five fingers ( i.e. two joints on each finger) while the eleventh flex sensor account for the bending of the palm during gesturing. The accelerometer, gyroscope and the compass account for the 3D motion of the hands.

Enable Talk is a product which converts word gestures into speech as well. Word gestures take motion of the hands into account which hence posed the need for the Accelerometer, Gyroscope and Compass. On the other hand, making gestures character by character in a word i.e. fingerspelling does not account for this 3D motion and hence can be covered just by making use of flex sensors to account for the degree of bending

Despite having received a grant of $25000, the product never crossed the prototype stage as it was never able to account for facial expressions, movement of the torso or placement of hands relative to the body. These problems have to be addressed when we try to convert gesture into speech word by word. The same is not true for fingerspelling.

# Sign Language Glove



Figure 9 Sign Language Glove

A group of Cornell students made a similar device in 2014 for the International Sign which fingerspelled the English alphabet using 5 flex sensors and a few pressure sensors. This was worked on the Single Handed system of American Sign Language[8].

Also, in case of single handed fingerspelling, flex sensors alone are not enough to differentiate between various symbols. For example, below is a picture of the various fingerspelling gestures of the American Sign Language with one hand. The only difference between the gestures of ‘u’ and ‘v’ is whether the index and middle finger are in contact or not. Similar is the case for alphabet ‘r’ where the two fingers are twisted across each other. This is not something a flex sensor can differentiate between and hence pressure sensors are used.



Figure 10 Indian Single handed alphabets

One limitation of this project was that it used a single long flex sensor for each hand. While the sensor does tell us which finger is being bent, it does not recognise the joint across which the finger is being bent.

To further elaborate on this problem, let us look at the picture again. The difference between the index fingers of letter’s ‘x’ and ‘t’ or ‘s’ cannot be identified using one flex sensor across the whole finger. Even though the finger is the same, there is a change across the joint where the bending is taking place.

# 3 Methodology

Taking all such previous attempts into consideration, we decided to work on a device for the Indian Sign Language that can fingerspell the English Alphabet.

The advantage of working with the alphabet gestures is that the problems faced by the previous prototypes are taken care of automatically as these most of these gestures do not involve motion of the hand and none of them have any significance for the positioning of the hands with respect to the body.

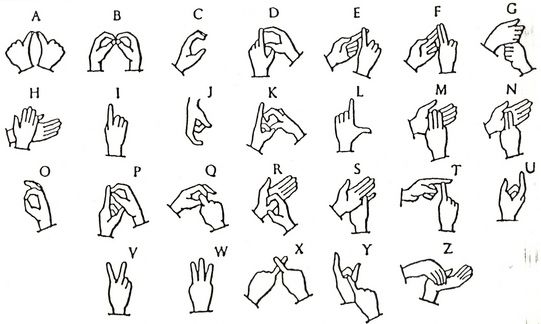


Figure 11 Indian Two-Handed alphabets

Taking the EnableTalk prototype as a reference, the initial plan was to appoint bits to the two joints of all fingers and one across the palm of each hand making a total of 22 flex sensors along with accelerometer and gyroscope to give the orientation of the hand. After eliminating the redundant bits we came down to 10 joints while making minimum alterations to the ISL.

We plan on having a linear structure wherein the gestures from the hand glove are converted to digital values using individual comparator circuits for each flex sensor and are transmitted to the microcontroller thereon using a Bluetooth module is sent to a remote device (laptop/smartphone) that outputs speech.

Each comparator outputs one bit which tells us whether the sensor is bent or not and the combination of all such output bits gives us a 10 bit serial key which is then processed to out the word paired to the key and sends it through the google TTS API and spells out the word

Reason for using more than one Flex Sensor per finger:

If we use one flex sensor per finger it would not be able to differentiate along which joint the finger is bent or whether it is bent along one joint or both.

