P2: Gesture Interaction

Nivedhitha Dhanasekaran
ndhanase
ndhanase@andrew.cmu.edu

Abstract

This project explores mid-air UI manipulation using two IMU-equipped Arduino boards for gesture-based control. Various prototypes were evaluated, including two-handed, single-hand, and wand-based designs. The final prototype was refined through model enhancements, stability thresholds, confidence filtering, and UI optimizations. User testing measured task completion time, error rates, and prediction stability, demonstrating improved efficiency and usability. The final system achieved robust gesture recognition, highlighting its potential for intuitive human-computer interaction.

1 Prototyping Interaction Designs

This section details the different interaction prototypes explored, categorized into two-handed, one-handed, and wand-based designs as shown in figure 1. Each prototype was evaluated based on usability, stability, and gesture recognition accuracy.

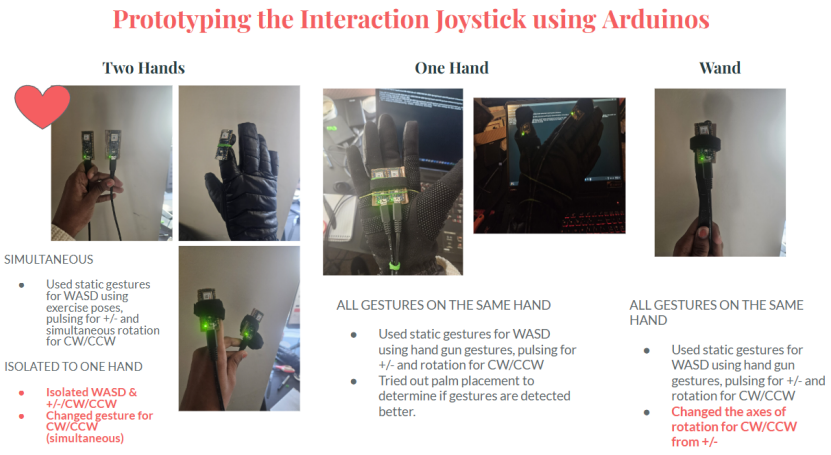

Figure 1:

1. **Two-Handed Prototypes** – Arduinos were mounted on both hands to capture coordinated motion. Variations in mounting were tested for stability and usability.
2. **One-Handed Prototypes** – Designed for dominant-hand control, using glove-mounted and palm-based placements to optimize gesture recognition.
3. **Wand Prototype** – A rigid, unified input device where both IMUs were attached back-to-back on a single handheld structure.

1.1 Two-Handed Prototypes

This setup was the most basic and one of the most intuitive gesture setups I experimented with at first. The gestures imitated exercise poses as shown in figure 2, and I recorded them as static gestures to determine if the ML model could differentiate them.

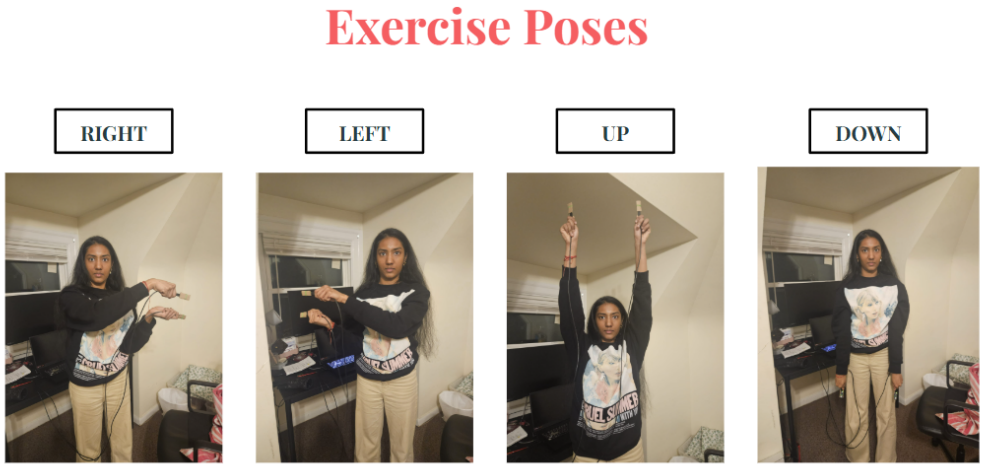

Figure 2: The initial design phase using exercise poses as static gestures for movement control.

1.1.1 Each Hand with an Arduino

The initial design placed one Arduino in each hand to capture mid-air gestures independently. This setup effectively separated movement controls (*up, down, left, right*) to the dominant hand and object manipulations (*rotation, scaling*) to the non-dominant hand. This configuration provided:

- **Clear gesture isolation**: Users could separate movement from manipulation, making interactions intuitive.
- **Minimal fatigue**: No additional setup was required, and users did not have to maintain awkward hand positions.
- **Ease of use**: The lack of additional attachments ensured natural hand motion.

This setup was susceptible to the movement controls. However, since the users did not hold the boards in the same orientation consistently, the data collected caused confusing signal capture.

