Project Report

Smart Glove

Supervised By: Dr. Junaid Ahmed

Engr. Asif Ali

Submitted By:

Muhammad Faizan

Abdul Rehman

Muhammad Zuhaib

ABSTRACT

People interact to share their thoughts, ideas, and experiences with others. However, this form of communication is often challenging for deaf and mute individuals. Sign language has become a vital tool for enabling communication within the deaf community, allowing them to express themselves without the need for spoken words. This study focuses on developing a sign language recognition system to bridge the communication gap between individuals with speech impairments and the broader public. The system utilizes hand gestures, which are a faster and more effective means of expression compared to other gestures such as those involving the arms, face, head, or body. A flex sensor-based gesture recognition module was designed to identify letters and certain words, displaying the recognized words on a mobile application.

INTRODUCTION

Communication between the speech-impaired community and the public has always been a challenge. While sign language provides a means of communication, it requires widespread awareness and learning. Technology offers a solution to bridge this gap, empowering individuals with speech impairments to engage more effectively in society. This project aims to develop a smart glove that translates hand gestures into speech and text, promoting inclusivity and confidence. The glove recognizes gestures corresponding to sign language alphabets and converts them into a form understandable by anyone, addressing a significant real-world communication barrier. Sign language, a visual method of communication using gestures, body language, and hand movements, is essential for the Deaf and Hard of Hearing communities, as well as individuals with conditions like Autism, Apraxia, Cerebral Palsy, and Down Syndrome. Similar to spoken languages, sign languages vary regionally, with an estimated 138 to 300 variations worldwide, including American Sign Language (ASL), British Sign Language (BSL), and Australian Sign Language (Auslan). ASL, widely used and chosen for this project, employs one-handed signs with its own grammar and structure. Additionally, aspects of Chinese Sign Language (CSL), which visually represents written Chinese characters, are integrated into the system. By combining gestures from ASL and CSL, the glove ensures accurate and adaptable recognition, providing an efficient, user-friendly tool to bridge the communication gap and create a more inclusive environment.

TECHNOLOGY FOR CAPTURING GESTURE

Mechanics of the Sign Language:

American Sign Language (ASL) relies on unique hand movements to convey information. To recognize these gestures, technology capable of tracking hand orientations and movements is essential. We have developed a hybrid system powered by a microcontroller that uses flex sensors to detect hand gestures. These wearable sensors capture finger and hand movements, transmitting data that is processed into text via a mobile application. The hardware collects sensor readings, which are analyzed by software to match them with predefined values, mapping the gestures to specific alphabets. Flex sensor outputs vary based on the bending angles of each finger, allowing precise identification of different angle combinations for each alphabet. Furthermore, the K-Nearest Neighbors (KNN) model enhances the system's accuracy in distinguishing between sign language alphabets, ensuring reliable recognition.

Figure 1 Gesture Implemented

Communication between hearing and non-hearing:

Gestures serve as a primary mode of communication between hearing and non-hearing individuals, making sign language interpretation an invaluable tool for integrating deaf and mute individuals into society without barriers. This paper presents a smart system featuring a wearable glove capable of detecting hand gestures, transmitting signals, and converting them into text. The system collects sensor data, processes it through software to identify individual words, and transmits the results via USB-connector to a mobile application, which further converts the recognized word into text output.

DESIGN SELECTION

We carefully evaluated and selected the following components to ensure the system functions effectively and efficiently:

1. Flex Sensors

A **flex sensor** is a low-cost, easy-to-use variable resistor that is designed to measure the amount of deflection it experiences when bent. The sensor's resistance is lowest when it's flat on the surface, increases when we bend it slowly and reaches its maximum when it's at a 90-degree angle. Flex sensors are popular because they are used in many different applications like game controllers, data gloves, motion trackers, and even in biomedical devices to register static and dynamic postures. So, in today's project, we will learn all about flex sensors, how it works, and how you can interface them with an Arduino.