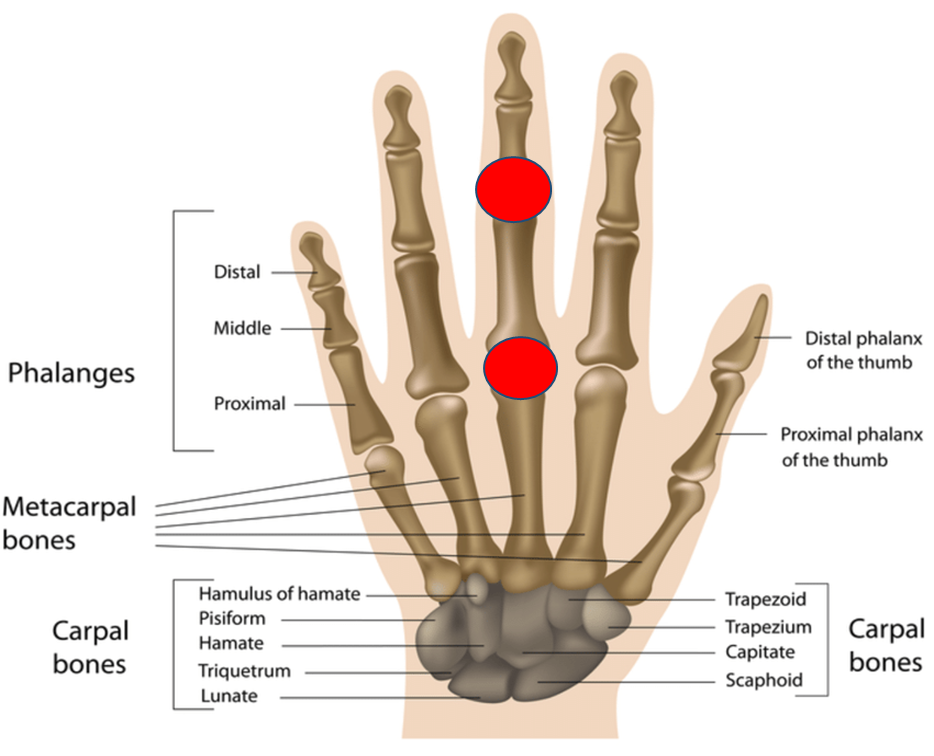
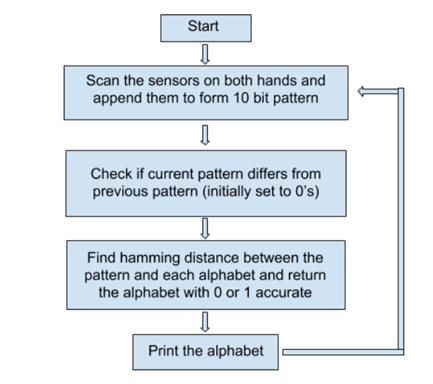


Figure 12 Anatomy of Human Hand

# **3.1 Algorithm**

Before beginning with the algorithm a crucial aspect was to finding the closeness between the alphabets i.e., the similarity between each pattern. Therefore, to find the same we have calculated the Hamming Distance between each alphabet with the rest 25 alphabets giving us a 26X26 matrix. Hamming Distance, when comparing two binary patterns is essentially the minimum number of bits to be toggled in order for a pattern to be same as the other pattern. For example while alphabet A is mapped to the pattern 0111111111, and alphabet D is mapped to 0111011110 the Hamming Distance between them is given as 2.

The necessity for finding Hamming Distance arose from the question of differentiating the alphabets with the pattern inputted from the sensors. Therefore, with this information we’ve figured out 4 cases in total where two alphabets differ by a single bit while the rest of the patterns are more than one bit apart thereby compelling us to have an accuracy of 9 bits at the least. With this in mind we have developed the following algorithm.



In the Start here, we initialise the bluetooth modules on both and hands and program them to pair with each other. Also, we initialise the variables and flags to 0’s accordingly while starting a transmission from the left hand module and keeping the right hand module as the receiver.

Then we read the data available from both the hands and form a 10 bit pattern and verify if this data is different from the previous scanned data. This is done to avoid data duplication. If it is different we call the search function and if not we input the data again.

In the search function we find the hamming distance between the input pattern and the patterns associated with each of the alphabet and look for the best match with an accuracy of 1 bit distance. This alphabet is then returned and transmitted via bluetooth to be printed on a mobile phone viz an app.

