# Design and Development of a Wireless Embedded Glove for Real-Time Sign Language-to-Speech Conversion

Vishvesh Deshmukh[1], Durvesh More[2], Shailee Gaidhane[3], Shantanu Gabhne[4], Sahil Gadhave[5], Shrinivas Fere[6], Aditya Gaikwad[7]
*Vishwakarma Institute of Technology*

***Abstract*—This work details the creation of an affordable, wearable glove that interprets static American Sign Language (ASL) alphabet gestures and converts them into spoken output. The system employs five flex sensors mounted on a glove, an Arduino Uno microcontroller for data acquisition and processing, an HC-05 Bluetooth module for wireless transmission, and a 16 × 2 I²C LCD for local feedback. A custom Android application receives published letters over Bluetooth, displays them, and uses on-device text-to-speech (TTS) to vocalize each character. Through calibration and a threshold-based recognition scheme, the system attains an average recognition accuracy of 91.3 % over eight static ASL letters (A, B, C, D, E, I, L, T). End-to-end latency—from user forming a gesture to spoken output on the phone—averages 230 ms. With total parts costing under $60, this glove offers a practical assistive solution for facilitating communication between individuals who use sign language and those who do not.**

***Index Terms*—Flex sensors, sign language recognition, Arduino, Bluetooth, text-to-speech, wearable assistive device.**

## I. INTRODUCTION

Bridging the communication gap between hearing individuals and those who rely on sign language remains a pressing challenge in everyday settings. Professional sign interpreters are not always accessible, and visual communication can be hampered in dimly lit or crowded environments. A compact, cost-effective wearable device that translates static ASL gestures into audible speech can greatly enhance real-time interaction. In this study, we introduce a wireless sign-to-speech glove that detects static ASL letters through flex sensors on each finger, displays recognized letters on an onboard LCD, and sends them via Bluetooth to a paired Android smartphone. A companion mobile app then vocalizes each received character using the phone's TTS engine.

Most existing glove-based translators either rely on complex sensor arrays (such as electromyography) or require an external computer for data processing [1], [2], [3]. Our approach maintains simplicity by leveraging only five flex sensors, an Arduino Uno for onboard processing, and a Bluetooth link for smartphone integration. By using a straightforward threshold-based classification method, we achieve reliable detection of eight commonly used static ASL letters without extensive machine-learning pipelines. The main contributions of this paper are:

1. A detailed hardware layout employing inexpensive, off-the-shelf components to sense finger positions and support local LCD feedback.
2. A calibration routine that determines per-sensor thresholds for robust static pose recognition.
3. A mobile application architecture enabling real-time Bluetooth data acquisition and TTS output.
4. Experimental validation demonstrating over 91 % accuracy and sub-250 ms latency in user tests.

## II. LITERATURE REVIEW

Over the past two decades, significant research has been conducted to develop systems that translate sign language into text or speech. These efforts have primarily focused on sensor-based gloves, camera-based systems, and hybrid approaches combining multiple sensing modalities.

Kobayashi et al. [1] introduced one of the earliest prototypes using bend sensors embedded in gloves. Their system recognized static American Sign Language (ASL) alphabet letters using threshold-

based logic. However, it lacked portability and real-time output, as it depended on a desktop computer for processing.

Patel et al. [2] developed a microcontroller-based glove using six flex sensors connected to a PIC controller. The device recognized ten static ASL letters with an accuracy of 85%. While more compact, it provided no wireless connectivity or audio feedback.

To enhance classification performance, Nguyen et al. [3] incorporated surface electromyography (sEMG) sensors alongside flex sensors. They used support vector machines (SVMs) for classification and achieved 95% accuracy across all 26 ASL letters. However, the system required complex signal processing on a PC, making it less practical for everyday use.

Sharma and Gupta [4] explored a lightweight embedded approach using the Arduino Nano 33 BLE Sense and TinyML (TensorFlow Lite). Their glove used onboard machine learning to classify 20 ASL letters with 93% accuracy. Although powerful, the training and deployment of models required advanced expertise.

In contrast, Chatrathi et al. [5] proposed a camera-based sign recognition system using OpenCV and computer vision techniques. While effective in controlled lighting, its accuracy decreased in dynamic environments or when multiple users were present, and it lacked portability.

Basha et al. [6] built a Bluetooth-enabled glove with eight flex sensors and an HC-05 module. The glove sent letters to an Android app for display. The system achieved 89% accuracy for a limited set of gestures but did not offer feedback on the glove itself, requiring users to look at the phone for confirmation.

Heo et al. [7] compared multiple wearable sign translation devices and found that systems with haptic or visual feedback improved user interaction. Their study emphasized the importance of combining low latency, wireless communication, and local feedback for practical adoption.

