RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

EL OBSERVATION RECORD

Experiential Learning

For the III Semester B.E PROGRAMS (ACY 2024-25)

Team Name	Robo Squad
Theme	Cyber Physical Systems
Team ID	Team 7
Topic	Robotic Arm System for Real-Time Sign Language Translation
Team Leader Name	Joel Saha

Details of the Group Members

Sl. No.	Program	USN	Name	Email Id
1.	ECE	1RV23EC061	Joel Saha	joelsaha.ec23@rvce.edu.in
2.	ECE	1RV23EC055	Heeral Khare	heeralkhare.ec23@rvce.edu.in
3.	ETE	1RV23ETO30	Preetham V	preetham.et23@rvce.edu.in
4.	ETE	1RV23ETO46	Sneha Pandey	snehapandey.et23@rvce.edu.in

Student-1 Signature | Student-2 Signature | Student-3 Signature | Student-4 Signature

CONTENT

CHAPTER 1- INTRODUCTION

This project aims to bridge communication barriers between sign language users and non-signers by developing a Robotic Arm System for Real-Time Sign Language Translation. Utilizing natural language processing and robotic control, the system converts spoken or written language into accurate sign language gestures. This innovative solution has applications in education, public services, and more, promoting inclusivity and seamless communication.

PROBLEM STATEMENT

Communication barriers faced by individuals with hearing impairments often limit their participation in social, educational, and professional settings.
This project proposes a Robotic Arm System for Real-Time Text-to-Sign Language Translation to automatically perform sign gestures based on text input, providing an affordable and accessible solution to enhance inclusivity and bridge communication gaps.

OBJECTIVES

Develop Real-Time Translation: Create a system that translates spoken or written language into sign language gestures accurately and efficiently.

Design a Robotic Arm: Build and program a robotic arm capable of performing precise and fluid sign language gestures.

Integrate Advanced Technologies: Utilize programming languages, natural language processing, and advanced hardware control for gesture synthesis.

Enhance Accessibility: Ensure the system is user-friendly, cost-effective, and deployable in diverse environments.

Promote Inclusivity: Bridge the communication gap between sign language users and non-signers, fostering better understanding and interaction.

CHAPTER 2 LITERATURE SURVEY

1.An Adaptive, Affordable, Open-Source Robotic Hand for Deaf and Deaf-Blind Communication Using Tactile American Sign Language

Authors: *Samantha Johnson, Geng Gao, Todd Johnson, Minas Liarokapis, Chiara Bellini*

The paper titled "An Adaptive, Affordable, Open-Source Robotic Hand for Deaf and Deaf-Blind Communication Using Tactile American Sign Language" presents a novel solution to enhance communication for individuals who are deaf or deaf-blind. The authors introduce an open-source robotic hand capable of performing tactile American Sign Language (ASL), facilitating effective interaction for those relying on tactile signing methods.

By utilizing tactile ASL, the robotic hand provides a means for deaf-blind individuals to receive information through touch, bridging a critical communication gap. The paper details the design, development, and potential applications of this robotic hand, highlighting its significance in promoting inclusivity and accessibility for the deaf and deaf-blind communities.

This innovative approach demonstrates the potential of combining robotics with sign language to create assistive technologies that empower individuals with sensory impairments, fostering greater independence and social integration.

Link: https://ieeexplore.ieee.org/abstract/document/9629994

2. Sign Language Translation with Iterative Prototype

Authors: *Huijie Yao, Wengang Zhou, Hao Feng, Hezhen Hu, Hao Zhou1 Houqiang L*

The paper titled "Sign Language Translation with Iterative Prototype" introduces IP-SLT, a novel framework designed to enhance sign language translation (SLT) by iteratively refining semantic representations of input sign language videos.Traditional SLT systems often rely on a one-pass forward process, which may struggle to bridge the inherent gap between visual sign language inputs and textual translations. In contrast, IP-SLT employs a recurrent structure that mimics human reading behavior, where repeated digestion of content leads to better understanding.Utilizes a cross-attention mechanism to iteratively enhance the prototype by integrating it with the original video features. Through repeated refinement, the prototype converges to a more accurate and stable state, resulting in fluent and appropriate translations.

