# SMART SIGN GLOVES FOR DISABLED PEOPLE

**A PROJECT REPORT**

**APRIL 2024**

***Submitted by***


| NAME OF THE CANDIDATE | REGISTER NO |
|---|---|
| ASWIN AADHITHYA.M | 21EE03 |
| KAVIYARASU.A | 21EE15 |
| SARAN.M | 21EE23 |
| SASI PRASANTH.A | 21EE24 |
| SIVA KUMAR.G | 21EE25 |

**Guided by**

**Dr. K.BHUVANESWARI M.E., Ph.d.,**

**DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING**




Estd : 1957

**NACHIMUTHU POLYTECHNIC COLLEGE**

**POLLACHI-642003.**


*Government Aided * Autonomous Institution * Approved by AICTE, New Delhi.*
*Affiliated to State Board of Technical Education and Training, Tamil Nadu.*
*Accredited by National board of Accreditation (NBA) - (Civil, Mech & ECE) & APACC*
*(Asia Pacific Accreditation and Certification Commission),*
*Philippines with Gold Medal*

# NACHIMUTHU POLYTECHNIC COLLEGE
# POLLACHI 642003.

**Government Aided; Approved by AICTE, New Delhi.**

**Affiliated to state board of Technical Education & Training,**

**Tamilnadu**

**Accredited by NBA (Civil,Mech,ECE)**

**BONAFIDE CERTIFICATE**

Certified that this project report entitled

# SMART SIGN GLOVES FOR DISABLED PEOPLE

Is the bonafide record work carried out by

| NAME OF THE CANDIDATE | REGISTER NO |
|---|---|
| SASI PRASANTH.A | 21EE24 |

During the academic year 2023-2024

**GUIDE**

**Dr. K.BHUVANESWARI M.E.,Ph.d.,**

**HEAD OF THE DEPARTMENT**

**Dr. K.BHUVANESWARI M.E.,Ph.d.,**

**INTERNAL EXAMINER**

**NAME:**

**DESIGNATION:**

**EXTERNAL EXAMINER**

**NAME:**

**DESIGNATION:**

# ACKNOWLEDGEMENT

We pay our sincere and honourable salutation to the almighty for equipping us with all the strength and courage throughout our venture in the project.

At this pleasing moment of having successfully completed our project with proper guidance, we wish to convey our sincere and heartful thanks and gratitude to our honorable chairman **Dr. M. MANICKAM M.Sc., MBA.,** for his encouragement throughout our endeavour. We also express our sincere and heartful thanks to our Correspondent **Shri. M. HARIHARA SUDHAN, B.E., MS.** for his motivation and support to complete the project.

We would like to express our sincere and heartful thanks to our beloved secretary **Dr. C. RAMASAMY M.E., Ph.d.,** for his encouragement and uplifting us throughout our endeavour. We would like to express our sincere and heartful thanks to our beloved Principal **Dr. M. ASHOK. M.E., Ph.d.,** for forwarding us to do our project and offering adequate facilities for completing our project in a successful manner.

We are also grateful to **Dr. K. BHUVANESWARI M.E.,Ph.d** Head of the Department and **Mr. S.RANGANATHAN M.E.,** Head of the Section for their constructive suggestions and encouragement during our project.

We would like to convey our sincere and heartful thanks to our project guide **Dr. K. BHUVANESWARI M.E.,Ph.d** for her constructive guidance and valuable suggestions

The various suggestions related to this project work was collected from various staff members to **ELECTRICAL AND ELECTRONICS ENGINEERING DEPARTMENT** and this kind of encouragement from our staff has made this project work as successful one. We are much thankful for all other teaching and non-teaching staff members of our institution

## TABLE OF CONTENTS

## TABLE OF CONTENTS

## LIST OF FIGURES

# SMART SIGN GLOVES FOR DISABLED PEOPLE.


## ABSTRACT:

Nowadays, there is a challenging effort to communicate with people who have hearing difficulties. There is a communication barrier because the sign language used by these people is not understood by the general public. Paralyzed people also frequently need assistance. The Smart Sign Gloves represent a promising advancement in assistive technology, providing an innovative solution for disabled individuals to communicate effectively. The proposed system uses IoT-based smart support gloves for disabled persons as a solution for these people. Compared to the existing system, the gloves are straight forward and more effective. The gloves are equipped with an array of flex sensors strategically placed on the fingers. These sensors capture the wearer's hand movements and finger gestures, which are then processed by an onboard arduino nano. A mobile app interprets the gestures and converts them into meaningful communication in real time. In an emergency, a warning message will be transmitted quickly to mobile application via the Bluetooth Module.

# CHAPTER-1
# INTRODUCTION

# CHAPTER-1

# INTRODUCTION

## 1.1 INTRODUCTION:

Nowadays, there is a challenging effort to communicate with people who have hearing difficulties. There is a communication barrier because the sign language used by these people is not understood by the general public. Paralyzed people also frequently need assistance. The Smart Sign Gloves represent a promising advancement in assistive technology, providing an innovative solution for disabled individuals to communicate effectively and participate more in society. The proposed system uses IoT-based smart support gloves for disabled persons as a solution for these people. Compared to the existing system, the gloves are straight forward and more effective. The gloves are equipped with an array of flex sensors strategically placed on the fingers. These sensors capture the wearer's hand movements and finger gestures, which are then processed by an onboard arduino. A mobile app interprets the gestures and converts them into meaningful communication. In an emergency, a warning message will be transmitted via the Bluetooth Module.

## 1.2 LITERATURE SURVEY:

Research into smart sign gloves using Arduino Nano for disabled individuals has gained traction in recent years due to the pressing need for more accessible communication technologies. Several studies have explored various aspects of this technology, including sensor integration, gesture recognition algorithms, user experience, and practical applications.

Sensor Integration Studies have focused on integrating flex sensors to accurately capture hand movements and gestures. Research has demonstrated the effectiveness of combining multiple sensors to enhance gesture recognition accuracy and reduce false positives.

User Experience User-centered design approaches have been employed to assess the usability and acceptability of smart sign gloves among individuals with disabilities. Studies have investigated factors such as comfort, ergonomics, ease of use, and customization options to enhance the overall user experience and promote long-term adoption.

Practical Applications Research has explored various practical applications of smart sign gloves beyond basic communication, including educational tools, assistive devices for daily living, and interfaces for accessing digital content. Studies have highlighted the potential of smart sign gloves to empower individuals with disabilities and promote social inclusion.

Overall, background research on smart sign gloves using Arduino Nano for disabled individuals underscores the importance of multidisciplinary approaches integrating sensor technology, signal processing algorithms, and user-centered design principles to develop effective and accessible communication solutions. Further research is needed to address technical challenges, refine system performance, and evaluate real-world usability in diverse user populations.

## 1.3 PROBLEM STATEMENT:

Communication Barrier for the Deaf and Hard of Hearing: Address the challenge faced by the deaf and hard of hearing community in communicating effectively with those who don't understand sign language.

**Limited Accessibility to Sign Language Interpreters:**

Many regions lack access to sign language interpreters, especially in remote areas or during emergencies.

**Real-time Translation**:

Develop a solution that translates sign language gestures into spoken or written language in real-time to facilitate communication between sign language users and non-signers.

**Portability and Convenience**:

Create a portable and easy-to-use device that individuals can carry with them for communication purposes wherever they go.

**Affordability:**

Design a cost-effective solution that is accessible to a wide range of users, including those from low-income backgrounds or developing countries.

**Integration with Digital Platforms**:

Explore options for integrating the Smart Sign Glove with digital platforms like smartphones or computers for seamless communication.

**User-Friendly Interface:**

Ensure the device has an intuitive user interface, making it easy for both sign language users and non-signers to operate and understand.

**Customization and Personalization:**

Allow users to customize and personalize the device settings according to their preferences and needs.

**Battery Life and Power Consumption:**

Optimize the device's battery life and power consumption to ensure long-lasting usability without frequent recharging.

## 1.4 RESEARCH OBJECTIVES:

The main objective of this project is to provide communication between disabled people and public through text and speech.

- To design a smart glove which converts sign language into text an spech.
- To provide a easily operatble glove using arduino nano.
- To develop a good condition of device with good connectivity using modern technology.
- In this project, we design a simple embedded system based device for problem solving.

## 1.5 SCOPE OF RESEARCH:

**Communication Barrier for the Deaf and Hard of Hearing**:

Address the challenge faced by the deaf and hard of hearing community in communicating effectively with those who don't understand sign language.

**Limited Accessibility to Sign Language Interpreters:**

Many regions lack access to sign language interpreters, especially in remote areas or during emergencies.

**Real-time Translation:**

Develop a solution that translates sign language gestures into spoken or written language in real-time to facilitate communication between sign language users and non-signers.

