SMART GLOVE FOR SIGN LANGUAGE TRANSLATION USING ARDUINO

A Project report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by

K.RAVI CHANDRA (317126512032)

K.S.D.CHARISHMA PATNAIK (318126512L06)

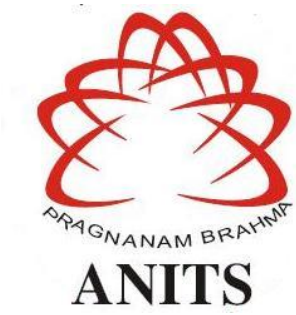
V.SUHRUTH(317126512056)

M.J.N.SANDEEP(317126512035)

Under the guidance of

Mr.D.Anil Prasad

Assistant Professor



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES
(UGC AUTONOMOUS)
(*Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC with 'A' Grade*)
Sangivalasa, bheemili mandal, visakhapatnam dist.(A.P)

(2020-2021)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES
(UGC AUTONOMOUS)
(Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC with 'A' Grade)
Sangivalasa, Bheemili mandal, Visakhapatnam dist. (A.P)

PRAGNANAM BRAHMA

ANITS

CERTIFICATE

This is to certify that the project report entitled **"SMART GLOVE FOR SIGN LANGUAGE TRANSLATION USING ARDUINO"** submitted by **K.RAVI CHANDRA (317126512032), K.S.D.CHARISHMA PATNAIK (318126512L06), V.SUHRUTH (317126512056), M.J.N.SANDEEP (317126512035)** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Electronics & Communication Engineering** of Andhra University, Visakhapatnam is a record of bonafide work carried out under my guidance and supervision.

Project Guide

Mr. D. Anil Prasad
Assistant Professor
Department of E.C.E
ANITS

Assistant Professor
Department of E.C.E.
Anil Neerukonda
Institute of Technology & Sciences
Sangivalasa, Visakhapatnam-531 162

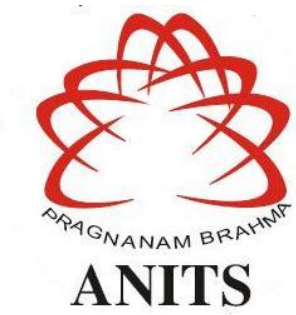
Head of the Department

Dr. V. Rajyalakshmi
Professor & HOD
Department of E.C.E
ANITS

Head of the Department
Department of E C E
Anil Neerukonda Institute of Technology & Sciences
Sangivalasa - 531 162

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES
(UGC AUTONOMOUS)
(*Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC with 'A' Grade*)
Sangivalasa, Bheemili mandal, Visakhapatnam dist. (A.P)



CERTIFICATE

This is to certify that the project report entitled **"SMART GLOVE FOR SIGN LANGUAGE TRANSLATION USING ARDUINO"** submitted by **K.RAVI CHANDRA (317126512032), K.S.D.CHARISHMA PATNAIK (318126512L06), V.SUHRUTH (317126512056), M.J.N.SANDEEP (317126512035)** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Electronics & Communication Engineering** of Andhra University, Visakhapatnam is a record of bonafide work carried out under my guidance and supervision.

Project Guide

Mr. D. Anil Prasad
Assistant Professor
Department of E.C.E
ANITS

Head of the Department

Dr. V. Rajyalakshmi
Professor & HOD
Department of E.C.E
ANITS

CONTENTS

ABSTRACT

Speech is the easiest way for communication in the world. It becomes difficult for speech impaired people to communicate with normal people as they use sign language for communication. When a speech-impaired person communicates with normal person, the bridge gap between speech impaired and normal masses is too much to fill. The gesture recognition can be done in two ways, Image processing based and sensor-based.

The Objective of the project is to design a smart glove for sign language translation that helps an easy way of communication for speech impaired or hearing-impaired people.

In this project, glove need to be equipped with sensors such as Flex sensor, Accelerometer, Touch sensor which sense different sign language gestures. Flex sensors are placed on fingers which measure the bending of fingers according to a gesture made. An accelerometer is placed on the palm which measures the location of the hand in X, Y, Z axes. Touch sensors are placed in between the fingers and measures if there is any contact between the fingers. The sensed data from sensors is sent to Arduino UNO board for further processing and transfer data to an android phone via bluetooth module. The data we get will be in the form of text. This text data is then converted into speech through Google text -speech converter.

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

In the present world it is very complicated for the deaf & dumb people to talk with the ordinary people as impaired people lacks the amenities which a normal person should own. It actually becomes the same problem of two persons which knows two different language, no one of them knows any common language so its becomes a problem to talk with each other and so they requires a translator physically which may not be always convenient to arrange and this same kind of problem occurs in between the Normal Person and the Deaf person or the Normal Person and the Dumb person.

Although technology has been evolving rapidly in this information age, deaf/mute people still use sign language as their only way of communication. Using sign language as a communication tool can be beneficial among those who are familiar with this language, but the problem remains when communicating with the wider community. Sign Language Translator is the appropriate solution that enables deaf/mute people to communication fluently through technology in different languages. As sign language is a formal language employing a system of hand gesture for communication (by the deaf). Many projects used glove-based systems for automatic understanding of gestural languages used by the deaf community [1]. The systems developed in these projects differed in characteristics such as number of classifiable signs, which could range from a few dozen to several thousand, types of signs, which could be either static or dynamic, and percentage of signs correctly classified. The simplest systems were limited to understanding of finger spelling or manual alphabets (a series of hand and finger static configurations that indicate letters). Takashi and Kishino [2] and Murakami and Taguchi [3] used a Data Glove for recognition of the Japanese alphabets. For recognition of the American alphabet, Hernadez-Herbollar used an AcceleGlove [4]. The more complex systems aimed at understanding sign languages, a series of dynamic hand and finger configurations that indicate words and grammatical structures. For instance, Kim and colleagues used a Data Glove for recognition of the Korean language [5], Kadous a Power Glove for the Australian language [6], Vamplew a CyberGlove for the Australian language [7], Gao and colleagues a CyberGlove for the Chinese language [8], [9], and Liang and Ouyoung a Data Glove for the Taiwanese language [10]–[12]. Some systems embedded

interfaces for translating sign languages into text or vocal outputs [13]–[15]. For instance, the Talking Glove used a Cyber Glove and recorded, recognized, and translated American sign language into text or spoken English [16].

Hand movement data acquisition is used in many engineering applications ranging from the analysis of gestures to the biomedical sciences. Glove-based systems represent one of the most important efforts aimed at acquiring hand movement data. While they have been around for over three decades, they keep attracting the interest of researchers from increasingly diverse fields. 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. We choose to study the glove systems for sign language understanding.

1.1 PROJECT OBJECTIVE

The Aim of the project is to develop a hand glove equipped with sensors such as Flex sensor, Accelerometer, Touch sensor which sense different sign language gestures. Flex sensors are placed on fingers which measure the bending of fingers according to a gesture made. An accelerometer is placed on the palm which measures the location of the hand in X, Y, Z axes. Touch sensors are placed in between the fingers and measures if there is any contact between the fingers. Firstly sensors were simulated to extract the sensed data. Secondly the sensed data from sensors is sent to Arduino UNO board for further processing and transfer data to an android phone via bluetooth module. The data will be in the form of text. This text data is then converted into speech through Google text -speech converter.

1.2 PROJECT OUTLINE

This project report is presented over the four remaining chapters. Chapter 2 describes the sign languages. Chapter 3 presents the principle of operation of sensors and various components used in the project. Chapter 4 explains the concepts of Arduino programming. Chapter 5 presents the simulation results of the various signs simulated using the Arduino simulator. Finally, conclusions are drawn in chapter 6.

CHAPTER 2
SIGN LANGUAGE DESCRIPTION

Sign languages are visual languages that use hand, facial and body movements as a means of communication. There are over 135 different sign languages all around the world including American Sign Language (ASL), Australian Sign Language (Auslan) and British Sign Language (BSL). There are also signed representations of oral languages such as Signed Exact English (SEE) and mixes such as Pidgin Signed English (PSE). Sign language is commonly used as the main form of communication for people who are Deaf or hard of hearing, but sign languages also have a lot to offer for everyone. Sign languages are an extremely important communication tool for many deaf and hard-of-hearing people. Sign languages are the native languages of the Deaf community and provide full access to communication. Although sign languages are used primarily by people who are deaf, they are also used by others, such as people who can hear but can't speak. People who know a sign language are often much better listeners. When using a sign language, a person must engage in constant eye contact with the person who is speaking. Unlike spoken language, with sign languages a person cannot look away from the person speaking and continue to listen. This can be an extremely beneficial habit to have for spoken language as well as sign language. By maintaining eye contact in spoken language, it shows that a person is genuinely interested in what the other is saying.

2.1 Different Sign Languages

American Sign Language (ASL):

Although ASL has the same alphabet as English, ASL is not a subset of the English language. American Sign Language was created independently and it has its own linguistic structure. (It is, in fact, descended from Old French Sign Language.) Signs are also not expressed in the same order as words are in English. This is due to the unique grammar and visual nature of the sign language. ASL is used by roughly half a million people in the world and it is shown in Fig.2.1

Fig 2.1 : American Sign Language

British, Australian and New Zealand Sign Language (BANZSL):

Sharing a sign language alphabet is British Sign Language, Australian Sign Language (Auslan) and New Zealand Sign Language. Unlike ASL, these alphabets use two hands, instead of one and it is shown in fig.2.2

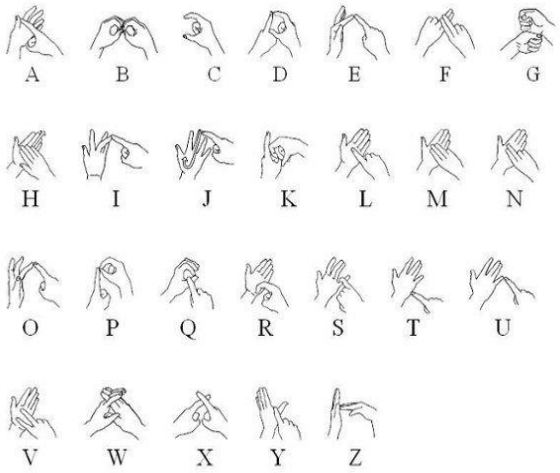


Fig 2.2 : Mexican Sign Language

Chinese Sign Language (CSL):

CSL's signs are visual representations of written Chinese characters, they use a_one handed alphabet as shown in fig.2.3. There are many CSL dialects but the Shanghai dialect is the most common. The language has been developing since the late 1950's and The Chinese National Association of the Deaf, is working hard to raise awareness and promote use of the language throughout the country.

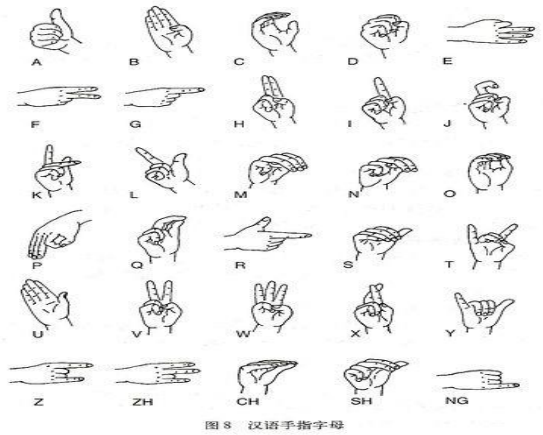


Fig 2.3 Chinese Sign Language

French Sign Language (LSF):

French Sign Language is similar to ASL – since it is in fact the origin of ASL – but there are minor differences throughout. LSF also has a pretty fascinating history. LSF is shown in fig.2.4

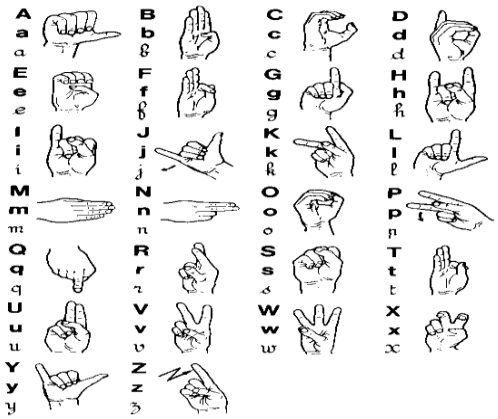


Fig 2.4 French Sign Language

Japanese Sign Language (JSL) Syllabary:

The Japanese Sign Language (JSL) Syllabary is based on the Japanese alphabet, which is made up of phonetic syllables. JSL is known as Nihon Shuwa in Japan and as shown in fig.2.5

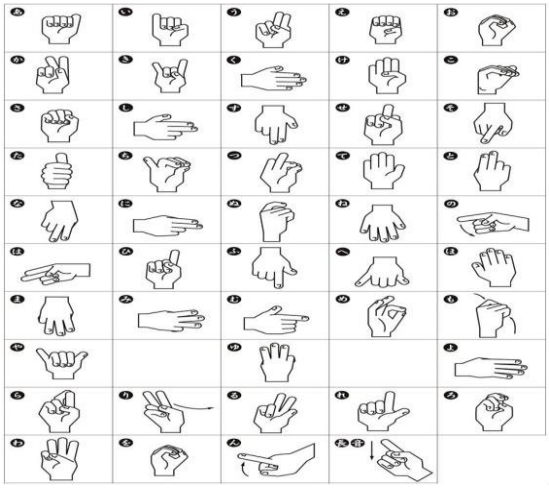


Fig 2.5 Japanese Sign Language

Arabic Sign Language:

The Arab sign-language family is a family of sign languages across the Arab Mideast. Data on these languages is somewhat scarce, but a few languages have been distinguished, including Levantine Arabic Sign Language. Arabic sign language is shown in fig.2.6

Fig 2.6 Arabic Sign Language

Spanish Sign Language (LSE):

Spanish Sign Language is officially recognized by the Spanish Government. It is native to Spain, except Catalonia and Valencia. Many countries that speak Spanish do not use Spanish Sign Language! (See Mexican Sign Language below, for example.) SSL is mainly used in Spain and there is an estimated 100,000 signers of SSL. SSL is completely different from ASL, in the same way that English is different from Spanish. SSL is used across all of Spain,

except in Catalonia which uses Catalan Sign Language and Valencia which uses Valencian Sign Language. LSE is shown in fig.2.7

