CHAPTER 1

INTRODUCTION

The development of the most popular devices for hand movement acquisition, glove-based systems, started about 30 years ago and continues to engage a growing number of researchers. Communication means exchange of information; it becomes effective if all are using same media/language for conveying information. Generally mute people uses sign language for communication in which gestures are used to convey meaning instead of sound. It is nonverbal form of language uses gestures to convey thoughts gesture is a particular movement of the hands with a specific shape made out of them. Signs are used to communicate words and sentences to audience. In this system flex sensor plays the major role, flex sensors are sensors in which resistance changes according to degree of bending. This microcontroller and sensor based data glove helps to lower the communication gap between deaf, dumb and normal person. This paper contains the map to develop a gesture vocalize. Sign language recognition system mainly have two well-known approaches these are Image processing technique and another is microcontroller and sensor based data glove. These approaches are also known as vision based and sensor based techniques. In the image processing technique camera is used to capture the image/video, in this static images are analyzed and recognition of the image carried out using algorithms that produce sentences in the display. The algorithms used in vision based sign language recognition system are Hidden Markov Mode (HMM), Artificial Neural Networks (ANN) and Sum of Absolute Difference (SAD) Algorithm use to extract the image and eliminate the unwanted background noise. In sign language recognition system which uses image processing technique, image acquisition process has many environmental apprehensions such as background condition and lightning sensitivity. Higher resolution camera takes up more computation time and occupy more memory space, user always need camera forever and cannot implement in public place. These are the drawbacks of this system. In the approach data gloves are used for sign language recognition. In this user need to wear glove consist of flex sensor and motion tracker. Data are directly obtained from each sensor depends upon finger flexures and computer analysis sensor data with static data to produce sentences. It’s using neural network to improve the performance of the system.

CHAPTER 2

HARDWARE COMPONENTS

The different types of hardware components used are as follows.

2.1 Arduino:

The Arduino Uno is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

The power pins are as follows:

VIN. The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

5V. This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.

3V3. A 3.3-volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

GND. Ground pins.

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader) of memory. It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Each of the 14 digital pins on the Uno can be used as an input or output, using pin Mode (), digital Write (), and digital Read () functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 Kohms.

In addition, some pins have specialized functions:

Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip. **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attach Interrupt () function for details.

PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analog Write () function.

SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.

LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off. The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e.1024 different values). By default, they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analog Reference () function. Additionally, some pins have specialized functionality:

TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library. There are a couple of other pins on the board:

AREF. Reference voltage for the analog inputs. Used with analog Reference ().

Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

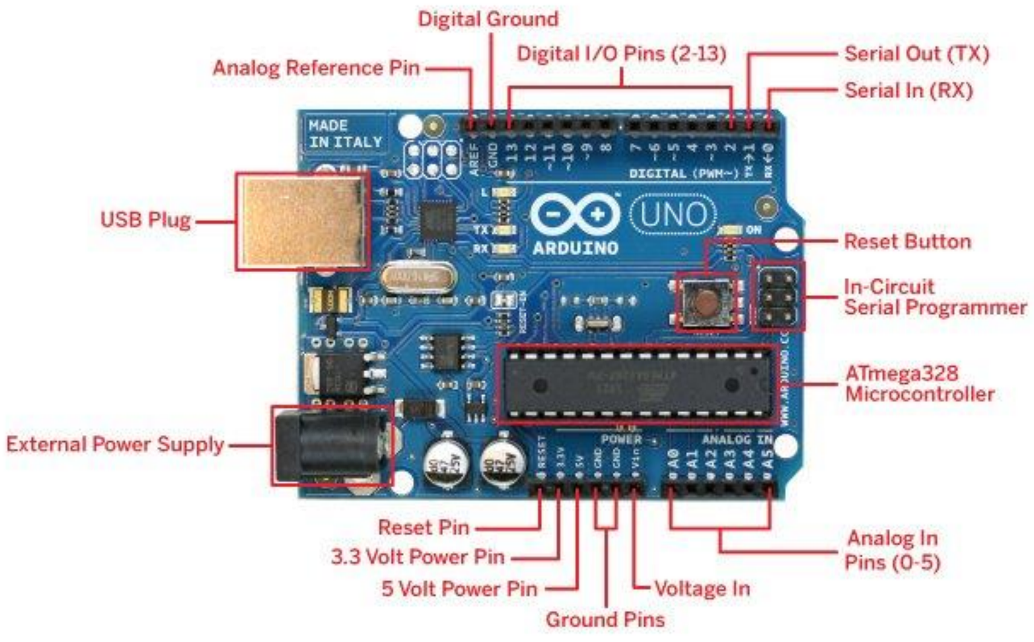


Figure: 2.1 Overview of Arduino UNO.

2.2 Flex Sensor:

A flex sensor or bend sensor is a sensor that measures the amount of deflection or bending. Usually, the sensor is stuck to the surface, and resistance of sensor element is varied by bending the surface. Since the resistance is directly proportional to the amount of bend it is used as goniometer, and often called flexible potentiometer. There are three types of flex sensors namely, Conductive ink based flex sensor, Fiber optic flex sensor, Capacitive flex sensor. This flex sensor is a variable resistor like no other. The resistance of the flex sensor increases as the body of the component bends. Sensors like these were used in the Nintendo Power Glove. They can also be used as door sensors, robot whisker sensors, or a primary component in creating sentient stuffed animals, Human Machine Interface devices and Security Systems, etc.