**Sign Language Recognition Algorithms:**

Investigate existing algorithms or develop novel ones for accurately recognizing sign language gestures captured by sensors embedded in the glove.

**Sensor Integration and Calibration:**

Explore different sensor types (such as flex sensors, accelerometers, gyroscopes) and their optimal placement on the glove. Determine calibration methods to ensure accurate gesture detection.

**Bluetooth Communication Protocol:**

Research Bluetooth communication protocols compatible with Arduino Nano and HC05 module for establishing reliable communication between the glove.

## 1.6 PROJECT SIGNIFICANCE:

Developing a smart sign glove for disabled individuals using Arduino Nano and a Bluetooth module like HC-05 holds immense significance in the realm of assistive technology. By leveraging these accessible and affordable technologies, the project aims to enhance the lives of people with disabilities in several ways. Firstly, it promotes inclusivity by providing a means for individuals with communication impairments to express themselves more effectively, thereby reducing barriers to social interaction and participation. Secondly, the glove promotes independence by enabling users to communicate without relying on constant assistance from others, empowering them to navigate their daily lives with greater autonomy. Additionally, the customizable nature of the glove allows for tailored solutions to meet diverse needs and preferences, ensuring that it can be adapted to suit individual requirements. Beyond its immediate practical benefits, this project also contributes to raising awareness about the challenges faced by people with disabilities and fosters innovation within the maker community through its open-source development approach. Overall, the smart sign glove represents a promising example of how technology can be harnessed to promote accessibility, inclusion, and empowerment for individuals with

disabilities.Furthermore, the project's use of Arduino Nano and the HC-05 Bluetooth module showcases the potential for cost-effective solutions in the field of assistive technology. By utilizing readily available and affordable components, it demonstrates that impactful innovations can be achieved without the need for prohibitively expensive equipment.

This aspect is particularly significant in contexts where resources may be limited, making the technology more accessible to a broader range of users. Moreover, the open-source nature of the project encourages collaboration, knowledge sharing, and further iteration, potentially leading to improvements and adaptations by a diverse community of developers and users. Ultimately, the development of a smart sign glove not only addresses practical communication challenges but also contributes to a more inclusive and supportive society, where individuals of all abilities can participate and thrive.

## 1.7 CHAPTER SUMMARY:

To summarize,this smart sign glove utilizes an Arduino Nano microcontroller and a Bluetooth module HC-05 to create a wearable device for individuals with hearing or speech impairments. The Arduino Nano processes data from sensors embedded in the glove, detecting hand gestures used in sign language. The HC-05 Bluetooth module enables wireless communication with a smartphone or other devices, allowing the glove to transmit interpreted sign language into text or speech. This technology enhances communication accessibility for the deaf and hard of hearing community by bridging the gap between sign language and spoken or written language.

# CHAPTER-2
# BLOCK DIAGRAM

## 2.1 BLOCK DIAGRAM:

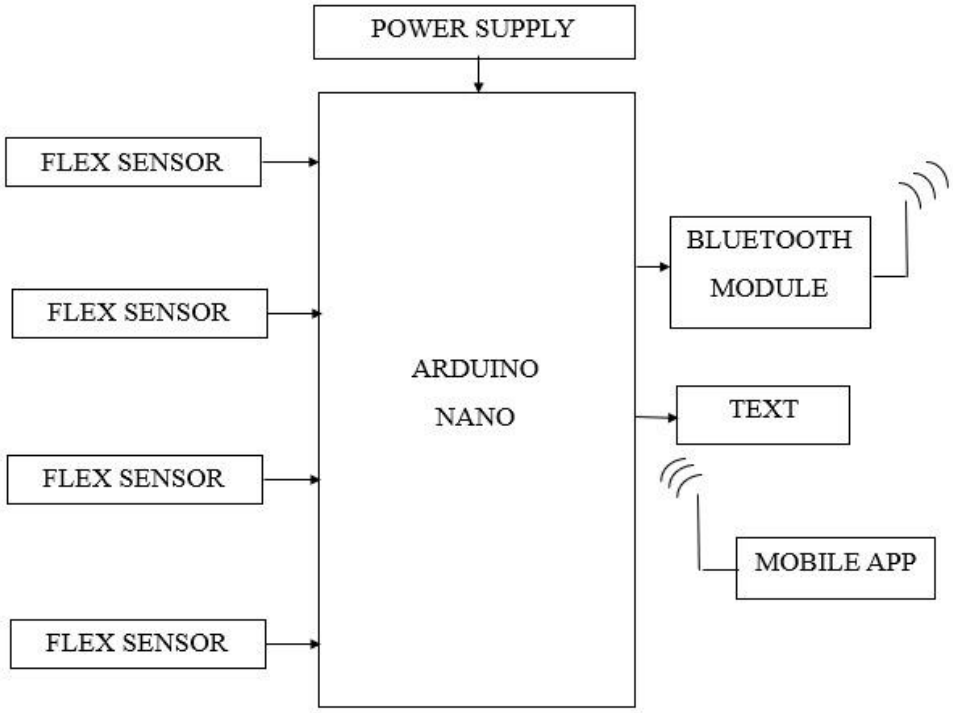



**Fig 2.1 Block Diagram**

The smart sign glove converts sign language to text and voice through the Bluetooth module. Here four flex sensors are used and fixed on four fingures. When we fold an figure the flex sensor bends and the resistance value of the flex sensor is varied.

This varied resistance value is given to the analog input pins of ardunino nano and arduino nano process the data according to the coded instructions and the output will be given to the Bluetooth module . By installing arduino Bluetooth text to speech converter app and connecting to Bluetooth module (HC - 05) the respective voice and text output will be sounded and displayed.

# CHAPTER-3

# HARDWARE DESCRIPTION

## 3.1 ARDUINO NANO:

Arduino boards are widely used in robotics, embedded systems, automation, Internet of Things(IoT) and electronics. These boards were initially introduced for students.

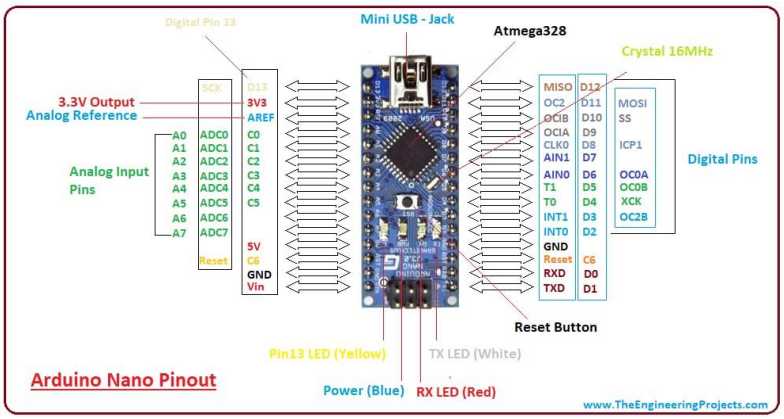



**Fig 3.1 Arduino nano pinout**

### 3.1.1 Key Points Of Arduino Nano:

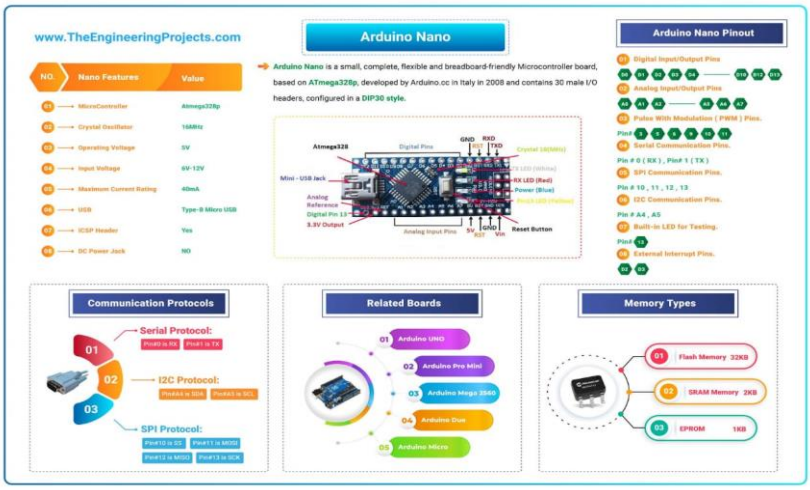



**Fig 3.2 Arduino nano key points**

### 3.1.2 Features of Arduino Nano:

| No. | Nano Features | Value |
|---|---|---|
| `1 | Microcontroller | Atmega328p |
| 2 | Crystal Oscillator | 16MHz |
| 3 | Operating Voltage | 5V |
| 4 | Input Voltage | 6V-12V |
| 5 | Maximum Current Rating | 40Ma |