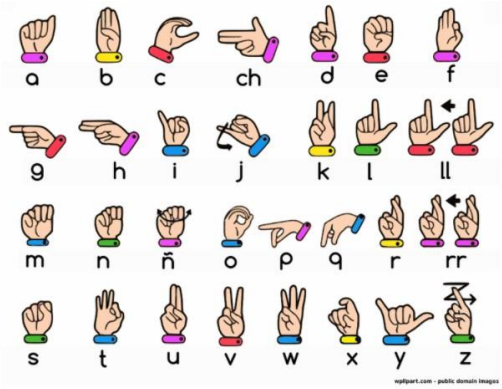


Fig 2.7 Spanish Sign Language

Mexican Sign Language (LSM):

Mexican Sign Language (‘lengua de señas mexicana’ or LSM) is different from Spanish, using different verbs and word order. The majority of people who use Mexican Sign Language reside in Mexico City, Guadalajara and Monterrey. Variation in this language is high between age groups and religious backgrounds. LSM is shown in fig.2.8

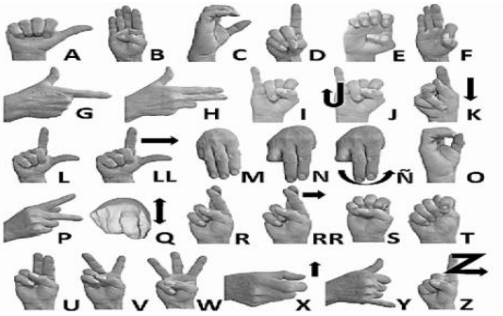


Fig 2.8 Mexican Sign Language

Ukrainian Sign Language (USL):

Ukrainian Sign Language is derived from the broad family of French Sign Languages. It uses a one-handed manual alphabet of 33 signs, which make use of the 23 handshapes of USL and is shown in fig.2.9

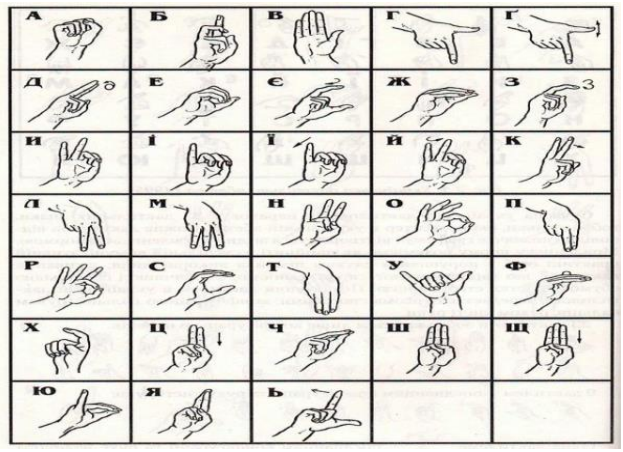


Fig 2.9 Ukrainian Sign Language

2.2 STANDARD SIGNS

There is no universal sign language. Different sign languages are used in different countries or regions. For example, British Sign Language (BSL) is a different language from ASL, and Americans who know ASL may not understand BSL. Some countries adopt features of ASL in their sign languages.

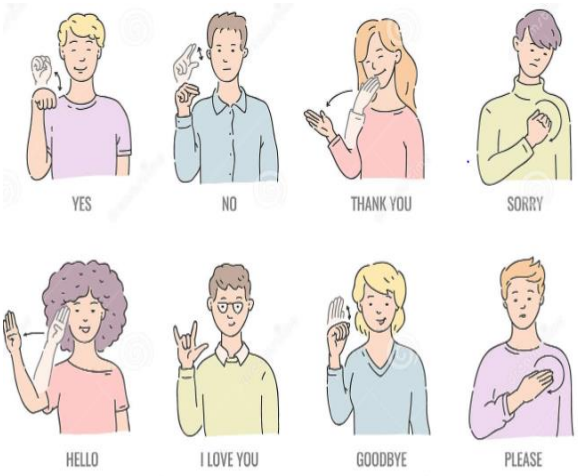


Fig 2.10 Standard Signs

It’s not always practical to spell out words for everyday interactions. That’s where these expressions come in handy! And are as shown in fig.2.10. We can use common expressions

to meet people, show your appreciation, and communicate with friends. It becomes easy for impaired persons to communicate with normal persons.

2.3 ADVANTAGES

1. It reduces frustration.
2. It increases self esteem.
3. It enhances languages and listening skills.
4. It enriches relationships.
5. It provides a window into your child's world.
6. It increases their IQ.

CHAPTER 3
SENSORS AND COMPONENTS

The proposed system consists of primarily two sections: 1. Transmitter Section 2. Receiver Section. The devices contained in the transmitter section are:

1. Flex sensors
2. Accelerometer Sensor (MPU6050)
3. Touch Sensor
4. HC-05 Bluetooth Module
5. Arduino Uno Microcontroller.

The gloves contain flex sensors which are the main sensors for this product. They are devices which can show variable resistance based on various bend angles. The sensors are connected in a voltage divider circuit such that the resultant analog voltage is sent to one analog port of the micro-controller. The glove is mounted with 4 flex sensors, each on one finger of the glove except the thumb finger.

3.1 FLEX SENSORS

A **flex sensor** or **bend sensor** is a low-cost and easy-to-use sensor specifically designed to measure the amount of deflection or bending. It became popular in the 90s due to its use in the Nintendo Power Glove as a gaming interface. Since then people have been using it as a goniometer to determine joint movement, a door sensor, a bumper switch for wall detection or a pressure sensor on robotic grippers.

3.1.1 Flex Sensor overview

A flex sensor is basically a variable resistor that varies in resistance upon bending. Since the resistance is directly proportional to the amount of bending, it is often called a **Flexible Potentiometer**. Flex sensors are generally available in two sizes: one is 2.2″ (5.588cm) long and another is 4.5″ (11.43cm) long. A flex sensor consists of a phenolic resin substrate with conductive ink deposited and is shown in fig.3.1. A segmented conductor is placed on top to form a flexible potentiometer in which resistance changes upon deflection. Flex sensors are designed to flex in only one direction – away from ink. Bending the sensor in another direction may damage it. Also take care not to bend the sensor close to the base, because the

bottom of the sensor (where the pins are crimped on) is very fragile and can break when bent over.

Fig 3.1 Flex Sensor

3.1.2 Flex Sensor Working

The conductive ink printed on the sensor acts as a resistor. When the sensor is straight, this resistance is about 25k as shown in fig.3.2

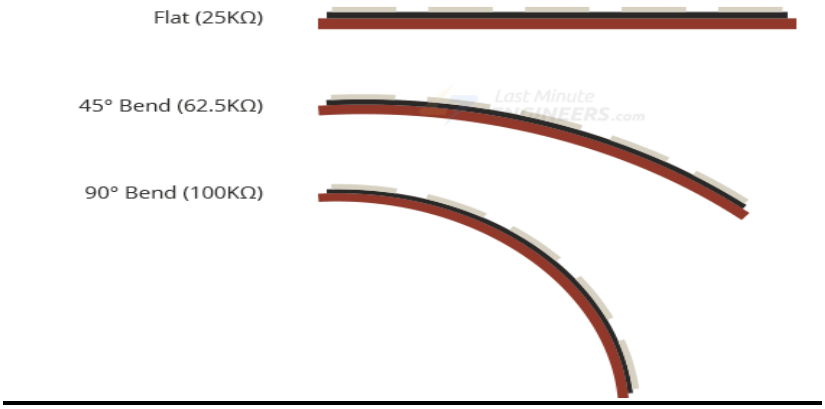
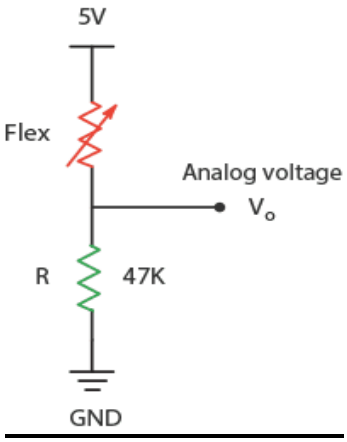


Fig 3.2 Flex Sensor working

When the sensor is bent, conductive layer is stretched, resulting in reduced cross section (imagine stretching a rubber band). This reduced cross section results in an increased resistance. At 90° angle, this resistance is about 100KΩ. When the sensor is straightened again, the resistance returns to its original value. By measuring the resistance, you can determine how much the sensor is bent.

3.1.3 Reading a Flex Sensor

The easiest way to read the flex sensor is to connect it with a fixed value resistor (usually 47kΩ) to create a voltage divider. To do this you connect one end of the sensor to Power and the other to a pull-down resistor as shown below. Then the point between the fixed value pull-down resistor and the flex sensor is connected to the ADC input of an Arduino. This way you can create a variable voltage output, which can be read by Arduino's ADC input.



Note that the output voltage you measure is the voltage drop across the pull-down resistor, not across the flex sensor. In the shown configuration, the output voltage decreases with increasing bend radius. The output of the voltage divider configuration is described by the equation $V_0 = V_{CC} \frac{R}{R+R_{Flex}}$

3.1.4 Basic Flex circuit and Characteristics

Figure 3.3 shows circuit of basic flex sensor which consist of two or three sensors which are connected. The outputs from the flex sensors are given as inputs to op-amp and used a noninverted style setup to amplify their voltage. The greater the degree of bending the lower the output voltage. By voltage divider rule, output voltage is determined and given by

$$V_{0ut} = V_{in}\frac{R1}{R1 + R_2}$$

where R1 is the other input resistor to the non-inverting terminal and the characteristics are shown in fig.3.4

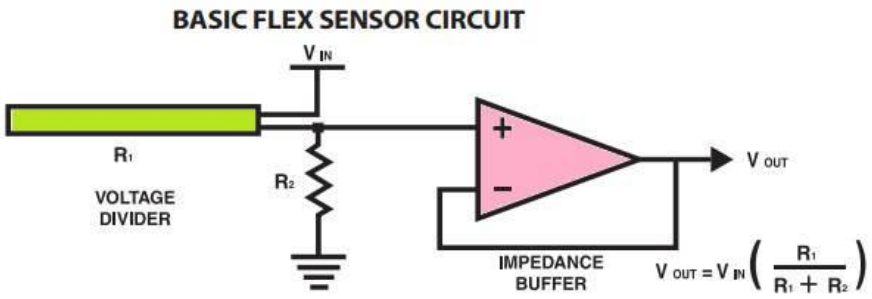


Fig 3.3 Basic Flex Circuit

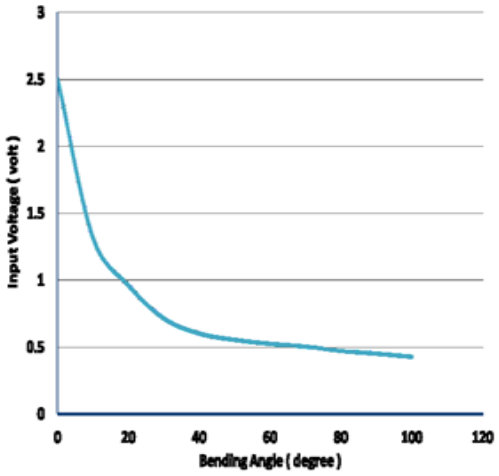


Fig 3.4 Characteristics of flex sensor

3.2 ACCELEROMETER SENSOR: (MPU6050)

In recent years, some crafty engineers successfully made micromachined gyroscopes. These MEMS (microelectromechanical system) gyroscopes have paved the way to a completely new set of innovative applications such as gesture recognition, enhanced gaming, augmented reality, panoramic photo capture, vehicle navigation, fitness monitoring and many more, no doubt the gyroscope and accelerometer are great in their own way. But when we combine them, we can get very accurate information about the orientation of an object. This is where the **MPU6050** comes in. The **MPU6050** has both a gyroscope and an accelerometer, using which we can measure rotation along all three axes, static acceleration due to gravity, as well as motion, shock, or dynamic acceleration due to vibration.

3.2.1 MPU6050 Module Hardware Overview

At the heart of the module is a low power, inexpensive 6-axis Motion Tracking chip that combines a 3-axis gyroscope, 3-axis accelerometer, and a Digital Motion Processor (DMP) all in a small 4mm x 4mm package as shown in fig.3.5. It can measure angular momentum or rotation along all the three axes, the static acceleration due to gravity, as well as dynamic acceleration resulting from motion, shock, or vibration.

Fig 3.5 MPU6050

The module comes with an on-board LD3985 3.3V regulator, so you can use it with a 5V logic microcontroller like Arduino without worry. The MPU6050 consumes less than 3.6mA during measurements and only 5μA during idle. This low power consumption allows the

implementation in battery driven devices. In addition, the module has a power LED that lights up when the module is powered.

3.2.2 MPU6050 Module Pinout

The pin diagram of the module is as shown in fig.3.6 below

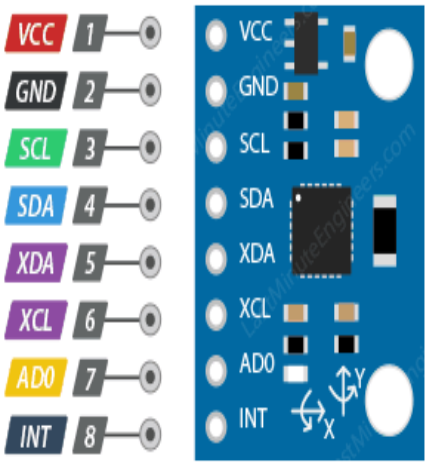


Fig 3.6 MPU6050 Module Pinout

VCC is the power supply for the module. Connect it to the 5V output of the Arduino.

GND should be connected to the ground of Arduino.

SCL is a I2C Clock pin. This is a timing signal supplied by the Bus Master device. Connect to the SCL pin on the Arduino.

SDA is a I2C Data pin. This line is used for both transmit and receive. Connect to the SDA pin on the Arduino.

