GESTURE VOCALIZER WITH LCD DISPLAY

Project Report

Submitted in partial fulfilment of the requirements for the award of the degree of

Bachelor of technology

ELECTRONICS AND COMMUNICATION ENGINEERING

Namita Rastogi (2020BECE034) **Manan Sanson** (2020BECE039)

Under the supervision of

Prof. Aijaz Ahmad Mir

NATIONAL INSTITUTE OF TECHNOLOGY, SRINAGAR

Department of Electronics and Communication Engineering
National Institute of technology, Srinagar

CERTIFICATE

This is to certify that the project report entitled **Gesture vocalizer with LCD display** submitted by **Namita Rastogi** and **Manan Sanson** to the Department of Electronics and Communication Engineering, National Institute of Technology Srinagar, Kashmir, in partial fulfilment for the award of the degree of **B. Tech in Electronics and Communication Engineering** is a bona fide record of project work carried out by him/her under my/our supervision.

Prof. Aijaz Ahmad Mir
(Supervisor)

Department of Electronics and Communication Engineering

National Institute of Technology, Srinagar

STUDENTS' DECLARATION

We declare that this project report titled 'Gesture Vocalizer with LCD display' submitted in partial fulfilment of the degree of B.Tech in Electronics and Communication Engineering is a record of original work carried out by me under the supervision of Prof. Aijaz Ahmad Mir. The matters embodied in this project, in full or in parts, have not been submitted to any other Institution or University for the award of any degree or diploma. I also declare that the work submitted by me is entirely original, free from plagiarism.

Dated:

(i) Name: **Namita Rastogi**
Signature: _____

(ii) Name: **Manan Sanson**
Signature:_____

ACKNOWLEDGEMENT

It is our privilege to express our sincerest regards to our department head **Prof. Gausia Quazi** for encouraging and allowing us to present the project on the topic **Gesture Vocalizer with LCD display** at our department premises for the partial fulfilment of the requirements leading to the award of B.Tech degree. We deeply express our sincere thanks to our project guide, **Prof. Aijaz Ahmad Mir**, for his valuable inputs, able guidance, encouragement, whole-hearted cooperation and constructive criticism throughout the duration of this project. We take this opportunity to express gratitude to all of the department faculty members for their help and support. We also place on record our sense of gratitude to one and all, who directly or indirectly, have lent their hand in this venture.

Namita Rastogi (2020BECE034)

Manan Sanson (2020BECE039)

Abstract

The "Gesture Vocalizer" project addresses the communication challenges faced by the deaf and dumb community by introducing an innovative Hand Signs to Speech conversion system. Leveraging the capabilities of Arduino Uno, this project aims to create an accessible and efficient solution that facilitates seamless communication between individuals proficient in sign language and those relying on verbal communication.

The project encompasses the design and implementation of a wearable device capable of recognizing and interpreting sign language gestures through sensors and translating them into audible speech and LCD output. The Arduino Uno microcontroller serves as the core processing unit, orchestrating the integration of various sensors, including accelerometers and flex sensors, to capture and analyse hand movements and gestures.

The report details the theoretical foundations, design methodologies, implementation strategies, and experimental results of the Gesture Vocalizer system. The outcomes of user testing and performance evaluations demonstrate the effectiveness and reliability of the proposed solution. Future enhancements and potential applications in assistive technology further underscore the project's significance in the realm of electronic communication systems for individuals with unique communication needs.

Contents

List of Figures

Chapter 1
Introduction

In the realm of electronic communication systems, the Gesture Vocalizer project emerged as a pioneering endeavour, focusing on the development of a sophisticated Sign Language to Speech conversion system tailored to the specific needs of the deaf and dumb community. The profound impact of effective communication on individual well-being and societal inclusivity cannot be overstated, and this project addresses a critical gap by providing a technological bridge between the expressive world of sign language and the auditory realm of spoken language. The deaf and dumb community faces distinct challenges in verbal communication, relying primarily on sign language as a means of expressing thoughts, emotions, and ideas. While sign language serves as a rich and nuanced mode of communication, it can present barriers when interacting with those who are not proficient in its interpretation. The Gesture Vocalizer project seeks to break down these barriers by harnessing the power of technology, specifically through the integration of Arduino Uno and advanced sensors, to facilitate seamless communication between individuals who communicate through sign language and those who rely on spoken language. This introduction sets the stage for a comprehensive exploration of the technical intricacies, theoretical foundations, and practical applications of the Gesture Vocalizer system. By leveraging data-driven approaches, advanced gesture recognition algorithms, and real-time processing capabilities, the project aims to offer a robust, accurate, and low-latency solution for translating sign language gestures into audible speech. The subsequent sections of this report delve into the project's methodology, implementation details, experimental results, and the potential implications of its findings in the broader context of assistive technology and electronic communication systems. Through the fusion of technology and communication, the Gesture Vocalizer project endeavours to empower individuals with hearing and speech impairments, fostering a more inclusive and connected society.

1.1 Problem Statement & Objectives

Problem Statement

Effective communication is fundamental to human interaction and social integration. For individuals who are deaf or mute, traditional verbal communication presents significant challenges, often leading to feelings of isolation and difficulty in expressing thoughts and emotions to those who do not understand sign language. Existing communication aids such as sign language interpreters and text-based communication tools have limitations in accessibility and practicality, especially in real-time conversational scenarios.

The primary challenge is to create a system that can accurately and efficiently translate sign language into audible speech, enabling seamless communication between sign language users and those who rely on verbal language. The system should be user-friendly, portable, and capable of recognizing a wide range of gestures with high accuracy and low latency.

Objectives

1. **Develop a Gesture Recognition System**:

   Design and implement a wearable device equipped with sensors (accelerometers, flex sensors) to capture hand gestures.

   Utilise the Arduino Uno microcontroller to process sensor data and interpret gestures.

2. **Translate Gestures into Speech**:

   Implement algorithms to convert recognized gestures into corresponding speech outputs.

   Integrate a speech synthesis module to vocalise the interpreted gestures.

3. **Create a User Interface with LCD Display**:

   Incorporate an LED display to provide visual feedback and user prompts.

   Ensure the display can show relevant information such as recognized gestures and system status.

4. **Ensure Real-time Processing and Accuracy**:

   Achieve real-time gesture recognition and translation with minimal delays.

Strive for high accuracy in gesture recognition to ensure effective communication.

5. **Design a Portable and User-friendly Device**:

   Focus on the ergonomics and portability of the wearable device.

   Ensure ease of use for individuals with varying levels of technological proficiency.

6. **Test and Validate the System**:

   Conduct user testing to evaluate the system's effectiveness and reliability in real-world scenarios.

   Gather feedback to identify areas for improvement and potential enhancements.

7. **Explore Future Enhancements**:

   Investigate the integration of additional features such as multi-language support and customizable gestures.

   Consider the application of the system in different contexts such as educational settings, public services, and personal communication aids.

These components provide a comprehensive framework for the Gesture Vocalizer with LCD Display project, addressing the communication needs of the deaf and mute community by leveraging technology to bridge the gap between sign language and spoken language.

1.2 Organisation

The rest of the report is organised as follows.

We review the existing literature in Chapter 2. In chapter 3, we introduced the components used and the connections made. In chapter 4, we discussed the implementation details of the project, explaining all its features, and specified the code snippets wherever required. In chapter 5, we have displayed the results achieved. Finally in chapter 6, we have briefly summarised the project, and given directions for future work.

Chapter 2

Literature Survey

In this chapter, we survey the recent literature to gain insights into the prevailing research in the field of Gesture Vocalizers. These systems are designed to convert hand gestures into audible speech or text, facilitating communication for individuals with speech and hearing impairments. The literature provides a foundation for understanding the technologies, applications, and challenges associated with Gesture Vocalizers.

Gesture Vocalizer Systems and Sensor Technologies

1. **Gesture Vocalizer Using Flex Sensor and Software Visualization** (Kumar, 2022)

   Description: This paper discusses a multi-microcontroller-based Gesture Vocalizer using flex sensors and an Arduino Mega microcontroller. Flex sensors detect hand gestures by measuring resistance changes as the fingers bend, converting these movements into speech.

