Swish and Flick: Using Dynamic Hand Gestures to Control Smart Devices: Report

Mobile and Ubiquitous Computing - CS 7470 Spring 2024

Team What the Hack (12)

Jason Bunyea
+1 (352) 586-1797
jbunyea3@gatech.edu

Peter Hernandez Fuentes
+1 (813) 774-1630
pahf3@gatech.edu

Colin James
+1 (425) 518-5517
cjames79@gatech.edu

Jasmine Manansala
+1 (210) 251-9612
jmanansala3@gatech.edu

John West
+1 (269) 873-5731
jwest318@gatech.edu

Georgia Institute of Technology
Atlanta, GA 30332

ABSTRACT

Smart devices and smart device hubs intend to improve users' experience in interacting with everyday household objects. However, these devices lack the control methods we crave and may become even more convenient to use through a gesture-based input system. Gestures are a key component of nonverbal communication whose ability to express and relay intent can be extended to human-computer interaction. We have developed a wearable prototype that will read a user's dynamic hand gestures (using a Random Forest classifier) with the intent of using these gestures to control smart devices' actions. This prototype presents users with a more seamless method of interaction without having to rely on an intermediary device such as a remote control.

Keywords

gestures; gesture recognition; ubiquitous computing; interaction design; human-computer interaction; smart devices; gestural input

1. INTRODUCTION

In today's interconnected world, smart devices are everywhere. Our primary goal was to provide a convenient means to utilize gestures as a means of delivering input to these devices. This can be visualized as two tangible goals:

1. Design and develop a wearable prototype that recognizes the user's dynamic gestures, and
2. Have this prototype test well in terms of both system proficiency and user experience.

Our primary goal for the prototype was to reach a state where it performed highly with both quantitative and qualitative measures. We aimed to redefine the way users interact with smart devices, making it more intuitive than ever before. For our final prototype, we aimed for an average recognition accuracy (the number of dynamic gestures recognized divided by the number of deliberate dynamic gestures performed) of at least 85% amongst novice and experienced users, which we were able to both meet and exceed with the latest version of our prototype. We set an ambitious goal to design a wearable prototype that recognizes hand gestures with precision, ensuring a seamless and intuitive experience for users of all backgrounds. To gauge the user experience quality, we provided a survey after prototype user testing that included both 1-5 Likert scale ratings and free response items. Our goal for user experience was to have scores of at least 3 on the Likert scale items (indicating an average or better experience with the prototype) and positive and constructive feedback in the free responses. Our participant evaluation also exceeded our usability target by a large margin, with an average Likert scale value of 4.7.

2. BACKGROUND

Hand gestures are a universal part of nonverbal communication. They have long been used to emphasize emotions, issue commands, and express visual symbolism. Hand gestures are traditionally split into two categories [1, 2, 3]:

- **Static gestures**, which consist of a single hand "frame" at a specific position and orientation, and
- **Dynamic gestures**, which consist of a series of hand movements.

Due to the complexity of the human hand, static gestures can portray a myriad of concepts from position and orientation alone; dynamic gestures only increase this by adding movement as a parameter.

As a rich method of expressing information, gestures show potential as a possible input system for computers for several reasons. In addition to their ability to generate countless ideas from so few parameters, they also provide natural and direct interaction [1]. Hand gestures are familiar and easy to learn, and they do not require the user to express commands through a "middle-man" device (such as a remote control). However, their primary drawbacks come in the form of physical fatigue and low discoverability. As such, functional gestures should have the following qualities [1, 3]:

- *Intuitiveness*. A user should be able to easily intuit the mapping between a gesture and its respective function to increase discoverability.
- *Simplicity*. A gesture should be easily performable.
- *Conciseness*. A gesture should not contain any excessive movement.
- *Learnability*. A user should easily become familiar with and use the gestures and functions available to them.
- *Consistency*. Functionally or semantically similar functions should have similar gestures mapped to them.

In addition, the user should be given quick and appropriate feedback for their actions.

The VPL DataGlove was among the first devices to transmit a user's hand movements in real-time to an external machine [1, 7].

The DataGlove measured finger bending via flux sensors, as well as used ultrasonics to detect the hand's position and orientation. Users gave commands within an "active zone"; gestures made outside of this zone were ignored [1]. Currently, there are several existing systems that can track and process hand gestures, including wired gloves and depth-sensing cameras [3, 4].

Technology-based gesture detection faces several crucial problems, the most important of which being a high rate of false positives (the "Midas touch"). Arbitrary hand movements may accidentally be read as intentional gestures. A combination of well-suited gestures, a fine-tuned algorithm, and individual user calibration can combat this issue.

