TeleTouch

David Hwang, Xuanye Li, Gram Liu

Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—Interacting with objects in 3D space requires users to be physically adjacent to their computers during live presentations. TeleTouch uses gesture recognition to wirelessly interact with 3D entities using intuitive gestures that easily integrate with live presentations. The system will leverage a glove with flex sensors on each finger and an inertial measurement unit (IMU) to collect data across 11 degrees of freedom. This data is transmitted to a peripheral attached to the user's computer which will perform gesture recognition and dispatch the appropriate input macros to the system.**

***Index Terms*—Gestures, glove, haptic, interactive, kinesthetic, sensor, decision tree**

I. Introduction

Live presentations of 3D entities and schematics is difficult and requires the presenter to be physically adjacent to their computer [1]–[3]. The current state of tools for live presentation consists of handheld devices such as laser pointers or remotes with buttons to move forward or backwards between slides. This does not, however, scale well into manipulating entities in 3D space. Interacting with 3D entities in popular applications requires enabling a specific interaction mode (zoom, pan, etc.) in conjunction with continuous mouse input (dragging) to control the degree and intensity of the performed action. The challenge with wirelessly performing these actions is due to the continuous input which would require a dial or some other similar mechanism.

Attempts have been made to repurpose laser pointers into behaving like a mouse [4], [5]. However, these attempts either resulted in low accuracy, or required users to memorize complex gestures, which makes the overall system less accessible and intuitive to use. There also exist VR glove controllers in the market [6]–[8]. While these gloves could be repurposed for normal presentation, they were designed for a much higher level of precision than is required for presentations. The resulting gloves are too expensive to be accessible to most educators [6]–[8]. This is problematic as educators comprise a significant demographic if not a majority of the presenters that interact with 3D entities. TeleTouch aims to become the ideal solution for educators by combining usability with accessibility.

II. Use-Case Requirements

In order to meet its goal of making 3D presentation seamless, intuitive, and accessible, TeleTouch must achieve specific use case requirements.

TeleTouch should enable a user to remotely interact with 3D entities on their computer. It should allow users to interact with 3D entities using hand gestures without being physically adjacent to their computer. Moreover, it should also be able to distinguish between when a user is making gestures with the intent of interacting with 3D entities or if they are simply making normal hand gestures that accompany their presentation. Since the system is meant to be used in conjunction with a live presentation, it should be easy to enable and disable the recognition of gestures so that the presenter's normal hand gestures won't interfere and accidentally interact with 3D objects. When gesture recognition is disabled, no input should be dispatched to the connected computer from the system.

In order to interact with 3D objects, TeleTouch should be able to support gestures pertaining to basic camera manipulations: zoom, rotation, and panning. It should be able to identify these gestures correctly and correctly dispatch the appropriate input macros into the connected computer. The experience should also feel seamless to users and so the intended effect on their computer should be observed in a short enough amount of time that their train of thought is not disturbed.

TeleTouch should be lightweight and portable. Since it is designed for users to present untethered to their computers, it should at least allow them to wirelessly operate the system from the center of their stage without a physical connection to their computer. The wearable component of the system should be light enough that a user will not feel burdened by the weight. Since the glove is designed to help educators, its battery life should also be long enough to last at least the length of one entire lecture.

TeleTouch should be usable and accessible. Since the goal of the system is to create a mechanism for interacting with and controlling 3D objects more intuitively than a handheld device, the gestures that the system will support should closely resemble existing conventions for those actions so that even after just one presentation session, a user will be able to remember most if not all the gestures. Furthermore, the system should be easy to set up and configure. An inexperienced user should be able to set up and calibrate a new system in 10 minutes while a pre-calibrated system should take 5 minutes. Lastly, in order to be accessible to educators, the system should be economical. It should cost less than $1000, the price of major commercial alternatives [6]–[8].

III. Architecture and/or Principle of Operation

The system is composed of two main modules: (1) the glove module and (2) the compute module. The interactions between

and within these two modules are indicated on the block diagram in Fig. 2. The glove module serves as the point of data collection, leveraging inputs from five flex sensors (one for each finger on one hand) and an inertial measurement unit (IMU). From these sensors, the microcontroller (Arduino Nano 33 IoT) on the glove module bundles the data into a data packet to be transmitted to the compute module. The compute module consisting of a Raspberry Pi 4 acts as the data processing hub of the system, receiving the raw input data and performing gesture recognition on this data. Additionally, the compute module also handles communicating the gesture to the user's personal computer (host PC) as an appropriate command or macro.

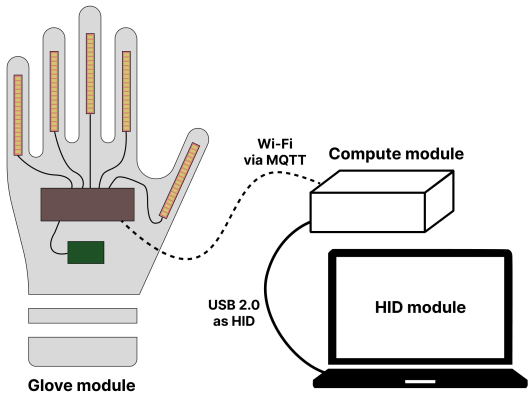

Figure 1 System overview

A. Glove Module

For ease of setup and use, the sensors are all mounted on wearable gloves. In order to capture the state and movement of the user's hand, the system shall consist of flex sensors for each finger to determine the amount of flex in each finger, as well as an on-board IMU part of the microcontroller to determine the rotation and acceleration of the hand over time. Altogether, this produces 11 distinct degrees of freedom.