   Year: 2022

2. **Finger Gesture Vocalizer** (International Journal of Engineering and Advanced Technology, 2019)

   Description: This paper explores the Finger Gesture Vocalizer, where sensors attached to gloves capture finger movements to translate them into speech. The system uses an APR33A3 audio processing unit to convert these movements into audible outputs.

   Year: 2019

3. **A Review on Gesture Vocalizer** (Nath, 2018)

   Description: This paper reviews the basic architecture of Gesture Vocalizers, focusing on the use of accelerometers and flex sensors embedded in gloves to detect hand movements. These sensors convert hand gestures into electrical signals, which are then processed to produce speech outputs.

   Year: 2018

4. **An Enhanced Gesture Vocaliser for the Vocally Challenged People** (Sinthu et al., 2018)

   Description: This study presents a Gesture Vocalizer using image processing techniques to capture and interpret gestures. The input gestures are compared with a database to identify and produce corresponding speech outputs.

   Year: 2018

Applications and Benefits

5. **Gesture Vocalizer** (Haridas et al., 2023)

   Description: The system converts sign language into voice by using sensors to capture hand movements, which are then processed and translated into speech. This approach allows for effective communication over the internet, using real-time video and cloud storage.

   Year: 2023

6. **Hand Gesture-Based Vocalizer for the Speech Impaired** (Ail et al., 2019)

   Description: This paper details a microcontroller-based system that converts hand gestures into audible sounds, making it easier for speech-impaired individuals to communicate. The system is customizable to accommodate different hand sign conventions.

   Year: 2019

Challenges and Future Directions

7. **Gesture Vocalizer Using Flex Sensor and Software Visualization** (Rishitha et al., 2022)

   Description: Highlights challenges such as ensuring accuracy in gesture recognition and adapting to different user needs. Future directions include integrating more robust machine learning models for better recognition and personalization.

   Year: 2022

8. **Speak with Your Hands - Using Continuous Hand Gestures to Control Articulatory Speech Synthesizer** (Saha et al., 2021)

   Description: Explores controlling a speech synthesizer using continuous hand gestures. The system uses a Cyberglove II to capture finger and wrist movements, which are translated into speech sounds through a virtual tongue model.
   Year: 2021

Literature Review

Recent developments in Gesture Vocalizers have introduced advanced technologies and innovative approaches. Haridas et al. (2023) presented a cutting-edge Gesture Vocalizer that converts sign language into voice using sensors and cloud storage, highlighting the integration of modern data management techniques. Kumar (2022) discussed a microcontroller-based system in "Gesture Vocalizer Using Flex Sensor and Software Visualization", focusing on flex sensors and software visualization. Rishitha et al. (2022) revisited the subject in their work of the same title, exploring the challenges and future directions for Gesture Vocalizers. Saha et al. (2021), in "Speak with Your Hands", investigated the use of continuous hand gestures to control a speech synthesizer, demonstrating a novel approach to gesture-based speech synthesis.

Earlier advancements laid the groundwork for these innovations. "Finger Gesture Vocalizer" (2019) introduced gloves with sensors to capture finger movements, translating them into speech. Ail et al. (2019) explored customizable settings in their "Hand Gesture-Based Vocalizer for the Speech Impaired", converting hand gestures to audible sounds. Sinthu et al. (2018) utilized image processing to translate gestures into speech in their work, "An Enhanced Gesture Vocaliser for the Vocally Challenged People". Nath (2018) provided a comprehensive review of Gesture Vocalizers, focusing on the use of accelerometers and flex sensors, establishing a foundational understanding of the technology.

Chapter 3
Components Used and Connections

ARDUINO UNO

The Arduino Uno is a widely-used microcontroller board that incorporates the ATmega328P microcontroller. It features 14 digital input/output pins, 6 of which can serve as PWM (Pulse Width Modulation) outputs, and 6 analog inputs. The board is equipped with a 16 MHz quartz crystal, a USB interface, a power jack, an ICSP header, and a reset button. This combination of components provides all necessary functionality to support the microcontroller. The board can be powered by connecting it to a computer via a USB cable or by using an AC-to-DC adapter or battery.

Vin: This pin accepts voltage input from an external power source to supply the board.

5V: This regulated power supply pin provides 5 volts to the board and connected components.

3.3V: This pin supplies 3.3 volts generated by an onboard voltage regulator.

GND: Ground pin for completing electrical circuits.

Reset: Used to restart the microcontroller.

Analog Pins (A0 to A5): These pins read analog input signals within a range of 0 to 5 volts.

Digital Pins (0 to 13): These can serve as digital inputs or outputs.

Serial Pins: Known as UART pins, pin 1 (TX) transmits data, and pin 0 (RX) receives data for serial communication.

External Interrupt Pins (2 and 3): Allow external devices to interrupt the microcontroller.

PWM Pins (3, 5, 6, 9, 10, 11): Used for generating analog outputs by varying pulse widths.

SPI Pins: Essential for Serial Peripheral Interface communication.

- **SS (Pin 10):** Slave Select
- **MOSI (Pin 11):** Master Out Slave In
- **MISO (Pin 12):** Master In Slave Out
- **SCK (Pin 13):** Serial Clock

LED Pin: The onboard LED is controlled by digital pin 13 and lights up when the pin is set to high.

AREF Pin: Analog Reference pin for providing external reference voltage.

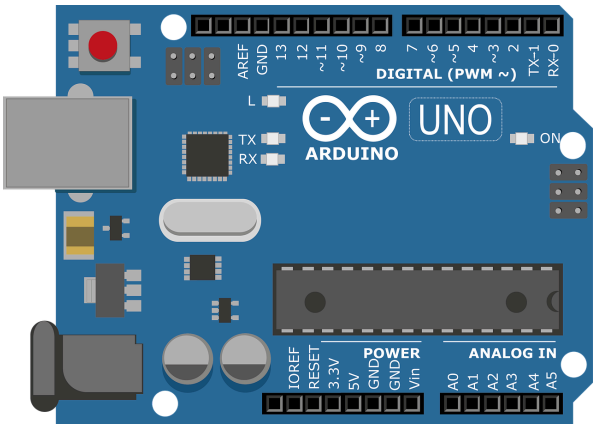

Fig. 1: Arduino UNO

FLEX SENSOR

A flex and force sensor is a versatile device that modifies its electrical resistance when bent or subjected to force. It has two output wires through which the varying resistance can be measured, making it suitable for detecting bending and force.

The sensor consists of a material printed with polymer ink containing conductive particles. When the sensor is in a straight position, the resistance of the ink is approximately 30k Ohms. As the sensor bends, the distance between the conductive particles increases, causing the resistance to rise (typically between 50k to 70k Ohms when bent to a 90° angle, as shown in the diagram).

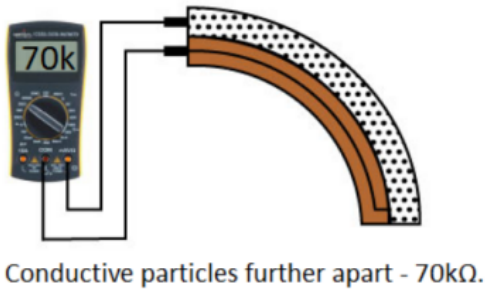

Fig. 2: Flex sensor bend

Upon straightening, the resistance returns to its initial value. By monitoring these resistance changes, it is possible to determine the degree of bending experienced by the sensor.

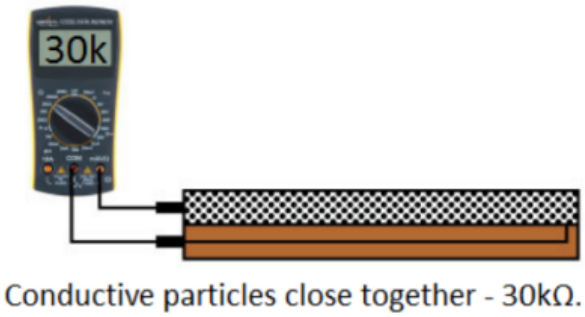

Fig. 3: Flex sensor straight

LCD DISPLAY

An LCD screen is an electronic display module commonly used in various applications. Character LCDs are specialised displays that show individual ASCII characters of fixed size. They use a grid of 5×8 pixels in small rectangular sections. Each pixel can be illuminated individually to form characters within these grids.