**3.2 Machine Learning Aspect**

We have 10 flex sensors each giving a 1-bit output as an input to the arduino for processing. These altogether make a 10-bit serial input which can have a total of 1024 (=2^10) possible outcomes which have to be mapped to at most 30 different characters (i.e. the 26 alphabets, blank space symbol, start-stop toggle etc.).

Since the scope of error is reasonably high while making said gestures as seen in the Fig. below, using ML we train the system in such a way that is recognises common errors and tries to give the correct output despite it.

Meaning, the system tries to identify the preset binary code mapped to one of the alphabets/symbols, which lies closest to the binary code of the actual gesture made. So for each such alphabet, we can train the system to remember all of the closest set of serial bits which are otherwise not mapped to any other character/symbol. In the end, we get a dataset wherein each of the alphabets have a wide range of serial bits are mapped to them.

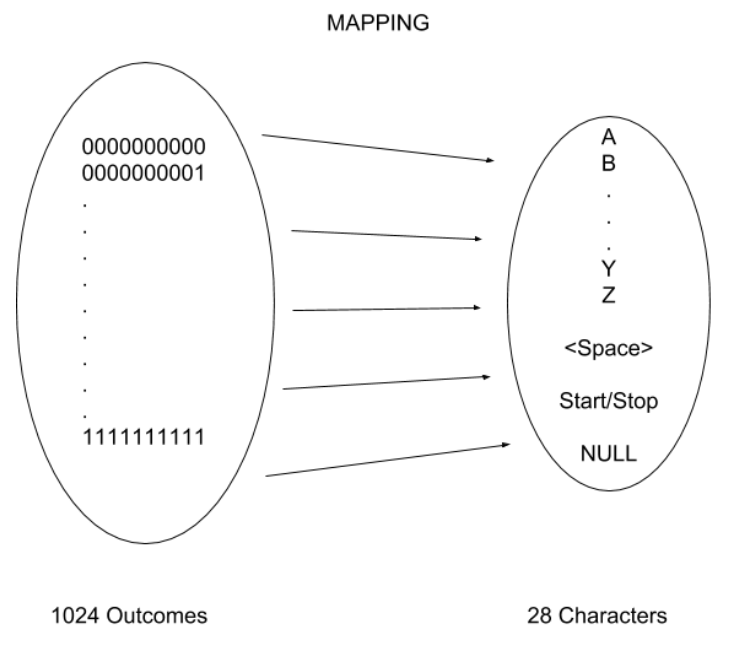


Figure 13 Mapping of Alphabets to patterns

# 4 Results and Discussion

As discussed, the first step for the project was to form the mapping of the alphabets to the 10 bit pattern formed when forming the respective gesture. The same was made by taking every relaxed joint as a ‘1’ and every bent joint as a ‘0’. The following shows the mapping of these alphabets more clearly.

