Smart Glove for Sign Language Translation

1. Sazzad Hossen
(ID- 021 221 026, United International University)

2. Suvom Karmakar
(ID- 021 221 027, United International University)

3. Azad Hossain Saykat
(ID- 021 221 017, United International University)

4. Md. Tahman Hossain
(ID- 021 221 099, United International University)

5. Sabiha Hossain Mim
(ID- 021 221 007, United International University)

***Abstract*—Many people with hearing or speech impairments use sign language to communicate, but most of the general population does not understand it. This creates a communication barrier in daily life.**

Our project introduces a smart glove that helps bridge this gap by converting sign language gestures into text and speech. The glove uses special sensors to detect finger movements, hand orientation, and touch patterns. These signals are processed by an STM32 microcontroller, which then translates the gestures into words using a trained machine learning model. This system ensures accurate and fast recognition, making it a useful tool for people who rely on sign language to communicate with others.

INTRODUCTION: Sign language is a vital means of communication for individuals with hearing and speech impairments, yet most people are not proficient in it, leading to communication difficulties. To address this issue, various technological solutions have been developed, with smart gloves emerging as a promising approach. Our project focuses on designing a smart glove that translates sign language gestures into text and speech output, facilitating smoother interaction between sign language users.

Unlike traditional designs that rely on low-power microcontrollers with limited processing capabilities, our implementation uses an STM32 microcontroller, which offers better real-time data processing and sensor integration. The glove incorporates flex sensors to detect finger bending, an MPU6050 accelerometer-gyroscope to capture hand orientation and motion, and touch sensors to differentiate similar gestures. Machine learning techniques are employed to classify gestures with high accuracy, ensuring an efficient system.

High Level Design

Rationale

Our aim was to develop a solution that enables individuals with speech impairments to communicate more easily with the general public. By utilizing various sensors, we can convert the position and movement of the right hand into numerical data. Collecting a sufficient amount of this data for each letter or word allows us to train a machine learning model to recognize and associate specific hand gestures with their corresponding signs.

To achieve this, we use flex sensors as variable resistors to measure the degree of finger bending, while the MPU6050 accelerometer and gyroscope capture hand orientation and movements that flex sensors alone cannot detect. Additionally, to differentiate between highly similar signs, contact sensors are incorporated into the glove, providing extra data to enhance recognition accuracy.

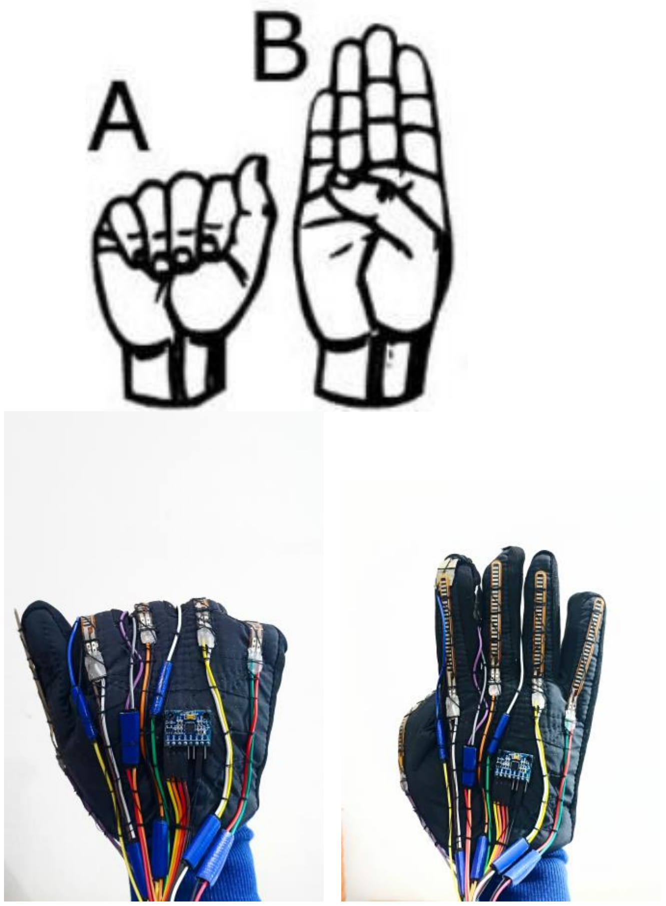


FIGURE 1. ASL (American Sign Language) signs for ”A”, and ”B”

