MedCrave
Step into the World of Research

International Robotics & Automation Journal

Research Article

Open Access

CrossMark

Smart glove for sign language translation

Abstract

The development of a wearable device-based sign language translator is discussed in this study. It can translate sign language into both speech and text. The glove-based gadget can read the motions of one arm and five fingers. The system consists of an accelerometer and five flex sensors to measure finger bending and arm motions, respectively. Based on the interaction of these sensors, the gadget can recognize specific motions that in American Sign Language (ASL) correspond to the alphabet and convert them into speech and text using a mobile phone application. The hardware design of the device is mostly explained in this publication. The gadget displayed an average value of 0.6 s to translate a sign language into speech and text based on early trial results on six simple sign languages, demonstrating the applicability of the suggested device.

Keywords: ASL, flex sensor, MIT, smart glove, sign language

Volume 8 Issue 3 - 2022

Ahmed J Abougarair, Walaa A Arebi
Electrical & Electronics Engineering, University of Tripoli, Libya

Correspondence: Ahmed J Abougarair, Electrical & Electronic Engineering, University of Tripoli, Tripoli Libya, Tel +218916094184, Email a.abougarai@uot.edu.ly

Received: December 09, 2022 | **Published:** December 22, 2022

Introduction

Humans have used sign languages as a means of communication and message delivery, particularly for the deaf cultures. In order to convey a speaker's thoughts using sign language, hand shapes, hand orientation and movement, and facial expressions may all be used simultaneously. However, hardly many everyday individuals are conversant with sign language. Therefore, those who use sign language as a regular form of communication may find it challenging to converse with others or simply to express their views to others.[1] Therefore, as a result of the rapid advancement of technology, tools have been developed to enable deaf communities to hear or speak with others. There are several varieties of over-the-counter hearing aids available to help with deafness and other communication disorders, including behind-the-ear, in-the-ear, and canal aids.[2] Even while hearing aids are useful, using one of these devices may cause the user to feel uncomfortable or to hear background noise. As a result, scientists have been working on numerous techniques that can translate sign language gestures. The two main techniques are wearable technology and vision-based systems. Vision-based systems use feature extraction techniques in image processing to determine the movements of the hands and fingers.[3-5] Numerous further studies on the use of a vision-based system for sign language translation are described in.[6] The wearable will be coupled to a microcontroller in order to do away with cords, and the smart glove will use a variety of methods to interpret sign language into written or spoken words. In this research, Bluetooth-enabled mobile phones are utilized to display translated text and audio. The American Sign Language alphabet was used in this paper, as indicated in Figure 1. The major goal of this endeavor is to create a wearable device that will facilitate communication between deaf groups and the general public while taking user comfort into account. The following is a description of how the paper is structured, hardware component section II, Hardware circuit design in section III, Results and Discussion in section IV, The conclusion provided in V.

Hardware components

A smart glove was constructed to translate ASL characters into written and spoken characters in mobile phone application, the glove was constructed using five flex sensors one for each finger, the flex sensor is used to indicate the bend of a finger, an accelerometer was used in the back of the hand to indicate the position of the hand as horizontal or vertical, an Arduino nano was used for the code, and an HC-05 Bluetooth was used to connect the glove to the mobile phone.[7,8]

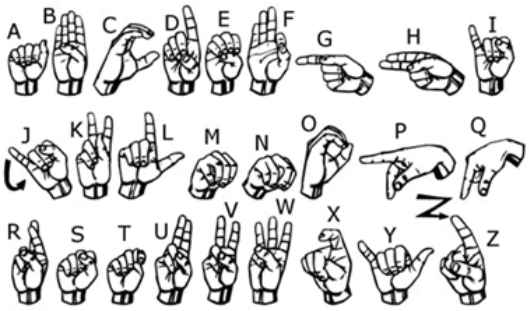

Figure 1 American Sign Language Alphabetic representation.

