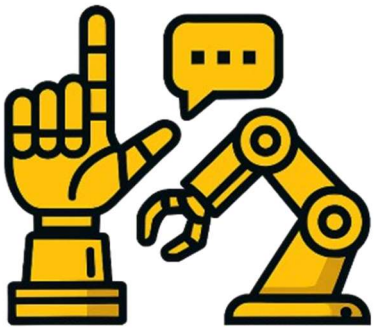
An-Najah National University
Faculty of Engineering and Information Technology
Electrical and Computer Engineering Department



PREPARED BY:

Ayman Abu Hijleh
Baha Alawneh

SUPERVISED BY:

Dr. Hanaal Abu-Zant

June 06, 2025

Presented in partial fulfillment of the requirements for Bachelor degree in Computer Engineering.

Dedication

First and foremost, to God be all the praise. We have reached this point in our educational path thanks to his heavenly guidance.

To the person whose wisdom and advice have consistently inspired us. Our Master Muhammad, may God bless him and grant him peace.

To the souls of our brave martyrs. Whose sacrifices have taught us the true meaning patience and motivated us to continue pursuing dreams on behalf of many of them.

To those whose prayers and supplications were the secret of our success.

To all those who believed in us, enveloped us with love and encouragement, lifted our spirits, and extended a hand when we felt on the brink of giving up our dear family, friends, and teacher thank you from the depths of our hearts for being an inseparable part of this journey.

Here stands this modest effort, a proof of the love and support that have driven us along the way.

Acknowledgment

We would like to express our sincere gratitude to all those individuals for mentoring and supporting us in completing this project. Our first expression of gratitude and appreciation goes to our supervisor **Dr. Hanaal Abu-Zanat**, for his great feedback and assistance during this project. We also extend our heartfelt gratitude to all the teachers in the Com- puter Engineering Department who gave their time for teaching us, working hard to improve our academic production, and recognizing our efforts throughout our university journey until we reached this point.

Disclaimer Statement

This report was written by ***Ayman Abu Hijleh*** and ***Baha Alawneh*** at the Computer Engineering Department, Faculty of Engineering, An-Najah Na- tional University. It has not been altered or corrected, other than editorial corrections, as a result of assessment and it may contain language as well as content errors. The views expressed in it together with any outcomes and recom- mendations are solely those of ***Ayman Abu Hijleh*** and ***Baha Alawneh***. An-Najah National University accepts no responsibility or liability for the con- sequences of this report being used for a purpose other than the purpose for which it was commissioned.

Contents

Contents

List of Figures

List of Tables

Abstract

SignTalk is an assistive system designed to enhance communication between the deaf and hearing community using sign language technology. The system includes three main features, each addressing a unique interaction challenge and helping to break down communication barriers in public and private settings, such as hospitals, airports, and schools.

Text/speech to sign language (letter-by-letter): Through a mobile app developed using the React framework, the user can input text or voice. The input is then sent to a 3D-printed robotic hand, controlled by servo motors, which moves individual sign language letters one by one, helping the hearing-impaired communicate.

Sign language to text/speech recognition: A camera captures the sign gestures produced by the person, and using MediaPipe and a trained CNN model (via PyTorch), the system recognizes the hand gestures and classifies them by letter. The recognized letters are then converted into text that is displayed on an LCD screen and spoken aloud via a loudspeaker, helping deaf people communicate with their peers.

Rock, Paper, Scissors: A fun and interactive application, the system allows users to play the classic game of rock, paper, scissors. Using computer vision and gesture detection on a Raspberry Pi 4, the user plays against the system, which randomly selects a move and announces the winner with visual and audio feedback. A robotic hand then displays the system's actual movement.

The hardware used includes an Arduino Mega 2560, an ESP32, and a Raspberry Pi 4 Model B (4GB RAM). The ESP32 manages the wireless connection between the application and the main device (Arduino Mega), while the Raspberry Pi processes video inputs and runs the AI models. The system integrates mechanical motion, AI-based recognition, and real-time interaction into a single, compact, and scalable solution aimed at making communication more inclusive and accessible.

Chapter 1: Introduction

The number of individuals with hearing and speech disabilities has increased significantly in recent years, leading to increased interest in assistive communication technologies. According to the World Health Organization, approximately 2.5 billion people are expected to have some degree of hearing loss by 2050, with more than 700 million of them requiring hearing rehabilitation due to disabling hearing loss. Currently, more than 430 million people already require such rehabilitation, including 34 million children [1]. Many of these individuals rely on sign language as their primary means of communication, highlighting the global need for accessible tools that facilitate access and help bridge the gap between sign language users and the rest of society.

With the rapid development of embedded systems, microcontrollers, and artificial intelligence, new solutions are emerging that seek to remove communication barriers by enabling real-time translation between sign language and spoken or written language [2]. Despite this progress, most interactions involving sign language still rely on human translators or simple devices with limited capabilities, and a comprehensive automated system has yet to be widely deployed.

Our proposed system is an intelligent robotic hand that provides an innovative multi-mode communication interface by integrating mechanical, electronic, and AI-based components. This hand is 3D printed and powered by an Arduino Mega 2560 board to control servo motors that mimic human finger movements. An ESP32 microcontroller enables wireless connectivity to a mobile app, while a Raspberry Pi 4 handles real-time gesture recognition and AI-based reasoning using an OAK-D depth-sensing camera [3].

The system operates in three different modes:

Text-to-sign translation: The user enters text or voice via the mobile app, and the robotic hand signs the corresponding characters using finger movements driven by a servo module.

Sign-to-text/speech translation: The camera detects the user's hand gestures, classifies them using a CNN model based on MediaPipe and PyTorch, and then converts the recognized gestures into text (displayed on an LCD screen) and audio (via a speaker) [4].

A "rock-paper-scissors" game mode allows users to play against a robotic hand. The

system interprets the user's movement through computer vision and responds with a randomly selected gesture, accompanied by real-time scoring and audio feedback.