1.1. Flex Sensor Pinout

a) The flex sensor, also sometimes called as **bend sensor**, has two pins, one is P1 and the other one is P2, these two pins can be used to retrieve data from the flex sensor. The sensor acts more like a variable resistor, whose resistance changes based on how much it is bent, hence just like a resistor, the pins on this sensor are also interchangeable. The **pinout of Flex Sensor** is given below:

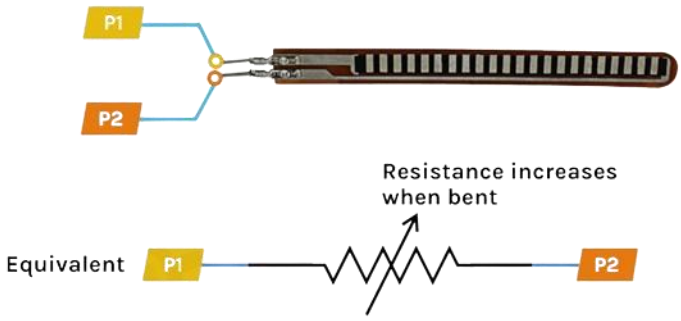

Figure 2 Flex Sensor Pinout Diagram

Resistors have two leads, there is no polarity for a resistor and hence can be connected in both directions. Since the flex sensor acts as a resistor, pins P1 and P2 are the same.

1.2. Flex Sensor Working

a) In simple terms, a flex sensor is a variable resistor that varies its resistance upon bending. As the resistance of the sensor is directly proportional to the amount of bending, it's often called **Flexible potentiometer**. This sensor is commonly available in two sizes, the first is 2.2" and the second one is 4.5" long. When the sensor is straight the resistance is about 10K, and when the sensor is bent the value is 22K.

When the sensor is bent the conductive layer inside the sensor gets stretched, the cross-section of the sensor gets thinned, and the resistance of the sensor increases. For example, shown above, the max resistance we

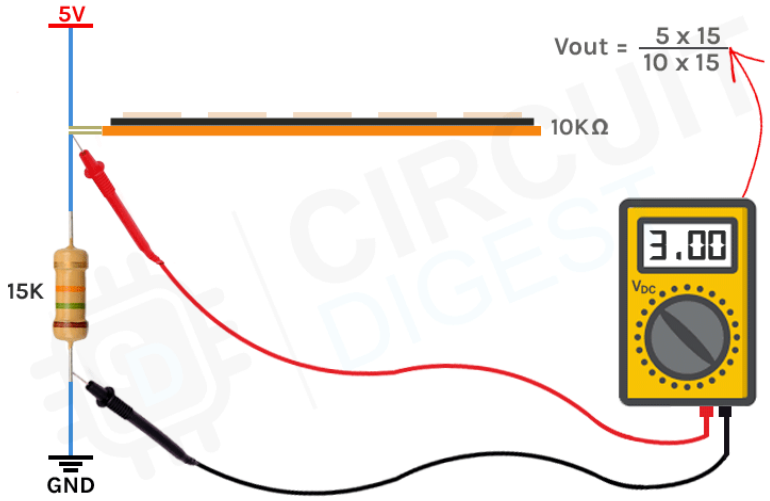

Figure 3 Flex Sensor Simulation

can achieve is 22K. When the sensor is straightened again, the resistance returns to its normal value.

1.3. Flex Sensor - Parts

a) The construction of a Flex sensor consists of different materials. The sensor we are using is made of conductive ink protected with Phenol Formaldehyde Resin substance. A segmented conductor is placed on top of this construction in which resistance changes upon deflection which is why it can be produced in a great form factor in a thin flexible substrate. When the substrate is bent, the sensor produces a resistance output correlated to the bend radius - the smaller the radius, the higher the resistance value.

The flex sensors are designed to be bent in only one direction, as shown in the figure below, bending the sensor in the opposite direction can inflict permanent damage to the sensor. Also, you need to be careful with the bottom of the sensor where the pins of the sensor are crimped because that part of the sensor is very fragile and can break easily when it's bent over.

2. Arduino Uno

- Acts as the central processing unit (CPU) of the system.
- Reads and processes data from flex sensors.
- Sends the processed data to a mobile application for conversion into text and speech.

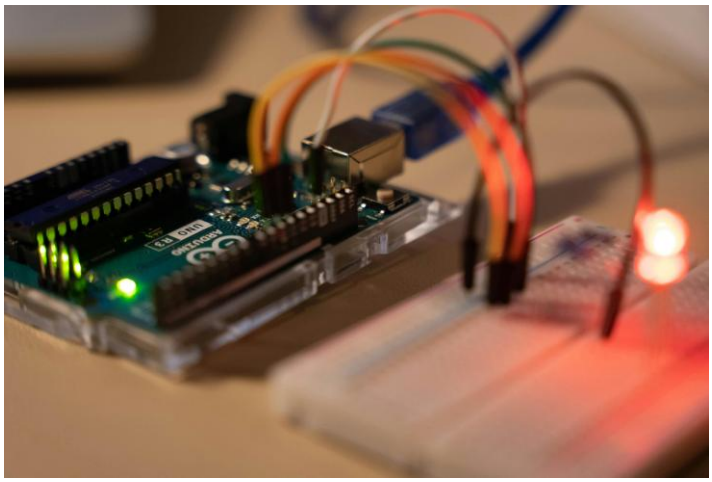

Figure 4 Arduino Uno

3. Resistors

- Used to regulate voltage and current within the circuit.
- Ensure stable operation of the sensors by preventing signal distortion and protecting components from damage.