● 16×2 LCD Pinout

The 16×2 LCD has 16 pins, arranged as follows from left to right:

1. **Ground (GND):** Connects to ground.
2. **VCC:** Connects to a 5V supply, typically from an Arduino board.
3. **Vo:** Connects to a potentiometer to adjust the display contrast.
4. **RS (Register Select):** Selects whether the LCD receives commands or data. A low state (0V) sends commands (e.g., cursor positioning, display clearing), while a high state (5V) sends data (characters).
5. **R/W (Read/Write):** Determines the mode of operation—whether data will be written to or read from the LCD. For typical usage, this is set to write mode.
6. **E (Enable):** Activates the writing process to the registers. 7-14. **D0 to D7:** Data pins used to send 8-bit data to the LCD. For instance, sending the binary sequence 0100 0001 displays the letter 'A' on the screen according to the ASCII table.
7. **A (Anode):** Connects to the LED backlight's positive terminal.
8. **K (Cathode):** Connects to the LED backlight's negative terminal.

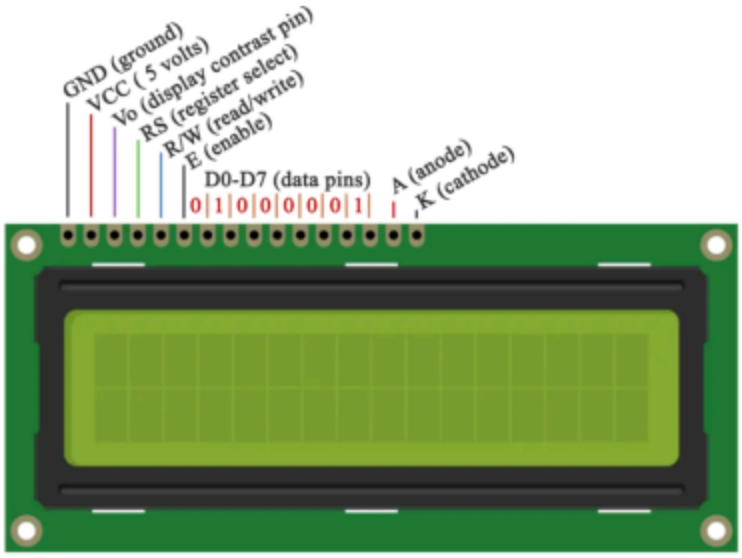

Fig. 4: LCD Display Pin configuration

How to Connect Arduino to LCD – Wiring Diagram

For connecting the LCD to an Arduino, only 6 digital pins are required. The LCD pins D4 to D7 connect to Arduino digital pins 4 to 7, respectively. The Enable pin (E) connects to Arduino pin 2, and the RS pin connects to Arduino pin 1. The R/W pin is connected to ground, and the Vo pin is connected to the middle pin of a potentiometer to control the contrast.

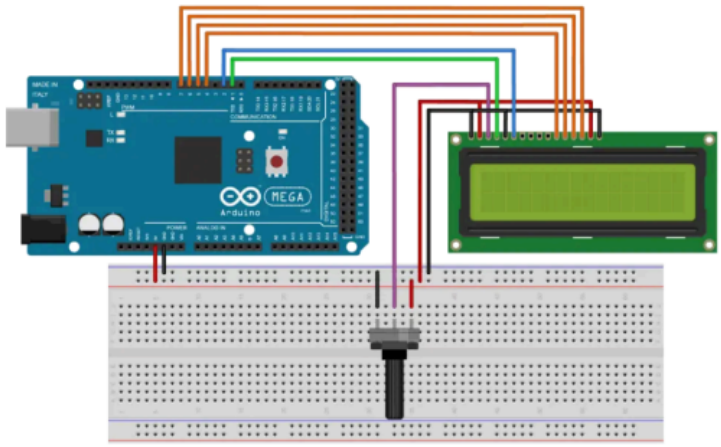

Fig. 5: LCD display connection with Arduino

ACCELERATOR

The ADXL345 is a small, thin, low power, 3-axis accelerometer with high resolution (13-bit) measurement at up to ±16g. Digital output data is formatted as 16-bit twos complement and is accessible through either a SPI (3- or 4-wire) or I2C digital interface.

The ADXL345 measures the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (4 mg/LSB) enables measurement of inclination changes less than 1.0°.

Several special sensing functions are provided. Activity and inactivity sensing detect the presence or lack of motion by comparing the acceleration on any axis with user-set thresholds. Tap sensing detects single and double taps in any direction. Free-fall sensing detects if the device is falling. These functions can be mapped individually to either of two interrupt output pins. An integrated memory management system with a 32-level first in, first out (FIFO) buffer can be used to store data to minimize host processor activity and lower overall system power consumption.

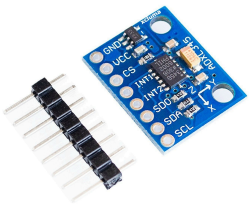

Fig. 6: ADXL345 Accelerometer

Circuit Diagram:

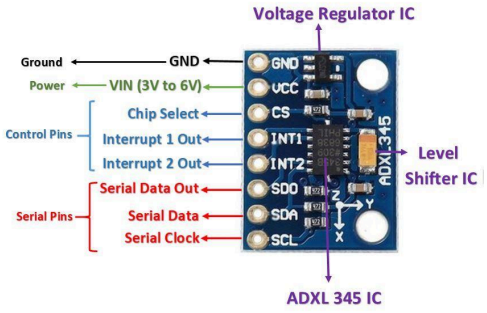

Fig. 7: Accelerometer pins

DF Player mini

The DFPlayer Mini MP3 Player For Arduino is a small and low cost MP3 module with an simplified output directly to the speaker. It can be used alone or with microcontrollers such as Arduino, ESP32, Raspberry Pi etc.

It takes an SD card that stores audio files to be played. These can be played using DF player mini libraries in Arduino. It has a 30-level adjustable volume. The audio data is sorted by folders, it can have 100 folder that can have up to 225 songs each.

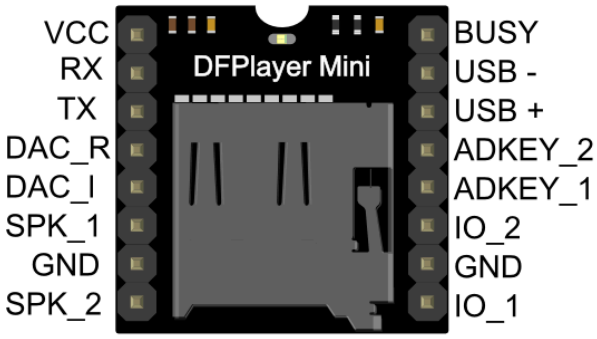

Fig. 8: DF Player Mini

Circuit Diagram:

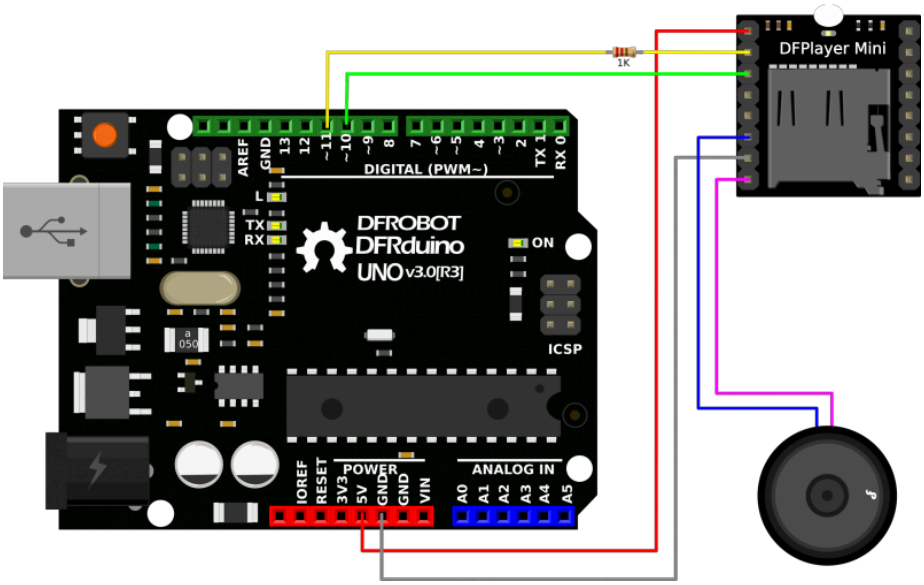

Fig. 9: Connection with Arduino & speaker

I2C module for 16x2 LCD Display

The I2C (Inter-Integrated Circuit) module is commonly used with 1602 LCD displays to simplify wiring and reduce the number of pins required for connection to a microcontroller, such as an Arduino. It has an inbuilt PCF8574 I2C chip that converts I2C serial data to parallel data for the LCD display. It has a 4-pin interface.
The module typically connects to the microcontroller using four pins:

- GND (Ground)
- VCC (Power Supply, usually 5V)
- SDA (Serial Data Line)
- SCL (Serial Clock Line)

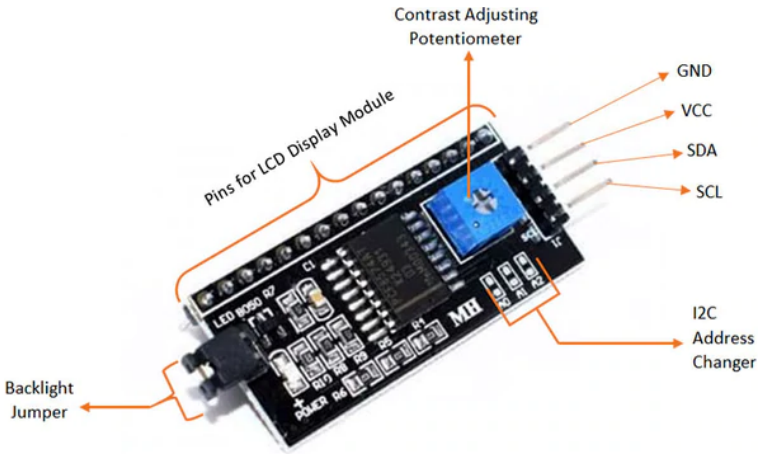

Fig. 10: I2C module

SPEAKER

A loudspeaker is a device designed to convert electrical signals into audio signals. It operates by applying an alternating current electrical audio signal to a voice coil, which is a coil of wire situated in a gap between the poles of a permanent magnet. The interaction between the current in the coil and the magnetic field causes the coil to move rapidly back and forth, according to Faraday's law of induction. This movement is transferred to a cone (often conical in shape) attached to the coil, causing the cone to push against the air and generate sound waves.

While this is the most common method of sound production, various other technologies can also be used to transform electrical signals into sound. Before reaching the loudspeaker, the audio signal (originating from a sound source like a recording or microphone) usually needs to be amplified using an amplifier.

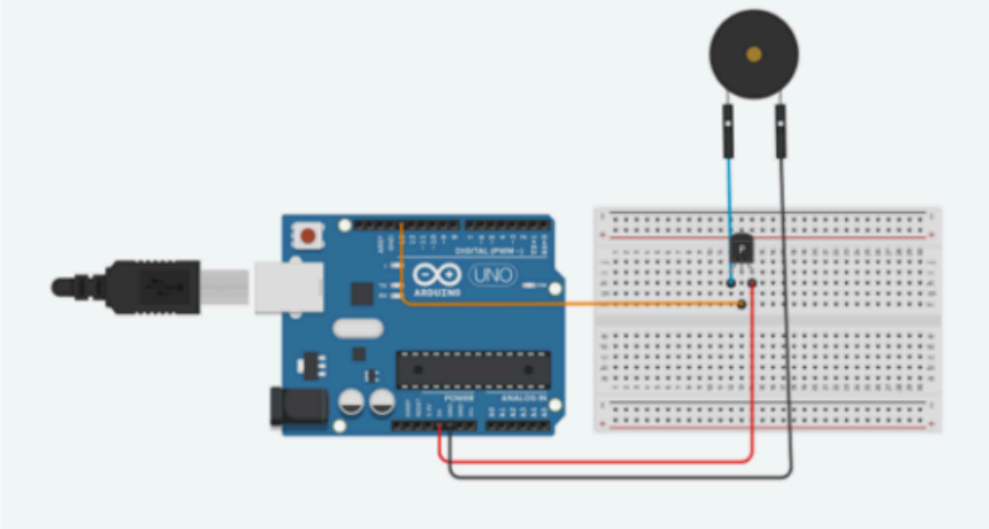

Fig. 11: Speaker connections with Arduino

Chapter 4
Implementation

This chapter elucidates the project overview, including the code and key technologies used in the project.

1. Implementation Steps

Flex sensors are utilized to measure the bending of fingers, providing precise feedback on the extent of flexion. Accelerometers capture both the orientation and movement of the hand, enabling accurate detection of gestures. An LCD display visually represents recognized gestures alongside corresponding text, enhancing user interaction. Additionally, a speaker produces audible speech output, enriching the user experience with vocal feedback based on detected gestures.

2. Hardware Setup

Flex sensors, attached along the length of each finger, measure finger bending, while accelerometers placed on the back of the hand or wrist capture hand orientation and movement using I2C or SPI interfaces for data acquisition. An LCD display, typically a 16x2 or 20x4 character screen with an I2C interface, connects via I2C pins to the microcontroller, which displays recognized gestures and corresponding text. A small speaker with an audio amplifier, connected to the microcontroller's digital pin for PWM control or an external audio module, provides audible speech output. A microcontroller, such as an Arduino or ESP32, manages sensor data acquisition, processing, LCD control, and audio output, integrating all components for effective gesture recognition and communication.

3. Data Acquisition

Flex sensors generate analog signals from each finger, which are then read and normalized to a standard range for consistency in data processing. Accelerometers provide 3-axis data (X, Y, and Z) capturing hand orientation and movement. Calibration is essential to remove any offsets and ensure accurate data interpretation from the accelerometers.

4. Preprocessing

Segmentation involves identifying gesture boundaries by analyzing changes in sensor data. This process helps distinguish one gesture from another based on variations detected by the flex sensors and accelerometers.

5. Gesture Mapping

Gesture mapping employs threshold-based logic to interpret sensor data. For flex sensors, thresholds define finger states according to American Sign Language and short messages, such as considering a finger bent if the sensor value exceeds 700. Similarly, accelerometers use thresholds for orientation, like interpreting an X-axis value over 500 as a right tilt. By combining these sensor readings, specific gestures are defined. For example, a clenched fist is identified when all flex sensor values exceed 700 and the accelerometer reading is near zero, while an open hand is recognized when all flex sensor values are below 300 with the accelerometer also near zero. These gestures are then mapped to corresponding text using a lookup table, enabling gesture-to-text conversion.

6. Visual Feedback on LCD

For text display, show the recognized gesture name and its corresponding text; for instance, a clenched fist would display “Gesture: Fist” and “Text: Hello.” For audio output, integrate a text-to-speech (TTS) module, like DFPlayer Mini, or an external service to convert the text to speech, connecting the audio output to the speaker to play the corresponding audio for the recognized gesture.

8. Integration and Testing

Integrating sensor data acquisition, gesture recognition, LCD display functionality, and audio output creates a unified system. This integration ensures that data from flex sensors and accelerometers are processed to accurately identify gestures. The LCD display then communicates the recognized gesture name and corresponding text, while audio output, facilitated by a text-to-speech module, provides audible feedback. Testing involves validating the system's performance across different gestures, refining thresholds and mappings as needed to enhance accuracy and usability based on user feedback and testing scenarios.

9. Deployment

Deploy on a suitable platform with portable power. Provide documentation for usage and calibration.