### 3.1.3 Overview of Arduino Nano Pinout:

.

| No. | Pin Number | Pin Description |
|---|---|---|
| 1 | D0 - D13 | Digital Input / Output Pins |
| 2 | A0 - A7 | Analog Input / Output Pins |
| 3 | Pin # 3, 5, 6, 9, 10, 11 | Pulse Width Modulation ( PWM ) Pins |
| 4 | Pin # 0 (RX) , Pin # 1 (TX) | Serial Communication Pins. |
| 5 | Pin # 10, 11, 12, 13 | SPI Communication Pins. |
| 6 | Pin # A4, A5 | I2C Communication Pins. |
| 7 | Pin # 13 | Built-In LED for Testing. |
| 8 | D2 & D3 | External Interrupt Pins |

### 3.1.4 Arduino Nano offers three types of communications protocols:

| No. | Communication Protocols | Description |
|---|---|---|
| 1 | Serial Port | 1 (Pin#0 is RX, Pin#1 is TX). |
| 2 | I2C Port | 1 (Pin#A4 is SDA, Pin#A5 is SCL). |
| 3 | SPI Port | 1 (Pin#10 is SS, Pin#11 is MOSI, Pin#12 is MISO, Pin#13 is SCK). |

### 3.1.5 Memory details present in Arduino Nano:

| No. | Memory Type | Value |
|---|---|---|
| 1 | Flash Memory | 32KB |
| 2 | SRAM Memory | 2KB |
| 3 | EEPROM | 1KB |

### 3.1.6 Compare with other Arduino Boards:

- Arduino UNO
- Arduino Pro Mini
- Arduino Mega 2560
- Arduino Due

- Arduino Micro
- Arduino Lilypad

### 3.1.7 Introduction to Arduino Nano


**Fig 3.3 Arduino nano specification**

**Arduino Nano** is a small, complete, flexible and breadboard-friendly Microcontroller board, based on ATmega328p, developed by Arduino.cc in Italy in 2008 and contains 30 male I/O headers, configured in a DIP30 style. **Arduino Nano Pinout** contains 14 digital pins, 8 analog Pins, 2 Reset Pins & 6 Power Pins.It is programmed using Arduino IDE, which can be downloaded from the Arduino Official site.**Arduino Nano** is simply a smaller version of Arduino UNO, thus both have almost the same functionalities.

It comes with an operating voltage of 5V, however, the input voltage can vary from 7 to 12V. Arduino Nano's maximum current rating is 40mA, so the load attached to its pins shouldn't draw a current more than that. Each of these Digital & Analog Pins is assigned with multiple functions but their main function is to be configured as Input/Output. Arduino Pins are acted as Input Pins when they are interfaced with sensors, but if you are driving some load then we need to use them as an Output Pin.

Functions like pin Mode() and digital Write() are used to control the operations of digital pins while analog Read() is used to control analog pins. The analog pins come with a total resolution of 10-bits which measures the value from 0 to 5V.Arduino Nano comes with a crystal oscillator of frequency 16 MHz. It is used to produce a clock of precise frequency using constant voltage.

There is one limitation of using Arduino Nano i.e. it doesn't come with a DC power jack, which means you can not supply an external power source through a battery.This board doesn't use standard USB for connection with a computer, instead, it comes with Type-B Micro USB .The tiny size and breadboard-friendly nature make this device an ideal choice for most applications where the size of the electronic components is of great concern.

Flash memory is 16KB or 32KB that all depends on the Atmega board i.e. Atmega168 comes with 16KB of flash memory while Atmega328 comes with a flash memory of 32KB. Flash memory is used for storing code. The 2KB of memory out of total flash memory is used for a bootloader.The SRAM memory of 2KB is present in Arduino Nano.Arduino Nano has an EEPROM memory of 1KB.

### 3.1.8 Specifications Of The Arduino Nano Board:

It is programmed using Arduino IDE which is an Integrated Development Environment that runs both offline and online.No prior arrangements are required to run the board. All you need is a board, a mini USB cable and Arduino IDE software installed on the computer.USB cable is used to transfer the program from the computer to the board.No separate burner is required to compile and burn the program as this board comes with a built-in boot-loader.

### 3.1.9 Arduino Nano Pinout

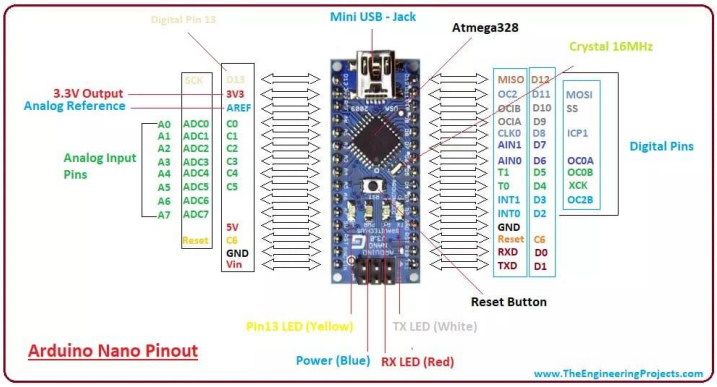



**Fig 3.4 Arduino nano pinout details**

Each pin on the Nano board comes with a specific function associated with it. the analog pins that can be used as analog to digital converters, where A4 and A5 pins can also be used for communication. Similarly, there are 14 digital pins, out of which 6 pins are used for generating PWM.

### 3.1.10 Arduino Nano Power Pins

**Vin**:

It is the input power supply voltage to the board when using an external power source of 7 to 12 V.

**5V:**

It is a regulated power supply voltage of the board that is used to power the controller

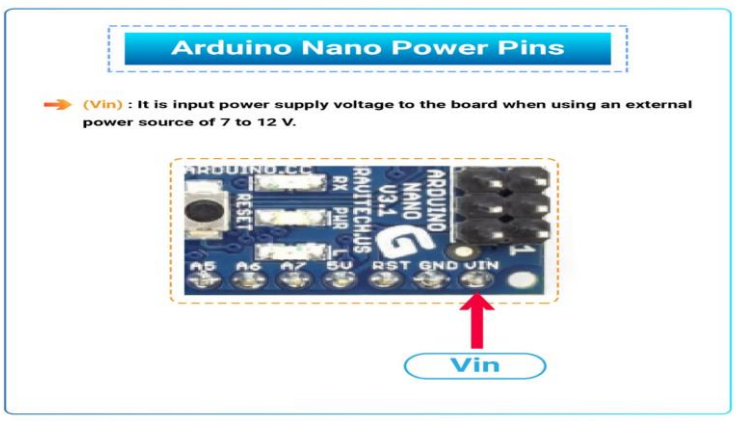



**Fig 3.5 Arduino nano power pins**

### 3.1.11 Arduino Nano Function Pins

**Reset Pin**:

Arduino Nano has 2 reset pins incorporated on the board, making any of these Reset pins LOW will reset the microcontroller.

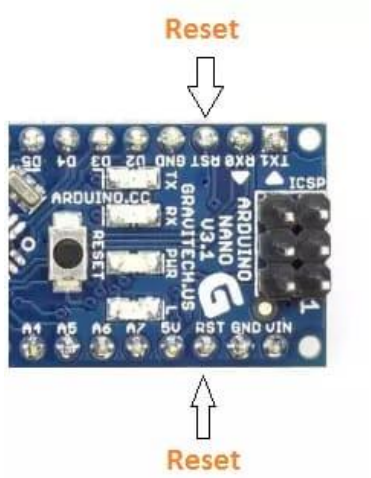



**Fig 3.6 Arduino nano function pins**

**Pin#13:**

A built-in LED is connected to pin#13 of the nano board.This LED is used to check the board i.e. it's working fine or not.

**AREF:**

This pin is used as a reference voltage for the input voltage.

### 3.1.12 Arduino Nano I/O Pins

**Analog Pins**:

There are 8 analog pins on the board marked as A0 - A7.These pins are used to measure the analog voltage ranging between 0 to 5V.


**Fig 3.7 Arduino nano I/O pins**

**Digital Pins**:

Arduino Nano has 14 digital pins starting from D0 to D13.These digital pins are used for interfacing third-party digital sensors and modules with Nano board.

**PWM Pins:**

Arduino Nano has 6 PWM pins, which are Pin#3, 5, 6, 9, 10 and 11. (All are digital pins).These pins are used to generate an 8-bit PWM (Pulse Width Modulation) signal.

**External Interrupts:**

Pin#2 and 3 are used for generating external interrupts normally used in case of emergency, when we need to stop the main program and call important instructions. The main program resumes once interrupt instruction is called and executed.

## 3.1.13 Nano Pinout for Communication Protocols

**Serial Pins:**

These pins are used for serial communication where:

1,Pin#0 is RX used for receiving serial data.