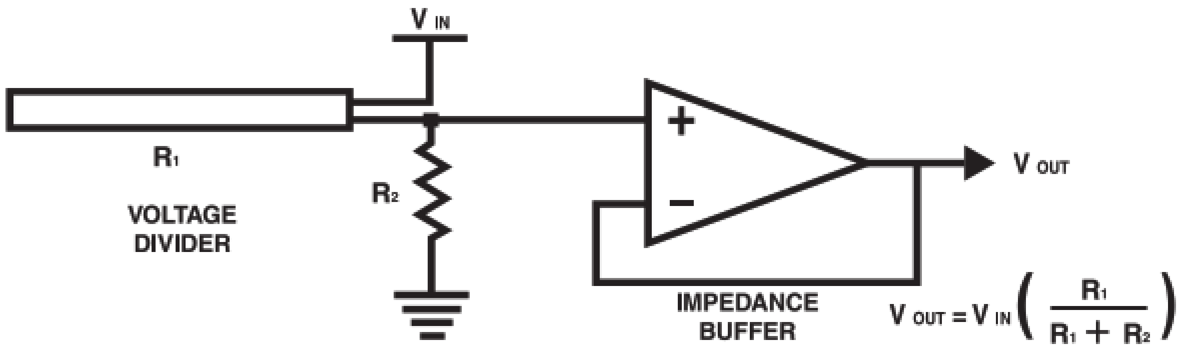


Figure:2.2 Basic circuit of a flex sensor.

2.2.1 Voltage divider:

A voltage divider (also known as a potential divider) is a passive linear circuit that produces an output voltage (Vout) that is a fraction of its input voltage (Vin). Voltage division is the result of distributing the input voltage among the components of the divider. A simple example of a voltage divider is two resistors connected in series, with the input voltage applied across the resistor pair and the output voltage emerging from the connection between them.

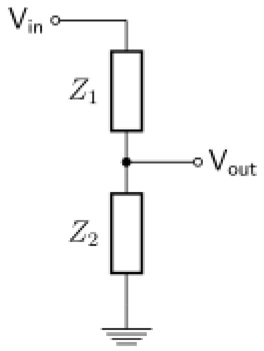


Figure:2.3 Voltage divider.

A voltage divider referenced to ground is created by connecting two electrical impedances in series, as shown in Figure 1. The input voltage is applied across the series impedances Z1 and Z2 and the output is the voltage across Z2. Z1 and Z2 may be composed of any combination of elements such as resistors, inductors and capacitors.

If the current in the output wire is zero then the relationship between the input voltage, Vin, and the output voltage, Vout, is:

$$V_{\mathrm{out}} = \frac{Z_2}{Z_1 + Z_2} \cdot V_{\mathrm{in}}$$

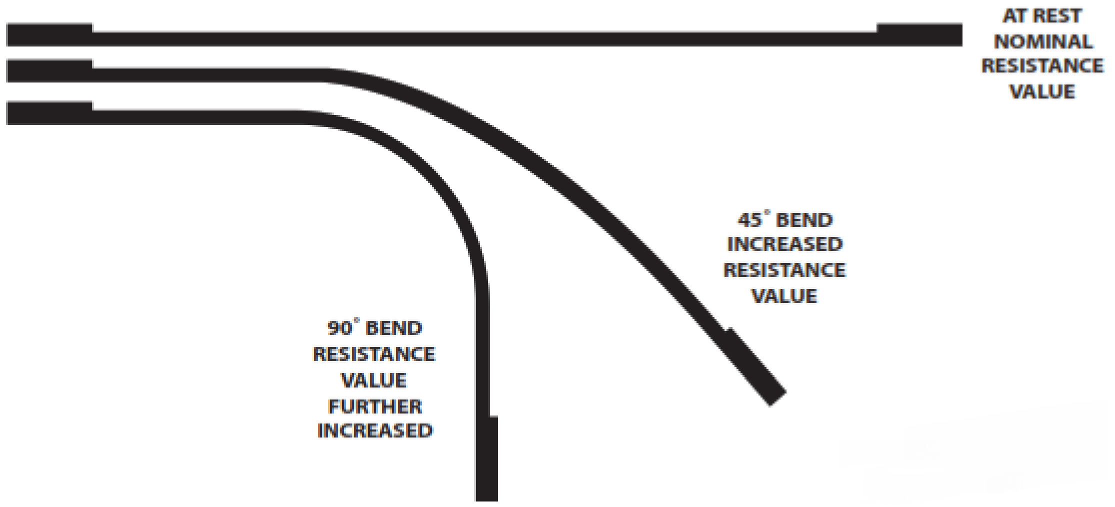


Figure: 2.4 Variable resistance readings of a flex sensor.

2.3 APR33A3 Voice Module:

The aPR33A series are powerful audio processor along with high performance audio analog-to-digital converters (ADCs) and digital-to-analog converters (DACs). The aPR33A series are a fully integrated solution offering high performance and unparalleled integration with analog input, digital processing and analog output functionality. The aPR33A series incorporates all the functionality required to perform demanding audio/voice applications. High quality audio/voice systems with lower bill-of-material costs can be implemented with the aPR33A series because of its integrated analog data converters and full suite of quality-enhancing features such as sample-rate convertor.