These components work together to form a reliable and precise sign language recognition system.

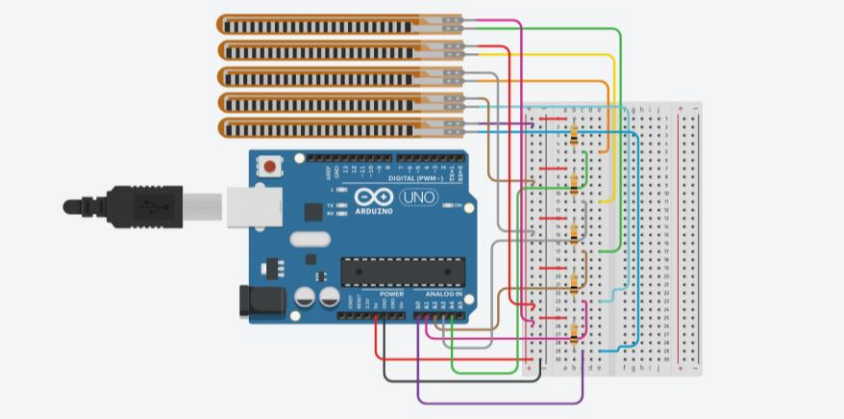

Figure 5 Schematic Diagram

WORKING PRINCIPLE OF SMART GLOVE

The operation of the speaking glove follows these steps:

- Initially, five flex sensors are attached to each finger and connected to a microcontroller (Arduino Uno) to detect finger movements. The flex sensors are part of a voltage divider circuit, so when the finger bends, the resistance of the corresponding sensor changes. This results in a change in the voltage drop across the sensor.

- The sensor readings are sent to the Arduino Uno, which processes these values and compares them using the K-Nearest Neighbors (KNN) model to find the closest match for a specific gesture.

- The microcontroller analyzes the data from the sensors to identify the gesture being made.

- Once the gesture is identified, the corresponding sign is transmitted to the user's mobile phone via a USB connection.

- The mobile phone receives the sign language data from the microcontroller and, through a mobile application, displays the output as text.

This system works by accurately detecting hand gestures through sensor data, processing the information, and delivering the recognized sign language output on a mobile app in text form.