2,Pin#1 is Tx used for transmitting serial data.

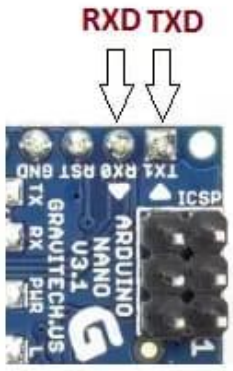



**Fig 3.8 Arduino nano communication pins**

**SPI Protocol:**

Four pins 10(SS->Slave Select), 11(MOSI -> Master Out Slave In), 12(MISO -> Master In Slave Out) and 13(SCK -> Serial Clock) are used for SPI (Serial Peripheral

Interface) Protocol.SPI is an interface bus and is mainly used to transfer data between microcontrollers and other peripherals like sensors, registers, and SD cards.

**I2C Protocol:**

I2C communication is developed using A4 and A5 pins, where A4 represents the serial data line (SDA) which carries the data and A5 represents the serial clock line (SCL) which is a clock signal, generated by the master device, used for data synchronization between the devices on an I2C bus.

## 3.1.14 Arduino Nano Programming & Communication

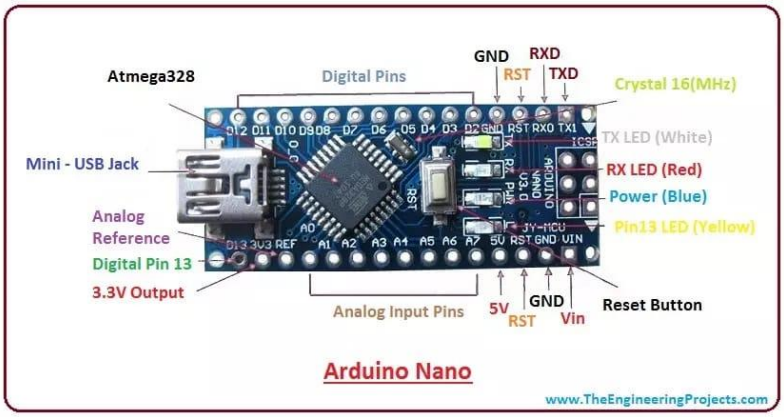



**Fig 3.9 Arduino nano full details**

The Nano board comes with the ability to set up communication with other controllers and computers. The serial communication is carried out by the digital pins, Pin 0(Rx) and Pin 1(Tx) where Rx is used for receiving data and Tx is used for the transmission of data. The serial monitor is added to the Arduino IDE, which is used to transmit textual data to or from the board.

FTDI drivers are also included in the software which behaves as a virtual com port to the software.The Tx and Rx pins come with an LED which blinks as the data is transmitted between FTDI and USB connection to the computer.Arduino Software Serial Library is used for carrying out serial communication between the board and the computer.

Apart from serial communication the Nano board also supports I2C and SPI communication. The Wire Library inside the Arduino Software is accessed to use the I2C

bus. The Arduino Nano is programmed by Arduino Software called IDE which is a common software used for almost all types of board available. Simply download the software and select the board you are using.

Uploading code to Arduino Nano is quite simple, as there's no need to use any external burner to compile and burn the program into the controller and you can also upload code by using ICSP (In-circuit serial programming header).Arduino board software is equally compatible with Windows, Linux or MAC, however, Windows are preferred to use.

### 3.1.15 Arduino Uno Vs Arduino Nano

Both Arduino Uno and Arduino Nano come with the same functionality with little difference in terms of PCB layout, size and form factor. Arduino Uno is a microcontroller board based on Atmega328 and comes with 14 digital I/O pins out of which 6 are PWM. There are 6 analog pins incorporated into the board.

This arduino comes with everything required to support the microcontroller like a USB connection, a Power jack, a 16MHz oscillator, a reset button and an ICSP header. It is a complete ready-to-use device that requires no prior technical skills to get hands-on experience with it. it can power it by using a DC power jack, battery or simply plug it into the computer using a USB cable to get started.

### 3.1.16 Applications of Arduino Nano

Arduino Nano is a very useful device that comes with a wide range of applications and covers less space as compared to other Arduino boards. Breadboard-friendly nature makes it stand out from other boards.

- Medical Instruments
- Industrial Automation
- Android Applications
- GSM Based Projects
- Embedded Systems
- Automation and Robotics
- Home Automation and Defense Systems
- Virtual Reality Applications

## 3.2 FLEX SENSOR :

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 becomes 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.

### 3.2.1 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.

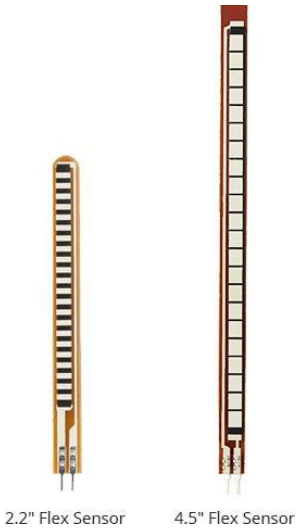



**Fig 3.10 Flex sensors**

### 3.2.2 Construction

A flex sensor consists of a phenolic resin substrate with conductive ink deposited and is **shown in fig.3.**11. A segmented conductor is placed on top to form a flexible potentiometer

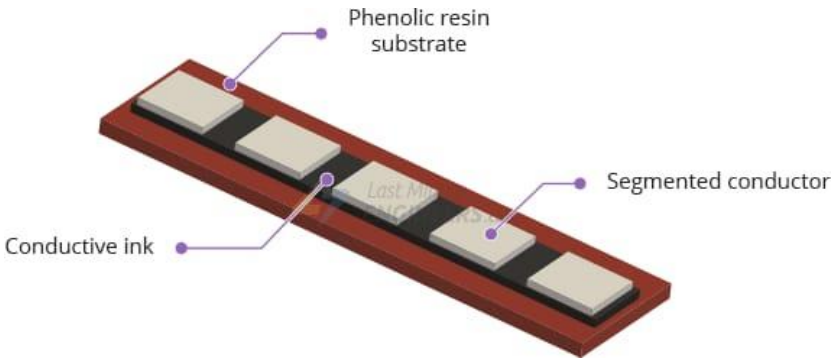



**Fig 3.11 Flex sensor construction**

### 3.2.3 Direction to Use

The flex sensor is only designed to be flexed in one direction, away from the ink, as shown in the image below. If you bend the sensor in the opposite direction, you will not receive accurate data and you may even damage it.Also, avoid bending the sensor too close to the base (where the pins are crimped), as this can cause it to kink and fail.

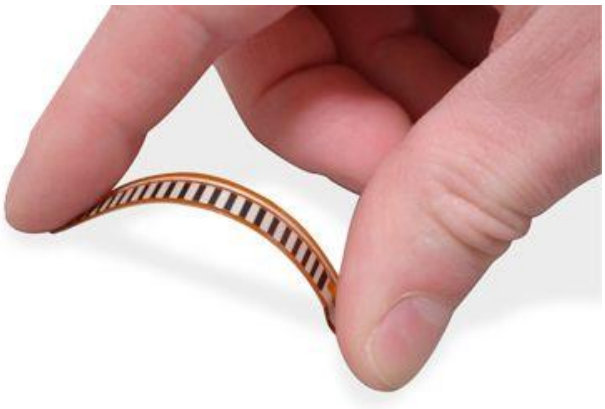


**Fig 3.12 Flex sensor working**

### 3.2.4 Working of Flex Sensor

The conductive ink on the sensor serves as a resistor. When the sensor is straight, this resistance is around 25k.When the sensor is bent, the conductive layer is stretched, resulting in a reduced cross section (imagine stretching a rubber band) and increased

resistance. At a 90° angle, this resistance is approximately 100K

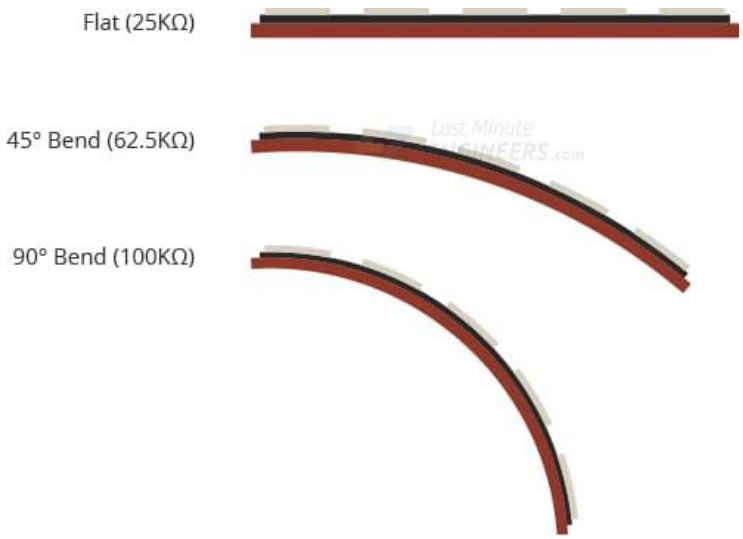



**Fig 3.13 Flex sensor value**

When the sensor is straightened out again, the resistance returns to its original value. By measuring the resistance, you can determine how much the sensor is bent

## 3.2.5 Reading from Flex Sensor

The simplest way to read the flex sensor is to combine it with a static resistor to form a voltage divider, which produces a variable voltage that can be read by the analog-to-digital converter of a microcontroller.