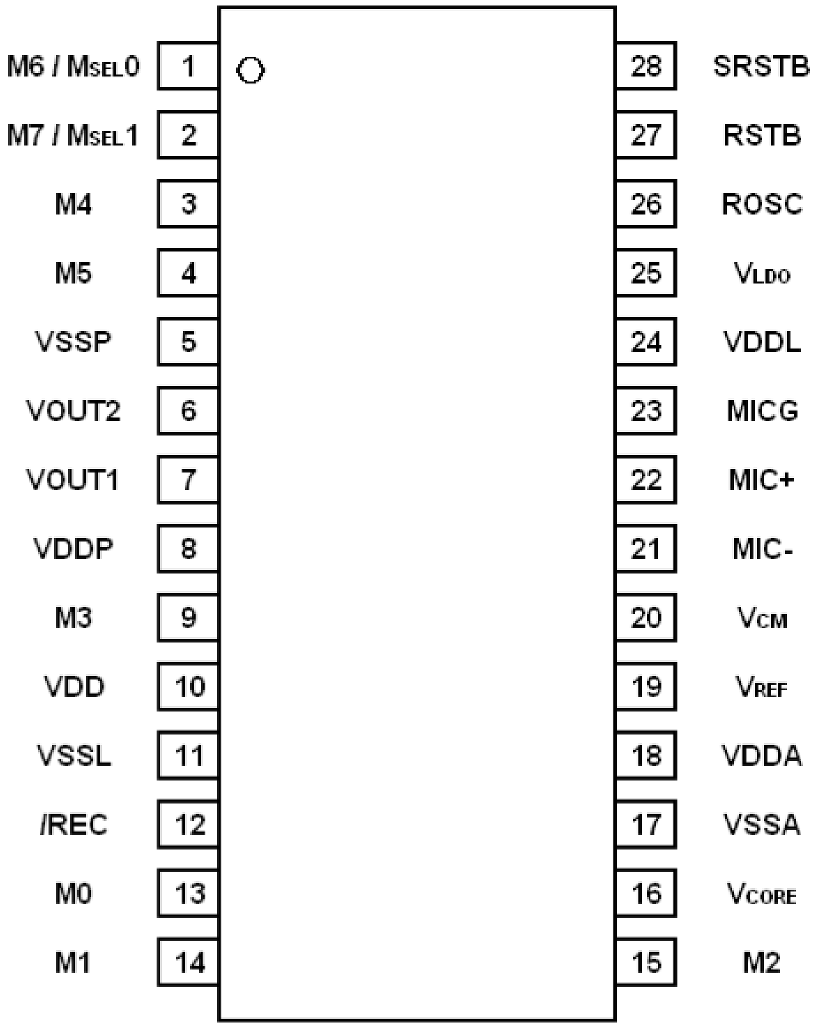


Figure: 2.5 Pin Configuration.

Figure: 2.6 APR33A3 Voice module.

2.3.1 Message mode:

One (1) Message Mode - The memory will be for 1 message when both MSEL0 and MSEL1 pin connected to VSS after chip reset.

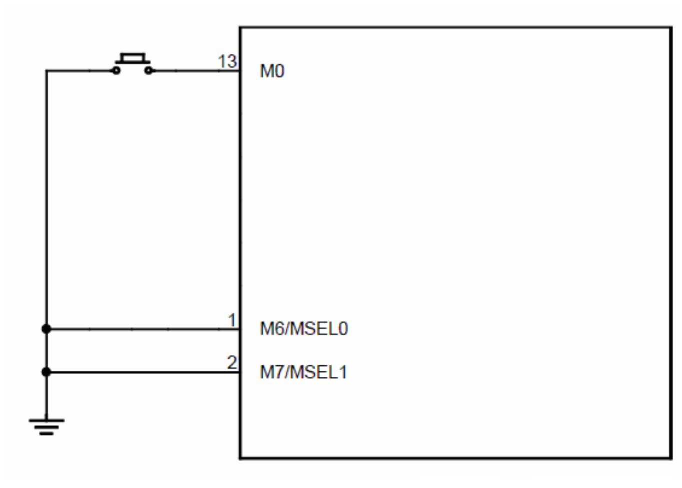


Figure:2.7 One Message mode

Two (2) Message Mode - The memory will be divided to 2 messages averagely when MSEL1 pin connected to VSS and MSEL0 pin float after chip reset.

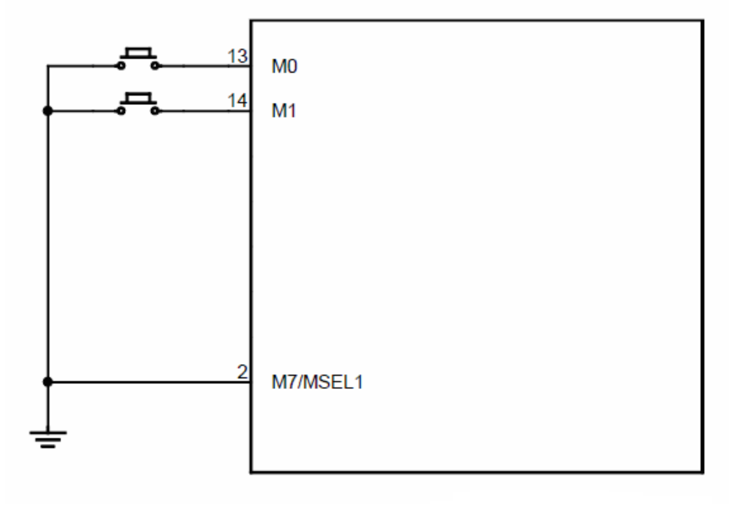


Figure:2.8 Two Message mode

Four (4) Message mode -The memory will be divided to 4 messages averagely when MSEL0 pin connected to VSS and MSEL1 pin float after chip reset.