Link:https://openaccess.thecvf.com/content/ICCV2023/papers/Yao_Sign_Language_Translation_with_Iterative_Prototype_ICCV_2023_paper.pdf

3.Machine translation from text to sign language: A systematic review

Authors: *NavrozKaurKahlon1, WilliamjeetSingh*

The paper titled "Machine Translation from Text to Sign Language: A Systematic Review" provides a comprehensive analysis of existing methodologies and technologies in the field of text-to-sign language machine translation. The authors systematically survey both traditional and contemporary projects, highlighting the evolution and current state of sign language machine translation and generation systems.

The review delves into various approaches employed in the translation process, including rule-based, statistical, and neural network-based methods. It examines the challenges associated with each approach, such as the scarcity of parallel corpora, the complexity of sign language grammar, and the need for accurate facial expression and body movement representation.

Link:
https://www.researchgate.net/publication/352966000_Machine_translation_from_text_to_sign_language_a_systematic_review

4. American Sign Language Recognition System: An Optimal Approach

Authors: *Dr. Shivashankara SS*

The paper "American Sign Language Recognition System: An Optimal Approach" presents a method for translating 24 static American Sign Language (ASL) alphabets and numbers into machine-readable English text.

The proposed system operates in three phases known as Pre-processing, where the input gesture image undergoes pre-processing to enhance quality and prepare for analysis; Feature Extraction: where the system computes various region properties of the pre-processed gesture image, identifying key features essential for accurate recognition and Transliteration: based on the extracted features, the system translates the signed gesture into corresponding English text.

The authors conducted statistical evaluations comparing their method with existing techniques, demonstrating improved accuracy and efficiency in ASL gesture recognition.

This research contributes to the development of more effective communication tools for individuals with speech and hearing disabilities.

CHAPTER 3-DESIGN

Tools and Techniques Required

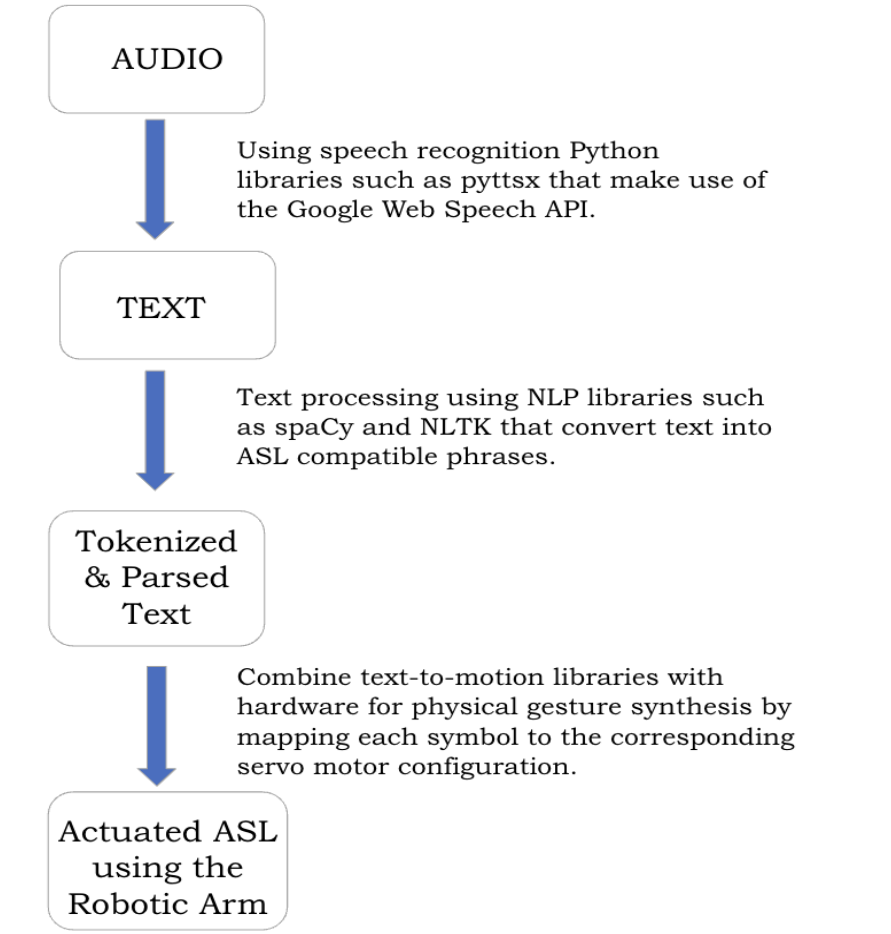
HARDWARE:

- Arduino UNO: to control robotic arm actuators
- SG90 Servo Motors: For precise control of joint movements
- Microphone: to record speech input
- Jumper wires
- Breadboard
- Elastic Wire, Fishing Line

SOFTWARE:

- Audio to Text Converter: To convert audio input to text which can be achieved using speech recognition libraries in python language
- Text Processing Software: To convert text into a sequence of sign language gestures using NLP and NLTK libraries.
- Programming Languages: Python for developing the control logic and translation algorithms.
- Motor Control Libraries: Libraries such as PWM libraries for controlling servo motors

METHODOLOGY



We created a code which successfully parses the sentences given to it, and gives us the words in such a way that it can be easily expressed through ASL by the robotic hand. We used NLTK and pyfirmata2 libraries in the code.

We designed the 3D model of the robotic hand on SolidWorks, and got it 3D printed, and we

used components such as servo motors, elastic wires, strings and Arduino UNO in the hand.

BASIC WORKING

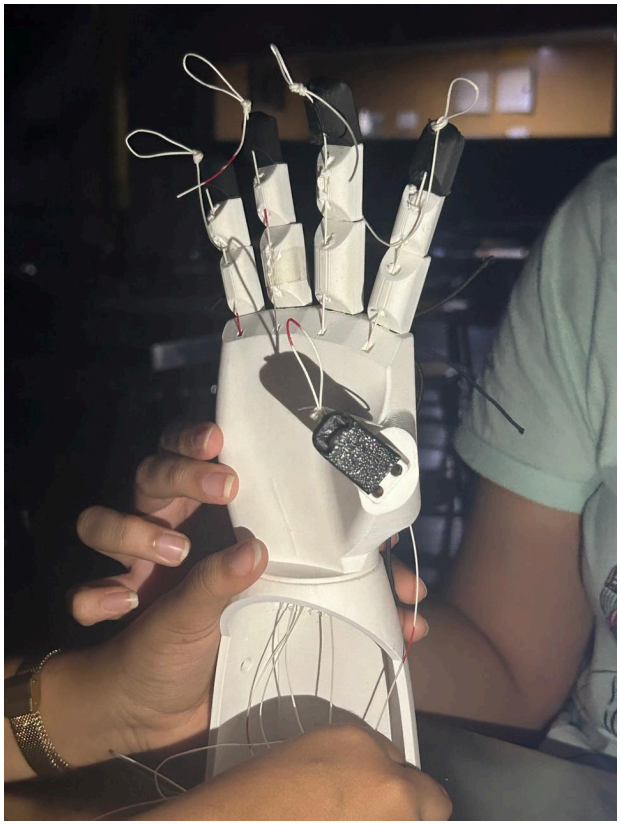
The hand is mainly manipulated with the help of a string, the string and elastic wire passes through the hand, the elastic wire helps in getting the hand back to its initial position. The servo motors are connected to the string,they are rotated according to the input provided, as they move they manipulate the fingers with the help of the string.

Type of servo motor used is **SG90 Micro Servo,** because**:**

- **Advantages**: Low cost, compact, lightweight.
- **Use Case**: Good for small and low-load joints.