FLOW CHART OF SMART GLOVE

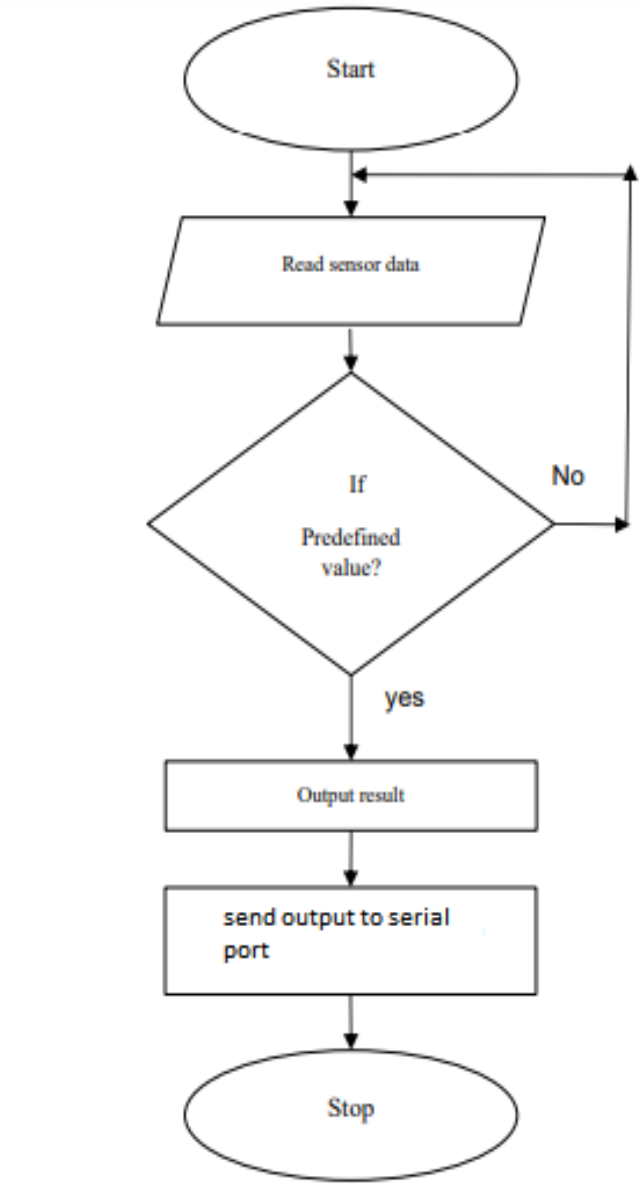

fig shows: Flow Chart of Smart Glove

The flowchart illustrates the operational process of a **Smart Glove System** designed to interpret sensor data and provide corresponding outputs. The steps are as follows:

1. **Start**
   The process begins.
2. **Read Sensor Data**
   The glove continuously reads data from its sensors (e.g., flex sensors detecting finger positions).
3. **Check for Predefined Values**

   - The system checks if the sensor data matches any predefined values that correspond to specific gestures or actions.

4. **Decision (Predefined Value?)**

   - **No:** If the data does not match any predefined value, the system loops back to **Read Sensor Data** and continues monitoring.
   - **Yes:** If a match is found, the flow proceeds to generate an output.

5. **Output Result**
   The system processes the recognized gesture and generates the corresponding result.
6. **Send Output to Serial Port**
   The output is transmitted to the serial port for further use, such as displaying the result or interfacing with other devices.
7. **Stop**

Key Highlights:

- The system operates in a continuous loop, ensuring real-time gesture recognition.
- It efficiently checks for predefined values and only proceeds when valid input is detected.
- Outputs are transmitted via a serial port, enabling communication with external devices for further applications (e.g., display systems or software).

This approach makes the Smart Glove a **dynamic and responsive solution** for translating gestures into meaningful outputs.

RESULT ANALYSIS

After completing the design and development of the glove, we conducted tests to ensure it met our initial design specifications and development goals. The primary specifications included ease of use, portability, affordability, reliability, and aesthetics, all of which were successfully addressed during the construction of the glove.

- **Ease of Use:**
  Ease of use is a fundamental feature, ensuring users can operate the glove without difficulty. We designed the glove to be intuitive, allowing users to begin translating gestures quickly and without delay. The glove is soft, comfortable, and user-friendly, ensuring a seamless experience for the wearer.

- **Portability:**
  Portability was a key consideration from the outset. The glove is lightweight and compact, making it easy to carry and use in real-life situations. The flex sensors and components are thin and flexible, almost as slim as paper, contributing to its portability. This compact design allows users to easily store and transport the glove, ensuring it is practical for everyday use.

- **Reliability:**
  Reliability was one of the most crucial design factors for this glove. It was not only important for the glove to provide accurate translations but also to do so consistently. We focused on making the glove as reliable as possible to ensure it performs correctly and consistently in recognizing gestures and translating them into text.

- **Aesthetics:**
  If this glove were to be marketed, aesthetics would play an important role in its appeal. The design was crafted to be visually appealing, resembling a standard glove to make it more attractive to users. Additionally, Wearability is a key part of the aesthetics, ensuring that the glove does not restrict movement during extended use. The glove is lightweight and comfortable, making it suitable for long periods of wear without discomfort, which is essential for user satisfaction.

This comprehensive evaluation confirmed that the glove meets all the necessary specifications, making it a functional, reliable, and user-friendly tool for sign language recognition as shown in the given figure.