Flex sensor

A flex sensor as shown in Figure 2 is a type of sensor that gauges how much deflection or, alternatively, bending has occurred. Materials like plastic and carbon can be used to build this sensor. The sensor's resistance will alter when the plastic strip holding the carbon surface is turned aside. It is also known as a bend sensor as a result. According to their size, these sensors can be divided into two categories: 2.2-inch flex sensors and 4.5-inch flex sensors. Except for the operating principle, these sensors' size and resistance are different. 2.2-inch flex sensors were employed in this study. Applications for this kind of sensor include computer interface, rehabilitation, servo motor control, security systems, music interface, intensity control, and anywhere the user wants to adjust the resistance while bending. The flex sensor is a two-terminal device; it lacks any polarized terminals, such as capacitors or diodes, therefore there aren't any positive or negative terminals. This sensor needs 3.3V to 5V DC to activate, and this voltage can be obtained using any kind of interface. This sensor is utilized anywhere it is necessary to determine how much a device or instrument has bent, flexed, or changed its angle. This sensor's internal resistance varies roughly linearly with the sensor's flex angle. Thus, by attaching the sensor to the apparatus, we may obtain the flex angle within electrical parameter resistances.[9]

One flex was used to each finger as shown in figure 2.2, by connecting one pin to the GND, and the other pin with Arduino analog pin and to 5V through 10kΩ resistance, each flex was compensated for its minimum and maximum value according to finger bent, using the appropriate code the value was represented as range of angle from 0 to 90 degree, finally each letter of ASL alphabet was represented in code by using the appropriate combination of finger degrees.

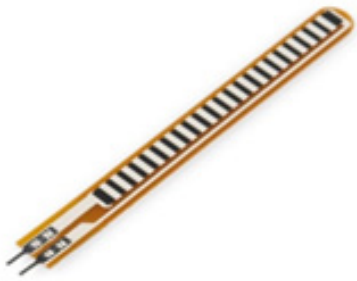

Figure 2 Flex sensor.

Accelerometer ADXL335

Accelerometers are often employed in low-power, low-cost motion and tilt detection applications, including mobile devices, gaming systems, disk drive protection, picture stabilization, and sports and fitness equipment. A silicon wafer serves as the foundation for the micro-machined MEMS (Micro-Electro-Mechanical System) accelerometer. Polysilicon springs support this construction. When the X, Y, and/or Z axes are accelerated, the structure may deflect. The capacitance between fixed plates and plates attached to the hanging structure varies as a function of deflection. The acceleration along that axis is proportional to this change in capacitance. The capacitance change is processed by the sensor and converted to an analog output voltage. A tiny, low-power, low-noise triple-axis MEMS accelerometer, which can measure static and dynamic acceleration brought on by motion, shock, or vibration, is at the heart of the ADXL335 module as shown in Figure 3.[10]

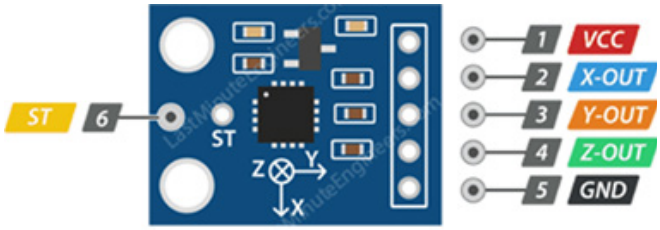

Figure 3 ADXL335 accelerometer pinout.

• VCC: provides the module with electricity. Connect it to your Arduino's 5V output.

• X-Out outputs: a voltage analog proportional to X-axis acceleration.

• Y-Out outputs: a voltage analog proportional to Y-axis acceleration.

• Analog voltage outputs along the Z axis that are proportional to acceleration.

The ground pin is labeled GND.