One technique for recognizing hand gestures is the support vector machine (SVM) model [2]. This approach consists of three main stages:

1. Deriving gesture classification from a series of known gestures,
2. Recognizing a known gesture from an unedited stream of otherwise random hand movements, and
3. Developing rules to filter out or discard non-gesture movements.

Due to the overwhelming number of different techniques, approaches, and technologies available in the realm of gesture recognition, it is important to select the appropriate modalities pertaining to a desired outcome or application of gesture recognition. Throughout the duration of this project, our team tested many of the different methods researched for our own needs, leading to the deliverables and results reported in this paper.

3. DELIVERABLES AND METHODS

3.1 Wearable Glove

The wearable device we created was the physical backbone of this project. The glove device consists of three main components, which can be seen in the photo provided in *Appendix 9.3: Wearable Glove*.

1. An Arduino board
2. An accelerometer chip
3. A Bluetooth module

The wearable glove was constructed using these specific components in order to create a prototype that performed all the functions we originally planned for our project while using as simple of a device design that we could accomplish.

The Arduino board houses the control program for the device as well as acts as the power source and information hub for the other device components. The accelerometer is required in order to provide a method by which the wearable device can measure movement (which is coupled to the data model discussed in Section 3.2). The wearable device uses the bluetooth module in order to connect to a computer or other device that is running the data model in the form of a Python script.

All three components act together so that the Arduino can sample the accelerometer data at a rate of 100hz and then send that data to our model for processing. The electronic connections and schematics used for the wearable are also included in *Appendix 9.1: Prototype Diagram*.

Overall, we were able to successfully create a robust wearable device that a user could put on and use to wirelessly interact with other connected devices.

3.2 Data Model

Our gesture recognition algorithm consists of 4 parts: a sliding window, gesture detection algorithm, gesture bounds tuning, and a machine learning gesture classification model.

To construct the gesture recognition algorithm, we start by deploying a sliding window mechanism on accelerometer data, which is essential for capturing dynamic movements (see *Appendix 9.2: Sliding Window*). This technique involves segmenting the continuous stream of data from the accelerometer into fixed-size windows that overlap, ensuring no gesture is cut off between windows. Each window might typically encompass a few hundred milliseconds of data, depending on the expected speed and complexity of the gestures. As the window slides over the data, it captures new segments incrementally, maintaining a constant flow of data points for analysis. This approach ensures comprehensive coverage and continuity in gesture recognition, making it highly effective for real-time applications. To mitigate the "Midas touch" problem, where unintentional gestures might be registered as commands, we implement strict thresholds and conditions within this stage to discern intentional movements from casual or idle motions.

In the next step of the process, the segments captured by the sliding window are analyzed by the gesture detection algorithm. This algorithm processes the accelerometer data to identify potential gestures based on significant changes in movement, such as spikes or unique patterns in the three-dimensional acceleration vectors. The identification process involves meticulous hyperparameter tuning for each component of the accelerometer, optimizing the sensitivity and specificity of gesture detection. Thresholds for detecting a gesture start and end are finely adjusted to distinguish between different gestures and to filter out non-gestural movements or noise, enhancing the algorithm's accuracy. Further addressing the Midas touch issue, this step includes a secondary validation where detected gestures must meet additional criteria before being classified, ensuring only deliberate gestures trigger the following actions.

Finally, the refined segments of data, now confirmed to contain gestures, are fed into a Random Forest classifier. This model is trained on a rich dataset comprising labeled examples of various gestures, allowing it to learn and predict the specific type of gesture performed. The Random Forest classifier is selected for its robust performance in classification tasks, ability to handle the intricacies of sensor data, and its effectiveness in reducing the risk of overfitting through its ensemble approach. Each tree in the forest contributes a vote towards the final classification, making the model both resilient and reliable. The output of this classifier enables the direct control of technology—such as turning on a light bulb by a specific hand gesture—with potential expansions into more complex controls like operating a television or automating door mechanisms to assist users when their hands are occupied.

3.3 User Interface

We also developed a mock-up for a potential visual interface that could be paired with our prototype. The interface mock-up was developed using the Ren'Py visual novel engine to demonstrate how it may be used with our prototype.

Firstly, the interface has a visually minimalistic "wait mode" screen that simply has text indicating that the system is waiting for the user to make a gesture ("Waiting for a gesture…"). Future versions of this interface may include animated demonstrations for suggested gestures; this would be similar to an Amazon Echo

Show suggesting auditory commands for an Amazon Alexa system.

After a gesture is detected and processed, the interface would transition to an overlay (layered on top of a smart device's base UI) that would:

1. Visually indicate that a gesture was indeed detected.
2. Say the gesture that the system predicted from the user's input.
3. Say the action that corresponds to the gesture read in (2).

These elements provide enough feedback for the user while avoiding visual clutter. Future versions of this interface may also include accompanying UI sound effects associated with common actions (e.g., a high pitched chime for turning the device on). An example of this screen may be viewed in *Appendix 9.4: Interface Mock-Up*.

