Lebanese American University

Electrical and Computer Engineering Department

COE595/ELE595/MCE591 Capstone Design Project I

E-S.H.A.R.A.

Electronic Sign-Language Hand Augmented Recognition Assistant

Advisor: Dr. Samer Saab Jr

Razan Hmede 202103291
Farah Al-Nassar 202101685
Aya Jouni 202100151
Georgio El-Khoury 202102787

October 9, 2024

Contents

Abstract

This paper presents a comparative review of various approaches to sign language interpretation, with the aim of improving communication between deaf and hearing individuals. The literature review highlights existing field complications and serves as a basis for the development of a smart glove that facilitates a two-way communication protocol in everyday interactions. Smart glove interpreters offer a promising solution to overcome communication barriers caused by limited sign language knowledge. By integrating both sign language interpretation and speech-to-text conversion, this dual-mode system addresses the limitations of one-way communication tools, enhancing user comfort, interaction fluidity, and gesture recognition accuracy. Our proposed solution leverages specific sensor integration for precise gesture tracking combined with visual and auditory outputs, eliminating therefore the user's need to download additional applications or the system rely on Internet connectivity. By offering a complete inclusive communication solution, this system has the potential to significantly improve daily interactions between deaf and hearing individuals by bridging the communication gap between the two communities.

Index terms— *Sign language Interpretation, Smart Glove, Gesture Recognition, Speech-To-Text Conversion, American Sign language, Gesture Classification, Finger Bending.*

1 Introduction

1.1 Introduction to the Subject

The World Federation of the Deaf (WFD) estimates that around 72 million people globally are deaf, with over 80 percent residing in developing countries [33]. Individuals with hearing impairments face significant challenges when communicating with hearing people who do not particularly understand sign language. Interpretation of sign language serves as the primary means of communication for the deaf community, but limited knowledge of sign language in the general population restricts accessibility.

1.2 Problem Statement

The lack of widespread sign language education within mainstream society exacerbates communication difficulties for the deaf.This gap significantly affects the everyday lives of people with hearing impairments, creating barriers to essential social, educational, and professional interactions. Without widespread sign language proficiency, opportunities for inclusion and equal access remain limited.

1.3 Approach

To address these challenges, significant technological advancements in wearable sensor-based hardware have been made to bridge the communication gap between the hearing impaired and hearing-capable people. This project focuses specifically on sensory gloves that offer real-time sign language interpretation, as well as speech-to-text technology to offer two-way communication on the fly. E-S.H.A.R.A, Electronic Sign-language Hand Augmented Recognition Assistant, sensory gloves are equipped with embedded sensors to detect hand orientation and finger bending. The captured data is preprocessed and then processed using machine learning algorithms to translate gestures into text which is displayed on an OLED screen for the deaf-user and output through a speaker for the hearing individual. This facilitates real-time communication between deaf and hearing individuals. In addition, the system integrates speech-to-text conversion , enabling two-way communication and making interactions more dynamic and accessible. With this approach, both communicators will be able to interact seamlessly.

1.4 Contributions

This report reviews previous advancements in the field and introduces key contributions to smart glove technology:

- **Optimized glove-sensor configurations:** Focused on choosing the most effective combination of embedded sensors, this contribution enhances the accuracy of gesture recognition, particularly for complex signs, thereby improving communication efficiency.
- **Regional coverage and language adaptation:** Added support for Arabic Sign Language (ArSL) alongside the American Sign Language (ASL) to foster accessibility for deaf communities in the MENA region. In Arab countries, the hearing impaired people exceeds 10 million [35]. This enhancement not only broadens the

system's usability but also acknowledges and respects the linguistic diversity of local users, ensuring that individuals can communicate effectively within their cultural context.

- **Custom sign language dataset:** Developed from scratch to ensure compatibility with the customized data glove.
- **Low-cost dual-mode communication system:** Integrated both gesture-to-text-to-speech translation and speech-to-text, alongside graphic, representation, helping hearing-impaired individuals.
- **Prerequisite-free functionality:** By simply wearing the glove, deaf users can effortlessly start and engage in conversations with non-deaf users without the need to download any software or setup any other hardware for communication. This ease of use fosters more natural and fluid interactions, enhancing the communication experience between the two communities involved.
- **Internet-independent operation:** The system can operate without requiring the user to be connected to the internet which is ideal for daily outdoor accessible interactions.

2 Background

A translation system intended to interpret sign language can be categorized as either vision-based or sensor-based, depending on the type of input it receives [3]. In the image-based approach, cameras capture images of signs, employing image processing techniques to identify the signs and algorithms to interpret their meanings. In contrast, the sensor-based method involves either constructing a wearable device such as armbands, gloves or wristbands equipped with sensors such as accelerometers, flex or electromyography (EMG) sensors that track the hand gestures, muscle movements and finger positions. [7]. Obstacles in vision-based systems such as gesture obstruction during expression, low lighting, and background clutter can lead to inaccurate recognition [29]. In contrast to vision-based technology, a smart glove equipped with flexible sensors offers a practical solution for hand gesture and sign language recognition, due to its real-time responsiveness and portability.

2.1 Background on Sign Language Interpretation Technology

Among the most promising technologies addressing communication challenges for deaf and hearing individuals are smart gloves, capable of translating sign language alphabets and sentences, depending on the complexity of their design and the accuracy of gesture recognition they provide. These systems, based on sensory gloves, are critical innovations that focus on capturing data regarding the shape and movement of the human hand. Sensory gloves, which can gather detailed information about hand motions, have been of interest since the 1980s in fields such as Human-Machine Interaction (HMI) and human motion analysis. According to the work in [8], the first significant development in sensory gloves dates back to 1982, when Thomas G. Zimmerman filed a patent for the Data Glove, using flexible optical sensors to measure finger bending. Zimmerman

later advanced his design with the Power Glove, which incorporated ultrasonic and magnetic sensors to track hand movements and poses. Since then, data gloves have gained increasing attention in research and practical applications. In the early 2000s, the first commercially available glove designed specifically for interpreting hand gestures, including sign language, was introduced as the AcceleGlove [13]. Today, various commercial models featuring different sensor configurations and capabilities are available, such as the 5DT Data Glove Ultra, Hi5 VR Glove, Enable Talk and Manus Prime II [8]. However, these devices remain expensive, with an average price exceeding one thousand dollars, limiting their accessibility to a broader user base. Additionally, they lack tactile sensing, which reduces their accuracy and flexibility. Nevertheless, many research groups worldwide have developed custom data glove solutions from scratch, using various designs and approaches to improve performance. This review examines several of these gloves, comparing them in terms of accuracy, speed, and range of gesture recognition. We will also consider additional factors such as portability and hardware cost, highlighting the advancements and ongoing challenges in this technology.

2.2 Sensor Technology in Sign Language Interpreter Gloves

Praveen et al. [30] developed a glove design utilizing a pair of LED-LDR components to gather data on each finger's position for distinguishing letters. The analog signals from the sensors are converted into digital data using an 8-channel ADC, which are subsequently encoded into ASCII code and displayed to the user. This design operates on the principle that bending a finger alters the resistance of the LDR due to changes in light emitted by the LED. Experimental tests were conducted on the 10 most commonly used alphabet letters, focusing on gestures that neither required wrist movements nor finger contact. While this design effectively detects finger bending, it cannot recognize complex sign language gestures or rapid finger movements.

Ahmed et al. [1], in their review of system-based sensory gloves developed between 2007 and 2017, detailed the sensory configurations in commercially available gloves such as the Cyber Glove and 5DT Data Glove Ultra. According to their findings, the Cyber Glove is equipped with 22 thin, flexible, and virtually undetectable sensors. These include three flex sensors per finger, four abduction sensors between fingers, a palm-arch sensor, and sensors for wrist flexion and abduction, achieving recognition accuracy between 92% and 95%. The authors also described the 5DT Data Glove Ultra, which uses five fiber optic sensors to measure finger flexion and an additional sensor to detect hand orientation. A more advanced version includes 14 fiber optic sensors (two per finger), abduction sensors, and two tilt sensors for roll and pitch measurement. They emphasized that finger bending, particularly the frequent motion of the four fingers bending toward the palm while the thumb moves freely in six degrees of freedom, presents a primary design challenge.

To elaborate on the flex sensor technology in the Cyber Glove, Ahmed et al. explained its principle: bending the sensor substrate produces resistance proportional to the bend radius, with smaller radii yielding higher resistance. This phenomenon correlates bending motions with sensor resistance changes. Similarly, Amin et al. [5] discussed the flex sensor's use in an E-voice glove for sign language interpretation. This glove design incorporated five flex sensors attached to an Arduino microcontroller. Each alphabet was linked to specific value ranges from the sensors, achieving 98% accuracy for alphabets and 100% for numbers.

Pezzuoli et al. [28] introduced "Talking Hands," a low-cost data glove equipped with

10 Spectra Symbol flex sensors, an IMU (BOSCH BNO055) for hand orientation detection, and a system initialization button. This design also featured an arm module comprising a 32-bit Cortex M3 Atmel SAM3X8E microprocessor, an IMU for arm orientation detection, RGB LED for system status, a battery and charge module, and a Bluetooth module (Microchip RN42). This approach distinguishes itself by incorporating arm and hand orientation detectors as additional features.

Lastly, Lu et al. [11] introduced a design combining five flex sensors with a novel "WonderSense" sensor. This sensor collects acceleration data using a 9-axis inertial sensor model (MPU9250). Data from "WonderSense" is transmitted to a PC via a "WonderBox," which relies on the PCA10040 chip for Bluetooth reception. Each glove in this system is equipped with five flex sensors, an Arduino board, and a WonderSense sensor, with the WonderBox serving as the central receiver.

In addition, surface electromyography (sEMG) sensors can be utilized to enhance gesture detection. Ben Haj Amor et al. [9] argue that incorporating sEMG signals marks a significant advancement towards an effective gesture recognition system, differing from conventional approaches. sEMG signals are biological signals that capture the electric current generated on the skin surface near a muscle during contraction or relaxation. These signals' complex patterns and accompanying noise present classification challenges that require sophisticated signal processing and machine learning techniques.

The hand-talk system proposed in [6] incorporates a glove circuit comprising one flex sensor per finger, a 3-axis accelerometer, and three sEMG sensors, including one reference electrode. Signals from the flex sensors and sEMG sensors are filtered and amplified before being converted to digital format via an Analog to Digital Converter (ADC). The electrodes used to measure sEMG waves are strategically placed on specific muscles: the flexor carpi radialis, the extensor carpi radialis longus, and a ground electrode on a non-muscular area for reference.

According to Figure 7 in [9], the flexor carpi radialis is a muscle on the anterior side of the forearm, responsible for wrist flexion and abduction. Conversely, the extensor carpi radialis longus is located on the posterior forearm and contributes to wrist extension and abduction. These placements optimize the glove's ability to capture meaningful signals for gesture recognition.

Netchanok and Surapa [36] argue that incorporating EMG sensors does not consistently enhance recognition accuracy, with subject-independent recognition results averaging a relatively low 60%. In an earlier design [36], a wireless sensor glove was developed featuring five flex sensors, a 3D accelerometer, and a BSN node. However, this design suffered from frequent misclassifications of letters such as M, N, O, X, S, U, and V due to the insufficient information provided by the sensory signals.

To resolve these challenges, a revised glove design proposed in [36] integrated contact sensors to improve recognition accuracy. This updated design includes two positive poles positioned at the thumb and index finger, alongside five negative poles, whose placement was determined experimentally for optimal functionality. The revised approach also incorporates a multiplexing technique that combines multiple sensory signals and transmits them via the same microcontroller input.

The flex and contact sensors are connected in parallel, augmented by two resistors. During contact, a high voltage range is generated, while resistors 1-4 adjust the output voltage to match the flex sensor's operating voltage level. This configuration enables accurate flex value extraction during both contact and non-contact gestures, thereby enhancing the overall recognition performance of the system.

Korzeniewska et al. [20] utilized custom-made bending sensors in their sensory glove, which offered a cost-effective and flexible solution along the entire sensor length. The design features a core material placed between two separate conductive layers, which are constructed by stitching onto a piece of fabric. The core materials evaluated for this purpose included Velostat (a polymer film coated with carbon black), Cordura (a durable polyamide fiber fabric resistant to mechanical damage), and a Goretex membrane (a flexible, Teflon-based material coated with a conductive silver layer through a vacuum deposition process). They also showed that the electrical current flows sequentially through the first conductive layer, the core material, and then the second conductive layer. Additionally, the glove was outfitted with a FLORA accelerometer from Adafruit, which integrates a 3-axis accelerometer and magnetometer using an I2C interface. Testing the sign translation algorithm revealed that Velostat was the most effective core material for the bending sensor, demonstrating superior performance compared to the alternatives.

Wu et al. [40] propose yarn sensors as an alternative to film-based sensors for detecting human joint motion. Full-fiber yarn sensors are flexible, invisible, and can be easily integrated into wearable textile gloves. The Artificial Intelligent Yarn Sensors (AIYS) were created using a yarn-wrapping machine, with silver-coated PA yarn wrapped onto hollow bobbins in both clockwise (CW) and counterclockwise (CCW) directions for the inner and outer layers. The yarns were spun with various twist counts: 300, 600, 900, and 1200 twists.After spinning, the AIYS sensors were connected to copper electrodes using conductive silver paste and encapsulated in epoxy resin for protection. In this study, the authors incorporated sixteen AIYS sensors onto a knitted glove. Fourteen sensors were positioned vertically on the movable joints of the fingers, one on the wrist, and one between the index and middle fingers, providing a comprehensive setup to track hand movements.

Jeon et al. [16] categorize hand gestures into two types: static and dynamic gestures. Considering both types is crucial for achieving reliable recognition in real-world applications, as humans perform both types of gestures. The system they propose utilizes a soft-embedded sensor, the Mollison hand, a commercial VR glove by Feel The Same, which includes 10 sensors such as haptics, pressure sensors, and a built-in HIFLEX sensor that measures the degree of flexion or bending at the metacarpophalangeal (MCP) and proximal interphalangeal (PIP) joints, as shown in Figure 3.

Figure 2 illustrates the fabrication of the soft sensors used by Jeon et al. in [16]. These soft sensors are composed of Eutectic Gallium-Indium (EGaIn), a liquid metal ink that offers excellent conductivity, flexibility, and stretchability. Additionally, the sensor features a substrate material, such as elastomers, a protective encapsulation layer, and an ink deposition method used to precisely print EGaIn patterns on the substrate, as discussed by Kim et al. [19].

Lee and Bae demonstrated that the resistance of this soft sensor increases as the finger flexes, showing a linear relationship from the clenched fist (90° flexion) to the flat position (0°) [23]. While both soft sensors and flex sensors are used to detect finger bending, soft sensors offer more versatility in capturing multidimensional deformations, making them suitable for more complex applications. Flex sensors, on the other hand, are optimized for one-dimensional bending detection with limited stretch capabilities [19].

Additionally, according to the work in [41], that uses 5 flex sensors, an MPU6050, and an Arduino as a microcontroller, by incorporating additional accelerometer sensors, the glove can detect gestures with unique slopes, such as the letters G, H, J, P, and Q. These gestures exhibit significantly different accelerometer readings compared to other letters,

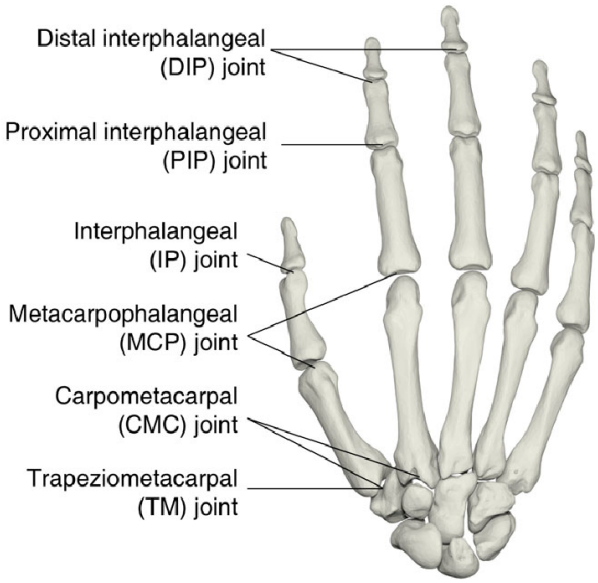


Figure 1: MCP and PIP joints of the fingers measured by the HIFLEX sensor

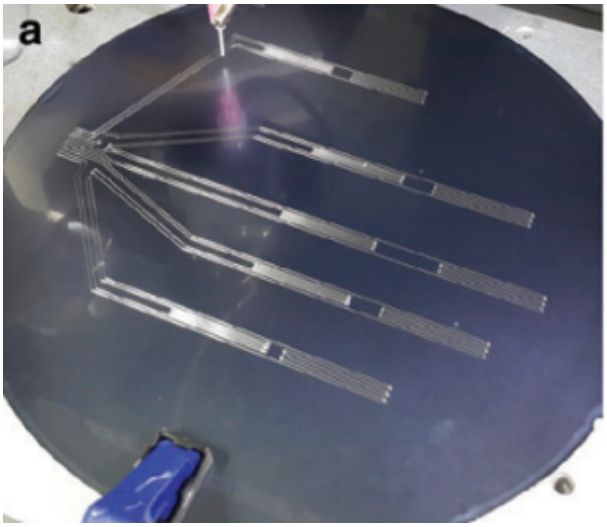


Figure 2: Soft sensors for the embedded data glove

allowing for more accurate differentiation.

As for the sensors used in gloves that interpret Arabic sign language (ArSL), as per Sadek, Mikhael, and Mansour, and after valuable statistics for ArSL that depend on the anatomical shape of the hand in addition to words with double hand movement, words with one or more stages by single hand and words with one or more stages by the double hand, the smart glove was designed to have 6-flex sensors, 4-contact sensors, 1-Gyroscope, and 1-Accelerometer, in addition to the controller [35]. The work also analyzes the total number of times that a certain sensor was used in a movement detection or orientation of a sign, to reduce the hardware cost, and concluded that the proposed design is less than 5% of the lowest price of commercial smart gloves, as well as simpler. The design of this glove was tested by simulation and proven only using PROTEUS program.