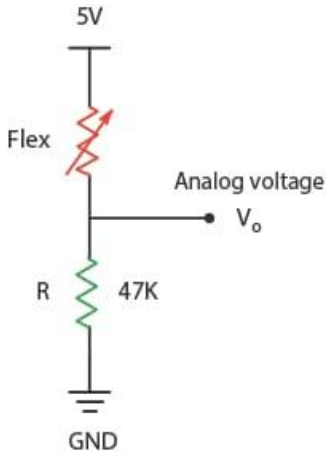



**Fig 3.14 Flex sensor voltage divider**

It is important to note that the output voltage you measure is the voltage drop

across the pull-down resistor, not the voltage drop across the flex sensor. We can use this equation to calculate the output voltage (Vo).

$$V_O = V_{CC} \frac{R}{R + R_{Flex}}$$

In this configuration, the output voltage decreases as the bend radius increases.For example, with a 5V supply and a 47K pull-down resistor, when the sensor is flat (0°), the resistance is relatively low (around 25k). This produces the following output voltage:

$$V_O = 5V \frac{47k\Omega}{47k\Omega + 25K\Omega}$$

$$= 3.26V$$

When bent to its full extent (90°), the resistance increases to approximately 100K. As a result, the output voltage becomes23

$$V_O = 5V \frac{47k\Omega}{47k\Omega + 100K\Omega}$$

$$= 1.59V$$

### 3.2.6 Specifications & Features

- Operating voltage of this sensor ranges from 0V to 5V.
- It can function on low-voltages.
- Power rating is 1 Watt for peak & 0.5Watt for continuous.
- Operating temperature ranges from -45ºC to +80ºC.

- Flat resistance is 25K Ω.
- The tolerance of resistance will be ±30%.
- The range of bend resistance will range from 45K -125K Ohms.

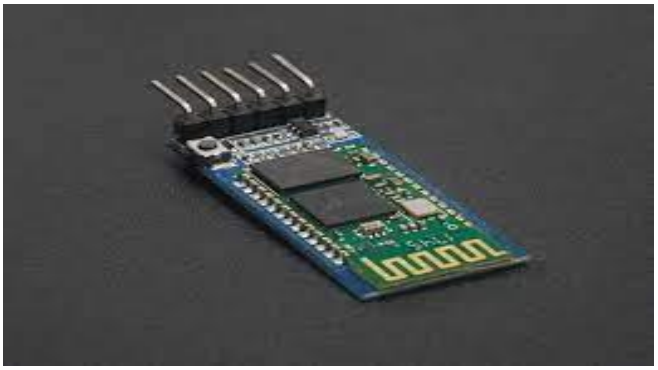
### 3.2.7 The applications of the flex-sensors:

- Medical Instruments.
- Peripherals of Computer.
- Robotics.
- Physical Therapy.
- Virtual Motion (Gaming).

## 3.3 BLUETOOTH MODULE:

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.

**Fig 3.15 Bluetooth module diagram**



### 3.3.1 HC-05 pin description:

| | | |
|---|---|---|
| 1 | Enable | 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<br>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 |

### 3.3.2 HC-05 Bluetooth Module Pinout :

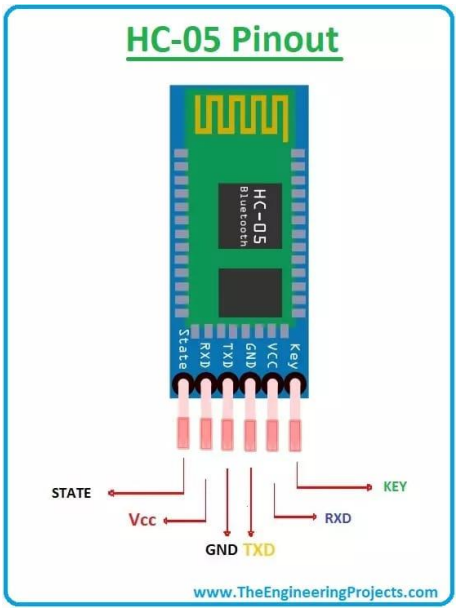



**Fig 3.16 Blutooth module pinout**

### 3.3.3 HC-05 Specifications

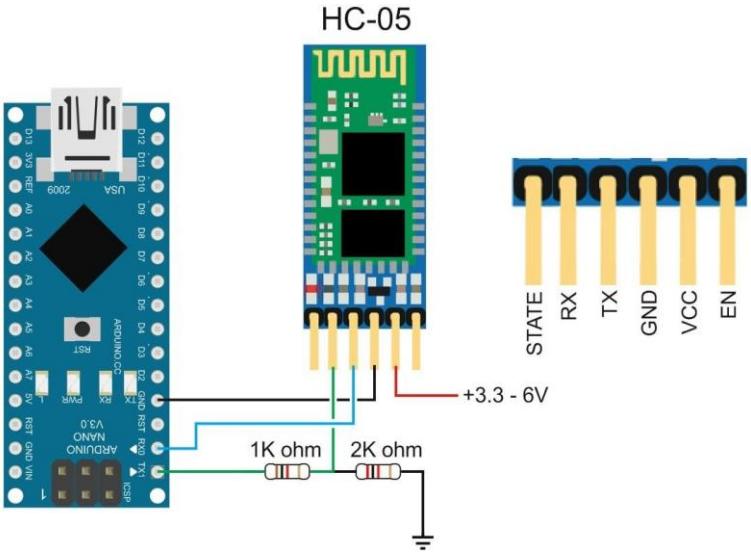



**Fig 3.17 Blutooth module connection diagram**

- 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

### 3.3.4 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

### 3.3.5 Overview of HC-05 Bluetooth module

The **HC-05** is a very cool module which can add two-way (full-duplex) wireless functionality. this module is used 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.

### 3.3.6 Working Of 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 **Fig 3.16** below.

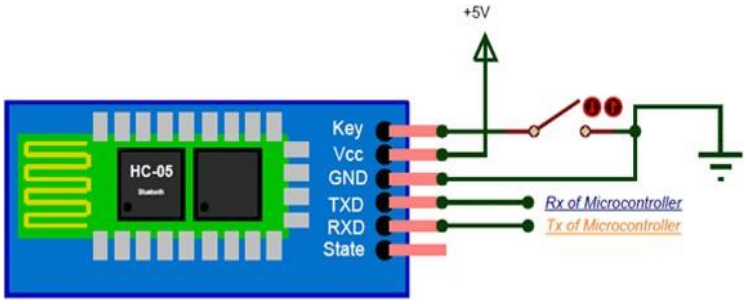



**Fig 3.18 Blutooth module working**

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.