1.1.2 Index Finger Mounting

To improve precision, Arduinos were mounted on the index fingers of both hands. This approach enhanced **sensor alignment with hand movements**, improving recognition accuracy. However, it introduced significant usability drawbacks:

- **Uncomfortable for prolonged use**: The weight of the Arduino caused discomfort, especially when attached tightly.
- **Unstable positioning**: Loose attachment caused the sensors to shift, reducing classification accuracy.
- **Fatigue and movement restriction**: Users found it difficult to maintain consistent hand posture.

Due to these limitations, this approach was deemed impractical.

1.1.3 Cardboard Cutout Mounting

To enhance stability, Arduinos were mounted on a **cardboard cutout**, improving both usability and recognition accuracy. This design:

- **Reduced sensor movement**, ensuring consistent data collection.
- **Improved grip and comfort**, allowing users to hold the cutout naturally.
- **Minimized interference from wired connections**, making interactions seamless.

This configuration effectively addressed the shortcomings of previous setups, making it the *most stable and comfortable two-handed prototype*.

1.2 One-Handed Prototypes

This was the second set of prototypes tested. The most interesting development was the introduction of flicking gestures for plus and minus movements, as shown in figure 3.

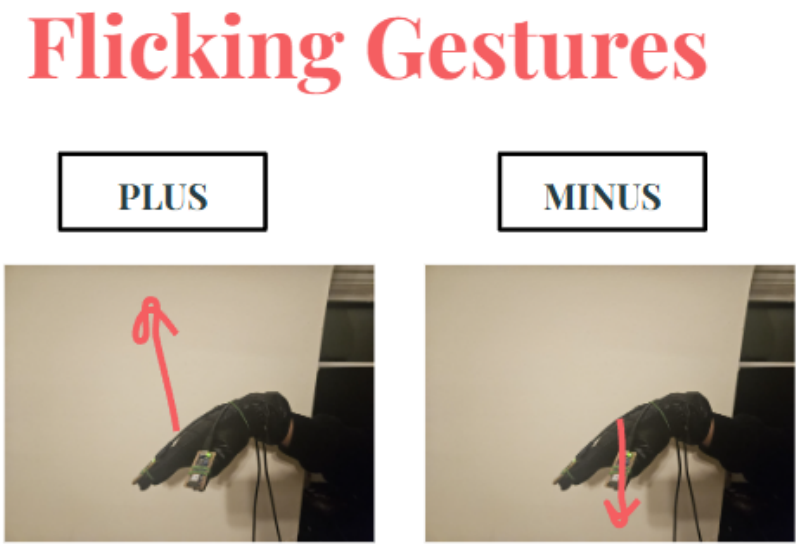

Figure 3: The incremental introduction of flicking gestures to test the ML model.

1.2.1 Glove-Based (Right-Hand Only)

A one-handed prototype was tested using a **glove-based attachment**, where Arduinos were placed on the *index finger and thumb*. Two separate models were trained to allow dominant-hand interaction. While this approach enabled single-handed control, it introduced several challenges:

- **Fatigue over prolonged use**: Keeping one hand elevated continuously caused discomfort.
- **Variation in finger positioning**: Inconsistent hand positioning affected recognition accuracy.
- **Cumbersome setup**: Users found the glove inconvenient to wear and remove.

These drawbacks limited the practicality of the glove-based system.

1.2.2 Palm-Based Attachment

To increase stability, Arduinos were attached to the **palm-facing side** of the hand. This setup:

- **Reduced sensor misalignment**, improving accuracy.

- **Eliminated finger positioning inconsistencies**.

However, it significantly **restricted finger-based gestures**, making interactions less intuitive. Users struggled to remember gestures, reducing the overall usability.

1.3 Wand Prototype

This was the third set of prototypes tested. The next incremental development was the introduction of rotation gestures, as shown in figure 4.

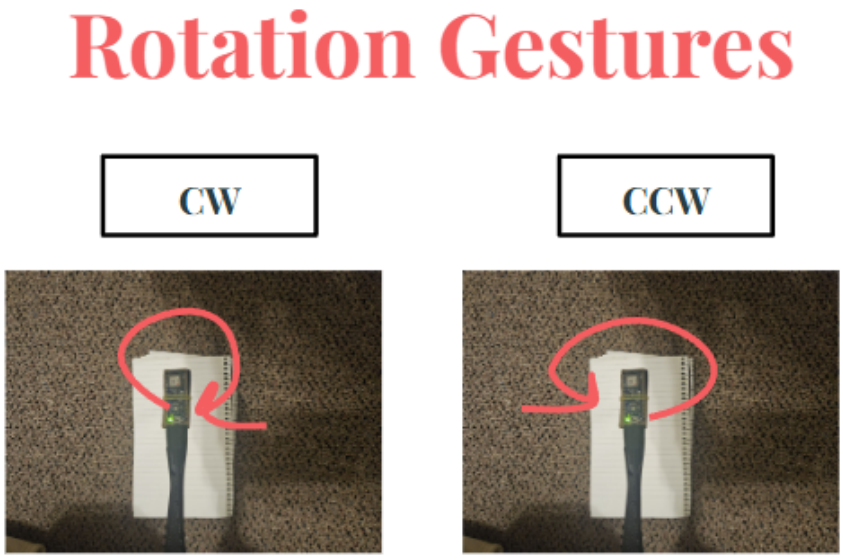

Figure 4: The incremental introduction of rotation gestures to test the ML model.