Bluetooth module HC-05

The HC-05 Bluetooth Module as shown in Figure 4 is made for setting up transparent wireless serial connections. It communicates using serial transmission, which makes interacting with a controller or PC simple. The HC-05 has two operating modes: AT Command mode, where default device settings can be altered, and Data mode, where it can send and receive data from other Bluetooth devices. Using the key pin, we can control the gadget in either of these two modes. The module was used to use a certain application to send the specified letter from the Arduino to the phone.[11]

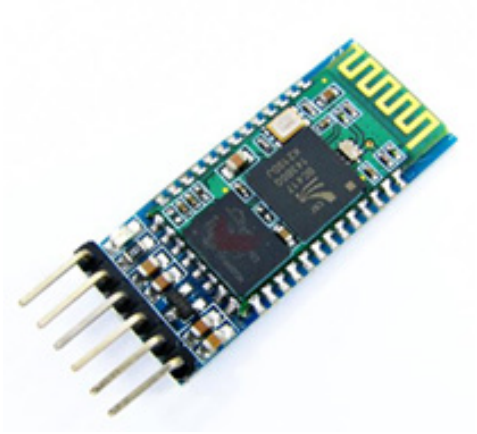

Figure 4 Bluetooth HC05 module.

Arduino nano

The smallest and most traditional breadboard-friendly Arduino board is called the Nano. The Arduino Nano in Figure 5 has a Mini-B USB connector and pin headers that make it simple to link it to a breadboard. Due to its compact size, the Arduino nano was employed in this work. The Arduino IDE was used to upload the code to the Arduino, which was then attached to the flex sensors, accelerometer, some of the digital pins, contact sensors, and the Bluetooth model.

Figure 5 Arduino nano.

The Arduino Nano 33 BLE can be used in place of the Arduino Nano in this configuration. Contrary to the Nano Every and Nano 33 IoT, the Arduino Nano 33 BLE is not designed around a Microchip CPU. Instead, it is equipped with a Nordic nRF52840, an Arm Cortex-M4F, built on top of a u-blox NINA B306 module. Better still, the Nano 33 BLE includes a 9-axis IMU. The Nano 33 BLE Sense is equipped with sensors that can detect color, motion, temperature, humidity, and more in addition to the ability to connect via Bluetooth® Low Energy. The method used to disseminate information on energy savings differs. Although it can handle a lot of data, Bluetooth quickly drains battery life and is significantly more expensive. Applications that do not exchange significant amounts of data and can operate on battery power for years at a lower cost use Bluetooth Low Energy.

Hardware circuit design

The hardware circuit shown in the Figure 6, the Arduino nano receives information from the accelerometer and flex sensors. The Arduino nano then uses this data to translate any gestures using the gesture recognition code that is described in the latter section of the paper. The commands are then sent to the output component by the Arduino once it has processed the input signals from the sensors. A total of five analog inputs from flex sensors and an accelerometer

can be handled by the board's 16 analog input pins. The results of the translation are processed, and the end result is simultaneously displayed on a mobile device and delivered through audio speech utilizing a speaker. The necessary application was coded using the MIT App Inventor website, and the structure code depicted in Figure 7 was created from scratch to satisfy the requirements.[12-15]

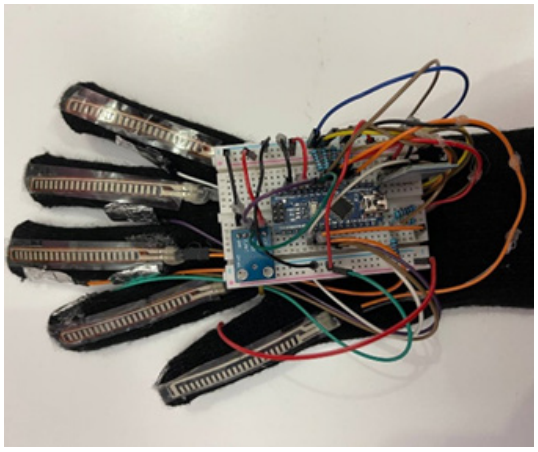

Figure 6 Hardware circuit.

Figure 7 Sample of the code.

Results and Discussion

The results of some of the ASL letters are listed in the Figure 8 to Figure 13, since the results include sounds a video of all the ASL alphabetic is recorded

Figure 8 The letter A.