XDA is the external I2C data line. The external I2C bus is for connecting external sensors.

XCL is the external I2C clock line.

AD0 allows you to change the internal I2C address of the MPU6050 module. It can be used if the module is conflicting with another I2C device, or if you wish to use two MPU6050s on the same I2C bus. When you leave the ADO pin unconnected, the default I2C address is $0x68_{HEX}$ and when you connect it to 3.3V, the I2C address becomes $0x69_{HEX}$.

INT is the Interrupt Output. MPU6050 can be programmed to raise interrupt on gesture detection, panning, zooming, scrolling, tap detection, and shake detection.

3.2.3 The I2C Interface

The module uses the I2C interface for communication with the Arduino. It supports two separate I2C addresses: $0x68_{HEX}$ and $0x69_{HEX}$. This allows two MPU6050s to be used on the same bus or to avoid address conflicts with another device on the bus.

The ADO pin determines the I2C address of the module. This pin has a built-in 4.7K pull-down resistor. Therefore, when you leave the ADO pin unconnected, the default I2C address is $0x68_{HEX}$ and when you connect it to 3.3V, the line is pulled HIGH and the I2C address becomes $0x69_{HEX}$.

MPU6050 module doesn't only provide the measurement of acceleration but it is also useful in measurement of Rotation i.e. as a gyroscope and also it can measure the temperature.

3.2.4 Measuring Acceleration

The MPU6050 can measure acceleration using its on-chip accelerometer with four programmable full-scale ranges of ±2g, ±4g, ±8g and ±16g.

The MPU6050 has three 16-bit analog-to-digital converters that simultaneously sample the 3 axes of movement (along X, Y and Z axis) as shown in fig.3.7

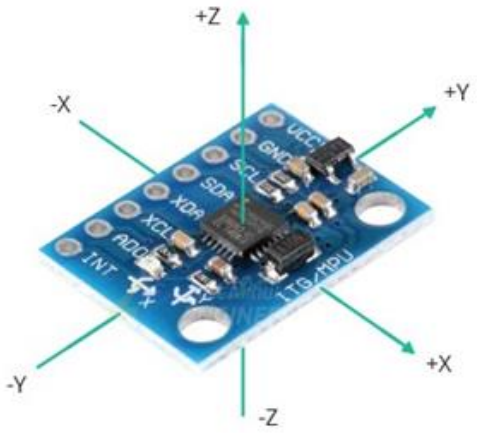


Fig 3.7 Accelerometer

3.2.5 Measuring Rotation

The MPU6050 can measure angular rotation using its on-chip gyroscope with four programmable full-scale ranges of ±250°/s, ±500°/s, ±1000°/s and ±2000°/s.

The MPU6050 has another three 16-bit analog-to-digital converters that simultaneously samples 3 axes of rotation (around X, Y and Z axis). The sampling rate can be adjusted from 3.9 to 8000 samples per second.

3.2.6 Measuring Temperature

The MPU6050 includes an embedded temperature sensor that can measure temperature over the range of -40 to 85°C with accuracy of ±1°C.

Note that this temperature measurement is of the silicon die itself and not the ambient temperature. Such measurements are commonly used to offset the calibration of accelerometer and gyroscope or to detect temperature changes rather than measuring absolute temperatures.

3.3 TOUCH SENSOR

The human body has five sense elements which are used to interact with our surroundings. Machines also need some sensing elements to interact with their surroundings. To make this possible sensor was invented. The invention of the first manmade sensor, thermostat, dates back to 1883. In 1940s infrared sensors were introduced. Today we have sensors that can sense motion, light, humidity, temperature, smoke, etc. Analog and digital both types of sensors are available today. Sensors have brought a revolutionary change in the size and cost of various control systems. One of such sensors which can detect touch is the Touch sensor.

Touch Sensors are the electronic sensors that can detect touch. They operate as a switch when touched. These sensors are used in lamps, touch screens of the mobile, etc. Touch sensors offer an intuitive user interface.

Touch sensors are also known as Tactile sensors. These are simple to design, low cost and are produced in large scale. With the advance in technology, these sensors are rapidly replacing the mechanical switches. Based on their functions there are two types of touch sensors- Capacitive sensor and Resistive sensor.

Capacitive sensors work by measuring capacitance and are seen in portable devices. These are durable, robust and attractive with low cost. Resistive sensors don’t depend on any

electrical properties for operation. These sensors work by measuring the pressure applied to their surface.

3.3.1 Touch Sensor Pinout

The touch sensor pinout is shown in fig.3.8 below.

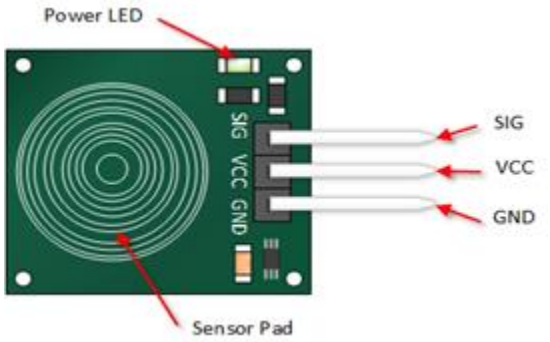


Fig 3.8 Touch Sensor Pinout

Touch sensor has 3 pins:

1. GND pin needs to be connected to GND (0V)
2. VCC pin needs to be connected to VCC (5V or 3.3v)
3. SIGNAL pin is an output pin: LOW when it is NOT touched, HIGH when it is touched.

3.3.2 Working Principle of Touch Sensor

Touch sensors work similar to a switch. When they are subjected to touch, pressure or force they get activated and acts as a closed switch. When the pressure or contact is removed, they act as an open switch. Capacitive touch sensor contains two parallel conductors with an insulator between them. These conductor plates act as a capacitor with a capacitance value C_0. When these conductor plates come in contact with our fingers, our finger acts as a conductive object. Due to this, there will be an uncertain increase in the capacitance. A capacitance measuring circuit continuously measures the capacitance C_0 of the sensor. When this circuit detects a change in capacitance it generates a signal. The resistive touch sensors calculate the pressure applied on the surface to sense the touch. These sensors contain two conductive films coated with indium tin oxide, which is a good conductor of electricity, separated by a very small distance. Across the surface of the films, a constant voltage is applied. When pressure is applied to the top film, it touches the bottom film. This generates a

voltage drop which is detected by a controller circuit and signal is generated thereby detecting the touch.

3.3.3 Circuit of Touch Sensor Interfacing with Arduino

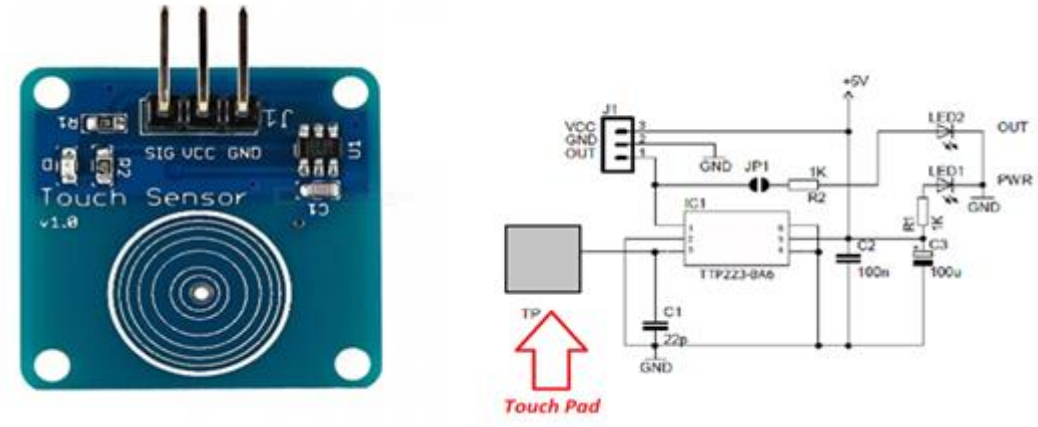


Fig 3.9 Circuit of Touch Sensor Interfacing with Arduino

Capacitor sensors are easily available and are of very low cost. These sensors are highly used in mobile phones, iPods, automotive, small home appliances, etc. These are also used for measuring pressure, distance, etc. A drawback of these sensors is that they can give a false alarm. Resistive touch sensors only work when sufficient pressure is applied. Hence, these sensors are not useful for detecting small contact or pressure. These are used in applications such as musical instruments, keypads, touch-pads, etc. where a large amount of pressure is applied.

3.4 HC-05 BLUETOOTH MODULE

HC-05 Bluetooth Module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. Its communication is via serial communication which makes an easy way to interface with controller or PC. HC-05 Bluetooth module provides switching mode between master and slave mode which means it able to use neither receiving nor transmitting data.

Table 3.1 : HC-05 pin description

1	Enable / Key	This pin is used to toggle between Data Mode (set low) and AT command mode (set high). By default it is in Data mode
2	Vcc	Powers the module. Connect to +5V Supply voltage
3	Ground	Ground pin of module, connect to system ground.
4	TXD– Transmitter	Transmits Serial Data. Everything received via Bluetooth will be given out by this pin as serial data.
5	RXD– Receiver	Receive Serial Data. Every serial data given to this pin will be broadcasted via Bluetooth
6	State	The state pin is connected to on board LED, it can be used as a feedback to check if Bluetooth is working properly.
7	LED	Indicates the status of Module Blink once in 2 sec: Module has entered Command Mode Repeated Blinking: Waiting for connection in Data Mode Blink twice in 1 sec: Connection successful in Data Mode
8	Button	Used to control the Key/Enable pin to toggle between Data and command Mode

3.4.1 HC-05 BLUETOOTH MODULE PINOUT

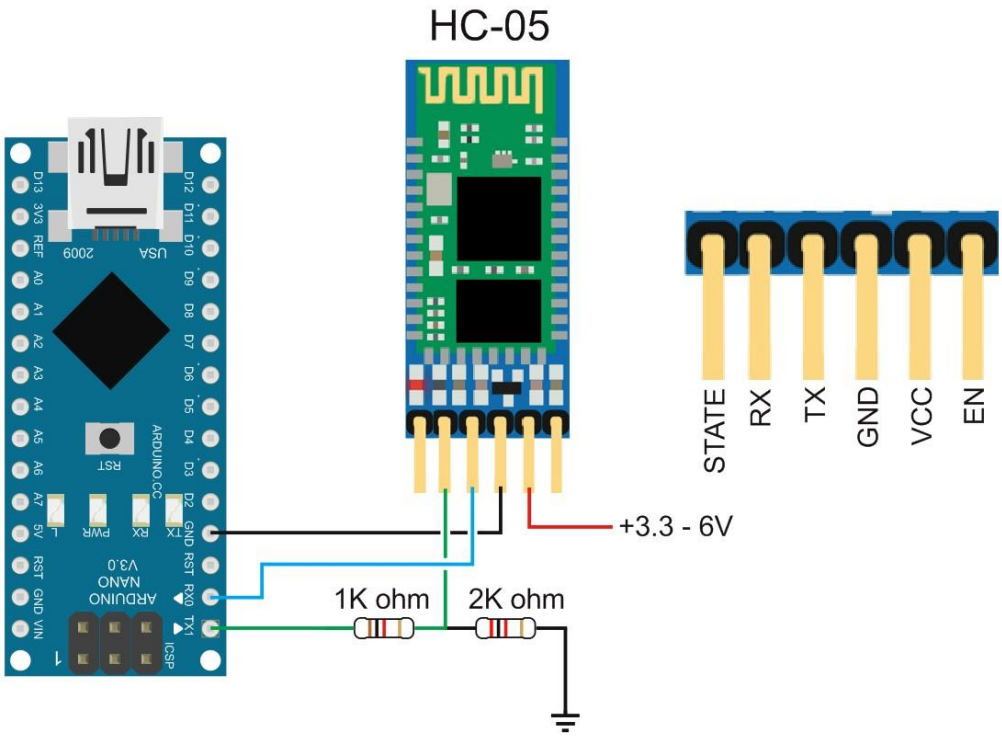


Fig 3.10 HC-05 BLUETOOTH MODULE

3.4.2 HC-05 Technical Specifications

- Serial Bluetooth module for Arduino and other microcontrollers
- Operating Voltage: 4V to 6V (Typically +5V)
- Operating Current: 30mA
- Range: <100m
- Works with Serial communication (USART) and TTL compatible
- Follows IEEE 802.15.1 standardized protocol
- Uses Frequency-Hopping Spread spectrum (FHSS)
- Can operate in Master, Slave or Master/Slave mode
- Can be easily interfaced with Laptop or Mobile phones with Bluetooth
- Supported baud rate: 9600,19200,38400,57600,115200,230400,460800.

3.4.3 HC-05 Default Settings

- Default Bluetooth Name: “HC-05”
- Default Password: 1234 or 0000
- Default Communication: Slave
- Default Mode: Data Mode
- Data Mode Baud Rate: 9600, 8, N, 1
- Command Mode Baud Rate: 38400, 8, N, 1
- Default firmware: LINVO

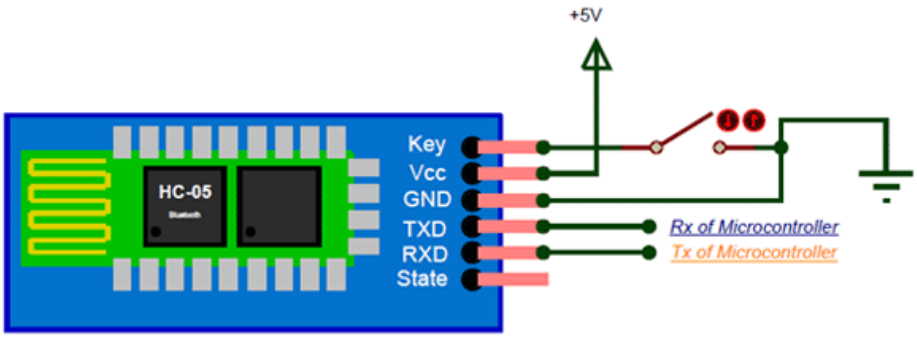
3.4.4 Where to use HC-05 Bluetooth module

The **HC-05** is a very cool module which can add two-way (full-duplex) wireless functionality to your projects. You can use this module to communicate between two microcontrollers like Arduino or communicate with any device with Bluetooth functionality like a Phone or Laptop. There are many android applications that are already available which makes this process a lot easier. The module communicates with the help of USART at 9600 baud rates hence it is easy to interface with any microcontroller that supports USART. We can also configure the default values of the module by using the command mode. So, if you looking for a Wireless module that could transfer data from your computer or mobile phone to microcontroller or vice versa then this module might be the right choice for you. However, do not expect this module to transfer multimedia like photos or songs; you might have to look into the CSR8645 module for that.