FlowChart:

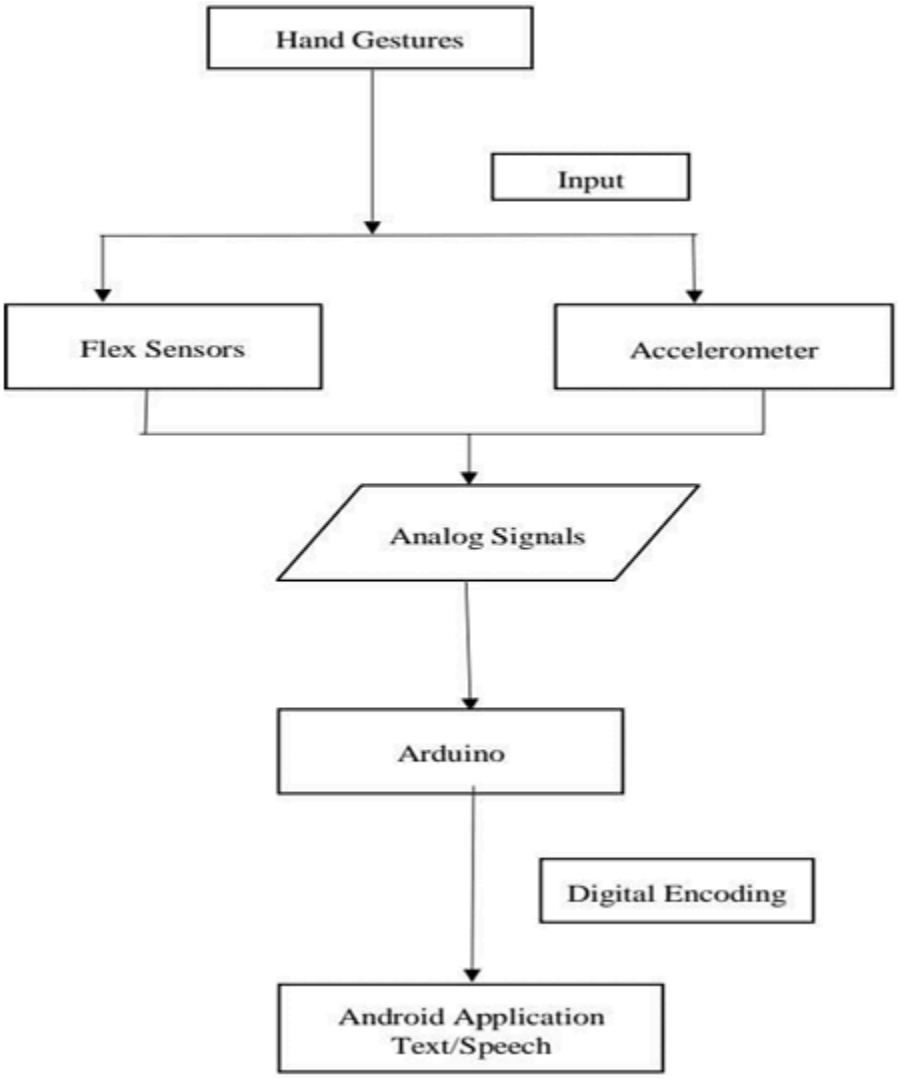

Fig. 12: Flowchart of software design

Project Image:

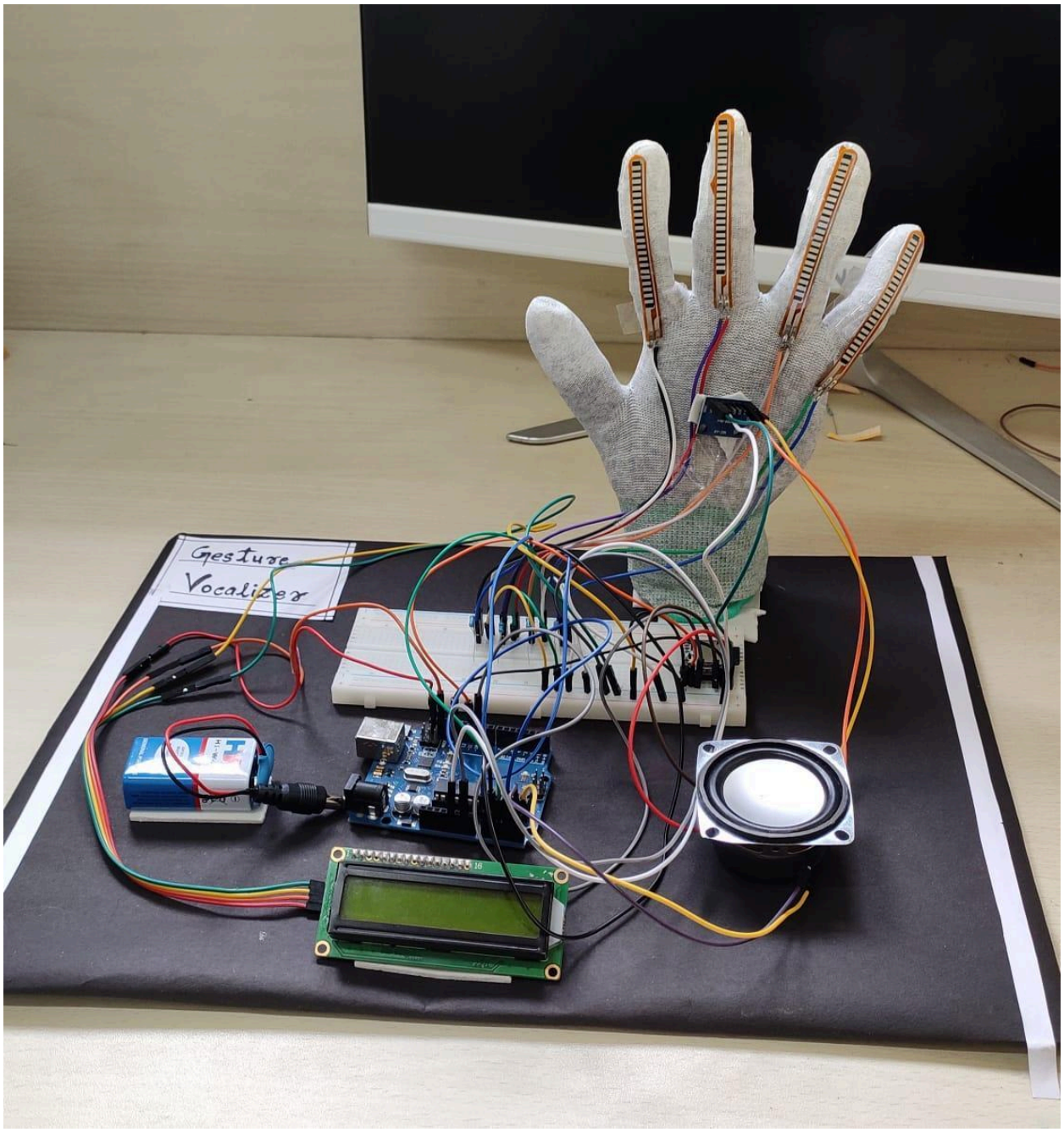

Fig 13: Project Image

Methodology

1. Flex sensors:

First we connected one flex sensor to the Arduino as shown below and calibrated it to get the threshold values using the following code.