In contrast, Zhu et al. argue that soft fiber optic sensors are the best choice for dual-glove systems [42]. A soft multi-mode sensor is vital for detecting finger movements in a wireless smart glove system. Optical fibers were selected for their flexibility, lightweight

nature, high sensitivity, and immunity to electromagnetic interference, ensuring the glove remains portable. The sensor employs a soft shell and liquid core to detect various types of deformation, effectively demonstrating its ability to respond to a range of external stimuli. They also investigate in detail the sensing performance of a soft fiber optic sensor under three deformation models: elongation, bending, and pressing. COMSOL Multiphysics, a widely used finite element analysis software, was employed to simulate the sensor's behavior under these deformations. COMSOL enables precise modeling and simulation of complex physical phenomena, making it particularly suitable for quantitatively calculating optical losses under varying forces. The output power loss in decibels is defined as:

$$10 \log_{10} \left(\frac{I_0}{I} \right)$$

For elongation, soft fiber optic sensors exhibit a change in power loss, as the longer path generates more attenuation. Due to the diffusion of LED light, light leakages occur, but the majority propagates along successfully [42]. As for bending, it also causes intensity loss in the sensor, where the curvature results in a change in the angle of incidence between the light ray and the reflecting plane, hence leading to refraction and power loss. Lastly for pressing, a small force applied to the fingertip region causes significant deformation within the optic fiber, making it suitable for pressure sensing, all due to the low Young's modulus of the elastic material that makes up the fiber [42]. Therefore, they use a controller block consisting of a micro-controller (STM32F407), a six-axis gyroscope (JY901), a Bluetooth module (HC-05), and signal amplifiers (TLC272IDR) connected to five optical fiber sensors. For testing, 36 commonly used hand gestures as the fundamental elements of communication from American Sign Language (ASL), consisting of 26 alphabets and 10 numbers, were selected. Zhu et al. argue that unlike TENG sensors, fiber optic sensors can recognize hand posture in real-time, regardless of whether the hand stays stationary for 4 seconds or not [42]. By integrating gyroscopes for dynamic motion measurement, the sensing system can capture a wide range of static and dynamic hand gestures. Machine learning was utilized for data processing, enabling the smart glove system to accurately recognize 10 digits, 26 letters, 18 words, and 5 sentences. The system achieved an accuracy of 98.6% for static gestures and 95% for dynamic gestures.

According to Bukhari et al. in [10], SignSpeak utilized 2.2-inch flex sensors to measure bending movements by monitoring resistance changes through a voltage divider. For contact detection, conductive plates were connected through pull-up resistors. Contact was recognized when two plates touched, which altered the state of the sensor from its initial value, indicating a contact event. The ADXL 345 accelerometer was employed to detect hand acceleration, recognize the letter Z, and distinguish between I and J. They also argue that the location of the sensors in sign language recognition technology matter a lot. Since ASL was the reference, the sensors were strategically placed to ensure efficient gesture recognition while minimizing the number of sensors. This approach allowed for maximum freedom of hand movement while maintaining accurate detection of gestures. A total of 21 sensors were used; namely, 9 flex sensors to measure bending movements, 11 contact sensors to detect finger interactions, and 1 sensor to measure acceleration, ensuring comprehensive coverage of both static and dynamic hand gestures. In SignSpeak, different flex sensors were designated to help in distinguishing certain sets of letters. Flex sensors were placed to distinguish between specific gestures, while contact sensors were used for recognizing other letters, including T, N, M, U, and V. The GND

sensor on the back of the index finger was used to detect R.

A study conducted by Ambar et al. in [4] used 5 units of 4.5-inch Spectra Symbol's flex sensors on the back of each finger connected in parallel with 220-ohm resistors and a GY-61 accelerometer sensor on the back of the hand. The combination of 5 analog flex sensor inputs from the flex sensors and the 3 analog inputs from the accelerometer were connected to the Arduino Uno's Analog to Digital converter where they were converted to 10 bit digital data. An HC-08 Bluetooth module was used to send the processed data to the android application.

Another study by Lee and Lee discussed a method that uses flex and motion sensors to measure finger flexion and hand orientation, respectively [22]. Dawane and Sayyed implemented five flex sensors on a glove, each corresponding to a finger, to recognize hand gestures by comparing the motions with those stored in a motion database [12]. Similarly, Preetham et al. introduced a technique that employed flex sensors to map sensor data to a character set, utilizing a minimum mean square error (MMSE) algorithm for gesture recognition, with the results shown as text on an LCD screen [31]. Patil *et al.* in [26] further refined this approach by dividing the bending of each sensor into three categories: a full bend (finger closed), a partial bend, and a straight position (finger open). Each ASL letter is then associated with these flexions for template matching. Physiological sensors, such as sEMG, have also gained popularity for gesture recognition. Yu *et al.* in [24] introduced a score-based sensor fusion method using four sEMG sensors and a three-axis accelerometer, which was linked to a mobile application to enable real-time gesture-based interaction. Their study involved a set of 4 small-scale gestures, 15 large-scale gestures, and user-specific personalized gestures. Similarly, Wu *et al.* utilized four sEMG sensors along with an inertial sensor positioned on the wrist to recognize 40 frequently used ASL words, concluding that a single sEMG sensor on the wrist was sufficient for recognition [39]. In a further study, Wu *et al.* integrated data from both an inertial sensor and sEMG sensor in a wearable system to identify 80 ASL signs, processed through a support vector machine classifier using a chosen feature subset [38].

A study by Ji et al. in [17] developed a DataGlove that uses a motion acquisition node (MAN) to capture the sensor values for 20 Chinese sign language gestures, and a data aggregation node (DAN) that sends the received data from the MAN to the recognition system through Wi-Fi. The MAN consists of 16 Inertial Measurement Unit sensors (MPU9250). The MPU9250 is a 9-axis motion sensor that combines 3-axis gyroscope, 3-axis accelerometer and 3-axis magnetometer.3 IMU sensors were placed on each finger. They were connected using soft wires to ensure flexibility of each finger. One IMU sensor was placed at the back of the hand to obtain the motion information of the hand as a whole. The data from each IMU was 18 bytes long and was then sent to the DAN using an SPI protocol (Serial Peripheral Interface). The DAN was made of an MCU (STM32407VGT6) that recieves the data from the MAN and concatenates them into 289 bytes of data (18 bytes for each IMU and 1 header byte for packet verification). This data is transmitted to a Wi-Fi module which in turn sends the data to an ESP8266 through TCP.

2.3 Detailed Review of Hardware Components

Patil, Pendharkar, and Gaikwad cover the hardware aspect of ASL detection system which is built using a sensor glove setup and a microcontroller [26]. The over is as follows:

1. **Sensor Glove:**

(a) The glove is equipped with five flex sensors, one for each finger.

(b) These sensors detect finger bends by measuring resistance changes based on each finger's position.

(c) The glove material is non-conductive latex, ensuring that the sensors align correctly and can capture gestures accurately.

2. **Microcontroller unit (MCU):**

(a) The ATMEGA-16 microcontroller is used to process the data from the glove.

(b) It features 16K bytes of flash memory, 1KB SRAM, and an 8-channel 10-bit ADC (Analog-to-Digital Converter).

(c) The ADC converts the analog signals from the flex sensors into digital values, which correspond to different ASL gestures.

To connect the above components and display the results on the LCD, we do the following steps: In the ASL detection system, the LCD display and flex sensors are connected to the microcontroller (MCU), which acts as the central processing unit, handling both input from the sensors and output to the LCD. Here's how the connection works:

- **Flex Sensors to Microcontroller:**
  - Each of the five flex sensors, mounted on the glove fingers, connects to separate analog input pins on the microcontroller's Port A (e.g., PA0-PA4).
  - When a finger bends, the corresponding flex sensor changes its resistance, creating an analog signal.
  - These analog signals are fed into the ATMEGA-16's Analog-to-Digital Converter (ADC), converting them to digital values, which the MCU interprets as different gesture codes.
- **Microcontroller to LCD:**
  - The MCU processes the digital values and compares them to a pre-stored database of ASL gestures (letters or commands).
  - Once a gesture is recognized, the MCU sends the corresponding ASCII character or text to the LCD.
  - The LCD is connected to the MCU through digital I/O pins, typically via a parallel interface or a serial connection, depending on the LCD model.
- **Communication and Display:**
  - When the MCU identifies a matching gesture, it sends the corresponding output to the LCD in ASCII format.
  - The LCD then displays the recognized letter or word, translating the sign language gesture into readable English text.

The proposed system in [25] had built a glove-based solution that interprets hand gestures to aid communication for deaf and aphasia individuals. Key components and their roles are as follows:

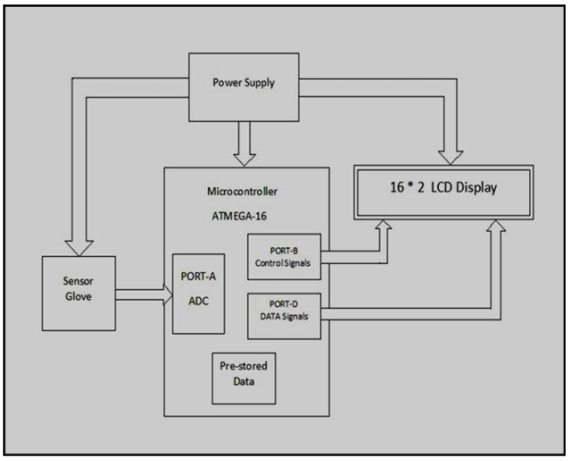


Figure 3: Connection of sensors and LCD to each port of MCU.

1. **Flex Sensors:** The glove contains five flex sensors, each attached to a finger, which measure resistance changes corresponding to finger bending angles. Resistance values, ranging from 10 kΩ to 25 kΩ, are converted to digital values by the Arduino Nano's ADC.

Figure 4: Flex sensor

2. **MPU6050 Accelerometer:** A triple-axis MEMS accelerometer and gyroscope detect hand orientation and motion on the X, Y, and Z axes, enabling accurate gesture recognition.

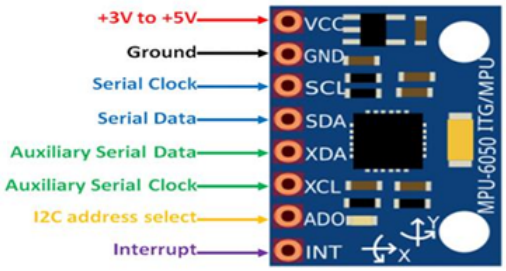


Figure 5: Accelerometer MPU 6050

3. **Arduino Nano Microcontroller:** The Arduino Nano, based on the ATmega328P, processes signals from the flex sensors and MPU6050 and converts them into corresponding text outputs based on pre-set threshold values.

4. **Heart Rate Sensor (MAX30102):** This sensor monitors heart rate and oxygen concentration by measuring the difference between oxygen-rich and oxygen-less

blood. The readings are used if a gesture indicates illness.

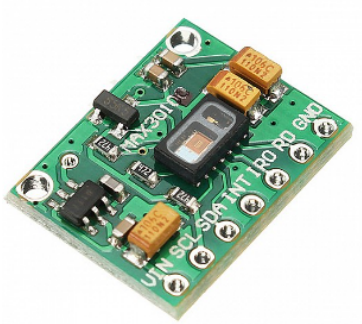


Figure 6: Heart Rate Sensor (MAX30102)

5. **Bluetooth Module HC-05:** The HC-05 module wirelessly transmits text and health data from the Arduino Nano to a mobile app for display.
6. **NFC (Near Field Communication) Module:** Used as an ID feature, the NFC module allows the user to store and share essential information like contacts and addresses.

Figure 7: Near Field Communication

Working Principle: Flex sensors detect gestures by measuring resistance changes due to finger bending, while the MPU6050 provides orientation data. The Arduino Nano processes this input, cross-references gesture data with preset values, and transmits the associated text to a mobile app via Bluetooth. If health-related gestures are detected, the MAX30102 sensor sends heart rate and oxygen data to the app as well. The below figure illustrates the connection of the above hardware parts:

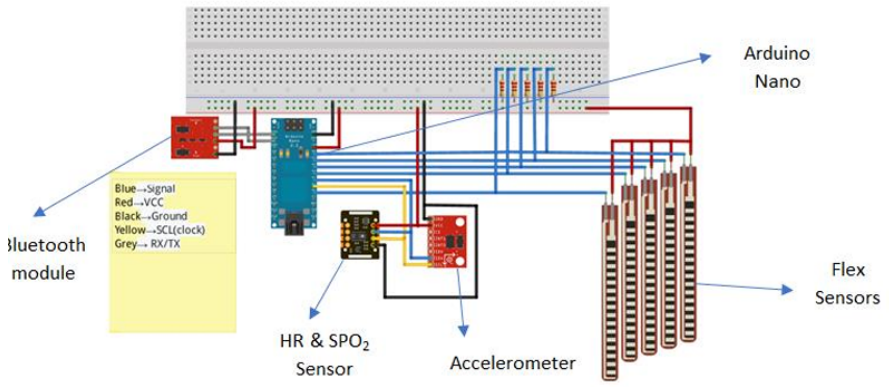


Figure 8: Circuit configuration of the proposed system

2.4 Dataset customization and preprocessing techniques

The subjects performing the sign language gestures constituting the training dataset executed calibration steps to cover differences in hand anatomy, such as non-contact, thumb with contact, index with contact, etc. [36]. Moreover, the signals studied by Tanyawiwat and Thiemjarus in [36] were preprocessed using median filtering, and a threshold-based segmentation method was applied. Each segment starts when at least one of the fingers is flexed and ends when all the fingers are stretched. A defined number of features were extracted for each data segment.

However, Rosero-Montalvo et al. acquired data by allowing ten people to use the glove and perform the SL gestures for one minute for each position [34]. The new data were stored in a matrix T, of mxn order, where m is the number of samples and n is the number of attributes that represent each data and one position label. Hence, m=5000 samples and n=5 attributes.The methodology involved dividing the data into training sets, test sets, data balancing, prototype selection comparison, and classifier development. Data balancing was performed for two reasons. First, the long processing time for a small data set on standard computers compiling machine learning algorithms (e.g., DROP3 and CHC). Second, the variability in SL gesture times among deaf individuals. The Kennard-Stone algorithm, commonly used for selecting samples from large datasets, ensures that chosen samples are evenly distributed across the predictor space. Using the Kennard-Stone method, a subset of samples was selected to create a new matrix U, containing the most representative points, of p×n, where p=1000. Afterward, a dimensionality reduction stage was applied to reduce the five dimensions to two.

Wu et al. in [40] collected 100 samples, dividing them into 60% for training, 20% for validation, and 20% for testing. To form a dataset with generalization ability, a volunteer wore the smart glove and repeated each alphabet letter 100 times. Additionally, two actions—full bending and full extension—were performed between two data points corresponding to two alphabet sign languages. The dataset was collected using a data-acquisition system, DAQ 970A.

Bukhari et al. in [10] collected data for SignSpeak using a Data Acquisition (DAQ) system capable of capturing data from flex and contact sensors. This DAQ system consisted of a PXI module, PXI controller, PCI chassis, and software.

The dataset for the Data Glove was built by collecting gesture data from seven participants. They were instructed to face north for three to four seconds to calibrate the initial attitude and wait for the glove initialization to complete. Participants were then prompted by a screen to complete each sign language action and let their arms sag naturally after each gesture. Each gesture had to be completed within four seconds, followed by an interval of two to three seconds before starting the next. The 3D raw acceleration signals and angular velocity signals obtained from the MANs, Motion Acquisition Nodes, were combined with pose quaternions into a 160-dimensional data structure. The sign language data were manually segmented by comparing them with high-frame-rate video. Since the MANs sampled data synchronously, only the back-of-hand MANs required segmentation. The time-series data were further segmented using a sliding window sequence of 400 sample points, corresponding to four seconds of data. The window slid forward by 20 sample points (0.2 s) at each step. Feature extraction from the 160-dimensional time-series resulted in 1280-dimensional feature data, incorporating variables from time-domain and frequency-domain analysis, such as mean, standard deviation, skewness, kurtosis, corrugation factor, quartile, spectrum peak, and peak frequency [17].

According to Wen et al. in [37], a set of 50 commonly used gestures was demonstrated, covering a wide range of daily-life signs. Gesture data were collected as triboelectric signals captured using an Arduino MEGA 2560 microcontroller. The data were then utilized to generate training and testing datasets for segmentation and non-segmentation methods. Each gesture in the dataset includes 100 samples, ensuring comprehensive representation. Correlation analysis of gesture signals included extracting average signal values for each gesture to facilitate comparison. Beyond individual gestures, 20 daily-used sentences were analyzed, revealing high correlation coefficients among some pairs. The dataset split allocated 80% of samples (4,000 word samples and 1,600 sentence samples) for training and 20% (1,000 word samples and 400 sentence samples) for testing.

As stated by Alosail et al. in [2], data acquisition involved collecting 200 instances for each ASL alphabet (excluding J and Z due to movement) and for ArSL alphabets. Features included readings from five flex sensors (Flex0 to Flex4) and three accelerometer axes (X, Y, Z). Data preprocessing transformed raw data into a clean and structured format by handling missing values and labeling the data accurately. This preprocessing step minimized errors and prepared the data for model training and testing. The dataset was divided such that 80% of the data (160 instances) was allocated for training, and the remaining 20% (40 instances) was reserved for testing [2].

2.5 Machine learning and Gesture Recognition Techniques

The choice of an optimal and accurate machine learning algorithm is crucial to ensure the correct gesture recognition especially for more complex signs that have some similarities with other signs. The dynamic time warping(DTW) algorithm was used by Chenghong et al. in [11] to measure waveform similarity. The DTW algorithm calculates the similarity of time-series data using Euclidean distance. The equation is used to determine the weighted sum used to compute a similarity score in the DTW algorithm is as follows:

$$DTW = \alpha DTW(WS) + (1 - \alpha)DTW(B) \tag{1}$$

where both DTW(WS) and DTW(B) are weighted sums of the data gathered from the sensors and the weights are normalized accordingly. Equation (1) combines data from 2 different sensors. The first sensor corresponds to the 10 bending sensors(sequence B) and the second sensor corresponds to the 2 WonderSense sensors(sequence WS).The weights in the equation are assigned to prioritize specific parts of the sequence.This machine learning algorithm resulted in an 85.21% average recognition rate [11]. Jeon et al. stated that the dataset for the glove used was collected from 11 right-handed subjects of varying hand sizes consisting of eight static and 11 dynamic gestures [16]. From each subject, 20 isolated samples of each gesture and 20 continuous samples are acquired. Splits of the dataset with five random seeds are used to conduct the experiments five times and analyze the results where min-max normalization was applied to each of the ten channels of the data.

On the other hand, Anetha and Parvin in [6] mentioned that a Levenberg–Marquardt back propagation algorithm was used for training the ANN model for gesture recognition which includes the selection of the first training pair,the application of the input vector to the net,the calculation of the net output,the comparison between the target and actual output to find the error, the modification of the weights to reduce the error.The steps of this algorithm are repeated until an accepted error is obtained. For every ASL

sign, Anetha and Parvin used a total of 45 features which are extracted from five different signers: 25 for hand shape,10 for hand movement (x,y,z) and 10 for EMG feature sequences.These extracted features are used as input to the network.The ANN trained and tested for two different data sets,single-user data and multi user data using the right hand, proved a recognition accuracy of over 95. Pezzuoli et al. argued that the necessity to use sophisticated machine learning models such as HMMs and LSTMs in order to take into account the spatio-temporal structure of the data for dynamic signs translation [28]. Traditional gesture representation methods often include data such as Histograms of hand position, configuration and velocity(Information about the shape of the hand and how fast it moves),time and distance of gestures(How long the gesture takes and how far the hand moves).However , in order to eliminate the usage of sophisticated machine learning algorithms, Pezzuoli et al. in [28] used feature vectors which represents gestures using coefficients from curves that are fitted to the sensor data. This feature vector is a fixed-length vector that encodes both spacial and temporal characteristics rather than the raw sequence of data points and is independent on the sign's duration. Pezzuoli et al. in [28] compared 5 classifiers for signs translation implemented using the Python library Scikit-learn. The results of classification accuracy and time for the nearest neighbors classifier using a K-nearest algorithm (K=3), the linear support vector machine (SVMs) using a C-support vector classification ($C = 0.025$), the random forest classifier (100 trees), the neural network (logistic sigmoid function), the naive bayes classifier using a Gaussian Naive Bayes algorithm are collected and compared. The random Forest classifier resulted in the highest sign recognition accuracy which is 99.7%. Also, Tanyawiwat et al. in [36] used Multivariate Gaussian distribution with diagonal covariance matrix as the classification model for sign language recognition. The study accompanies the classification model with a multi-objective Bayesian Framework for feature Selection (BFFS) to ensure the enhancement of the recognition accuracy and fault tolerance. The BFFS approach is a filter-based approach used to differ between features based on the expected area under the ROC curve.Using this model of classification along with the 21 features from the sensory glove and the multi-objective BFFS yields an accuracy of 77.9%. Moreover, Wu et al. used an ANN model (PyTorch library) with an architecture composed of 16 input nodes presenting the number of yarn sensors used in the glove, two hidden layers with 100 nodes each, and 26 output nodes [40]. A ReLU function was used for the two hidden layers while a Softmax function was used for the output layer. The ANN model was trained through backward propagation using the stochastic gradient descent method and the cross entropy function as the loss function. The learning rate was periodically adjusted using a learning rate scheduler StepLR.