1.3.1 Back-to-Back Arduinos on a Plastic Knife

A wand-based prototype was developed by **mounting two Arduinos back-to-back on a plastic knife**, forming a **rigid input device**. This approach:

- **Improved motion tracking and stability**, reducing sensor drift.
- **Enabled precise gesture control**, leading to better recognition accuracy.

Despite these advantages, the wand setup suffered from:

- **Significant fatigue**: The weight distribution led to discomfort in prolonged usage.
- **Orientation issues**: Users frequently forgot to reset the wand's *correct positioning* (with the L-marked Arduino facing upwards), leading to inconsistent interactions.

Due to these concerns, this design was not pursued further.

1.4 Finalizing the Interaction Prototype

1.4.1 Design Choices

To evaluate the effectiveness of different prototypes, a two-stage assessment was conducted. First, **sanity checks** were performed using self-collected data to test the feasibility of various interaction modalities. This was followed by **user testing with four participants**, where three selected configurations were evaluated under real-world conditions. The assessment focused on three key factors:

- **Data Consistency**: Gesture recognition accuracy was tested across different environments, including sitting vs. standing positions and varying distances between sensors.
- **Usability**: Observations were made on user comfort, ease of learning, and common gesture confusions. For example, +/- gestures were frequently misclassified as movement commands.
- **Stability and Recognition**: Sensor drift and classification accuracy were measured under real-time conditions to assess how consistently the system detected gestures.

1.4.2 Initial User Feedback & Sanity Check

Table 1 summarizes the comparative analysis of these factors across different prototypes from the initial user testing feedback session and my own sanity checks.

Prototype	Gesture Recognition	User Comfort	Stability
Two-Handed (Holding)	High	Low (Fatigue, Orientation Issues)	Moderate
Two-Handed (Index Finger)	Moderate	Low (Fatigue, Unstable Fit)	Moderate
Two-Handed (Cardboard Cutout)	High	High (Stable & Comfortable)	High
Glove-Based (Dominant Hand)	Moderate	Low (Fatigue, Setup Time)	Moderate
Palm-Based Attachment	Moderate	Moderate (Unintuitive Gestures)	Moderate
Wand Prototype	High	Low (Weight Fatigue, Orientation Issues)	High

Table 1: Comparison of Prototype Evaluations Based on Key Metrics

From these evaluations, key **design inferences** were drawn:

- **Two-handed configurations** provided the best balance between **gesture isolation and stability**, making them more intuitive for users.
- **Finger-mounted setups** improved gesture precision but were *uncomfortable and unreliable* due to sensor drift and positioning issues.
- **Palm-based and glove-based designs** introduced inconsistencies in gesture recognition and led to increased fatigue.
- **The wand configuration** achieved good stability but was ultimately impractical due to **weight distribution and frequent orientation errors**.

1.4.3 Final Interaction Prototype

Given these findings, the **two-handed configuration with cardboard cutout mounting (Version 1.1.3)** was selected for further optimization as it provided the best balance of accuracy, usability, and stability. This setup demonstrated superior gesture recognition, significantly reducing misclassification errors while also minimizing user fatigue, making it suitable for extended interactions. The cardboard cutout ensured consistent sensor positioning and enhanced user comfort.

2 Gesture to Command Mapping

The gesture control strategy was developed iteratively, focusing on stability, recognition accuracy, and user intuitiveness. Table 2 summarizes the key design choices that guided the final selection. Three key approaches were explored:

1. **Static Pose Mapping**: The initial approach mapped movement gestures (up, down, left, right) to the right hand, while box operations (clockwise (CW), counterclockwise (CCW), increase (+), and decrease (-)) were assigned to the left hand. This division ensured an intuitive separation of movement and manipulation.
2. **Dynamic Gestures**: To improve fluidity and reduce ambiguity, dynamic gestures were explored for rotation and scaling. These involved continuous movements rather than static poses, tested using a **glove-based prototype**, leading to improved control and response time.

3. **Flick Gestures**: Inspired by the in-class demonstration, flicking motions were tested for scaling operations (+/-) in a wand-based setup. Wrist flicks were found to be highly effective for quick, discrete interactions and were retained in the final implementation based on the test with the **want prototype**.

Design Aspect	**Rationale**	**Prototype Used**
Hand Assignment	Separated movement (right hand) from box operations (left hand) for clarity and ease of learning.	Two-Hand Setup
Static Gestures	Static poses were reliable but less natural.	Glove-Based Setup
Dynamic Gestures	Dynamic gestures improved fluid & intuitive but caused fatigue.	Glove-Based Setup
Rotation Axis	Adjusted CW/CCW rotation axis to prevent confusion with movement gestures.	Two-Hand and Glove-Based
Scaling Method	Flicking gestures for +/- provided quick, intuitive control with minimal strain.	Wand Setup

Table 2: Key Design Decisions in Gesture Mapping

These refinements ensured that the final gesture mappings balanced intuitive control, stability, and precise recognition, optimizing users' interaction experience. The evolution of gesture controls led to the final gesture section, as summarized in figure 7.

