GUESTURE VOX: ENHANCING COMMUNICATION FOR INDIVIDUALS WITH SPEECH DISABILITIES

A PROJECT REPORT

Submitted by

ABHIMANYU P 2214201

in partial fulfillment for the requirements for the degree

Of

BACHELOR OF ENGINEERING AND TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

ALAGAPPA CHETTIAR GOVERNMENT COLLEGE OF ENGINEERING AND TECHNOLOGY, KARAIKUDI-630003

(A Government Autonomous Institution Affiliated to Anna University)

NOVEMBER 2024

GUESTURE VOX: ENHANCING COMMUNICATION FOR INDIVIDUALS WITH SPEECH DISABILITIES

A PROJECT REPORT

Submitted by

ABHIMANYU P 2214201

in partial fulfillment for the requirements for the degree

Of

BACHELOR OF ENGINEERING AND TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

ALAGAPPA CHETTIAR GOVERNMENT COLLEGE OF ENGINEERING AND TECHNOLOGY, KARAIKUDI-630003

(A Government Autonomous Institution Affiliated to Anna University)

NOVEMBER 2024

BONAFIDE CERTIFICATE

Certified that the project report titled **"GESTURE VOX: AN ASSISTIVE COMMUNICATION DEVICE FOR PEOPLE WITH SPEECH IMPAIRMENTS"** is the Bonafide work of **ABHIMANYU P (2214201)** who carried out the project work under my supervision. Certified further that to the best of my knowledge, the work reported herein does not form part of any other thesis or dissertation based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

Signature of the HOD
Dr. A. SIVANANTHA RAJA M.E., Ph.D.,
Professor and Head of the Department,
Department of Electronics and
Communication Engineering,
A.C. Government College of
Engineering and Technology,
Karaikudi – 630003.

Signature of the Supervisor
Dr. G. KARPAGARAJESH M.E., Ph.D.,
Associate Professor,
Department of Electronics and
Communication Engineering,
A.C. Government College of
Engineering and Technology,
Karaikudi – 630003.

Submitted for Viva-Voce Examination held at A.C. Government College of Engineering and Technology, Karaikudi-3 on 27.11.2024

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

Speech impairments create significant challenges in daily life, impacting communication and self-expression. The "Speaking Hands: Gesture Vox for Individuals with Speech Impairments" project introduces an assistive device designed to bridge the communication gap for non-verbal individuals. This innovative system detects finger movements through a glove with embedded flex sensors, converting gestures into voice output. Each unique gesture triggers a pre-recorded audio message stored on an SD card, allowing users to convey essential phrases through simple hand signs. With personalized outputs and efficient, gesture-based interaction, this device empowers non-verbal users to communicate, fostering inclusivity and self-reliance. This project represents a step towards a world where speech impairments no longer hinder communication or independence.

ACKNOWLEDGEMENT

We are very thankful to **Dr. K. BASKARAN M.E., Ph.D**. Principal, Alagappa Chettiar Government College of Engineering and Technology, Karaikudi for providing us an excellent infrastructure and studying environment in the institution to do this project.

We sincerely thank **Dr. A. SIVANANTHA RAJA M.E., Ph.D.,** Head of the Department, Electronics and communication Engineering for encouragement, support, and valuable guidance in this project.

It is our privilege to work under the guidance of Associate Professor **Dr. G. KARPAGARAJESH M.E., Ph.d.,** Department of Electronics and Communication Engineering for encouraging and valuable guidance throughout this project in all possible ways.

Our sincere thanks to our faculty advisor **Mrs. X. MERLIN SHEEBA M.E.,** Department of Electronics and Communication Engineering for guidance and motivation throughout this project.

We would like to thank the staff members of the ECE department for their help and assistance during the entire course of our project. Our heartfelt thanks to all who helped us to complete the project successfully.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Communication is an essential human experience, forming the bedrock of connection, self-expression, and independence. Gestures and spoken words allow us to share our thoughts, emotions, and needs with those around us. However, for individuals with speech impairments, verbal communication is often limited, leading to challenges in expressing themselves and forming connections.

The "Speaking Hands: Gesture Vox for Individuals with Speech Impairments" project emerges as a transformative solution, rooted in the vision of a more accessible world where non-verbal individuals are empowered to communicate through intuitive technology. This project was born from an understanding of the isolation often felt by those without a voice, seeking to bridge the gap by translating gestures into audible words.

At its heart, this project harnesses technology’s power to address the unique challenges that speech-impaired individuals face daily. It moves beyond mere words, capturing the essence of meaningful communication by allowing users to convey essential messages with ease and independence. By transforming hand signs into audio output, it helps bridge the communication barrier, empowering users to interact with others without requiring intermediaries or reliance on written notes.

As gestures are translated into words, the system doesn't just enable speech; it facilitates personal expression and autonomy. Each gesture-voice pairing provides an accessible means for non-verbal users to convey basic phrases, such as greetings or needs, in a way that feels natural and fluid.

CHAPTER 2

LITERATURE REVIEW

NAMES OF RESEARCH PAPER	REFERENCE
Hand Gesture Recognition Using Flex Sensors and Microcontroller	https://doi.org/10.1109/ISCAIE61308.2024.10576251 **Publisher:** IEEE **Published on :** 23 April 2020
Glove-Based Assistive Device for the Speech and Hearing Impaired	https://doi.org/10.1109/HNICEM48295.2019.9072695 **Publisher:** IEEE **Published on :** 10 May 2021
Embedded Glove System for Speech-Impaired People	https://doi.org/10.1109/ICACDOT.2016.7877706 **Publisher:** IEEE **Published on :** 16 March 2017
Analysis of Flex Sensor Technology for Wearable Systems	https://doi.org/10.1109/ICEEICT61591.2024.10718430 **Publisher:** IEEE **Published on :** 26 July 2024
Audio Playback on Microcontrollers for Real-Time Applications	https://doi.org/10.1109/ICIPCN63822.2024.00115 **Publisher:** IEEE **Published on :** 09 September 2024

CHAPTER 3

HARDWARE DESCRIPTION

3.1 Arduino Uno Microcontroller

The Arduino Uno is a microcontroller board developed by Arduino.cc, known for its simplicity and versatility in various applications.