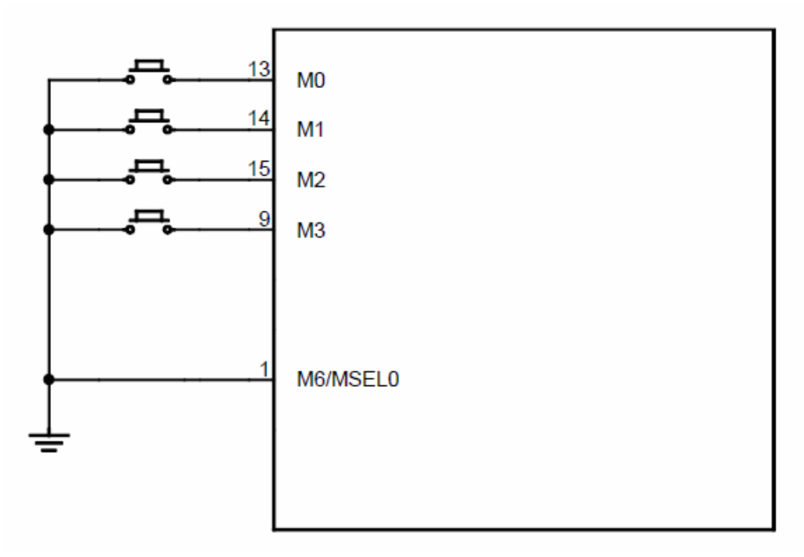


Figure:2.9 Four Message mode

Eight (8) Message mode - The memory will be divided to 8 messages averagely when both MSEL0 and MSEL1 pin float after chip reset.

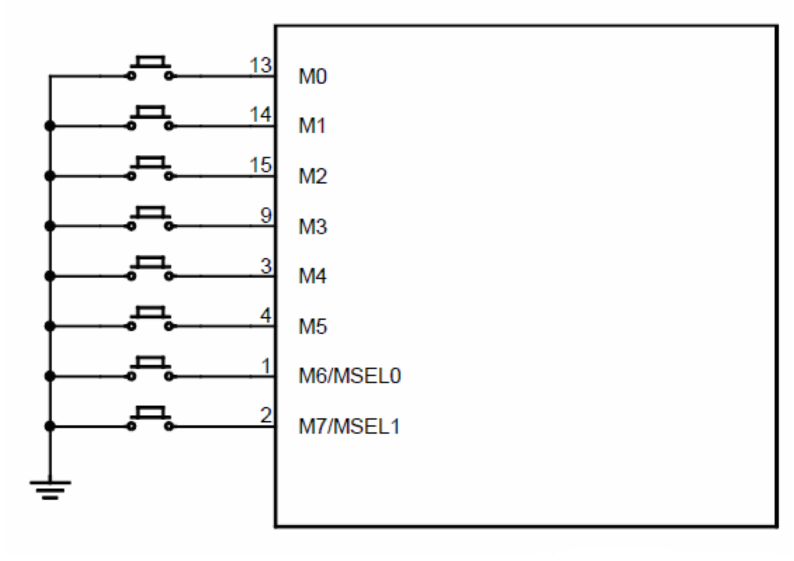


Figure:2.10 Eight Message mode

1.3.2 Record Message:

During the /REC pin drove to VIL, chip in the record mode. When the message pin (M0, M1, M2 … M7) drove to VIL in record mode, the chip will playback “beep” tone and message record starting. The message record will continue until message pin released or full of this message, and the chip will playback “beep” tone 2 times to indicate the message record finished. If the message already exists and user record again, the old one's message will be replaced.

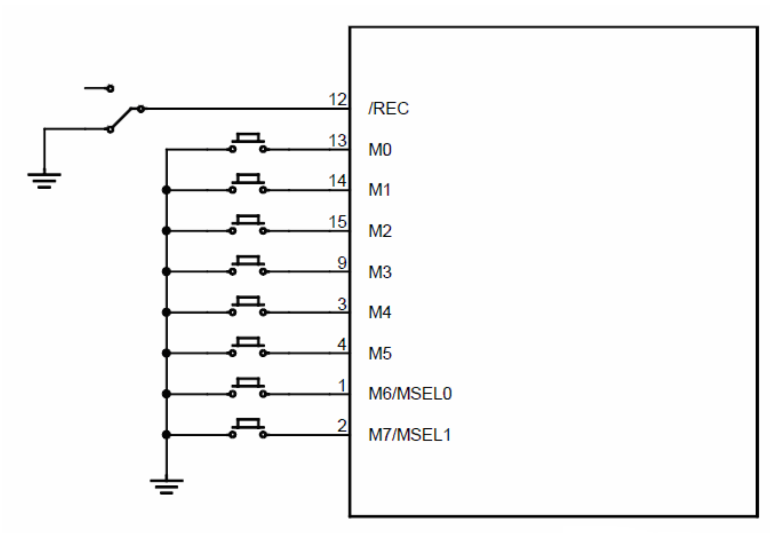


Figure: 2.11 Typical record circuit for 8-message mode.