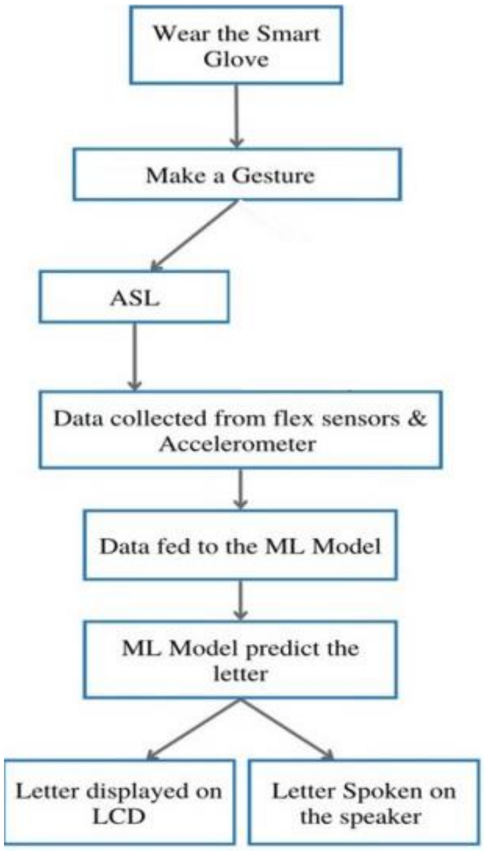

Wear the Smart Glove
Make a Gesture
ASL
Data collected from flex sensors & Accelerometer
Data fed to the ML Model
ML Model predict the letter
Letter displayed on LCD
Letter Spoken on the speaker

FIGURE 2. Proposed smart glove approach for ASL using ML

Logical Structure

- **Flex Sensor:** Acts as a variable resistor, changing resistance when bent. Integrated into a voltage divider circuit, its output is processed through the STM32’s ADC to determine the degree of finger bending.
- **MPU-6050:** Uses the I2C interface to send accelerometer and gyroscope data to the STM32 via the SDA bus, providing motion tracking.
- **Contact Sensors:** Made of copper tape, these sensors generate binary input to indicate finger contact or separation.
- **STM32 Processing:** Collects data from flex sensors, MPU-6050, and contact sensors, converts it into a structured digital format, and packages it into packets for transmission.
- **Data Transmission:** The STM32 sends data to a PC via a serial bus at regular intervals for real-time processing.

Python Algorithm (ML):

- **Serial Wrapper:** Manages STM32 communication.
- **Data Processing Script:** Stores and structures sensor data.
- **Machine Learning Model:** Learns and classifies gestures.
- **Visualization & Debugging Scripts:** Provide real-time data monitoring and troubleshooting.

Hardware/Software Tradeoffs

MCU vs CPU

Machine Learning (ML) can be computationally intensive depending on the dataset, learning function, and data structure. For our project, we opted to offload most of the computation to a PC rather than performing it on the STM32 microcontroller. This decision was based on several factors:

- **Training Efficiency:** Training ML models is resource-intensive, and performing this process on a high-performance PC allows for faster computation and access to a variety of well-established ML libraries.

- **Data Visualization & Debugging:** Handling the data on a PC provides better visualization tools, making it easier to analyze and manipulate data.
- **Classification Performance:** While inference can sometimes be efficient, we chose to perform classification on the PC to leverage its greater processing power.
- **Scalability:** Future iterations of the system may integrate ML inference directly on the STM32 or a more advanced microcontroller.

Prediction Without a Button Push

A key design decision was determining how the system would recognize when a gesture was being made. Instead of using a push button, we introduced ”nothing” and ”relaxed” states in the classification model, allowing for seamless detection of user gestures.

Designation of Sensor Pins

One advantage of ML-based gesture recognition is that exact sensor-to-pin mapping is not critical. Instead, the ML model learns patterns from data regardless of the specific sensor assignment, increasing flexibility and robustness.

Changes from Initial Prototyping

Given time constraints, we prioritized accuracy over portability in our initial prototype. Future improvements will focus on integrating wireless communication to enhance the system’s usability.

Hardware Design

Glove Construction

The smart glove is embedded with sensors and necessary wiring. Each finger has a flex sensor stitched on the back, and an MPU-6050 is attached to the center of the back of the glove. Additionally, copper tape was affixed at four key locations for contact sensors.