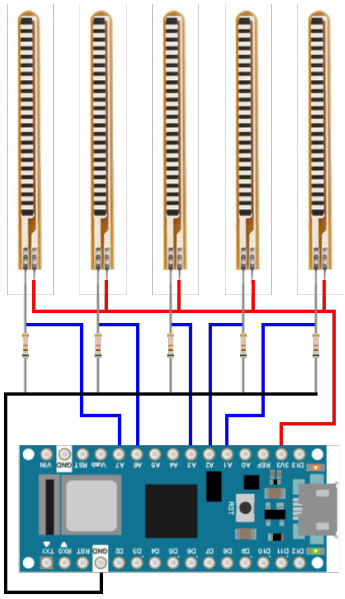

Figure 2 Circuit diagram for the Glove Module

The module is powered through and controlled by an Arduino Nano 33 IoT microcontroller. This board synchronizes the sensor inputs and publishes the data representing the state across the 11 degrees of freedom via Wi-Fi with the MQTT protocol, a publisher/subscriber messaging protocol used in many Internet of Things (IoT) devices. The main motivation behind transmitting the data wirelessly is to maximize portability. Moreover, Wi-Fi is a widespread standard that is performant, has high range, and is easy to set up.

B. Compute Module

The compute module houses the software for both gesture recognition and the human interface device (HID) driver. As the data from the glove module is being published to the MQTT topic, the compute module subscribes to the topic to receive the data. This data is then input into the gesture recognition program which classifies the data into one of six gestures and calculates the confidence level of the output gesture. Depending on the confidence level of the output gesture, the program can determine whether an actual gesture was performed or if the data was mostly noise.

Once a gesture is definitively identified, it is passed along to the HID controller program which communicates to the host PC via a USB connection. In this context, the compute module is known as the peripheral device. For each gesture, the compute module sends an appropriate command or macro to the host PC conforming to the USB HID specification [9]. The command or macro, in turn, controls the 3D visualization application running on the host PC.