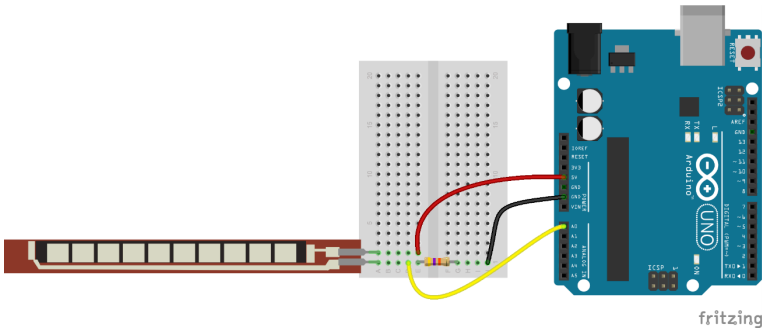

Fig. 14: Flex sensor connections with Arduino

Code to check the threshold values:

```
#include <Wire.h>

#define adc1 A0
#define adc2 A1
#define adc3 A2
#define adc4 A3
#define ledd 13
int flex2=0,flex1=0,flex3=0,flex4=0;
void setup()
{
 Serial.begin(9600 );
}

void loop()
{
  flex1=analogRead(adc1);
  flex2=analogRead(adc2);
  flex3=analogRead(adc3);
  flex4=analogRead(adc4);

  Serial.print(flex1);
  Serial.print("-");
  Serial.print(flex2);
  Serial.print("-");
  Serial.print(flex3);
  Serial.print("-");
  Serial.print(flex4);
  Serial.println("");
  delay(1000);
}
```

2. Accelerometer:

We connected one ADXL345 sensor to the Arduino as shown below and calibrated it to get the threshold values of acceleration in X, Y and Z direction using the following code.

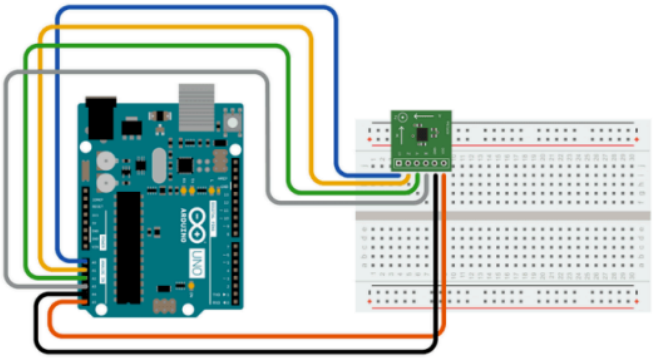

Fig. 15: Accelerometer connections with arduino

The ADXL345 can be configured to measure different ranges of acceleration: ±2g, ±4g, ±8g, and ±16g, where "g" is the acceleration due to gravity (approximately 9.8 m/s²).

Accelerometer Range Settings:

1. ±2g Range: The accelerometer can measure acceleration values from -2g to +2g.
2. ±4g Range: The accelerometer can measure acceleration values from -4g to +4g.
3. ±8g Range: The accelerometer can measure acceleration values from -8g to +8g.
4. ±16g Range: The accelerometer can measure acceleration values from -16g to +16g.

Possible Acceleration Values:

1. For each range setting, the possible values for acceleration in x, y, and z axes are as follows:
2. ±2g Range: -19.6 m/s² to +19.6 m/s² (approx.)
3. ±4g Range: -39.2 m/s² to +39.2 m/s² (approx.)
4. ±8g Range: -78.4 m/s² to +78.4 m/s² (approx.)
5. ±16g Range: -156.8 m/s² to +156.8 m/s² (approx.)

Scaling Factor:

The raw data from the ADXL345 needs to be scaled to get the actual acceleration in g. The scaling factor depends on the range setting:

- ±2g Range: Scale factor is 1/256 (since 2g range with 10-bit resolution gives 256 LSB per g)
- ±4g Range: Scale factor is 1/128
- ±8g Range: Scale factor is 1/64
- ±16g Range: Scale factor is 1/32

Example of Applying the Scaling Factor:

If you read a raw value of 128 in the ±2g range, the actual acceleration is:

Acceleration = $128 \times 1256 = 0.5g$

Acceleration = $128 \times 2561 = 0.5g$

By using the scaling factor, we can convert raw accelerometer readings to meaningful acceleration values in g.

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL345_U.h>

// Initialize the ADXL345 accelerometer
Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345);

void setup() {
  Serial.begin(9600);
  Wire.begin();

  // Initialize the accelerometer
  if (!accel.begin()) {
    Serial.println("No ADXL345 detected");
    while (1);
  }

  // Set the range to ±2g (you can change this to ±4g, ±8g, or ±16g)
  accel.setRange(ADXL345_RANGE_2_G);
}

void loop() {
  sensors_event_t event;
  accel.getEvent(&event);

  // Print the acceleration values in g
  Serial.print("X: "); Serial.print(event.acceleration.x); Serial.print(" m/s^2");
  Serial.print("\tY: "); Serial.print(event.acceleration.y); Serial.print(" m/s^2");
  Serial.print("\tZ: "); Serial.print(event.acceleration.z); Serial.println(" m/s^2");

  delay(500); // Adjust the delay as necessary
}
```

After finding threshold values for the flex sensors and accelerometer we put them in out code to play corresponding audio and text on LCD.

Code:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SoftwareSerial.h>
#include <DFRobotDFPlayerMini.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL345_U.h>

// Define flex sensor pins
const int flexPin1 = A0;
const int flexPin2 = A1;
const int flexPin3 = A2;
const int flexPin4 = A3;

// Define thresholds for finger position detection
const int threshold = 220;
const int neg = -7;
const int pos = 7;

// Initialize the ADXL345 accelerometer
Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345);

// Define DFPlayer Mini pins
SoftwareSerial mySerial(10, 11); // RX, TX
DFRobotDFPlayerMini myDFPlayer;

// Define LCD properties
LiquidCrystal_I2C lcd(0x27, 16, 2);  // I2C address 0x27, 16 columns, 2 rows

void setup() {
  Serial.begin(9600);
  Wire.begin();
  mySerial.begin(9600);
  lcd.init();
  lcd.backlight();

```

```
  // Initialize DFPlayer Mini
  if (!myDFPlayer.begin(mySerial)) {
    Serial.println("Unable to begin: Check connection or SD card.");
    //while (true);
  }
  Serial.println("DFPlayer Mini online.");
  //myDFPlayer.volume(30);  // Set volume (0-30)

  // Initialize the accelerometer
  if (!accel.begin()) {
    Serial.println("No ADXL345 detected");
    while (1);
  }

  // Set the range to ±2g (you can change this to ±4g, ±8g, or ±16g)
  accel.setRange(ADXL345_RANGE_2_G);
}

void loop() {
  // Read flex sensor values
  int flexValue1 = analogRead(flexPin1);
  int flexValue2 = analogRead(flexPin2);
  int flexValue3 = analogRead(flexPin3);
  int flexValue4 = analogRead(flexPin4);

  // Read accelerator values
  sensors_event_t event;
  accel.getEvent(&event);

  int x = event.acceleration.x;
  int y = event.acceleration.y;
  int z = event.acceleration.z;
  // Print the acceleration values in g
  Serial.print("X: "); Serial.print(event.acceleration.x); Serial.print(" m/s^2");
  Serial.print("\tY: "); Serial.print(event.acceleration.y); Serial.print(" m/s^2");
  Serial.print("\tY: "); Serial.print(event.acceleration.y); Serial.print(" m/s^2");
  Serial.print("\tZ: "); Serial.print(event.acceleration.z); Serial.println(" m/s^2");
  // Determine finger positions based on threshold
  bool finger1Up = (flexValue1 > threshold);
  bool finger2Up = (flexValue2 > threshold);
  bool finger3Up = (flexValue3 > threshold);
  bool finger4Up = (flexValue4 > threshold);
```

```
78    if(y<neg && x>neg && z>neg){ // straight front
79      if (finger1Up && finger2Up && !finger3Up && !finger4Up) {
80        lcd.clear();
81        lcd.print("Hi, Nice to");
82        lcd.setCursor(0, 1);
83        lcd.print("Meet you.");
84        myDFPlayer.play(7);
85      } else if (finger1Up && finger3Up && !finger2Up && !finger4Up) {
86        lcd.clear();
87        lcd.print("Where is the");
88        lcd.setCursor(0, 1);
89        lcd.print("washroom?");
90        myDFPlayer.play(2);
91      }
92    } else if(y>pos){ // straight back
93      if (finger1Up && finger4Up && !finger2Up && !finger3Up) {
94        lcd.clear();
95        lcd.print("Can you please");
96        lcd.setCursor(0, 1);
97        lcd.print("pass that?");
98        myDFPlayer.play(3);
99      } else if (finger2Up && finger3Up && !finger1Up && !finger4Up) {
100       lcd.clear();
101       lcd.print("Can you please");
102       lcd.setCursor(0, 1);
103       lcd.print("help me?");
104       myDFPlayer.play(4);
105     }
106   }else if (x<neg) {
107     if (finger2Up && finger4Up && !finger1Up && !finger3Up) {
108       lcd.clear();
109       lcd.print("Please turn off");
110       lcd.setCursor(0, 1);
111       lcd.print("the light.");
112       myDFPlayer.play(5);
```

```
124         lcd.clear();
125         lcd.print("Track 8 Playing");
126         myDFPlayer.play(8);
127       }
128     } else if (z<neg) {
129       if (finger1Up && finger3Up && finger4Up && !finger2Up) {
130         lcd.clear();
131         lcd.print("Track 9 Playing");
132         myDFPlayer.play(9);
133       } else if (finger2Up && finger3Up && finger4Up && !finger1Up) {
134         lcd.clear();
135         lcd.print("Track 10 Playing");
136         myDFPlayer.play(10);
137       }
138     } else if (z>pos) {
139       if (finger1Up && finger2Up && finger3Up && finger4Up) {
140         lcd.clear();
141         lcd.print("Track 11 Playing");
142         myDFPlayer.play(11);
143       } else {
144         lcd.clear();
145         lcd.print("Please give me");
146         lcd.setCursor(0, 1);
147         lcd.print("food.");
148         myDFPlayer.play(12);
149       }
150     } else {
151       lcd.clear();
152       lcd.print("Thank you.");
153       myDFPlayer.play(9);
154     }
155 
156     delay(3000); // Adjust delay as needed for responsiveness
157   }
```