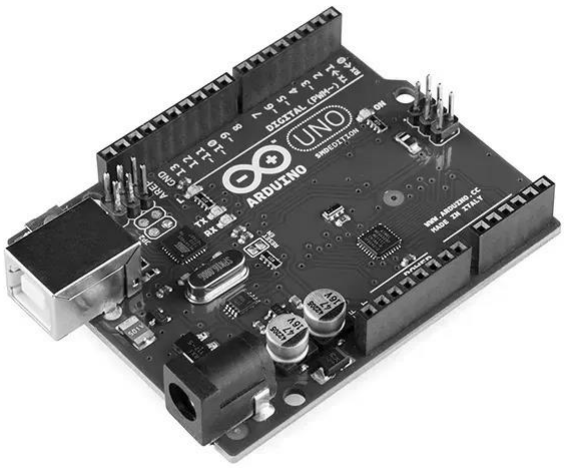

Figure 3.1 Arduino uno

3.1.1 Hardware Overview of the Arduino Uno:

Microcontroller Unit (MCU):

The Uno is powered by the ATmega328P, an 8-bit AVR microcontroller. This single-core setup is ideal for straightforward tasks and basic real-time operations.

Clock Speed:

The ATmega328P operates at a clock speed of 16 MHz, offering performance for basic embedded applications.

Memory:

The Arduino Uno includes 2KB of SRAM, with 32KB of Flash memory for program storage. This allows space for both program code and variable data.

GPIO Pins:

The Uno has 14 digital I/O pins (6 can be used as PWM outputs) and 6 analog input pins, configurable for digital or analog signals, PWM, and other functions.

Analog-to-Digital Converters (ADC):

The Uno has a 10-bit ADC on its 6 analog pins, allowing for analog measurements with a 5V reference voltage.

PWM:

The Arduino Uno provides 6 PWM channels, suitable for dimming LEDs, controlling servos, and other analog signal generation.

I2C, SPI, UART:

The board supports I2C, SPI, and UART protocols, allowing connectivity with external sensors, displays, and communication modules.

USB:

The Uno includes a USB 2.0 port for programming and power supply, also allowing it to act as a USB device for serial communication with a computer.

Low Power Features:

The ATmega328P provides low-power modes, making it suitable for battery-powered and energy-efficient applications.

Peripherals:

It includes a range of peripherals, such as timers, a watchdog timer, and external interrupts, which support various embedded tasks.

Power Supply:

The Uno operates on a 5V supply voltage, but it can be powered through a barrel jack input of 7-12V or via USB.

Form Factor:

The Arduino Uno is designed as a compact board but is compatible with a wide variety of Arduino shields for added functionality.

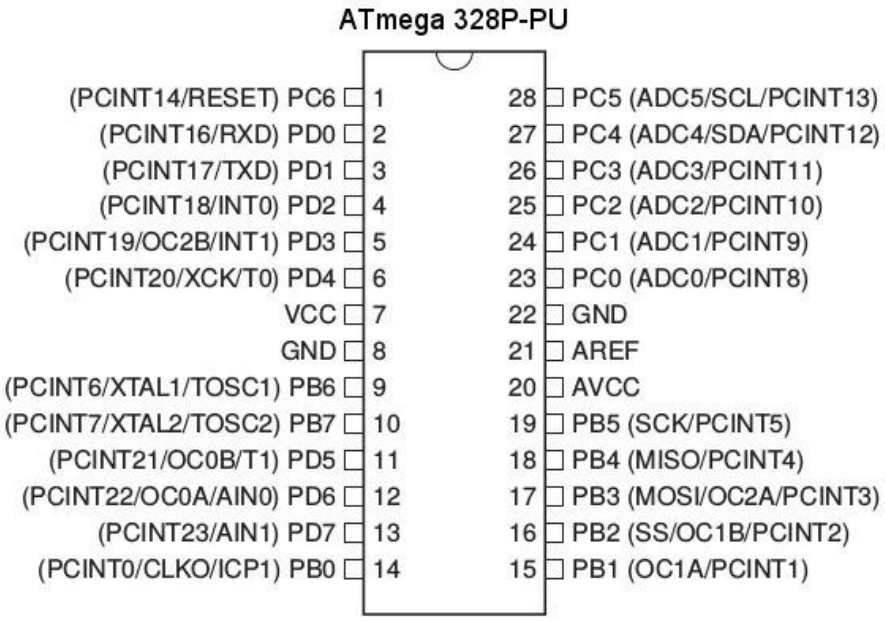

Figure 3.2 Arduino Uno Board Pin Configuration

Development Tools:

Arduino provides an easy-to-use IDE and extensive library support in C/C++, allowing for straightforward programming and debugging. Its robust community and support make it accessible for beginners and professionals alike.

3.1.2 Applications:

IoT Devices: The Uno is commonly used in IoT applications, collecting data and controlling connected devices through simple wireless modules.

Robotics: It can drive motors, read sensors, and manage basic robot systems, often used in DIY and educational robotics projects.

Data Logging: With its SD card add-ons, the Uno is useful for recording environmental data or sensor readings over time.

Automation: It is widely used in home automation systems for controlling lighting, managing appliances, and security features.

Education: As a staple in learning environments, the Uno helps teach microcontroller programming, electronics, and embedded systems basics.

Wearable Devices: Although it has limited power features, the Uno can still power small wearable devices, useful for prototyping health monitors and interactive wearables.

DIY Projects: Arduino Uno is popular in the maker community for projects like simple games, home weather stations, and remote-controlled devices.

Industrial Control: With robust connectivity options, it can be integrated into small industrial control systems for automation tasks.

Audio and Sound Processing: While limited, it can handle simple audio tasks, such as sound effects or basic audio frequency recognition projects.

Assistive Technology: The Uno is commonly used in assistive tech, such as creating affordable communication aids or alerting systems for the visually or hearing impaired.

3.2 FLEX SENSOR

A flex sensor is a thin, bendable component commonly used in electronics projects to detect bending or flexing, often enabling gesture recognition in wearable and assistive devices. It provides an easy way to translate physical movements into variable resistance that microcontrollers can read and process.

Figure 3.3 Flex Sensor

3.2.1 Hardware Overview of Flex Sensor

Working Principle:

A flex sensor is a passive component that changes its resistance based on bending or flexing. As it bends, the resistance increases, allowing measurement of the bending angle.