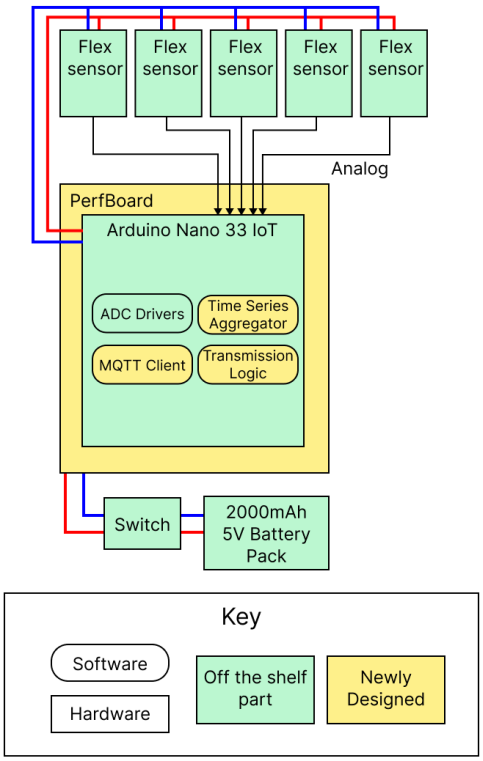

Figure 3 Glove module subsystem block diagram

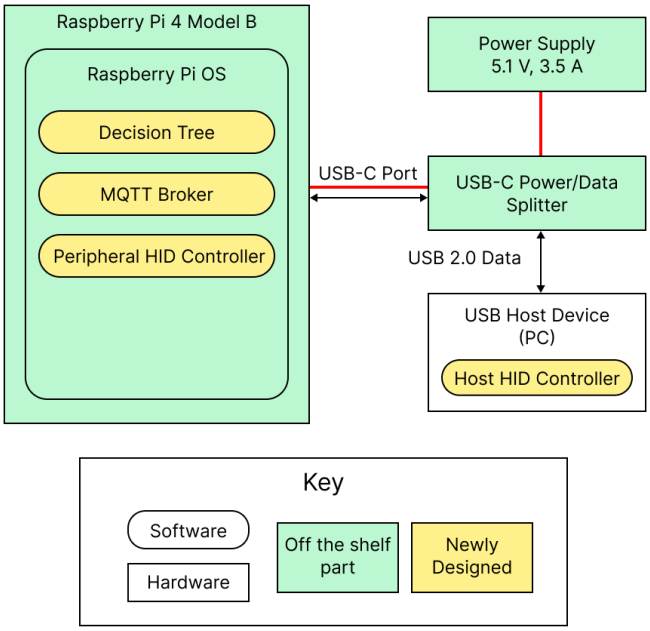

Figure 4 Compute module subsystem block diagram

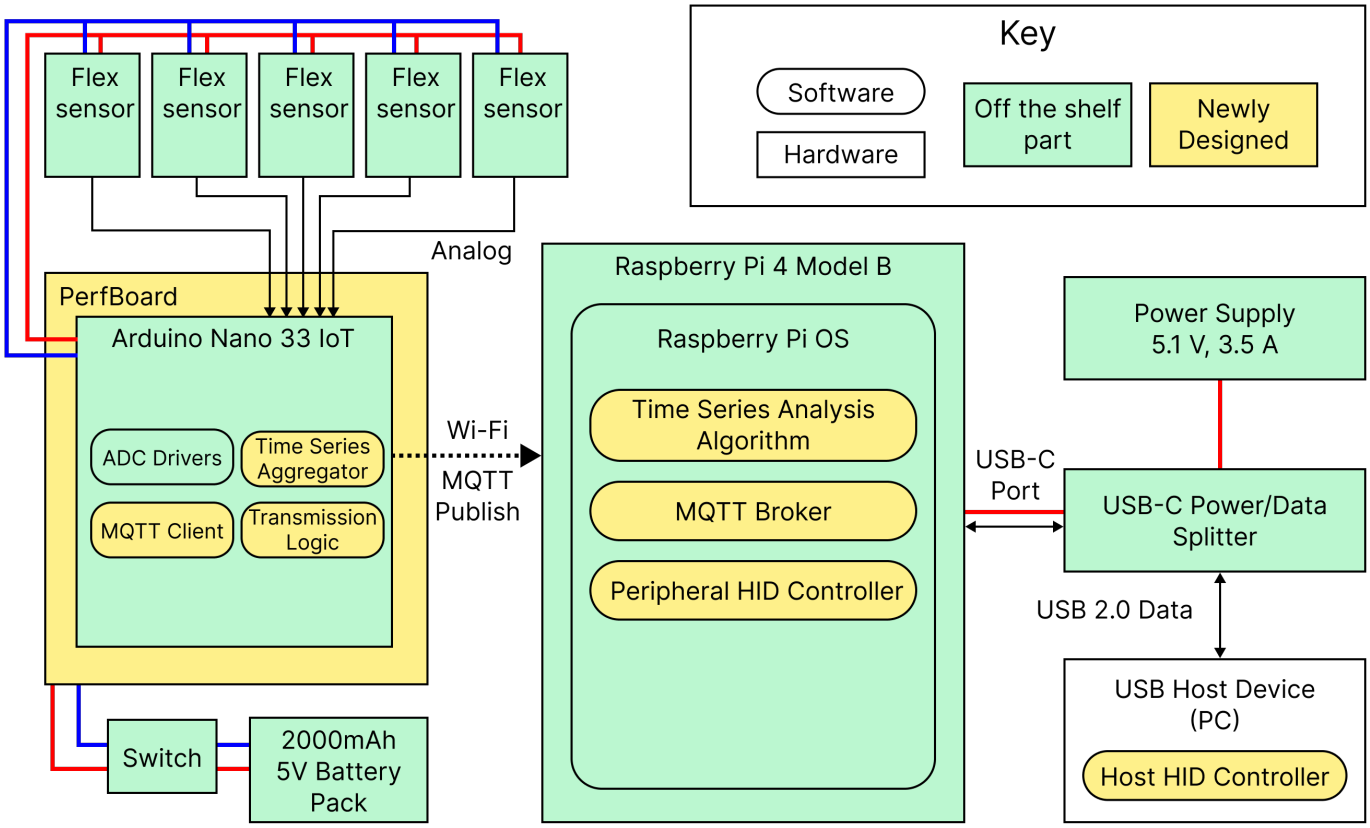

Figure 5 Entire system block diagram

IV. Design Requirements

Based on the Use-Case requirements, TeleTouch should fulfill six key design requirements, which are functionality, accuracy, latency, portability, range, and battery life. Functionality, accuracy and latency correspond to the user experience relating to the software aspects of gesture detection, while the other design requirements correspond to hardware targets to ensure accessibility. All of this combine to form a seamless, intuitive, and accessible experience for the target user.

A. Functionality

TeleTouch should be able to support six different gestures: enable gesture recognition, disable recognition, zoom in, zoom out, panning, and rotation. These six gestures encapsulate the scope of the desired functionality, allow the product to recognize when the user is making gestures, and distinguish between intended and natural gestures.

B. Accuracy

TeleTouch should have a gesture recognition accuracy of at least 90% accuracy. This not only applies to the gesture classification itself, but also the recognition of the magnitude performed by a gesture. The ideal accuracy should be 100% for perfect smoothness, however, current state-of-the-art gesture recognition models are around 90% accuracy [10], [11], hence the target goal was set accordingly.

C. Latency

TeleTouch should recognize user gestures and translate it into a macro on the user's computer within 1 second. This latency requirement encapsulates the entire process from sensor reading detection, to network packet transmission, to model recognition, and finally to a macro output on the user's computer. A latency less than one second would not interrupt the user's train of thought even if the delay was slightly noticeable [12]. Hence, the target latency metric aims to maximize the available slack time while still maintaining a relatively seamless experience from the user's perspective.

D. Portability

TeleTouch should not weigh more than 300 grams. A standard winter glove weighs around 100 g per hand, hence the target for the product is to weigh no more than thrice this amount. Any weight above this 300-gram threshold will begin to place a burden on the user's hand while using the product.

E. Range

TeleTouch should support up to 50 m away from the user's computer. Since the main use case of the product is aimed at presentation, the user should not feel hindered while naturally moving around the room. An average lecture hall is no more than 50 m in length, and, Rangos Hall has dimensions 108' X 56', or 33 m X 17 m. Hence, the metrics enables the user to perform gestures while still retaining the ability for the user to move around, which is relevant in a classroom context.

F. Battery Life

TeleTouch should be able handle up to 1 hour of continuous use and active gesture detection. A typical academic lecture would be around 1 hour, and this requirement guarantees that the product will be usable throughout the duration of an entire presentation.

V. Design Trade Studies

A. Data Collection Subsystem

1. Computer Vision vs. Sensors

Gesture detection can be implemented in two main ways: either with computer vision, or with sensor data [13]. With computer vision, the system would involve a mounted camera pointed at the user, with a computer vision model analyzing the hand positions and gestures being performed. Although the portability requirement would be easily satisfied, the trade-offs in usability are very significant.

Firstly, a computer vision approach would necessitate the user to always face the camera throughout the presentation when performing the gestures. The user would also be limited in their mobility while presenting, as they would need to also remain within the range of the camera view. Moreover, the setup requires a high-definition camera that has enough granularity to infer from the user's hand, which would be more costly compared to a sensor-based system. It also requires the user to mount a camera in the presenting room which is difficult in the absence of a dedicated camera mount.