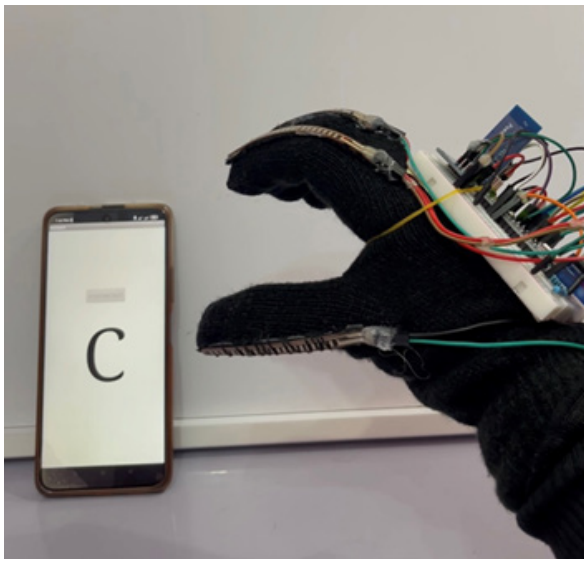

Figure 9 The letter C.

Figure 10 The letter F.

Figure 11 The letter H.

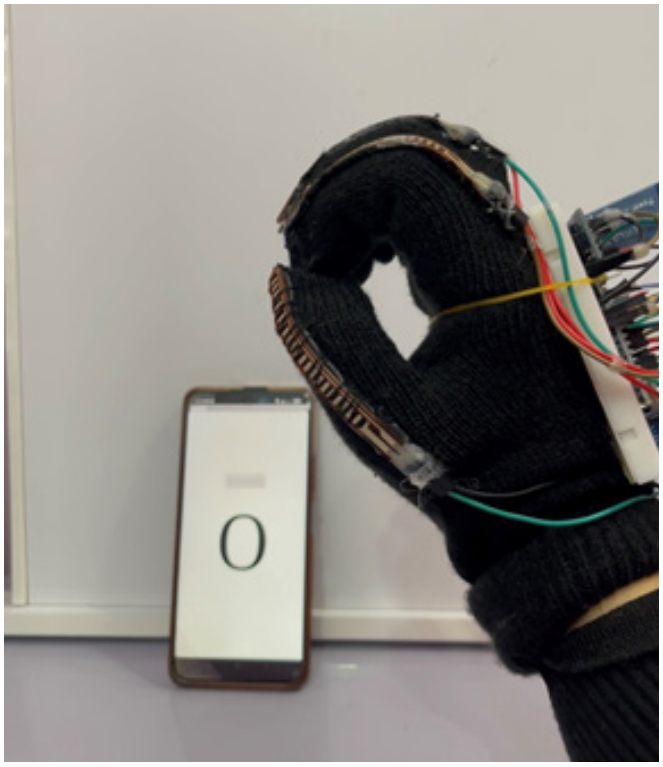

Figure 12 The letter O.

Figure 13 The letter S.

Conclusion

The creation of a glove-based sign language translator was discussed in this paper. Using two types of sensors, an accelerometer and five units of flex sensors, the devised device can read the movements of every finger and arm. The device's intricate hardware design as well as the findings of the experiments are presented in this paper. The results confirm the effectiveness of the suggested gadget by showing that it can almost entirely translate the movement of the arm's fingers into spoken and written English letters. This device's inability to distinguish between the letters U and V is one of its flaws. The gadget and microcontroller are currently connected by cables. According to the device's preliminary testing, the cords can obstruct hand movements. Therefore, a wireless connection using "E-TEXTILE" between the device and microcontroller may be used in the future to implement a solution to this issue. Since this was only a prototype, our main goal was to create a design that can help impaired individuals communicate more effectively. We did not decode any words for this prototype; we merely represented some alphabetic. This is a flaw in our paper that can be fixed in subsequent research.

Acknowledgments

None.

Conflicts of interest

Author declares that there is no conflict of interest.