4. CHALLENGES

One major change to the initial plan was the ability to actually connect to a smart device, such as an Amazon Echo (which uses the Alexa system). We found the relevant APIs difficult to work with and outside the scope of the project. While we regret that we could not make that connection within the semester, we believe that future versions of the project could indeed use these APIs to make use of actual smart device functionality.

We also had to slightly adjust our schedule to dedicate more time to model development and communication between the Arduino-based wearable and the model itself. However, we were able to compensate for this by moving up tasks that did not depend on these (such as establishing testing methods) and allocating more time to development.

Another challenge was needing multiple members to individually create the wearable glove prototype. Since our team worked remotely, we could not transfer the physical prototype amongst ourselves; multiple members had to buy the necessary components to build and test their own gloves.

5. ALIGNMENT TO CLASS

We were able to align closely with the concept from Mark Weiser called the "shroud". This encompasses technologies bridging the physical and digital worlds, including IoT and wearable technology. Our wearable prototype serves as a prime example of this, as it integrates seamlessly into the user's environment, enabling interaction with smart devices through dynamic hand gestures.

Our project addresses several challenges in ubiquitous computing, particularly in the areas of power, scale, and form factor. By utilizing functional materials that do not require significant power, our wearable prototype eliminates the need to be connected to a computer and utilizes a 9V battery. Additionally, our focus on scalability and affordability enables the mass production of devices at low cost, making them accessible to a wide range of users. Our emphasis on form factor ensures that our prototype blends seamlessly into the user's environment, resembling everyday objects and enhancing user acceptance and adoption.

6. RESULTS

6.1 Initial Survey

At the beginning of the project, we released an initial survey to help inform future development. The survey evaluated respondents' familiarity with existing gesture recognition systems (primarily those found in gaming consoles such as the Nintendo Wii), as well as their perception on which gestures ought to map to which actions. They also answered questions on the relationship between gestures corresponding with related actions, as well as how a change in magnitude ought to be represented via gesture.

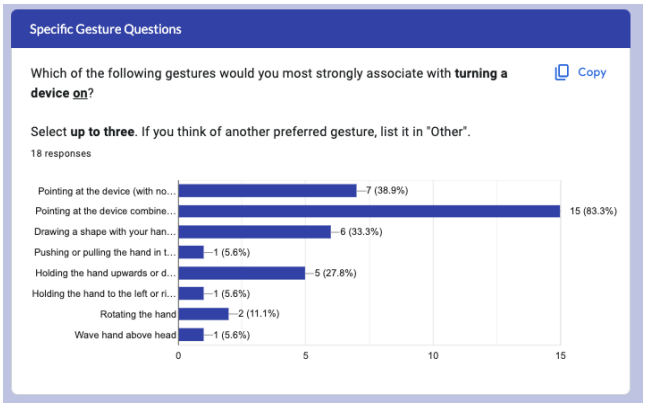


Figure 1: Example Gesture-Action Question Results

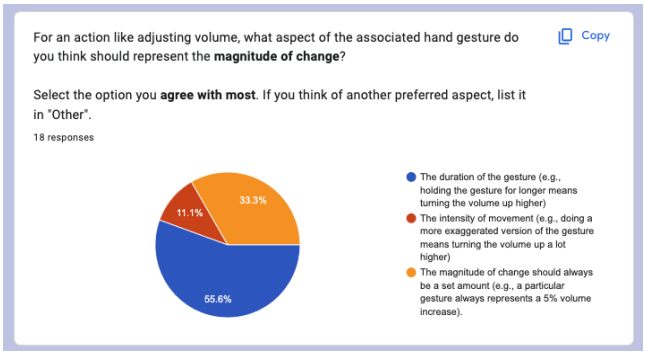


Figure 2: Magnitude Change Question Results

In this survey, we focused on three specific actions:

1. Turning a device on.
2. Turning a device off.
3. Changing a device's volume.

We focused on these actions since actions (1) and (2) represent opposite ends of a polar action, whereas (3) represents a change in magnitude. The majority of our 18 respondents preferred a "point and flick" motion for (1) and (2), where the direction of motion ought to be opposite between the two corresponding gestures. (3) had more mixed results, with an equally large number of respondents preferring pointing and flicking, holding the hand upwards or downwards, or rotating the hand.

6.2 Prototype Testing

Our experiments focused on evaluating the performance of our wearable prototype in recognizing dynamic hand gestures and its impact on user experience. We conducted both quantitative and qualitative analyses to assess the effectiveness of our system.

Throughout our user experience research we recruited a diverse group of participants, including both novice and experienced users, to ensure comprehensive testing. Participants were instructed to perform a series of predefined dynamic hand

gestures aimed at controlling smart devices. We measured the accuracy of gesture recognition and gathered feedback on user experience through the survey.