## 3.4 CONNECTION DIAGRAM

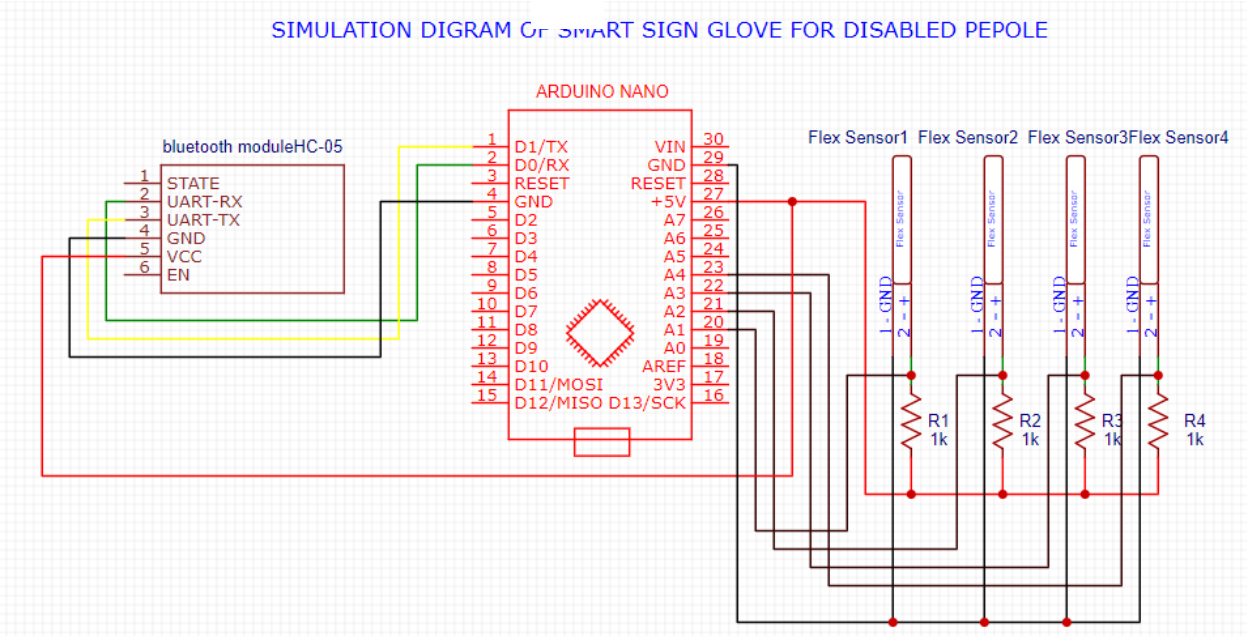



**Fig 3.19 Simulation diagram**

# CHAPTER-4

# SOFTWARE REQUIREMENT

# CHAPTER-4

# SOFTWARE REQUIREMENT

## 4.1 ARDUINO IDE INTRODUCTION:

Introduction to Arduino IDE, IDE stands for “Integrated Development Environment” it is an official software introduced by Arduino.cc, that is mainly used for editing, compiling and uploading the code in the Arduino Device. Almost all Arduino modules are compatible with this software that is an open source and is readily available to install and start compiling the code on the go. we introduce the Software, how we can install it, and make it ready for developing applications using Arduino modules.

### 4.1.1 Arduino IDE Definition:

- Arduino IDE is an open source software that is mainly used for writing and compiling the code into the Arduino Module.
- It is an official Arduino software, making code compilation too easy that even a common person with no prior technical knowledge can get their feet wet with the learning process.
- It is easily available for operating systems like MAC, Windows, Linux and runs on the Java Platform that comes with inbuilt functions and commands that play a vital role for debugging, editing and compiling the code in the environment.
- A range of Arduino modules available including Arduino Uno, Arduino Mega, Arduino Leonardo, Arduino Micro and many more.
- Each of them contains a microcontroller on the board that is actually programmed and accepts the information in the form of code.
- The main code, also known as a sketch, created on the IDE platform will ultimately generate a Hex File which is then transferred and uploaded in the controller on the board.
- The IDE environment mainly contains two basic parts: Editor and Compiler where former is used for writing the required code and later is used for compiling and uploading the code into the given Arduino Module.
- This environment supports both C and C++ languages.

**Details on IDE:** The IDE environment is mainly distributed into three sections

i. Menu Bar
ii. Text Editor
iii. Output Pane

As we download and open the IDE software, it will appear like an image below.

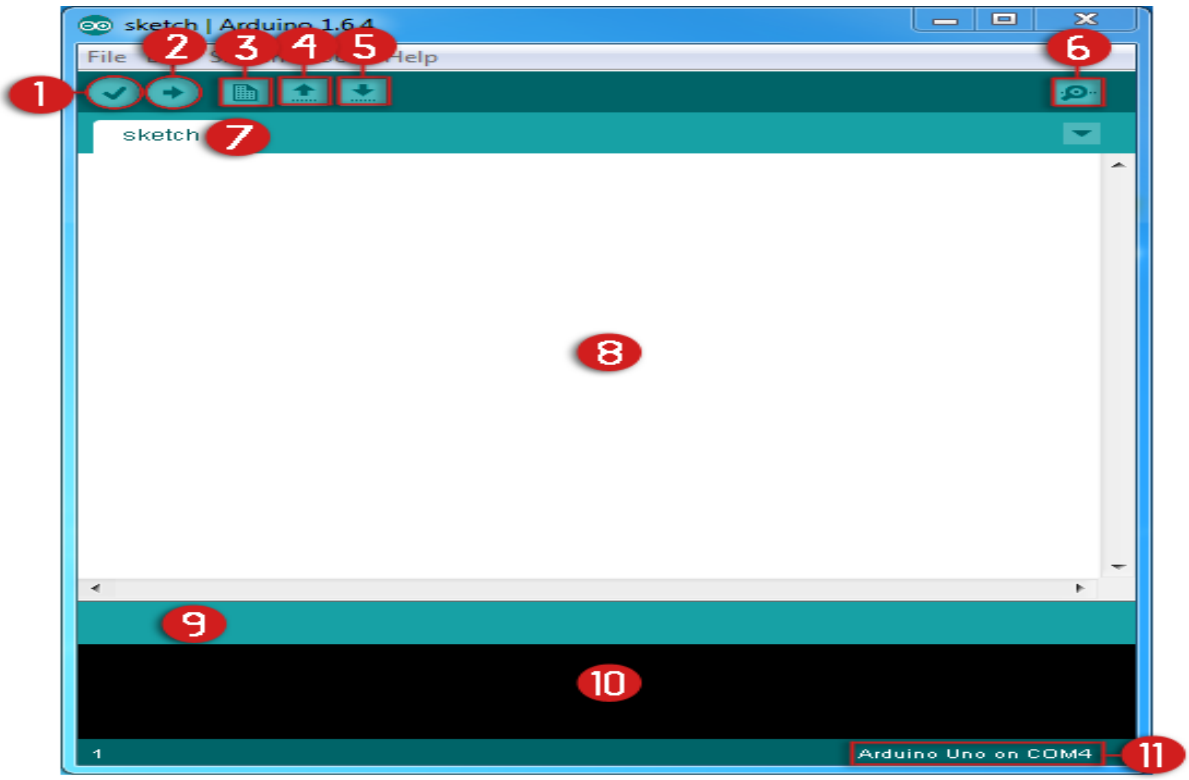
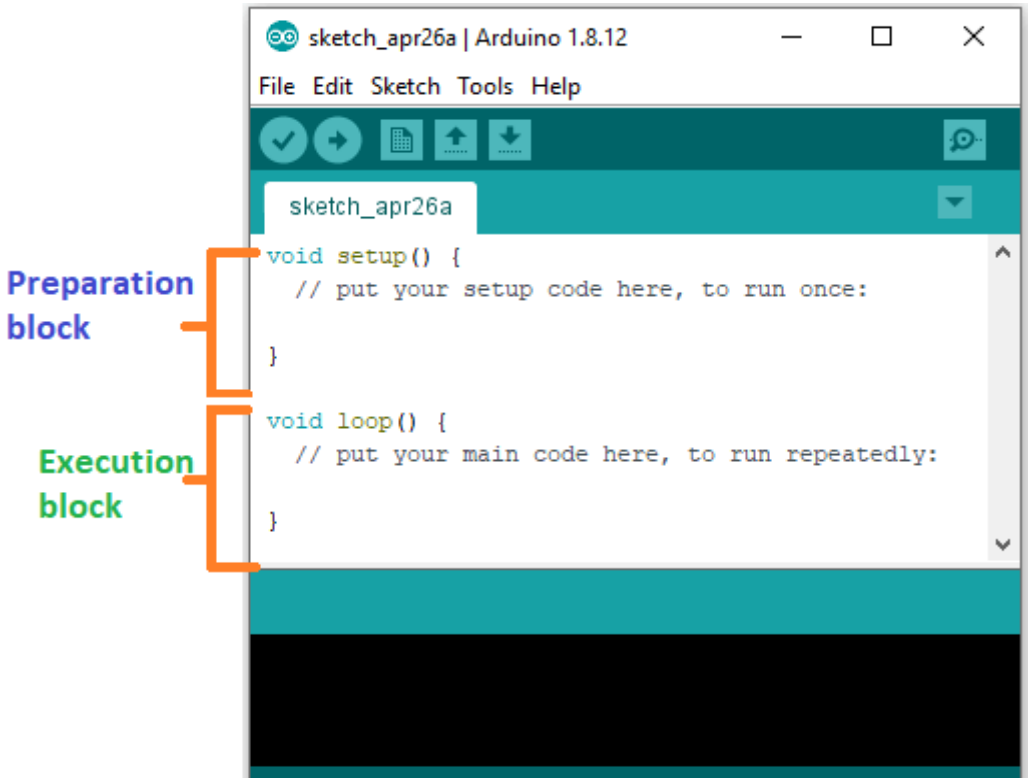



**Fig 4.1 Arduino IDE**
**Description**

iv. **Verify:** Compiles and checks your code. It will catch errors in syntax (like missing semi-colons or parenthesis).
v. **Upload:** Sends your code to the Arduino board. When you click it, you should see the lights on your board blink rapidly.
vi. **New:** This buttons opens up a new code window tab.
vii. **Open:** This button will let you open up an existing sketch.
viii. **Save:** This saves the currently active sketch.