2. Hall Effect Sensor vs. Flex Sensor

For a sensor-based system, another alternative design would involve having the user wear sensor rings instead of an entire glove. Each ring would utilize a Hall effect sensor to measure how much each finger is bent. This design seeks to minimize intrusiveness from the user's standpoint. However, unlike using full-length flex sensors, these rings would not easily be able to detect as many degrees of freedom on hand gestures as a glove system. Additionally, these sensors require a magnetic component which needs to be mounted elsewhere to be effective, which may not be significantly better than just wearing a glove in terms of being intrusive.

B. Raspberry Pi Compute Module

A single board computer was chosen to power the compute module instead of having the user install additional software to listen for input because it would allow the system to come packed with all the dependencies installed instead of pushing the complexity of setting up the system to the user. For the choice of single board computer, two options were considered: NVIDIA Jetson and Raspberry Pi. The Jetson was originally considered due to its higher processing capabilities. However, after reasoning about the nature of the data collected and the gesture recognition model that would be used for classification, it became evident that a Jetson would not be necessary for the amount of computation performed. Furthermore, although a Jetson had better processing capabilities, it was lacking in many other areas such as the absence of built-in Wi-Fi capabilities, no support for behaving as an HID, and high cost which challenged the goals of accessibility and economy. A Raspberry Pi, however, meets all the technical requirements needed while being significantly cheaper. Moreover, despite its lower processing capabilities for machine learning models, there exist TPU peripherals that can accelerate computation for these specific tasks.

C. Gesture-driven Activation

While identifying gestures that were both familiar and easy to perform, it became evident that basic features available to touch screen interfaces would become difficult on a system like TeleTouch that continually measured input from its users. On touch screen interfaces, it is trivial for the system to differentiate between when a user is making gestures as input and when the user is making gestures for other reasons. When a user does not intend to control the system with their gestures, they simply need to avoid making contact with the touch interface such as a trackpad or touch screen. On TeleTouch, however, the system needs to make that distinction itself.

Two states were identified pertaining to whether the system is actively recognizing performed gestures. When the system is in the disabled state, it will ignore any gestures performed by the user until it changes into the activated state. The first attempt at solving this problem was the introduction of a switch on the glove to switch between the enabled and disabled states. However, it was quickly realized that requiring the user to change the switch whenever they needed to change between the states is unwieldy and obtrusive to the presentation experience. Thus, it became clear that a better alternative would be to instead use specific gestures for enabling and disabling recognition. Having specific gestures for this would allow presenters to toggle states without having to use their other hand, which increases accessibility for its users and also makes the presentation experience much more seamless.

D. Separate Zoom Gestures

The system has two distinct gestures for zooming in and out. The motivation behind this is that on most touch screen interfaces, the gestures for zooming in and out are orthogonal – opposites of each other. The challenge with implementing this on TeleTouch, however, is that when the system is continuously measuring input from the user, recovering to a neutral position from performing gesture typically entails performing the orthogonal gesture. For example, when a user performs a pinching gesture to signify zooming in, if they recover back to the starting position, to perform another zoom, their hands would be performing the opposite action and would be interpreted by the system as a zoom out action. This would mean that performing a zoom in gesture would immediately be followed by a zoom out when the user recovers their hand to a neutral position. To circumvent this issue, separate gestures for zooming in and zooming out were used to ensure that recovery to a neutral state was through a distinct gesture path.

E. Time Series Analysis

Time series analysis was chosen to be the basis of the gesture recognition model. Other alternatives were considered, including a ConvLSTM-based network and a simple recurrent neural network (RNN).

ConvLSTM networks are an extension of fully-connected LSTM networks with convolutional structures for better spatiotemporal sequence forecasting [14]. Initially, the nature of the data appeared to fit well with the type of problem that ConvLSTMs were designed to solve. However, with further research, it became clear that this was not the case. While convolution could be performed across the acceleration values (each axis) and across the rotation values (each axis), they needed to be performed independently. Moreover, the flex sensors were completely independent of the other measurements. Thus, the data points in each packet of data were independent enough that it did not make sense to add a convolution step.

The next idea considered was to simplify the network into using a simple RNN with 11 units, for each degree of freedom afforded by the different sensors. Due to the LSTM component of the RNN, this network would still be able to incorporate the temporal aspect of the data into the classification process as opposed to a plain fully connected network. This may be sufficient to address the problem of classification, however, due to the strict latency requirements of the system, involving a neural network classification step may pose too much latency overhead.

The initial ideas considered were primarily neural networks, because neural networks offered a greater degree of flexibility and extensibility in terms of the number of gestures that could be supported. Due to the nonlinearity of the problem space, neural networks would be able to learn and identify patterns that would otherwise be nontrivial to solve. However, after identifying the gestures that the product would support, it became clear that a neural network would not even be necessary. With the high degrees of freedom afforded by the sensors relative to the number of supported gestures, it became much more possible to classify gestures based purely on time series analysis of the data. Specifically, each gesture will be mapped to an expected trend over a specific window. For example, a gesture that involves closing a hand will correspond to a decreasing trend in the values of the flex sensors. Similarly moving a hand side-to-side will result in changing accelerometer readings along the x-axis. Furthermore, by simplifying and replacing the gesture recognition model with a decision tree based on time series trends, it will be much easier to reason about bugs in the model and calibrate accordingly as opposed to a deep neural network.

VI. System Implementation

A. Glove

The glove subsystem consists of five Spectra Symbol flex sensors and an Arduino Nano 33 IoT microcontroller with its on-board IMU, all connected using a perfboard attached to the glove. These components are powered by a 5V 2,000 mAh portable battery pack.