However, Jeon et al. in [16] proposed a system of both static and gesture recognition (Figure 9). The first step of the process consists of a pre-classification step in which gesture patterns from a continuous stream of ten-channel time-series data are first segmented and then classified as static or dynamic.

Gesture segmentation and pre-classification are conducted using a DP model, which is a Long Short-Term Memory (LSTM)-based classifier. The DP model classifies each time step into one of three categories: non-gesture, static gesture, or dynamic gesture. LSTMs are particularly suited for this task as they are designed to handle sequential data by capturing long-term dependencies between data points, overcoming the vanishing gradient problem. The DP model processes data sequentially, using information from previous time steps to predict the state of the current time step. It consists of a three-layer LSTM. To produce meaningful gesture states, the output of the LSTM layers is passed through

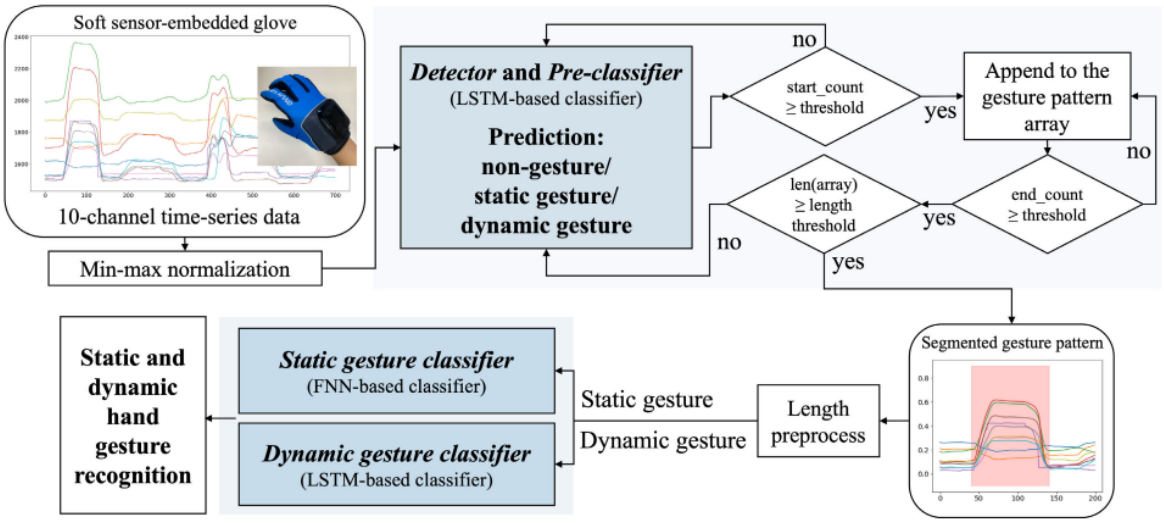


Figure 9: Static and dynamic hand gesture recognition system for a soft-embedded data glove

a dropout layer for regularization, followed by two fully connected layers with rectified linear unit (ReLU) and softmax activation functions. To ensure uniformity in processing, the segmented gesture patterns are adjusted to have a constant length N, regardless of the speed at which the gesture was performed. This is achieved by inserting or removing data points as needed. Based on the pre-classification results (static, dynamic, or non-gesture), the extracted gesture pattern is routed to the appropriate classifier. For static gesture recognition, a three-layer Feedforward Neural Network (FNN) is used. This network comprises a dropout layer, two fully connected layers with ReLU activation, and an output layer with a softmax activation function. The FNN is suitable for static gestures as it processes each input independently, without considering temporal dependencies. For dynamic gesture recognition, an LSTM-based classifier is employed. This classifier shares the same architecture as the DP model but operates as a many-to-one LSTM, predicting the gesture based on sequential data. LSTMs are ideal for dynamic gestures as they leverage their memory cells to capture and retain temporal features over time. The proposed system, which integrates these components, achieved an overall accuracy of 99.2% [16].

On the other hand, Lee et al. in [23] utilized the same soft sensor glove as the the one used by Jeon et al. in [16] but developed a different approach through three algorithms to achieve real-time dynamic gesture recognition: a gesture spotting algorithm to estimate gesture progress sequence (GPS),a sequence simplification algorithm to remove data with non-notable changes,and a gesture recognition algorithm as shown in 11.

The GPS is a scalar between 0 and 1, when the gesture is about to start the GPS is close to zero whereas it will be close to one when the gesture is about to end. The algorithm used for GPS estimation consists of of two long-short term memory (LSTM) layers, six fully connected layers, and one output layer. After generating the training data, the gesture spotting algorithm was trained with Adam optimization algorithm and a learning rate of 0.001. In addition, the sequence simplification algorithm reduces speed variation in gesture sequences by extracting key points and minimizing the computational cost for gesture recognition. It identifies the sensor with the greatest range of motion

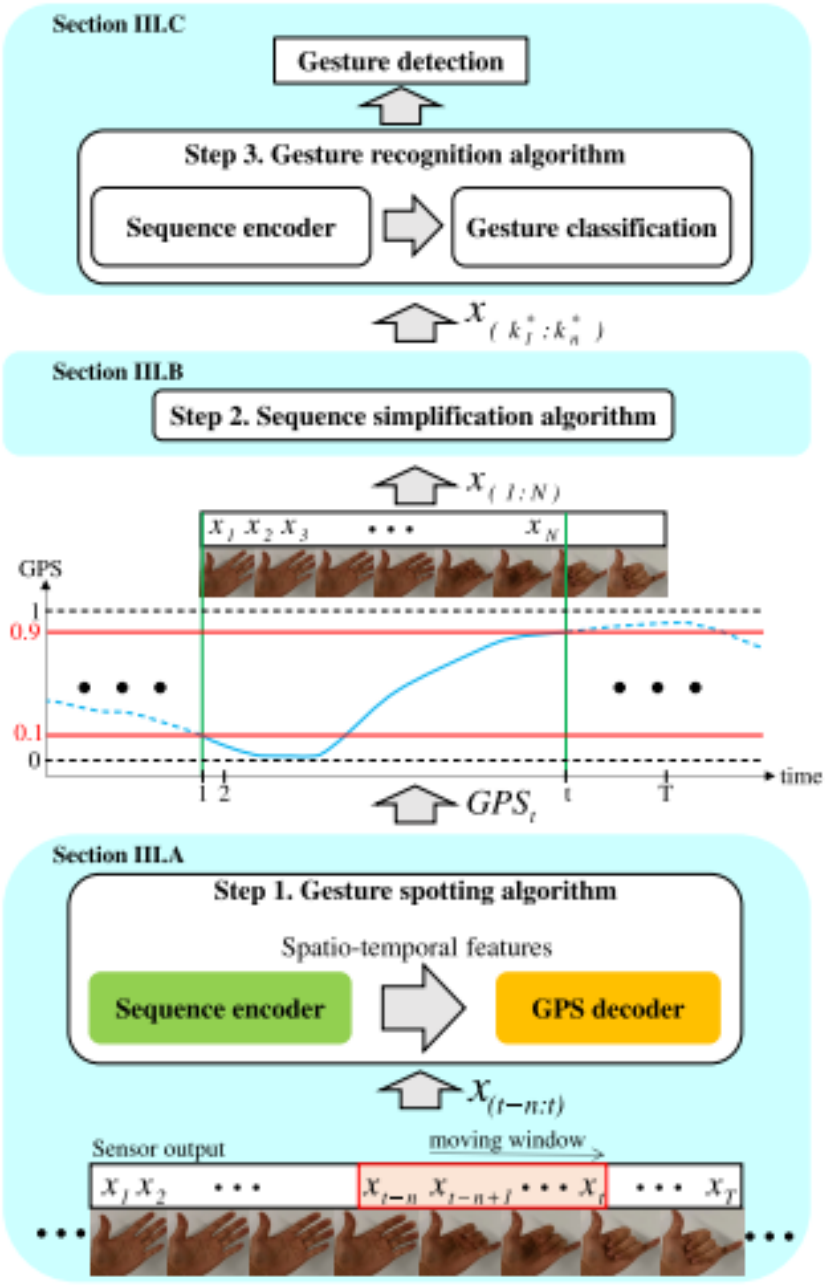


Figure 10: Static and Dynamic Gesture recognition system proposal with GPS

among the overall ten sensors of the soft glove, calculates a line between its start and end points, and finds the data point furthest from this line. This process divides segments by selecting points with maximum distance from each segment line until a set tolerance is reached, capturing essential inflection and concave/convex points and removing the others. The gesture recognition algorithm consists of two LSTM layers with 64 hidden units for each layer, three fully connected layers with 64 hidden units for each layer, and one output layer that has an output dimension of 11. The activation function of the three fully connected layers is ReLU, and a softmax activation function is used for the output layer. The gesture recognition algorithm considers the gesture sequence from initiation to the moment that the GPS value reaches 0.9.The cross validation result showed 100% accuracy for the training set and 98.5% accuracy for the validation set [23]. Montalvo et al. in [34] studied two prototype selection algorithms to select the best one, considering removed instances (RI), time execution (TE), and classifier accuracy (CA). For RI, CHC has removed 515 instances (68.8%), while DROP3 discarded 482 instances (64.26%) of the training set of 750 points. In CA, CHC obtained 86.8%, while DROP3 obtained 87.8%. In terms of TE, CHC's execution time is significantly lower than that of POP3's. The final training matrix is V of qxn, where q is 268 instances. The classifier's performance was 86.7% when using the training matrix V (which involved applied prototype selection), and 89.2% when using the matrix U. Hence, Montalvo et al. argued that CHC is the most adequate in sensor data, due to removal of instances, classifier accuracy, and its ability to work in R environment [34]. For the purpose of classification, data was needed to be arranged according to the difference in features in each gesture as mentioned by Bukhari et al. in [10]. During the assembling of SignSpeak, Gentle AdaBoost, a machine learning algorithm for Adaptive Boosting,was tried to build ensembles of classifiers with great performance, all to weigh the data instead of sampling and discarding it. Also, Bukhari et al. [10] mentioned that although gestures were efficiently and accurately classified using Gentle AdaBoost, it was more complex than SignSpeak needed, which is why they applied a simpler algorithm, Principle Component Analysis. A total of 26 gestures were trained, with 20 recordings of each consisting of 250 samples each. Averagely, the final set was a 520 x 17 matrix where 17 was the number of dimensions. Principal Component Analysis (PCA) was applied for classification and feature extraction, aiming to reduce dimensionality while retaining as much class-discriminating information as possible. PCA's input was a data matrix n x p where n corresponds to observations (520 for SignSpeak training set) and n is the number of variables (number of sensors , 17), while PCA's output is p x p matrix of Principle Coefficient and loadings, that were determined by MATLAB built-in functions and commands. For input,

$$\text{Input\_PCA} = \text{input} \times \text{PCA\_Vector}$$

was used. The coefficients, along with the corresponding means of the alphabets, were used to classify the real-time input by applying the Euclidean distance formula. This approach measures the distance between the real-time input and the pre-calculated means, allowing for accurate classification based on the closest match [10]. The $(p \times p)$ matrix of PCA coefficients stored in the microcontroller was used to match the real-time input, determining a valid gesture. Once the program starts, the Arduino continuously takes input from all sensors attached to the glove until it detects a stable input. A stable input is defined by the condition:

$$\text{Minimum} - x \leq \text{Maximum} \leq \text{Minimum} + x$$

The input is received in binary format, which is then converted into voltage using the formula:

$$\text{Voltage} = \text{bits} \times \left(\frac{3.3\,V}{1023}\right)$$

The real-time inputs are processed using PCA coefficients, and their corresponding Euclidean distances are calculated. The gesture with the shortest distance is deemed the most valid. After identifying a valid gesture, the accelerometer is activated to record any potential acceleration. Lastly, when tested and When more training sets were taken, accuracy was found to be around 92% for an untrained user, which made PCA best for feature extraction [10].

The below studies explore and compare various machine learning models and techniques aimed at enhancing sign language recognition. The study conducted by Ji et al. in [17] used a bidirectional long-short-term memory (Bi-LSTM) based on LSTM that combines both the forward and reverse information of input sequences, resulting in better performance in sequence labeling. Due to the differences in sequence lengths in sign language samples in practice, the model risks returning poor recognition performance. Therefore, the study introduced an attention mechanism to break the problem of fixed vector length in Bi-LSTM and give the corresponding weights according to the characteristics of the sequence to clearly highlight key information, which helps the model make accurate recognition after improving its training efficiency.

The model was made of an input layer, two Bi-LSTM hidden layers, two Dropout Layers, an Attention layer, a fully connected layer and an output layer. The learning rate was set to 0.001, the epoch to 100 and the batch size of the training sample was 256. After training and testing, the model's accuracy was recorded to be 98.19%, precision 98.39%, recall 98.03% and F1-score 98.15%.

For the application of the sign language glove developed by Wen et al. in [37], the CNN model achieved optimal performance with 5 kernels, 64 filters, and 4 convolutional layers, as shown in Figure 16. This setup allows for effective feature extraction from the time-series data, leading to high recognition accuracy. After passing through the CNN architecture, each gesture signal undergoes feature extraction, which transforms the original high-dimensional data into a more manageable format. The final output of the CNN model is stretched from 3000 data points to 3584 features through convolution and pooling operations. They also stated that Principal Component Analysis (PCA) is used to further reduce the dimensionality of the data, making it easier to visualize and analyze. The CNN model demonstrates high classification accuracy, achieving 91.3% for individual word recognition with a training-to-testing ratio of 80:20. This result is illustrated using a confusion matrix that highlights the model's ability to differentiate between the 50 gesture classes. For sentence recognition, the model effectively handles longer, unsegmented time-sequence signals. By using supervised learning, it achieves an even higher accuracy of 95%, as sentence signals tend to have more distinguishable features compared to individual word signals. Wen et al. also argued that despite high accuracy, the non-segmentation approach of the CNN model is limited to recognizing only the gestures and sentences that are part of the training dataset. It cannot generalize to new or unseen sentence structures, which comes from the methodology where there's no built-up relation between word units and sentences. Hence, Wen et al. introduced segmentation method which enables effective real-time recognition and the ability to identify new or never-seen sentences. These sentence signals are broken into smaller data fragments using a sliding window mechanism. This window identifies different types of

signals, such as intact words, incomplete words, and background signals. The fragments are then labeled based on the principal components of the data in each window. If the intact word signal covers more than 50% of the window, it gets labeled as that word; otherwise, it may be tagged as background noise. This model classifies all fragments into word units or background signals. Although it achieves a reasonable recognition rate, the presence of random empty signals affects its performance. To improve accuracy, a hierarchical model first filters out background noise and then classifies the meaningful word fragments to enhance word recognition and to lead to better sentence reconstruction. The hierarchical classifier showed improved accuracy and reliability. It was particularly more effective in handling real-time sentence recognition and predicting new sentence labels, with an average correct rate of 86.67%, compared to 60% for the single classifier.

Another several algorithms were employed and discussed in the study done by Alosail et al. in [2]. Logistic Regression (LR) is used for classifying discrete data types, including binary outcomes with Binary Logistic Regression, as well as predictions for multi-class values with Multinomial and Ordinal Logistic Regression. Experiments were then categorized based on the sensor inputs used: flex sensors, accelerometers, and a combination of both. The initial implementation using only flex sensors revealed that the Random Forest (RF) classifier achieved the highest accuracy of 94.5% for ASL and 91.8% for ArSL, highlighting the challenge of recognizing ArSL due to its greater number of signs and their similarities. Incorporating an accelerometer shifted the performance, with the Multilayer Perceptron (MLP) classifier performing best, achieving 92.4% accuracy in ASL and 98.4% in ArSL, due to the accelerometer's ability to capture rotational and movement data more effectively. The final experiment, utilizing both sensor types, further improved accuracy, with the RF classifier reaching 99.7% for ASL and 99.8% for ArSL, suggesting that the combination of features from both sensors enhances recognition capabilities. A feature importance analysis indicated that accelerometer features were more significant than those from flex sensors, particularly for ArSL. Real-time testing revealed a drop in accuracy compared to theoretical predictions, with practical implementations showing 76.22% for ASL and 79% for ArSL, illustrating the impact of factors like improper readings and calibration issues. It was found by Alosail et al. in [2] that the accelerometer contributed more to the ML performance than flex sensors.

2.5.1 Deep Learning Architectures

This subsection provides an in-depth exploration of various deep learning algorithm structures used in sign language recognition. Those focusing on broader findings and implications may prefer to proceed to subsequent sections.

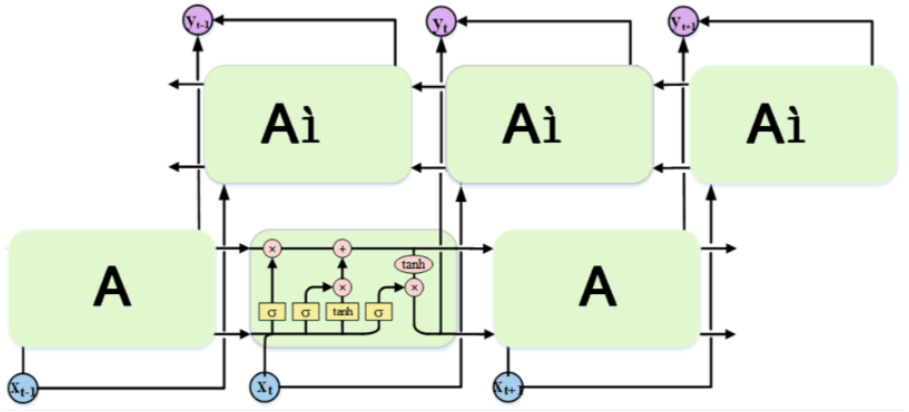


Figure 11: Bidirectional Long-Short-Term Memory (Bi-LSTM) Architecture

Hidden Markov Model (HMM), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM) have been fundamental methods in handling sequential data. These models are advantageous in gesture recognition because they can retain and update contextual information through time, which is crucial for interpreting gestures accurately as cited by Wen et al. [37]. For further illustration of the above algorithms, Boulard and Morgan in [14] explained about HMM and Kim et al. in [18] discussed the use of several deep learning models which are: RNN, LSTM and GRU for sign language and explained their strengths and weaknesses.