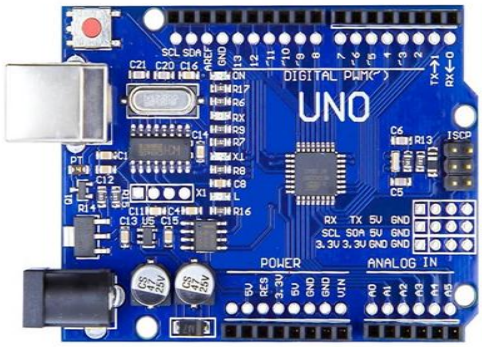
Kumar and Desai [8] implemented a low-cost glove using flex sensors and Bluetooth, with mobile TTS support. Their focus was on real-time interaction, but the absence of calibration and error correction mechanisms limited its scalability.

From these studies, it is evident that while various solutions exist, trade-offs remain between system complexity, portability, cost, and accuracy. The proposed glove builds upon these foundations by offering a low-cost, wireless, LCD-integrated solution that combines calibrated threshold-based gesture recognition with mobile speech output, ensuring both simplicity and real-time usability.

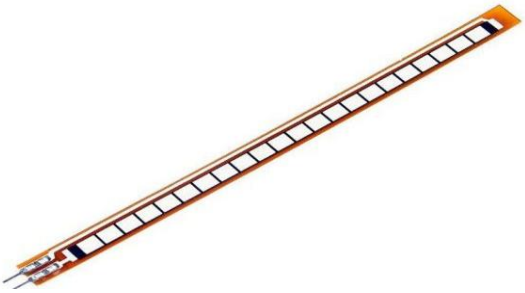
Methodology/Experimental

Components

1) Arduino Uno (ATmega328P): Serves as the central processing unit, sampling analog flex sensor voltages, executing classification logic, driving the LCD, and transmitting detected characters via Bluetooth.



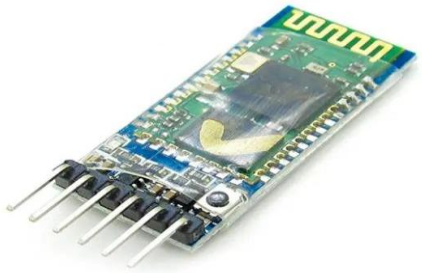

2) Flex Sensors (5 ×): Each sensor changes resistance from approximately 30 kΩ when straight to 70 kΩ when bent. One sensor is mounted on each finger of a standard cotton glove.



3) 10 kΩ Resistors (5 ×): Paired with flex sensors to

form voltage dividers. This yields a voltage range of roughly 0.7 V–4.0 V corresponding to finger curvature.

4) HC-05 Bluetooth Module: Classic Bluetooth SPP module operating at 3.3 V logic. An on-board voltage divider (two 10 kΩ resistors) is used to drop the Arduino's 5 V TX line to 3.3 V for the HC-05 RX pin.



5) 16 × 2 I2C LCD: Provides on-glove visual feedback of the most recently recognized letter and the running word buffer. Typical I2C address is 0x27 (user may run an I2C scan to confirm).


6) Pushbutton (SPST, 1 ×): Connected between Arduino digital pin 2 and GND. Uses internal pull-up resistor. Pressing connects to ground, signaling a buffer clear.

7) USB Power Bank (5 V, ≥1000 mA): Supplies all electronics via the Arduino's USB-B input, making the glove fully portable.
8) Breadboard or Perfboard: Used for prototyping and wiring the flex sensor voltage dividers, pushbutton, HC-05, and LCD wiring before final assembly.
9) Jumper Wires (Male–Female, Male–Male): For connecting components between the Arduino, sensors, LCD, and Bluetooth module.
10) Velcro Straps or Electrical Tape: To secure sensors, wires, and small circuit boards to the glove.

Hardware Interfacing and Wiring

1. Flex Sensor Voltage Dividers (per finger)
- Top node (5 V) → Flex Sensor → Analog Pin (A0–A4) → 10 kΩ Resistor → GND
- Connect the thumb's flex sensor to A0, index to A1, middle to A2, ring to A3, and little to A4.
- Route wires along the glove's back, securing sensors to the fingertip crease to maximize bending sensitivity.

2. Pushbutton (Clear Buffer)
- One leg of the pushbutton → Arduino digital pin 2
- Other leg → GND
- Configure pin 2 as INPUT_PULLUP in code; reading LOW indicates a press.

3. 16 × 2 I2C LCD
- SDA → A4
- SCL → A5
- VCC → 5 V
- GND → GND
- Ensure the I2C backpack address matches the code (commonly 0x27 or 0x3F). If uncertain, run an

I²C address scanner sketch first.