1. Flex Sensor

For the solution to satisfy the design requirements, the flex sensors would only need to measure a noticeable increase in resistance with an increase in the degree of bending from a finger. The gesture detection algorithm would be able to optimize its thresholds as long as a reasonable change in resistance from a neutral to flexed position can be measured. Thus, the main target for selecting the flex sensor model would be that of cost effectiveness.

The most widely available model on the market is the Spectra Symbol Flex Sensor, and it cost $7.16 per unit at the time of purchase. With each on one finger, the total cost for five of these sensors would be $35.8, which is only a small fraction of the total budget. Furthermore, these specific sensors offer a

reasonable range of bend resistance, starting from 45K up to 125K Ohms.

Some alternatives include the FlexPoint Bend sensor and BendLabs One Axis sensor. However, both alternatives have significant drawbacks. The FlexPoint sensor started at $10 per unit, and only ships after 35 days. Moreover, the resistance range was unclear from the datasheet. The BendLabs sensor started at $52.50 per unit; if the BendLabs sensor was selected, five sensors alone would cost $250, which is almost half the budget, and it is also not considering spare parts for risk mitigation. No amount of additional benefits, if any, would justify the cost of this alternative.

2. IMU

The Inertial-Measurement unit needs to support at least six-axis degrees of freedom of measurement and I2C. The on-board LSM6DS3 IMU of the Arduino Nano 33 IoT was used as this eliminates the need of procuring a separate board, reducing the overall form factor of the system. This IMU supports 3-axis accelerometer and gyroscope data, which enables the system to collect enough data to effectively classify gestures.

An alternative would be the Adafruit MPU-6050 at $12.95, which is a reasonable price. While reasonably priced, having a separate IMU is unnecessary due to its availability on the microcontroller.

The LSM6DS3 IMU outputs three dimensions of accelerometer and gyroscope data each in terms of *g*, the standard gravitational acceleration on the surface of the Earth. Unconfigured, the acceleration values include the gravitational acceleration such that when the controller is at rest, the magnitude of the acceleration vector is 1 *g*. In order to account for gravity, the accelerometer values must be calibrated with an appropriate method. For the use case of this project, an orientation filter like the Madgwick filter is sufficient. The Madgwick filter uses gradient descent to optimize the quaternion orientation of the IMU based on its accelerometer and gyroscope measurements. Once the orientation of the IMU is determined from the filter, the gravity components can be subtracted from the acceleration measurements. The resulting measurements ensure that at rest, the magnitude of the accelerometer vector is approximately 0.

3. Flex Sensor Attachment Mechanism

The flex sensors were originally going to be attached to the glove using cloth bands sewed into the bands. However, after testing this with the system, it was discovered that the cloth bands did not result in a sufficient amount of tension for the flex sensors to detect. This made it difficult to perform accurate gesture classification later in the pipeline. To resolve this, stronger adhesives were used instead. Tape was used to fix the flex sensors between each finger joint, and hot glue was used to fix them at the base of each finger. Tape ensured that a sufficient amount of tension could be detected by the flex sensors. Moreover, using hot glue to fix the base of each sensor was necessary to minimize stress from cross-sectional bending that gets transferred to the sensors. By fixing the base, it ensures that the flex sensors are always in a straight position and the cross-sectional stress instead gets transferred to the connecting wires.

4. Glove Wiring

The connection between the flex sensors and the protoboard was a failure point that was not realized until the soldering process began. Initially, solid wires were used, however, their rigidity placed a high amount of strain on the fragile flex sensors, which resulted in higher noise in the readings. To circumvent this, they were replaced with flexible wires and using hot glue to fix the base of the sensors to ensure that the sensors themselves always maintained a straight orientation during operation. While this addressed the issue of strain on the sensors, the subsequent strain on the wires resulted in frequent snapping at the solder joint when putting on the glove. These two issues were ultimately resolved by replacing the wires with jumper cables, which provided rigidity at the base without compromising flexibility.

5. Microcontroller

The main requirement for the microcontroller would be to have at least five analog pins for the flex sensors, and I2C capabilities for the IMU. The Arduino Nano 33 IoT is a newer model that supports both requirements, while coming at a reasonable cost and having a small form factor that is enough to fit the perfboard that was ordered. Moreover, it comes with an on-board IMU which helps reduce the overall form factor by eliminating the need for an additional IMU. The specific model has support for 8 ADC input pins and an I2C peripheral, while having 2.4 GHz Wi-Fi capabilities. In terms of performance, it operates a SAMD21 Cortex-M0+ 32-bit low power ARM MCU, with 256KB of Flash memory and 32KB SRAM. The alternative that was originally considered was the ESP32-C3. While similar in size and capabilities, it lacks an on-board IMU which significantly increases the form factor overhead of the system.

B. Compute Module

1. *Gesture Classification*

From a resting position with all fingers outstretched, analog readings from the flex sensors start at 0. As the fingers are increasingly flexed, the flex sensor readings will increase towards a maximum analog reading of 1023. For motions such as a hand with all fingers outstretched progressing towards a closed fist, a positive trend on the flex sensors can be expected as the readings increase from a starting state close to 0. Conversely, the progression from a closed fist to outstretched fingers would result in a negative trend in the flex sensor data.

Gesture recognition, therefore, will be performed through a decision tree based on a windowed analysis of overall trends of independent sensor readings. Specifically, a rolling window over 1 second of data will be analyzed. The trends over each window will then be determined using linear regression. The expected trends for each gesture are described in Table I.