HMM: A Hidden Markov Model (HMM) is a statistical model for systems with hidden states that influence observable outputs. The model assumes that the future state depends only on the current state (Markov assumption). It includes transition probabilities for state changes, emission probabilities for observed outputs, and initial state probabilities. Key challenges are evaluating the likelihood of observation sequences, decoding the most likely sequence of hidden states, and learning model parameters.

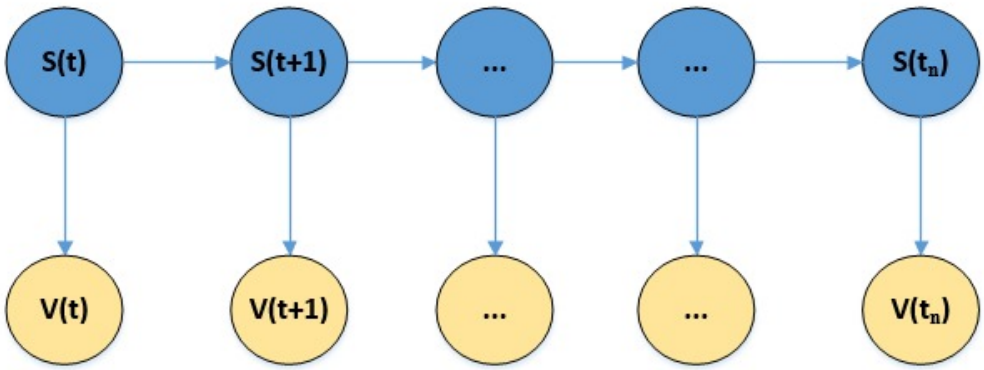


Figure 12: Architecture of HMM

RNN: This is the most basic form of recurrent neural network, suitable for short sequences. RNN processes data in units of cells with a simple structure, involving parameters that connect each cell in sequence. It mentioned that RNN stores information from the previous time step but struggles with longer sequences due to issues like the vanishing gradient problem. This makes it less suitable for handling complex patterns in sign language data, especially over extended sequences. In the study, RNN showed

the lowest accuracy and had difficulty managing the lengthier sequences required for sign language recognition.

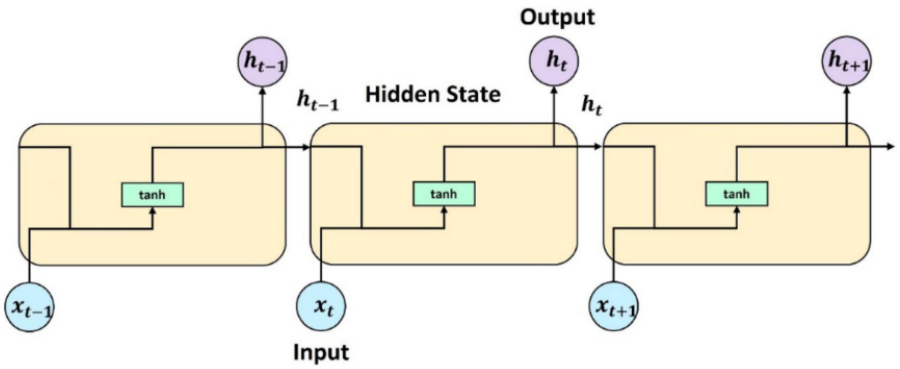


Figure 13: Architecture of RNN

LSTM: Designed to address RNN's limitations, LSTM networks incorporate memory cells that retain information over longer sequences, making them highly effective for tasks like sign language recognition, which involves temporal dynamics.Hence, LSTM is an improved model over RNN, with added components that help retain information over long sequences. Specifically, it mentioned that the addition of gates (Forget, Input, and Output gates) that regulate what information to keep or discard, allowing LSTM to manage dependencies over time. These gates enable LSTM to perform better on tasks that require understanding of longer temporal dependencies. The study found LSTM to perform well, though it required longer training times due to its complex structure.

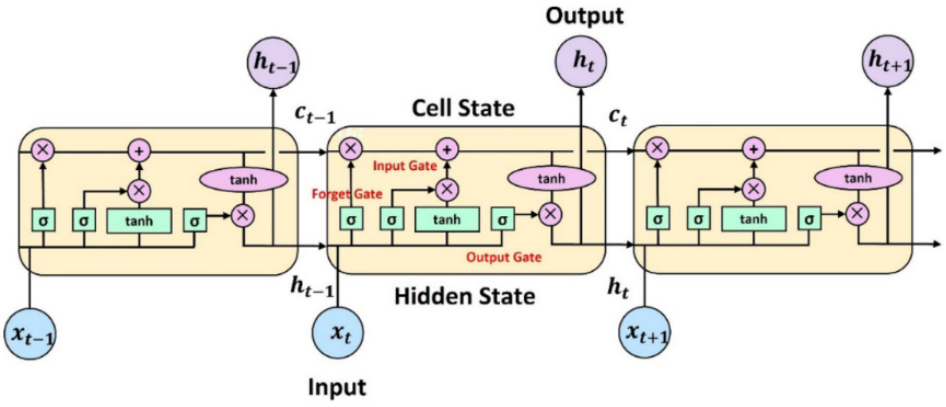


Figure 14: Architecture of LSTM

GRU: GRU is a type of recurrent neural network that, like the Long Short-Term Memory (LSTM) model, is designed to handle sequential data by managing dependencies across time steps. However, GRU is more computationally efficient than LSTM because of its streamlined architecture. GRU was developed to address the vanishing gradient problem, which occurs when trying to capture long-term dependencies in sequential data, as is common in tasks like sign language recognition.The study demonstrated that GRU outperformed SimpleRNN and achieved similar accuracy to LSTM but with shorter training times.

Wen et al. in [37] stated that Convolutional neural networks (CNN) are made to process data in the form of multiple arrays. 1D CNNs are specifically employed to recognize

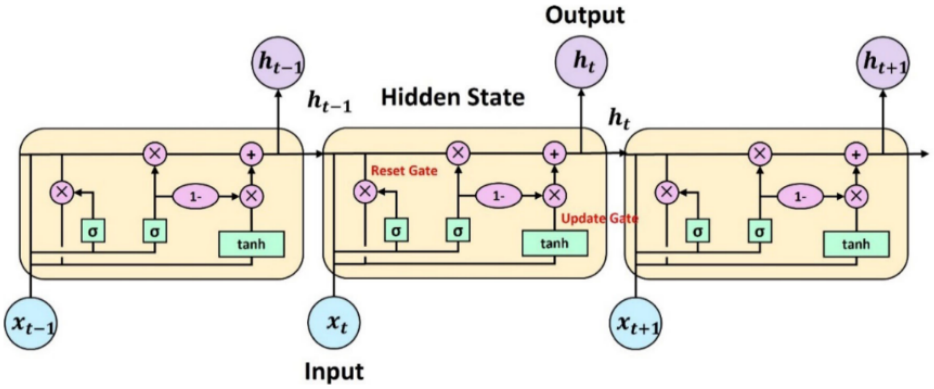


Figure 15: Architecture of GRU

human motion signals derived from triboelectric sensors. Unlike traditional methods, CNNs do not require segmented input, which means the models can process entire gesture sequences without manually breaking them down into individual components. The architecture optimization process involves tuning parameters like kernel size, the number of filters, and the number of convolutional layers.

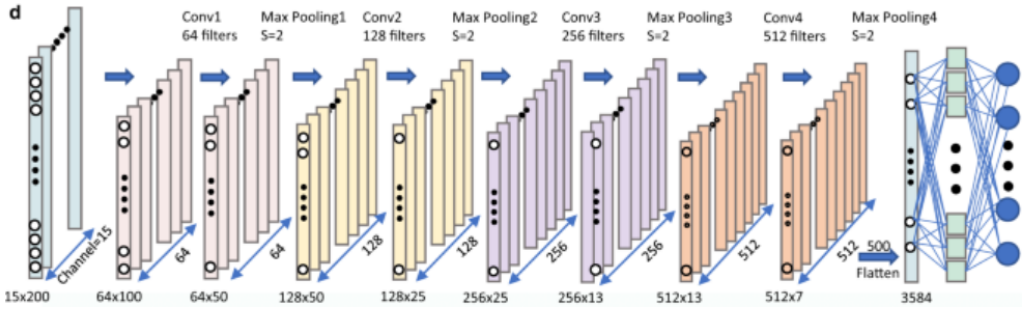


Figure 16: Architecture of CNN

3 Project Constraints

Defining project constraints is a crucial step in the decision-making process which dictates the feasible actions needed to achieve the project goals. Effective acknowledging the constraints tied to time, budget, location and resources limitations and restrictions early on enables alternative planning and resource allocation for a better realistic project completion. Three main project constraint categories are discussed below; namely, general project constraints, Sensor Integration project constraints, and the Audio-Visual Interface project constraints of the prototype.

3.1 General Project Constraints

The general project constraints mainly emphasize the accessibility and convenience of the prototype. The wearable glove should not be heavy or occupy a large space along the arm to ensure the user's comfort during long-term usage. Power consumption is also a significant consideration, as improper management can lead to the need for larger batteries, which are cumbersome to carry. Thus, power optimization techniques are

critical. The wearable prototype must have an acceptable mass, taking into account the number and type of sensors used as well as the material of the glove. Moreover, the project requires a fast sampling rate to handle interactions that need to be interpreted effectively, ensuring smooth operation and seamless gesture recognition. Tables 1 and 2 below describe the project's general constraints in detail.

Table 1: General Project Constraints - Part 1

	Implementation Scope	**Size Constraint**	**Mass Constraint**	**Power Constraint**
Description	The project is designed to be an accessible tool for deaf and hard-of-hearing individuals	The project size should not be larger than the average pocket size	The project is designed to have a light mass as a wearable to ensure user's comfort during extended periods of use	The project is designed to operate on an average of 5 hours per day before recharge.
Numerical Value		30 x 15 cm - Average adult hand size	$<$ 250 g per glove -Typical wearable device	4-6 operating hours per day
Foreseen Impact on the Project	The glove should be able to perform under different environmental conditions indoors and outdoors but not in extreme weather conditions such as rain and high heat	Restricts the number and choice of compact sensors and microprocessors. Allowed the usage of raspberry pi 4 (88 x 58 x 19.5 mm) as the processing unit	Limits the number of components and the material choice of the glove	Choose a high capacity, lightweight battery (3000-4000 mAh) for the desired runtime

Table 2: General Project Constraints - Part 2

	Accessibility Constraint	**Timing Constraint**	**Economic Constraint**
Description	The project is designed such that the users can choose the desired mode of communication (e.g., deaf user to hearing user or hearing user to deaf user). The input is sent to the processing unit for output display or sound output.	The system should sample data from all sensors at a high frequency to ensure accurate and real-time gesture recognition and translation. The processing time must be minimal to provide real-time feedback.	The project is designed to have a relatively low cost while balancing prototype accuracy and cost. Total cost < 250 USD.
Foreseen Impact on the Project	The glove is a user-friendly tool with diverse language coverage, accounting for non-English speakers in the MENA region.	The Raspberry Pi model is chosen to handle real-time sensor fusion without delay. High sampling rates might impact battery life.	The economic constraint impacted the choice of sensors and the microprocessor implemented.

3.2 Sensor Integration Project Constraints

This section focuses on the constraints of the implementation of the sensor glove based on the different sensor configurations discussed in the literature review. Comparison of the different methods viable to achieve this project such as the sensor combination: flex sensor - gyroscope - force sensitive resistor (FSR) and the sensor combination: flex sensor - accelerometer -sEMG sensor is discussed thoroughly in the tables below along with the decision made on each method.

Table 3: Sensor Integration Project Constraints - Method 1

Method	**Flex Sensor - Gyroscope - FSR**
Components per glove	• 5 flex sensors • 3 FSRs • 1 MPU 6050
Size per glove	• Flex sensor [Length x Width: 73.66 x 6.35mm] • FSR 402 [68 mm diameter - 56 mm length] • MPU 6050 [50 × 30 × 10 mm]
Mass per glove	• Flex sensor [0.27g x 5 = 1.35g] • FSR 402 [0.26g x 5 = 1.3g] • MPU 6050 [2g] • Total = 4.65g
Advantages	• Provides a comprehensive gesture recognition method • Multi-model Data: Combines different types of sensors with diverse data • Enhanced Accuracy: Cross-referencing multiple sensor inputs reduces gesture recognition errors • Low power consumption
Disadvantages	• Synchronization of multiple sensor data requires advanced signal processing techniques • Effective calibration requirements • Potential latency in real-time gesture recognition
Cost	**Total Cost per glove** = 13.5 x 5 (flex) + 6.5 x 3 (FSR) + 1.5 (MPU) = 88.5 $
Decision	YES

Table 4: Sensor Integration Project Constraints - Method 2

Method	**Accelerometer - sEMG Sensors**
Components per glove	• 1 3D accelerometer • 4 sEMG sensors
Size per glove	• sEMG sensor [Diameter: 20 mm; Thickness: 6 mm] • Accelerometer [Length x Width: 5 mm × 5 mm; Thickness: 0.9 mm to 2 mm]
Mass per glove	• sEMG sensor [15 g] • Accelerometer [2 g] • Total = 17 g
Advantages	• Good accuracy in terms of gesture recognition • Ambiguity reduction by collecting both internal signals and external physical movement • Lower latency in multi-sensor fusion by using fewer sensors
Disadvantages	• Bulkier and heavier set-up for dual and triple-sensor combinations • Requires proper electrode placement, which can be uncomfortable • Results in a lot of noise and may need significant preprocessing for accurate signal interpretation • Affected by muscle fatigue over time
Cost	**Total Cost per glove** = 4.25 $ (3D ACC) + 20 $ (sEMG) = 24.25 $
Decision	NO

3.3 Audio-Visual Interface Project Constraints

This section highlights the different solutions for the audio input system via the microphone, the visual display for text output, and the audio output system represented by the speaker. Each component plays a crucial role in ensuring seamless interaction and effective communication within the system.

Table 5: Audio-Visual Interface Constraints and Solutions

Component	Solution	Evaluation of Solution	Selected Decision
Microphone	**Dynamic Microphone**	Bulky and requires amplification.	**MEMS Microphone INMP441**
	Electret Condenser Microphone	Moderate size, less sensitive than MEMS.	
	MEMS Microphone INMP441	Compact size, high sensitivity, and ease of integration.	
Speaker	**Wired Speaker**	Requires direct connections, limiting portability.	**Miniature Bluetooth Speaker**
	Onboard Buzzer	Limited sound quality, suitable for simple tones.	
	Miniature Bluetooth Speaker	Compact, wireless, and offers good sound quality.	
Display	**LCD Display**	Heavier and consumes more power.	**I2C OLED Display 128X64**
	E-Ink Display	Low refresh rate, unsuitable for dynamic updates.	
	I2C OLED Display 128X64	Lightweight, high contrast, and suitable for real-time visuals.	

4 Standards and Codes

Before implementing the project, it was essential to comply with various standards and codes. These standards, established by international technical committees, ensure safety, quality, and compliance with regulatory requirements. They address specific aspects of the project, such as hardware design, software functionality, and user safety, and are crucial for its successful operation.

To enhance clarity and accessibility, the standards are organized in a table, highlighting their relevance to different components of the project.

Table 6: Standards and Codes - Sensors and Components

Category	Standard	Description
Environmental Safety	ISO/IEC 60529 – Degrees of protection provided by enclosures (IP Code)	Defines levels of protection for electronic devices, crucial for environmental conditions (e.g., dust or water resistance).
Sensor Accuracy	IEC 60747-14 – Semiconductor Devices	Specifies standards for the performance and testing of sensor components in electronic devices.
General Sensor Performance	IEC 61326-1 – Electrical equipment for measurement, control and laboratory use	Covers electromagnetic compatibility requirements for sensor operation and measurement accuracy.
Sensor Testing	IEC 62828 – Reference conditions and procedures for testing industrial and process measurement transmitters	Provides guidelines for sensor calibration and performance testing.

Table 7: Standards and Codes - Safety and Compliance

Category	Standard	Description
Device Safety	UL 60950-1	Standards for electrical safety in IT equipment.
Environmental Safety	RoHS (Restriction of Hazardous Substances)	Ensures materials used are eco-friendly and safe.
Product Safety	IEC 61010-1	Safety requirements for electrical equipment for measurement, control, and laboratory use.
Electromagnetic Compatibility	IEC 61000-4	Testing and measurement techniques for electromagnetic compatibility.
Material Safety	REACH (Registration, Evaluation, Authorization of Chemicals)	Regulations regarding chemical substances and their safe use.
Electrical Safety	IEC 62368-1	Safety requirements for audio/video, information and communication technology equipment.

5 Alignment with the Sustainable Development Goals (SDGs)

In alignment with LAU's commitment to sustainable development, **E-S.H.A.R.A** directly supports specific Sustainable Development Goals (SDGs) by addressing communication barriers faced by individuals with hearing impairments. By leveraging innovative wearable technology, the project promotes inclusivity, accessibility, and improved quality of life, as outlined in the following table.

Table 8: Alignment with the Sustainable Development Goals (SDGs)

SDG	Target	Contribution to the Project
SDG 4: Quality Education	Target 4.5: Eliminate gender disparities in education and ensure equal access to education for vulnerable groups.	The glove promotes educational inclusivity by enabling seamless communication for individuals with hearing impairments, improving access to quality education.
SDG 10: Reduced Inequality	Target 10.2: Empower and promote the social, economic, and political inclusion of all.	By facilitating communication between deaf individuals and non-sign language users, the glove empowers the hearing impaired and promotes social and professional inclusion.
SDG 3: Good Health and Well-being	Target 3.8: Achieve universal health coverage and ensure access to quality health care services.	The glove enhances communication between patients with hearing impairments and healthcare providers, supporting improved healthcare access and outcomes.
SDG 9: Industry, Innovation, and Infrastructure	Target 9.5: Enhance scientific research and encourage innovation in technology.	The glove leverages cutting-edge wearable technology to foster innovation in the tech and healthcare sectors, contributing to industry advancements.

6 Glove Hardware Design

In this section, we will discuss about our choice of hardware components for the sign language glove. We will go over the different options we had and explain the applicability of our final choice. In addition, we will also dive into the specifications of every piece of hardware equipment then explain about the connections between them in the system and the purpose of their positioning on the glove.

6.1 Hardware Selection

Our final design consists of three force sensitive resistors (FSR), the Raspberry Pi 4 with the MCP3008 10 bit 8-channel ADC, a MEMS microphone, a speaker, the inertial measurement unit (IMU) MPU6050, five flex sensors and the SSD 1306 OLED Display.

In the below subsections we will go over the thought process we went through for each hardware component to reach the final design decision.