Voltage Divider Circuits

Each flex sensor is a variable resistor. To convert its resistance into a measurable voltage, we use a voltage divider circuit:

$$V_{out} = V_{in} \times \frac{R_1}{R_1 + R_2} \qquad (1)$$

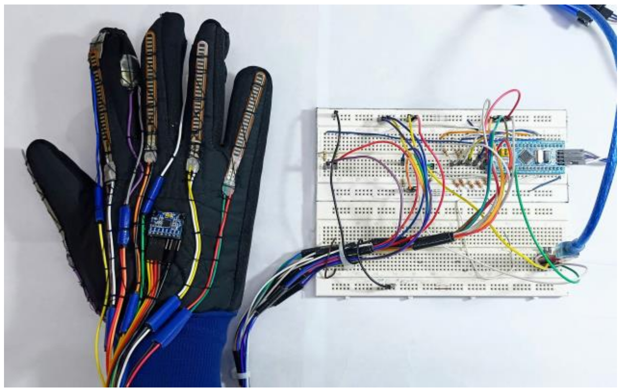


FIGURE 3. Hardware Implementation

where R_1 is a fixed resistor and R_2 is the flex sensor. We selected $R_1 = 10k\Omega$ based on experimen- tal analysis. Each flex sensor has a baseline resistance of $10k\Omega$ when straight and increases to approximately $27k\Omega$ when fully bent.

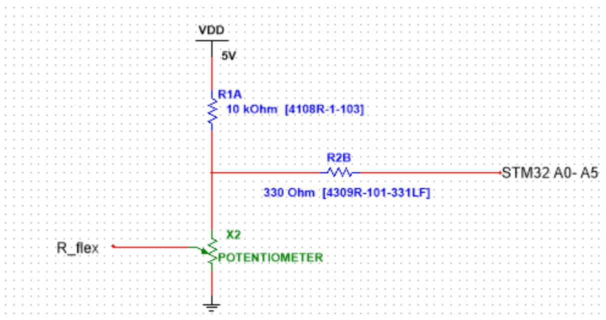


FIGURE 4. Voltage Divider Circuit

To determine the optimal R_1 value, we tested resistor values between $10k\Omega$ and $22k\Omega$ and found that the voltage range difference was negligible. Using $R_1 = 10k\Omega$ provided a sufficient range for the STM32’s ADC input, allowing the system to differ- entiate finger positions accurately.

The resulting voltage divider outputs a voltage range of approximately 1V, which was tested for sufficient variation in ADC readings. While we initially considered increasing the input voltage to extend the range, our tests showed that the existing range was effective for accurate machine learning classification.

Voltage Regulator

The MPU-6050 sensor operates at 3.3V, which is already provided by the STM32 microcontroller. To ensure stable communication over the I2C bus (SDA and SCL), we added pull-up resistors to the circuit.

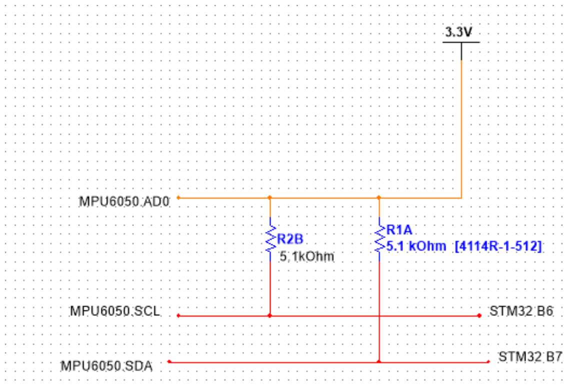


FIGURE 5. Pull-up Circuit

- Two 4.7kΩ pull-up resistors were connected to the SDA and SCL lines.
- These resistors ensure proper signal integrity for I2C communication.

Since the STM32 already supplies **3.3V**, no additional voltage regulation circuit was required.

Contact Sensors: Contact sensors are added at key spots on the glove, acting as switches that pull MCU input pins to ground when activated. This provides additional accuracy in differentiating similar gestures.