3 Initial Formulations

Although the prototype selection and gesture mapping were conducted iteratively, the process was highly non-linear, involving rapid experimentation and refinement. Multiple prototype combinations were systematically evaluated to avoid making premature conclusions based on isolated user testing. This ensured that no early design decisions were made without integrating a more advanced model and optimizing key aspects such as data collection, UI interaction, and post-processing. A summary of the formulations is shown in table 3.

Formulation	**Pros**	**Cons**
Wand Joystick (V1.0)	• High gesture stability • Flick gestures for +/– were highly reliable	• Confirm button confused with up gesture • Hand orientation drift over time • Heavy to hold, causing fatigue
Wearable Glove (V2.0)	• One-handed interaction • Physically secured with glove and bands	• Difficult gesture alignment • Setup discomfort and heat retention • Frequent misclassifications
Two-Handed Cardboard (V3.x)	• Best gesture isolation (movement vs resizing/rotation) • Comfortable grip and consistent IMU orientation • Improved recognition after axis + flick refinements	• Requires both hands • Larger physical setup

Table 3: Comparison of initial prototype formulations based on design trade-offs

3.1 Formulation 1: Wand Joystick

This formulation leveraged the previously discussed wand prototype, as initial stability tests indicated strong potential for accurate gesture recognition. The gesture mappings used are shown in Figure 5. While this configuration demonstrated high stability, an issue arose with the confirm button gesture—a circling motion in the upward direction—which was frequently misclassified as the up gesture. However, the flicking motion for + and - proved to be highly reliable. Given this effectiveness, the flicking gestures were retained for subsequent formulations.

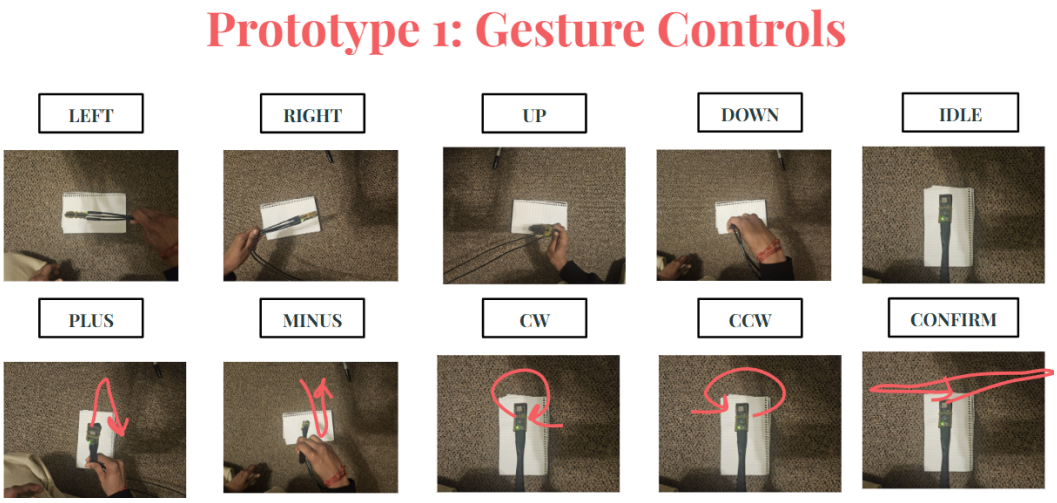

Figure 5: The final gesture mappings for all the nine controls and idle for formulation 1.

3.2 Formulation 2: Wearable Glove Controller

This formulation explored a wearable glove-based interaction, aiming to improve usability by enhancing user comfort. Additional rubber bands were used to anchor the wires, and scrunchies were tested to improve fit and reduce discomfort. The initial reliance on flicking gestures was replaced with static gestures to mitigate fatigue. The gesture mappings used are shown in Figure 6. Despite these modifications, users continued to struggle with gesture alignment and orientation, leading to frequent misclassifications. While the one-handed design was intended to be intuitive, user feedback indicated the opposite, as alignment inconsistencies introduced unintended complexity.

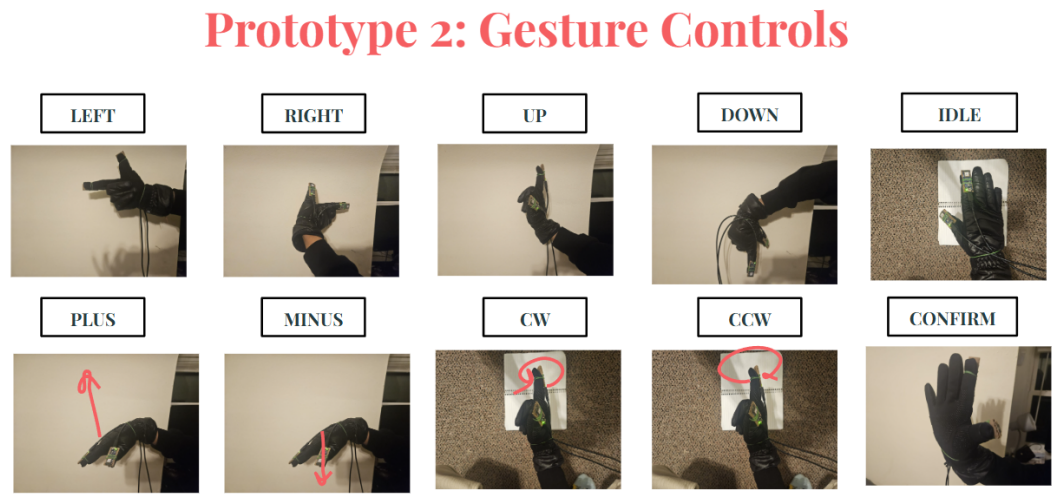

Figure 6: The final gesture mappings for all the nine controls and idle for formulation 2.