The string used is a badminton racket string. The elastic wires are regular elastic wires.

PROGRESS PICTURES OF THE HAND



.

This is the hand after the string and elastic wires were attached to it. Here the servos are yet to be attached.

CODE FOR THE PROJECT

```python
from pyfirmata2 import Arduino
import time
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

import speech_recognition as sr

# Initialize the recognizer
r = sr.Recognizer()

# Set up the Arduino board
board = Arduino('COM3')   # Change 'COM3' to your Arduino port
# Define servo pins
index_servo = board.get_pin('d:11:s')
middle_servo = board.get_pin('d:10:s')
ring_servo = board.get_pin('d:9:s')
little_servo = board.get_pin('d:6:s')
thumb_servo = board.get_pin('d:5:s')

def record_text():
    # Loop in case of errors
    while True:
        try:
            # Use the microphone as source for input
            with sr.Microphone() as source2:
                # Prepare recognizer to receive input
                r.adjust_for_ambient_noise(source2, duration=0.2)
                # Listens for the user's input
                audio2 = r.listen(source2)
                # Using google to recognize audio
                MyText = r.recognize_google(audio2)
                return MyText
        except sr.RequestError as e:
            print("Could not request results; {0}".format(e))
        except sr.UnknownValueError:
            print("Unknown error occurred")
        return

def output_text(text):
    with open("output.txt", "a") as f:
        f.write(text)
        f.write("\n")

def process_text():
    text = record_text()

    if not isinstance(text, str) or not text.strip():
        print("Invalid or empty text. Skipping...")
        return []

    # Save the raw text
    output_text(text)
    print("Wrote text:", text)

    # Tokenize the text into words
    try:
        word_tokens = word_tokenize(text)
    except Exception as e:
        print(f"Error during tokenization: {e}")
        return []

    # Remove stopwords
    filtered_words = [word for word in word_tokens if word.lower() not in stop_words]

    # Lemmatize
    lemmatized_words = [lemmatizer.lemmatize(word) for word in filtered_words]
```

```
    # Process into ASL format (optional rearranging)
    processed_sentence = " ".join(lemmatized_words)

    # Extract individual letters and numbers
    letters_numbers = [char for char in processed_sentence if char.isalnum()]

    print("Original Text:", text)
    print("Filtered Words:", filtered_words)
    print("Lemmatized Words:", lemmatized_words)
    print("Processed Sentence:", processed_sentence)
    print("Letters and Numbers:", letters_numbers)

    return letters_numbers

def set_fingers(index=0, middle=0, ring=0, little=0, thumb=0):
    index_servo.write(index)
    middle_servo.write(middle)
    ring_servo.write(ring)
    little_servo.write(little)
    thumb_servo.write(thumb)
    time.sleep(0.5)

# MAIN LOOP
try:
    set_fingers()  # Initialize the servos with default positions
    print("Begin voice input")
    while True:
        letters_numbers = process_text()  # Get the processed letters and numbers from speech

        # Check if letters_numbers is empty, if so, skip the iteration
        if not letters_numbers:
            continue

        for i in letters_numbers:
            if i.isalpha():
                if i == 'a':
                    set_fingers(180, 180, 180, 180, 0)
                elif i == 'b':
                    set_fingers(0, 0, 0, 0, 180)
                elif i == 'c':
                    set_fingers(45, 45, 45, 45, 45)
                elif i == 'd':
                    set_fingers(180,135,135,135,180)
                elif i == 'e':
                    set_fingers(180, 180, 180, 180, 180)
                elif i == 'f':
                    set_fingers(180, 0, 0, 0, 180)
                elif i == 'g':
                    set_fingers(0, 90, 0, 90, 0)
                elif i == 'h':
                    set_fingers(90, 0, 90, 0, 0)
                elif i == 'i':
                    set_fingers(180,180,180,0,180)
                elif i == 'j':
                    set_fingers(180,180,180,0,180)
                elif i == 'k':
                    set_fingers(0, 0, 180, 180, 90)
                elif i == 'l':
                    set_fingers(0, 180, 180, 180, 0)
                elif i == 'm':
                    set_fingers(180, 150, 150, 180, 180)
                elif i == 'n':
                    set_fingers(180, 150, 180, 180, 180)
                elif i == 'o':
                    set_fingers(100,100,100,100,100)
                elif i == 'p':
                    set_fingers(0, 155, 155, 155, 20) #not done
                elif i == 'q':
                    set_fingers(45, 155, 155, 155, 20) #not done
                elif i == 'r':
                    set_fingers(25,0,180,180,180)
                elif i == 's':
                    set_fingers(180,180,180,180,180)
                elif i == 't':
```

```
                    set_fingers(160, 180, 180, 180, 180)
                elif i == 'u':
                    set_fingers(0, 0, 180, 180, 180)
                elif i == 'v':
                    set_fingers(0, 0, 180, 180, 180)
                elif i == 'w':
                    set_fingers(0, 0, 0, 180, 180)
                elif i == 'x':
                    set_fingers(90, 180, 180, 180, 180)
                elif i == 'y':
                    set_fingers(180, 180, 180, 0, 0)
                elif i == 'z':
                    set_fingers(0, 180, 180, 180, 180)

            elif i.isdigit():
                if i == '1':
                    set_fingers(0,180,180,180,180)
                elif i == '2':
                    set_fingers(0,0,180,180,180)
                elif i == '3':
                    set_fingers(0,0,180,180,0)
                elif i == '4':
                    set_fingers(0,0,0,0,180)
                elif i == '5':
                    set_fingers(0,0,0,0,0)
                elif i == '6':
                    set_fingers(0,0,0,180,180)
                elif i == '7':
                    set_fingers(0,0,180,0,180)
                elif i == '8':
                    set_fingers(0,180,0,0,180)
                elif i == '9':
                    set_fingers(180,0,0,0,180)
                elif i == '0':
                    set_fingers(90,90,90,90,90)
            elif i == ' ':
                set_fingers()  # Reset all fingers