This project demonstrates the potential of multi-functional, AI-powered robotic systems to enhance accessibility, learning, and interactive entertainment. The modular design enables the system to support various use cases through a simple, mobile-controlled interface. The system also provides continuous availability (24/7), eliminates the need for constant human assistance, and ensures consistent and unbiased interaction.

Furthermore, automation reduces operational errors and delivers more accurate results compared to manual, human-based systems. According to recent statistics, automated input systems have an accuracy rate of 99.959% to 99.99%, while human input accuracy ranges from 96% to 99%. When entering 10,000 data items, between 100 and 400 manual errors may be made, compared to only 1 to 4.1 errors using automated systems, highlighting the significant difference in reducing human error [5].

In Chapter 2, we explore related work in assistive robotics and gesture recognition. Chapter 3 presents the system design, its hardware components, software framework, and control methods. Chapter 4 covers the results and testing procedures. Chapter 5 provides a detailed discussion of the system's effectiveness, challenges, and limitations. Finally, Chapter 6 concludes with recommendations and potential future improvements.

Chapter 2: Literature Review

2.1 Similar Machines (Systems)

Several systems and research projects have been developed to support communication for individuals with hearing or speech impairments by converting sign language into text/speech or vice versa. Some of the most notable examples include:

2.1.1 Smart Glove Translator – UCLA

Researchers at the University of California, Los Angeles (UCLA) developed a wearable smart glove capable of translating American Sign Language (ASL) into English speech in real-time. The glove uses flexible sensors along the fingers to detect hand gestures and sends the data wirelessly to a smartphone application, which then generates speech. The glove is lightweight, low-cost, and can recognize up to 660 signs [6].

2.1.2 Project ASLAN – University of Antwerp

Students from the University of Antwerp designed a 3D-printed robotic hand named ASLAN (Antwerp's Sign Language Actuating Node). The robot translates typed text into physical sign language gestures using 16 servo motors and 25 printed parts. It was developed to help address the shortage of sign language interpreters in Belgium. However, it cannot interpret sign language from human users [7].

2.1.3 Low-Cost Smart Glove – UC San Diego

At the University of California, San Diego, engineers developed a low-cost smart glove that translates the ASL alphabet into text. It uses stretchable sensors to monitor finger positions and sends the data to a smartphone via Bluetooth. The system is focused on recognizing alphabet letters (finger spelling) rather than whole words or sentences [8].

Although these systems are innovative, most of them:

Are limited to one-way translation (either sign-to-text or text-to-sign).

Rely on wearable sensors or external hardware.

Do not support full real-time interaction or natural signing.

Our system aims to address these limitations by offering bidirectional communication (sign-to-speech and text-to-sign), using vision-based hand tracking with depth cameras and convolutional neural networks for accurate real-time performance.

2.2 Sign Language Recognition Techniques

Various techniques have been explored in the literature for recognizing sign language gestures, depending on the tools and complexity of the gestures. Common approaches include:

Vision-Based Recognition: Uses standard or depth cameras with tools like OpenCV and MediaPipe to detect and track hand landmarks without the need for wearable hardware.

Sensor-Based Recognition: Employs data gloves or wearable devices such as flex sensors, IMUs, or accelerometers to track hand movements and angles. These are often accurate but less practical for daily use.

Machine Learning & Deep Learning: Models like CNNs (Convolutional Neural Networks) and LSTMs (Long Short-Term Memory) are trained on gesture features (hand landmarks, image frames) to classify signs, including both static and dynamic gestures.

Our system utilizes a vision-based approach with CNN classification, extracting hand keypoints using MediaPipe, and supports real-time gesture recognition. Furthermore, Text-to-Speech (TTS) modules are integrated to provide audible feedback, enabling smoother communication with non-signing individuals.

Chapter 3:Methodology

3.1 Design

3.1.1 System's User Interface

The user can interact with the system through the following components:

- **Mobile Application:**
  The primary interface for user interaction. It allows the user to select the operating mode (Text/Voice to Sign Language, Sign Language to Speech, or Rock-Paper-Scissors game) and input text or voice commands.
- **LCD Display:**
  Used to show the current mode and then shows the individual letters being processed—either those being represented by the robotic hand in Mode 1, or those being recognized from the user's hand gestures in Mode 2.
- **Robotic Hand (Smart Hand):**
  Acts as the main visual output. It performs finger movements using servo motors to represent sign language letters or gestures, mainly in the text-to-sign mode.
- **Camera:**
  Captures the user's hand gestures for real-time sign language recognition using AI models. It is mainly used in the sign-to-speech mode and Rock-Paper-Scissors game.
- **Speaker:**
  Outputs the spoken version of the recognized text , aiding communication with hearing individuals.

Figure 1 System's User Interface

3.1.2 Getting Inside

The following image shows the inner design of the machine:

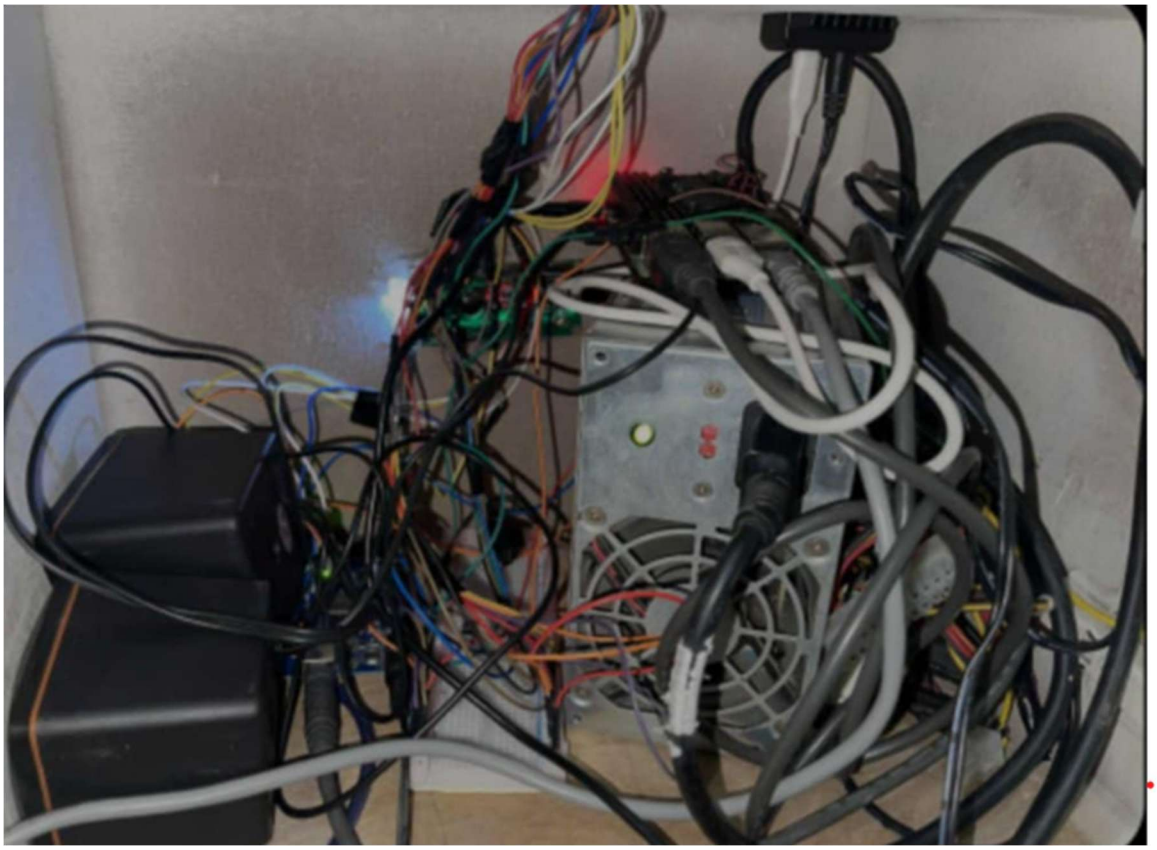


Figure 2 Inner design of the machine

3.2 Materials and Tools

3.2.1 Microcontrollers

- **Arduino-Mega 2560** [9]: A well-known microcontroller based on ATmega2560, It has the following features:
  - 54 digital input/output pins (of which 15 can be used as PWM outputs).
  - 16 analog inputs
  - 4 UARTs (hardware serial ports)
  - 16 MHz crystal oscillator
  - USB connection
  - Power jack
  - ICSP header
  - Reset button

  It is used to control the hardware.

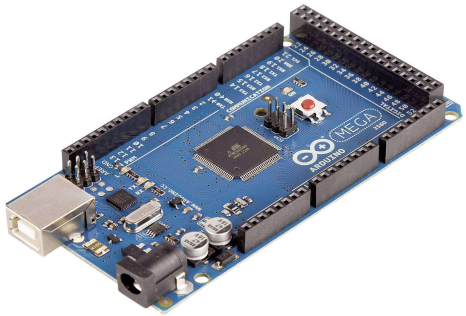


Figure 3 Arduino-Mega 2560

- **ESP32** [10]: A well-known microcontroller that supports 2.4 GHz Wi-Fi and Bluetooth, it is used to enable wireless control of the system via via the project's mobile application.

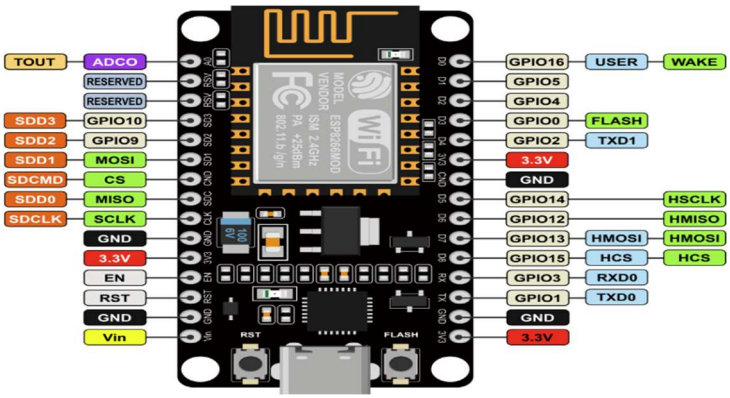


Figure 4 ESP32

• **Raspberry Pi 4 Model B [11]:**

A powerful and compact single-board computer based on the Broadcom BCM2711 quad-core Cortex-A72 (ARM v8) 64-bit SoC, commonly used for edge computing, IoT applications, and robotics. It features the following hardware specifications:

– Quad-core ARM Cortex-A72 @ 1.5 GHz
– 4 GB LPDDR4 RAM
– 2× micro-HDMI ports, supporting up to 4K output
– 40 GPIO pins (general purpose input/output)
– Dual-band wireless LAN (2.4GHz and 5.0GHz)
– Bluetooth 5.0
– Gigabit Ethernet port
– 2× USB 3.0 ports and 2× USB 2.0 ports
– microSD card slot (used for booting and storage)
– USB-C power port (5V, 3A recommended)
– Camera and Display interfaces (CSI and DSI)

It runs a 64-bit Linux-based OS (e.g., Raspberry Pi OS 64-bit) and is used as a high-level controller capable of handling user interfaces, networking, file systems, and communication with other microcontrollers (such as the Arduino Mega) via UART, I2C, or SPI.