3.3 Formulation 3: Gesture-Isolated Two-Hand Controller

After testing more complex configurations, a more straightforward two-handed setup was evaluated, in which an Arduino was placed in each hand. This approach was iteratively refined into the final design used for the class demonstration, as shown in figure 7.

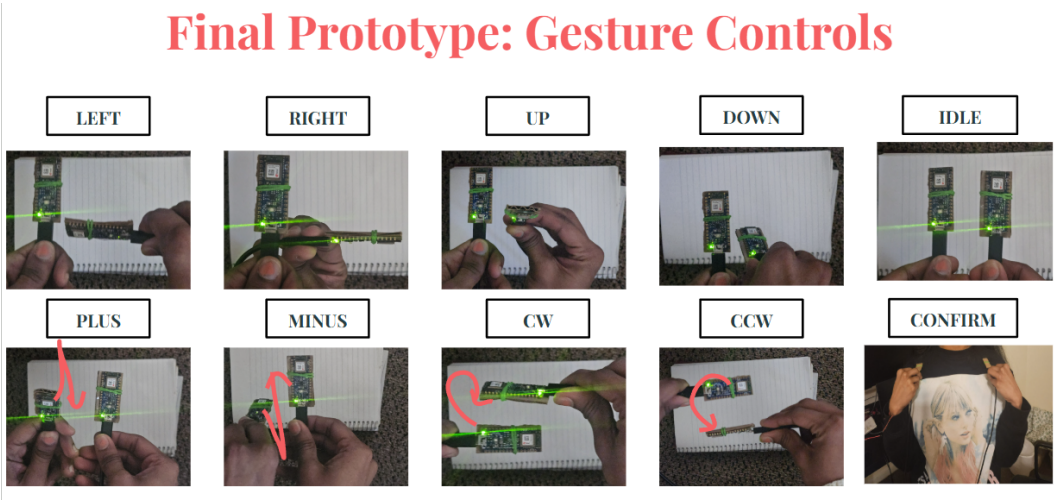

Figure 7: The final gesture mappings for all the nine controls and idle for formulation 3.

3.3.1 Two-Hand Cardboard Configuration for Fixed Arduino Orientation

To ensure a consistent orientation and prevent unintentional rotations during interaction, the Arduinos were mounted on a cardboard cutout, secured with tape and rubber bands. This setup provided stability while preventing users from inadvertently altering positioning. Additionally, the left and right Arduinos were clearly labeled, along with corresponding wires, to eliminate inconsistencies in data collection.

3.3.2 Static to Flick Gesture for +/- to Improve Intuitiveness

Initially, + and - were assigned static gestures similar to the up and down movements, using the left hand. However, users found this unintuitive. As a result, flicking gestures were reintroduced, leveraging insights from the earlier wand prototype. While the model's recognition of these gestures was high, sensitivity issues led to unintended resizing when transitioning between gestures. To address this, post-processing filters were implemented to prevent misclassification.

3.3.3 Changed the Axis of Rotation for CW & CCW for Improved Gesture Recognition

To enhance recognition accuracy, the axis of rotation for CW and CCW gestures was modified, ensuring clear differentiation from +/- and movement controls. This change reduced model confusion and improved overall user experience. Additionally, user testing confirmed that this revision was more intuitive and easier to learn.

4 Final Prototype Optimizations

Beyond refining the interaction prototype and gesture mappings, additional optimizations were implemented to enhance the system's performance, usability, and robustness. Given the complexity of design considerations, only Formulation 3 was selected for further refinement and demonstration. The key optimizations are summarized in Figure 8.