The enable and disable gestures change the state of the system from either enabling or disabling the recognition of gestures. When the system is in the disabled state, all gestures are ignored except for the enable gesture. This is to allow the user to perform other gestures without worrying about the system accidentally interpreting it as an interaction with their computer. Thus, the enable and disable gestures were

specifically chosen to involve flex, acceleration, and rotation readings to minimize the probability of a user accidentally activating or deactivating recognition.

When actions are being performed, if the measurements are insignificant or do not meet the threshold to be considered for any of the enumerated gestures, then no action is performed on the connected computer. If a gesture is identified by the system as being performed, the same window of data is used to compute the intensity or degree of the action (excluding enable and disable gestures). For example, in the case that a pan gesture is being performed, the x and y acceleration values are used to determine the direction of the pan being performed, as well as the amount of panning to recreate on the system.

2. *Wireless network*

The Raspberry Pi configures and creates a wireless access point (AP) for the glove module to connect to. The initial system design called for both the glove module and the Raspberry Pi to be configured to connect to the same WiFi network. While this would have allowed for much greater range since it externalized the hardware limitations of the WiFi network, it introduced additional setup complexity, since it would be difficult to upload new WiFi credentials into the Arduino.

To address this, we designed the system so that the compute module Raspberry Pi created a wireless AP which the glove was configured to automatically connect to. While decreasing the range, it resulted in a much better setup experience. Moreover, since the decreased range was still more than enough to use the device in a classroom setting, it appeared to be a reasonable tradeoff.

C. HID

The Human Interface Device (HID) device class specification is an extension of the Universal Serial Bus (USB) specification intended to provide support for a wide range of USB-compatible devices. Prior to the introduction of HID, manufacturers of non-mouse or keyboard devices needed to create their own device-specific drivers [15]. With this specification, manufacturers of USB devices can define how and what their devices intend to communicate with the host computer in a data-efficient, extensible, and robust manner [9]. Although the glove itself is not providing input to the computer directly, the compute module dispatches commands to control the host computer as an HID would. The USB HID specification considers such devices to also be HID-compatible [9].

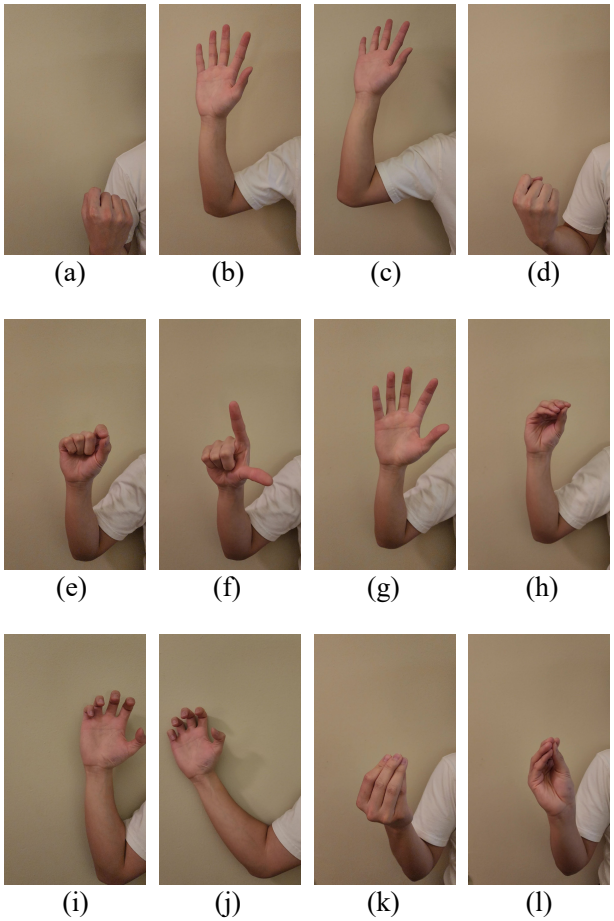

Figure 5 Gestures. (a) & (b): Enable recognition, (c) & (d): Disable recognition, (e) & (f): Zoom in, (g) & (h): Zoom out, (i) & (j): Panning, (k) & (l): Rotate

TABLE I. TABLE OF GESTURES WITH EXPECTED SENSOR TRENDS

Gesture	**Sensor Trends**						
	Flex	***Acceleration (x)***	***Acceleration (y)***	***Acceleration (z)***	***Rotation (x)***	***Rotation (y)***	***Rotation (z)***
Enable	–	0	+	0	+	0	0
Disable	+	0	+	0	+	0	0
Zoom in	– (thumb, index); 0 (others)	0	0	0	0	0	0
Zoom out	+	0	0	0	0	0	0
Pan	0 (constant-mid)	+/–	+/–	0	0	0	0
Rotate	0 (constant-mid)	0	0	0	+/–	+/–	0

TABLE II. TABLE OF HID REPORT USAGES AND VALUE SPECIFICATIONS

Data	HID Usage	Usage Type	Min Value	Max Value
Pan X Axis	X	Dynamic Value	-127	127
Pan Y Axis	Y	Dynamic Value	-127	127
"Zoom" Axis	Z	Dynamic Value	-127	127
Rotate X Axis	Rx	Dynamic Value	-127	127
Rotate Y Axis	Ry	Dynamic Value	-127	127

TABLE III. BYTE-WISE VISUALIZATION OF HID REPORT

Report Byte	Byte Offset							
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	Pan X Axis							
Byte 1	Pan Y Axis							
Byte 2	Zoom Axis							
Byte 3	Rotate X Axis							
Byte 4	Rotate Y Axis							

The controls dispatched from the TeleTouch system are more complex than simple mouse events and are related to the manipulation of the position and rotation of an object in 3D space. This kind of device is approximated by a gamepad, which is typically used for USB video game controllers. Gamepads typically have multiple multi-axis controllers which can be used to map panning, rotating, and zooming. The X and Y usages as defined under the Generic Desktop Usage Page on the HID usage specification are mapped to X and Y axis pan on the glove. For rotation, the corresponding Rx and Ry (rotation axes) usages are mapped to X and Y rotation on the glove. As the system only intends to support movement across 2 axes at a time, the Z axis usage is freed up, allowing it to be mapped to zoom control [16]. Tables II and III and Fig. 6 describe the HID report description as well as data structure for serial communication across USB.