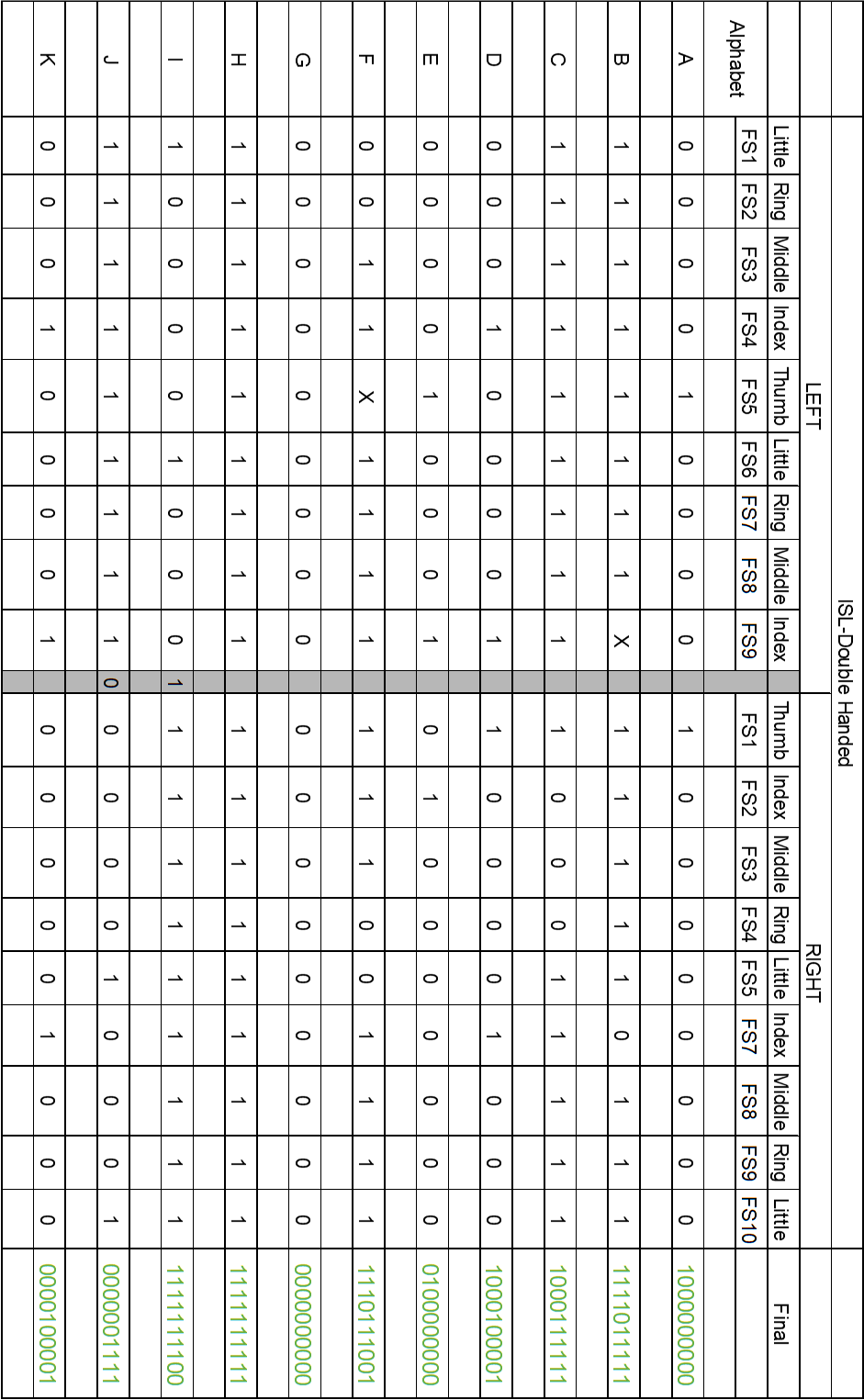


Figure 14 ISL Double Handed\_1

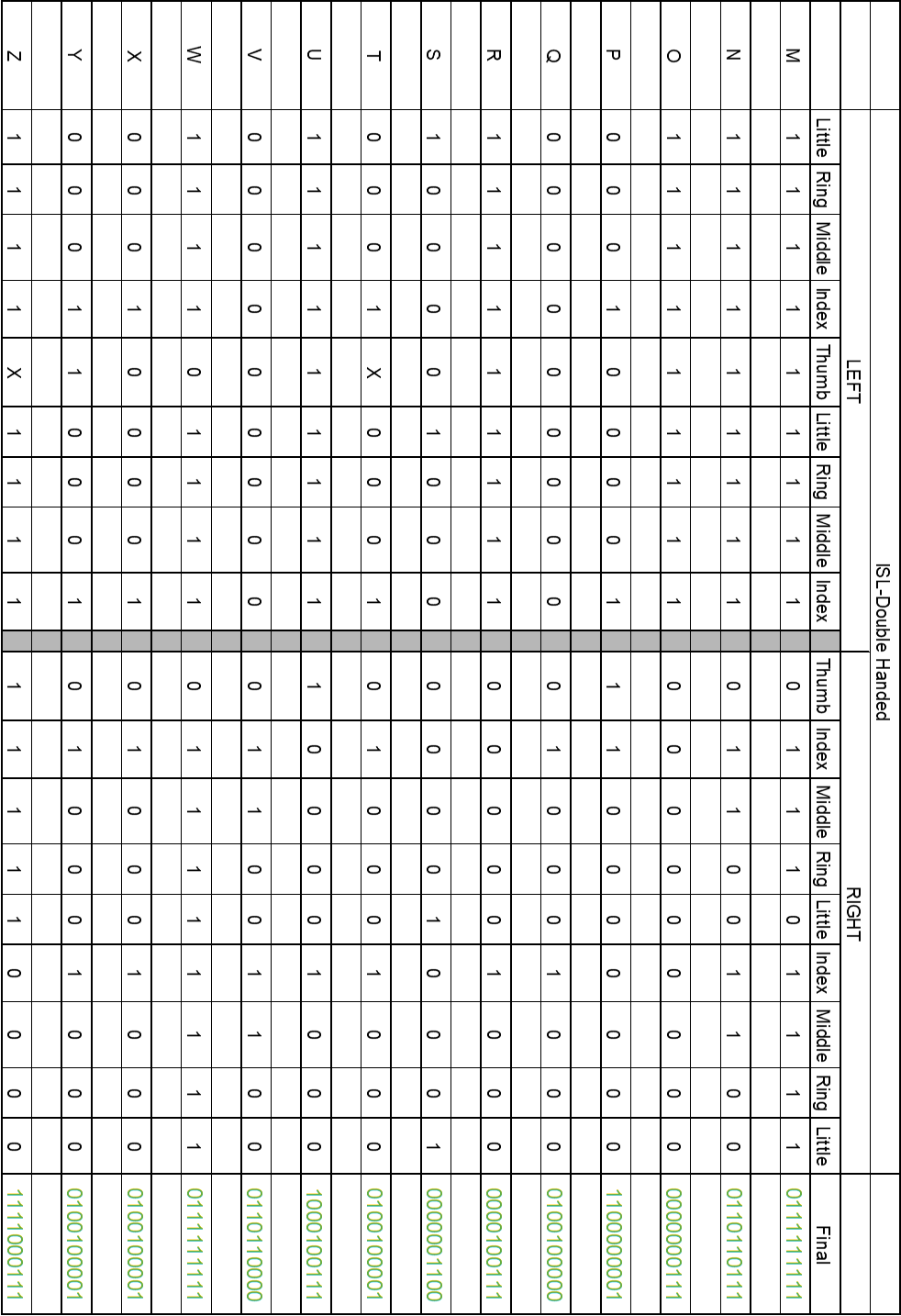


Figure 15 ISL Double Handed\_2

Furthermore the detailed analysis of these patterns was essential in developing the algorithms. Particularly, the Hamming Distance between these patterns. This was done with the help of a simple code on matlab to find the differences in the patterns of the alphabets giving us the following table.



Figure 16 Hamming Distance Test\

As seen from the above table the distance between the alphabets’ patterns is quite large (at least more than 2) in almost all cases barring a few. And a simple trial and error showed us that there are not more than 4-5 cases where the exceptions with Hamming Distance of 1 proved to be trouble worthy. Hence, we had developed the algorithm to map the alphabet to an accuracy of 9 bits to almost avoid the machine learning involvement.

Keeping in mind of all the requirements of the circuitry and the algorithms we have designed the gloves as seen below.