3.4.5 How to Use the HC-05 Bluetooth module

The **HC-05** has two operating modes, one is the Data mode in which it can send and receive data from other Bluetooth devices and the other is the AT Command mode where the default device settings can be changed. We can operate the device in either of these two modes by using the key pin as explained in the pin description.

It is very easy to pair the HC-05 module with microcontrollers because it operates using the Serial Port Protocol (SPP). Simply power the module with +5V and connect the Rx pin of the module to the Tx of MCU and Tx pin of module to Rx of MCU as shown in the Figure below



During power up the key pin can be grounded to enter into Command mode, if left free it will by default enter into the data mode. As soon as the module is powered you should be able to discover the Bluetooth device as “HC-05” then connect with it using the default password 1234 and start communicating with it.

3.5 ARDUINO UNO

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. 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. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. It is similar to the Arduino Nano and Leonardo. The hardware reference design is distributed under a Creative Commons Attribution Share-Alike 2.5 license and is available on the Arduino website. Layout and production files for some versions of the hardware are also available.

The word "uno" means "one" in Italian and was chosen to mark the initial release of Arduino Software. The Uno board is the first in a series of USB-based Arduino boards; it and version 1.0 of the Arduino IDE were the reference versions of Arduino, which have now evolved to

newer releases. The ATmega328 on the board comes preprogrammed with a bootloader that allows uploading new code to it without the use of an external hardware programmer. While the Uno communicates using the original STK500 protocol, it differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it uses the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

3.5.1 History

The Arduino project started at the Interaction Design Institute Ivrea (IDII) in Ivrea, Italy. At that time, the students used a BASIC Stamp microcontroller, at a cost that was a considerable expense for many students. In 2003, Hernando Barragan created the development platform Wiring as a Master's thesis project at IDII, under the supervision of Massimo Benzie and Casey Rees, who are known for work on the Processing language. The project goal was to create simple, low-cost tools for creating digital projects by non-engineers. The Wiring platform consisted of a printed circuit board (PCB) with an ATmega168 microcontroller, an IDE based on Processing, and library functions to easily program the microcontroller. In 2003, Massimo Benzie, with David Mallis, another IDII student, and David Cuartilla's, added support for the cheaper ATmega8 microcontroller to Wiring. But instead of continuing the work on Wiring, they forked the project and renamed it Arduino. Early Arduino boards used the FTDI USB-to-serial driver chip and an ATmega168. The Uno differed from all preceding boards by featuring the ATmega328P microcontroller and an ATmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

3.5.2 ARDUINO UNO PINOUT:

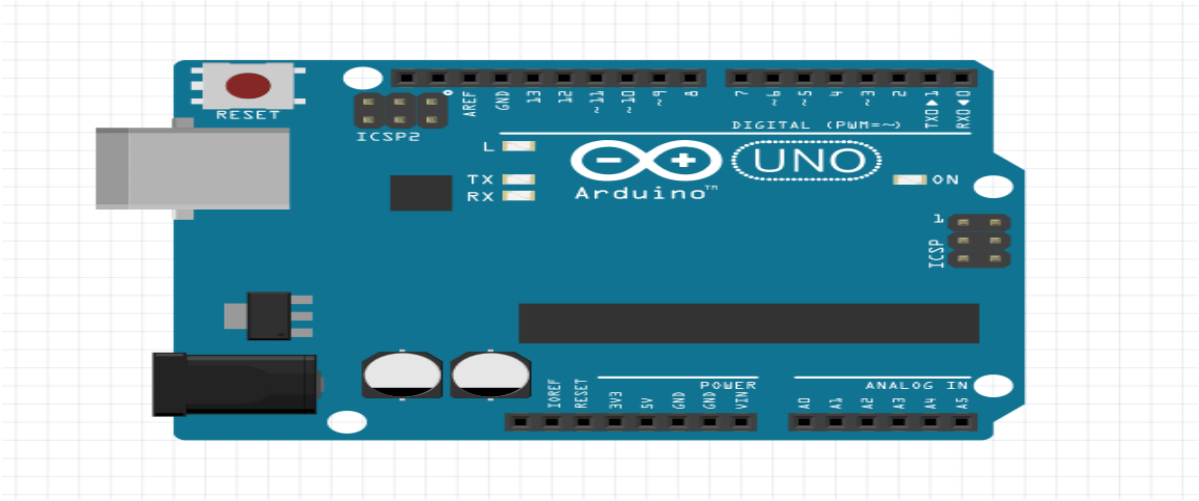


Fig 3.11 ARDUINO UNO

General pin functions

- **LED**: There is a built-in LED driven by digital pin 13. When the pin is high value, the LED is on, when the pin is low, it is off.
- **VIN**: The input voltage to the Arduino/Genuino board when it is 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 - 20V), the USB connector (5V), or the VIN pin of the board (7-20V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage the board.
- **3V3**: A 3.3-volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND**: Ground pins.
- **IOREF**: This pin on the Arduino/Genuino board provides the voltage reference with which the microcontroller operates. A properly conFigured shield can read the IOREF

pin voltage and select the appropriate power source, or enable voltage translators on the outputs to work with the 5V or 3.3V.

- **RESET**: Typically used to add a reset button to shields that block the one on the board.

Special pin functions

Each of the 14 digital pins and 6 analog pins on the Uno can be used as an input or output, under software control (using pinMode (), digitalWrite (), and digitalRead () functions). They operate at 5 volts. Each pin can provide or receive 20 mA as the recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50K ohm. A maximum of 40mA must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller. The Uno has 6 analog inputs, labeled A0 through A5; each provides 10 bits of resolution (i.e. 1024 different values). By default, they measure from ground to 5 volts, though it is possible to change the upper end of the range using the AREF pin and the analogReference () function.

- **Serial** / UART: pins 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**: pins 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.
- **PWM** (pulse-width modulation): pins 3, 5, 6, 9, 10, and 11. Can provide 8-bit PWM output with the analogWrite () function.
- **SPI** (Serial Peripheral Interface): pins 10 (SS), 11 (MOSI), 12 (MISO), and 13 (SCK). These pins support SPI communication using the SPI library.
- **TWI** (two-wire interface) / I²C: pin SDA (A4) and pin SCL (A5). Support TWI communication using the Wire library.
- **AREF** (analog reference): Reference voltage for the analog inputs.

3.5.3 Technical specifications

- Microcontroller: Microchip ATmega328P
- Operating Voltage: 5 Volts
- Input Voltage: 7 to 20 Volts
- Digital I/O Pins: 14 (of which 6 can provide PWM output)
- UART: 1
- I2C: 1
- SPPI: 1
- Analog Input Pins: 6
- DC Current per I/O Pin: 20 mA
- DC Current for 3.3V Pin: 50 mA
- Flash Memory: 32 KB of which 0.5 KB used by bootloader
- SRAM: 2 KB
- EEPROM: 1 KB
- Clock Speed: 16 MHz
- Length: 68.6 mm
- Width: 53.4 mm
- Weight: 25 g

3.5.4 Communication

The Arduino/Genuino Uno has a number of facilities for communicating with a computer, another Arduino/Genuino board, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The 16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent

to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A SoftwareSerial library allows serial communication on any of the Uno's digital pins.

3.5.5 Applications of Arduino Uno ATmega328

The **applications of Arduino Uno** include the following:

- **Arduino Uno** is used in Do-it-Yourself projects prototyping.
- In developing projects based on code-based control
- Development of Automation System
- Designing of basic circuit designs.

Thus, this is all about **Arduino Uno datasheet**. We can conclude that this is an 8-bit ATmega328P microcontroller. It has different components like serial communication, **crystal oscillator**, the voltage regulator for supporting the microcontroller. This board includes a USB connection, digital I/O pins-14, analog i/p pins-6, a power-barrel jack, a reset button, and an ICSP header.

CHAPTER 4
ARDUINO PROGRAMMING

Arduino is a prototype platform (open-source) based on an easy-to-use hardware and software. It consists of a circuit board, which can be programed (referred to as a microcontroller) and a ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board. Arduino provides a standard form factor that breaks the functions of the micro-controller into a more accessible package.

The key features are:

- Arduino boards are able to read analog or digital input signals from different sensors and turn it into an output such as activating a motor, turning LED on/off, connect to the cloud and many other actions.
- You can control your board functions by sending a set of instructions to the microcontroller on the board via Arduino IDE (referred to as uploading software).
- Unlike most previous programmable circuit boards, Arduino does not need an extra piece of hardware (called a programmer) in order to load a new code onto the board. You can simply use a USB cable.
- Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program.
- Finally, Arduino provides a standard form factor that breaks the functions of the microcontroller into a more accessible package.

4.1 ARDUINO BOARD TYPES

Various kinds of Arduino boards are available depending on different microcontrollers used. However, all Arduino boards have one thing in common: they are programed through the Arduino IDE.

The differences are based on the number of inputs and outputs (the number of sensors, LEDs, and buttons you can use on a single board), speed, operating voltage, form factor etc. Some boards are designed to be embedded and have no programming interface (hardware), which

you would need to buy separately. Some can run directly from a 3.7V battery, others need at least 5V. Here is a list of different Arduino boards available.

Table 4.1 Arduino boards based on ATMEGA32u4 microcontroller

Board name	Operating voltage	Clock speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Leonardo	5V	16MHz	20	12	7	1	Native USB
Pro micro 5V/16MHz	5V	16MHz	14	6	6	1	Native USB
Pro micro 5V/16MHz	3.3V	8MHz	14	6	6	1	Native USB
Lilypad Arduino USB	5V	16MHz	14	6	6	1	Native USB

Table 4.2 Arduino boards based on AT91SAM3X8E microcontroller

Board name	Operating voltage	Clock speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Due	3.3V	84MHz	54	12	12	4	USB native

Table 4.3 Arduino boards based on ATMEGA2560 microcontroller

Board name	Operating voltage	Clock speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface

Arduino Mega 2560 R3	5V	16MHz	54	16	14	4	USB via ATMega16U2
Mega Pro 3.3V	3.3V	8MHz	54	16	14	4	FTDI Compatible Header
Mega Pro 5V	5V	16MHz	54	16	14	4	FTDI Compatible Header
Mega Pro Mini 3.3V	3.3V	8MHz	54	16	14	4	FTDI Compatible Header

Table 4.4 Arduino boards based on ATMEGA328 microcontroller

Board name	**Operating voltage**	**Clock speed**	**Digital i/o**	**Analog Inputs**	**PWM**	**UART**	**Programming Interface**
Arduino Uno R3	5V	16MHz	14	6	6	1	USB via ATMega16U2
Arduino Uno R3 SMD	5V	16MHz	14	6	6	1	USB via ATMega16U2
Red Board	5V	16MHz	14	6	6	1	USB via FTDI
Arduino Pro	3.3V	8 MHz	14	6	6	1	FTDI Compatible

3.3v/8 MHz							Header
Arduino Pro 5v/16 MHz	5V	16 MHz	14	6	6	1	FTDI Compatible Header
Arduino mini 05	5V	16 MHz	14	8	6	1	FTDI Compatible Header
Arduino Pro mini 3.3v/8 MHz	3.3V	8 MHz	14	8	6	1	FTDI Compatible Header
Arduino Pro mini5v/16 MHz	5V	16 MHz	14	8	6	1	FTDI Compatible Header

4.2 BOARD DESCRIPTION

1-> Power USB:

ARDUINO – BOARD DESCRIPTION 12 Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection (1).

2-> Power (Barrel Jack):

Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2).

3-> Voltage Regulator:

The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.

4-> Crystal Oscillator:

The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz.

5, 17->Arduino Reset:

You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).

6,7,8,9 ->Pins (3.3, 5, GND, Vin) :

3.3V (6): Supply 3.3 output volt

5V (7): Supply 5 output volt

Most of the components used with Arduino board works fine with 3.3 volt and 5 volt.

GND (8)(Ground): There are several GND pins on the Arduino, any of which can be used to ground your circuit.

Vin (9): This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.

10-> Analog pins:

The Arduino UNO board has five analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.

11->Main microcontroller:

Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the data sheet.

12->ICSP pin:

Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial

Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus.

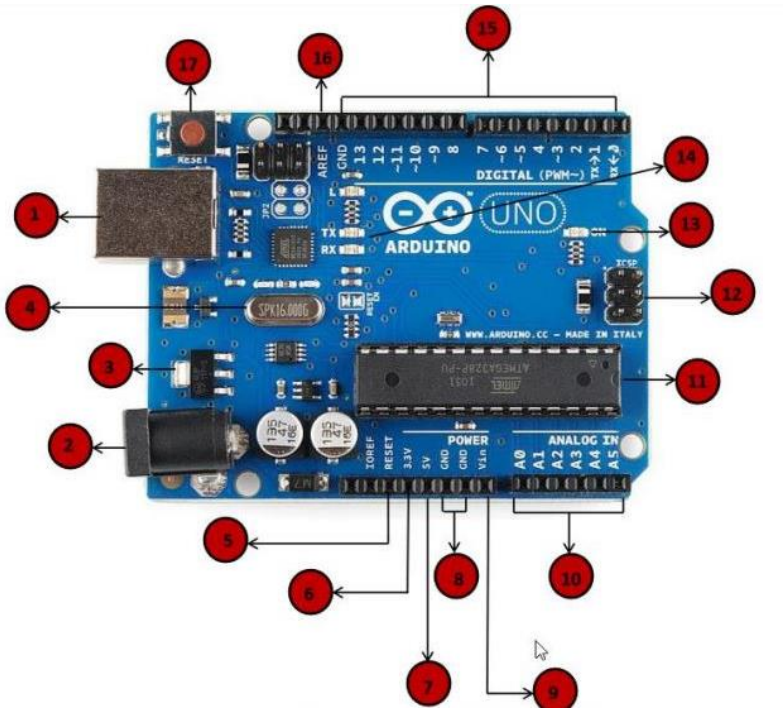


Fig 4.1 Arduino Uno Board

13->Power LED indicator:

This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.