Structure:

Flex sensors are typically made of a thin, flexible plastic strip with conductive elements inside. The resistance change is caused by the deformation of the conductive material as the sensor bends.

Size and Form Factor:

Commonly available in lengths of 2.2 inches or longer, flex sensors are thin, lightweight, and can easily be attached to gloves or other flexible surfaces. Their slim form factor makes them suitable for wearable applications, where they can detect hand, finger, or arm movements.

Resistance Range:

Flex sensors have a baseline resistance (usually around 10kΩ when straight), which increases as the sensor bends. The resistance range can vary depending on the specific sensor, with resistance values often doubling when bent to 90 degrees. This resistance range is used in programming to detect different levels of bending.

Voltage and Power Requirements:

Flex sensors usually operate at 3.3V or 5V, consuming minimal power, making them suitable for battery-operated devices.

Sensitivity and Accuracy:

The sensitivity of a flex sensor is determined by its resistance change per degree of bend. While accurate enough for general gesture detection, flex sensors may have a lower resolution for small or subtle movements, especially at low bending angles.

Response Time:

Flex sensors have a quick response time, which makes them ideal for real-time applications such as gesture recognition. However, they may exhibit minor lag or hysteresis, especially if repeatedly bent in quick succession.

Durability:

Flex sensors are generally durable and designed to withstand repeated bending. However, over time and with excessive flexing, they may experience wear and potential degradation in accuracy.

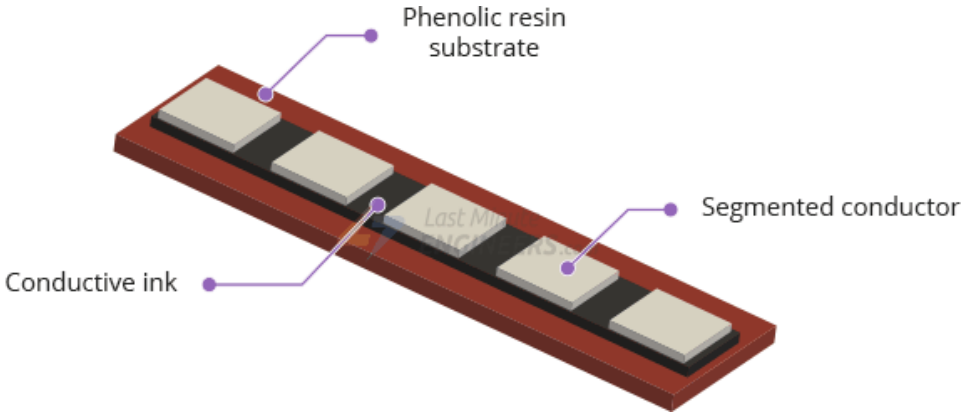

Figure 3.4 Flex sensor Cross Section

3.2.2 Applications:

Flex sensors are commonly used in applications such as wearable technology, robotic controls, gesture recognition systems, gaming controllers, and assistive devices for people with disabilities.

Integration with Microcontroller:

Flex sensors connect easily to microcontrollers like Arduino using an analog input. The sensor's resistance change is read as a voltage variation, which can then be processed in the Arduino code to identify specific gestures based on predefined thresholds.

3.3 MICRO-SD CARD MODULE

A microSD card module is an electronic component that enables microcontrollers to read from and write to microSD cards, providing external storage for data logging, multimedia retrieval, or file management in embedded systems. These modules are widely used in projects where additional memory is needed for storing audio, sensor data, or configuration files, offering a compact and efficient solution for expanding data capabilities in microcontroller applications.

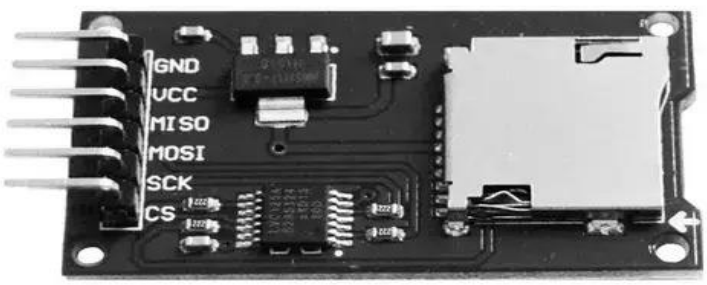

Figure 3.5 Module of MicroSD Card

3.3.1 Hardware Overview of microSD Card Module

The microSD card module enables microcontrollers to store and retrieve data on a microSD card, making it a crucial component for projects that require file storage, data logging, or media retrieval, such as audio playback. Its compact design includes a microSD card slot on a breakout board, with convenient pins for easy integration into embedded systems with limited space.

Operating within a voltage range of 3.3V to 5V, the microSD card module is compatible with common microcontrollers, including Arduino and ESP32. Many modules feature a built-in voltage regulator to provide stable operation within this range. Communication with the microcontroller is achieved through the SPI (Serial Peripheral Interface) protocol, utilizing pins for VCC (power), GND (ground), MOSI (Master Out Slave In), MISO (Master In Slave Out), SCK (Serial Clock), and CS (Chip Select).

3.3.2 Applications

The microSD card module is widely used in audio playback systems, where it stores pre-recorded sounds or music files that a microcontroller retrieves and plays. In data logging devices, it efficiently records information over time, such as environmental sensor readings in IoT applications. Its small form factor makes it ideal for portable electronics, storing multimedia files such as images and videos, and expanding the limited internal storage of microcontroller-based projects.

3.4 MICRO SD CARD

3.4.1 Hardware Overview of Micro SD Card

Core Component:

The core component of a micro SD card is the flash memory chip, which stores data digitally. This memory is non-volatile, meaning it retains information even when power is disconnected. The card also includes a controller that manages data storage and retrieval operations between the micro SD card and the device it interfaces with.

Form Factor:

Micro SD cards are small, typically measuring 15mm x 11mm x 1mm, making them one of the most compact forms of removable storage. They have a rectangular shape with contacts on one side for communication with devices. The card is designed to be inserted into compatible card slots or adapters.

Operating Voltage:

Micro SD cards generally operate at 3.3V, although they may also be designed to tolerate a range of voltages. Some cards are backward-compatible with 5V systems through specific adapters or voltage regulators.

Connection Interface:

Micro SD cards use a 9-pin connection interface that includes power, ground, and data transfer pins. These pins include data input (DI), data output (DO), clock (CLK), chip select (CS), and voltage pins for power and ground.