As a gamepad input, the compute module will not be able to directly control the mouse and keyboard of the host computer without a dedicated application. The HID control system will therefore consist of two parts: the peripheral and the host HID controllers.

1) Peripheral HID Controller

Based on the Linux Kernel documentation, the USB protocol is handled by the Linux kernel through the file descriptor abstraction [17], [18]. After configuring the USB device description through a system called configfs, reports from the peripheral device to the host device are accomplished by writing directly to the corresponding HID file descriptor (e.g., /dev/hidg0). For ease of development and integration with the gesture recognition software, the peripheral HID controller will be written in Python.

2) Host HID Controller

On the host computer's end, a program parsing the gamepad input and translating it to appropriate mouse and keyboard interactions is required. A system-level API for interfacing with HID input sources called HIDAPI is available for all major operating systems [19]. By using the Python bindings for this API, the host HID controller program can read the controller inputs and determine how to emulate each of the X, Y, Z, Rx, and Ry controls as mouse and keyboard controls. An advantage of performing this translation on the host computer rather than on the compute module is that each control mapping can be program specific and configurable with presets. Moreover, since the mouse cursor is controlled on the computer itself, the program is also able to ensure that the mouse cursor is in the appropriate interaction area without the need for additional communication of display and cursor states to the compute module.

```
USAGE_PAGE (Generic Desktop)
USAGE (Gamepad)
COLLECTION (Application)
  COLLECTION (Physical)
    USAGE_PAGE (Generic Desktop)
    USAGE (X)
    USAGE (Y)
    USAGE (Z)
    USAGE (Rx)
    USAGE (Ry)
    LOGICAL_MINIMUM (-127)
    LOGICAL_MAXIMUM (127)
    REPORT_SIZE (8)
    REPORT_COUNT (5)
    INPUT (Data,Var,Abs)
  END COLLECTION
END COLLECTION
```

Figure 6 HID report descriptor (pseudo-code/actual hex values omitted).

VII. TEST, VERIFICATION AND VALIDATION

A. Functionality

Functionality was measured based on whether performing a gesture successfully performed the intended action on the host computer. The initial requirement for a fully functioning system included having all six gestures: enable, disable, pan, rotate, zoom in, and zoom out. Unfortunately, due to time constraints in integration, only four of the basic gestures are working at a usable level: pan, rotate, zoom in, and zoom out. Additionally, dynamic calibration was not added to the system. Instead, the system is pre-calibrated so that it works on a normal performance of each gesture.

B. Accuracy

TABLE IV. CLASSIFICATION ACCURACY TEST RESULTS

Gesture	Accuracy
Pan	97%
Rotate	90%
Zoom In	73%
Zoom Out	97%

Testing accuracy measures whether the appropriate gesture is classified correctly after it is performed. If no gesture was detected, then the test would be considered to have failed.

The size of the dataset was initially 30 samples per gesture, although the sample size can easily be easily increased. The gesture recognition accuracy was measured by evaluating the test sensor inputs into the model and comparing the classification against the true labels. The accuracy will be calculated as the number of correctly classified gestures divided by the total number of gestures.

C. Latency

TABLE V. LATENCY TEST RESULTS AND BREAKDOWN

Measurement	Latency
Oscilloscope measurement of Pi 4 GPIO pin and sensor detection output (overall measurement)	~112 ms
Sensor Data Collection	*~100 ms*
Model Classification Time	*~5 ms*
Wi-Fi transmission delay with ping test	*~5 ms*
HID Dispatch Time	~14 ms
Total end-to-end Latency	**~126 ms**

For a system-wide test, the overall latency will be measured with an oscilloscope. In this test, the compute module will write to a GPIO pin once the HID macro is computed. Afterwards, the GPIO pin and the sensor detection output will be synchronized, and the oscilloscope will measure the time difference. With this method, the exact time it takes for a gesture to be recognized from the time the first relevant input is received from the sensors is measured. This measurement will be repeated for 20 attempts, which will enable the accurate measurement of latency.

Although not part of the design requirements, measuring the latency for each individual portion of the data pipeline will be useful when it comes to debugging. Thus, additional tests will be conducted on the network transmission time and the gesture classification time, as these would be the two biggest bottlenecks. Network latency will be tested by timestamping 20 packets using the Unix timestamp and calculating the time difference between sending from the glove module and receiving from the compute module. The same timestamping method can be used to calculate the time difference between the time when sensor data is received by the compute module and the time when the model finishes classifying the gesture. Finally, HID dispatch time is measured by performing a round-trip message and dividing the latency by two to retrieve one-way latency.

The total end-to-end latency was determined to be about 126 milliseconds, much less than our initial requirement of 1 second.

D. Portability

The final glove was weighed on a measuring scale to ensure that the total weight of the gloves will remain under the target 300 grams. The final weight of the glove was determined to be **91 grams**.

E. Range

Starting from 10 meters, the glove will be moved to a further location away from the user's computer, with an increment of 10 meters per attempt, and going up to 50 meters. In each location, latency and accuracy tests will be conducted to see if the distance affects the test. The measured range was 10 meters, well below the target. However, due to the change in network usage (using the Pi as an access point), this drop in range is expected. The initial requirement of 50 meters was under the assumption that the user would walk to the farthest points in a large lecture hall. Instead, a better metric would have been to find the width of the presenting area in a large lecture hall. As such, the measured range of 10 meters seems like an acceptable result.