6.1.1 Processing Unit

For the processing unit, we had a variety of locally available options such as Arduino boards, PIC microcontrollers, ESP32 boards, and FPGAs. Our task requires the processing unit to not only read and pre-process values from analog and digital sensors, but also run a trained neural network model that generates predictions based on the pre-processed input and perform conversions between text and speech. PIC microcontrollers have very limited computational power and very low memory and no built-in floating point unit, which makes them an invalid choice for our task. FPGAs possess more computational power and memory than PIC microcontrollers yet still not enough to run our neural network. In addition, FPGAs do not support deep learning software frameworks that run on GPUs and CPUs since they are programmed using hardware description languages such as VHDL and Verilog, which will require the conversion of our neural network model and optimizing it to hardware written code. Therefore, FPGAs are not a valid option for our task. Arduino boards are a perfect choice for reading values from both digital and analog sensors and preprocessing them. Some Arduino boards also possess WiFi and Bluetooth modules. On the other hand, ESP32 boards possess greater computational capabilities, consume less power, have more RAM, and have a stronger processor than Arduino boards. However, both boards lack the memory requirements to efficiently run a neural network model and make predictions on it, so they will have to offload that task to either another hardware component on the glove or through the cloud. Since we want to reduce the number of hardware components on the glove to avoid making it uncomfortable and since we want to make our predictions locally and not depend on an internet connection, the ESP32 and Arduino boards are not an optimal option.

The Raspberry Pi 4 is a small single-board computer that runs on the Raspberry Pi OS, a Unix-like operating system based on the Debian Linux distribution. It is capable of reading from digital sensors and performing complex operations. It has 2 GB of RAM, a Bluetooth module, a WiFi module and has 4 USB inputs, an Audio output jack and 40 GPIO pins that can read and write digital values and also handle I2C, SPI and UART protocols. It is the best choice we are left with for handling the speaker, OLED display and MEMS I2S microphone, and it has enough memory and computational power to handle our neural network model, making it the perfect choice of hardware for the processing unit. However, the Raspberry Pi does not include an ADC (Analog to Digital Converter) which is needed since our flex sensors and force resistive sensors (FSR) are analog. Therefore, we use the a light and compact ADC chip MCP3008 which contains 8 channels where every sensor is connected to a channel.

6.1.2 Sensors

The choice of the sensor-based glove instead of a camera-based system stems from the limitations of getting a very high quality camera as well as the requirement to use the vision based system in a good lighting environment which can effect the results. Our sensor-based glove consists of a total of 5 flex sensors, 3 force resistive sensors and 1 MEMS accelerometer and gyroscope (MPU 6050). The flex sensor is used to effectively detect the bending of the finger to recognize sign language hand gestures. Finger bending

is a crucial part in most sign language gestures which is why we chose flex sensors as our primary solution for detection. Flex sensors produce an analog output and it is connected in a voltage divider circuit to provide a corresponding voltage change as the resistance varies. The value of the obtained voltage can be used to quantify the degree of bending of the finger.

Since we are using a raspberry pi 4 as our processing unit, the flex sensor's analog output is fed into an analog-to-digital converter (ADC) which would result in a digital value. The MCP 3008 used is a 10-bit ADC and it samples the the analog voltage based on this 10 bit resolution therefore the output ranges from 0 to 1023 (0 to 2^{10} -1). The value is stored internally in the MCP 3008 as a binary representation and communicated via Serial Peripheral Interface (SPI) to the Raspberry Pi. The Raspberry Pi will read the received binary data as integer value. The FSRs are used to detect the pressure applied to the tip of the fingers. Some signs involve pressing fingers against each other or against the palm such as A,B,I,S etc. The FSRs therefore complement the flex sensors by measuring the magnitude of the force applied to enable the system to recognize a wider and more accurate range of signs. Some signs may appear visually the same and have similar values generated from the flex sensors bu the difference can be detected by the pressure level. For example, the letters O and C have similar shapes created by finger bending but differentiable nuances from the pressure level which is higher for the letter O and negligible for the letter C a shown in Figure 17.

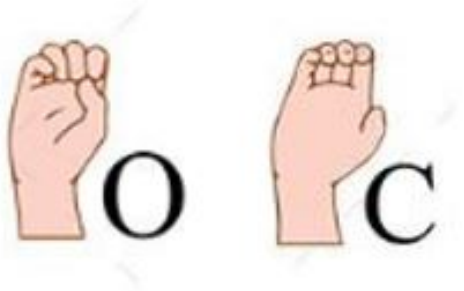


Figure 17: Difference between letter O and C signs

The difference between A and S as well can be detected by the difference in the pressure applied by the thumb as the thumb rests gently against the fingers in A but is firmly pressed in S to form a fist shape as shown in Figure 18. This can also be applied to intensity in sign language different gestures of greeting. Therefore, the addition of the FSR is crucial in our design. The sensor is only placed at the thumb, index and middle finger to decrease the cost of the system and after evaluation of the sign language gestures, we concluded that the essential placement with effective result in eliminating nuances is at the specified fingers.

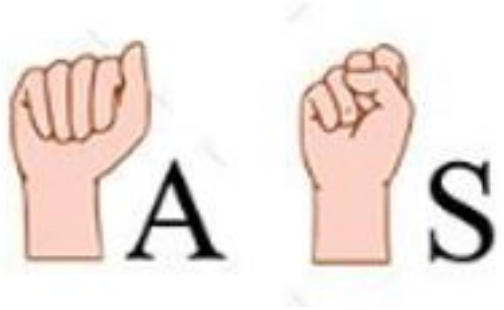


Figure 18: Difference between A and S signs

FSRs also generate an analog output corresponding to the force applied and the voltage is converted to a digital value using the MCP 3008 to be processed in the Raspberry Pi.The last element of our sensor-based design is the 3-axis gyroscope and 3-axis accelerometer(MPU 6050). Many sign language involve hand movements such as arcs or rotational gestures. The accelerometer captures linear movements such as translation in x, y, and z directions and the gyroscope captures the rotational movements such as wrist tilts. The addition of this 6-axis motion tracking device eliminates misclassification caused by the difference in motion between gestures. The MPU 6050 can be extremely useful in tracking the curved motion done when signing the letter J, as well as the zigzag motion involved to sign the letter Z as shown in Figure 19.

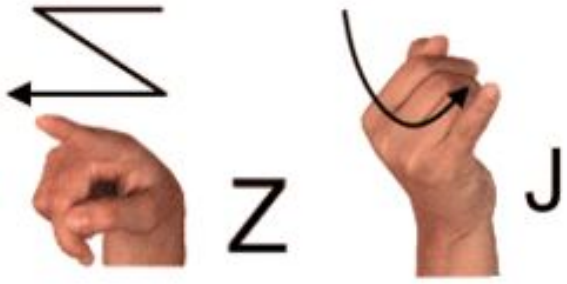


Figure 19: Motion in letters J and Z

The output of the sensor looks like the following: Accelerometer: [X:, Y:, Z:] - Gyroscope: [X:, Y:, Z:]. The accelerometer output values correspond to the acceleration in the X, Y and Z axes in terms of gravitational acceleration(g). The gyroscope output values correspond to the angular speed around the X, Y,and Z axes in terms of degrees per second. The MPU 6050 communicates over the I2C bus and the raspberry pi would read the data from the sensor's registers.

6.1.3 Audio-Visual Interface

OLED Display

The visual output of the system consists of an OLED screen display that can be used to output the alphabets, numbers and words recognized by the system and the machine

learning algorithm used in the deaf-to-hearing part of the interaction. The OLED display also supports graphical output, enabling the system to render visual elements such as basic animations of sign language hand gestures. This feature facilitates the speech-to-gesture functionality, providing support for deaf users who may not be proficient in reading. In addition, the output of the speech-to-text conversion is visualized on the OLED screen in the hearing-to-deaf part of the interaction. The I2C OLED screen uses two wires for communication: SCL or the serial clock and SDA or the serial data. I2C should be enabled on the Raspberry Pi in the interfaces options and the Pi can communicate with the OLED display via a set of libraries such as Adafruit-SSD1306.

Bluetooth Speaker

The auditory output of the system consists of a miniature Bluetooth speaker such as the Dewalt Bluetooth speaker. The steps for connecting the Bluetooth speaker to the Raspberry Pi are as follows [15]:

1. Installing Bluetooth and audio packages
2. Enabling Bluetooth on the Raspberry Pi
3. Sorting out Bluetooth by fixing the errors in the Bluetooth service on the Pi
4. Pairing and Connecting to the Bluetooth speaker

MEMS Microphone

The microphone in our system serves as the basis for the hearing-to-deaf communication. Therefore, our choice of microphone was the MEMS microphone INMP441. The INMP 441 is a digital microphone that communicates over the Inter-IC Sound (I2S) interface.Unlike the traditional analog microphones, the INMP441 sends digital data via the I2S protocol which reduces the need for ADCs. The microphone has a compact size, a low power consumption as well as high quality audio which makes it suitable for our wearable battery-powered application. To connect the microphone to the Pi, the I2S should be enabled in the interfacing options and audio libraries should be installed such as alsa-utils and PyAudio.

6.2 Wireless Data Transfer using ESP32

To establish wireless communication between the sensors on the second glove and the processing unit (Raspberry pi 4) located on the first glove, we chose the ESP32 microcontroller.

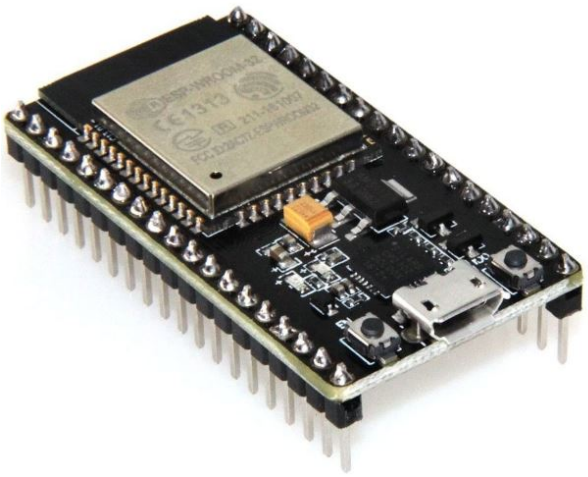


Figure 20: HiLetgo ESP32 ESP-32S

The ESP32 has built-in ADC channels, which reduces the need for the MCP3008 chip for the second glove.

The ADC pins of the ESP32 convert the analog signals of the flex and FSR sensors to digital values.The MPU-6050 communicates with the ESP32 via I2C protocol. The ESP32 collects the sensor data into packets and transmits the packets to the Raspberry Pi over Bluetooth Low Energy (BLE). Unlike Bluetooth that is always on, BLE remains in sleep mode constantly except for when a connection is initiated.

This establishes a power-efficient communication link with the Raspberry Pi where:

- **ESP32 as a BLE server:** Sensor data is gathered and made available for reading via BLE.
- **Raspberry Pi as BLE client:** The Raspberry pi will asct as the Central or client which will connect to the ESP32 and read the sensor data.

The ESP32 solution provides the following advantages in the system:

1. The use of BLE eliminates the need for wires between gloves, enhancing the user's comfort and practicality of the design for extensive use.
2. The use of I2C and the presence of multiple ADC channels allow efficient handling of data from all types of sensors on the second glove.
3. The integration of the ESP32 with BLE ensures low power consumption for extended use and constructs a lightweight unit.

For implementation, using ESP32 with BLE can result in challenges such as:

- Ensuring the ESP32 could handle the data influx from multiple sensors while maintaining BLE transfer rates. This can be done by increasing the polling interval (how frequently we read data from the sensors) to eliminate traffic over the BLE link and bottlenecks.
- BLE latency can be minimized by tuning the BLE connection parameters and packet sizes.

Raspberry Pi Connection:

For the Raspberry pi, we need to install BlueZ stack which is the official Linux Bluetooth protocol stack as well as the pybluez library for Python. In the code, the UUIDs and characteristic from the ESP32 code should be used as well as the MAC address of the ESP32.

ESP32 Connection:

For ESP32, we need to install the BLEDevice library to set it up as a BLE server.UUIDs and characteristic for the BLE service should also be configured in the server Arduino code [32].

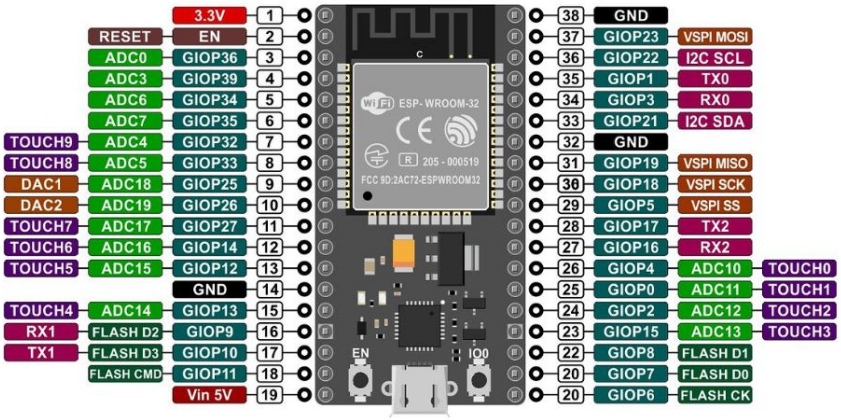


Figure 21: ESP32 pinout

6.3 Hardware Circuitry

The overall circuit of our design shown in the figure below:

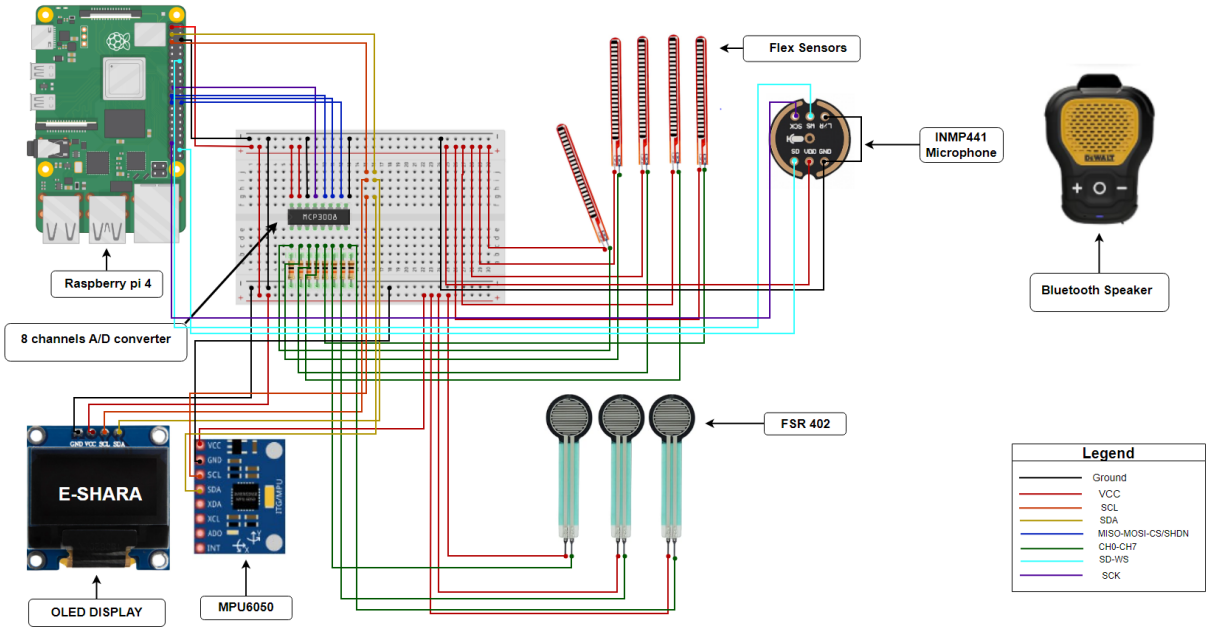


Figure 22: Circuit Design for First Glove

Each of the FSR and Flex sensors is connected to a channel on the 8 channel ADC MCP3008. The connections between the output of the ADC and the Raspberry Pi are shown in the figure below. The ADC uses the SPI0 pins of the Raspberry Pi.

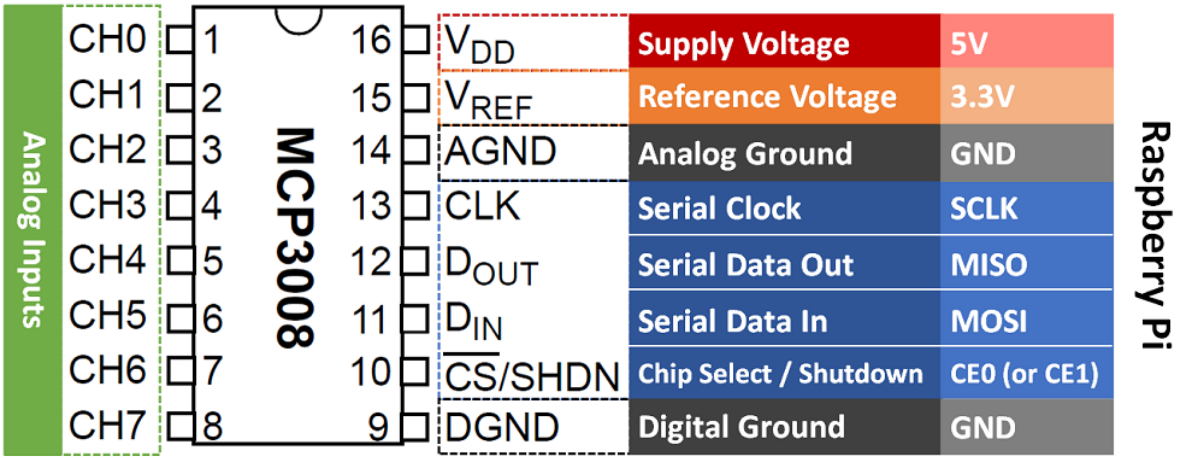


Figure 23: MCP3008 to Raspberry Pi Connections

The Raspberry Pi communicates with the MCP3008 via Serial Peripheral Interface (SPI) protocol, where the Raspberry Pi is the master and the MCP3008 is the slave. The microphone is connected to the SPI1 pins of the Raspberry Pi to also communicate via SPI where the microphone is the slave. The MPU6050 and the OLED Display are connected to the I2C1 module of the Raspberry Pi to establish the Inter-Integrated Circuit protocol (I2C) where the MPU6050 and the OLED Display are the slaves and the Raspberry Pi is the master. The speaker is connected to the Raspberry Pi via Bluetooth.