14->TX and RX LEDs:

On your board, you will find two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication. Second, the TX and RX led (13). The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.

15->Digital I / O:

The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be conFigured to work as input digital pins to read logic

14 values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled "~" can be used to generate PWM.

16->AREF:

AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

4.3 ARDUINO PINOUT

The Arduino UNO is a standard board of Arduino, which is based on an ATmega328P microcontroller. It is easier to use than other types of Arduino Boards.

The Arduino UNO Board, with the specification of pins, is shown below:

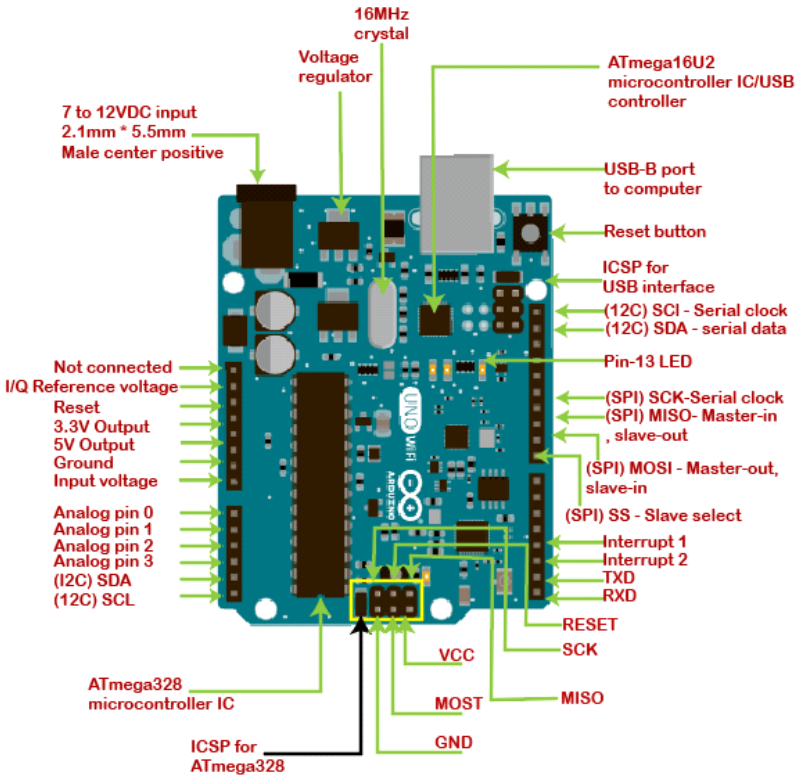


Fig 4.2 Pins Specifications

4.4 PROGRAMMING

4.4.1: BASICS

Coding Screen:

The coding screen is divided into two blocks. The ‘setup’ is considered as the preparation block, while the ‘loop’ is considered as the execution block. It is shown below:

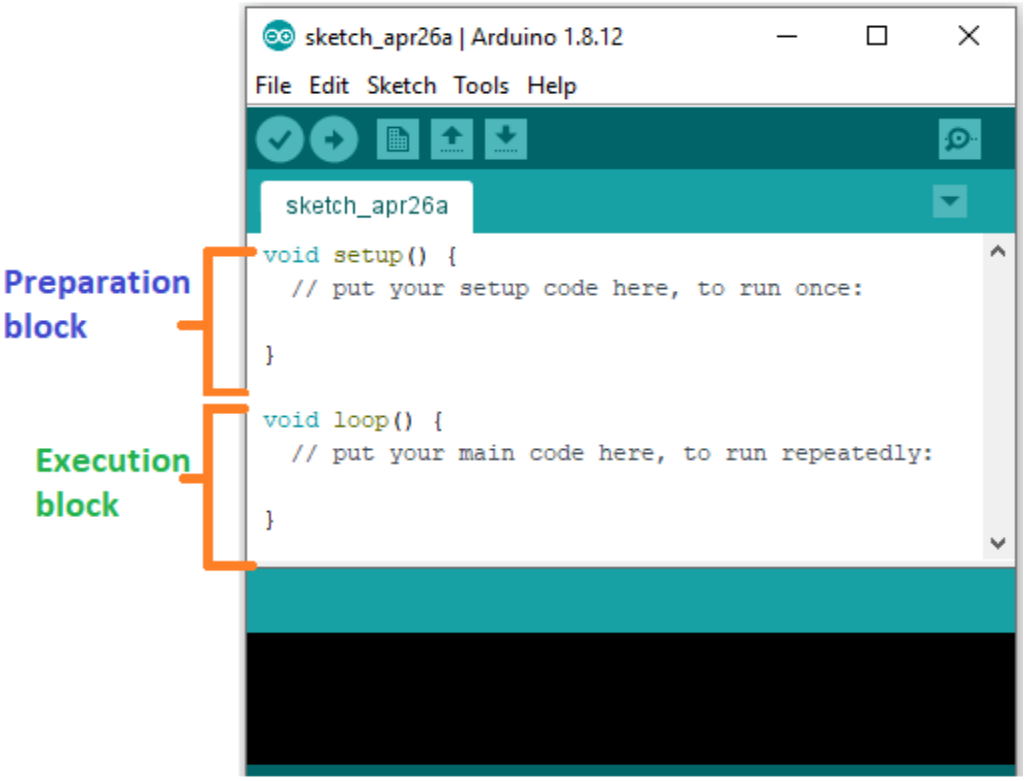


Fig 4.3 Coding Screen

The set of statements in the setup and loop blocks are enclosed with the curly brackets. We can write multiple statements depending on the coding requirements for a particular project.

Set Up (): It contains an initial part of the code to be executed. The pin modes, libraries, variables, etc., are initialized in the setup section. It is executed only once during the uploading of the program and after reset or power up of the Arduino board.Zero setup () resides at the top of each sketch. As soon as the program starts running, the code inside the curly bracket is executed in the setup and it executes only once.

Loop (): The loop contains statements that are executed repeatedly. The section of code inside the curly brackets is repeated depending on the value of variables.

Pin Mode ():

The specific pin number is set as the INPUT or OUTPUT in the pinMode () function.

The Syntax is: **pinMode (pin, mode)**

Where,

pin: It is the pin number. We can select the pin number according to the requirements.

Mode: We can set the mode as INPUT or OUTPUT according to the corresponding pin number.

The OUTPUT mode of a specific pin number provides a considerable amount of current to other circuits, which is enough to run a sensor or to light the LED brightly. The output state of a pin is considered as the low-impedance state.

The high current and short circuit of a pin can damage the ATmel chip. So, it is recommended to set the mode as OUTPUT.

Digital Write ():

The digitalWrite ( ) function is used to set the value of a pin as HIGH or LOW.

HIGH: It sets the value of the voltage. For the 5V board, it will set the value of 5V, while for 3.3V, it will set the value of 3.3V.

LOW: It sets the value = 0 (GND).

If we do not set the pinMode as OUTPUT, the LED may light dim.

The syntax is: **digitalWrite(pin, value HIGH/LOW)**

The digitalRead () function will read the HIGH/LOW value from the digital pin, and the digitalWrite () function is used to set the HIGH/LOW value of the digital pin.

pin: We can specify the pin number or the declared variable.

Delay ():

The delay () function is a blocking function to pause a program from doing a task during the specified duration in milliseconds.

For example, - delay (2000)

Where, 1 sec = 1000millisecond

Hence, it will provide a delay of 2 seconds.

4.4.2 Syntax and Programme Flow

Syntax:

Syntax in Arduino signifies the rules need to be followed for the successful uploading of the Arduino program to the board. The syntax of Arduino is similar to the grammar in English. It means that the rules must be followed in order to compile and run our code successfully. If we break those rules, our computer program may compile and run, but with some bugs.

Functions:

- The functions in Arduino combine many pieces of lines of code into one.
- The functions usually return a value after finishing execution. But here, the function does not return any value due to the presence of void.
- The setup and loop function have **void** keyword present in front of their function name.
- The multiple lines of code that a function encapsulates are written inside curly brackets.
- Every closing curly bracket ' **}** ' must match the opening curly bracket '**{** ' in the code.
- We can also write our own functions, which will be discussed later in this tutorial.

Spaces:

- Arduino ignores the white spaces and tabs before the coding statements.
- The coding statements in the code are intent (empty spacing at the starting) for the easy reading.
- In the function definition, loop, and conditional statements, 1 intent = 2 spaces.
- The compiler of Arduino also ignores the spaces in the parentheses, commas, blank lines, etc.

Tools Tab:

- The verify icon present on the tool tab only compiles the code. It is a quick method to check that whether the syntax of our program is correct or not.
- To compile, run, and upload the code to the board, we need to click on the Upload button.

Uses of Paranthesis ():

- It denotes the function like void setup ( ) and void loop ( ).
- The parameter's inputs to the function are enclosed within the parentheses.
- It is also used to change the order of operations in mathematical operations.

SemiColon ;

- It is the statement terminator in the C as well as C++.
- A statement is a command given to the Arduino, which instructs it to take some kind of action. Hence, the terminator is essential to signify the end of a statement.
- We can write one or more statements in a single line, but with semicolon indicating the end of each statement.
- The compiler will indicate an error if a semicolon is absent in any of the statements.
- It is recommended to write each statement with semicolon in a different line, which makes the code easier to read.
- We are not required to place a semicolon after the curly braces of the setup and loop function.

Arduino processes each statement sequentially. It executes one statement at a time before moving to the next statement.

Program Flow:

The program flow in Arduino is similar to the flowcharts. It represents the execution of a program in order.

The Arduino coding process in the form of the flowchart is shown below:

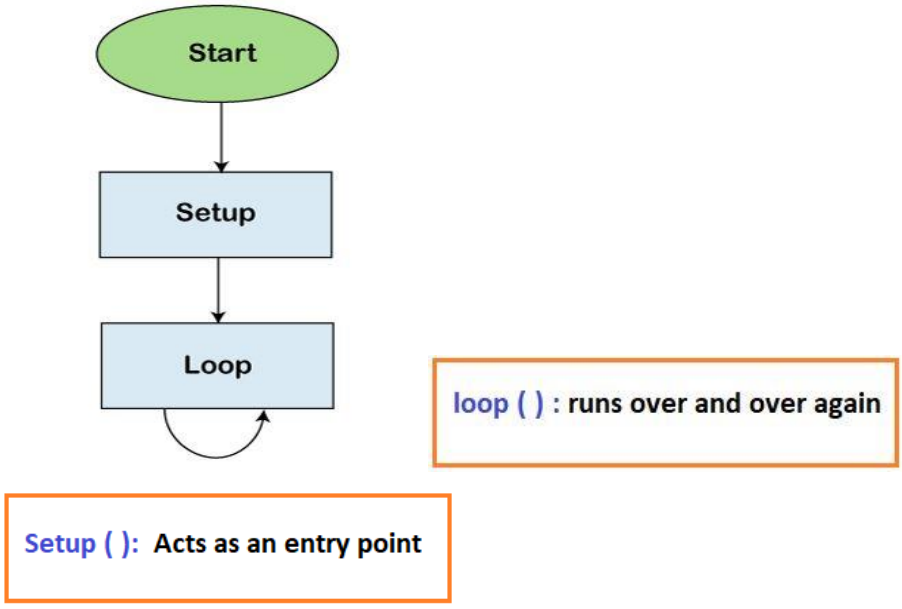


Fig 4.4 Program Flow Chart

4.4.3 Serial Functions

Serial.begin():

The serial.begin( ) sets the baud rate for serial data communication. The **baud** rate signifies the data rate in bits per second.

The default baud rate in Arduino is **9600 bps (bits per second**). We can specify other baud rates as well, such as 4800, 14400, 38400, 28800, etc.

The Serial.begin( ) is declared in two formats, which are shown below:

- begin(speed)
- begin(speed, conFig)

Where,

serial: It signifies the serial port object.

speed: It signifies the baud rate or bps (bits per second) rate. It allows ***long*** data types.

conFig: It sets the stop, parity, and data bits.

Serial.print:

The serial.print ( ) in Arduino prints the data to the serial port. The printed data is stored in the ASCII (American Standard Code for Information Interchange) format, which is a human-readable text.

Each digit of a number is printed using the ASCII characters.

The printed data will be visible in the **serial monitor**, which is present on the right corner on the toolbar.

The Serial.print( ) is declared in two formats, which are shown below:

- print(value)
- print(value, format)

serial: It signifies the serial port object.

print: The print ( ) returns the specified number of bytes written.

value: It signifies the value to print, which includes any data type value.

format: It consists of number base, such as OCT (Octal), BIN (Binary), HEX (Hexadecimal), etc. for the integral data types. It also specifies the number of decimal places.

4.4.4 analogRead():

The **analogRead( )** function reads the value from the specified analog pin present on the particular Arduino board. The ADC (Analog to Digital Converter) on the Arduino board is a multichannel converter. It maps the input voltage and the operating voltage between the values 0 and 1023. The operating voltage can be **5V or 3.3V.** The values from 0 to 1023 are the integer values. It can also be written as 0 to (2^10) -1. The time duration to read an analog input signal on the boards (UNO, Mega, Mini, and Nano) is about 100 microseconds or 0.0001 seconds. Hence, the maximum reading rate of analog input is about 10000 times per second.

Let's discuss operating voltage and resolution of some Arduino boards.

- The Operating voltage of Arduino UNO, Mini, Mega, Nano, Leonardo, and Micro is **5V,** and resolution is **10 bits**.
- The Operating voltage of MKR family boards, Arduino Due, and Zero is **3 V,** and resolution is **12 bits**.