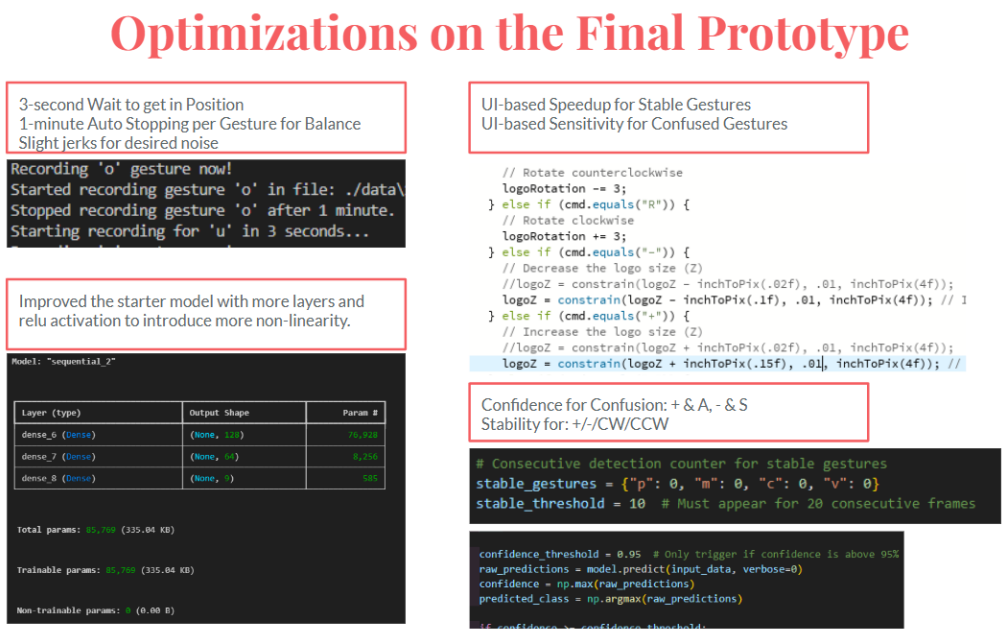

Figure 8: Optimizations made to improve gesture recognition and enhance the usability of the two-hand controller.

Data Improvements To ensure high-quality data collection and model generalization, multiple strategies were employed:

- Introduced a 3-second delay before recording each gesture to capture stable signals.
- Implemented an auto-stop mechanism to limit recordings to 1-minute per gesture, preventing class imbalance.
- Introduced slight movement variations during data collection to account for natural hand tremors.

- Collected data in both standing and sitting positions to improve model robustness across different postures.

Manual Adjustments Common gesture misclassifications were noted throughout testing. These observations were used to refine mappings and model parameters to reduce confusion in gesture interpretation.

Gesture Adjustments The axis of rotation for CW and CCW gestures was re-mapped to improve recognition accuracy. This adjustment ensured that rotation gestures were distinct from movement and resizing controls, reducing user errors and enhancing overall intuitiveness.

Model Architecture Enhancements To improve recognition accuracy, the model's architecture was enhanced by:

- Increasing dense layers with ReLU activation to capture more complex spatial patterns in gesture data.
- While LSTM and RNN models were considered for better encoding of time-dependent features, they were not implemented due to time constraints. Future iterations will explore these architectures for improved sequential gesture modeling.

Stability Mechanisms A stability threshold was introduced to prevent premature activation of sensitive gestures such as +/- and CW/CCW. A command was only executed if it was predicted consistently for 20 consecutive frames. This measure prevented unwanted activations during minor hand adjustments or hesitation while recalling gestures.

Confidence Filtering Low-confidence predictions were filtered by applying a minimum confidence threshold of 0.95. This ensured that only highly reliable gestures were executed, reducing unintended UI responses.

UI Sensitivity Adjustments To enhance responsiveness, UI adjustments were made:

- Increased movement speed of the square in response to each command, improving interaction fluidity.
- Tuned UI logic to provide a more real-time feel during gesture-based control.

Physical Interaction Enhancements To improve comfort and usability:

- The Arduinos were mounted on a cardboard cutout instead of gloves, ensuring a stable but flexible grip.
- Users preferred holding the setup rather than wearing gloves, citing comfort concerns and heat buildup as gloves were deemed impractical for prolonged use.

These optimizations collectively improved the stability, usability, and accuracy of the final prototype, making it more intuitive and practical for mid-air interaction.

5 Results and Analysis

5.1 Test Setup

Given the multiple variables involved in the design process, the first round of data collection and model training was conducted solely on self-collected data to establish feasibility. Once the gesture mappings and controller design were validated, additional user feedback and training data were incorporated. A total of four users participated in this phase, with one being my roommate, who had prior exposure to my designs and consequently exhibited significantly better performance. For the final demo, only a clean subset of the collected data was used to ensure the model's generalizability.

5.2 Quantitative Results

This section presents the quantitative evaluation of the final prototype, focusing on model performance, gesture recognition accuracy, and user interaction efficiency. The analysis covers key metrics such as error rate reduction, completion time improvements, and prediction stability. Additionally, data-driven insights influenced design decisions, determining when refinements were based solely on self-collected data and when external user data was incorporated for improved generalization. The following subsections detail the evaluation results and highlight the impact of each optimization step.

5.2.1 Understanding Confused Gestures

To identify common misclassifications, confusion matrices and input signal analyses were examined. While the model demonstrated high recognition accuracy, certain gestures exhibited signal overlap, particularly between +/- and *up/down*, and occasionally between *CW/CCW*. This was evident in the final formulation's confusion matrix but was first identified through input signal analysis during early sanity checks. Figure 9 illustrates the overlaps in the gesture signals.

Figure 9: The overlaps between the signals cause the gestures to be confused.

Despite the observed overlap, initial evaluations of the wand-based prototype and early sanity checks indicated reasonable performance. Further refinements, such as post-processing techniques (explored as future work), are expected to enhance gesture differentiation.