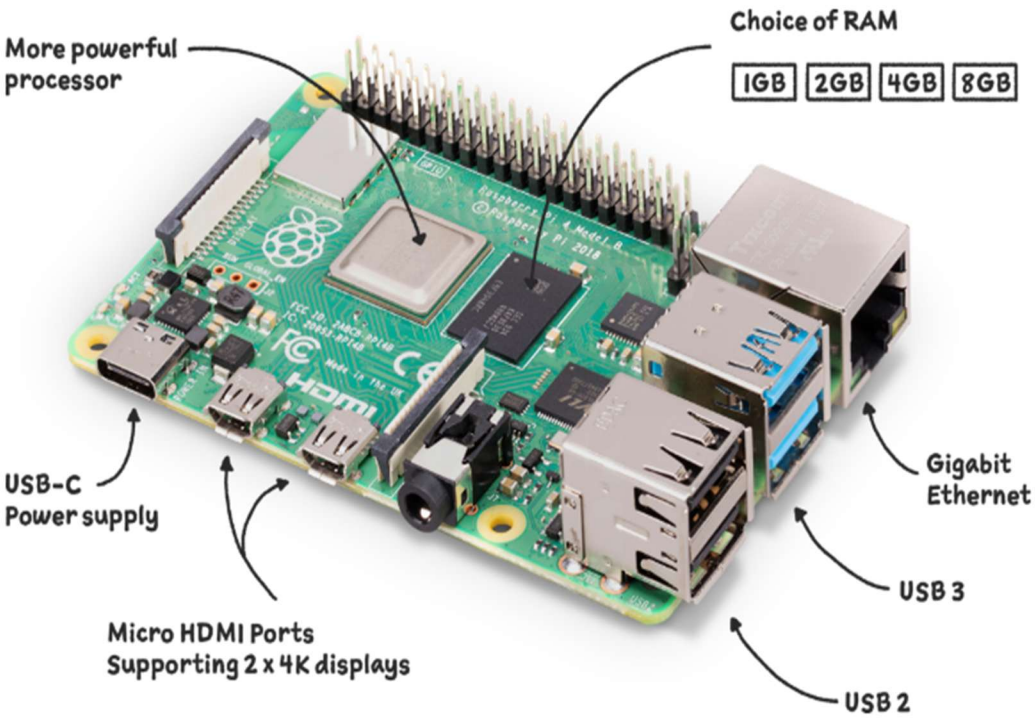


Figure 5 Raspberry pi 4 (model b)

3.2.2 Motors

- **Servo Motor (DS558HV)** [12]: A DC motor with some extra features:
  - **Control circuit**: set the angle [0°– 180°].
  - **Gears**: transform speed into torque.
  - **Potentiometer**: keeps track of the motor angle.

  It contains three connection pins:

  - V_{cc}: The red one.
  - GND: The brown one.
  - Control (Signal): The orange one and must be attached to PWM pin in the microcontroller.

  Six servo motors are used as the following:

  - 5 servo motors rotate at an angle of 0-180 degrees to move the fingers according to the letter to be represented
  - 1 servo motor rotates at an angle of 0-180 degrees to move the wrist in line with the movement of the fingers to represent some letters that require wrist movement.

Figure 6 Servo Motor

3.2.3 I/O Devices

- **16 * 2 LCD** [13]: An informative output that can be used to display characters (with limited custom symbols) , it supports two types of information: data or control commands (clear the screen, set the cursor, etc.) and it can be used in both byte-mode (8-bits) or nibble mode (4-bits). It has the following connection pins:
  - V_{cc}
  - V_{ss} or GND
  - V_o: for controlling the contrast via a potentiometer.
  - D0 - D7: data pins.
  - R/W (Read / Write): for selecting reading mode or writing one.
  - RS (Register Select): for setting the information type (data or control command).
  - EN (Enable): for enabling the LCD.

  LCD requires two many pins. So, we used I^2C LCD that uses an adapter PCF8574 that converts the LCD multiple connection pins into only four ones (V_{cc}, GND, SCL and SDA).

  It used in this machine to allow the user to monitor and track the transaction (displaying menus, showing results, etc.).