4.4.5 Arduino Data Types

The data types are used to identify the types of data and the associated functions for handling the data. It is used for declaring functions and variables, which determines the bit pattern and the storage space. The data types that we will use in the Arduino are listed below:

- void Data Type
- int Data Type
- Char Data Type
- Float Data Type
- Double Data Type
- Unsigned int Data Type

- short Data Type
- long Data Type
- Unsigned long Data Type
- byte data type
- word data type

4.4.6 Arduino Variables

The variables are defined as the place to store the data and values. It consists of a name, value, and type. The variables can belong to any data type such as int, float, char, etc. Consider the url -Arduino data types for detailed information.

Ex: **int pin=8**
Here, the **int** data type is used to create a variable named **pin** that stores the value **8**. It also means that value 8 is initialized to the variable **pin**.
We can modify the name of the variable according to our choice.

4.4.7 Arduino Operators

The operators are widely used in Arduino programming from basics to advanced levels. It plays a crucial role in every programming concept like C, C++, Java, etc. The operators are used to solve logical and mathematical problems. For example, to calculate the temperature given by the sensor based on some analog voltage.

The types of Operators classified in Arduino are:

1. Arithmetic Operators
2. Boolean Operators
3. Comparison Operators
4. Bitwise Operators

1. Arthematic Operators:

There are six basic operators responsible for performing mathematical operations in Arduino, which are listed below:

Assignment Operator (=):

The Assignment operator in Arduino is used to set the variable's value. It is quite different from the equal symbol (=) normally used in mathematics.

Addition (+):

The addition operator is used for the addition of two numbers. For example, P + Q.

Subtraction (-):

Subtraction is used to subtract one value from the another. For example, P - Q.

Multiplication (*):

The multiplication is used to multiply two numbers. For example, P * Q.

Division (/):

The division is used to determine the result of one number divided with another. For example, P/Q.

Modulo (%):

The Modulo operator is used to calculate the remainder after the division of one number by another number.

2. Boolean Operators:

The Boolean Operators are NOT (!), Logical AND (& &), and Logical OR (| |).

Let's discuss the above operators in detail.

Logical AND (& &):

The result of the condition is true if both the operands in the condition are true.

Logical OR (| |):

The result of the condition is true, if either of the variables in the condition is true.

NOT (!):

It is used to reverse the logical state of the operand.

3. Comparision Operators:

The comparison operators are used to compare the value of one variable with the other.

The comparison operators are listed below:

less than (<):

The less than operator checks that the value of the left operand is less than the right operand. The statement is true if the condition is satisfied.

greater than (>):

The less than operator checks that the value of the left side of a statement is greater than the right side. The statement is true if the condition is satisfied.

equal to (= =):

It checks the value of two operands. If the values are equal, the condition is satisfied.

not equal to (! =):

It checks the value of two specified variables. If the values are not equal, the condition will be correct and satisfied.

less than or equal to (< =):

The less or equal than operator checks that the value of left side of a statement is less or equal to the value on right side. The statement is true if either of the condition is satisfied.

greater than or equal to (> =):

The greater or equal than operator checks that the value of the left side of a statement is greater or equal to the value on the right side of that statement. The statement is true if the condition is satisfied.

4. Bitwise Operators:

The Bitwise operators operate at the binary level. These operators are quite easy to use.

There are various bitwise operators. Some of the popular operators are listed below:

bitwise NOT (~):

The bitwise NOT operator acts as a complement for reversing the bits.

bitwise XOR (^):

The output is 0 if both the inputs are same, and it is 1 if the two input bits are different.

bitwise OR (|):

The output is 0 if both of the inputs in the OR operation are 0. Otherwise, the output is 1. The two input patterns are of 4 bits.

bitwise AND (&):

The output is 1 if both the inputs in the AND operation are 1. Otherwise, the output is 0. The two input patterns are of 4 bits.

bitwise left shift (< <):

The left operator is shifted by the number of bits defined by the right operator.

bitwise right shift (> >):

The right operator is shifted by the number of bits defined by the left operator.

4.4.8 Arduino IF statement

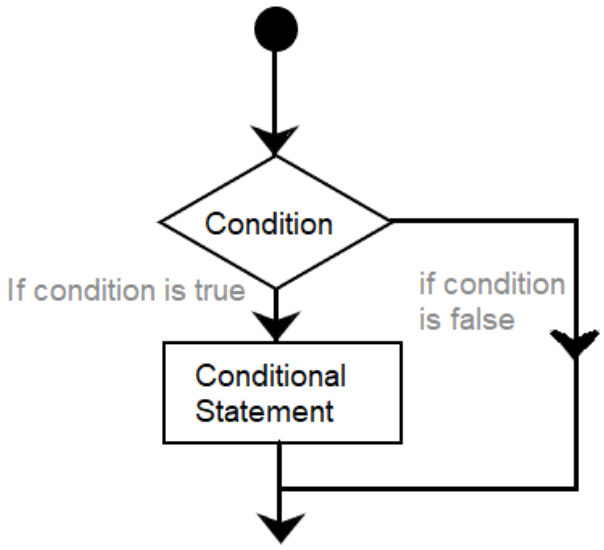


Fig 4.5 Flow chart of IF statement

1.The if ( ) statement is the conditional statement, which is the basis for all types of programming languages. If the condition in the code is true, the corresponding task or

function is performed accordingly. It returns one value if the condition in a program is **true**. It further returns another value if the condition is **false**. It means that if ( ) statement checks for the condition and then executes a statement or a set of statements.

2.If-Else:

The if-else condition includes if ( ) statement and else ( ) statement. The condition in the else statement is executed if the result of the If ( ) statement is false.

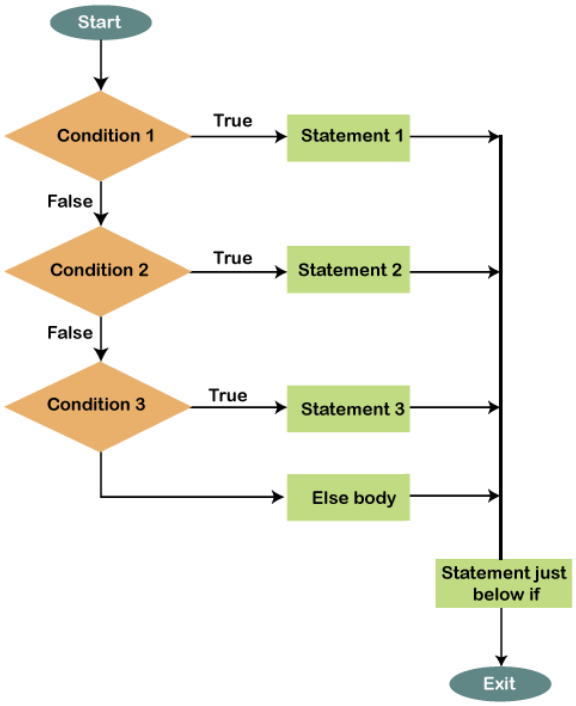


Fig 4.6 Flow Chart of If-Else

The statements will be executed one by one until the true statement is found. When the true statement is found, it will skip all other if and else statements in the code and runs the associated blocks of code.

3.Else-if:

The else if statement can be used with or without the else ( ) statement. We can include multiple else if statements in a program.

The else if ( ) statement will stop the flow once its execution is true.

4.5 ARDUINO SENSORS

The sensors are defined as a machine, module, or a device that detect changes in the environment. The sensors transfer those changes to the electronic devices in the form of a signal. A sensor and electronic devices always work together. The output signal is easily readable by humans. Nowadays, Sensors are used in daily lives. For example, controlling the brightness of the lamp by touching its base, etc. The use of sensors is expanding with new technologies. The sensors are used to measure the physical quantities, such as pressure, temperature, sound, humidity, and light, etc. An example of sensors is Fire Alarm, a detector present on the fire alarm detects the smoke or heat. The signal generated from the detector is sent to the alarming system, which produces an alert in the form of alarm. The types of detectors are smoke detectors, heat detectors, carbon monoxide detectors, multi-sensors detectors, etc. The data signal runs from the sensor to the output pins of the Arduino. The data is further recorded by the Arduino. Some of the types of sensors in Arduino are listed below:

1. Light sensor

The light sensor is used to control the light. It is used with LDR (Light Dependent Resistor) in Arduino.

2. Ultrasonic sensor

The ultrasonic sensor is used to determine the distance of the object using SONAR.

3. Temperature sensor

The temperature sensor is used to detect the temperature around it.

4. Knock Sensor

The knock sensor is used to pick the vibrations of the knocking. It is a common category of a vibration sensor.

5. Object Detection Sensor

It is used to detect the object by emitting infrared radiations, which are reflected or bounced back by that object.

6. Tracking Sensor

It allows the robots to follow a particular path specified by sensing the marking or lines on the surface.

7. Metal Touch Sensor

It is suitable for detecting the human touch.

8. Water Level Sensor

It is used to measure the water or the depth of the water level. It is also used to detect leaks in containers.

9. Vibration Sensor

The vibration sensor is used to measure the vibrations.

10. Air Pressure sensor

It is commonly related to meteorology, biomedical fields.

11. Pulse Sensor

The pulse sensor is used to measure the pulse rate.

12. Capacitive soil moisture sensor

It is used to measure the moisture level of the soil.

13. Microphone sensor

The microphone sensor in Arduino is used to detect the sound. The analog and digital are the two outputs of this module. The digital output sends the high signal when the intensity of sound reaches a certain threshold. We can adjust the sensitivity of a module with the help of a potentiometer.

14. humidity sensor

The humidity sensor is used to monitor weather conditions.

15. Motion sensor

The motion sensor detects the movement and occupancy from the human body with the help of Infrared radiation.

16. Vibration sensor

The vibration sensor is used to detect the vibrations.

17. Sound sensor

The sound sensor is suitable to detect the sound of the environment.

18. Pressure Sensor

The pressure sensor is used to measure the pressure. The sensor in Arduino measures the pressure and displays it on the small LCD screen.

19. Magnetic field sensor

The magnetic field sensor measures the magnetic field strength and produces a varying voltage as the output in Arduino.

CHAPTER 5
RESULTS AND DISCUSSIONS

5.1 ARDUINO SIMULATOR

The Arduino simulator is a **virtual portrayal of the circuits of Arduino** in the real world. We can create many projects using a simulator without the need for any hardware. The Simulator helps to learn, program, and create their projects without wasting time on collecting hardware equipment's.

5.1.1 ADVANTAGES OF USING A SIMULATOR

There are various advantages of using simulator, which are listed below:

- It saves money, because there is no need to buy hardware equipment's to make a project.
- The task to create and learn Arduino is easy for beginners.
- We need not to worry about the damage of board and related equipment's.
- No messy wire structure required.
- It supports line to line debugging, and helps to find out the errors easily.
- Coding can be learned and build projects anywhere with our computer and internet connection.
- We can also share our design with others.

5.1.2 TYPES OF SIMULATOR

There are various simulators available. Some are available for free, while some require a license to access the simulators. Some types of simulators are listed below:

- Autodesk Tinkercad
- Emulator Arduino Simulator
- Autodesk Eagle

- Proteus Simulator
- Virtronics Arduino Simulator
- ArduinoSim

Autodesk Eagle is an advanced simulator, which is used to design 2D and 3D models of PCB, modular designs, multi-sheet schematics, real-time synchronization, etc.

5.1.3 ACCESSING SIMULATOR

Here, we are using the **Autodesk Tinkercad Simulator**.

The steps to access the TINKERCAD are listed below:

1. Open the official website of tinkercad. **URL: https://www.tinkercad.com/**

A window will appear, as shown below:

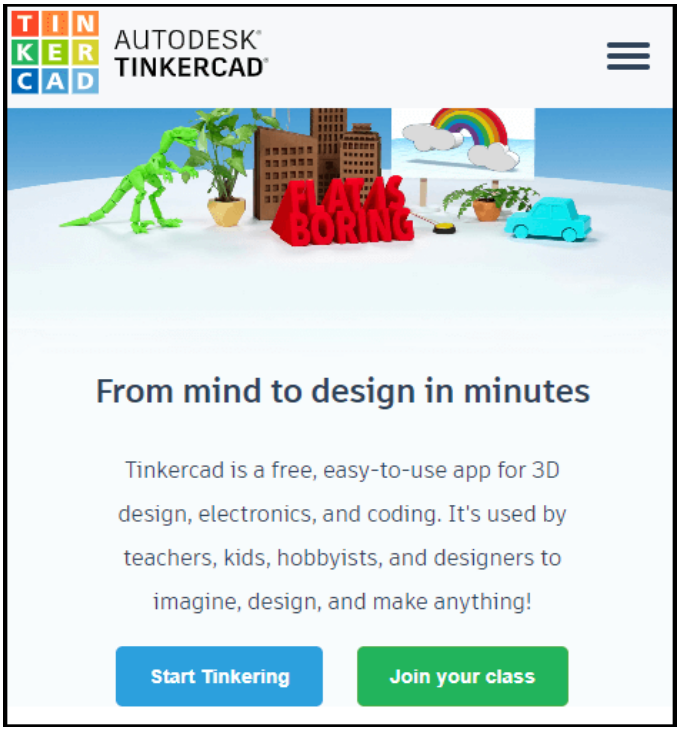


Fig. 5.1 AUTODESK TINKERCAD window

2. Click on the three horizontal lines present on the upper right corner.

3. Click on the '**Sign in**' option, if you have an account in Autodesk. Otherwise, click on the '**JOIN NOW**' option if you don't have an account, as shown below:

Fig 5.2 Menu Window

The **SIGN IN** window will appear as:

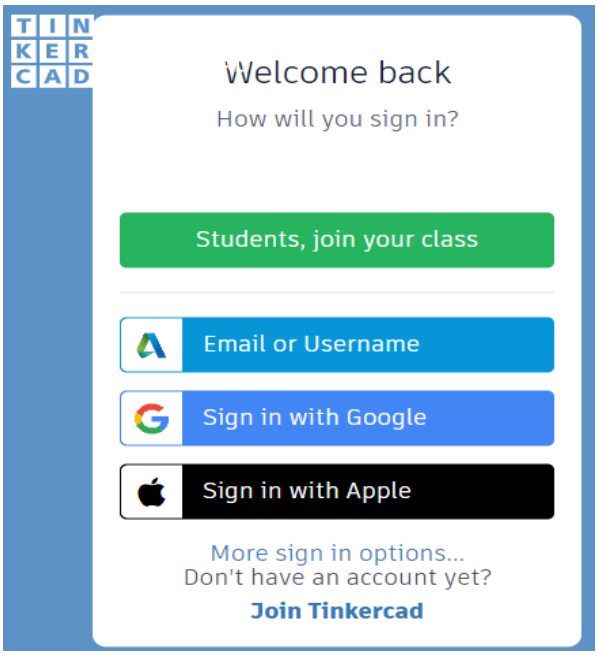


Fig 5.3 Sign in Window

We can select any sign-in method. Specify the username and password. We already have an account in Autodesk, so we will sign-in directly with the username and password.