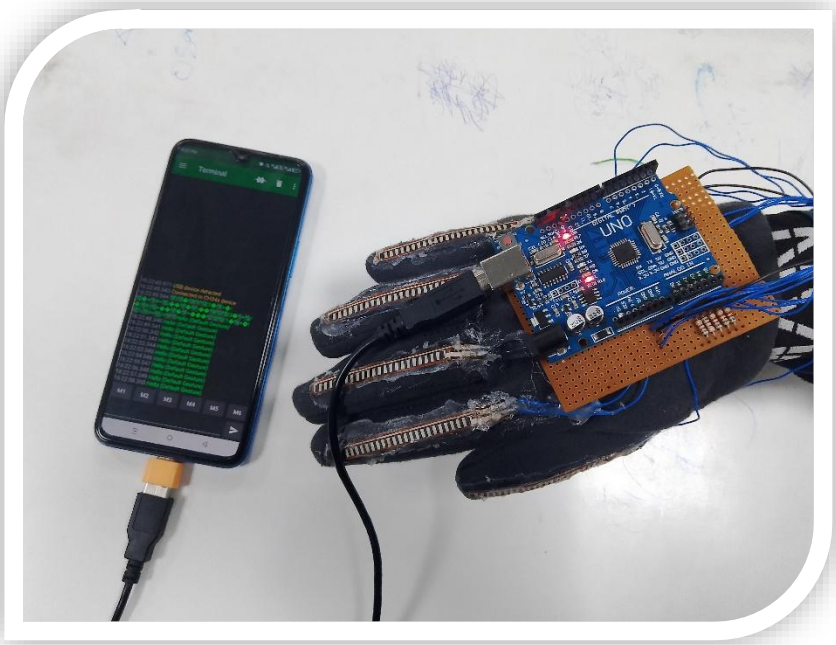

Figure 6:Final Product

Gestures Explanation with Results

Food Gesture

Figure 7 shows Food Gesture

- **Glove Gesture:** The glove shown is equipped with flex sensors and wires connected to an Arduino board. The flex sensors detect the bending of fingers when a gesture is made. When the user makes the food gesture, the sensors send values to the Arduino, which processes the input.
- **Result:** The display on the screen repeatedly shows "Detected: Food". This means the glove successfully identified the gesture based on the sensor values and recognized it as "Food." This process involves mapping the sensor readings for specific finger positions to predefined gestures.

In simple terms, when the user wears the glove and makes the food gesture, the sensors measure how much the fingers are bent, send the data to the Arduino, and display the recognized word "Food" on the screen.

Nice Gesture

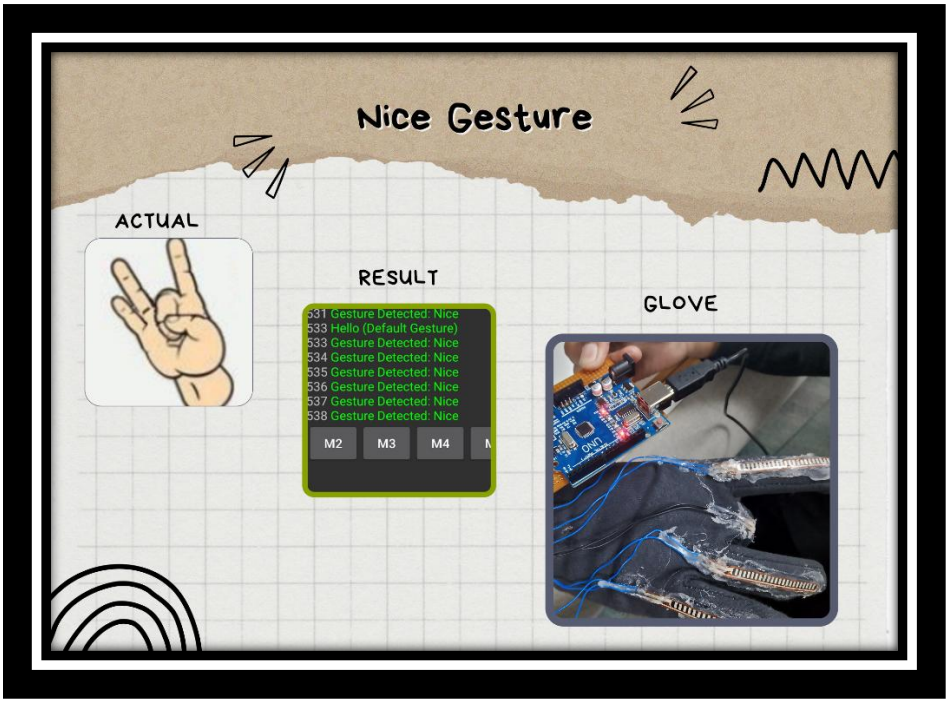

Figure 8 Nice Gesture

- **Actual Gesture:** The "Nice Gesture" resembles a rock-and-roll hand sign where the index finger and pinky finger are extended while the rest are bent. This is the target gesture for the glove to recognize.
- **Glove Gesture:** The glove detects the position of each finger through its flex sensors. The extended index and pinky fingers produce different sensor values compared to the bent middle, ring, and thumb fingers. The Arduino processes these sensor readings and matches them to the predefined "Nice Gesture."
- **Result:** The display shows "Gesture Detected: Nice" multiple times. This confirms that the glove and gesture recognition system successfully identified the correct gesture.

In simple terms, when the user makes the "Nice Gesture," the glove's sensors measure which fingers are bent or straight. The Arduino matches this data with the gesture for "Nice" and displays the result.