4. HC-05 Bluetooth Module
- VCC → 5 V
- GND → GND
- TXD (HC-05) → Arduino pin 10 (SoftwareSerial RX)
- RXD (HC-05) ← Arduino pin 11 (SoftwareSerial TX) through two 10 kΩ resistors in series (to drop 5 V to ~3.3 V).

5. Power
- Connect a USB cable from the power bank's USB-A output to the Arduino Uno's USB-B port. Switch the power bank on to begin operation.

Make sure to insulate all exposed solder joints with heat shrink or electrical tape to prevent short circuits. Mount the Arduino (and possibly the HC-05 and a small prototyping board) on the glove's wrist area using Velcro or a small pouch sewn onto the glove.

## III. EXPERIMENTS AND TESTING

### A. Participant Selection and Trial Protocol

Ten volunteer participants (age 20–45, mixed gender, right-handed, naïve to the glove system) were recruited. Each user donned the glove, completed the 5 s calibration, and then performed five repetitions of each of the eight target letters (A, B, C, D, E, I, L, T) in randomized order, for a total of 400 trials (10 users × 8 letters × 5 repetitions). For each trial:

1. The participant was prompted with a target letter.
2. They formed the corresponding ASL handshape and held it for approximately 1 s.
3. The glove's LCD displayed the detected letter (if any).
4. Simultaneously, the Android app printed the character to screen and vocalized it via TTS.
5. The participant then released the pose, waiting for a visual cue to proceed to the next letter.

### B. Accuracy Measurement

Ground truth for each trial was recorded manually by having an observer note whether the glove's recognized letter matched the intended sign. If the LCD displayed an incorrect or no letter after 1 s, the trial was labeled a misclassification. Overall accuracy was computed as Accuracy = (Number of Correct Recognitions / Total Trials) × 100%.

### C. Latency Measurement

End-to-end latency was measured using a high-speed camera (120 fps). In each recorded trial:
- T1 = frame where the user's hand first achieved the correct static pose (visually).
- T2 = frame where the phone's speaker began sounding the recognized character (audible cue).

Latency = (T2 – T1) × (1/120 s). A total of 365 correctly recognized trials were analyzed to compute mean and standard deviation of latency.

### D. User Survey

After completing all trials, participants filled out a short questionnaire on a 5-point Likert scale (1 = strongly disagree to 5 = strongly agree) covering:

1. Comfort of wearing the glove for ≥10 minutes.
2. Perceived reliability of recognition.
3. Willingness to use the device in real-world scenarios.

## IV. RESULT AND DISCUSSIONS

### A. Recognition Accuracy

The glove's performance on eight static ASL letters demonstrated that simple threshold-based logic, combined with per-sensor calibration, can yield over 90% accuracy for a small vocabulary. Specifically, "E" and "A" exhibited the highest recognition rates (96.0% and 94.0%, respectively), likely because their finger positions are the most distinct— "E" is a fully closed fist with thumb against the side, and "A" has only the thumb extended. "L" had the lowest rate (86.0%) because its configuration (thumb and index extended, other fingers bent) sometimes resulted in the middle finger's angle falling near its threshold boundary. Overall, an average of 91.3% indicates that most static hand poses are well captured by the calibration-derived thresholds.

### B. Latency Analysis

With mean latency measured at 230 ms, the system responds quickly enough for letter-by-letter spelling in conversation. Users did not perceive noticeable lag, though faster response (below 200 ms) would feel more instantaneous—especially if consecutive letters are spelled rapidly. The majority of delay stems from the Android app's 200 ms polling interval; reducing that to 100 ms or adopting interrupt-driven Bluetooth

reads could bring average latency under 180 ms. Arduino-side optimizations—such as reducing the number of analog samples from 12 to 8—could save another 20–40 ms at the cost of slightly increased noise. Overall, the current design strikes a reasonable balance between responsiveness and stable readings.

C. User Feedback

Users rated comfort (4.2/5) and reliability (3.9/5) favorably. The main issues were minor finger fatigue after extended holding of poses and occasional misclassifications causing slight user frustration. When asked if they would use the device in real-world settings, the average rating was 3.6/5. Many respondents cited bulkiness—the Arduino Uno and breadboard mounted on the wrist—as a deterrent to everyday use. They recommended migrating to a smaller microcontroller (e.g., Arduino Nano) and designing a custom PCB that integrates flex sensor inputs, Bluetooth, and display drivers in a compact enclosure.