```

```
except KeyboardInterrupt:
    print("Program stopped by user.")

finally:
    board.exit()  # Clean up and close the connection
```

EXPLANATION OF THE CODE

This Python script facilitates a real-time sign language translator using a robotic arm. The system captures voice input, processes the text into individual characters, and translates them into gestures using servo motor movements.

Libraries Used:

1. **`pyfirmata2`**
    - Used for interfacing with the Arduino board. It allows sending commands to control servos connected to the board.
    - Key function:
        - `Arduino()`: Initializes communication with the Arduino.
        - `get_pin()`: Specifies the pin mode (digital pin for servo in this case).

2. `time`

- Used to introduce delays between servo movements to ensure smooth operations

.

3. `nltk` (Natural Language Toolkit)

- Provides tools for processing text, including tokenization, stop word removal, and lemmatization.
- Key modules:
    - `stopwords`: Removes commonly used words (e.g., "the," "and") irrelevant to gesture recognition.
    - `word_tokenize()`: Breaks sentences into individual words or tokens.
    - `WordNetLemmatizer()`: Reduces words to their base form (e.g., "running" → "run").

`speech_recognition`

- Captures and converts speech into text using the Google Speech Recognition API.
- Key functions:
    - `Recognizer()`: Creates a recognizer instance.
    - `listen()`: Captures audio input from the microphone.
    - `recognize_google()`: Converts speech into text using Google's API.

Code Components

1. Voice Input and Processing

- Functionality: Captures and processes voice input into text.
- Steps:
    - Use the microphone to record speech (`sr.Microphone()`).
    - Adjust for ambient noise and capture audio input (`adjust_for_ambient_noise()` and `listen()`).
    - Convert speech to text using Google's Speech Recognition API (`recognize_google()`).

2. Text Processing

- Purpose: Prepare the text for sign language translation.
- Steps:
    - Tokenize the input text into words (`word_tokenize()`).