The **JOIN** window will appear as:

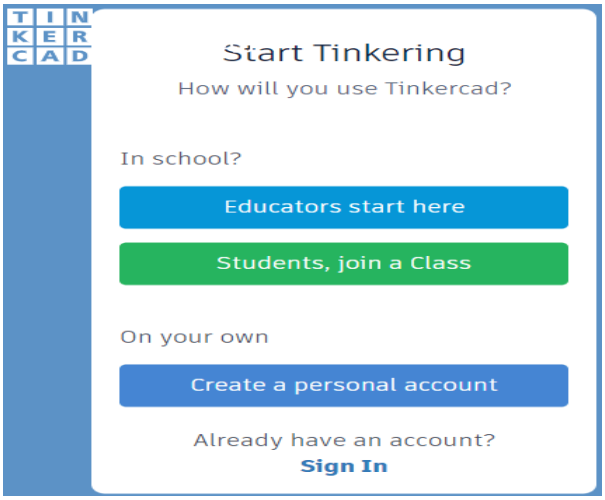


Fig 5.4 Join window

Select the preference according to the requirements and sign-in using Gmail, etc.

4. Now, a window will appear, as shown below:

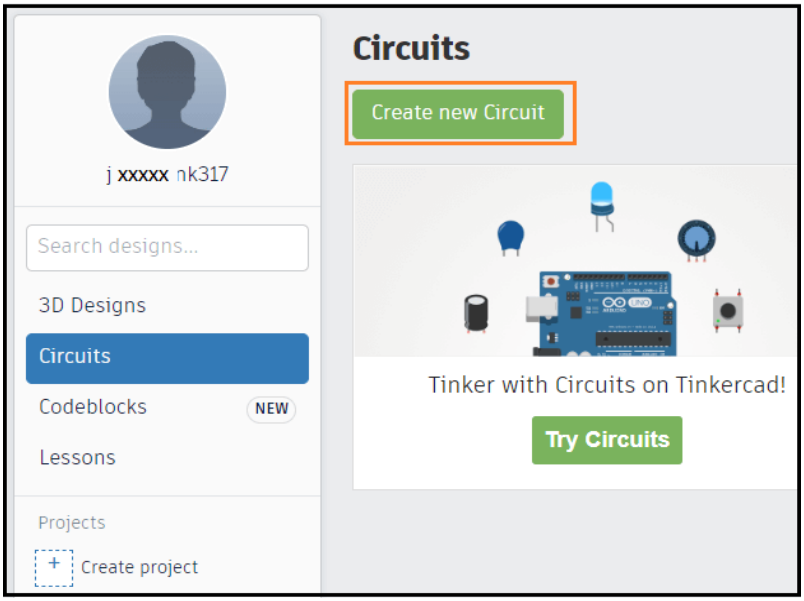


Fig 5.5 Tinkercad Window

5. Click on the '**Create new circuit**' option to start designing the Arduino circuit, as shown above. The '**Circuits**' option will also show the previous circuits created by user. The design option is used for creating the 3D design, which is of no use in Arduino.

6. We are now ready to start with the Autodesk Tinkercad. We can start creating our projects.

5.1.4 FEATURES OF SIMULATOR

- Glow and move circuit assembly. It means we can use the components of a circuit according to the project requirement. Glow here signifies the glowing of LED.
- Integrated product design. It means the electronic components used in the circuitry are real.
- Arduino Programming. We can directly write the program or code in the editor of the simulator.
- We can also consider some ready-made examples provided by the tinkercad for better understanding.
- Realtime simulation. We can prototype our designs within the browser before implementing them in real-time.

5.2 FLEX SENSOR SIMULATION

5.2.1 Connection Of Flex Sensor

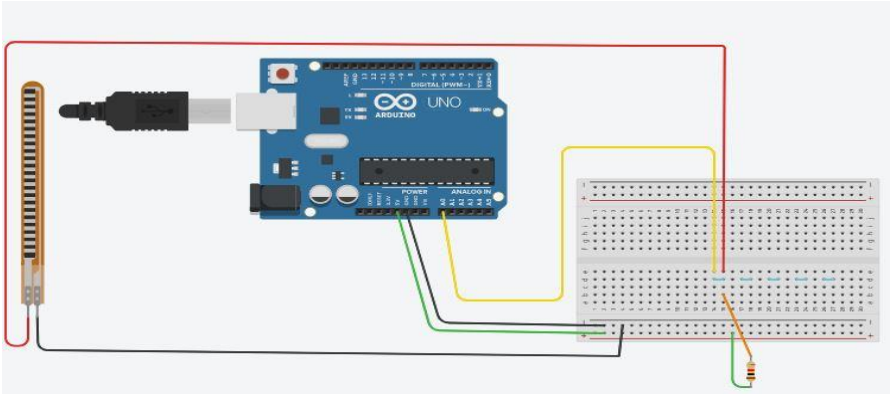


Fig 5.6 Flex Sensor connection in tinkercad

One pin of Flex Sensor is connected to Ground and another pin is connected to +5V through 1Kohm resistor.

5.2.2 Results Of Flex Sensor

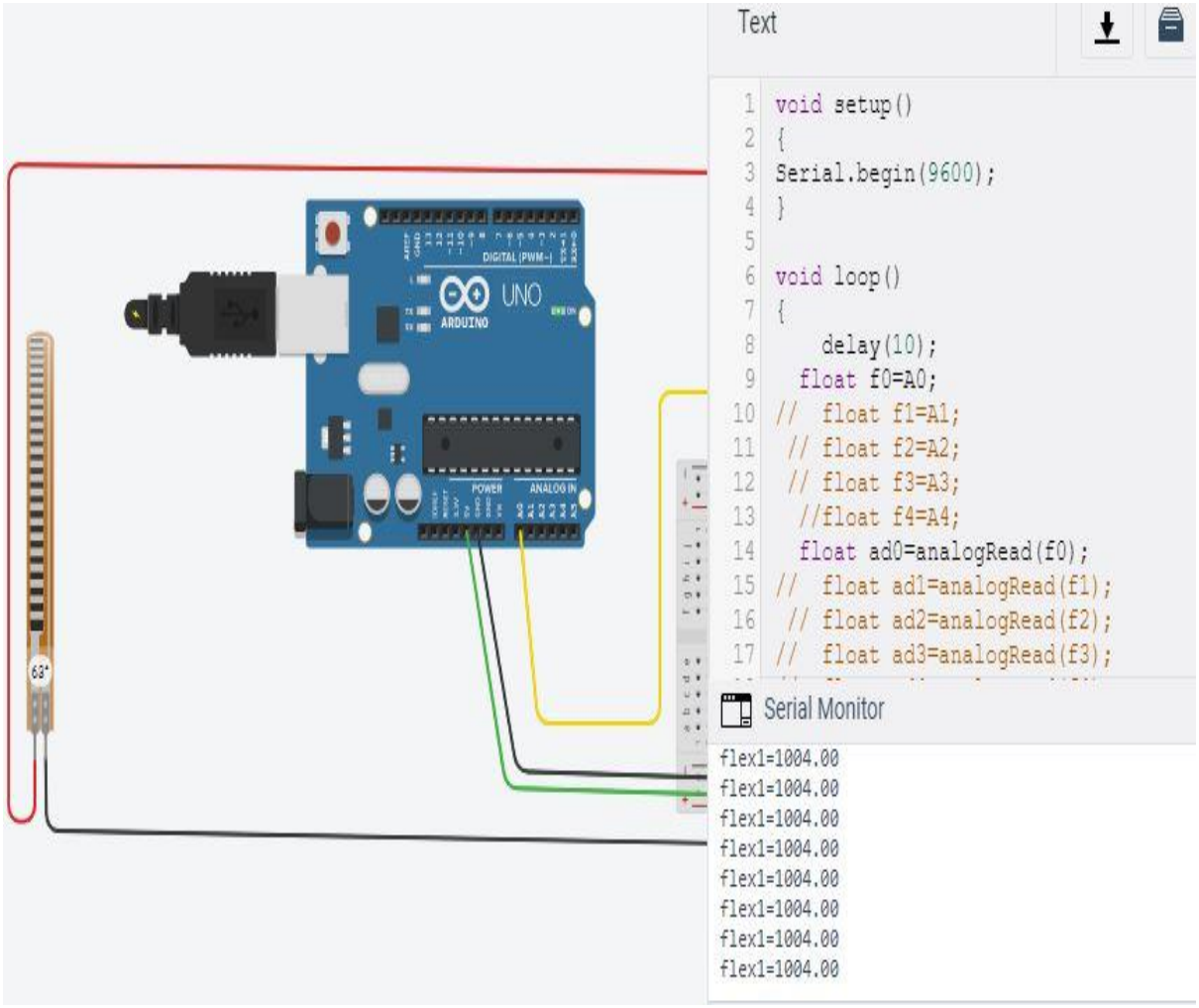


Fig 5.7 Output of Flex Sensor

Here, we had made a bend in the flex sensor. The bend angle is 63 degrees, So According to that end in the flex sensor, The output is printed in the Serial Monitor.