2.3.2 Playback Message:

During the /REC pin drove to VIH, chip in the playback mode. When the message pin (M0, M1, M2 … M7) drove from VIH to VIL in playback mode, the message playback starting. The message playback will continue until message pin drove from VIH to VIL again or end of this message. After reset, /REC and M0 to M7 pin will be pull-up to VDD by internal resistor.

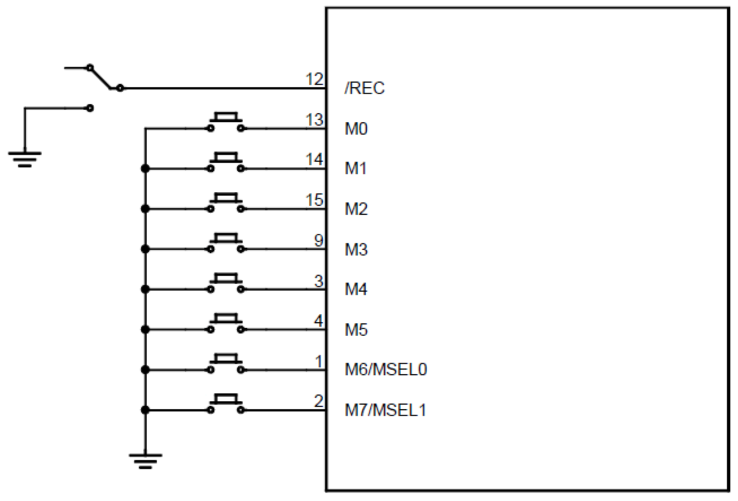


Figure: 2.12 Typical playback circuit for 8-message mode.

2.3.3 Reset:

APR33A series can enter standby mode when RSTB pin drive to low. During chip in the standby mode, the current consumption is reduced to ISB and any operation will be stopped, user also cannot execute any new operate in this mode. The standby mode will continue until RSTB pin goes to high, chip will be started to initial, and playback “beep” tone to indicate enter idle mode. User can get less current consumption by control RSTB pin specially in some application which concern standby current.

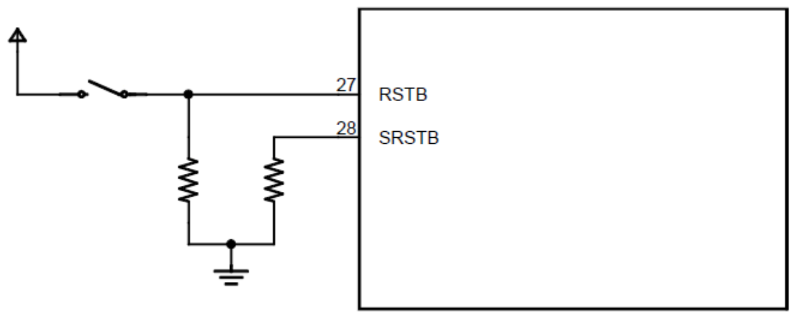


Figure: 2.13 Reset circuit.

The above are the main components of the and the remaining are:

- Resistors
- Gloves
- Jumper wires
- LCD 16*2
- Speaker
- Breadboard or PCB

CHAPTER 3

SOFTWARE

3.1 Arduino IDE:

A program for Arduino may be written in any programming language with compilers that produce binary machine code for the target processor. Atmel provides a development environment for their microcontrollers, AVR Studio and the newer Atmel Studio.

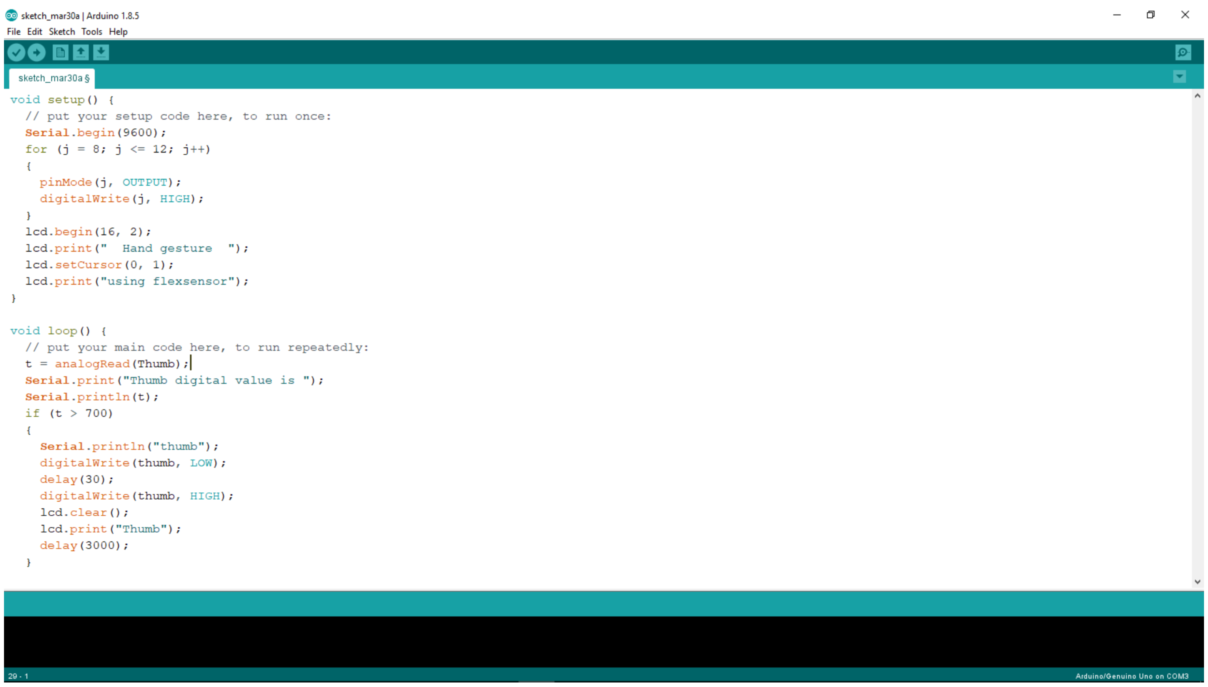
The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform application written in the programming language Java. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one-click mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus. A program written with the IDE for Arduino is called a sketch.[56] Sketches are saved on the development computer as text files with the file extension .ino. Arduino Software (IDE) pre-1.0 saved sketches with the extension. pde.