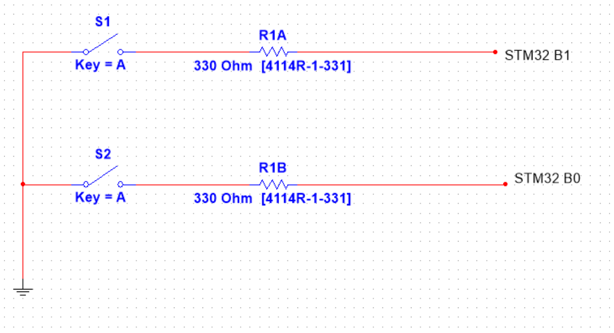


FIGURE 6. Contact Sensor Circuit

Sensor Mounting

To ensure comfort and consistency, flex sensors are stitched in a way that allows natural sliding with- out shifting, which prevents data inconsistencies. The MPU-6050 is placed to avoid fabric bunching, ensuring accurate motion tracking. Wires are securely stitched to prevent obstruction during signing.

Software

I2C Communication

Interfacing with the MPU-6050 sensor required I2C communication. Although the STM32 microcontroller doesn't have dedicated I2C libraries for this application, we used a modified version of Peter Fleury's public I2C library. I2C allows multiple devices to share a single data (SDA) bus and a clock (SCL) bus, which facilitates communication with the MPU-6050 sensor.
Macros for issuing start and stop conditions were used to manage the bus and initialize the interface, set up the MPU-6050, and request sensor data. This process ensured the efficient polling of the accelerometer and gyroscope data.

Watchdog Timer

To prevent the system from hanging during I2C communication, we enabled a watchdog timer on the STM32. The watchdog timer resets the system every 0.5 seconds unless the program progresses to a checkpoint where it resets the timer. This mechanism ensured that any stalling during sensor data retrieval would not affect the system's operation.

TinyRealTime Kernel

For real-time operations, we utilized the TinyRealTime kernel, developed by Dan Henriksson and Anton Cervin. This kernel was particularly useful due to its built-in UART library, enabling communication with a PC. The program initializes the IMU, I2C, and ADC, and continuously collects 16 sensor readings from accelerometers, gyroscopes, and other sensors. The data is then processed and sent in packets to a PC via a USB connection.

Machine Learning Algorithm

The system operates in two modes: data gathering and continuous prediction. In the data gathering mode, the user provides labels (e.g., 'a', 'b'), and the system records sensor data at the highest possible rate. In the continuous prediction mode, the system continuously polls the microcontroller for data, processes it, and makes real-time predictions.

Support Vector Machines (SVM) are used for sign classification. Initially, raw sensor data from various sensors with different ranges posed challenges, but normalization (zero mean and unit variance) resolved these issues. The classifier achieved 98% accuracy after applying this transformation.

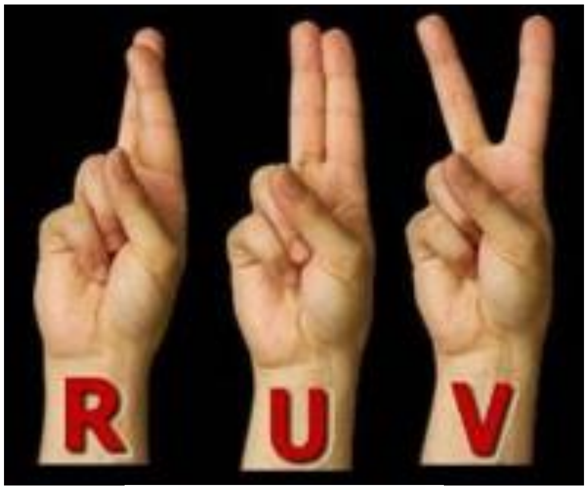
Data Visualization

To evaluate the performance of the classification algorithm, sensor data was plotted as images. By concatenating multiple sensor readings, we created a matrix to visualize the differences between sensor readings corresponding to different signs. Clear divisions in the data allowed us to verify that the classifier could distinguish between signs effectively.

Modifications from Initial Prototype

Our original glove did not have contact sensors on the index and middle fingers. As a result, it had difficulty predicting the ASL signs for ”R,” ”U,” and ”V” accurately. These signs are quite similar in terms of hand orientation and finger flex, making them hard to differentiate without proper sensor placement. To mitigate this issue, we added two contact sensors: one set on the tips of the index and middle fingers to detect the ”R” sign, and another pair placed between the index and middle fingers to distinguish between the ”U” and ”V” signs.

We also considered implementing a wireless communication feature for the glove using Bluetooth. However, due to time constraints, we were unable to integrate this feature into the prototype at this stage. The Bluetooth integration would have allowed for wire-free communication with the PC, but we decided to prioritize other aspects of the glove's functionality first.