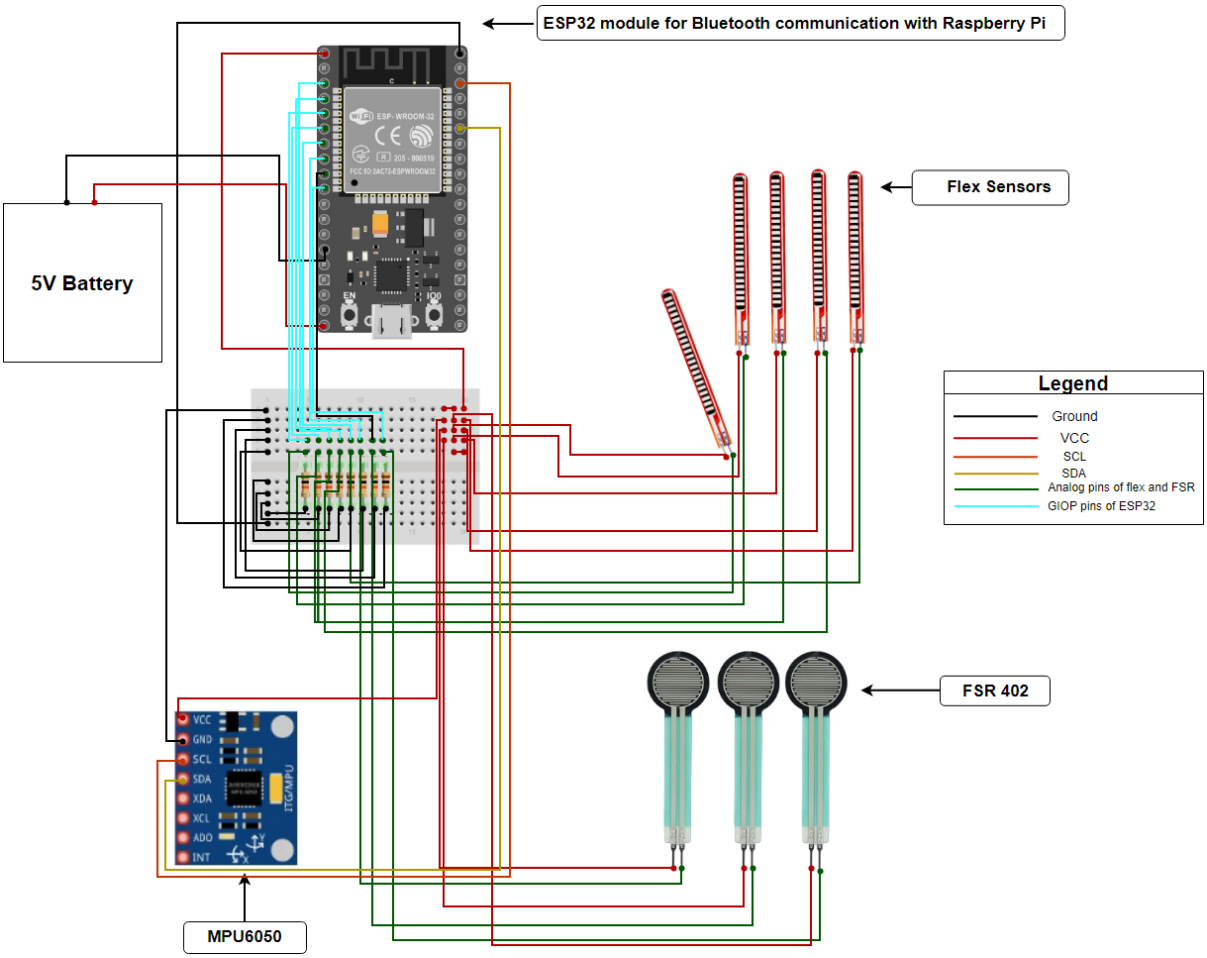


Figure 24: Circuit Design for Second Glove

For the second hand, the 5 flex sensors and 3 FSR 402 sensors were connected to the ESP32's analog GPIO pins mapping them to GPIO 32-39, which support analog inputs. The sensors were connected in a voltage divider with resistors to enable accurate analog voltage readings. The MPU-6050 was connected to the ESP32 using the I2C protocol, with its SDA and SCL pins wired to GPIO 21 and GPIO 22, respectively. The power for the sensors is provided by the 3.3V pin of the ESP32.The ESP32 itself was powered through the 5V Vin pin connected to an external battery.

6.4 Glove Design

The placement of components in the glove design is carefully chosen to ensure both functionality and user comfort.
Flex sensors are strategically positioned along the length of each finger on the front of the glove to accurately measure bending and flexing, enabling precise detection of hand gestures.
Force-sensitive resistors (FSRs) are located on the back of the glove near the fingertips to capture pressure feedback, enhancing the recognition of gestures involving finger pressing or gripping.
The Inertial Measurement Unit (IMU), which integrates an accelerometer and gyroscope, is centrally mounted on the front of the glove on the back of the hand to reliably track hand orientation and movement during dynamic gestures. An omni-directional micro-

phone is positioned below the wrist on the front of the glove to capture sound-based cues, while an OLED display is placed nearby to provide real-time visual feedback, ensuring ease of use without hand mobility.
The Raspberry Pi 4 and breadboard are located on the back of the glove near the wrist, serving as the processing and wiring organizer, with the MCP3008 analog-to-digital converter placed adjacent to the Raspberry Pi for efficient communication. This will be covered by a 3D-printed case, ensuring a neat and ergonomic design.

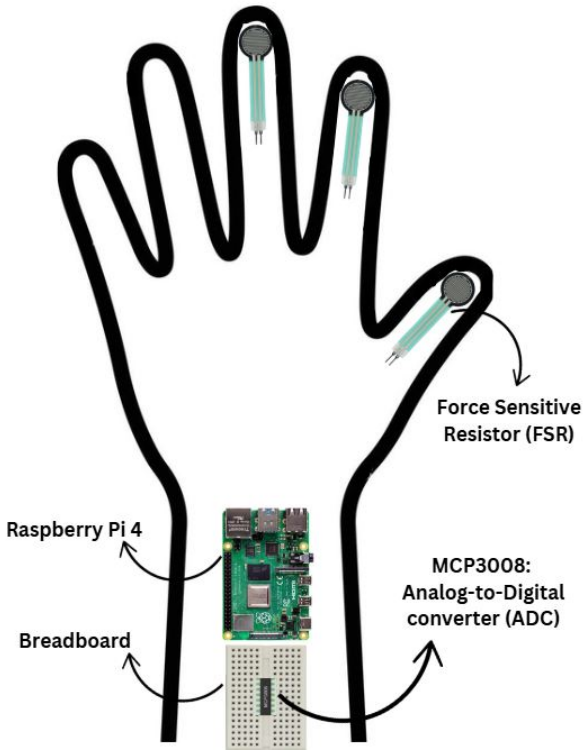


Figure 25: Back of the Glove Design

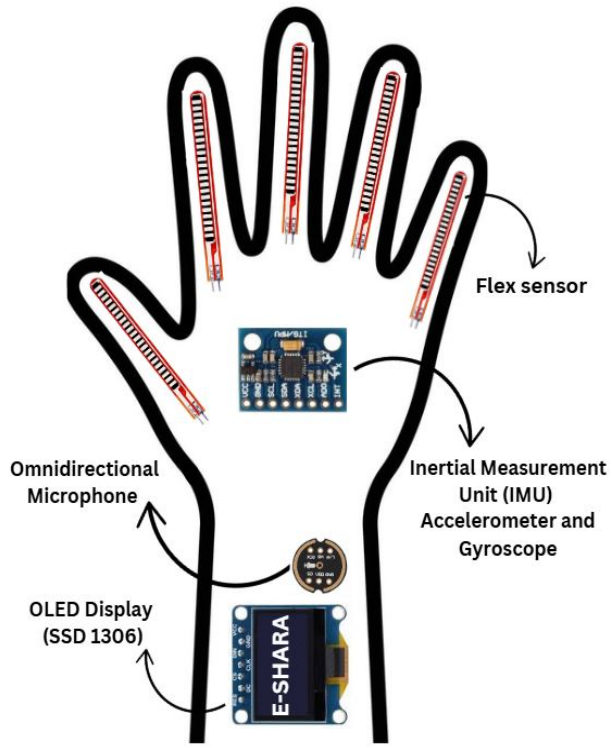


Figure 26: Front of the Glove Design

This combination of carefully chosen sensors and processing power enables the glove to interpret complex gestures in real-time, making it an effective and user-friendly tool for communication.

7 Signal Processing

In this section, we will describe the tools and methods chosen for processing signals and generating outputs for the microphone, speaker, and sensors. We will highlight the key criteria that influenced our decisions and provide an explanation for selecting these specific technologies.

7.1 Signal Processing of the Microphone

7.1.1 Post-Microphone Noise Cancellation

The use of MEMS resonant microphone arrays (RMAs) combined with active noise cancellation (ANC) and automatic speech recognition (ASR), can be effectively adapted for our project where spoken words are captured by a microphone and displayed on an

OLED. MEMS RMAs offer high sensitivity and natural noise filtering, which ensures accurate speech capture even in noisy environments. The integration of ANC techniques, such as digital adaptive filters or phase compensators, can further enhance audio quality by suppressing background noise, making it easier for ASR systems to process and convert spoken words into text. This setup is particularly beneficial in scenarios requiring real-time speech-to-text conversion and display, as it provides robust performance in challenging acoustic environments. Although this approach adds complexity and cost compared to simpler microphone systems, its advantages in noise suppression and accuracy make it beneficial to our design.

7.1.2 Audio-to-Text Display on OLED via Microphone

The microphone captures the speech input as an analog audio signal and converts it into an electrical signal. This signal is passed through a denoising process, such as adaptive filters (as previously explained), where ambient noise is effectively removed to isolate the intended speech. Adaptive filters are particularly advantageous in this context, as they dynamically adjust their parameters to continuously suppress varying background noise in real time. This capability ensures that the system remains effective even in challenging acoustic environments, as discussed in the integration of ANC techniques with MEMS resonant microphone arrays.

Once the audio is filtered, it is sent to the Raspberry Pi via a USB connection. The Raspberry Pi, serving as the central processing unit, employs a speech recognition module, such as the Google Speech-to-Text API, to convert the denoised audio input into text. This combination of adaptive filtering and advanced speech recognition enables the system to deliver accurate, real-time transcription and display of spoken words, even in noisy surroundings.To achieve this, the audio signal is encoded in MP3 format and transmitted to the Google API server over the internet using a Remote Procedure Call (RPC). The API processes the audio, converts it into a string of text, and sends the text back to the Raspberry Pi. Once received, the Raspberry Pi displays the text on an interfaced OLED screen. Simultaneously, the text can be amplified and played back through a speaker. This process allows real-time conversion of spoken words into text, enabling enhanced accessibility and communication.

However, the ideal system functionality for our design is to perform audio-to-text conversions locally on the Raspberry Pi using software and libraries such as Mozilla DeepSpeech to remove the dependency on online services. Also, PyAnnote can be integrated to add advanced features like speaker diarization and voice activity detection, which are critical for distinguishing and segmenting individual speakers in multi-speaker audio files

7.2 Signal Processing of the Speaker

7.2.1 Pre-Speaker Noise Cancellation

Adaptive filtering techniques can be effectively implemented on a Raspberry Pi to filter noise from speech signals. The process begins with the input of a noisy speech signal, which contains both the desired speech and unwanted noise. An adaptive filter is set up, consisting of a set of coefficients that define its behavior, and it can be based on algorithms like Least Mean Square (LMS), Normalized Least Mean Square (NLMS), or Recursive Least Square (RLS).

The noisy input signal is then passed through the adaptive filter, producing an output that estimates the desired clean speech signal. This output is compared to a reference signal, ideally the clean speech, resulting in an error signal that represents the deviation from the desired output. Based on this error, the filter coefficients are updated; in the LMS and NLMS algorithms, the update is proportional to the error and the input signal, allowing the filter to learn and adapt to the noise characteristics. In contrast, the RLS algorithm employs a more complex calculation that considers past inputs and outputs for faster adaptation.

This process is repeated for each sample of the input signal, enabling the filter to improve its ability to distinguish between the desired speech and noise over time. Ultimately, the output of the adaptive filter is a cleaner version of the speech signal, with significantly reduced noise interference. Choosing the appropriate technique depends on which algorithm produces the cleanest output signal.

7.2.2 Text-to-Audio via Speaker Displayed on OLED

Once the audio input is captured and processed by the microphone, it is converted into text using the Google Speech-to-Text API. This text is then received by the Raspberry Pi and displayed on the OLED. Simultaneously, the text is fed into a speech synthesis module within the Raspberry Pi. The module converts the text back into audio form, generating a synthesized voice output. This audio signal is sent to the speaker, connected to the Raspberry Pi via its 3.5 mm audio jack or another output port. The speaker amplifies the audio, allowing the synthesized voice to be heard clearly by the user.

Ideally, we want the system to perform text-to-audio conversions locally and not depend on cloud services that require an online connection. This can be achieved by using text-to-audio software engines such as Piper.

7.3 Signal Processing of Sensors

7.3.1 Signal Processing of Flex sensors

To utilize flex sensor values as input for our machine learning model, we begin by establishing a hardware interface with a microcontroller such as the Raspberry Pi. Flex sensors operate as variable resistors, altering their resistance in response to bending angles. These sensors are integrated into a voltage divider circuit, where one terminal connects to a power source, and the other connects to an analog input pin on the microcontroller via a resistor. The Raspberry Pi captures the sensor's output as a voltage signal, which corresponds directly to the bending angle.

For data collection, the flex sensors are affixed to designated locations (e.g., fingers) to capture specific gestures or actions. Sensor readings are recorded for each gesture and labeled appropriately (e.g., "gesture 1"). The labeled data is then saved in a structured format, such as a CSV file. To prepare the data for machine learning, preprocessing steps are undertaken, including normalization to ensure consistent scaling across samples and noise reduction. Additionally, feature extraction is performed to enrich the dataset with characteristics such as raw sensor values, squared values, first-order differences (changes between consecutive readings), and second-order differences (differences across three consecutive readings). These features provide the model with a more nuanced understanding of the data patterns.

Subsequently, the dataset is divided into training and testing subsets. A machine learning mode is trained on the processed training data and its performance is evaluated using metrics like accuracy, precision, recall, and F1 score. After successful training, the model is deployed for real-time use by integrating it with live sensor data from the Raspberry Pi. Incoming data is preprocessed in real-time to extract the same features as in training, and these are input to the trained model for classification of gestures or actions. This end-to-end workflow ensures effective interpretation of sensor readings, enabling accurate predictions and facilitating practical applications such as gesture-controlled devices and assistive technologies.

7.3.2 Signal Processing of Force Resistive Sensors (FSR)

To effectively utilize force-sensing resistors (FSRs) for hand gesture recognition, sensors are chosen to accurately detect the pressure changes associated with hand gestures. These sensors capture the forces exerted by the fingers and palm during various gestures. FSRs must be strategically positioned on the hand to capture meaningful pressure changes. Typical placements include the fingertips to detect pinches or taps, the palm to measure grip strength or clenching, and the sides of the hand to monitor lateral movements. A combination of these placements can provide comprehensive gesture recognition.

During data acquisition, the FSRs are connected to a microcontroller, such as a Raspberry Pi, to read the analog voltage outputs. These outputs, which correspond to resistance changes in the sensors, are converted into pressure values. The system must sample data at a sufficient rate to capture dynamic hand movements effectively.

Signal processing is then applied to refine the raw data using techniques such as low-pass filters to reduce noise, normalization to account for variations in sensor sensitivity, and feature extraction to identify key characteristics like peak pressure values, pressure duration, and temporal patterns.

For gesture recognition, algorithms are developed to classify different hand gestures based on the processed data where machine learning approaches are employed and trained on labeled gesture datasets to identify complex patterns in the pressure data.

Once the system is developed, testing and calibration are carried out with a diverse group of users to ensure accurate recognition across different hand sizes and strengths. Adjustments are made as needed to enhance sensitivity and performance.

8 Dataset Customization and Machine Learning

In this section, we will outline the technologies selected for dataset customization, preprocessing, and cleaning, as well as the machine learning algorithms. We will detail the metrics that informed our decisions and explain the rationale behind our technology choices.

8.1 Dataset Customization

8.1.1 Dataset Collection

Given the constraints of manually collecting data and training machine learning algorithms within a single semester, we have chosen to focus on a subset of English and

Arabic words. After reviewing LeapScholar (2024) [21], the 21 words we chose to test in basic communication for ASL are the following:

Category	Words
Pronouns	I, You, We, They, Me
Courtesy Words	Please, Thank you
Descriptors	Good, Bad
Action Verbs	Eat, Drink, Go, Stop, Want, Help, See
Question Words	Where, When, What, How
Greetings	Hello

Figure 27: Words Chosen for ASL

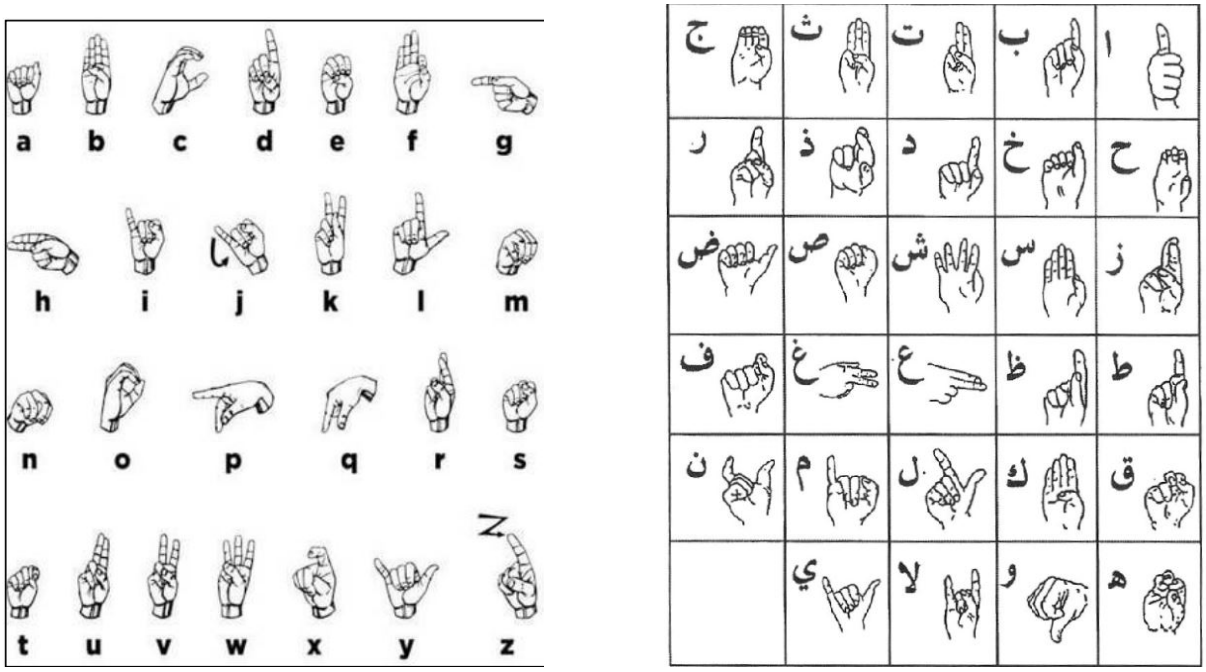
These words enable the formation of approximately 280 coherent sentences covering basic statements, questions, and requests.
As for ArSL, the 21 words we chose to test in basic Arabic communication are the following:

Category	Words
Pronouns	أنا، أنت، نحن، هم، لي
Courtesy Words	من فضلك، شكراً
Descriptors	جيد، سيء
Action Verbs	أكل، شرب، ذهب، توقف، أراد، ساعد، رأى
Question Words	أين، متى، ماذا، كيف
Greetings	السلام عليكم

Figure 28: Words Chosen for ArSL

These words enable the formation of 100 to 282 distinct combinations covering basic statements, questions, and requests in Arabic.

Our data collection protocol begins with testing individual letters, followed by words, and then sentences. We use a wearable sensor system comprising flex sensors, FSRs, and IMU sensors.
Participants are required to perform initial calibration steps to accommodate variations in hand anatomy. They are positioned facing north for 3 to 4 seconds to establish a baseline attitude. Each gesture is held for 4 seconds, followed by a rest period of 2 to 3 seconds between gestures. After completing each gesture, participants return their arms to a natural position. The dataset consists of 100 samples per gesture, adhering to Wu et al.'s methodology [38], with a split ratio of 60% for training, 20% for validation, and 20% for testing. Additionally, between consecutive sign language gestures, participants perform two calibration actions: full finger bending and full finger extension.