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware. The open-source nature of the Arduino project has facilitated the publication of many free software libraries that other developers use to augment their projects.

3.1.1 Program structure:

A minimal Arduino C/C++ program consist of only two functions:

- **setup ()**: This function is called once when a sketch starts after power-up or reset. It is used to initialize variables, input and output pin modes, and other libraries needed in the sketch.
- **loop ()**: After setup () has been called, function loop () is executed repeatedly in the main program. It controls the board until the board is powered off or is reset.



```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  for (j = 8; j <= 12; j++)
  {
    pinMode(j, OUTPUT);
    digitalWrite(j, HIGH);
  }
  lcd.begin(16, 2);
  lcd.print("  Hand gesture  ");
  lcd.setCursor(0, 1);
  lcd.print("using flexsensor");
}

void loop() {
  // put your main code here, to run repeatedly:
  t = analogRead(Thumb);
  Serial.print("Thumb digital value is ");
  Serial.println(t);
  if (t > 700)
  {
    Serial.println("thumb");
    digitalWrite(thumb, LOW);
    delay(30);
    digitalWrite(thumb, HIGH);
    lcd.clear();
    lcd.print("Thumb");
    delay(3000);
  }
```

Figure: 3.1 Arduino IDE showing the structure of setup and loop program.

3.1.2 Code:

The following is the code for the project.

#include <LiquidCrystal.h>

#define Thumb A4

#define Index A3

#define Middle A2

#define Ring A1

```
#define Small A0
#define thumb 8
#define index 9
#define middle 10
#define ring 11
#define small 12
Liquid Crystal lcd(2,3,4,5,6,7);
unsigned int j,t,i,m,r,s;
void setup () {
  // put your setup code here, to run once:
  Serial.begin(9600);
for(j=8;j<=12;j++)
{
pinMode(j,OUTPUT);
digitalWrite(j,HIGH);
}
lcd.begin(16,2);
lcd.print("  Hand gesture  ");
lcd.setCursor(0,1);
lcd.print("using flexsensor");
}
void loop ()
 {
  // put your main code here, to run repeatedly:
t=analogRead(Thumb);
Serial.print("Thumb digital value is ");
Serial.println(t);
if(t>800)
{
  Serial.println("thumb");
  digitalWrite(thumb,LOW);
  delay (30);
  digitalWrite(thumb,HIGH);
  lcd.clear();
  lcd.print("Hello !");
  delay (3000);
  }
i=analogRead(Index);
Serial.print("Index digital value is ");
Serial.println(i);
if(i>850)
{
  Serial.println("index");
  digitalWrite(index,LOW);
  delay (30);
  digitalWrite(index,HIGH);
  lcd.clear();
  lcd.print("I Am Sudheer.");
  delay (3000);
  }
m=analogRead(Middle);
Serial.print("Middle digital value is ");
Serial.println(m);
if(m>880)
{
  Serial.println("middle");
  digitalWrite(middle,LOW);
  delay (30);
  digitalWrite(middle,HIGH);
  lcd.clear();
  lcd.print("She Is Sindhuja.");
  delay (3000);
  }
r=analogRead(Ring);
Serial.print("Ring digital value is ");
Serial.println(r);
if(r<1010)
{
  Serial.println("ring");
  digitalWrite(ring,LOW);
  delay (30);
  digitalWrite(ring,HIGH);
  lcd.clear();
  lcd.print("She Is Bhanu.");
  delay (3000);
  }
s=analogRead(Small);
Serial.print("Small digital value is ");
Serial.println(s);
if(s>900)
{
  Serial.println("small");
  digitalWrite(small,LOW);
  delay (30);
  digitalWrite(small,HIGH);
  lcd.clear();
  lcd.print("Thank You !");
  delay (3000);
  }
  delay (4000);
}
```

CHAPTER 4

WORKING

In general, deaf/dumb people have difficulty in communicating with others who do not understand sign language. The Hand gesture glove is a normal, cloth driving glove fitted with flex sensors. The sensors output a stream of data that varies with degree of bend made by the fingers. Flex sensors are sensors that change in resistance depending on the amount of bend on the sensor. They convert the change in bend to electrical resistance - the more the bend, the more the resistance value. The output from the sensor is converted to digital and processed by using microcontroller and then it responds in the voice using speaker and it also can be displayed on the LCD screen.