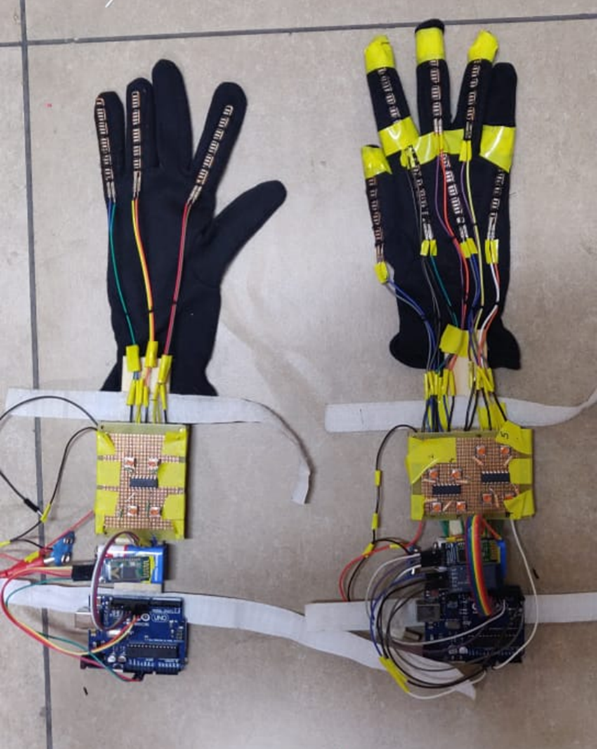


Figure 17 Final Setup

For the gloves to communicate with the mobile phone there are several smart phone applications for android on Google playstore. One such application that we’ve picked to work with is the Serial Bluetooth Terminal by Kai Morich. This application is used for receiving the data via Bluetooth on the right hand glove and printing the alphabets as received. Although the initial plan was to develop a mobile application that is to receive the test and convert it to speech, lack of knowledge in building the app and limited understanding of the Google Text to Speech API set us back in developing such an application. However, the following images show the alphabets printed w.r.t the gestures of the gloves.

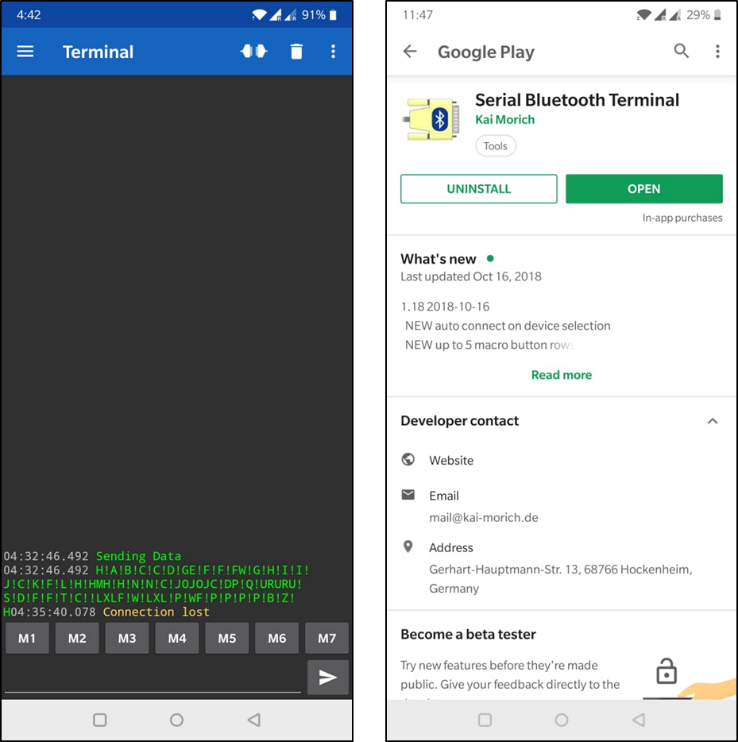


Figure 18 Image of the Mobile Application

# 5 Conclusion

Starting with an initial plan to develop a simple pair of gloves that convert the sign language of the deaf and mute community, we had gone through understanding the problems and technical differences in sign languages. With these problems in mind we had limited our scope to working on specifically the Indian Sign Language (ISL), dictionary of which was recently launched in March 2018. Furthermore, after realising the increasing complexity in decoding the gestures with increasing database of signs we had further zeroed our scope to work on the English alphabets in the ISL. We had, therefore, developed a pair of gloves mounted with 10 flex sensors detecting the bent of particular joints, forming a binary pattern of the input, analysing the pattern, mapping with the alphabets and finally an algorithm to print the alphabets with minimum errors onto a mobile application. Although the plan was to extend the text to be output as speech limited knowledge of Android development set us back on this extension.