How Gesture

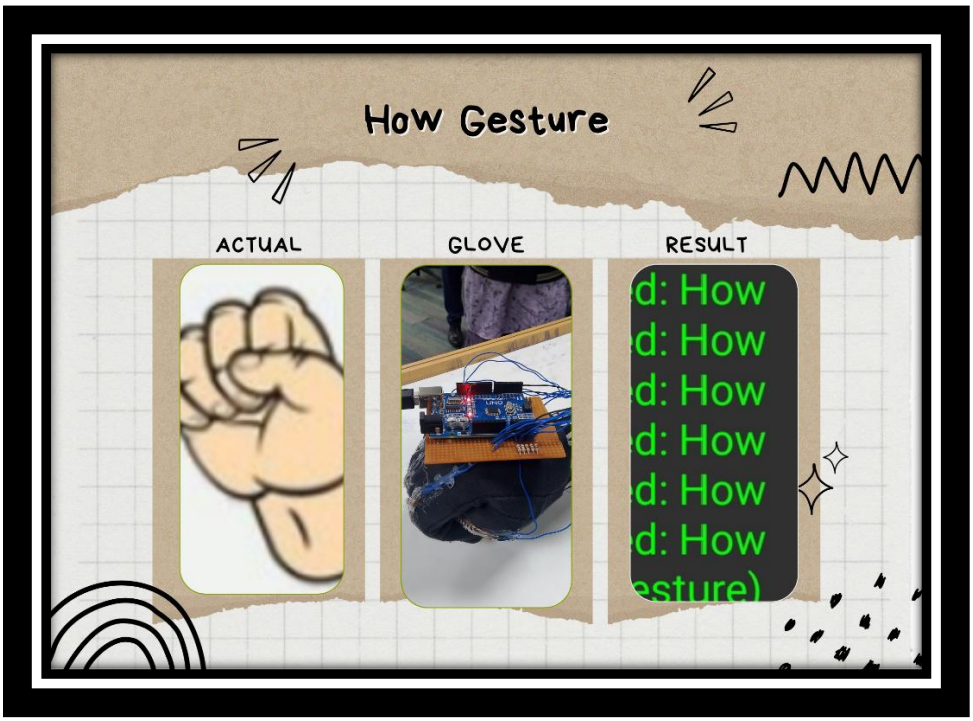

Figure 9 How Gesture

- **Actual Gesture:** The "How Gesture" is shown as a closed fist, where all fingers are bent into the palm. This is the gesture the glove is detecting.
- **Glove Gesture:** The glove, again equipped with flex sensors, registers the bending of all fingers. The Arduino board processes this sensor data and determines that the "How Gesture" is being made. This involves comparing the sensor values to the predefined threshold for a closed fist.
- **Result:** On the screen, the text "Detected: How" appears multiple times. This confirms that the gesture recognition system successfully detected the "How Gesture" based on the bending data of the flex sensors.

In simple terms, when the user makes a fist while wearing the glove, the system matches this with the gesture for "How" and shows the result on the display.

COST ANALYSIS OF SMART GLOVE

Component	Quantity	Cost per Unit (PKR)	Total Cost (PKR)
Arduino Uno	1	2000	2000
Flex Sensors	5	2200	11000
Resistors (8.2kΩ)	5	20	100
Glove	1	250	250
Wires	20	10	200
Miscellaneous		500	500
Total	-	-	14050

IMPACT OF SMART GLOVE ON SOCIETY

The Smart Gloves project has far-reaching implications:

1. **Bridges Communication Gaps**
    - Converts sign language into text and speech, enabling Deaf and mute individuals to communicate seamlessly with the hearing community.
2. **Promotes Inclusivity and Independence**
    - Empowers users to interact confidently in social, educational, and professional settings, reducing dependency on interpreters.
3. **Improves Accessibility**
    - Provides a portable and user-friendly tool for daily communication, enhancing the quality of life for individuals with hearing and speech impairments.
4. **Reduces Social Isolation**
    - Encourages greater social interaction and participation, helping users feel more connected to society.
5. **Supports Technological Advancement**
    - Allows for future improvements, including the recognition of complete sentences and expansion of the dataset, increasing the system's accuracy and versatility.
6. **Economic Benefits**
    - Creates opportunities in assistive technology and wearable tech industries, potentially driving job creation and market growth.
7. **Fosters Integration and Equality**
    - Contributes to a more inclusive society, promoting communication access and equality for individuals with speech impairments.