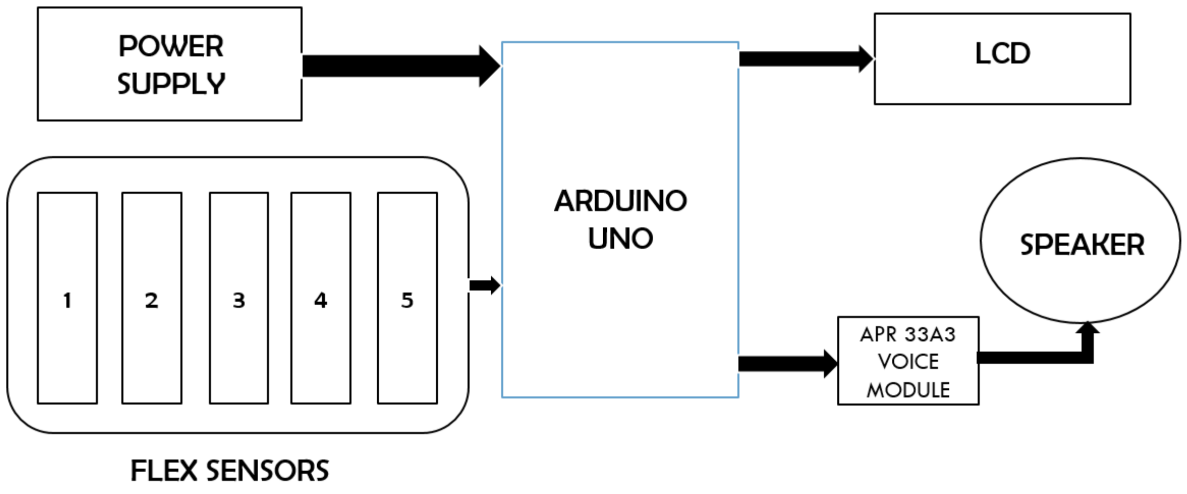


Figure: 4.1 Block Diagram.

Since each flex sensor has resistance that varies depending on how much the finger is bent, we attach each flex sensor as part of a voltage divider circuit in order to obtain a corresponding voltage that can then be input into the Arduino.

$$Vout = vin*(R1/(R1+R2)).$$

We determine a good value for R1 by analyzing expected values from the flex sensor. Each one has a flat resistance of 10kohms and a maximum expected resistance of about 27 Kohms.

In order to obtain the maximum range of possible output voltages from the divider circuit given an input voltage of 5v. They convert the change in bend to electrical resistance - the more the bend, the more the resistance value. The user creates data sets of information from the glove for each gesture that should eventually be translated to predict later at run time what a user is signing. The output from the sensor is converted to digital and processed by using microcontroller and then it responds to the voice in voice module using speaker and it a can also be displayed on the LCD screen.

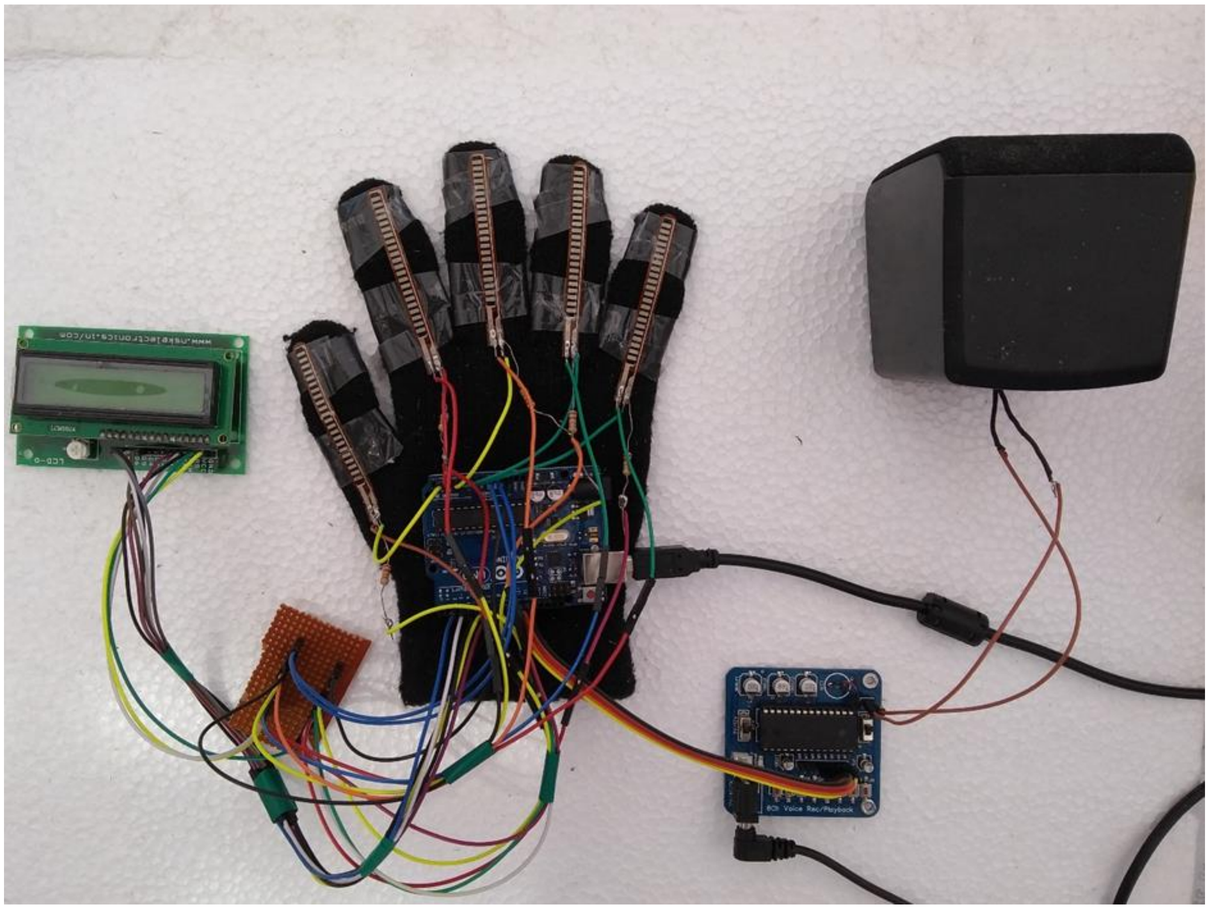


Figure: 4.2 Image of the project with all connections.