Our survey questions focused on evaluating usability and comfort:

- *On a scale from 1 to 5, with 1 being the least comfortable and 5 being the most comfortable, how comfortable were you with wearing the prototype?*
- *On a scale from 1 to 5, with 1 being the least comfortable and 5 being the most comfortable, how comfortable were you with making the gestures?*
- *On a scale from 1 to 5, with 1 being the least usable and 5 being the most usable, how usable do you find this prototype?*

Our prototype achieved an average recognition accuracy of 87%, surpassing our target of 85%. This indicates robust performance in interpreting dynamic hand gestures. The sliding window mechanism combined with the gesture detection algorithm effectively captured and analyzed accelerometer data, enabling real-time gesture recognition. Participants rated their experience with the prototype using a 1-5 Likert scale. The average score across all usability metrics was 4.7, indicating a positive user experience.

Participants provided qualitative feedback on their experience with the prototype. Common themes included ease of use, intuitiveness of gestures, and satisfaction with the responsiveness of the system. Our results align with the original motivation of providing a convenient and seamless means of interacting with smart devices. Participants expressed enthusiasm about the potential of gesture-based input systems to enhance everyday tasks.

7. CONCLUSION AND REFLECTION

At the beginning of the project, we aimed to increase convenience in using smart devices. We hoped to create a marketable technology that would eliminate the need for remote controls that would be convenient, accessible, and fun for the average user.

A common theme to our approach of problem solving was, "how would we do this if it were a product coming to market instead of a class project?" From ideation to implementation, we applied concepts and best practices of entrepreneurship and engineering across the industry that allowed us to stay flexible and deliver a solution with user needs in mind. Primary among these was the use of iterative practices, both in product design and software development, to create a continuously more refined project.

With a cornerstone of our project being the gesture recognition model, we invested significant resources into developing a robust pipeline that carries sensor data to models for training. Not only did this tool pay dividends when we needed to retrain a new model to more readily fit our needs, we believe that it could be used to expand our current model with additional sensor data or train other models to perform other sorts of analysis.

The interface design for our project was an opportunity to step outside our comfort zones and do new things. The input system was a huge departure from the standard point-and-click or touch interfaces that are ubiquitous in modern computer systems and leaned heavily on human factors and HCI principles to ensure that our solution is one that meets user needs.

We have learned that the current state of common smart home APIs is not currently, in our estimation, developer-friendly. While we were able to pivot away from this as a project requirement, it is worth noting that any project that plans to interface with one of these APIs will require significantly more development time, developers that have preexisting knowledge of the API integration, or time to allow these APIs to experience significant maturation in the area of usability.

Finally, the least tangible point of reflection relates to the skills we have gained in many phases of the project. Our group has developed through the phases of hardware and software development and integration. With no skills in working with an Arduino, embedded software development, soldering connections, or integrating AI into a tool, we successfully completed a project that utilizes all of these features.

8. REFERENCES

[1] Baudel, T. and Beaudouin-Lafon, M. 1993. Charade: remote control of objects using free-hand gestures. *Commun. ACM*. 36, 7 (July 1993), 28–35. DOI=https://doi.org/10.1145/159544.159562.

[2] Gong, S., Ma, H., Wan, Y., and Anran Xu, A. 2019. Machine Learning in Human-computer Nonverbal Communication. In *NeuroManagement and Intelligent Computing Method on Multimodal Interaction (ICMI '19)*. Association for Computing Machinery, New York, NY, USA, Article 4, 1–7. DOI=https://doi.org/10.1145/3357160.3357670.

[3] Koh, J, Cherian, J, Taele, P, and Hammond, T. 2019. Developing a Hand Gesture Recognition System for Mapping Symbolic Hand Gestures to Analogous Emojis in Computer-Mediated Communication. *ACM Trans. Interact. Intell. Syst*. 9, 1, Article 6 (March 2019), 35 pages. DOI=https://doi.org/10.1145/3297277.

[4] Li, X., Chen, Y., and Tang, X. 2023. GesMessages: Using Mid-air Gestures to Manage Notifications. In *Proceedings of the 2023 ACM Symposium on Spatial User Interaction (SUI '23)*. Association for Computing Machinery, New York, NY, USA, Article 48, 1–2. DOI=https://doi.org/10.1145/3607822.3618023.