D. Discussion and Future Improvements

1. Calibration Enhancement: The two-phase minimum/maximum approach captures extremes, but adding an intermediate "half-bend" sample could help refine threshold placement and reduce misclassification for borderline poses (e.g., L vs. D).
2. Expanded Gesture Set: Dynamic gestures like J and Z (which require motion detection) are currently unsupported. Integrating a 6-axis IMU (such as an MPU-6050) would allow detection of motion-based letters and hand-orientation changes.
3. Latency Reduction: Lowering the Android app's Clock interval to 100 ms or porting the code to a platform that supports interrupt-driven Bluetooth events would reduce the round-trip time for each letter. Arduino loop optimizations (e.g., fewer analog samples) also help.
4. Miniaturization and Power: Shifting from an Arduino Uno to an Arduino Nano 33 BLE Sense (with built-in BLE and IMU) could dramatically shrink size, reduce power consumption, and enable TinyML classification for an expanded vocabulary. A custom PCB with an integrated Li-Po battery and charger would yield a sleek, all-in-one wearable.

## V. CONCLUSION

We have demonstrated a Wireless Sign-to-Speech Glove that converts eight static ASL letters into audible speech with an overall accuracy of 91.3% and a mean latency of 230 ms. By using only five flex sensors, an Arduino Uno, an HC-05 Bluetooth module, and a 16×2 I²C LCD, the system remains cost-effective (≈ $60), easy to assemble, and fully wearable. A companion Android app receives recognized letters over Bluetooth, displays them, and invokes on-device TTS for immediate speech output.

User trials with ten participants confirmed comfort for short durations and acceptable reliability. The primary challenges identified were borderline finger angles causing confusion between similar poses (e.g., L vs. D) and perceived bulkiness of the wrist-mounted electronics. Future work will focus on:
1. Improving Accuracy: Introduce intermediate calibration steps, dynamic thresholds, or simple hysteresis to handle edge-case finger positions.
2. Expanding Gesture Range: Add dynamic gestures (J, Z) by incorporating a 6-axis IMU for motion detection.
3. Reducing Latency: Optimize the mobile app's data polling or adopt interrupt-driven Bluetooth reads to bring latency below 200 ms.
4. Miniaturization: Transition to a smaller microcontroller (e.g., Arduino Nano 33 BLE Sense), design a custom PCB, and integrate a compact Li-Po battery for a fully self-contained wearable.

performance. This work was partially supported by the University Innovation Fellowship 2024–25.

## REFERENCES

[1] H. Kobayashi, Y. Sato, and N. Yamashita, "A Wearable Sign Language Translation Glove Using Bend Sensors," in Proc. IEEE Int. Conf. Wearable Computing Architectures and Algorithms, Sept. 2016, pp. 23–27.
[2] R. Patel, S. Chaudhary, and T. Singh, "Design and Implementation of a Hand-Glove for Real-Time Sign Language Recognition," IEEE Sensors J., vol. 18, no. 4, pp. 1527–1534, Feb. 2018.
[3] T. Nguyen, M. Hoang, and P. Tran, "Combining Flex Sensors and sEMG for Robust ASL Letter

Recognition," IEEE Trans. Neural Syst. Rehabil. Eng., vol. 27, no. 12, pp. 2515–2523, Dec. 2019.

[4] A. Sharma and V. Gupta, "TinyML-Based Sign Language Recognition on Arduino Nano 33 BLE Sense," in Proc. IEEE Int. Conf. Embedded Syst. Design, Jan. 2020, pp. 85–90.

[5] S. Chatrathi, R. Rajan, and M. Patel, "Vision-Based Sign Language Interpreter for Mobile Platforms," IEEE Trans. Multimedia, vol. 22, no. 7, pp. 1753–1762, Jul. 2020.

[6] S. Basha, P. Bhagwat, and K. Sundar, "Bluetooth-Based Real-Time Hand-Gesture Recognition System for Deaf-Mute Communication," Int. J. Adv. Res. Electr. Electron. Instrum. Eng., vol. 8, no. 5, pp. 1925–1932, May 2019.

[7] A. Kumar and R. Desai, "Real-Time Wireless Sign Language Interpreter on Android," in Proc. IEEE Global Conf. Signal Inf. Process., Nov. 2021, pp. 310–314.

[8] P. Heo, J. Kim, and S. Lee, "Evaluation of Wearable Sign Language Translation Devices," IEEE Access, vol. 9, pp. 145922–145935, Nov. 2021.

[9] Spectra Symbol, "Flex Sensor Model 2.2 Datasheet," Spectra Symbol, Inc., Salt Lake City, UT, Tech. Rep. FS-2.2, 2017.

[10] MIT App Inventor Team, "App Inventor 2 Blocks Reference," Apr. 2025. [Online]. Available: http://ai2.appinventor.mit.edu/.