Control Mechanism:

Data transfer is controlled by the micro SD card's internal controller, which manages reading and writing operations. Communication is often conducted through protocols like SPI (Serial Peripheral Interface) or SDIO (Secure Digital Input Output), which provide efficient ways to transfer large volumes of data.

Figure 3.6 Micro SD Card

3.4.2 Applications:

Micro SD cards are used in mobile phones for storing media files, apps, and data. In wearable devices, they store health-related data, logs, and configurations. Cameras use them to store photos, videos, and media files. In IoT devices, they are used for data logging and storing configurations or firmware. Micro SD cards are also widely used in embedded systems to store firmware, program code, and system data. In gaming devices, they store game data, downloadable content, and apps.

3.5 LOUD SPEAKER

3.5.1 Hardware Overview of Loudspeaker Module

The loudspeaker module is an audio output device that converts electrical signals into sound, making it essential for projects requiring audible feedback, voice output, or music playback. This module is commonly integrated with microcontroller systems to deliver sound output from pre-recorded files or synthesized audio, providing clear, amplified sound for various embedded applications.

Operating typically within a voltage range of 3.3V to 5V, the loudspeaker module is compatible with microcontroller platforms such as Arduino and ESP32. It often consists of a small speaker driver housed within a protective enclosure, which amplifies and projects sound for better audio clarity. Some modules may include additional circuitry, such as audio amplifiers, to enhance output volume and ensure optimal sound quality.

The loudspeaker module usually connects to a microcontroller through a digital or analog pin, enabling the playback of stored audio files (e.g., .wav format) or generated tones. Audio control can be achieved using PWM (Pulse Width Modulation) signals or DAC (Digital-to-Analog Converter) outputs, depending on the microcontroller and project requirements.

3.5.2 Applications

Loudspeaker modules are widely used in voice-assistive devices, such as text-to-speech or gesture-to-voice systems, where they play pre-recorded audio in response to specific inputs. In IoT systems, they are often employed to signal alerts or provide status notifications.

CHAPTER 4
SOFTWARE DESCRIPTION

4.1 Arduino IDE Tool

The Arduino IDE (Integrated Development Environment) is an open-source software tool widely used for programming Arduino boards and compatible microcontrollers. It provides a user-friendly interface for writing, compiling, and uploading code to a microcontroller, making it an essential tool in embedded systems development.

The Arduino IDE includes a range of built-in libraries and a community-driven library repository, which simplifies the process of adding functionality to projects. Libraries for components like flex sensors, SD card modules, and OLED displays allow for easy integration without extensive low-level coding.

4.2 Audio Mapping

In projects involving audio playback on microcontrollers, audio file settings are crucial to achieving high-quality sound output with limited hardware resources. For this application, MP3 files are converted to WAV format, using specific parameters that enhance playback quality on microcontrollers. Below are the primary settings used and an explanation of their importance to the project.

Conversion Tool: Audio Online Convert

The website [Audio Online Convert] (https://audio.online-convert.com/convert-to-wav) is utilized for MP3-to-WAV file conversion, chosen for its flexibility and ease of use.

Sample Rate: 16,000 Hz

The sample rate, set to 16,000 Hz, defines the number of samples taken per second. This frequency was chosen because it balances sound clarity and data size, offering a good quality audio representation while minimizing memory usage. For voice and simple sound playback, 16,000 Hz is sufficient to capture clear audio without requiring the higher rates typically used, thus reducing data size and processing demands on the microcontroller.

Bit Resolution: 16-bit

Setting the bit resolution to 16-bit is critical for delivering a range of audio dynamics and ensuring accurate sound reproduction. Higher bit depth provides a smoother audio experience by allowing finer sound gradations, which is important for distinguishing subtle audio cues. Using a 16-bit depth is a standard choice for many embedded systems, as it provides good quality without excessively large file sizes, making it suitable for microcontroller-based projects where storage is limited.

Audio Channel: Mono

The audio is converted to a mono channel rather than stereo to conserve memory and processing power, as mono only requires one channel of audio data. This is particularly beneficial in microcontroller projects, where resources are limited, and stereo sound is not necessary. Mono audio reduces

the file size, making it easier to store and process without compromising the clarity of voice playback.

PCM Format: U8

The PCM format chosen is U8 (unsigned 8-bit PCM), which is a common format for embedded applications. This format is beneficial because it is simple to process, with each sample represented as an 8-bit unsigned integer. U8 PCM allows for relatively low data rates and is compatible with microcontroller hardware that may not support more complex audio formats. This setting ensures that the audio files can be played back with minimal strain on the microcontroller's processing capabilities.