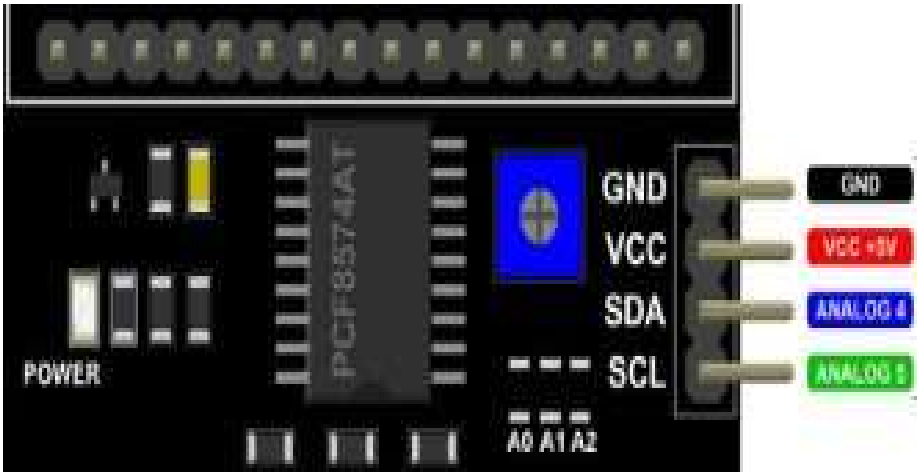


Figure 7 PCF8574 Adapter

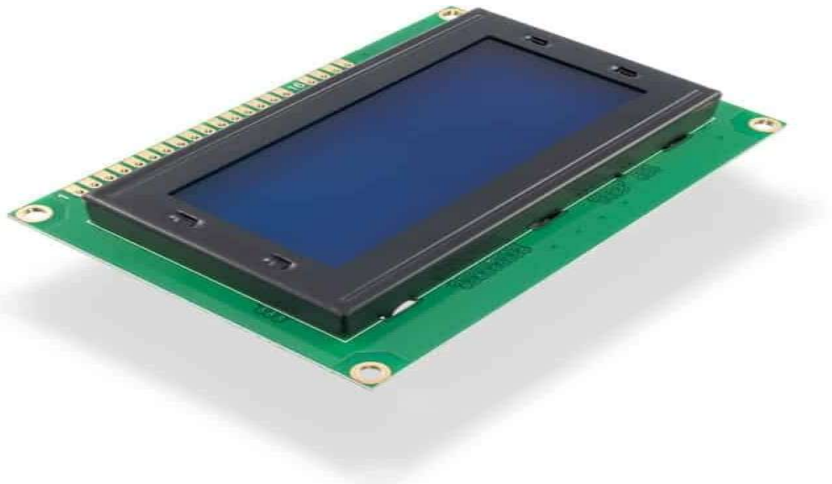


*Figure 8 16 * 2 LCD*

• **USB Speakers – Perfeo Wave PF_5127 [14]:**

An audio output device used to provide audible feedback or play sounds in the system. These **compact 2.0 stereo speakers** are compatible with any audio source that supports a **3.5 mm audio output**, such as computers, tablets, or smartphones. They are powered via a standard **USB port**, which eliminates the need for a separate power adapter, making them highly portable and easy to integrate.

The main features are:

– **Dual 2-inch speakers**, each rated at **4 Ohm**
– **Output power:** 3W × 2 (RMS)
– **Frequency response range:** 40 – 20,000 Hz
– **Connection:**
– **Audio input via 3.5 mm jack**
– **Power supply via USB (5V)**
– **Volume control** integrated into the cable
– **Compact dimensions:** 60 × 55 × 73 mm
– **Weight:** 186 g
– **Material:** Durable plastic housing
– **Bluetooth support:** Not available

These speakers are used in this machine to **play audio feedback or synthesized speech**, enhancing the user interaction and accessibility by providing voice or sound cues based on the operation performed.

Figure 9 USB Speakers

3.2.4 Others

- **Power Supply**: It is used provide power for the system, the following table shows the voltage required for each device:

Table 1 Power Supply Requirements for Each Device

Tool	Number of Item(s)
Arduino-Mega 2560 16 * 2 LCD Servo Motor Raspberry pi 4	5V
ESP32	3.3V

Figure 10 Power Supply

The following table shows the number of each tool used in this machine:

Table 2 Tools' Number of Items

Tool	Number of Items
Arduino-Mega 2560	1
ESP32	1
Raspberry pi 4 model b	1
Servo Motor	6
16 * 2 LCD	1
Speaker	1
Power Supply	1

3.3 Software Development

3.3.1 Tools

- **Arduino IDE** [15]: An open-source IDE to write Arduino / ESP codes and upload them into the microcontrollers.
- **CP210x USB to UART Bridge VCP Drivers** [16]: Drivers to install ESP32's code on it (to allow the communication between the computer and ESP32).

3.3.2 Arduino, ESP32 and Raspberry pi 4 Libraries

Arduino, ESP32 and Raspberry pi 4 provide multiple libraries to deal with various hardware tools, the following table shows the libraries used in this project:

Table 3 Arduino, ESP32 and Raspberry pi 4 Libraries

Tool	Library -If any-
Arduino-Mega 2560	Arduino.h
ESP32	SoftwareSerial.h WiFi.h
Raspberry pi 4	HandDetector depthai mediapipe CNNModel
Servo Motor	Servo.h
speaker	os
16 * 2 LCD	LiquidCrystal_I2C.h

3.3.3 Source Codes

- **Arduino Mega Code**: It is attached in Appendix A.
- **ESP32 Code**: It is attached in Appendix B.
- **Raspberry pi 4**: It is attached in Appendix C.
- **Mobile Application Code**: It is attached in Appendix D.

3.4 Mobile Application

3.4.1 Main Framework

• **React Native** [17]: A JavaScript framework developed by Meta for building cross-platform mobile applications with native-like performance.

3.4.2 Tools

Workspace Tools

- **Visual Studio Code** [18]: Very popular code editor that can be customized with multiple extensions to support any programming language and saves time and effort.
- **Android Studio** [19]: Very popular IDE in mobile development, we use it to create virtual devices to build and test the mobile application on them.
- **Postman** [20]: Desktop application (Provides also a website and VS Code extension) to test the back-end APIs and generate documentations for their usage.
- **Git / GitHub**:
  - **Git** [21]: Open-source version control system to deal with cloud-based platforms such as GitHub and GitLab.
  - **GitHub** [22]: Cloud-based platform for saving and sharing codes.

Development Tools

- **JavaScript** [23]: A widely-used programming language used in React Native to build cross-platform mobile applications for Android and iOS.

3.5 Standards and Constraints

3.5.1 Standards

- **RESTful API standards**: The RESTful API on ESP32 microcontroller requires the following standards -REST itself is not a standard- in order to communicate with the mobile application and exchange information:
  - Wi-Fi [IEEE 802.11]
  - HTTPS [RFC 2818]
  - URI [RFC 3986]
  - JSON [RFC 8259]
- **UART**: for serial communication between Arduino Mega 2560, ESP32, and Raspberry Pi 4.
- **USB 3.0**: for data transfer and power.

3.5.2 Constraints

• The inability to accurately represent characters on the robotic hand because the human hand contains many muscles and joints, while the robotic hand is controlled by six motors.
• The camera capture process was somewhat slow due to the Raspberry Pi's limited specifications. This problem was mitigated by reducing the display resolution.

Chapter 4: Results

The system supports three operational modes designed to facilitate communication for individuals with hearing or speech impairments. These modes include:

1. **Text-to-Sign Mode** – converts text input into sign language using a robotic hand.
2. **Sign-to-Speech Mode** – captures sign language via a camera, recognizes it using AI, and outputs the corresponding spoken text.
3. **Game Mode** – a rock-paper-scissors game using computer vision and a robotic hand to simulate gestures.