### 4.1.2 Coding Screen:

The coding screen is divided into two blocks. The ‘setup’ is considered as the



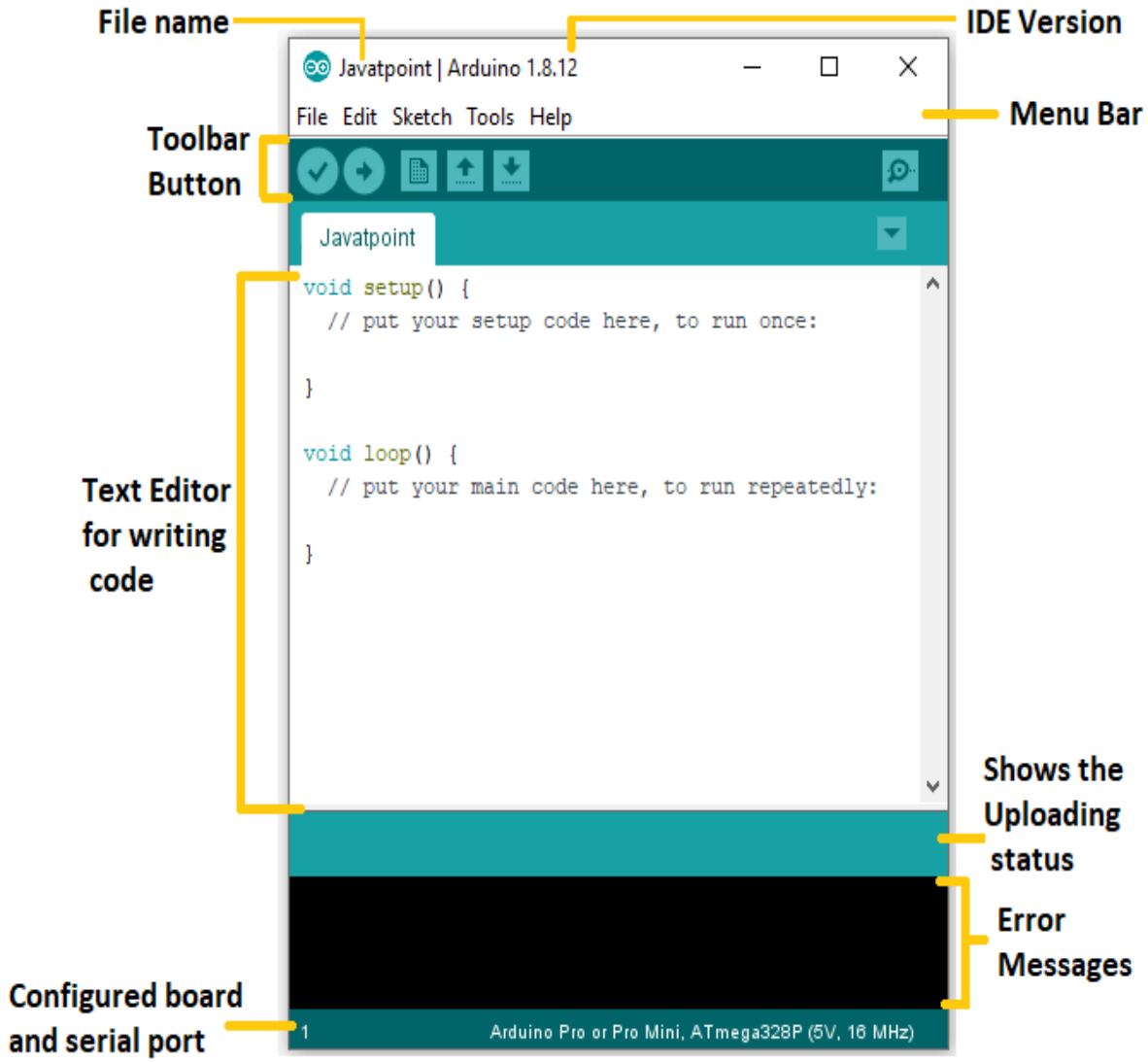

preparation

**Fig 4.2 coding screen**

block, while the ‘loop’ is considered as the execution block. It is shown **Fig 4.2** below:

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.

### 4.1.3 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.

**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 **Fig 4.3** below:

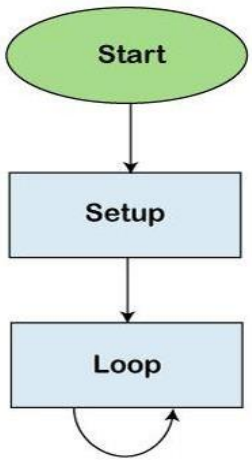



**Fig 4.3 Program Flow Chart**

## 4.1.4 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.

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.

### 4.1.5 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**.

### 4.1.6 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:

- Char Data Type
- Float Data Type
- Double Data Type
- Unsigned int Data Type
- Void Data type

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

### 4.1.7 Aruduino 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.

### 4.1.8 Aruduino 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.

### 4.1.9Arduino IF statement

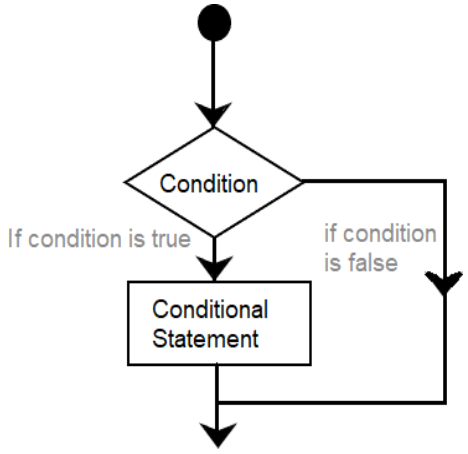



**Fig 4.4 Flow Chart Of IF Statement**

**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

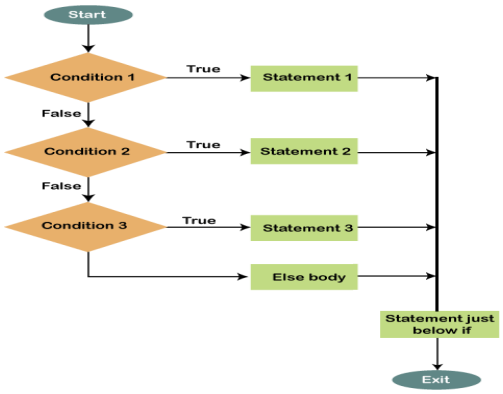



**Fig 4.5 Flow Chart Of IF-ELSE**

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.

## 4.2 TTS APPLICATION

### 4.2.1 INTRODUCTION:

An Arduino Bluetooth text-to-speech application converts written text into spoken words using an Arduino microcontroller and a text-to-speech (TTS) module or software. It typically involves sending text input to the Arduino, which then processes the text and triggers the TTS module to produce corresponding speech output. This can be useful for creating voice alerts, assistive devices, or interactive projects where verbal communication is required. The process involves programming the Arduino to interface with the TTS module and controlling the speech synthesis based on the input text.

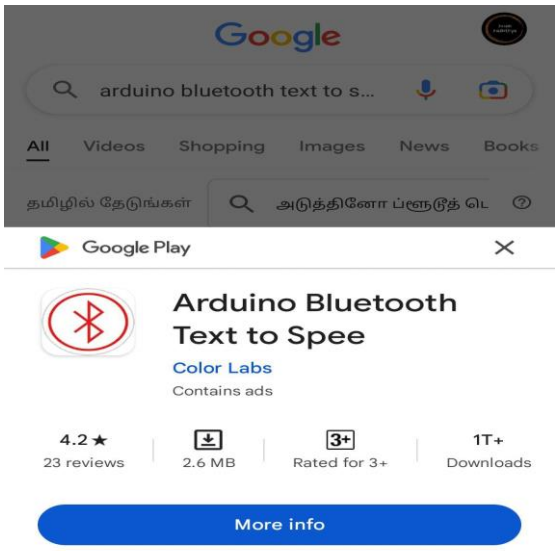



**Fig 4.6 Arduino Bluetooth Text to Speech Application**

### 4.2.2 OVERVIEW OF TTS:

An Arduino text-to-speech (TTS) application can enhance a smart sign glove by providing audio feedback alongside the visual feedback provided by the glove's display. Here's how:

Accessibility For individuals with visual impairments, the TTS functionality can audibly convey the information displayed on the glove's sign. This makes the device more inclusive and accessible to a wider range of users. Having both visual and auditory feedback increases redundancy and ensures that the information is effectively communicated, even in noisy or distracting environments where visual cues might be missed.