Appendix

The code used for this project:

```
#include<SoftwareSerial.h>
SoftwareSerial bt(2,3); /* (Rx,Tx) */
Green and black thread for thumb //
int FLEX_PIN1 = A0;
int flexADC1 = 0;
int sensorMin1 = 745;
int sensorMax1 = 815;
Red and brown thread for the index finger //
int FLEX_PIN2 = A1;
int flexADC2 = 0;
int sensorMin2 = 740;
int sensorMax2 = 840;
Red and green string for the middle //
int FLEX_PIN3 = A2;
int flexADC3 = 0;
int sensorMin3 = 715;
int sensorMax3 = 820;
Yellow and brown thread for the ring finger //
int FLEX_PIN4 = A3;
int flexADC4 = 0;
int sensorMin4 = 760;
int sensorMax4 = 870;
Green and blue thread for the pinky //
int FLEX_PIN5 = A4;
int flexADC5 = 0;
int sensorMin5 = 740;
int sensorMax5 = 835;
```

```
int xpin = A5;
int xadc = 0;
int ypin = A6;
int yadc = 0;
char letter = ‘ ‘;
void setup() {
// put your setup code here, to run once:
Serial.begin(1200);
bt.begin(9600);
float flexADC1 = analogRead(FLEX_PIN1);
if(flexADC1<sensorMin1){sensorMin1=flexADC1;}
if(flexADC1>sensorMax1){sensorMax1=flexADC1;}
float flexADC2 = analogRead(FLEX_PIN2);
if(flexADC2<sensorMin2){sensorMin2=flexADC2;}
if(flexADC2>sensorMax2){sensorMax2=flexADC2;}
float flexADC3 = analogRead(FLEX_PIN3);
if(flexADC3<sensorMin3){sensorMin3=flexADC3;}
if(flexADC3>sensorMax3){sensorMax3=flexADC3;}
float flexADC4 = analogRead(FLEX_PIN4);
if(flexADC4<sensorMin4){sensorMin4=flexADC4;}
if(flexADC4>sensorMax4){sensorMax4=flexADC4;}
float flexADC5 = analogRead(FLEX_PIN5);
if(flexADC5<sensorMin5){sensorMin5=flexADC5;}
if(flexADC5>sensorMax5){sensorMax5=flexADC5;}
}
void loop() {
// put your main code here, to run repeatedly:
Serial.print(“angle1:”);
float flexADC1 = analogRead(FLEX_PIN1);
flexADC1 = constrain(flexADC1,sensorMin1, sensorMax1);
float angle1= map(flexADC1, sensorMin1, sensorMax1, 0, 90);
Serial.print(angle1);
Serial.print(“\t”);
Serial.print(“angle2:”);
float flexADC2 = analogRead(FLEX_PIN2);
flexADC2 = constrain(flexADC2,sensorMin2, sensorMax2);
float angle2= map(flexADC2, sensorMin2, sensorMax2, 0, 90);
Serial.print(angle2);
Serial.print(“\t”);
Serial.print(“angle3:”);
```

```
float flexADC3 = analogRead(FLEX_PIN3);
flexADC3 = constrain(flexADC3,sensorMin3, sensorMax3);
float angle3= map(flexADC3, sensorMin3, sensorMax3, 0, 90);
Serial.print(angle3);
Serial.print("\t");
Serial.print("angle4:");
float flexADC4 = analogRead(FLEX_PIN4);
flexADC4 = constrain(flexADC4,sensorMin4, sensorMax4);
float angle4= map(flexADC4, sensorMin4, sensorMax4, 0, 90);
Serial.print(angle4);
Serial.print("\t");
Serial.print("angle5:");
float flexADC5 = analogRead(FLEX_PIN5);
flexADC5 = constrain(flexADC5,sensorMin5, sensorMax5);
float angle5= map(flexADC5, sensorMin5, sensorMax5, 0, 90);
Serial.print(angle5);
Serial.print("\t");
Serial.print("x:");
xadc = analogRead(xpin);
Serial.print(xadc);
Serial.print("\t");
Serial.print("y:");
yadc = analogRead(ypin);
Serial.println(yadc);
bool horizantol (((xadc>=309)&&(xadc<=390))&&((yadc>=270)&&(yadc<=320))) ;
bool vertical = (((xadc>=410)&&(xadc<=462))&&((yadc>=310)&&(yadc<=405)));
bool equaliburium =(((xadc>=317)&&(xadc<=385))&&((yadc>=315)&&(yadc<=390)));
if((angle1<=40)&&(angle2>=60)&&(angle3>=72)&&(angle4>=68)&&(angle5>=72))
{Serial.println(letter);
letter = 'A';}
if((angle1>=35)&&(angle2<=15)&&(angle3<=15)&&(angle4<=15)&&(angle5<=15))