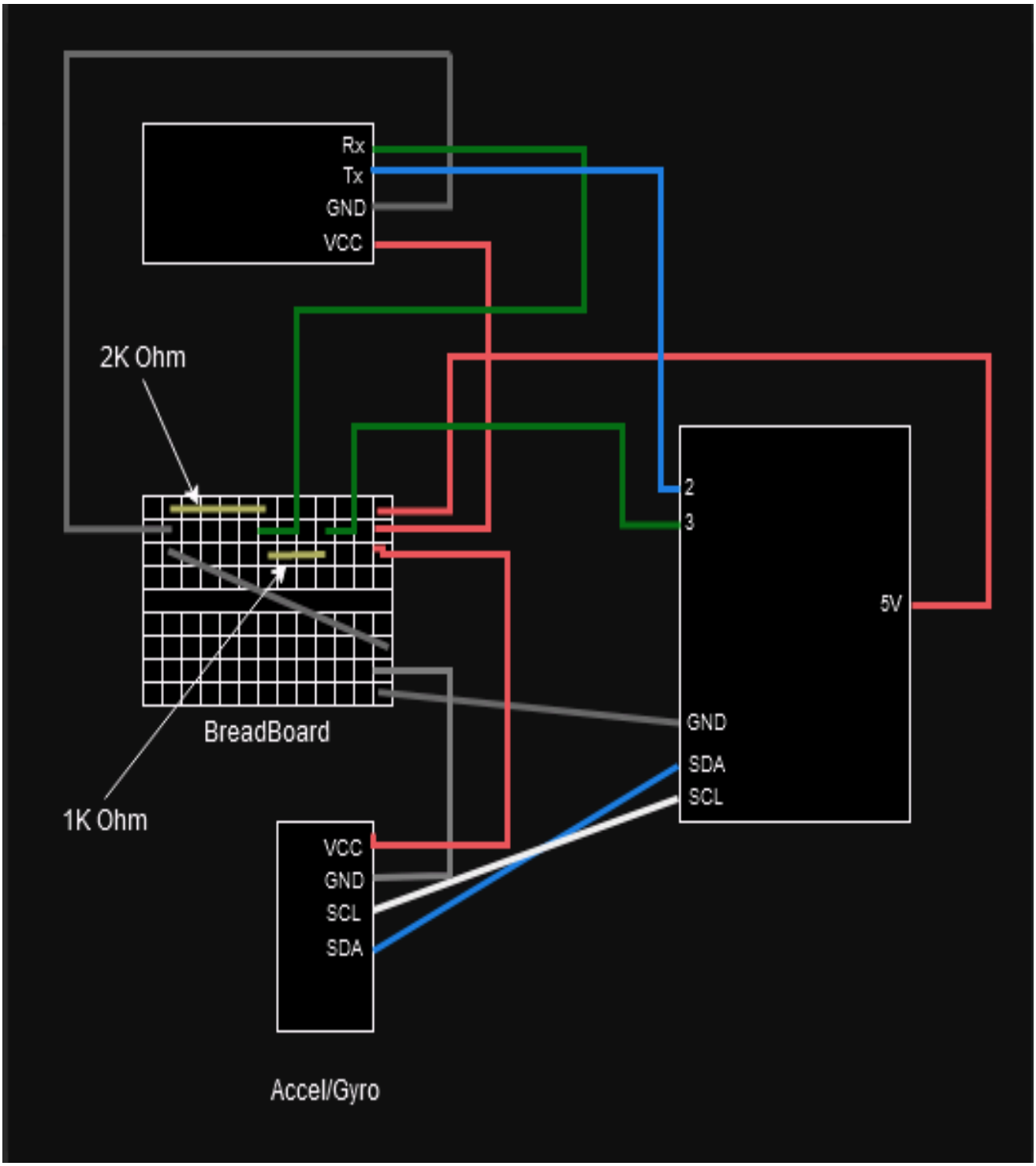
[5] Weiser, M. 1999. The computer for the 21st century. *ACM SIGMOBILE Mob. Comput. Commun. Rev.* 3, 3 (July 1999), 3–11. DOI=https://doi.org/10.1145/329124.329126.

[6] Wheatley, M. J. 2002. It's An Interconnected World: *Shambhala Sun* DOI=https://margaretwheatley.com/articles/interconnected.html.