- Remove stop words like "is" and "the" (`stopwords.words('english')`).
- Lemmatize words to their base forms (`WordNetLemmatizer().lemmatize()`).
- Extract alphanumeric characters for gesture mapping.

3. Servo Motor Control

- Purpose: Translate processed text into gestures.
- Steps:
  - Set up servo motors using `pyfirmata2` and connect them to specific Arduino pins (`get_pin()`).
  - Define finger movements using degrees (0° to 180°) to represent each gesture.
  - Map characters (letters and numbers) to specific finger positions using `set_fingers()`.

4. Main Loop

- Functionality: Continuously listen for speech, process it, and execute corresponding gestures.
- Steps:
  - Initialize servos to default positions.
  - Capture and process voice input into individual characters.
  - For each character:
    - Identify if it's a letter, digit, or space.
    - Move servo motors to predefined positions based on the character.

CHAPTER 4 RESULTS & DISCUSSIONS

RESULTS

The results of this project show that the robotic hand is effective in converting text into American Sign Language (ASL) gestures. By using Python for processing the text, Arduino to control the hardware, and servo motors to move the fingers, the robotic hand successfully performs a variety of ASL signs. Each finger is moved precisely and smoothly, making the gestures clear and accurate.

The system works as expected: when text is entered, the python code quickly interprets it and the robotic hand translates it into the corresponding ASL signs. The servo motors control each finger individually, allowing the hand to replicate letters and numbers accurately. Elastic wires in the design help the fingers return to their starting positions after each gesture, making the hand ready for the next sign. This combination of smooth movements and precise control ensures the system can operate continuously without issues.

Overall, the results are promising. The robotic hand performs well with basic ASL gestures, proving that this approach is effective. Although the system is currently limited to a specific set of gestures, it shows strong potential for future improvements. By adding more signs, improving the accuracy of finger positions, and speeding up the response time, the robotic hand could become an even more useful tool for helping people communicate.

These results show that this project is a successful first step toward creating a reliable, practical device for bridging communication gaps between signers and non-signers.

Prototype Description:

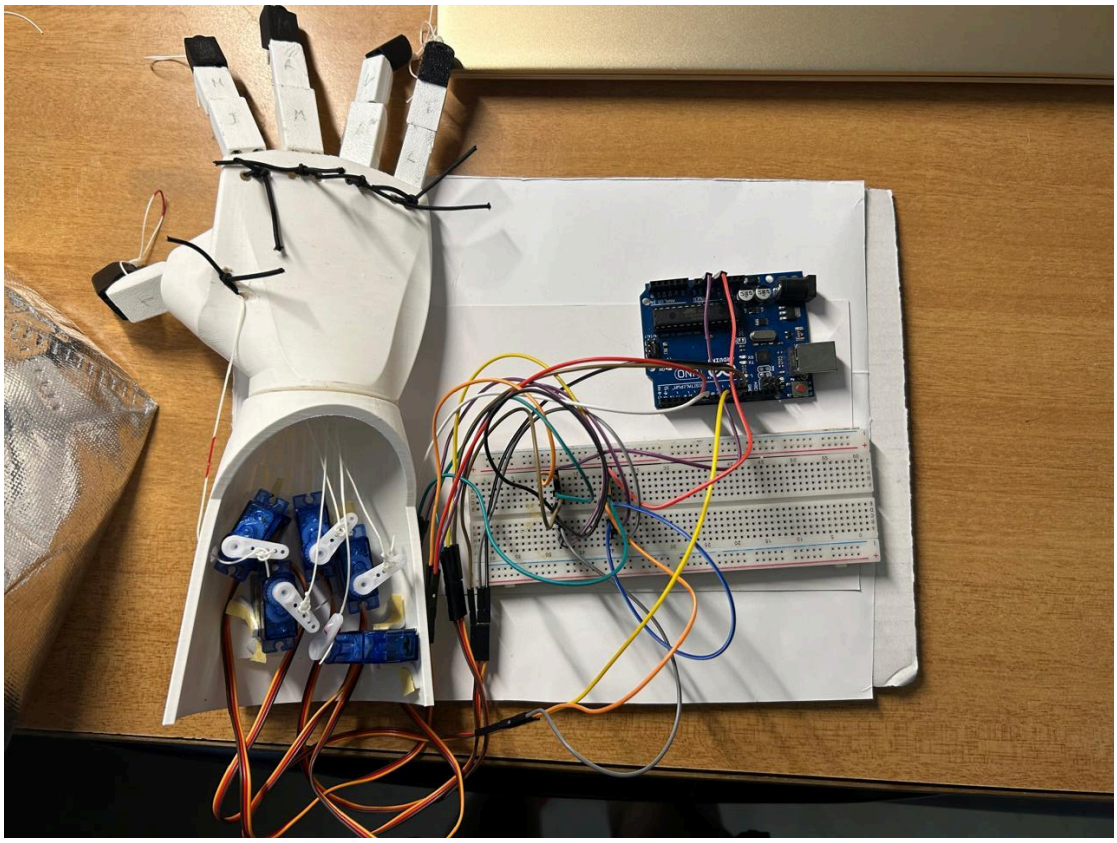
The hand is 3D modeled using SolidWorks and is 3D printed.The hand is mainly manipulated with the help of a string, the string and elastic wire passes through the hand, the elastic wire helps in getting the hand back to its initial position. The servo motors are connected to the string,they are rotated according to the input provided, as they move they manipulate the fingers with the help of the string.