CHAPTER 5
BLOCK DIAGRAM DESCRIPTION OF GESTRUE VOX

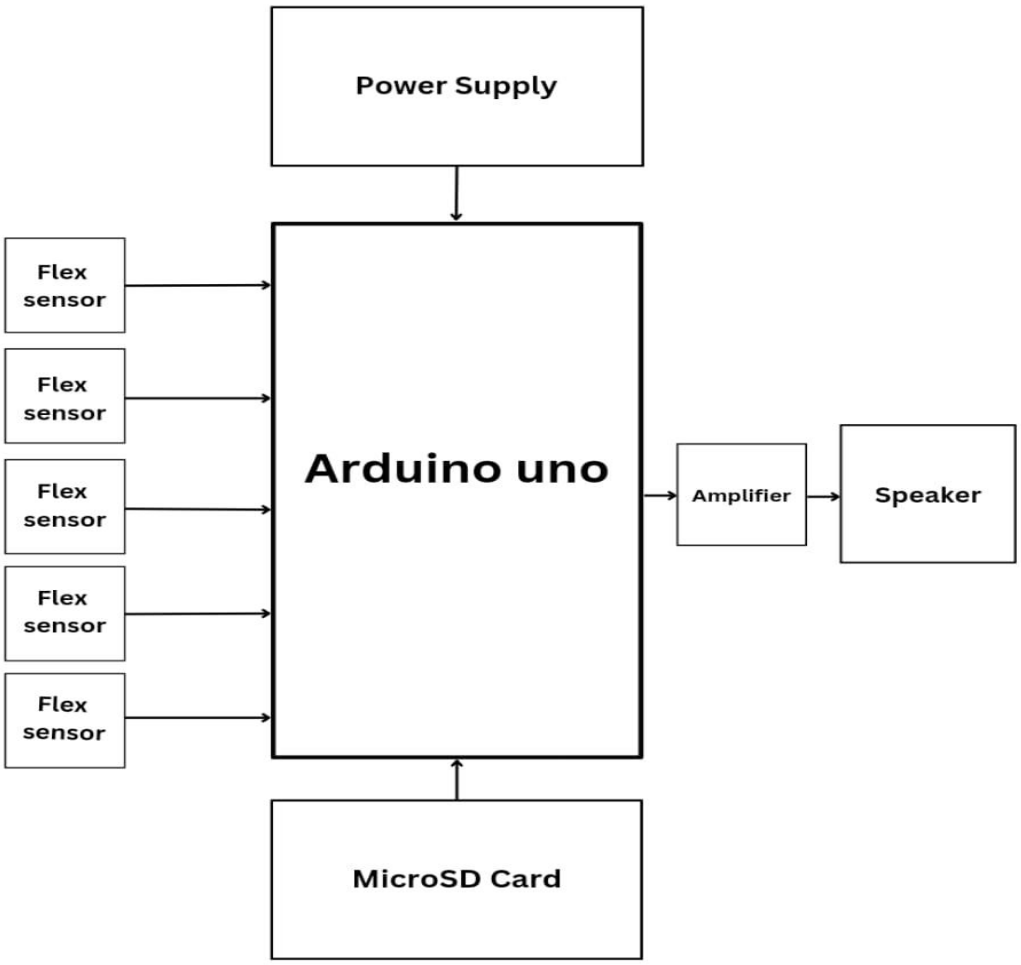

Figure 5.1 Block Diagram of Gesture Vox

- First, flex sensors placed on each finger of the glove detect finger bending, translating each finger's position into an analog voltage signal that is sent to the Arduino Uno.

- Second, the Arduino Uno reads the analog values from the flex sensors and converts them into a binary code. Each finger's bent or straightened position corresponds to a binary state, forming a 5-bit binary code based on the gesture made.

- Third, based on the unique binary code generated by each gesture, the Arduino Uno references pre-stored audio files in the SD card.

- Finally, the selected audio file is played through the loudspeaker, allowing the user to communicate pre-defined phrases via synthesized voice outputs.