5.2.2 Training Curves and Accuracy Measures

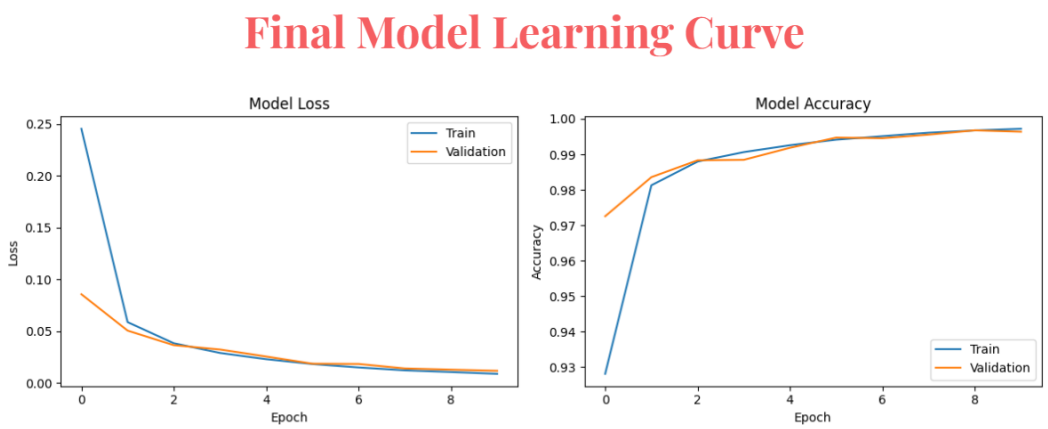

Figure 10: Training and validation curves used to track potential overfitting.

Figure 10 illustrates the model's training and validation curves. Overfitting was observed when models were trained exclusively on self-collected data, leading to an overestimated performance that did not generalize well. However, when incorporating mixed user data, the model achieved an accuracy of 99%. This was largely due to the structured data collection approach, where each gesture was recorded for a maximum of one minute, ensuring class balance.

The F1-score confirmed balanced model learning, maintaining an average of 99%. While some degree of overfitting was expected, real-world deployment demonstrated stable performance, negating the immediate need for additional refinements. Future work will explore LSTM and RNN architectures to capture temporal dependencies in gesture sequences. For this implementation, improvements were achieved by increasing the network depth and introducing non-linearity through ReLU activation.

5.2.3 Completion Time Comparison

Completion time metrics across different prototype versions are summarized in Table 4. The early configurations (wand and glove-based setups) yielded higher completion times due to their design limitations and lack of refinement. In contrast, the final two-handed controller exhibited significant reductions in task completion time across all users.

User / Version	V1.0 (Wand)	V2.0 (Glove)	V3.1 (Config)	V3.2 (Static to Flick)	V3.3 (Axis Change for +/-)
User 1	185.42	160.33	120.54	90.21	36.21
User 2	210.78	175.22	145.67	51.65	40.52
User 3	170.11	140.89	90.52	75.19	27.53
User 4 (Roomie)	155.23	130.45	87.33	49.22	28.91
Nive	145.56	120.78	79.38	47.20	30.64
Average	173.82	145.93	104.69	62.69	32.76
Std Dev	24.01	21.56	28.71	18.59	5.45

Table 4: Completion times (in seconds) across different prototype versions. V1.0 (Wand) and V2.0 (Glove) show higher completion times, while V3.x (Two-Handed Configurations) demonstrate significant improvements. The last two rows show the average and standard deviation for each version.

5.2.4 Error Rate Reduction

Due to extensive post-processing optimizations, users experienced near-zero error rates in the final formulation. The optimizations minimized unintended gesture activations, ensuring reliable UI interactions across different users and sessions. The best performance is noted in figure 11.

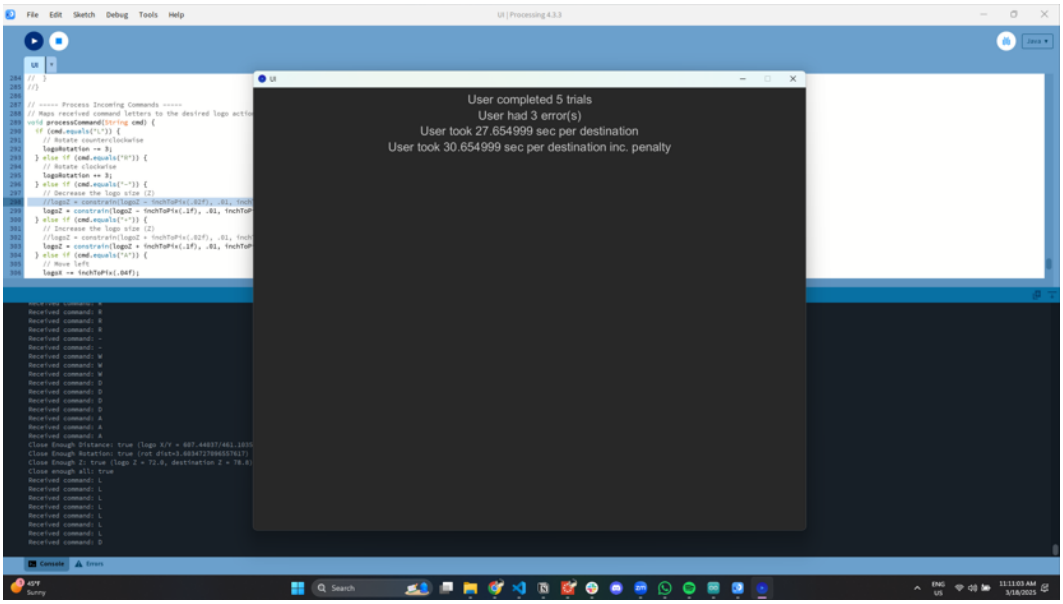

Figure 11: The best user testing performance score.