(a) American Sign Language (b) Arabic Sign Language

Figure 29: Comparison of ASL and ARSL

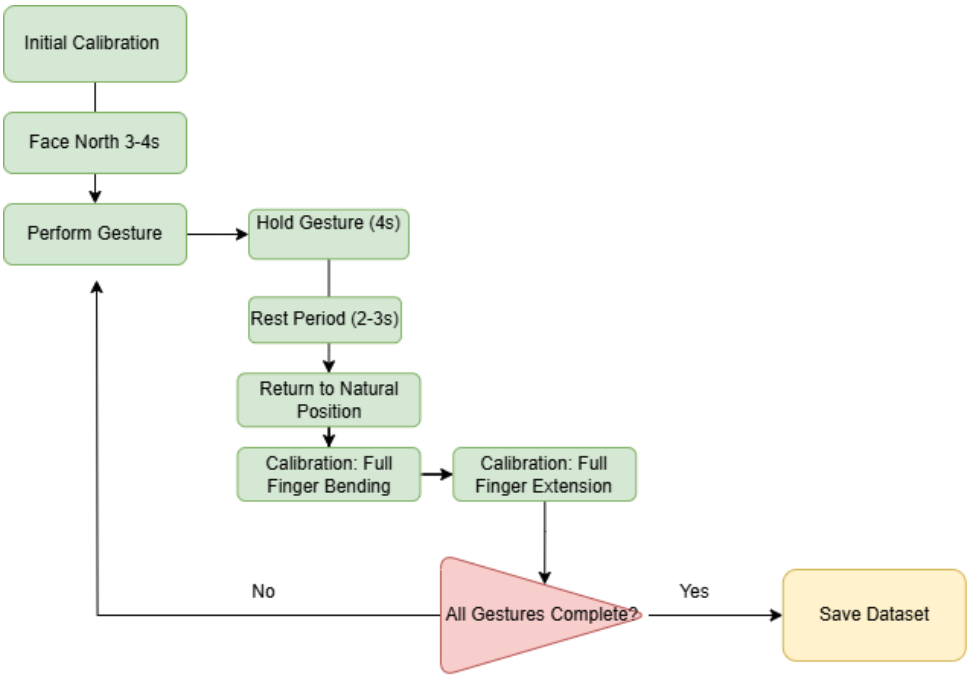


Figure 30: Dataset Collection Process

8.1.2 Dataset Preprocessing

After collecting the dataset, preprocessing becomes a critical step to prepare the data for analysis and model development. The first stage involves cleaning the data through signal processing techniques, with its details specified in Section 7. These steps are essential to enhance data quality and ensure reliable input for subsequent analysis.

Following data cleaning, feature extraction is performed to derive essential characteristics from the time and frequency domains. Based on our review of previous research, key features were selected, including mean, standard deviation, skewness, kurtosis, corrugation factor, quartiles, spectrum peak, and peak frequency. These features effectively capture the statistical, variability, and frequency properties of each gesture, making them critical for classification tasks. Additional features may also be extracted during the model training phase if needed to improve performance. In cases where dimensionality reduction is required, techniques such as PCA are employed to optimize computational efficiency while retaining critical information. This structured preprocessing pipeline ensures that the dataset is well-prepared for effective machine learning applications.

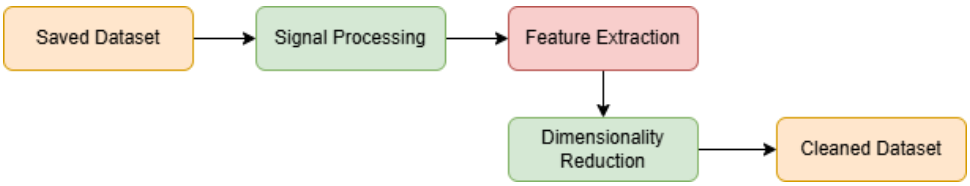


Figure 31: Dataset Preprocessing Process

8.2 Machine Learning

8.2.1 Hybrid Deep Learning Architecture for Sign Language Recognition

Based on a comprehensive analysis of the literature, a hybrid deep learning architecture was selected to optimize sign language recognition accuracy while maintaining computational efficiency. This decision was influenced by successful approaches documented in recent studies, particularly those by Ji et al. [17] and Jeon et al. [16], which demonstrated superior performance in sign language recognition tasks. While this architecture forms the foundation of our proposed system, several models will be tested on our dataset to ensure optimal performance, with this hybrid approach serving as the starting point for evaluation.

Figure 32: Static vs. Dynamic Gestures

The proposed system incorporates three primary components, each contributing to the overall recognition capability:

- **Pre-classification System**: Inspired by Jeon et al. [16], this system employs a three-layer Long Short-Term Memory (LSTM) network to classify input streams into three categories: non-gesture, static gesture, and dynamic gesture. The network includes dropout layers and fully connected layers with ReLU and softmax activations, enabling effective processing of continuous time-series data. As mentioned in the review, LSTM networks use gates to control the flow of information and how this allows the model to remember relevant patterns in time-series data. Hence, the LSTM network is particularly useful for handling sequential data, as it can maintain long-range dependencies, which is crucial when classifying gestures in sign language.

- **Static Gesture Recognition**: For recognizing static gestures, a three-layer Feed-forward Neural Network (FNN) is utilized, following the architecture proposed by Jeon et al. [16]. Static gestures are typically based on the shape or configuration of the hands, which do not involve temporal dependencies. An FNN is well-suited for this task as it processes each input independently. The network includes dropout layers for regularization and ReLU activations to introduce non-linearity, improving the model's ability to learn complex patterns.

- **Dynamic Gesture Recognition**: A Bidirectional LSTM (Bi-LSTM) network with attention mechanisms, as detailed by Ji et al. [17], is employed to recognize dynamic gestures. Dynamic gestures require understanding the temporal relationships between consecutive frames. The Bi-LSTM captures both forward and reverse sequence information, while the attention mechanism helps focus on relevant parts of the gesture sequence, addressing variable sequence lengths.

This hybrid architecture balances accuracy with computational efficiency by integrating these components, drawing from proven methods and innovations in the field of sign language recognition.

8.2.2 Model Implementation and Validation

The model's hyperparameters will be determined during the training phase based on the requirements of the dataset and the model's performance. Reference to Ji et al. [17] and other relevant literature will guide the selection of appropriate hyperparameters, such as learning rate, batch size, and dropout rate, to ensure stable convergence, efficient computation, and minimized overfitting. These parameters will be iteratively adjusted to optimize the model's performance, ensuring effective training within the available constraints.

The Bi-LSTM architecture stands as the optimal choice for this project, substantiated by its exceptional performance metrics and resource efficiency. Ji et al. [17] demonstrated Bi-LSTM's superiority through rigorous validation, achieving remarkable performance metrics: 98.19% accuracy, 98.39% precision, 98.03% recall, and a 98.15% F1-score. Jeon et al. [16] reported that their proposed hand gesture recognition system achieved an accuracy rate of 99.19% for static gestures. These balanced metrics underscore the model's robust generalization capabilities and reliable performance across diverse sign language patterns. While alternatives like Random Forest achieve marginally higher accuracy (99.7%), they lack the sophisticated temporal understanding crucial for sign language interpretation. The Bi-LSTM's unique ability to process bidirectional temporal dependencies, combined with its attention mechanism, enables nuanced gesture recognition while maintaining computational efficiency. This architecture proves particularly advantageous for real-time applications, offering superior sequence handling compared to CNNs (91.3%) and traditional LSTMs. Furthermore, its streamlined architecture requires fewer resources than complex systems such as Lee and Bae [23], making it more practical for deployment while maintaining competitive accuracy. These characteristics establish the Bi-LSTM as the most balanced and practical solution for real-world sign language recognition applications.

8.2.3 System Architecture Diagrams

The figure below illustrates the Machine Learning System Architecture.

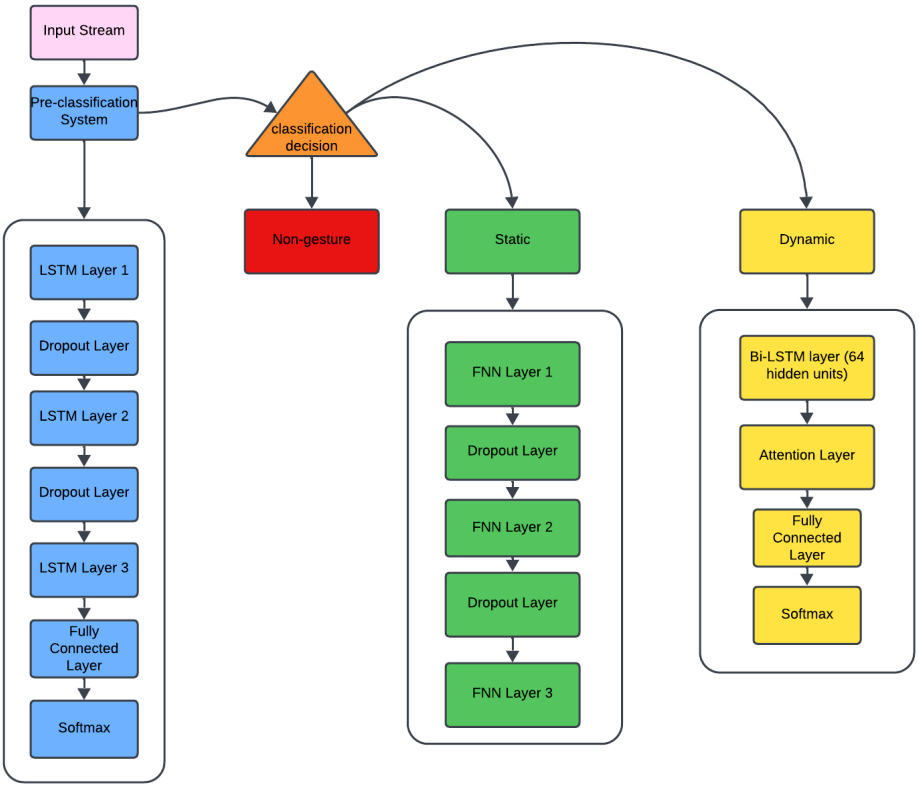


Figure 33: Machine Learning System Architecture Diagram

As proposed by Alosail et al. (2023) [2], the expected behavior of the system is illustrated in the diagram above.

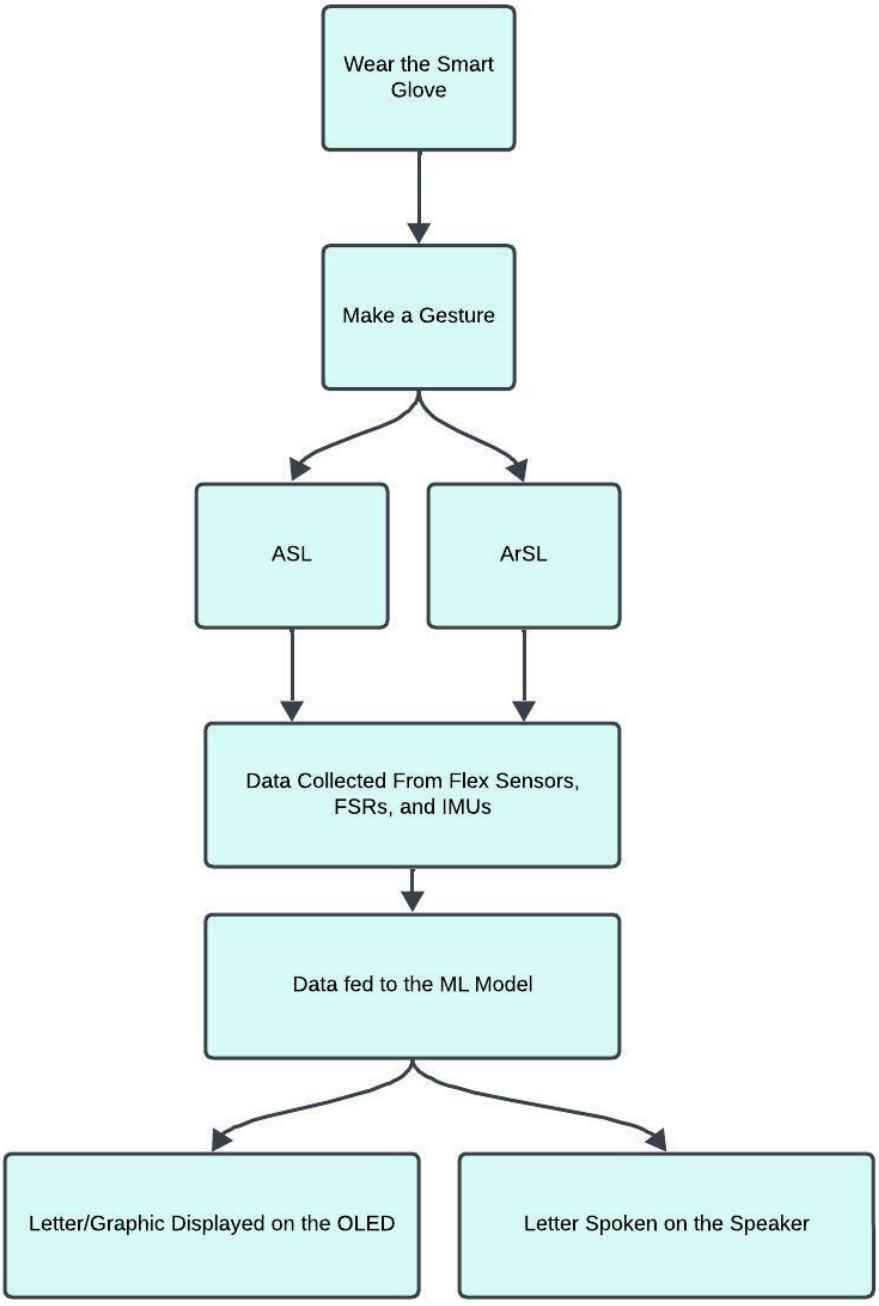


Figure 34: Bi-lingual System Diagram

9 Results

The main sensors of the design are preliminarily tested using an Arduino to understand the working mechanism of each.

Flex Sensor

The flex sensor was first tested in a voltage divider circuit using the following Arduino code:

```
#define sensorPin A0    // Flex Sensor is connected to this pin
float VCC = 5.0;
float R2 = 10000.0;     // 10K fixed resistor in the voltage divider

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

void loop() {
  int ADCRaw = analogRead(sensorPin); // Read the raw ADC value (0 to 1023)
  float ADCVoltage = (ADCRaw * VCC) / 1023.0; // Convert ADC value to voltage
  float Resistance = R2 * (VCC / ADCVoltage - 1); // Calculate resistance
  Serial.print("ADCVoltage: ");
  Serial.println(ADCVoltage);
  Serial.print("Resistance: ");
  Serial.println(Resistance);

  delay(500);
}
```

Figure 35: Flex sensor testing code

The results obtained from the serial monitor are as follows:

- When the flex sensor is straight, the approximate values are:
  - **ADC Voltage:** 0.34
  - **Resistance:** 138,260.87 Ω
- When the flex sensor is bent at 90 degrees, the approximate values are:
  - **ADC Voltage:** 0.04
  - **Resistance:** 1,268,750.00 Ω

The 5 flex sensors were tested multiple times to ensure the range of values fall within the values above ± 0.1 V for ADC voltage and ± 1000 Ω for Resistance.
We can conclude from these results that when the sensor is straight its resistance is relatively low and increases significantly when it is bent. This is caused by the deformation of the conductive material inside the flex sensor. In contrast, the voltage value across R2 is higher in the straight position, and it decreases significantly when the flex sensor is bent. The results obtained are consistent with the research done in previous sections.

Force Sensitive Resistor

Next, we tested the FSR402 in a voltage divider circuit using the following Arduino code:

```
#define FSR_PIN A0
int fsrVal = 0;
int fsrVoltage = 0;
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  fsrVal = analogRead(FSR_PIN);
  fsrVoltage = map(fsrVal, 0, 1023, 0, 5000);
   Serial.print("SSR: ");
  Serial.print(fsrVal);
  Serial.print(", ");
  Serial.print(fsrVoltage);
  Serial.print("mV ");
   if(fsrVal < 100)
    Serial.println("- No pressure");
  else if(fsrVal < 200)
    Serial.println("- Light touch");
  else if(fsrVal < 500)
    Serial.println("- Light squeeze");
  else if(fsrVal < 800)
    Serial.println("- Medium squeeze");
  else
    Serial.println("- Big squeeze");
  delay(50);
}
```

Figure 36: FSR Arduino Code

The code shown above classifies the pressure applied into five levels based on the raw analog readings from the FSR pin.
Different levels of pressure were applied to the sensor, and the results obtained from the serial monitor are as follows:

- **SSR: 0, 0 mV** - No pressure
- **SSR: 197, 962 mV** - Light touch
- **SSR: 324, 1583 mV** - Light squeeze
- **SSR: 647, 3162 mV** - Medium squeeze
- **SSR: 790, 3861 mV** - Medium squeeze
- **SSR: 931, 4550 mV** - Big squeeze
- **SSR: 976, 4770 mV** - Big squeeze

From the results obtained, we can conclude that the FSR is highly responsive to variations in pressure and can accurately differentiate between different levels of applied force.
The results can be explained by the working principle of the FSR:

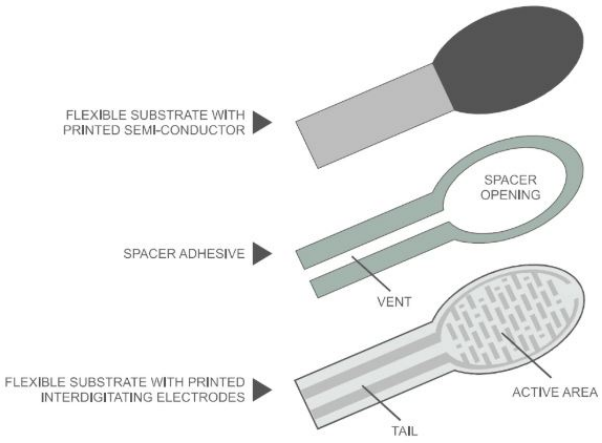


Figure 37: FSR Structure

As shown in the figure above, an air gap layer separates two membranes in a force-sensitive resistor. Each membrane has a conducting wire connected to it. One layer contains conductive ink, while the other is metallic. When no pressure is applied, both layers remain separated. When pressure is applied, the conductive path increases, causing the resistance to decrease. As pressure increases, the resistance decreases further, making the FSR output inversely proportional to the pressure coefficient [27].
When pressure or weight is applied to the FSR, the resistance decreases, which causes the voltage across the FSR to decrease while the voltage across the fixed resistor increases.

$$V_{\text{out}} = V_{\text{in}} \times \frac{R_{\text{res}}}{R_{\text{res}} + R_{\text{fsr}}}$$

The results obtained are consistent with the working principle of the FSR explained above.