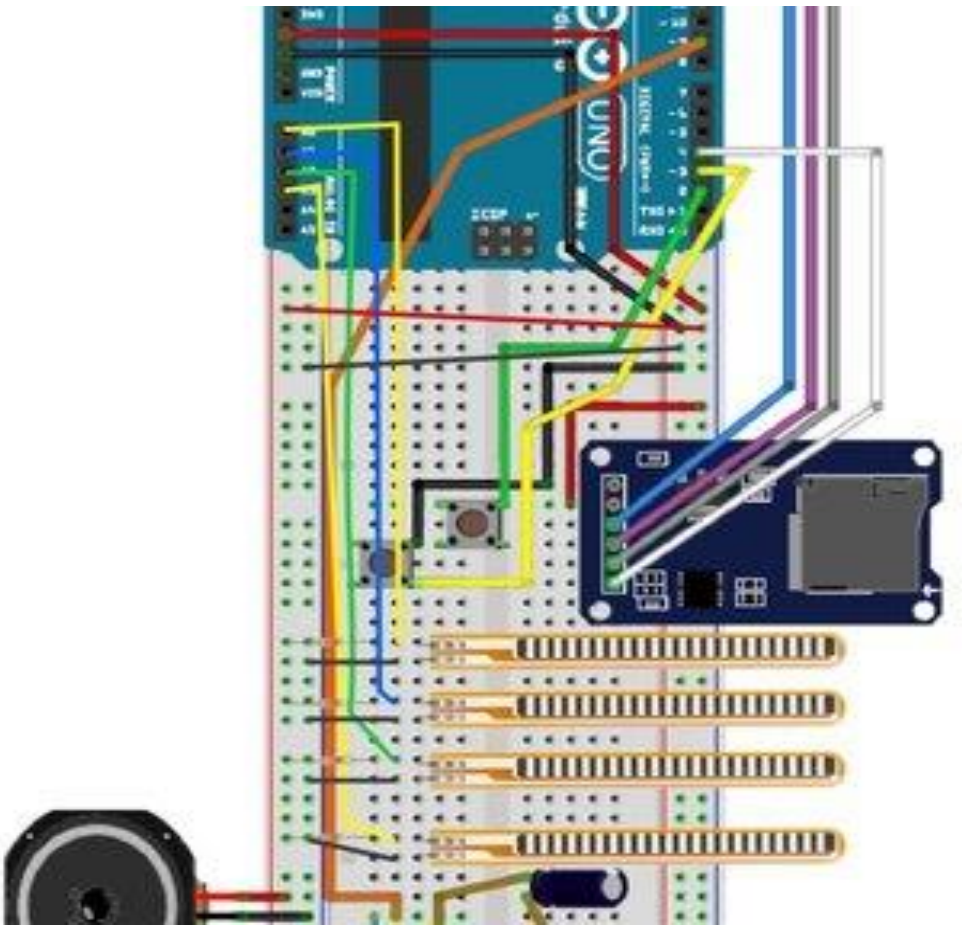

Figure 5.2 Circuit Diagram of Gesture Vox

CHAPTER 6

FLOWCHART OF GESTURE VOX

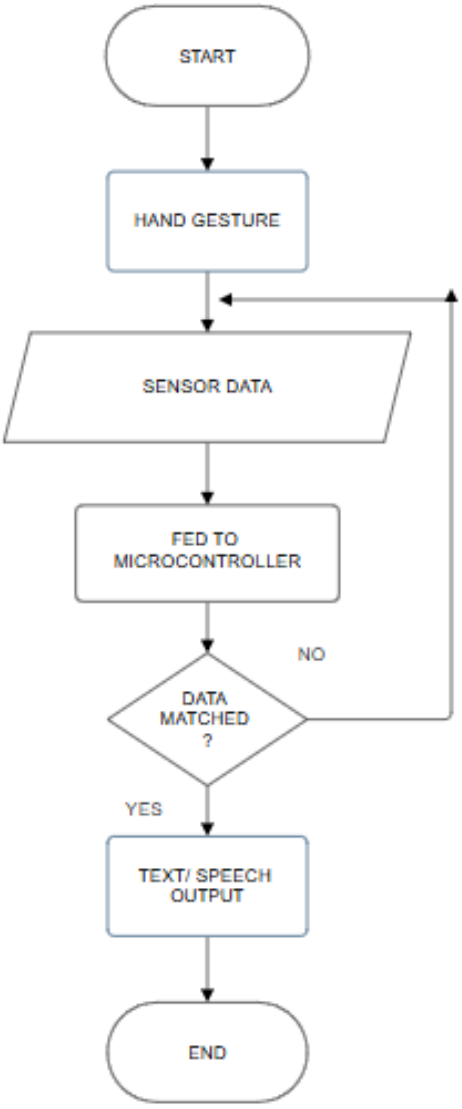

Figure 6.1 Flow Chart of Gesture Vox

CHAPTER 7

WORKING OF GESTURE VOX

Step 1: Data Collection from Flex Sensors

The project begins by gathering data from the flex sensors attached to each finger of the hand glove. These sensors are analog devices, and as the fingers bend, their resistance changes, which alters the analog voltage value. Each flex sensor is connected to one of the analog input pins of the Arduino Uno, which continuously reads these analog values. The values from all five flex sensors are then compared to predefined threshold values in the Arduino code to determine which fingers are bent.

Step 2: Binary Code Generation

Once the analog values are read from the flex sensors, the Arduino compares them with pre-set threshold values to identify if a finger is bent or not. If a finger is bent, its corresponding binary value is set to 0, and if it is not bent, the value is set to 1. The five binary values are then combined to form a 5-digit binary code. This code is unique for each hand gesture, as different combinations of finger positions result in different binary codes.

Step 3: Audio File Retrieval from SD Card

The 5-digit binary code is then used to map the hand gesture to a specific audio file stored on the microSD card. Each gesture corresponds to a specific audio file in the .wav format stored on the SD card. The Arduino Uno, equipped with an SD card module, accesses the SD card to fetch the appropriate audio file based on the generated binary code. The SD card module reads the data and sends it to the Arduino to trigger playback.

Step 4: Audio Playback through Loudspeaker

Once the correct audio file is identified, the Arduino triggers the playback of the corresponding audio file through the connected loudspeaker. The audio file, stored on the SD card, is played through a compatible audio playback module, such as a DF-Player Mini. The loudspeaker will emit the pre-recorded sound or speech, allowing the user to communicate through hand gestures, which is especially beneficial for non-verbal communication.

Step 5: Continuous Gesture Recognition

The system is designed to continuously monitor the flex sensors for changes in finger positions. As the user changes their hand gesture, the sensors update the analog values, and the Arduino constantly compares them with the threshold values to generate the correct binary code. This process ensures that the system can detect a wide variety of hand gestures and provide the corresponding audio feedback in real-time, facilitating dynamic communication for individuals with speech impairments.

CHAPTER 8
RESULT AND DISCUSSION

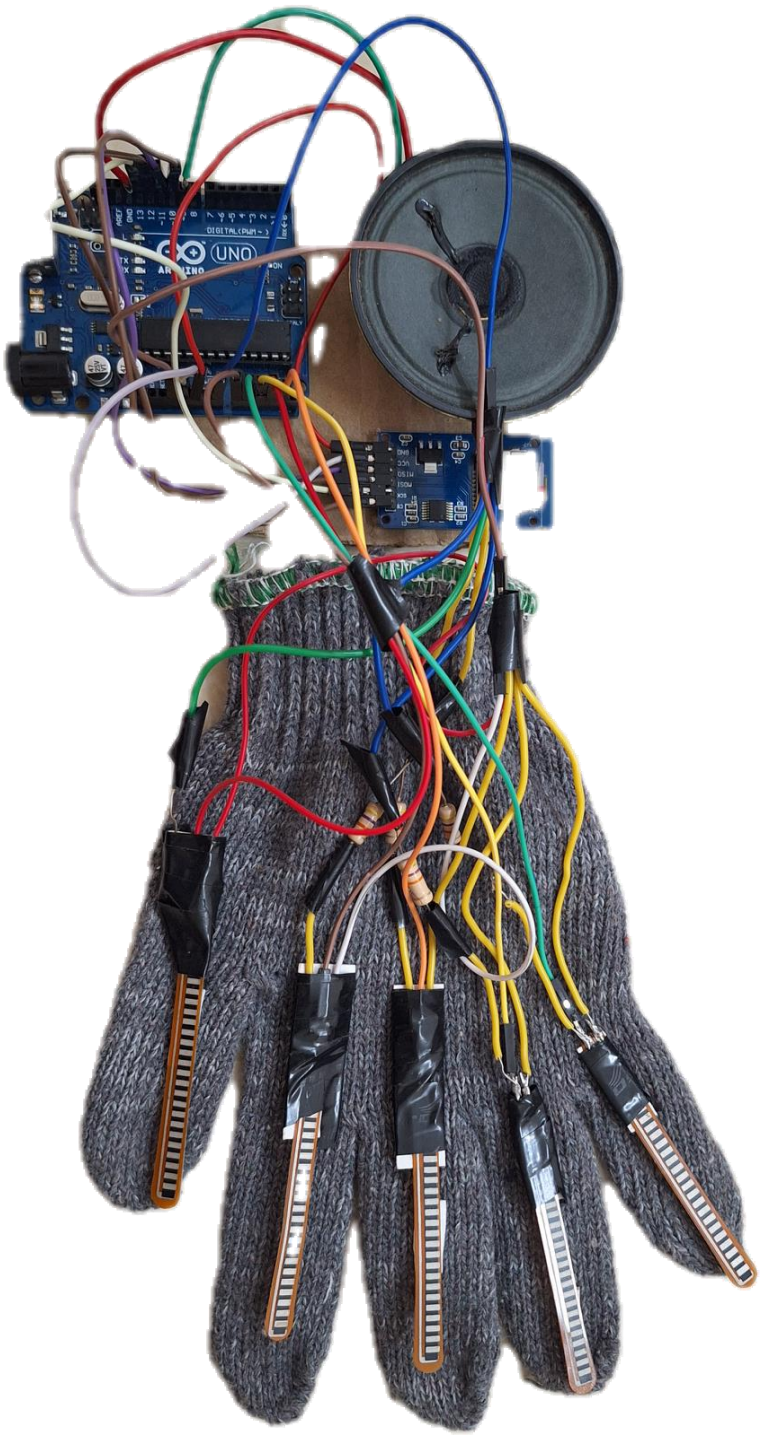

Figure 8.1 Result of Gesture Vox

The core design consists of a glove equipped with flex sensors on each finger, which capture finger movements as analog signals. Each flex sensor detects the degree of bending in the fingers, generating unique signals corresponding to specific hand gestures. These signals are then transmitted to an Arduino Uno microcontroller, which processes the input and identifies predefined gestures.