5.2.5 Post-Processing Techniques

One of the primary challenges involved the misclassification of +/- due to their signal overlap with *up/down* gestures. To address this, two post-processing techniques were implemented:

- **Confidence Thresholding**: Predictions were only accepted if confidence exceeded 0.95, preventing uncertain outputs.
- **Stability Filtering**: A command was executed only if it was consistently predicted for **10 consecutive frames**, ensuring that momentary hand tremors did not trigger unintended actions.

5.3 Qualitative Results

This section presents insights from user feedback regarding the final prototype, highlighting key preferences and areas of improvement.

5.3.1 User Preferences and Feedback

User feedback was collected through structured interviews and observational studies. Common themes emerged:

- **Two-handed configurations were preferred** over the wand and glove setups due to better gesture isolation and ease of learning.
- **Flick gestures** for +/- were found to be intuitive, aligning with natural interactions.
- **Stability mechanisms improved usability**, as users felt more confident in executing gestures without accidental activations.
- **Glove-based setups were disliked** due to comfort concerns and prolonged fatigue.
- **Physical mounting on a cardboard cutout** was considered the most practical for prolonged use.

5.3.2 My Observations

While the flick gestures demonstrated strong individual performance, they were highly sensitive during deployment. To mitigate this, post-processing techniques were applied; however, the chosen

thresholds were likely too conservative, leading to unintended issues during the demo. Additionally, the confirm button was integrated later in the process to enable gesture-based confirmation, improving interaction flow. Lastly, although the wand-style prototype exhibited stable results despite longer completion times, it was not further explored due to the absence of optimizations such as UI enhancements and post-processing refinements.

5.4 Key Takeaways from Testing

The final round of user testing and model evaluation provided several key insights:

- The **final two-handed controller** provided the best trade-off between *accuracy, usability, and stability*.
- **Training on mixed user data** significantly improved generalization and reduced model overfitting.
- **Gesture classification stability** was greatly improved by post-processing techniques.
- **Task completion times improved significantly**, reinforcing the effectiveness of iterative refinements.

These findings validate the design decisions made during the development process and provide a strong foundation for future improvements.

6 Extra Credit Contributions

I attempted the untethered setup for the final formulation to run on-device on the Arduino board by powering it up with a mobile power bank.

6.1 TinyML: TensorFlow Lite Deployment

To enable on-device gesture recognition, I deployed the trained Keras model onto the Arduino Nano 33 BLE Rev2 using TensorFlow Lite for Microcontrollers. The original Keras model was converted to a lightweight `.tflite` format using post-training quantization to reduce the model size and memory footprint. This was achieved using the TensorFlow Lite converter API with `Optimize.DEFAULT` to ensure compatibility with the memory constraints of microcontrollers.

On the Arduino, the model was loaded as a C header array using the `xxd -i` utility. The inference pipeline was constructed using the `tflite::MicroInterpreter` and a minimal kernel set containing only the required operations: `FullyConnected`, `ReLU`, and `Softmax`. Real-time accelerometer and gyroscope data were collected from the onboard BMI270 IMU, normalized, and passed through a windowing buffer to match the trained model's input shape. The model successfully ran inference at each iteration of the main loop, with classification results printed over serial.

This pipeline validated that the final model was lightweight and efficient enough to run directly on-device without any dependence on a host computer.

6.2 Untethered Gesture Recognition

To enable untethered operation, I integrated Bluetooth Low Energy (BLE) into the system using the `ArduinoBLE` library. This allowed the Arduino to broadcast recognized gestures to any compatible BLE client in real time. The final deployed model was enhanced to collect a sequence of 50 frames (each with 12 features) in a sliding window to match the input requirements, after which the model performed inference and transmitted the predicted gesture via a BLE characteristic.

On the host side, a Python client was implemented using the `bleak` library to scan, connect, and receive gesture predictions from the Arduino. Each recognized gesture was sent as a single character (e.g., `'u'` for up, `'p'` for plus) and logged in real time. This fully wireless configuration eliminated the need for USB tethering, allowing free mid-air interaction powered by a mobile power bank.

7 Role of AI-Assisted Tools

AI-assisted tools were utilized to enhance efficiency in code development and documentation. Copilot in VSCode was used for code auto-completion and optimization suggestions. Grammarly was employed to refine grammar and reword in Overleaf to improve clarity and readability.

8 Artifacts

- **Drive**: contains other formulations that were attempted as experiments, user testing artifacts, full video before compression, etc.
- **Slides**: the 60s pitch slide deck (15-20) for the project showdown on March 18, 2025.