{Serial.println('B');
letter = 'B';}
if((angle1<20)&&((angle2>=30)&&(angle2<80))&&((angle3>=30)&&(angle3<85))&&((
angle4>=30)&&(angle4<85))&&((angle5>=30)&&(angle5<85)))
{Serial.println('C');
letter = 'C';}
if(((angle1>=30)&&(angle1<=70))&&(angle2<=15)&&(angle3>=60)&&(angle4>=40)&&(
angle5>=40)&& vertical)
{Serial.println('D');
letter = 'D';}
```

```
if((angle1>=85)&&(angle2>=85)&&(angle3>=85)&&(angle4>=85)&&(angle5>=85))
{Serial.println('E');
letter = 'E';}
if((angle1>=30)&&(angle2>=40)&&(angle3<=15)&&(angle4<=15)&&(angle5<=15))
{Serial.println('F');
letter = 'F';}
if((angle1<=30)&&(angle2<=15)&&(angle3>=55)&&(angle4>=55)&&(angle5>=60)&&hor
izantol)
{Serial.println('G');
letter = 'G';}
if((angle1>=40)&&(angle2<=15)&&(angle3<=15)&&(angle4>=55)&&(angle5>=50)&&hor
izantol)
{Serial.println('H');
letter = 'H';}
if((angle1>=30)&&(angle2>=70)&&(angle3>=55)&&(angle4>=55)&&(angle5<=30)&&ver
tical)
{Serial.println('I');
letter='I';}
if((angle1>=30)&&(angle2>=70)&&(angle3>=55)&&(angle4>=55)&&(angle5<=30)&&!ve
rtical)
{Serial.println('J');
letter='J';}
if((angle1<=30)&&(angle2<=15)&&(angle3<=15)&&(angle4>=55)&&(angle5>=55)&&ver
tical)
{Serial.println('K');
letter = 'K';}
if((angle1<=10)&&(angle2<=18)&&(angle3>=40)&&(angle4>=40)&&(angle5>=40)&&ver
tical)
{Serial.println('L');
letter = 'L';}
if(((angle1>=30)&&(angle1<=75))&&((angle2>=40)&&(angle2<80))&&((angle3>=40)&
&(angle3<80))&&((angle4>=50)&&(angle4<=85))&&(angle5>=85))
{Serial.println('M');
letter = 'M';}
if(((angle1>=30)&&(angle1<=70))&&((angle2>=40)&&(angle2<80))&&((angle3>=40)&
&(angle3<80))&&(angle4>85)&&(angle5>=85))
{Serial.println('N');
letter = 'N';}
if((angle1>=20)&&((angle2>=30)&&(angle2<80))&&((angle3>=30)&&(angle3<85))&&(
(angle4>=30)&&(angle4<85))&&((angle5>=30)&&(angle5<85))&&vertical)
{Serial.println('O');
```

```
letter = 'O';}
if((angle1<=40)&&(angle2<=15)&&(angle3<=30)&&(angle4>=55)&&(angle5>=55)&&equ
aliburium)
{Serial.println('P');
letter = 'P';}
if((angle1<=15)&&(angle2<=15)&&(angle3>=50)&&(angle4>=50)&&(angle5>=60)&&equ
aliburium)
{Serial.println('Q');
letter = 'Q';}
if((angle1>=30)&&(angle2<=10)&&((angle3>=10)&&(angle3<=20))&&(angle4>=55)&&(
angle5>=55)&&vertical)
{Serial.println('R');
letter = 'R';}
if(((angle1>=60)&&(angle1<85))&&(angle2>=70)&&(angle3>=70)&&(angle4>=70)&&(a
ngle5>=70))
{Serial.println('S');
letter = 'S';}
if(((angle1>=10)&&(angle1<=40))&&((angle2>=20)&&(angle2<=50))&&(angle3>=80)&
&(angle4>=80)&&(angle5>=80))
{Serial.println('T');
letter = 'T';}
if((angle1>=30)&&(angle2<=10)&&(angle3<10)&&(angle4>=55)&&(angle5>=55)&&vert
ical)
{Serial.println('U');
letter = 'U';}
if((angle1>=30)&&(angle2<=10)&&(angle3<10)&&(angle4>=55)&&(angle5>=55)&&vert
ical)
{Serial.println('V');
letter = 'V';}
if((angle1>=40)&&(angle2<=10)&&(angle3<=15)&&(angle4<=15)&&(angle5>=55))
{Serial.println('W');
letter = 'W';}
if((angle1>=30)&&((angle2>=15)&&(angle2<=40))&&(angle3>=30)&&(angle4>=30)&&(
angle5>=30)&&vertical)
{Serial.println('X');
letter = 'X';}
if((angle1<=10)&&(angle2>=44)&&(angle3>=40)&&(angle4>=40)&&(angle5<=15))
{Serial.println('Y');
letter = 'Y';}
if(((angle1>=30)&&(angle1<=70))&&(angle2<=15)&&(angle3>=60)&&(angle4>=40)&&(
angle5>=40)&&!vertical)
```

```
{Serial.println('Z');
letter='Z';}
bt.write(letter);
delay(20);
}
```