# 

# 6 References

6.1 Text References

1. ISL Dictionary Launch | Indian Sign Language Research and Training Center (ISLRTC), [Online]. Available: <http://www.islrtc.nic.in/isl-dictionary-launch> [Accessed on 1st Oct, 2018]
2. Dictionary Indian Sign Language || FDMSE, RKMVERI, CBE (Deemed University), [Online]. Available: <http://www.indiansignlanguage.org/dictionary/> [Accessed on 1st Oct, 2018]
3. Melissa G Wallang, Introduction to Sign Language: The Visual Disctionary, New Delhi, Lakshi Publishers and Distributers, 2014.
4. Jon Sprouse, Psychology of language, Psych101/Lyn 155 [Online]. Available: <http://www.socsci.uci.edu/~jsprouse/courses/w13/150/lectures/3.12.sign.language> [Accessed on 2nd Oct, 2018]
5. Divya Deora, Nikesh Bajaj, “Indian Sign Language Recognition”, 2012 1st International Conference on Emerging Technology Trends in Electronics, Communication and Networking, Dec. 2012 [Online]. Available: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6470093>, Accessed on 3rd Oct, 2018
6. Manavender R, Jason J, Srirangaraj S, Venu G, Dinesh M, “A Framework for Hand Gesture Recognition and Spotting Using Sub-gesture Modeling”, 2010 International Conference on Pattern Recognition, Oct. 2010, [Online]. Available: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5597566> Accessed on 3rd Oct, 2018
7. Enable Talk, [Online]. Available: <http://enabletalk.com/>, Accessed On 2nd Oct, 2018
8. Monica L, Roberto V, “Sign Language Glove”, [Online]. Available: <https://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/f2014/rdv28_mjl256/webpage/>, Accessed on 3rd Oct, 2018.

6.2 Figure References

1. [www.arona.net](http://www.arona.net)
2. <https://learn.sparkfun.com/tutorials/flex-sensor-hookup-guide/all>
3. <http://edge.rit.edu/edge/P10010/public/PDF/Jan%20%20Flex%20Sensor%20Combined.pdf>
4. <https://potentiallabs.com>
5. <https://randomnerdtutorials.com/>
6. [www.nexiot.com](http://www.nexiot.com)
7. [www.github.com](http://www.github.com)
8. [www.fritzig.com](http://www.fritzig.com)
9. <http://enabletalk.com/>
10. <http://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/f2014/rdv28_mjl256/webpage/images/IMAG0146.jpg>
11. [https://encryptedtbn0.gstatic.com](https://encryptedtbn0.gstatic.com/images?q=tbn:ANd9GcTl13tJpWOsa4lxEHpodv_UNokf-JhGO5gUtyM92dOYQdp6H8SZuQ)
12. <http://www.deaftravel.co.uk/signs.php?id=27>
13. <https://www.researchgate.net>
14. [www.wikipedia.com](http://www.wikipedia.com)

# 7 Appendix

7.1 Code for Left Hand

#include <SoftwareSerial.h>

SoftwareSerial BT(9, 10); // TX, RX

char c;

int d;

int FSR1=13;

int FSR2=12;

int FSR3=11;

void setup() {

// put your setup code here, to run once:

Serial.begin(9600);

BT.begin(38400);

pinMode(FSR1, INPUT);

pinMode(FSR2, INPUT);

pinMode(FSR3, INPUT);

}

void loop() {

c=10;

BT.write(c);

delay(20);

Serial.print(c);

c=0;

d=digitalRead(FSR1);

c=d+48;

BT.write(c);

delay(20);

Serial.print(d);

c=0;

d=digitalRead(FSR2);

c=d+48;

BT.write(c);

delay(20);

Serial.print(d);

c=0;

d=digitalRead(FSR3);

c=d+48;

BT.write(c);

Serial.print(d);

// put your main code here, to run repeatedly:

}

6.2 Code For Right Hand

/\*

BT1 (Receiver-SLave) TX, RX to 6,5

BT2 (Transmitter to phone) TX, RX to 4,3

FS1-7 to 13-7

\*/

#include <SoftwareSerial.h>

SoftwareSerial BT1(6, 5); // TX, RX

char c=' ';