Chapter 5

Results

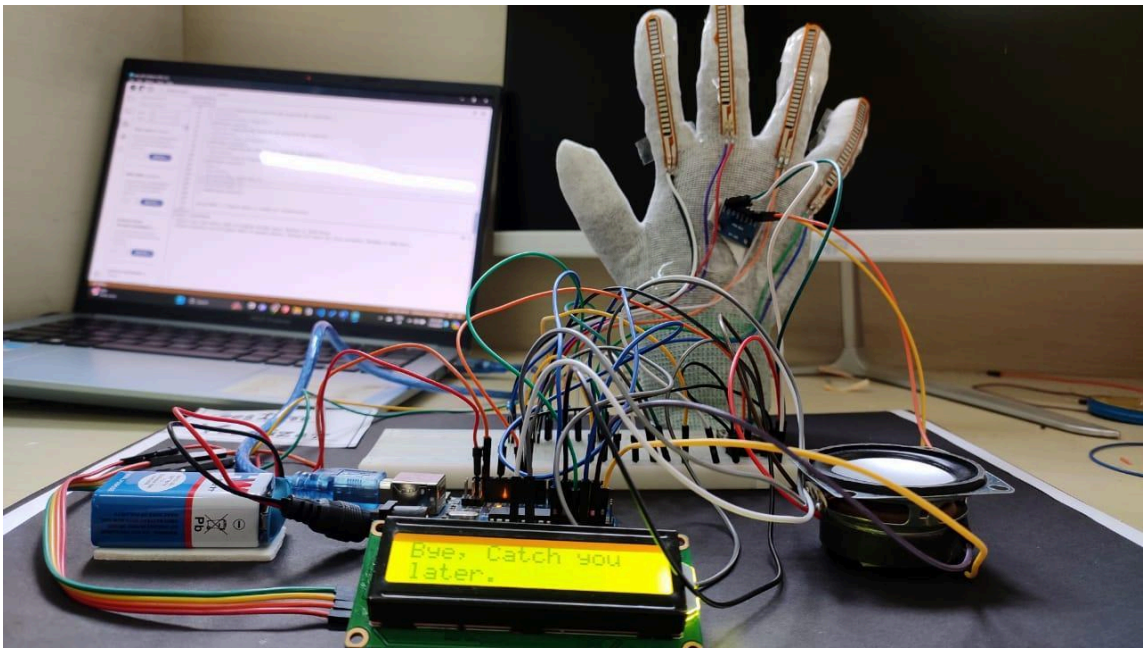

Fig. 16: Output on resting hand

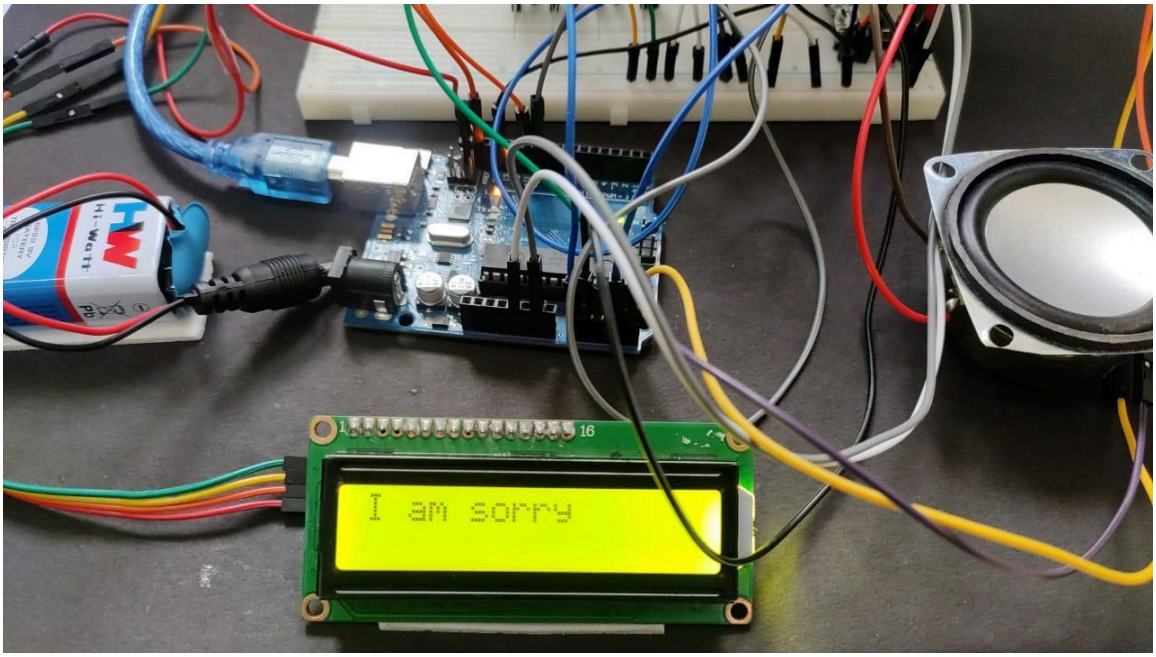

Fig. 17: Output 2

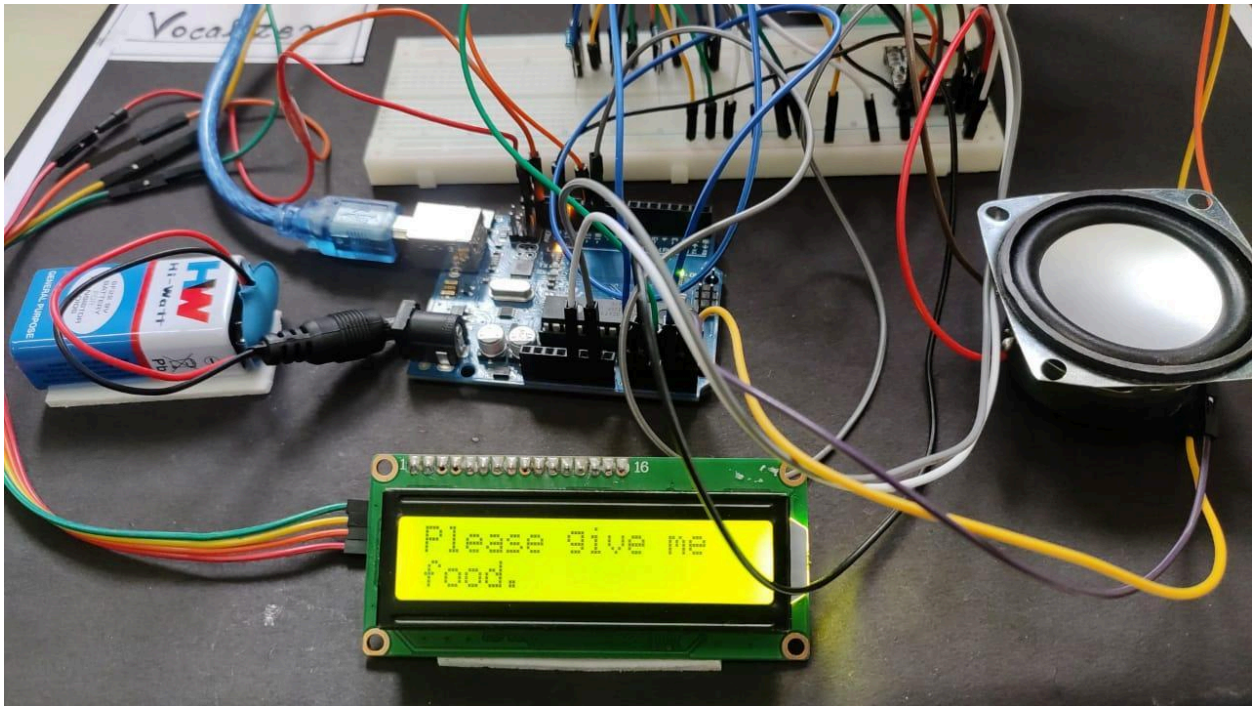

Fig. 18: Output 3

Inference from results:

In a hand vocalizer project, the integration of flex sensors, accelerometers, and a microcontroller allows accurate conversion of hand gestures into audible speech. The system effectively maps finger movements and hand orientations to predefined gestures, translating them into text and speech, enhancing communication for the vocally impaired.

Chapter 6
Conclusion and Future Work

Conclusion:

In conclusion, the Hand Gesture vocalizer project represents a significant achievement in the field of interactive technology, successfully integrating hardware components and software algorithms to enable intuitive gesture-based control of audio outputs. Leveraging an Arduino Uno microcontroller as the central processing unit, the system effectively utilizes four flex sensors to detect variations in finger bends and an ADXL345 accelerometer to capture hand tilt movements. These sensory inputs are processed in real-time to interpret specific hand gestures, which are then mapped to corresponding audio tracks through the DFPlayer Mini module. The inclusion of an LCD display provides immediate visual feedback, enhancing user interaction and usability.

Throughout the development process, significant challenges were overcome, particularly in sensor calibration and algorithm refinement to ensure accurate gesture recognition. Rigorous testing and iterative improvements were instrumental in achieving a system that reliably detects and responds to a variety of user gestures. This project not only demonstrates technical proficiency in hardware integration and software development but also underscores the potential of gesture-controlled interfaces in enhancing user experience across diverse applications.

The successful implementation of the Hand Gesture vocalizer opens doors to numerous practical applications. For instance, in educational settings, it could serve as an engaging tool for teaching concepts of technology and interactive systems. In healthcare, it holds promise as an assistive technology for individuals with mobility impairments, offering them a means to control audio devices hands-free. Furthermore, in entertainment and gaming industries, its intuitive control mechanism could enrich user immersion and interaction in virtual environments.

In summary, the Hand Gesture vocalizer project achieves its objectives of demonstrating a functional prototype capable of interpreting hand gestures for audio control. It highlights the

feasibility and potential of gesture-based interfaces in various domains, paving the way for future innovations in interactive technology.

Future Work:

Looking forward, several avenues for further development and enhancement of the Hand Gesture vocalizer project emerge, aimed at refining its functionality, expanding its capabilities, and exploring new applications. One key area for improvement is the enhancement of gesture recognition algorithms. Future iterations could incorporate machine learning techniques to adaptively learn and recognize a broader range of hand gestures with higher accuracy and robustness. This would involve collecting and analyzing extensive datasets to train the system to differentiate between subtle variations in hand movements.

Additionally, the integration of additional sensors could enrich the system's capability to interpret complex gestures. Sensors such as gyroscopes for rotational data or pressure sensors for grip strength could provide supplementary inputs, enabling more nuanced control and interaction possibilities. Moreover, exploring advanced signal processing techniques could further optimize sensor data interpretation and enhance the system's responsiveness in real-world environments.