References

1. Stokoe WC. Sign language structure: An outline of the communicative systems of the American deaf 1960. *J Deaf Stud Deaf Educ*. 2005;10(1):3–37.
2. Igari S, Fukumura N. *Recognition of Japanese sign language words represented by both arms using multi-stream HMMs*. In Proc of IMCIC-ICSIT; 2016. 157–162 p.
3. Madhuri Y, Anitha G, Anburajan M. *Vision-based sign language translation device*. In Proc of the 2013 Int Conf on Information Communication and Embedded Systems (ICICES); 2013. 565–568 p.
4. Ahmed J Abougarair, Shashoa NAA, Elmelhi AM, et al. *Real time classification for robotic arm control based electromyographic signal*. 2022 IEEE 2st International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA2022); 2022. 23–25 p.
5. Dreuw P, Rybach D, Desealers T, et al. *Speech recognition techniques for a sign language recognition system*. InterSpeech; 2007. 2513–2516 p.
6. Huang TS, Wu Y. *Vision-based gesture recognition: a review*. In Gesture Workshop: Gif-sur-Yvette, France, vol. 1739 of LNCS; 1999. 103–115 p.
7. Jingqiu W, Ting Z. *An ARM-based embedded gesture recognition system using a data glove*. The 26th Chinese Control and Decision Conference; 2014. 1580–1584 p.
8. Shukor AZ, Miskon MF, Jamaluddin MH, et al. A new data glove approach for Malaysian sign language detection. *Procedia Computer Science*. 2015;76:60–67.
9. Ahmed S, Islam R. *Electronic speaking system for speech impaired people: Speak up*. In Proc of 2nd Int Conf on Electrical Engineering and Information & Communication Technology (ICEEICT); 2015.
10. Ambar R. Preliminary design of a dual-sensor based sign language translator device. In: Ghazali R, Deris M, Nawi N, et al. *Recent Advances on Soft Computing and Data Mining. SCDM 2018*. Advances in Intelligent Systems and Computing, Springer, Cham. 2018;700:353–362 p.
11. Flex sensor hookup guide; 2017.
12. Ahmed J Abougarair, Mohamed K Aburakhis. Design and implementation of smart voice assistant and recognizing academic words. *Int Rob Auto J*. 2022;8(1);27–32.
13. Ahmed J Abougarair. *Integrated controller design for underactuated nonlinear system*. IEEE, Second International Conference on Power, Control and Computer Technologies (ICPC2T-2022); 2022. 1–3 p.
14. Abougarair AJ, Shashoa NA, Aburakhis MK. Performance of anti-lock braking systems based on adaptive and intelligent control methodologies. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*; 2022.
15. Ahmed J Aboguarair. *Intelligent control design for linear model of active suspension system*. 30th International Conference on Microelectronics (IEEE): Tunisia; 2018.