Development Process:

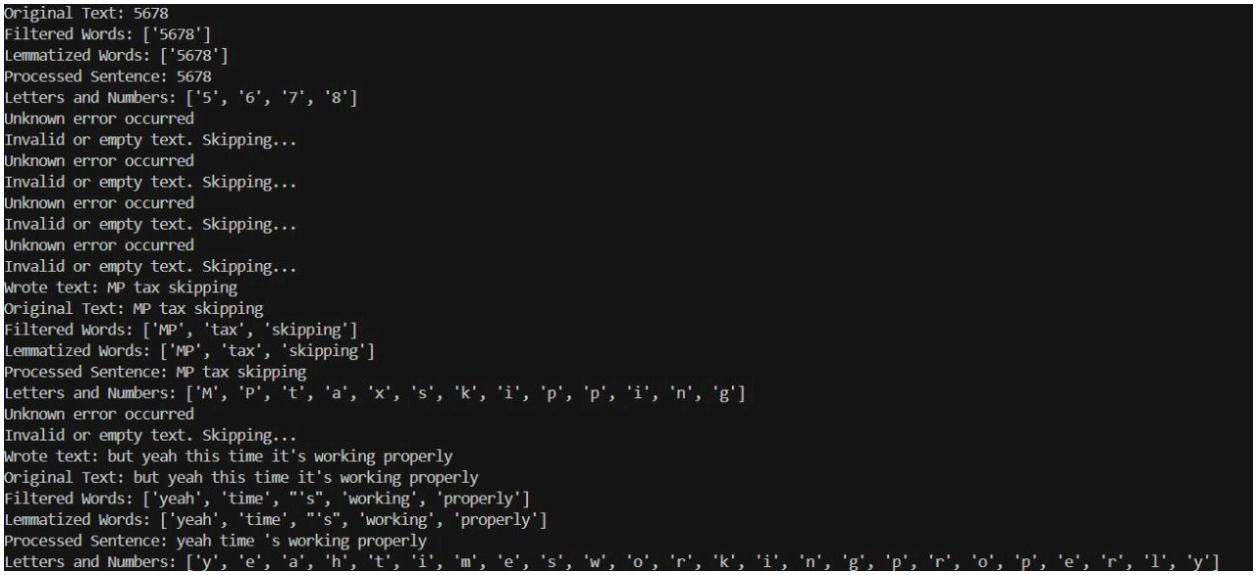
First we started this project by doing the Code part, then we proceeded by designing and getting the hand 3D printed, then we attached the strings and the elastic wires to the hand. Later we connected the Arduino to the servo motor and the servo motor to the strings and ran a basic code to check the working of the individual motor and to see if the hand operated as we imagined it would, and then we continued to run the main project code.