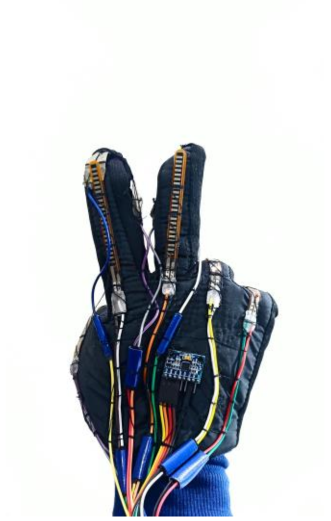


FIGURE 7. ASL signs for ”R”, ”U”, and ”V”, captured by contact sensors

Results

Speed of Execution

We determined the speed of data transmission to the PC by serially sending data and gradually increasing the send rate until we observed a discrepancy between the sending and receiving rates. We then reduced the send frequency to a reasonable value, which translated into a loop interval of 3 milliseconds.

To maximize data collection, the MCU gathered as much sensor data as possible between packet transmissions. Additionally, the MCU sent the number of readings it had performed with each packet. This allowed us to determine an optimal polling rate before aggregating the data and sending it to the PC. We concluded that the MCU was capable of reading and averaging each sensor 16 times per cycle.

Currently, the Python algorithm is limited by the MCU's data transfer rate and the time required for the speech engine to output the spoken word or letter. The transfer rate is about 30 Hz, and we wait to fill a buffer with about 10 unanimous predictions before outputting. This means the fastest prediction output is approximately 3 times per second, which is suitable for our needs.

Accuracy

The averaging performed at each interval helps mitigate noise and glitches from the flex sensors, enhancing the overall accuracy of the glove.

However, accuracy is somewhat limited by the size of the user's hands. The flex sensors' accuracy decreases beyond a certain point, as smaller hands result in a larger degree of flex, making it difficult to distinguish between signs with minimal differences. For instance, signs like “M” and “S” only differ in the amount of flex, but this difference might be too small for users with small hands to be detected effectively.

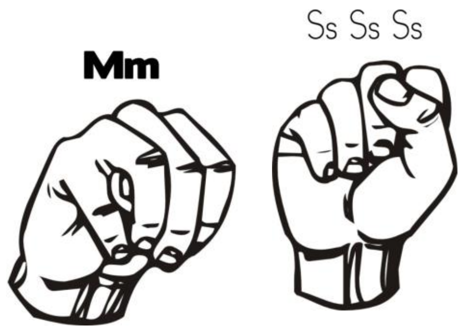


FIGURE 8. ASL signs for ”M” and ”S”

To mitigate potential data corruption in serial communication, we implemented a checksum for the packets sent. This added layer of validation ensured that the data received was accurate and intact.

Confusion Matrix for All Labels

The classifier currently achieves 98% accuracy on a task involving 28 signs, which includes the full ASL alphabet along with the labels “relaxed” and “nothing.” In comparison, a random classifier would achieve a 4% accuracy, clearly indicating that the device is highly accurate. However, the algorithm would benefit from improved touch sensors (as the most common error involves confusing ”U” and ”V”), as well as being trained on a larger dataset with a broader user population. With a sufficiently large dataset, we could provide new users with a script covering only difficult-to-predict signs and rely on the rest of the learned data for the remaining signs.

The software has been trained on data from two team members and tested on volunteers outside the team. The results were excellent for the team members who trained the glove and mostly satisfactory for other users. Since these volunteers had no prior experience with ASL and did not train the glove, it is difficult to determine if their lower accuracy was due to overfitting, unique signing styles, or unfamiliarity with ASL. Nevertheless, the software demonstrated near-perfect accuracy for users who trained the glove, and was mostly accurate for non-expert users.

Accuracy vs. Increasing Samples

As more samples were collected, the accuracy of the system improved significantly. This supports the importance of training the system with a broad and diverse dataset to improve prediction accuracy.

Safety and Interference with Other Devices

The device operates in a low-voltage environment with extremely low-frequency communication. The sensors are securely attached, and there are no sharp edges, so there are no major safety concerns associated with the glove.

Furthermore, since all communication is conducted via cables, the device does not interfere with other electronic designs or systems.