A microSD card module, also connected to the Arduino, stores audio files associated with each gesture. When a gesture is detected, the Arduino retrieves the corresponding audio file from the SD card, and the loudspeaker placed outside the glove plays the audio, transforming silent gestures into clear, audible speech. This process enables the user to communicate essential phrases through simple hand movements, making the device practical for everyday interactions.

This prototype demonstrates an effective approach to assistive technology, providing a direct, gesture-to-audio pathway for communication. Despite its functional success, the current setup can benefit from improvements in miniaturization and user comfort. Reducing the size of components, organizing wiring, and potentially integrating the microcontroller and SD module directly into the glove would make it more comfortable for prolonged use. Future iterations could also explore wireless connectivity and advanced gesture recognition, making the device even more adaptable and scalable for diverse communication needs. Overall, the "Gesture to Voice Converter" offers a promising solution for enhancing non-verbal communication, with significant potential for further development.

CHAPTER 9
9.1 CONCLUSION

The "Gesture Vox " project has successfully created an innovative assistive device that bridges the gap between silent gestures and audible speech, enabling non-verbal communication for individuals with speech impairments. By combining flex sensors, microcontroller processing, and audio playback, the device translates specific hand gestures into distinct audio responses. This intuitive and modular design offers a straightforward yet impactful communication tool, providing users with a reliable means to express essential phrases and enhancing their ability to connect with those around them.

This project not only addresses a significant communication need but also showcases the potential of affordable, adaptable assistive technology. The system lays a strong foundation for further enhancements, such as expanding its vocabulary, improving gesture recognition accuracy, and enhancing the comfort and wearability of the glove. Through these future improvements, the "Gesture Vox " can evolve into an even more powerful, customizable solution that continues to support diverse communication needs and empower individuals with disabilities in their daily lives.

9.2 FUTURE SCOPE

The future scope for the Gesture Vox holds exciting advancements aimed at making the system more practical and accessible. Initial goals include:

1. Miniaturization: Reducing the size of the components to create a more lightweight, comfortable, and unobtrusive glove for daily wear.
2. Enhanced Gesture Recognition: Expanding the system's ability to recognize a broader range of gestures, allowing for more complex phrases and communication needs.
3. Vocabulary Expansion: Incorporating a more extensive vocabulary to support diverse communication scenarios.
4. Wireless Connectivity: Adding Bluetooth or Wi-Fi capabilities for seamless integration with mobile apps or devices, allowing for remote updates and potentially enabling customization of responses.
5. Battery Optimization: Improving power efficiency to support longer usage times between charges.

These developments will not only improve usability but also broaden the device’s applications, creating a versatile communication aid for users in a variety of settings.

CHAPTER 10
REFERENCE

- Alert Hand Gesture Recognition Using Flex Sensors and Microcontroller
  https://doi.org/10.1109/ISCAIE61308.2024.10576251
  Publisher: IEEE
  Published on: 23 April 2020

- Glove-Based Assistive Device for the Speech and Hearing Impaired
  https://doi.org/10.1109/HNICEM48295.2019.9072695
  Publisher: IEEE
  Published on: 10 May 2021

- Embedded Glove System for Speech-Impaired People
  https://doi.org/10.1109/ICACDOT.2016.7877706
  Publisher: IEEE
  Published on: 16 March 2017

- Analysis of Flex Sensor Technology for Wearable Systems
  https://doi.org/10.1109/ICEEICT61591.2024.10718430
  Publisher: IEEE
  Published on: 26 July 2024

- Gesture Controlled Speaker with LC
  https://doi.org/10.1109/ICIPCN63822.2024.00115
  Publisher: IEEE
  Published on: 09 September 2024