CHAPTER 5- TESTING

- Write the code part for the project.
- Design the hand on SolidWorks and get the hand 3D printed.
- Assemble the hand using strings and elastic wires.
- Attach the servo motors to the Robotic hand and Arduino.
- Test the servo motors and the movement of the fingers using a demo code.
- Run the main project code for the Robotic hand
- Verification of the sign language was done for each letter and number



Testing the code and its partial results



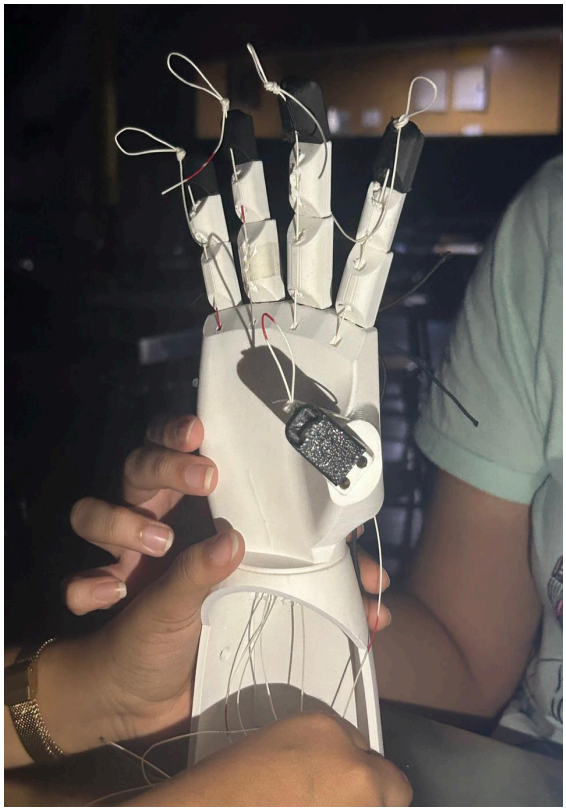
CHAPTER 6-CONCLUSION

In conclusion, the robotic hand that converts text into American Sign Language (ASL) using Python, Arduino, and servo motors is a valuable step forward in assistive technology. This system helps bridge communication gaps by translating text into accurate ASL gestures, making it easier for deaf and hard-of-hearing individuals to communicate with non-signers.

Python is used to process the text and map it to ASL gestures, while Arduino and servo motors control the movements of the robotic hand's fingers. The servo motors ensure precise finger movements, allowing the hand to replicate ASL signs effectively. Arduino acts as the central control system, connecting the software with the hardware components.

This project shows how robotics and programming can be combined to solve real-world challenges. It is especially useful in areas like education, public services, and everyday communication. With future improvements, such as adding more ASL signs, enhancing gesture accuracy, or enabling faster responses, this robotic hand could become an even more powerful tool for fostering inclusivity and breaking down communication barriers.

CHAPTER 7-VISUALS



CHAPTER 8-REFERENCES

[1] Z. Wang, Y. Liu, Z. Zhang, and J. Xu, "Design of Sign Language Translation System Using Machine Learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 7, pp. 2880-2892, Jul. 2021, doi: 10.1109/TNNLS.2020.3033874.

[2] Y. Yao, D. Zhang, and T. Han, "Sign Language Translation with Iterative Prototype," *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 9809-9818, doi: 10.1109/ICCV.2023.00967.

[3] A. Kumar and P. Yadav, "Machine Translation from Text to Sign Language: A Systematic Review," *ResearchGate*, 2021. [Online]. Available: https://www.researchgate.net/publication/352966000_Machine_translation_from_text_to_sign_language_a_systematic_review. [Accessed: Jan. 2, 2025].