Inertial Measurement Unit

Lastly, we tested the MPU-6050 using the following code:

```
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>
Adafruit_MPU6050 mpu;
void setup(void) {
  Serial.begin(115200);
  while (!Serial)
    delay(10);

  Serial.println("Adafruit MPU6050 test!");
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1) {
      delay(10);
    }
  }
  Serial.println("MPU6050 Found!");
```

Figure 38: MPU6050 Initialization

```
mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
Serial.print("Accelerometer range set to: ");
switch (mpu.getAccelerometerRange()) {
case MPU6050_RANGE_2_G:
  Serial.println("+-2G");
  break;
case MPU6050_RANGE_4_G:
  Serial.println("+-4G");
  break;
case MPU6050_RANGE_8_G:
  Serial.println("+-8G");
  break;
case MPU6050_RANGE_16_G:
  Serial.println("+-16G");
  break;
}
mpu.setGyroRange(MPU6050_RANGE_500_DEG);
Serial.print("Gyro range set to: ");
switch (mpu.getGyroRange()) {
case MPU6050_RANGE_250_DEG:
  Serial.println("+- 250 deg/s");
  break;
case MPU6050_RANGE_500_DEG:
  Serial.println("+- 500 deg/s");
  break;
case MPU6050_RANGE_1000_DEG:
  Serial.println("+- 1000 deg/s");
  break;
case MPU6050_RANGE_2000_DEG:
  Serial.println("+- 2000 deg/s");
  break;}
```

(a) Configuring accelerometer and gyroscope settings

```
mpu.setFilterBandwidth(MPU6050_BAND_5_HZ);
Serial.print("Filter bandwidth set to: ");
switch (mpu.getFilterBandwidth()) {
case MPU6050_BAND_260_HZ:
  Serial.println("260 Hz");
  break;
case MPU6050_BAND_184_HZ:
  Serial.println("184 Hz");
  break;
case MPU6050_BAND_94_HZ:
  Serial.println("94 Hz");
  break;
case MPU6050_BAND_44_HZ:
  Serial.println("44 Hz");
  break;
case MPU6050_BAND_21_HZ:
  Serial.println("21 Hz");
  break;
case MPU6050_BAND_10_HZ:
  Serial.println("10 Hz");
  break;
case MPU6050_BAND_5_HZ:
  Serial.println("5 Hz");
  break;
}
Serial.println("");
delay(100);
}
```

(b) Configuration of filter settings

```
void loop() {
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);
  Serial.print("Acceleration X: ");
  Serial.print(a.acceleration.x);
  Serial.print(", Y: ");
  Serial.print(a.acceleration.y);
  Serial.print(", Z: ");
  Serial.print(a.acceleration.z);
  Serial.println(" m/s^2");

  Serial.print("Rotation X: ");
  Serial.print(g.gyro.x);
  Serial.print(", Y: ");
  Serial.print(g.gyro.y);
  Serial.print(", Z: ");
  Serial.print(g.gyro.z);
  Serial.println(" rad/s");

  Serial.print("Temperature: ");
  Serial.print(temp.temperature);
  Serial.println(" degC");

  Serial.println("");
  delay(500);
}
```

(c) Reading sensor events and printing to the serial monitor

Figure 39: Grouped figures showing the configuration and usage of the MPU-6050 sensor.

The results obtained are as follows:

Acceleration: X: -2.19, Y: 8.28, Z: -3.43 m/s^2
Rotation: X: 0.00, Y: -0.23, Z: -0.23 rad/s
Temperature: 20.58°C

Explanation:

1. Tilt in the reverse direction along the X-axis and Z-axis.
2. No significant rotation around any of the three axes.

Acceleration: X: -1.59, Y: 8.26, Z: -3.48 m/s^2
Rotation: X: -0.30, Y: 0.88, Z: 0.46 rad/s
Temperature: 20.58°C

Explanation:

1. Continued tilt in the negative X-axis and Z-axis.
2. Clockwise rotation around the Y-axis and Z-axis.

Acceleration: X: 4.41, Y: 8.29, Z: -2.56 m/s^2
Rotation: X: 0.34, Y: 1.54, Z: 0.68 rad/s
Temperature: 20.57°C

Explanation:

1. Significant forward motion or tilt along the positive X-axis.
2. Significant clockwise rotational movement around the Y-axis.

Acceleration: X: 5.87, Y: 6.88, Z: -4.86 m/s^2
Rotation: X: 0.63, Y: 0.42, Z: -0.14 rad/s
Temperature: 20.56°C

Explanation:

1. Strong tilt or movement along the positive X-axis.
2. Y-axis no longer closely aligned with gravity, indicating combined tilt.
3. Significant tilt or motion away from the neutral Z-axis position.

Acceleration: X: 4.65, Y: 1.26, Z: -7.39 m/s^2
Rotation: X: -0.13, Y: 1.42, Z: 0.73 rad/s
Temperature: 20.55°C

Explanation:

1. Positive acceleration indicates strong forward tilt along the X-axis.
2. High negative Z value suggests the sensor is heavily tilted downward along this axis.
3. Significant clockwise rotation around the Y-axis.

These results align with the expected behavior of the MPU-6050, as the sensor accurately reflects changes in acceleration, rotation, and temperature based on applied movements.

10 Conclusion and Plan of Work

The design of **E-S.H.A.R.A** was developed after a detailed review of various sign language interpreter gloves and the best available sensors and audio-visual interface components based on specific constraints. Our project lends itself to map to a number of SDGs, specifically SDG 3 (Good Health and Well-being), SDG 4 (Quality Education), SDG 10 (Reduced Inequality), and SDG 9 (Industry, Innovation, and Infrastructure), by fostering inclusivity, promoting equality, and encouraging innovation. The glove integrates flex sensors, force-sensitive resistors, and an inertial measurement unit to accurately capture hand gestures, thereby enhancing communication for the hearing impaired. Furthermore, the implementation of machine learning techniques, particularly the Bi-LSTM architecture, enables effective real-time gesture recognition, showcasing the potential of combining hardware and software solutions to improve human-computer interaction in sign language communication.
Key contributions of this project include the optimization of sensor configurations for improved gesture recognition accuracy, the integration of audio-visual feedback mechanisms to facilitate seamless communication, and the adaptation of the system to support both ASL and ArSL, thereby broadening its accessibility to diverse user groups.
A notable limitation of this study, however, is that the training was based on a vocabulary of only 21 words in both sign languages, which may limit the glove's ability to recognize a wider range of signs and gestures commonly used in daily communication. To improve the system's effectiveness, future research should focus on expanding the vocabulary and enhancing its adaptability to different sign languages and regional dialects. By addressing this constraint, **E-S.H.A.R.A** can increase its utility and inclusivity, helping to break down communication barriers.

10.1 Plan of Work

Based on the comprehensive literature review and analysis of existing sign language recognition systems, the following plan of work has been developed to ensure systematic implementation and validation of the proposed system.

10.1.1 Research and Preparation

The initial phase involves extensive research and preparation to establish a solid foundation for the project; hence, the following steps were compeleted:

1. **Component Analysis and Selection:** Following methodologies from successful implementations in literature, careful selection of hardware components including flex sensors, FSRs, IMU, and microphone was conducted to ensure optimal performance within project constraints.

2. **Hardware Architecture Design:** Development of an ergonomic glove design incorporating sensor placement strategies derived from previous research, with emphasis on accuracy and user comfort as demonstrated in Figures 34 and 35.

Upon integrating the hardware, and in the event that improved components become suddenly available, certain hardware components may be subject to change to ensure the delivery of a higher-performing project.

10.1.2 Hardware Implementation

Building upon established methodologies in the field, the hardware implementation will proceed as follows:

- **Sensor Integration:** Strategic positioning and installation of sensors based on optimal configurations, discussed in Section 6.1.2.
- **Data Acquisition System:** Implementation of a robust data acquisition system utilizing MCP3008 A/D converter, following proven architectures from similar studies.
- **Processing Unit Setup:** Configuration of Raspberry Pi with necessary libraries and interfaces.
- **Sensitivity analysis:** Assess sensitivity and precision of each sensor
- **System Optimization:** Development and design of a more efficient and user-friendly system, like by customizing a 3D printed case.

10.1.3 Dataset Development

Following best practices identified, as discussed in Section 8.1.1:

1. Development of data collection methodology incorporating proven timing parameters (4-second holds, 2–3 seconds rest).
2. Cleaning the dataset according to the signal processing pipeline
3. Extracting features
4. Application of established dataset splitting ratios for robust model training and validation.

10.1.4 Software Architecture Development

Building upon the proven architectures identified in the literature, particularly those by Ji et al. [17] and Jeon et al. [16]:

1. **Hybrid Architecture Implementation:**
   - Development of a three-layer LSTM network with stacked layers of LSTM units for pre-classification
   - Implementation of an FNN architecture for static gesture recognition
   - Integration of a Bi-LSTM network with attention mechanisms to improve dynamic gesture recognition by focusing on important parts of the sequence
2. **Model Training and Evaluation**
   - Search for the optimal hyperparameters using techniques such as grid search, random search, or Bayesian optimization
   - Apply regularization techniques such as dropout, L2 regularization, or early stopping to prevent overfitting during training

- Monitor training and validation performance using performance metrics (accuracy, precision, recall, F1-score)

3. **Web Platform Integration:**
   - Development of user interface based on established accessibility guidelines
   - Integration of comprehensive documentation and user guidance systems

10.1.5 System Validation and Verification

An additional step will focus on validating and verifying the overall system's performance:

- **Validation Tests:** Conduct end-to-end testing to confirm system functionality, ensuring alignment with project goals and specifications.
- **Performance Benchmarks:** Measure system accuracy, latency, and efficiency under various scenarios to identify potential improvements.

This systematic approach, grounded in successful methodologies from the literature review, aims to achieve the validated performance metrics while ensuring robust and reliable system implementation. The project commenced with hardware implementation and web platform development during the initial phase, laying the foundation for integration. Throughout the duration of Final Year Project II, the dataset will be created and preprocessed, with signal processing techniques, such as noise filtering, feature scaling, and normalization, are applied to enhance data quality and feature extraction. Machine learning models will be developed, trained, and integrated into the system. Comprehensive testing of the system will include cross-validation, performance evaluation on a separate test set, and iterative refinement to ensure functionality, accuracy, and usability.

References

[1] Mohamed Aktham Ahmed et al. "A review on systems-based sensory gloves for sign language recognition state of the art between 2007 and 2017". In: *Sensors* 18.7 (2018), p. 2208.

[2] Deemah Alosail et al. "Smart glove for bi-lingual sign language recognition using machine learning". In: *2023 International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*. IEEE. 2023, pp. 409–415.

[3] Neena Aloysius and M Geetha. "Understanding vision-based continuous sign language recognition". In: *Multimedia Tools and Applications* 79.31 (2020), pp. 22177–22209.

[4] Radzi Ambar et al. "Development of a Wearable Sensor Glove for Real-Time Sign Language Translation". In: *Annals of Emerging Technologies in Computing (AETiC)* 7.5 (2023), pp. 25–38.

[5] Muhammad Saad Amin et al. "Alphabetical gesture recognition of american sign language using e-voice smart glove". In: *2020 IEEE 23rd International Multitopic Conference (INMIC)*. IEEE. 2020, pp. 1–6.

[6] K Anetha and J Rejina Parvin. "Hand talk-a sign language recognition based on accelerometer and SEMG data". In: *International Journal of Innovative Research in Computer and Communication Engineering* 2.3 (2014), pp. 206–215.

[7] Aneeta S Antony et al. "Sign language recognition using sensor and vision based approach". In: *2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*. IEEE. 2022, pp. 1–8.

[8] Valerio Belcamino, Alessandro Carfı, and Fulvio Mastrogiovanni. "A Systematic Review on Custom Data Gloves". In: *arXiv preprint arXiv:2405.15417* (2024).

[9] Amina Ben Haj Amor, Oussama El Ghoul, and Mohamed Jemni. "Sign language recognition using the electromyographic signal: a systematic literature review". In: *Sensors* 23.19 (2023), p. 8343.

[10] J Bukhari et al. "American sign language translation through sensory glove; signspeak". In: *International Journal of u-and e-Service, Science and Technology* 8.1 (2015), pp. 131–142.

[11] L Chenghong, A Shingo, and J Lei. "Data glove with bending sensor and inertial sensor based on weighted DTW fusion for sign language recognition [J]". In: *Electronics* 12.3 (2023), pp. 613–613.

[12] S. P. Dawane and H. G. A. Sayyed. "Hand gesture recognition for deaf and dumb people using GSM module". In: *International Journal of Science Research* 6.5 (May 2017), pp. 2226–2230.

[13] Kristina Grifantini. *Open-source data glove helps translate sign language*. Accessed: 2024-12-28. June 2009. URL: https://www.technologyreview.com/2009/06/23/212253/open-source-data-glove-2/.

[14] Nelson Morgan Herve Bourlard. "Connectionist Speech Recognition: A Hybrid approach". In: (1994). URL: https://www.researchgate.net/publication/230875873_Connectionist_Speech_Recognition_A_Hybrid_Approach.

[15] Instructables. *Raspberry Pi Bluetooth Speaker*. Accessed: 2025-01-01. n.d. URL: https://www.instructables.com/Raspberry-Pi-Bluetooth-Speaker/.

[16] Sujin Jeon et al. "Applying multistep classification techniques with pre-classification to recognize static and dynamic hand gestures using a soft sensor-embedded glove". In: *IEEE Sensors Journal* (2024).

[17] Ang Ji et al. "Dataglove for Sign Language Recognition of People with Hearing and Speech Impairment via Wearable Inertial Sensors". In: *Sensors* 23.15 (2023), p. 6693.

[18] Hyeon-Jun Kim and Soo-Whang Baek. "Implementation of wearable glove for sign language expression based on deep learning". In: *Microsystem Technologies* 29.8 (2023), pp. 1147–1163. DOI: 10.1007/s00542-023-05454-5. URL: https://doi.org/10.1007/s00542-023-05454-5.

[19] Suin Kim et al. "Consistent and reproducible direct ink writing of eutectic gallium–indium for high-quality soft sensors". In: *Soft robotics* 5.5 (2018), pp. 601–612.

[20] Ewa Korzeniewska, Marta Kania, and Rafał Zawiślak. "Textronic glove translating polish sign language". In: *Sensors* 22.18 (2022), p. 6788.

[21] LeapScholar. *A Guide on 500 Most Common English Words Used in Daily Life*. Accessed: 2025-01-02. 2024. URL: https://leapscholar.com/blog/most-common-english-words-used-in-daily-life/.

[22] Boon Giin Lee and Su Min Lee. "Smart wearable hand device for sign language interpretation system with sensors fusion". In: *IEEE Sensors Journal* 18.3 (Dec. 2017), pp. 1224–1232. DOI: 10.1109/jsen.2017.2779466. URL: https://doi.org/10.1109/jsen.2017.2779466.

[23] Minhyuk Lee and Joonbum Bae. "Deep learning based real-time recognition of dynamic finger gestures using a data glove". In: *IEEE Access* 8 (2020), pp. 219923–219933.

[24] Z. Lu et al. "A hand gesture recognition framework and wearable gesture-based interaction prototype for mobile devices". In: *IEEE Trans. Human–Mach. Syst.* 44.2 (Apr. 2014), pp. 293–299.

[25] Nesreen Mohamed. "Arabic sign language and vital signs monitoring using smart gloves for the deaf". In: *Deleted Journal* 53.2 (Apr. 2024), pp. 185–191. DOI: 10.21608/erjsh.2024.242501.1231. URL: https://doi.org/10.21608/erjsh.2024.242501.1231.

[26] K. Patil, G. Pendharkar, and G. N. Gaikwad. "American sign language detection". In: *International Journal of Science Research Publications* 4.11 (Nov. 2014), pp. 1–6.

[27] Peppe8o. *FSR402: Force Sensitive Resistor with Arduino*. Accessed: 2025-01-01. 2021. URL: https://peppe8o.com/fsr402-force-sensitive-resistor-arduino/.

[28] Francesco Pezzuoli, Dario Corona, and Maria Letizia Corradini. "Recognition and classification of dynamic hand gestures by a wearable data-glove". In: *SN Computer Science* 2.1 (2021), p. 5.

[29] Pramod Kumar Pisharady and Martin Saerbeck. "Recent methods and databases in vision-based hand gesture recognition: A review". In: *Computer Vision and Image Understanding* 141 (2015), pp. 152–165.

[30] Nikhita Praveen, Naveen Karanth, and MS Megha. "Sign language interpreter using a smart glove". In: *2014 international conference on advances in electronics computers and communications*. IEEE. 2014, pp. 1–5.

[31] C. Preetham et al. "Hand talk-implementation of a gesture recognizing glove". In: *Proceedings of the Texas Instruments India Educator Conference*. Bengaluru, India, Apr. 2013.

[32] The Engineering Projects. *How to Connect Raspberry Pi 4 and ESP32 via Bluetooth*. Accessed: 2025-01-09. 2023. URL: `https://www.theengineeringprojects.com/2023/04/how-to-connect-pi-4-and-esp32-via-bluetooth.html`.

[33] Roslyn Rosen. "The World Federation of the Deaf". In: *Deaf people around the world: Educational and social perspectives* (2009), pp. 374–391.

[34] Paul D Rosero-Montalvo et al. "Sign language recognition based on intelligent glove using machine learning techniques". In: *2018 IEEE Third Ecuador Technical Chapters Meeting (ETCM)*. IEEE. 2018, pp. 1–5.

[35] Mina I Sadek, Michael N Mikhael, and Hala A Mansour. "A new approach for designing a smart glove for Arabic Sign Language Recognition system based on the statistical analysis of the Sign Language". In: *2017 34th National Radio Science Conference (NRSC)*. IEEE. 2017, pp. 380–388.

[36] Netchanok Tanyawiwat and Surapa Thiemjarus. "Design of an assistive communication glove using combined sensory channels". In: *2012 Ninth International Conference on Wearable and Implantable Body Sensor Networks*. IEEE. 2012, pp. 34–39.

[37] Feng Wen et al. "AI enabled sign language recognition and VR space bidirectional communication using triboelectric smart glove". In: *Nature communications* 12.1 (2021), p. 5378.

[38] J. Wu, L. Sun, and R. Jafari. "A wearable system for recognizing American sign language in real-time using IMU and surface EMG sensors". In: *IEEE J. Biomed. Health Inform.* 20.5 (Sept. 2016), pp. 1281–1290.

[39] J. Wu et al. "Real-time American sign language recognition using wrist-worn motion and surface EMG sensors". In: *IEEE 12th Int. Conf. Wearable Implant. Body Sensor Netw.* Cambridge, MA, USA, June 2015.

[40] R Wu et al. *Full-fiber auxeticinterlaced yarn sensor for sign-language translation glove assisted by artificial neural network. Nano-Micro Lett. 14, 139 (2022).*

[41] Anton Yudhana, Jihad Rahmawan, and Cahya Utama Purwa Negara. "Flex sensors and MPU6050 sensors responses on smart glove for sign language translation". In: *IOP conference series: materials science and engineering*. Vol. 403. 1. IOP Publishing. 2018, p. 012032.

[42] Renjie Zhu et al. "Machine-learning-assisted soft fiber optic glove system for sign language recognition". In: *IEEE Robotics and Automation Letters* (2023).