Usability

The glove is designed to be used by anyone who can fit into it, with the caveat that they must train the system and generate new datasets for improved prediction accuracy or to incorporate new signs. While the glove is primarily intended for people with special needs who cannot speak, it is also suitable for gesture recognition and can be applied to a variety of interactive computing spaces or gaming interfaces. The labels for each dataset do not necessarily have to be converted to speech; they can instead function as commands for controlling other systems or devices.

Conclusion

Our Expectations

The project met our expectations quite well. Our initial goal was to create a system capable of recognizing and classifying gestures, and we achieved this with more than 98% average accuracy across all 28 classes. Although we did not have a strict time requirement for the rate of prediction, the resulting speed made using the glove comfortable, and it did not feel sluggish. With more time or if we were to continue the project, we would focus on improving the touch sensors, as the primary ambiguity in signs stems from the difference between ’u’ and ’v’. We would also aim to use materials more suited for clothing to provide a more reliable connection. Additionally, we hope to make the glove wireless, which would allow it to easily communicate with phones and other devices, making the system truly portable.

Applicable Standards

American Sign Language standards: For gesture recognition, we adhered to the signs established by communities such as the American School for the Deaf and the National Association of the Deaf.

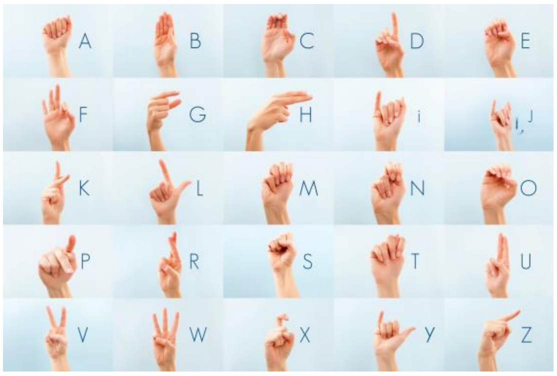


FIGURE 9. American Sign Language

I2C and USB standards: Although we did not need to worry about this ourselves, the libraries used for I2C and serial communication with the PC fol- lowed the proper standards as well.

Appendix

1. Source Code: The complete source code for the Smart Glove for Sign Language Translation project is available at the following link:

- https://drive.google.com/drive/folders/1lONdIx9QjWWD4NpTNTNKHwWhy45e0KGr?usp=drive_link

2. Video Presentation: A recorded video presentation demonstrating the project can be accessed at the following link:

- https://drive.google.com/drive/folders/1BEWGkBB1KvrAEYAYSvT5KqSy9BIay4L-?usp=drive_link

3. Project Slides: The project presentation slides are available at the following link:

- https://docs.google.com/presentation/d/1AVuAjq58pj6_DiOZOZkPb2YQYgis8BWN/edit?usp=drive_link&ouid=116451193356552420035&rtpof=true&sd=true

4. Project Schematic: The schematic diagram of the Smart Glove for Sign Language Translation project is available at the following link:

- https://drive.google.com/drive/folders/1BG5o04PlT62d4MtsxdqbDqgPk2bWFt0P?usp=drive_link

REFERENCES

1. A. Abougarair and W. Arebi, ”Smart Glove for Sign Language Translation,” *International Journal of Robotics and Automation*, vol. 8, pp. 109-117, 2022. doi: 10.15406/iratj.2022.08.00253.

2. D. Alosail, H. Aldolah, L. Alabdulwahab, A. Bashar, and M. Khan, ”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)*, Bengaluru, India, 2023, pp. 409-415. doi: 10.1109/IDCIoT56793.2023.10053470.

3. P. Rosero, P. Godoy, E. Flores, J. Carrascal Garc´ıa, S. Otero-Potosi, H. Benitez-Pereira, and D. Peluffo-Ordo´n˜ez, ”Sign Language Recognition Based on Intelligent Glove Using Machine Learning Techniques,” in *2018 IEEE Ecuador Technical Chapters Meeting (ETCM)*, 2018, pp. 1-5. doi: 10.1109/ETCM.2018.8580268.

4. A. Filipowska, W. Filipowski, P. Raif, M. Pieniaz˙ek, J. Bodak, P. Ferst, K. Pilarski, S. Siecin´ski, R. J. Doniec, J. Mieszczanin, et al., ”Machine Learning-Based Gesture Recognition Glove: Design and Implementation,” *Sensors*, vol. 24, p. 6157, 2024. doi: https://doi.org/10.3390/s24186157.