User interface enhancements represent another area of potential advancement. Integrating touchscreen interfaces or voice command capabilities could offer alternative interaction modalities, catering to diverse user preferences and accessibility needs. Furthermore, investigating wireless connectivity options, such as Bluetooth or Wi-Fi, would enable remote control and data transmission capabilities, facilitating seamless integration with other smart devices and IoT ecosystems.

Miniaturization and wearable design are also promising directions for future development. Transforming the Hand Gesture vocalizer into a compact, wearable device—perhaps integrated into gloves or wristbands—would enhance mobility and practicality, making it suitable for everyday use in various contexts, including healthcare and interactive entertainment.

Conducting comprehensive user testing and gathering feedback from target users would be essential for iteratively refining the system's design and functionality. This iterative approach would ensure that the Hand Gesture vocalizer meets user expectations in terms of usability, reliability, and performance across different use-case scenarios.

In conclusion, by pursuing these avenues for future work, the Hand Gesture vocalizer project can evolve into a more sophisticated and versatile technology, capable of making meaningful impacts in fields ranging from assistive technology and education to entertainment and beyond. Continued innovation and collaboration across disciplines will be key to realizing the full potential of gesture-based interfaces in enhancing human-computer interaction and user experience.

Bibliography

[1]: Nath, D. (2018). A Review on Gesture Vocalizer. *International Journal for Research in Applied Science and Engineering Technology*, 6, 2990-2994. https://doi.org/10.22214/IJRASET.2018.4498.

[2]: (2019). Finger Gesture Vocalizer. International Journal of Engineering and Advanced Technology. https://doi.org/10.35940/ijeat.b2554.129219.

[3]: Kumar, K. (2022). Gesture Vocalizer Using Flex Sensor and Software Visualization. International Journal for Research in Applied Science and Engineering Technology. https://doi.org/10.22214/ijraset.2022.44296.

[4]: Sinthu, K., Kavitha, G., & , R. (2018). An Enhanced Gesture Vocaliser for the Vocally Challenged People. International journal of engineering research and technology, 2.

[5]: Haridas, D., Johnson, R., Simon, R., Mohan, S., & Babu, L. (2023). Gesture Vocalizer. International Research Journal of Modernization in Engineering Technology and Science. https://doi.org/10.56726/irjmets42684.

[6]: Ail, S., Chauhan, B., Dabhi, H., Darji, V., & Bandi, Y. (2019). Hand Gesture-Based Vocalizer for the Speech Impaired. , 585-592. https://doi.org/10.1007/978-981-15-1002-1_59.

[7]: Rishitha, C., Bhavana, M., Sriya, P., & Sreenivasulu, D. (2022). Gesture Vocalizer Using Flex Sensor and Software Visualization. *International Journal for Research in Applied Science and Engineering Technology*. https://doi.org/10.22214/ijraset.2022.43737.

[8]: Saha, P., Mohapatra, D., & Fels, S. (2021). SPEAK WITH YOUR HANDS - Using Continuous Hand Gestures to control Articulatory Speech Synthesizer. ArXiv, abs/2102.01640.

[9]: SparkFun. "Flex Sensor Hookup Guide." SparkFun Electronics, https://learn.sparkfun.com/tutorials/flex-sensor-hookup-guide.

[10]: Wiki. “DF Player mini.” Wiki dfrobots, https://wiki.dfrobot.com/DFPlayer_Mini_SKU_DFR0299

[11]: Sunrobotics. “I2C Interface Module for LCD(16x2 , 20x4)”, https://www.sunrobotics.in/shop/7687-i2c-interface-module-for-lcd-16x2-20x4-541#attr=

[12]: Analog. “ADXL345”, https://www.analog.com/en/products/adxl345.