int AlphCode[27][10]={{1,0,0,1,0,0,0,0,0,0},//A

{1,1,0,1,1,1,1,1,1,1},//B

{1,0,1,1,1,0,1,1,1,1},//C

{1,0,1,1,0,0,0,0,0,1},//D

{0,1,0,1,0,0,0,0,0,0},//E

{1,1,1,1,1,0,1,0,0,1},//F

{0,0,0,1,0,0,0,0,0,0},//G

{1,1,1,1,1,1,1,1,1,1},//H

{1,1,1,1,1,1,1,1,0,0},//I

{0,0,0,1,0,0,1,1,1,1},//J

{0,0,1,1,0,0,0,0,0,1},//K

{1,1,1,1,0,0,0,1,1,1},//L

{0,1,1,1,1,1,1,1,1,1},//M

{0,1,1,1,1,0,0,1,1,1},//N

{0,0,0,1,0,0,0,1,1,1},//O

{1,1,0,1,0,0,0,0,0,1},//P

{0,1,1,1,0,0,0,0,0,0},//Q

{0,0,1,1,0,0,0,1,1,1},//R

{0,0,0,1,0,0,1,1,0,0},//S

{0,1,1,1,0,0,0,0,0,1},//T

{1,0,1,1,0,0,0,1,1,1},//U

{0,1,1,1,1,0,0,0,0,0},//V

{1,1,1,1,1,1,1,0,0,1},//W

{0,1,1,1,0,0,0,1,1,1},//X

{0,1,1,1,0,0,0,0,0,1},//Y

{1,1,0,1,0,1,0,1,1,1},//Z

{0,1,1,1,1,1,0,0,0,0}};

int Pattern[26]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

int FS1 = 13;

int FS2 = 12;

int FS3 = 11;

int FS4 = 10;

int FS5 = 9;

int FS6 = 8;

int FS7 = 7;

int FSR[10];

int PrevPattern[10]={0,0,0,0,0,0,0,0,0,0};

int PatternFlag=0;

char NewChar, OldChar=' ';

void setup() {

// put your setup code here, to run once:

BT1.begin(38400);

Serial.begin(38400);

Serial.println("Sending Data");

pinMode(FS1, INPUT);

pinMode(FS2, INPUT);

pinMode(FS3, INPUT);

pinMode(FS4, INPUT);

pinMode(FS5, INPUT);

pinMode(FS6, INPUT);

pinMode(FS7, INPUT);

}

void loop() {

// put your main code here, to run repeatedly:

while(!BT1.available());

c=BT1.read();

//Serial.print(c);

if(c==10){

while(!BT1.available());

c=BT1.read();

FSR[7]=c-48;

//Serial.print(FSR[7]);

while(!BT1.available());

c=BT1.read();

FSR[8]=c-48;

//Serial.print(FSR[8]);

while(!BT1.available());

c=BT1.read();

FSR[9]=c-48;

//Serial.print(FSR[9]);

FSR[0] = digitalRead(FS1);

FSR[1] = digitalRead(FS2);

FSR[2] = digitalRead(FS3);

FSR[3] = digitalRead(FS4);

FSR[4] = digitalRead(FS5);

FSR[5] = digitalRead(FS6);

FSR[6] = digitalRead(FS7);

// PrintPattern();

ScanPattern(); //Checks if pattern is same as old pattern

if(PatternFlag==0){

NewChar = GetChar();

if(NewChar!=OldChar){

//PrintPattern();

Serial.print(NewChar);

OldChar=NewChar;

}

}

}

delay(20);

}

void PrintPattern(){

int i;

for(i=0;i<10;i++){

Serial.print(FSR[i]);

}

Serial.println(" ");

}

void ScanPattern(){

int i,j;

PatternFlag=1;

for(i=0;i<10;i++)

{

if(FSR[i]!=PrevPattern[i]){

PatternFlag=0;

}

}

for(j=0;j<10;j++)

{

PrevPattern[j]=FSR[j];

}

return;

}

char GetChar(){

int i,j,ind,mapping;

for(i=0;i<26;i++)

{

Pattern[i]=0;

for(j=0;j<10;j++)

{

if(FSR[j]==AlphCode[i][j]){

Pattern[i]++;

}

}

}

int maxIndex = 0;

int maxValue = Pattern[maxIndex];

for( i = 1; i < 26; i++)

{

if(Pattern[i] > maxValue) {

maxValue = Pattern[i];

maxIndex = i;

}

}

if(maxValue>=10)

{

return 65+maxIndex;

}

else{

return '!';

}

}