APPENDIX

SOURCE CODE

CODE USING EMBEDED C

```
#include <TMRpcm.h>
#include <pcmConfig.h>
#include <pcmRF.h>
#include <SPI.h>
#include <SD.h>
TMRpcm tmrpcm;
// Pin connected to the output of the flex sensor
void setup() {  tmrpcm.speakerPin = 9;
  Serial.begin(9600);  // Correct baud rate is 9600, not 9690
  if (!SD.begin(10)) {
    Serial.println("SD initialization failed!");
    return;
  }
  Serial.println("SD card initialized successfully");

  // Set volume
  tmrpcm.setVolume(5);
  Serial.begin(9600);
}

void loop() {
  // Read the flex sensor output
  int sensorValue1 = analogRead(A0);
  int sensorValue2 = analogRead(A1);
  int sensorValue3 = analogRead(A2);
  int sensorValue4 = analogRead(A3);
  int sensorValue5 = analogRead(A4);
  bool digitalvalue1 = sensorValue1<12? false:true;
  bool digitalvalue2 = sensorValue2<12? false:true;
  bool digitalvalue3 = sensorValue3<12? false:true;
  bool digitalvalue4 = sensorValue3<12? false:true;
  bool digitalvalue5 = sensorValue3<12? false:true;
  Serial.print("Sensor Values: ");
  Serial.print(sensorValue1); Serial.print(", ");
  Serial.print(sensorValue2); Serial.print(", ");
  Serial.print(sensorValue3); Serial.print(", ");
```

```
  Serial.print(sensorValue4); Serial.print(", ");
  Serial.println(sensorValue5);


if(digitalvalue1==false&&digitalvalue2==false&&digitalvalue3==false&
&digitalvalue4==false&&digitalvalue5==false){
     Serial.println("what is the time");
     Serial.flush();
     tmrpcm.play("time.wav");
     }
  else
if(digitalvalue1==false&&digitalvalue2==false&&digitalvalue3==false&
&digitalvalue4==false&&digitalvalue5==true){
     Serial.println("Have a good day");
     tmrpcm.play("goodDay.wav");
     }
  else
if(digitalvalue1==false&&digitalvalue2==false&&digitalvalue3==false&
&digitalvalue4==true&&digitalvalue5==false){
     Serial.println("good job");
     tmrpcm.play("goodJob.wav");
     }
  else
if(digitalvalue1==false&&digitalvalue2==false&&digitalvalue3==false&
&digitalvalue4==true&&digitalvalue5==true){
     Serial.println("good morning");
     tmrpcm.play("goodMorning.wav");
     }
  else
if(digitalvalue1==false&&digitalvalue2==false&&digitalvalue3==true&
&digitalvalue4==false&&digitalvalue5==false){
     Serial.println("hi, how are you");
     tmrpcm.play("hi.wav");
     }
   else
if(digitalvalue1==false&&digitalvalue2==false&&digitalvalue3==true&
&digitalvalue4==false&&digitalvalue5==true){
     Serial.println("nice to meet you ");
     tmrpcm.play("niceToMeetYou.wav");
     }
   else
if(digitalvalue1==false&&digitalvalue2==false&&digitalvalue3==true&
&digitalvalue4==true&&digitalvalue5==false){
     Serial.println("see you again");
```

```
    tmrpcm.play("seeYouAgain.wav");
    }
  else
if(digitalvalue1==false&&digitalvalue2==false&&digitalvalue3==true&
&digitalvalue4==true&&digitalvalue5==true){
    Serial.println("thank you very much");
    tmrpcm.play("thanks.wav");
    }
  else
if(digitalvalue1==false&&digitalvalue2==true&&digitalvalue3==false&
&digitalvalue4==false&&digitalvalue5==false){
    Serial.println("go to washroom ");
    tmrpcm.play("washRoom.wav");
    }
  else
if(digitalvalue1==false&&digitalvalue2==true&&digitalvalue3==false&
&digitalvalue4==false&&digitalvalue5==true){
    Serial.println("need water ");
    }
  else
if(digitalvalue1==false&&digitalvalue2==true&&digitalvalue3==false&
&digitalvalue4==true&&digitalvalue5==false){
    Serial.println("need food ");
    }
  else
if(digitalvalue1==false&&digitalvalue2==true&&digitalvalue3==false&
&digitalvalue4==false&&digitalvalue5==true){
    Serial.println("enough food ");
    }
  else
if(digitalvalue1==false&&digitalvalue2==true&&digitalvalue3==true&&
digitalvalue4==false&&digitalvalue5==false){
    Serial.println("come here ");
    }
  else
if(digitalvalue1==false&&digitalvalue2==true&&digitalvalue3==true&&
digitalvalue4==false&&digitalvalue5==true){
    Serial.println("i need to go out ");
    }
  else
if(digitalvalue1==false&&digitalvalue2==true&&digitalvalue3==true&&
digitalvalue4==true&&digitalvalue5==false){
    Serial.println("i need bath ");
    }
```

```
  else
if(digitalvalue1==false&&digitalvalue2==true&&digitalvalue3==true&&
digitalvalue4==true&&digitalvalue5==true){
    Serial.println("go to bed ");
    }
  else
if(digitalvalue1==true&&digitalvalue2==false&&digitalvalue3==false&
&digitalvalue4==false&&digitalvalue5==false){
    Serial.println(" help to cross signal ");
    }
  else
if(digitalvalue1==true&&digitalvalue2==false&&digitalvalue3==false&
&digitalvalue4==false&&digitalvalue5==true){
    Serial.println("stop ");
    }
  else
if(digitalvalue1==true&&digitalvalue2==false&&digitalvalue3==true&&
digitalvalue4==true&&digitalvalue5==false){
    Serial.println("great applause ");
    }
  else
if(digitalvalue1==true&&digitalvalue2==false&&digitalvalue3==false&
&digitalvalue4==true&&digitalvalue5==true){
    Serial.println("good work ");
    }
  else
if(digitalvalue1==true&&digitalvalue2==false&&digitalvalue3==true&&
digitalvalue4==false&&digitalvalue5==false){
    Serial.println("attend the phone ");
    }
  else
if(digitalvalue1==true&&digitalvalue2==false&&digitalvalue3==true&&
digitalvalue4==false&&digitalvalue5==true){
    Serial.println("oped the door ");
    }
  else
if(digitalvalue1==true&&digitalvalue2==false&&digitalvalue3==true&&
digitalvalue4==true&&digitalvalue5==false){
    Serial.println("switch ON the light ");
    }
  else
if(digitalvalue1==true&&digitalvalue2==false&&digitalvalue3==true&&
digitalvalue4==true&&digitalvalue5==true){
    Serial.println("switch ON the fan");
```

```
    }
  else
if(digitalvalue1==true&&digitalvalue2==true&&digitalvalue3==false&&
digitalvalue4==false&&digitalvalue5==false){
    Serial.println("switch OFF the fan ");
    }
  else
if(digitalvalue1==true&&digitalvalue2==true&&digitalvalue3==false&&
digitalvalue4==false&&digitalvalue5==true){
    Serial.println("go for walking ");
    }
  else
if(digitalvalue1==true&&digitalvalue2==true&&digitalvalue3==false&&
digitalvalue4==true&&digitalvalue5==false){
    Serial.println(" need my wheel chair");
    }
  else
if(digitalvalue1==true&&digitalvalue2==true&&digitalvalue3==false&&
digitalvalue4==true&&digitalvalue5==true){
    Serial.println(" switch OFF the light");
    }
 else
if(digitalvalue1==true&&digitalvalue2==true&&digitalvalue3==true&&
digitalvalue4==false&&digitalvalue5==false){
    Serial.println("i am happy ");
    }
 else
if(digitalvalue1==true&&digitalvalue2==true&&digitalvalue3==true&&
digitalvalue4==false&&digitalvalue5==true){
    Serial.println("dislike ");
    }
 else
if(digitalvalue1==true&&digitalvalue2==true&&digitalvalue3==true&&
digitalvalue4==true&&digitalvalue5==false){
    Serial.println("keep rocking");
    }
 else
if(digitalvalue1==true&&digitalvalue2==true&&digitalvalue3==true&&
digitalvalue4==true&&digitalvalue5==true){
    Serial.println("stop ");
    }
 delay(3000);
}
```