The system is wirelessly controlled via a **mobile application**, which allows users to select the desired mode. The **ESP32 microcontroller** acts as a communication bridge, transmitting mode selection and commands to the **Raspberry Pi 4** or **Arduino Mega**, depending on the mode. Each module then performs its designated task, with real-time interaction between hardware components such as servo motors, camera, and speaker.

4.1 LCD Display of Recognized Letters

Here are some pictures of the LCD screen when displaying letters on it:

Figure 11 Example of LCD Outputs During Operation

Chapter 5: Discussion

5.1 Working Principle

5.1.1 Input Mechanism

Mode 1 - Text to Sign Language Conversion:

The user enters text or voice via a mobile app. The text is sent to the ESP32 microcontroller via Wi-Fi, which in turn sends it to the Arduino Mega via serial communication.

The microcontroller processes the text and converts it into individual characters. Each character is then translated into a corresponding hand sign using servo motors that control the mechanical hand, based on a pre-defined mapping of the characters to the servo angles.

Mode 2 - Sign Language to Text and Speech Conversion:

When this mode is selected, the ESP32 microcontroller sends a signal to the Arduino Mega, which transmits it to the Raspberry Pi to initiate the hand gesture recognition system using the OAK-D camera.

The system uses MediaPipe technology to extract hand features and a trained CNN model to classify the sign. The recognized sign is converted to text and then to speech using a text-to-speech (TTS) engine such as pyttsx3, which is then played over a speaker.

Mode 3 - Rock, Paper, Scissors:

The ESP32 transmitter sends a trigger to the Raspberry Pi via the Arduino Mega to start the game.

The camera detects the user's hand gesture. The system then selects a random movement (rock, paper, or scissors), which is sent to the Arduino Mega. The mechanical hand then executes the system-selected gesture using servo motors. It compares it to the user's gesture and determines the outcome (win/loss/draw).

5.1.2 Output Mechanism

In Mode 1 (Text-to-Speech):

For each letter, the corresponding set of servo motors is activated to form the sign language letter through the mechanical hand. Each gesture is held for a short time before moving on to the next letter.

In Mode 2 (Gesture-to-Text):

The translated text is displayed on the screen or within the mobile app and spoken aloud via the speaker connected to the Raspberry Pi.

In Mode 3 (RPS Game):

A mechanical hand connected to an Arduino Mega represents the system's chosen move (rock, paper, or scissors) based on the result received from the Raspberry Pi. The result is also displayed on the mobile app.

5.1.3 Hand Gesture Detection

The OAK-D camera is used to detect hand gestures in real time.

The algorithm relies on the MediaPipe library to extract 21 hand features in each frame, which are then passed to a trained CNN model that classifies the hand gesture into a character or corresponding game action.

In Mode 2, this classification is used to output text and speech. In Mode 3, it is used to identify the user's movement.

5.1.4 Communication and Synchronization

Communication between the three main components (ESP32, Arduino Mega, and Raspberry Pi) is organized as follows:

The mobile application sends a mode number (1, 2, 3, or 0 for exit) to the ESP32.

The ESP32 forwards the number to the Arduino Mega via the serial port.

If the selected mode is 2 or 3, the Arduino Mega sends a signal to the Raspberry Pi (via a shared file or serial port) to run the corresponding script.

Once the Raspberry Pi completes the task, it returns the result to the Arduino, which executes the appropriate action.

Challenge: Ensure synchronization between the three devices and avoid command conflicts.

Solution: Implement a clear communication protocol using unique identifiers and acknowledgments (ACKs), with non-blocking logic and proper separation of tasks for each mode.

5.2 Mobile Application GUI

The mobile application provides an easy-to-use user interface that allows the user to control the device remotely.

Start Screen: Contains a "Get Started" button that takes us to the main screen to run one of the mods and interact with the project. The screen contains a button that allows us to change the application's display language, as the application supports two languages: Arabic and English. The second button allows us to change the application's theme (Light/Dark Mode).