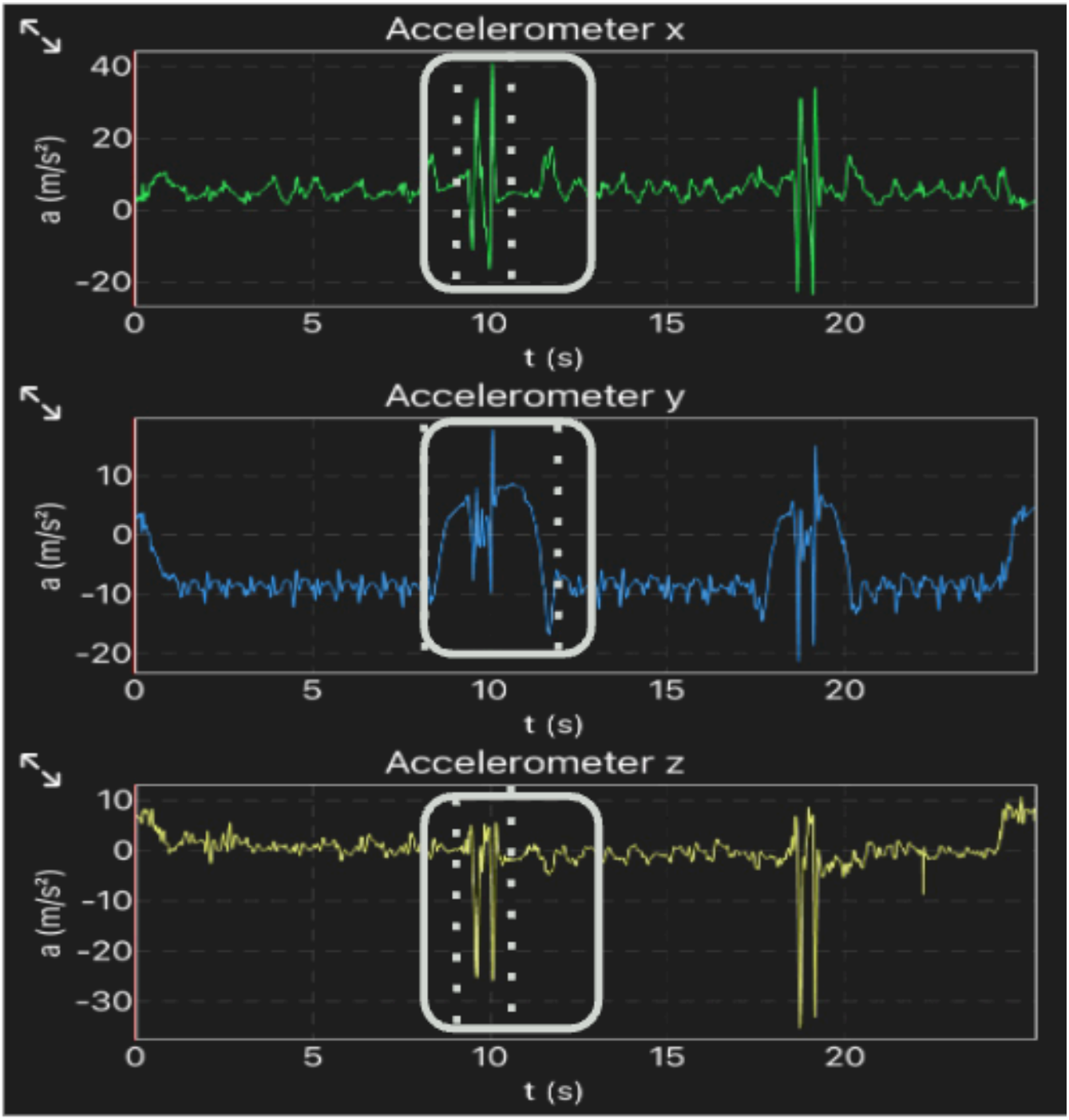
[7] Zimmerman, T. G., Lanier, J., Blanchard, C., Bryson, S., and Harvill, Y.. 1986. A hand gesture interface device. *SIGCHI Bull*. 17, SI (May 1987), 189–192. DOI=https://doi.org/10.1145/30851.2