Enhanced Communication: The combination of visual signs and spoken words can enhance communication in scenarios where the wearer needs to convey messages to others who may not be able to see the sign clearly. Alerts and Notifications: The TTS functionality can be utilized to provide alerts, notifications, or warnings to the wearer based on predefined conditions or events detected by sensors integrated into the glove. For example, it could alert the wearer about nearby obstacles or hazards.

Interactive Applications: The TTS capability opens up possibilities for interactive applications where the glove can receive text input from external sources (such as a smartphone or other sensors) and convert it into spoken words, allowing for two-way communication or feedback.By integrating Arduino TTS functionality into a smart sign glove, you create a more versatile and effective communication tool that caters to diverse user needs and environments.

# CHAPTER-5
# ADVANTAGES AND APPLICATIONS

# CHAPTER-5

# ADVANTAGES, DISVANTAGES AND APPLICATIONS

## 5.1 ADVANTAGES:

Smart sign gloves are special gloves that can understand sign language gestures and turn them into spoken words or text:

- **Helping People to Communicate:**

  They make it easier for people who are deaf or have trouble speaking to talk with others by translating sign language into words or text.

- **Translation in Real-Time:**

  These gloves can quickly turn sign language into spoken words or text, so people who don't know sign language can understand right away.

- **Being More Independent:**

  People can communicate without needing someone else to translate for them or special devices, which helps them be more independent in their daily lives.

- **Learning Tool:**

  They can also be used to teach sign language to others, making it easier for everyone to learn and understand.

- **Making Communication Faster:**

  In places like hospitals or during emergencies, where quick and clear communication is crucial, these gloves can help speed things up.

- **Using Technology in New Ways:**

  They show how technology can be used creatively to help people with disabilities and improve their lives.

- **Adjusting to Different Needs**:

  The gloves can be set up to recognize different sign languages or gestures, and people can adjust how sensitive they are to gestures.

- **Connecting with Other Devices:**

  They can work with smartphones, tablets, or other gadgets, making them even more useful in different situations.

- **Driving Innovation:**

  By developing and using these gloves, researchers are pushing the boundaries of technology and helping it advance even further.

- **Building Inclusive Communities:**

  Ultimately, these gloves help bring people together by breaking down communication barriers and making sure everyone can participate equally in conversations and activities.

## 5.2 APPLICATIONS:

- **Communication:**

  These gloves enable individuals who are deaf or have speech impairments to communicate effectively by translating their sign language into text or speech.

- **Independence:**

  They promote independence by allowing users to express themselves without relying on interpreters or other communication aids.

- **Education:**

  Smart sign gloves can facilitate learning for individuals with disabilities by providing real-time translation of sign language during lectures, presentations, or classroom activities.

- **Employment:**

  They can enhance employment opportunities by enabling deaf or speech-impaired individuals to communicate with coworkers, customers, and employers in various work environments.

- **Social Inclusion:**

By bridging the communication gap, smart sign gloves help disabled individuals participate more fully in social interactions, activities, and events.

- **Accessibility:**

These gloves promote accessibility by making public spaces, services, and information more easily accessible to individuals with disabilities.

- **Safety:**

They can improve safety by allowing users to communicate effectively in emergency situations or when seeking assistance

# CHAPTER-6

# CONCLUSION

# CHAPTER-6

## 6.1 CONCLUSION

In conclusion, the Smart Sign Glove leveraging Arduino Nano technology represents a promising advancement in assistive technology for individuals with disabilities, particularly those who rely on sign language as their primary means of communication. Despite some challenges and limitations, the Smart Sign Glove offers significant benefits and opportunities for enhancing accessibility, independence, and social inclusion for users.

Through its compact design, intuitive functionality, and wireless connectivity via Bluetooth module HC-05, the Smart Sign Glove provides users with a portable and versatile communication tool. By translating sign language gestures into text or speech and facilitating real-time interaction with external devices, the glove empowers users to express themselves effectively in a variety of social, educational, and professional settings.

Moreover, the Arduino Nano platform offers flexibility, customization, and affordability, allowing for the adaptation of the Smart Sign Glove to meet the diverse needs and preferences of users. While there may be challenges such as learning curves, technical issues, and socioeconomic barriers to adoption, these can be addressed through user-centered design, training, support, and ongoing development efforts.

Overall, the Smart Sign Glove holds tremendous potential to break down communication barriers, promote inclusivity, and enhance quality of life for individuals with disabilities. By leveraging technology to bridge the gap between sign language and spoken or written language, the glove fosters greater autonomy, independence, and participation in society. As advancements continue and accessibility improves, the Smart Sign Glove will undoubtedly play a vital role in empowering individuals with disabilities to communicate, connect, and thrive in a more inclusive world.

## 6.2 REFERENCE:


I. 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.

II. 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.

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

IV. 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.

V. 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.

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

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

VIII. 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.

IX. 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.

X. 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.

XI. 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.

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

XIII. 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.

XIV. 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.

XV. 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

**APPENDIX**

## I) PROGRAM:

```
// Include libraries
#include <SoftwareSerial.h>

// Define Bluetooth module pins
#define BT_TX 10
#define BT_RX 11

// Define flex sensor pins
#define FLEX_PIN_1 A0
#define FLEX_PIN_2 A1
#define FLEX_PIN_3 A2
#define FLEX_PIN_4 A3

// Define threshold value
#define THRESHOLD 850
#define THRESHOLD0 780
#define THRESHOLD1 800
#define THRESHOLD2 959

// Initialize SoftwareSerial object for Bluetooth communication
SoftwareSerial BTSerial(BT_TX, BT_RX);

void setup() {
  // Initialize serial communication for debugging
  Serial.begin(9600);
  // Initialize Bluetooth communication
  BTSerial.begin(9600);
}
```

```
void loop() {
  // Read flex sensor values
  int flexValue1 = analogRead(FLEX_PIN_1);
  int flexValue2 = analogRead(FLEX_PIN_2);
  int flexValue3 = analogRead(FLEX_PIN_3);
  int flexValue4 = analogRead(FLEX_PIN_4);

  // Convert flex sensor values to binary
  int flexBinary = 0;
  if (flexValue1 < THRESHOLD) {
    flexBinary |= 1 << 3;
  }
  if (flexValue2 < THRESHOLD0) {
    flexBinary |= 1 << 2;
  }
  if (flexValue3 < THRESHOLD1) {
    flexBinary |= 1 << 1;
  }
   if (flexValue4 > THRESHOLD2) {
    flexBinary |= 1 << 0;
  }

  // Convert binary to corresponding text
  String gesture;
  switch (flexBinary) {
    case 0b0000:
  gesture = "i need water";
      break;
    case 0b0001:
      gesture = "how are my dear friend";
      break;
    case 0b0010:
```

```
    gesture = "ok";
    break;
  case 0b0011:
    gesture = "yes";
    break;
  case 0b0100:
    gesture = "i have seen this one";
    break;
  case 0b0101:
    gesture = "help me";
    break;
  case 0b0110:
    gesture = "no";
    break;
  case 0b0111:
    gesture = "hi ranjith";
    break;
     case 0b1000:
    gesture = "hi mom";
    break;
      case 0b1001:
    gesture = "excellent";
    break;
      case 0b1010:
    gesture = "good";
break;
      case 0b1011:
    gesture = "good morning";
    break;
      case 0b1100:
    gesture = "goodnight";
    break;
```

```
      case 0b1101:
    gesture = "I want to go restroom";
    break;
      case 0b1110:
    gesture = "I need food";
    break;
      case 0b1111:
    gesture = "interstellor";
    break;

  default:
    gesture = "Unknown gesture";
 }

 // Print gesture to serial monitor
 Serial.println(gesture);

 // Send gesture via Bluetooth
 BTSerial.println(gesture);

 // Add delay to prevent flooding
 delay(2000);
}
```

## II) COST ESTIMATION:

| S.N0 | PARTS | QTY | AMOUNT(Rs) |
|---|---|---|---|
| 1 | Arudiuno nano | 1 | RS. 300 |
| 2 | Flex sensor | 4 | RS. 1700 |
| 3 | Bluetooth Module | 1 | RS. 300 |
| 4 | Glove | 1 | RS. 240 |
| 5 | 5V power supply | 1 | RS. 100 |
| 6 | USB Cable | 1 | RS. 80 |
| 7 | Resistor | 5 | RS.20 |
| 8 | Jumper wires | As Req | RS. 70 |
| 9 | Speaker | 1 | RS. 50 |
| 10 | Zero PCB board | 2 | RS. 50 |
| 11 | TOTAL | | RS. 2910 |

## III)PHOTOGRAPHS :