5.3 ACCELEROMETER SIMULATION

5.3.1 Connection Of Accelerometer

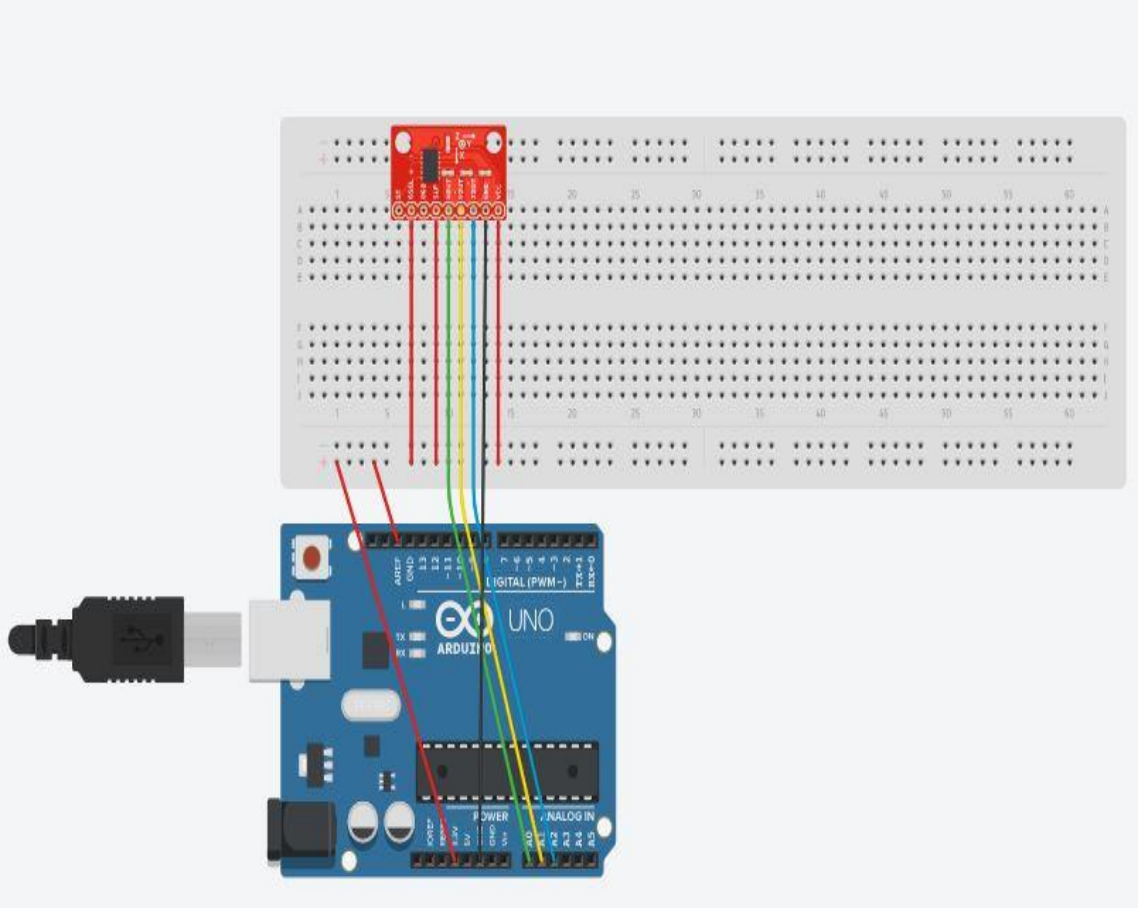


Fig 5.8 Accelerometer Connection

5.3.2 Results Of Accelerometer

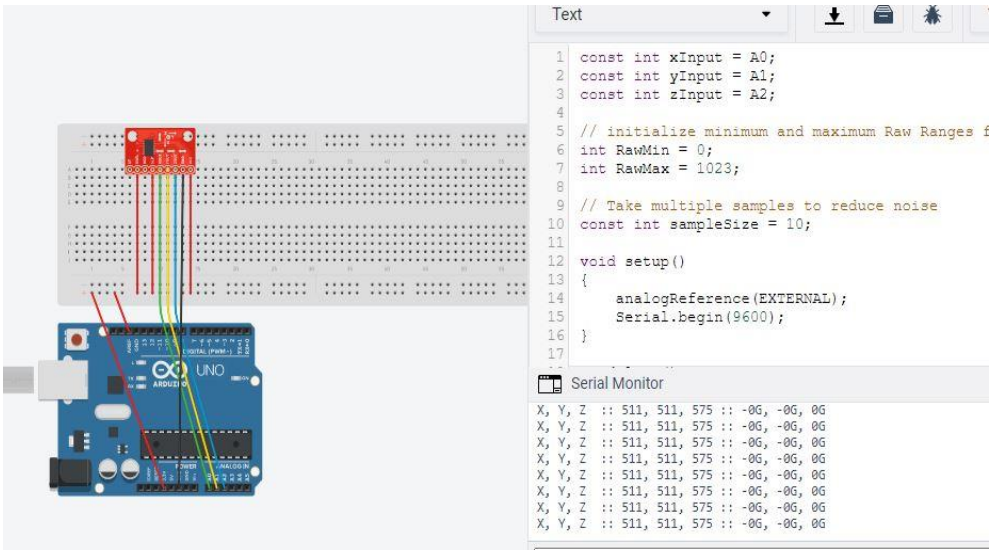


Fig 5.9 Output of Accelerometer

5.4 TOUCH SENSOR SIMULATION

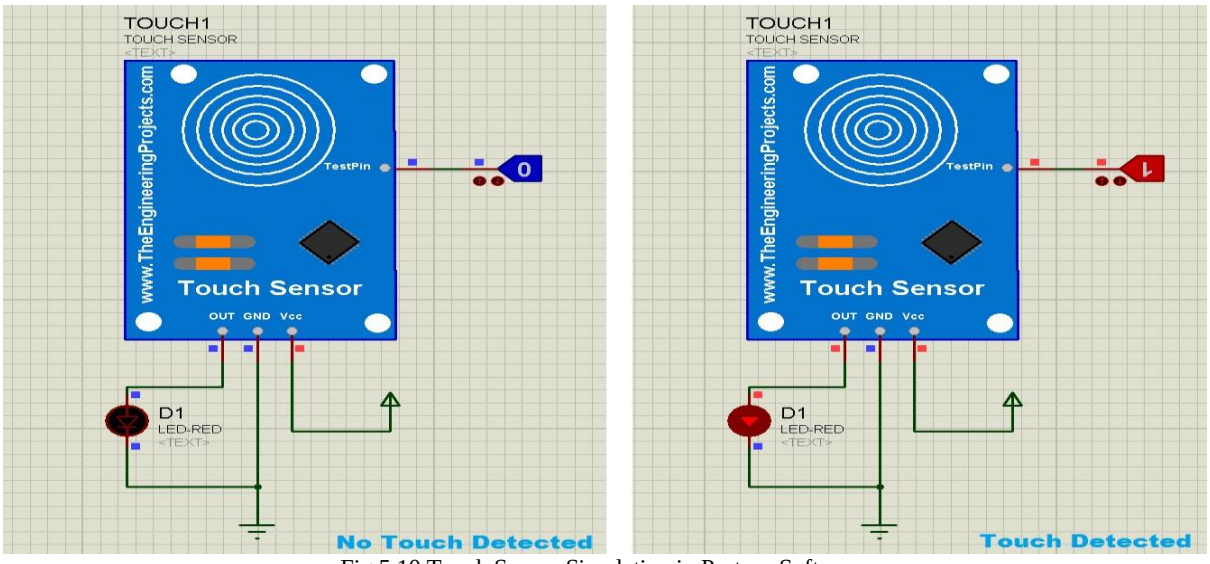


Fig 5.10 Touch Sensor Simulation in Proteus Software

If there is any touch detected the output value will be 1 and if there is no touch detected then the output will be 0.

RESULTS:

During the implementation, all the three sensors are connected to Arduino Uno board using jumper wires. Once the connections are made perfectly, then Arduino takes inputs from three sensors (Flex sensor, accelerometer, Touch sensor). Flex sensors are placed on fingers which measure the bending of fingers according to the gesture made with the glove. An accelerometer is placed on the palm which measures the location of the hand in X, Y, Z axes. Touch sensor is placed on the palm and detects if there is any touch between the fingers and palm. Initially, only flex sensors are implemented in this sign language transition. But, some hand gestures are similar to other gestures. To distinguish these types of gestures an additional sensor Accelerometer is also implemented. This is very important in distinguishing two signs when they have the same bend in the fingers but different bends in the palm. Similarly in the case of touch sensor.

The designed circuit has been connected and tested with many Hand gesture where the voice was clear for all the gestures, some examples are shown below

Fig 5.11: yes gesture

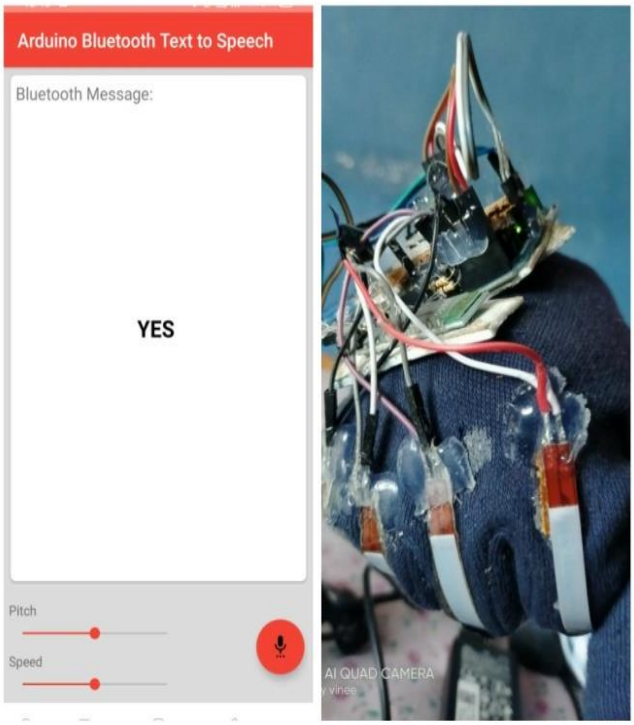


Fig 5.12: output for yes gesture

Fig 5.13: thank you gesture

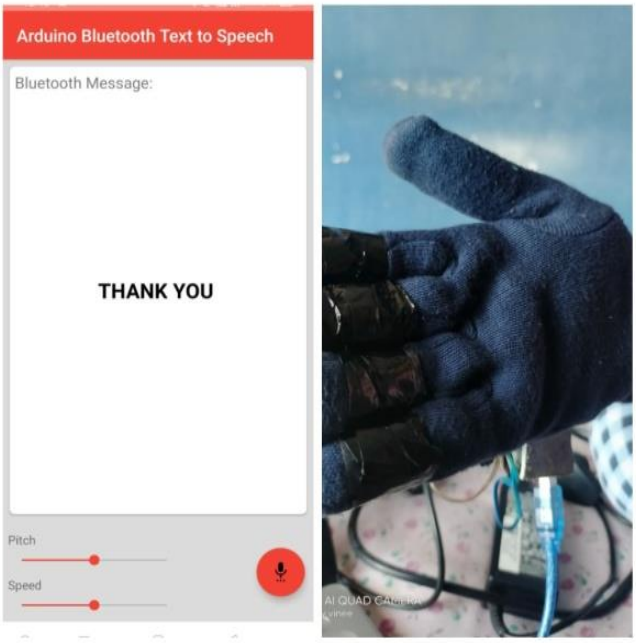


Fig 5.14: output for thank you gesture

Fig 5.15: yes gesture

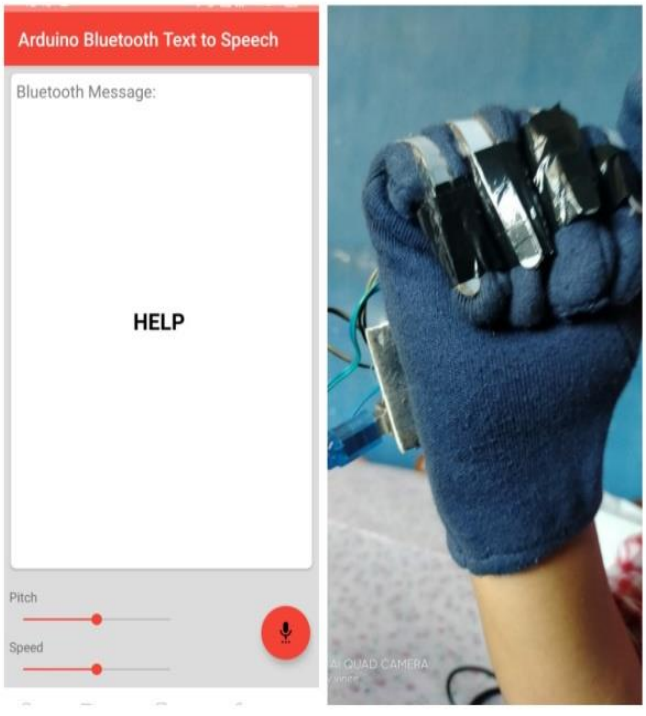


Fig 5.16: output for yes gesture

The gesture manager is the principal part of the recognition system. It contains data to match with incoming data. The system tries to match incoming data with existing posture. The bend values of the fingers and for each posture definition the distance to the current data is calculated. From the convenience of simple flex sensors, a user is able to interact with others in more comfortable and easier manner. This makes it possible for the user to not only interact with their community but with others also and they can also live normal life.

DISCUSSIONS:

During the implementation, each and every single part was tested alone the following points have been observed :

The flex sensors function correctly and accurately. They are able to identify the correct variance emitted by each finger or hand gesture. The controller (Arduino) collects the data and sends it directly to the mobile application through the BT module in an appropriate way. The BT module is able to directly link to the needed device once the system is plugged into the power source or power bank in order to transmit and receive the number variance. One of the most important aspects of this project is the correct classification of the received input, which is forwarded by the mobile app, producing the expected output which is to be translated by the mobile app. into speech and text. At first, the data should be sent from the Arduino to the BT that assures quick and reliable connectivity. The voice output could be understood clearly.

CHAPTER 6
CONCLUSION

In this project, we constructed a Smart-Glove for supporting blind and deaf-blind people in communicating with normal people that are not familiar with braille. The Smart-Glove is able to connect to Android mobile and facilitate exchange of messages. Whereas the android application is able to send and receive text messages from and to the Smart-Glove and the Smart-Glove able to send and receive braille messages from and to the application. The Smart-Glove is light, cheap, easy to use and no risk. We believe that the project is an effective and very useful for deaf-blind people if they are taught braille where they can communicate with their families and people around them.

As future scope of the project, the system may be extended to support other languages, and the system can use several way to communicate, it can use Wi-Fi connection, which enables a faster connection and better range from the base station or GSM module (Global System for Mobile communication) GSM is the most widespread and it's a cellular technology used for transmitting mobile data services, the most obvious advantage of it is widespread use throughout the world.

REFERENCES

[1] D. J. Sturman and D. Zeltzer, “A survey of glove-based input,” IEEE Comput. Graph. Appl., vol. 14, no. 1, pp. 30–39, Jan. 1994.

[2] T. Takahashi and F. Kishino, “Hand gesture coding based on experiments using a hand gesture interface device,” SIGCHI Bull., vol. 23, no. 2, pp. 67–74, Apr. 1991.

[3] K. Murakami and H. Taguchi, “Gesture recognition using recurrent neural networks,” in Proc. Conf. Human Factors Comput. Syst., 1991, pp. 237–242.

[4] J. L. Hernandez-Rebollar, R. W. Lindeman, and N. Kyriakopoulos, “A multi-class pattern recognition system for practical finger spelling translation,” in Proc. IEEE Int. Conf. Multimodal Interfaces, 2002, pp. 185– 190.

[5] J. S. Kim, W. Jang, and Z. Bien, “A dynamic gesture recognition system for the Korean sign language (KSL),” IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 26, no. 2, pp. 354–359, Apr. 1996.

[6] W. Kadous, “GRASP: Recognition of Australian sign language using instrumented gloves,” Bachelor’s thesis, Univ. New South Wales, Sydney, Australia, 1995.

[7] P. Vamplew, “Recognition of sign language gestures using neural networks,” presented at the Eur. Conf. Disabilities, Virtual Reality Associated Technol., Maiden

[8] W. Gao, J. Ma, J. Wu, and C. Wang, “Sign language recognition based on HMM/ANN/DP,” Int. J. Pattern Recognit. Artif. Intell., vol. 14, no. 5, pp. 587–602, 2000.

[9] C. Wang, W. Gao, and S. Shan, “An approach based on phonemes to large vocabulary Chinese sign language recognition,” in Proc. IEEE Int. Conf. Autom. Face Gesture Recognit., 2002, pp. 393–398.

[10] R. H. Liang and M. Ouhyoung, “A real-time continuous alphabetic sign language to speech conversion VR system,” Comput. Graph. Forum, vol. 14, no. 3, pp. 67–76, 1995.

[11] R. H. Liang and M. Ouhyoung, “A sign language recognition system using hidden Markov Model and context sensitive search,” in Proc. ACM Symp. Virtual Reality Softw. Technol., 1996, pp. 59–66.

[12] R. H. Liang and M. Ouhyoung, “A real-time continuous gesture recognition system for sign language,” in Proc. IEEE Int. Conf. Autom. Face Gesture Recognit., 1998, pp. 558–567.

[13] S. Fels and G. E. Hinton, “Glove Talk 2: A neural network interface which maps gestures to parallel formant speech synthesizer controls,” IEEE Trans. Neural Netw., vol. 9, no. 1, pp. 205–212, Jan. 1998. [14] W. J. Greenleaf, “Developing the tools for pratical VR applications,” IEEE Eng. Med. Biol. Mag., vol. 15, no. 2, pp. 23–30, Mar./Apr. 1996.

[15] W. Gao, J. Ma, S. Shan, X. Chen, W. Zeng, H. Zhang, J. Yan, and J. Wu, “Handtalker: A multimodal dialog system using sign language and 3-d virtual human,” in Proc. Int. Conf. Adv. Multimodal Interfaces, 2000, pp. 564–571.

[16] J. Kramer and L. Leifer, “The talking glove: An expressive and receptive verbal communication aid for deaf, deaf-blind and non-vocal,” Stanford Univ., Uninc Santa Clara County, CA, Tech. Rep., 1989