CONCLUSION

Our primary goal was to develop a functional model capable of translating basic gestures, and we have successfully built the necessary hardware for this project. While the current prototype is not yet perfect, it demonstrates the potential for using a glove equipped with sensors, a microcontroller, and wired communication to translate ASL signs. There is still room for improvement in areas such as accuracy, design, and other key specifications. We are encouraged by the progress made so far, and this prototype lays the foundation for future advancements in sign language translation technology. With continued focus on improving ASL translation, we hope to help bridge the communication gap between ASL users and the hearing community.

FUTURE SCOPE

Once the basic model is complete, the next step would be to extend its functionality to recognize complete sentences by translating various signs through wireless communication via a Bluetooth module. This would enable the glove to be used in everyday situations. Additionally, expanding the dataset used by the model would enhance its accuracy and range. By researching and developing a larger dataset, we can improve the system’s ability to recognize a broader range of signs and gestures, making it even more effective and versatile.

Code Detail

Python Code for Collecting data Samples

```
import serial


# ******************** Configure the serial connection *********************
serial_port = 'COM8'              # Replace with your serial port (e.g., COM3, /dev/ttyUSB0)
baud_rate = 9600                  # Must match the Arduino's Serial.begin rate
output_file = "Yes.txt"           # Output file to save data


try:
    # ***************** Open serial connection and initialize file ***************
    with serial.Serial(serial_port, baud_rate, timeout=1) as ser:
        print("Connected to Arduino. Waiting for data...")


        with open(output_file, 'w') as file:
            while True:
                # ****************** Read data from the serial port *****************
                raw_line = ser.readline()


                # ***************** Decode the raw bytes safely ********************
                try:
                    line = raw_line.decode('utf-8').strip()       # Attempt to decode as UTF-8
```

```
                if line:                        # Ensure the line is not empty
                    print(line)                 # Display in console
                    file.write(line + '\n')     # Save to file
            except UnicodeDecodeError as e:
                print(f"Decode error: {e}. Skipping this line...")


except serial.SerialException as e:
    # ********** Handle serial connection errors **********
    print(f"An error occurred: {e}")


finally:
    # ********** Close the serial connection **********
    print("Serial connection closed.")
```

MATLAB Code for Analyzing data Samples by Using KKN Model

```
# ******************** Load Data from Text File **********************
filename = 'Meds.txt'                          # Replace with your file name
try:
    with open(filename, 'r') as file:
        file_data = file.read()
except FileNotFoundError:
    raise FileNotFoundError(f'File "{filename}" does not exist. Please check the file name and path.')


# ***************** Parse the data into lines **********************
data_lines = [line.strip() for line in file_data.splitlines() if line.strip()]
num_samples = len(data_lines)


# ***************** Detect sensor labels from the first line **********************
header_tokens = re.findall(r'\w+:', data_lines[0])
if not header_tokens:
    raise ValueError('No sensor labels found in the first line. Please check the data format.')


sensor_labels = [label.rstrip(':') for label in header_tokens]
num_sensors = len(sensor_labels)
```

```
# ********** Preallocate matrix for storing sensor values **********
samples = np.zeros((num_samples, num_sensors))

# ********** Process each line to extract sensor values **********
for i, line in enumerate(data_lines):
    tokens = re.findall(r'(?:' + '|'.join(sensor_labels) + r'):\s*(-?\d+)', line)
    if len(tokens) == num_sensors:
        samples[i, :] = list(map(float, tokens))
    else:
        print(f'Warning: Line {i + 1} does not match the expected format and will be skipped: {line}')

# ********** Remove rows with all zeros (invalid lines) **********
samples = samples[~np.all(samples == 0, axis=1)]

# ********** Remove highly deviated values **********
mean_vals = np.nanmean(samples, axis=0)       # Calculate mean for each sensor column ignoring NaNs
std_vals = np.nanstd(samples, axis=0)              # Calculate standard deviation ignoring NaNs
dynamic_threshold = 2 * std_vals                  # Set dynamic threshold as twice the standard deviation

        # Identify and filter out rows with highly deviated values
is_within_threshold = np.all(np.abs(samples - mean_vals) <= dynamic_threshold, axis=1)
filtered_samples = samples[is_within_threshold]

# ********** Display results **********
```

```
print('Data after filtering highly deviated values:')
print(filtered_samples)

# ********** Calculate final mean and threshold values **********
final_mean = np.nanmean(filtered_samples, axis=0)          # Final mean ignoring NaNs
final_threshold = dynamic_threshold                        # Threshold remains the same after filtering

print('Final Mean Values for Each Sensor:')
print(pd.DataFrame([final_mean], columns=sensor_labels))

print('Dynamic Threshold Values for Each Sensor:')
print(pd.DataFrame([final_threshold], columns=sensor_labels))

# ********** Save mean and threshold values for further use **********
gesture_model = {
    'mean_vals': final_mean.tolist(),                      # Mean of filtered samples
    'threshold': final_threshold.tolist()                  # Threshold values
}
np.save('gestureModel.npy', gesture_model)          # Save the model parameters

print('Model saved successfully to "gestureModel.npy".')
```