CHAPTER 5

OUTCOME

Speed of execution: we determined how quickly we could send data to the pc by sending data serially and increasing the send rate until we noticed a difference between the rate at which data was being received and the rate at which data was being sent. We then reduced the send frequency back to a reasonable value and converted this into a loop interval)3milliseonds).

Safety and interference with other devices: Our devices use a low voltage environment, and extremely low frequency communication. The sensors are well-attached, and there are no sharp edges.

Usability: The glove can be used by anyone who fits into it, they would only have to train on it and generate new datasets.

Accuracy: The averaging we do at each interval helps to account for any noise or glitches that the flex sensors ae sometimes prone to. The accuracy also sometimes limited by the size of the person's hand. The accuracy of each flex sensor is limited beyond a certain point. Smaller hands will result in a larger degree of bend.

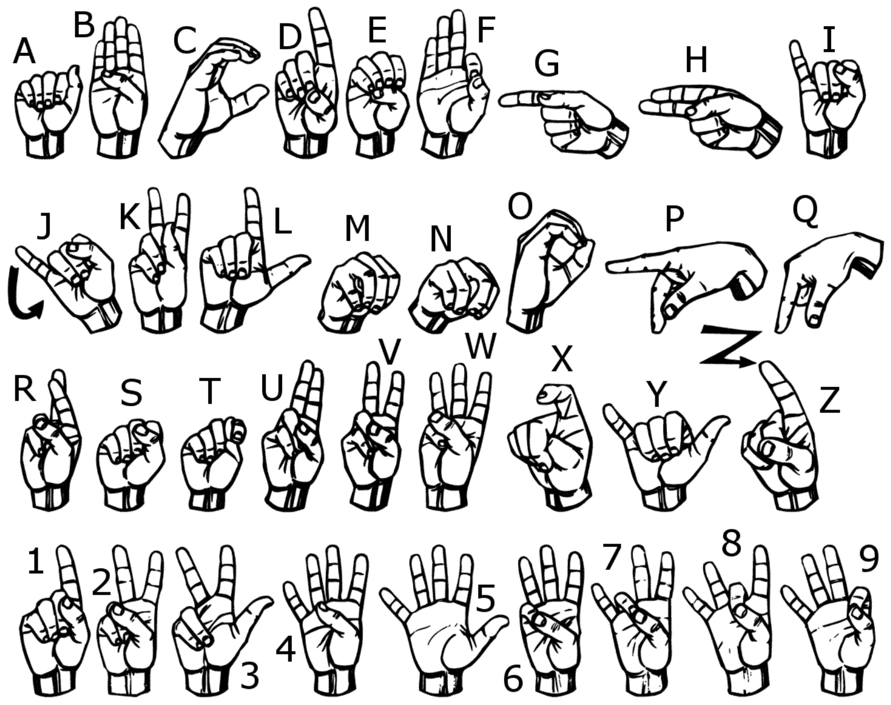


Figure:5 Predefined Sign languages.

CHAPTER 6

CONCLUSION

The project was able to meet our expectations quite well. Our initial goal was to create a system capable of recognizing and classifying gestures and we were able to do so with more than 98% average accuracy across all classes. While we did not have solid time requirement for the rate of prediction, the resulting speed made using the glove comfortable and it did not feel sluggish. In addition, we hope to make the glove wireless, therefore allowing it to communicate with phones and other devices and making the system truly portable. For gestures we adhered to the signs established by communities such as American sign language standards.

REFERENCES

1. V.Padmanabham and M.Sornalatha, “Hand gesture recognition and voice conversion system for dumb people” , International Journal Of Scientific and Engineering Research (IJSER), volume 5,issue 5,may-2014 ISSN 2229-5518.
2. Allada Ramesh, B.V. Reddy, “Gesture Recognization & Conversion for Deaf and Dumb People”, International Journal of innovative Technologies (IJITECH) - ISSN 2321-8665 Vol.04, Issue.03, March-2016, Pages:0525-0526 (WWW.IJITECH.ORG).
3. Ruize Xu, Shengli Zhou, And Wen J. Li, Mems Accelerometer Based Nonspecific-User Hand Gesture Recognition IEEE Sensors Journal, Vol. 12, No. 5, May 2012
4. C. M. Bishop (2006), Pattern Recognition and Machine Learning, 1st ed. New York: Springer.
5. S. S. Fels and G. E. Hinton (1993), ―Glove-talk: A neural network interface between a data glove and a speech synthesizer, ‖ IEEE Trans. Neural Network.
6. W. T. Freeman and C. D. Weissman (1995), ―TV control by hand gestures, ‖presented at the IEEE Int. Workshop on Automatic Face and Gesture Recognition, Zurich, Switzerland.