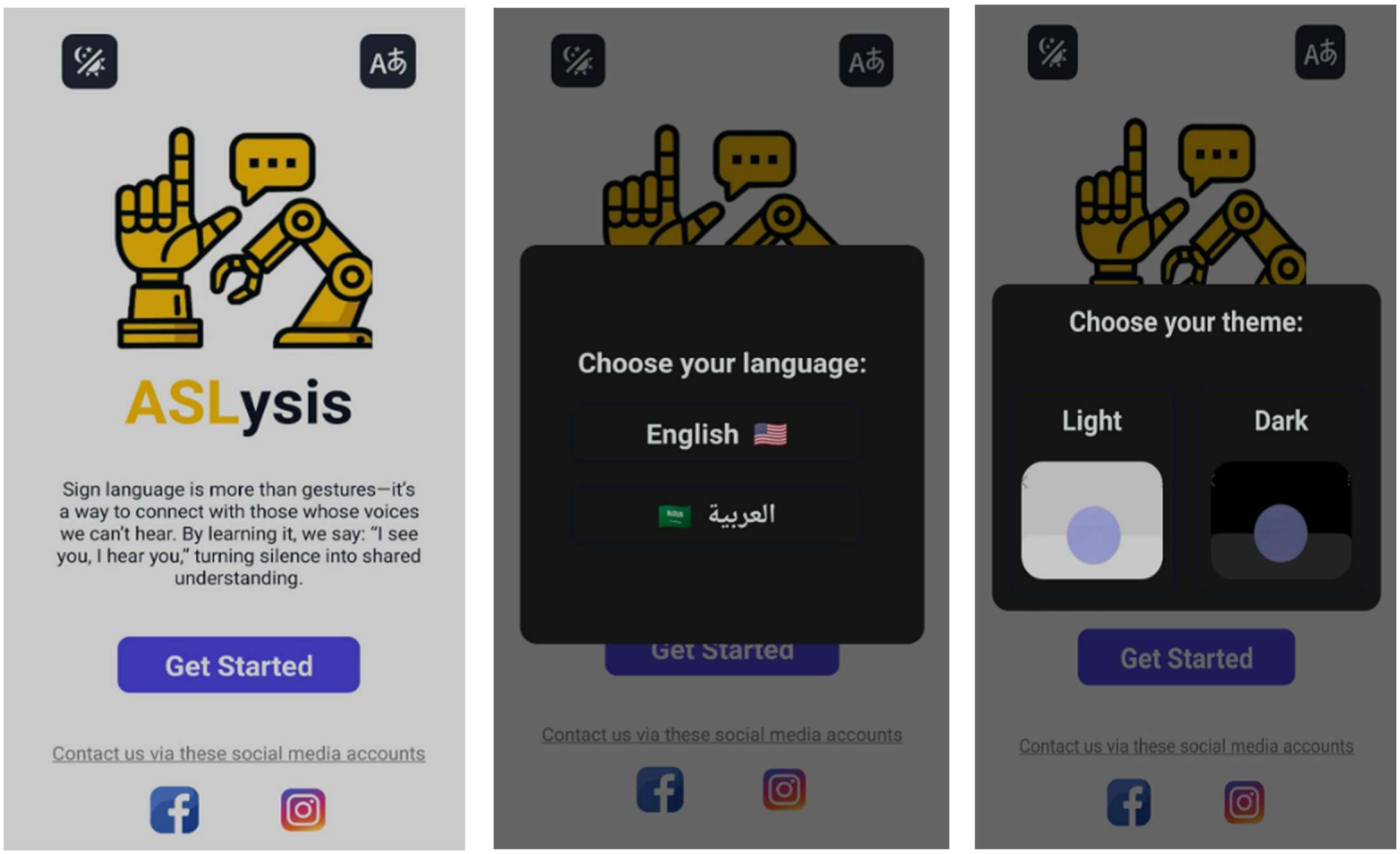


Figure 12 The Application's Start Screen

Home Screen: Contains three buttons to select one of the operating modes (Text to Signs, Sign to Speech, and RPS Game), in addition to an "Exit" button to stop any ongoing process:

Text Input Screen (Mode 1): Allows the user to enter text or audio that is sent directly to the system to represent sign language.

Text Display Screen (Mode 2): The text being represented is displayed in front of the camera after pressing the Stop button. The screen contains a New Trial button, which starts the camera's character recognition.

RPS Screen (Mode 3): This displays the game score. The round begins when you press the New Trial button, and the score is updated with each round. The screen contains a Restart button that resets the score.

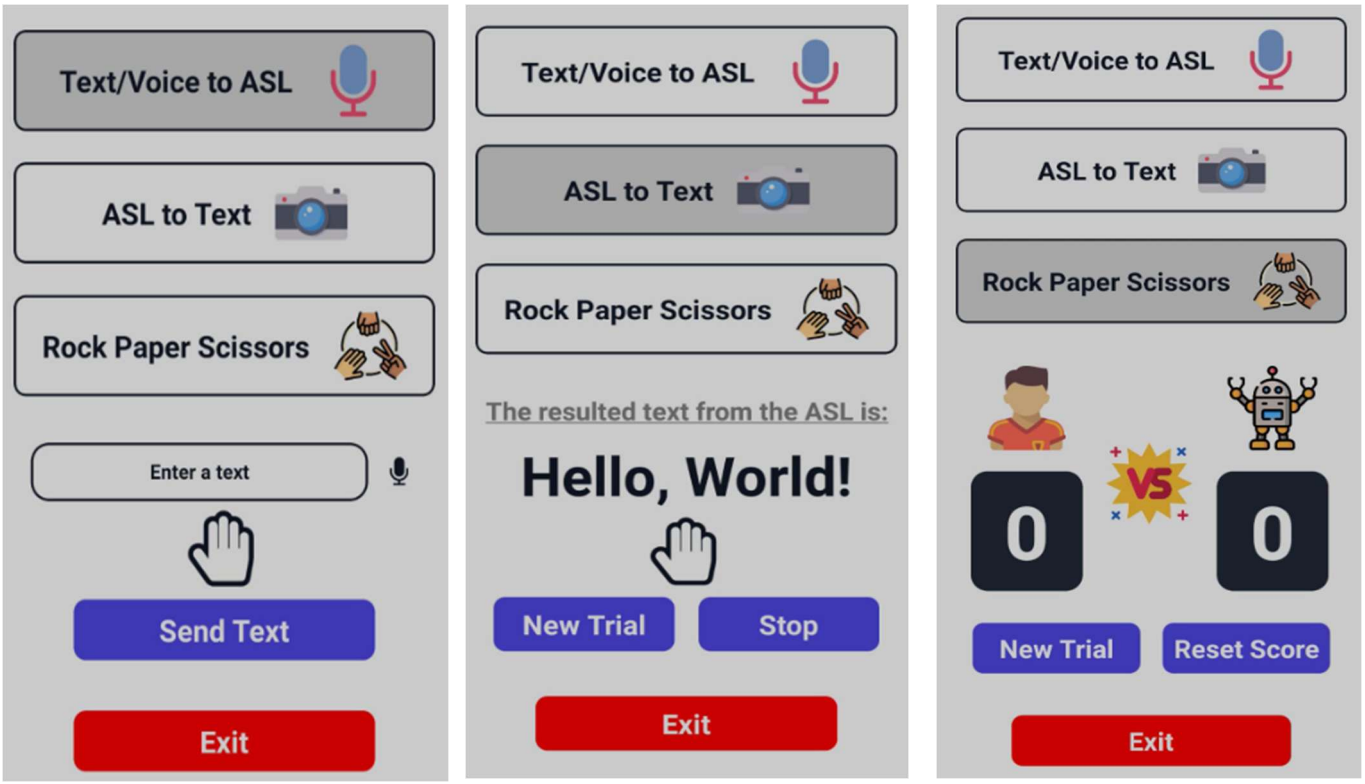


Figure 13 The Application's Modes Screen

It uses the following end-points from RESTful API on ESP32:

- GET: **http://192.168.4.1/mode1** - Selects mode (1)
- GET: **http://192.168.4.1/submit-text/{text}** -to send text to the Arduino to represent it letter by letter
- GET: **http://192.168.4.1/mode2** - Selects mode (2)
- GET: **http://192.168.4.1/mode3** - Selects mode (3)
- GET: **http://192.168.4.1/start** - to start mode 2 or mode 3 according to which mode is selected:

  Mode2: start predicting letters based on the representation of the human hand in front of the camera

  Mode3: start the round to play with the robotic hand
- GET: **http://192.168.4.1/stop** - stop predicting letters to be displayed on the app, as well as outputting them in audio and displaying them on the LCD.
- GET: **http://192.168.4.1/restart** - clears the current score so that someone else can play with the robotic hand.
- GET: **http://192.168.4.1/exit** - to exit the current mode

Chapter 6:Conclusion, Recommendations and Future Work

6.1 Conclusion

In this project, we successfully developed a smart robotic hand system capable of facilitating communication through sign language. The system supports three main modes:

Mode 1 allows users to input text via a mobile application, which is then translated into corresponding sign language gestures through servo-controlled hand movements.

Mode 2 utilizes an OAK-D camera with MediaPipe and a trained CNN model to recognize real-time hand gestures and convert them into text and speech, helping the hearing-impaired communicate effectively.

Mode 3 is an interactive Rock-Paper-Scissors game, where the system recognizes the user's gesture, randomly selects a move, determines the winner, and physically displays the system’s move using the mechanical hand.

Communication between the ESP32, Arduino Mega, and Raspberry Pi is well-structured, allowing seamless data flow and task delegation. The mobile application provides an intuitive interface for mode selection and text input.

6.2 Recommendations

We recommend that individuals planning to develop similar assistive technology projects begin by thoroughly analyzing the interaction between software components (like gesture recognition and wireless communication) and hardware components (such as servo control and camera input).

carefully define the system's operating modes from the beginning and ensure each has dedicated responsibilities to avoid unnecessary

6.3 Future Work

The current system can be improved with several ideas:

- Bidirectional Sign Language Support: Extend the character set to support full words and phrases for more natural communication.
- Language Customization: Support additional languages and regional sign language dialects.
- Cloud Synchronization and Logging: Allow the system to sync gesture logs and interactions with a cloud server for tracking or educational purposes.
- Battery Optimization and Power Management: Enhance the hardware design to be battery-operated with efficient power consumption.

References

[1] WorldHealth Organization. (2024). *Deafness and hearing loss*. Retrieved from: https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss.

[2] "Lenovo StoryHub. (2023). *Lenovo's AI-powered sign language translation solution empowers signers in Brazil.*" https://news.lenovo.com/ai-powered-sign-language-translation-solution-hearing-brazil/

[3] Luxonis. (2023). *OAK-D Camera Documentation*. Retrieved from: OAK-D – Luxonis.

[4] MediaPipe by Google. (2023). *Open-source Framework for Building Perception Pipelines*. Retrieved from: https://mediapipe.dev

[5] "DocuClipper. (2025). *67 Data Entry Statistics For 2025*". Retrieved from https://www.docuclipper.com/blog/data-entry-statistics/

[6] Matthew Chin, "Wearable-tech glove translates sign language into speech in real time," UCLA Newsroom, 2020. https://newsroom.ucla.edu/releases/glove-translates-sign-language-to-speech

[7] Arduino Team, "Project Aslan is a 3D-printed robotic sign language translator," Arduino Blog, 2017. https://blog.arduino.cc/2017/08/21/project-aslan-is-a-3d-printed-robotic-sign-language-translator/

[8] University of California, "Low-cost smart glove translates American Sign Language alphabet," UC News, 2020.

https://www.universityofcalifornia.edu/news/low-cost-smart-glove-translates-american-sign-language-alphabet

[9] “Arduino mega 2560 documentation.” https://docs.arduino.cc/hardware/mega-2560/.

[10] “Esp32 datasheet.” esp32-wroom-32_datasheet_en.pdf.

[11] “Raspberry pi 4 model b.” Buy a Raspberry Pi 4 Model B – Raspberry Pi.

[12] “Servo motor - circuit digest.” https://hobbyking.com/en_us/corona-ds538hv-digital-metal-gear-servo-8kg-0-12sec-58g.html.

[13] “16 * 2 lcd - winstar.” https://www.winstar.com.tw/products/oled-module/oled-character-display/oled-16x2-character.html.

[14] USB Speakers – Perfeo Wave PF_5127 https://perfeo.ru/kolonki-20-pf-128-wave-usb.html

[15] “Arduino ide website.” https://www.arduino.cc/en/software.

[16] “Cp210x usb to uart bridge vcp drivers website. CP210x USB to UART Bridge VCP Drivers - Silicon Labs.

[17] “React Native website.” React Native · Learn once, write anywhere.

[18] “Visual studio code website.” https://code.visualstudio.com/.

[19] “Android studio website.” https://developer.android.com/studio.

[20] “Postman website.” https://www.postman.com/.

[21] “Git website.” https://git-scm.com/.

[22] “Github website.” https://github.com/.

[23] “JavaScript website.” https://developer.mozilla.org/en-US/docs/Web/JavaScript.

Appendix A

Arduino Mega 2560 Source Code

Available upon request.

Appendix B

ESP32 Source Code

Available upon request.

Appendix C

Raspberry pi 4

Available upon request.

Appendix D

Mobile Application Code

Available upon request.