Final Code of Smart Gloves

```
// ********** Arduino Gesture Recognition Based on Sensor Data **********

#include <math.h>


// ********** Gesture Data **********

const char* gestureNames[] = {

   "Cool", "Food", "Leave", "Help", "How", "Nice", "Hello (Default Gesture)"

};


const float meanVals[][5] = {

   {103.74, 103.74, 103.92, 95.154, 94.156},                    // Cool

   {107.93, 107.8, 107.97, 135.31, 127.1},                      // Food

   {70.824, 70.807, 71.055, 176.58, 89.71},                     // Leave

   {122.33, 122.32, 122.29, 115.2, 106.74},                     // Help

   {81.43, 81.376, 81.581, 106.06, 90.071},                     // How

   {105.92, 105.86, 105.86, 95.04, 157.01}                      // Nice

};


const float dynamicThreshold[][5] = {

   {3.3782, 3.295, 3.3491, 17.999, 11.401},

   {6.3609, 6.3732, 6.2853, 11.894, 7.1208},

   {4.7239, 4.7469, 4.752, 24.473, 19.264},

   {12.334, 12.314, 12.41, 15.986, 15.906},

   {5.8153, 5.9079, 6.0028, 13.539, 13.007},
```

```
    {6.9979, 6.9503, 6.9486, 25.558, 10.607}
};

const int numGestures = 6;                    // Number of gestures

// ******************** Sensor Pin Definitions ********************
const int thumbPin = A0;
const int indexPin = A1;
const int middlePin = A2;
const int ringPin = A3;
const int littlePin = A4;

void setup() {
    // ************* Initialize Serial Communication and Sensor Pins *********************
    Serial.begin(9600);
    pinMode(thumbPin, INPUT);
    pinMode(indexPin, INPUT);
    pinMode(middlePin, INPUT);
    pinMode(ringPin, INPUT);
    pinMode(littlePin, INPUT);
}

void loop() {
    // ******************** Read Sensor Values *********************
```

```
float sensorVals[5];
sensorVals[0] = analogRead(thumbPin);
sensorVals[1] = analogRead(indexPin);
sensorVals[2] = analogRead(middlePin);
sensorVals[3] = analogRead(ringPin);
sensorVals[4] = analogRead(littlePin);

Serial.print("Thumb: ");
Serial.print(sensorVals[0]);
Serial.print("\tIndex: ");
Serial.print(sensorVals[1]);
Serial.print("\tMiddle: ");
Serial.print(sensorVals[2]);
Serial.print("\tRing: ");
Serial.print(sensorVals[3]);
Serial.print("\tLittle: ");
Serial.println(sensorVals[4]);

// ******************** Determine the Matching Gesture ********************
int matchedGesture = -1;
for (int g = 0; g < numGestures; g++) {
  bool match = true;
  for (int s = 0; s < 5; s++) {
    if (abs(sensorVals[s] - meanVals[g][s]) > dynamicThreshold[g][s]) {
```

```
            match = false;
            break;
        }
    }
    if (match) {
        matchedGesture = g;
        break;
    }
  }

  // ******************** Print the Gesture Name **********************
  if (matchedGesture != -1) {
    Serial.print("Gesture Detected: ");
    Serial.println(gestureNames[matchedGesture]);
  } else {
    Serial.println("Hello (Default Gesture)");
  }

  delay(1000); // Wait for stability
}
```

References

[1] University of Washington, "UW undergraduate team wins $10,000 Lemelson-MIT Student Prize for gloves that translate sign language," *University of Washington News*, Apr. 12, 2016. [Online]. Available: https://www.washington.edu/news/2016/04/12/uw-undergraduate-team-wins-10000-lemelson-mit-student-prize-for-gloves-that-translate-sign-language/. [Accessed: Dec. 15, 2024].

[2] OpenAI, *ChatGPT Assistance*, conceptual and coding guidance provided during the initial stages of development, Dec. 2024.

[3] [Faizan, Zuhaib, Rehman], "Arduino-based gesture recognition system using flex sensors and dynamic thresholding," self-implemented project using sensor data processing, gesture recognition algorithms, and testing, Dec. 2024.