9. APPENDIX

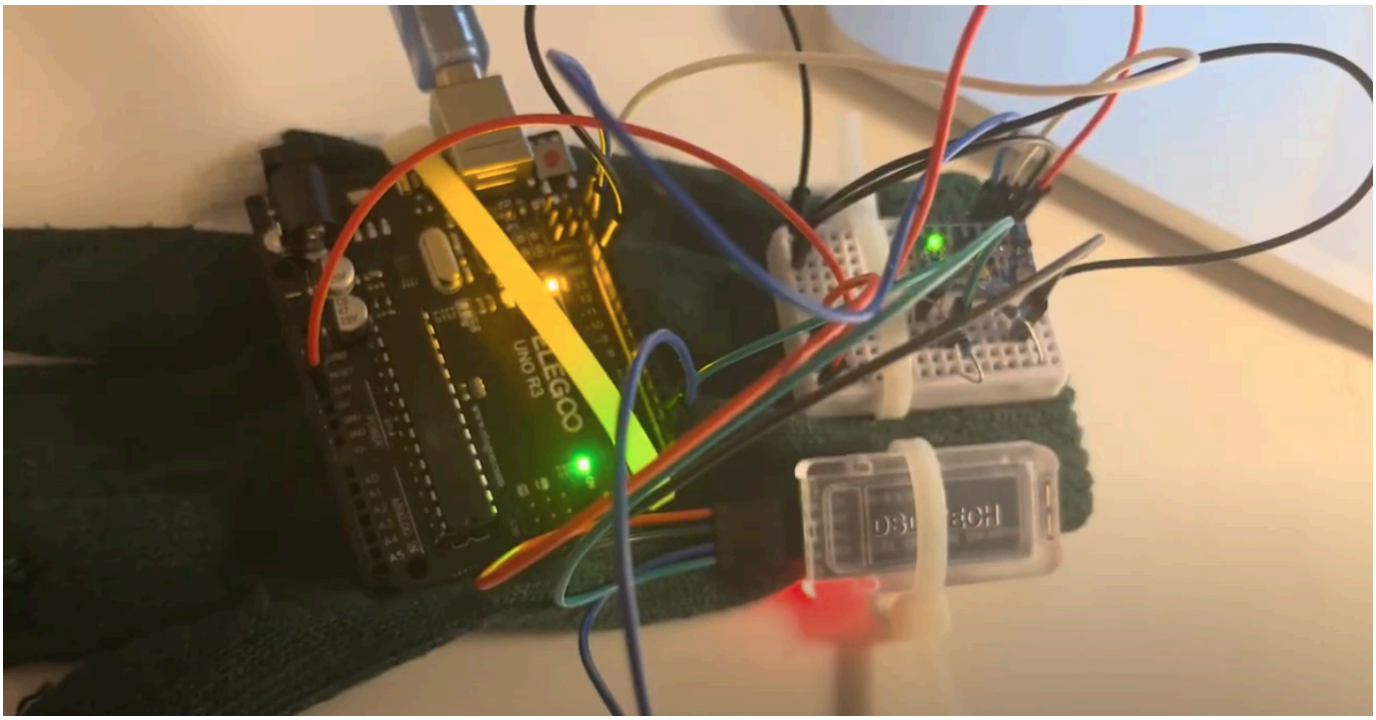
9.1: Prototype Diagram



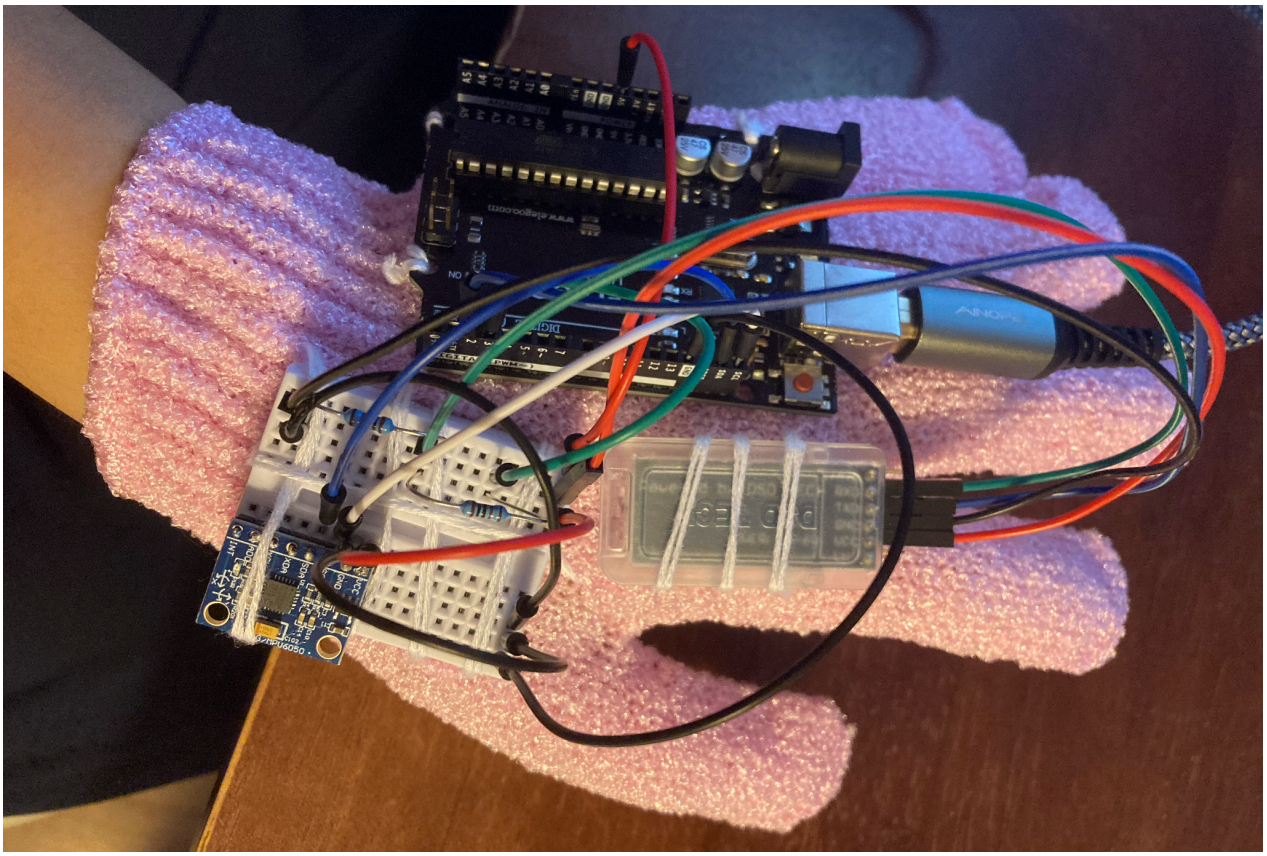
9.2: Sliding Window

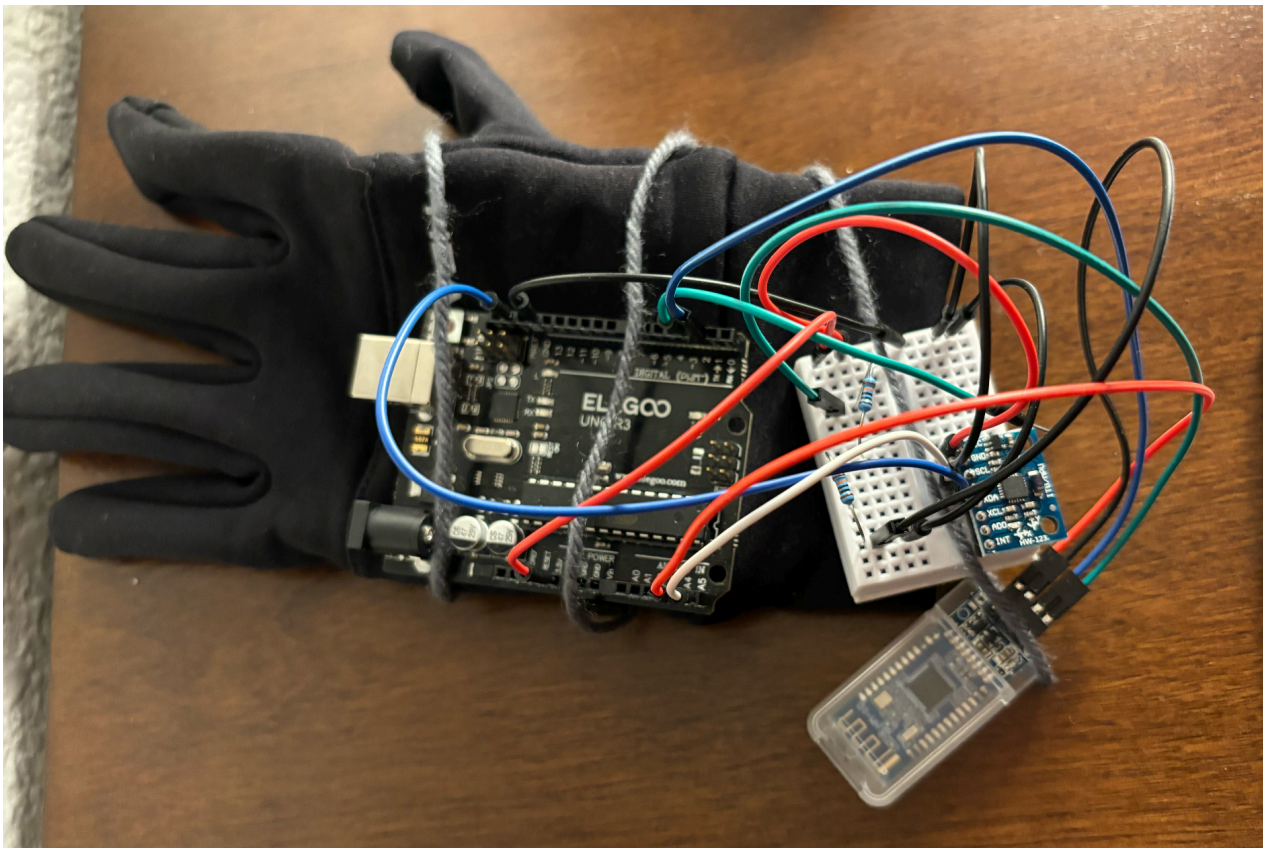

Accelerometer x
40
20
0
-20
a (m/s²)
0
5
10
15
20
t (s)
Accelerometer y
10
0
-10
-20
a (m/s²)
0
5
10
15
20
t (s)
Accelerometer z
10
0
-10
-20
-30
a (m/s²)
0
5
10
15
20
t (s)

9.3: Wearable Glove



9.3: Wearable Glove (cont.)





9.4: Interface Mock-Up