F. Battery Life

The battery life was measured by measuring the total current running through the system, given the specification of the battery. The current is then used to calculate the operation time of the battery from its capacity specification. The measured current was 0.037 amperes, which produces a total operation time of approximately 54 hours. This time is definitely longer than the duration of a lecture, so it meets the requirements.

$$t = \frac{\text{electric charge capacity}}{I} = \frac{2000\ \text{mAh}}{0.037\ \text{A}} \approx 54\ \text{h}$$

VIII. PROJECT MANAGEMENT

A. Schedule

The complete schedule is listed in the Appendix. From the design report, the schedule was on track for most of the development process, but testing ended up getting pushed to the last week due to complications in integration and unforeseen hardware failures (explained in System Implementation).

B. Team Member Responsibilities

TABLE IV. TEAM MEMBER RESPONSIBILITIES

Member	Responsibilities
David	• Setup compute module as an HID • Configure Raspberry Pi to run compute module on startup
Xuanye	• Assemble and solder glove circuit • Investigate different wiring mechanisms to minimize strain on sensors • Implement data collection and transmission code on the glove microcontroller
Gram	• Implement data reception pipeline with MQTT broker on the compute module • Design and implement gesture recognition model • Implement system mapping from HID input to application-specific macros

C. Bill of Materials and Budget

The bill of materials and budget is broken down in the Appendix.

D. Risk Management

During the design phase, the hardware components, accuracy, and latency were identified as the main points of failure. The primary risk reduction methods adopted was to allocate more time for development throughout the semester, as many of these risks would only really be discovered during the implementation process.

For the hardware components, risk was successfully mitigated by allocating sufficient time to test durability as well as repair and reinforce the glove as complications in the wiring and other components arose.

An unforeseen hardware risk that arose was the ordered LiPo battery not being of the correct voltage to power the board. Since the microcontroller board was changed during implementation, we failed to consider how the power supply specifications might have changed as well. Despite this, a backup risk mitigation plan that was implemented was to power the glove module using a portable battery pack that was readily available.

On the software side, the risk of low accuracy was mitigated by designing the gesture recognition model with highly configurable hyperparameters in mind. This made it easy to configure and adjust thresholds to filter out noise and identify gestures. Moreover, enough time for testing was allocated to achieve sufficiently high accuracy across the gestures.

Latency risks were mitigated by having different options for the wireless communication setups. During the development process, we realized that it would be difficult to have both modules connect to an external WiFi network, since it would require reflashing the glove module with the correct WiFi credentials. To address this, we instead had the compute module set up its own WiFi AP for the glove to connect to.

IX. ETHICAL ISSUES

The main ethical concern would be the product performing errantly in the middle of a presentation. This would disrupt the educational context that was meant for the product.

Another potential concern could be if an adversary were to connect to the wireless compute module's hotspot and hijack control of the host computer. While this was considered outside the scope of the project, one potential way to mitigate this in the future would be to establish a more secure channel of communication between the compute and glove modules.

X. RELATED WORK

There have been a few studies related to the field of improving accessibility for presenters. Some of these studies involved using cameras to track the position of laser pointers to classify gestures and actions drawn on the projection screen [4], [5]. While being economical, this is not very intuitive as users need to learn a new set of gestures and actions to interact with their computers. Moreover, requiring the setup of cameras means that the system is not very portable and needs to be installed with supports to hold the camera steady throughout the presentation – something that not many educators necessarily have access to.

Other research has also been performed in the field of creating virtual reality controllers in the form of gloves, very similar to the solution of TeleTouch. These virtual reality controllers are used as general-purpose input devices in virtual reality spaces. Thus, they require a much higher degree of precision [20], [21] which results in higher development and production costs [6]–[8]. TeleTouch aims to be a much more accessible, intuitive, and economical solution to the problem than other alternatives available on the market.

XI. SUMMARY

TeleTouch aims to become the ideal solution for educators and presenters to enhance the experience of presenting and interacting with 3D entities. Through carefully chosen gestures, the system will be intuitive to use, drawing from gestures that are already widely used in existing touch interfaces. Moreover, since the gestures required by this system are much more distinct and do not need as high level of precision as other general-purpose controllers for virtual reality environments, TeleTouch is able to circumvent higher production costs required by using more expensive sensors. Through the success of this product, TeleTouch aims to make education much more engaging and interactive for people around the world.

The system achieved the basic design specifications and requirements. However, due to time constraints, additional features that could have improved the user experience, such as seamless calibration or custom keybindings for different applications, were not implemented. In terms of performance, designing a specialized power circuit to utilize lighter power sources, such as coin batteries, might have been able to reduce the total weight of the glove. Nevertheless, TeleTouch is able to fulfill the original vision of enabling a more intuitive method of presenting 3D models.

Firstly, it would be good to allocate more time in experimenting with different models. In particular, it might be entirely possible to fit the classification model on the microcontroller found on the glove itself. If done properly, this would further reduce the cost needed as the compute module is the most expensive item of the product. There is an opportunity to do an edge learning based system, given the recent advancements of TinyML. This might also allow opportunities to explore more gestures, as an ML based system would not require manual time series analysis for each gesture.

Acronyms

AP – Access Point
CNN – Convolutional Neural Network
HID – Human Interface Device
I2C – Inter-Integrated Circuit
LSTM – Long Short-Term Memory
MQTT – Message Queuing Telemetry Transport
RNN – Recurrent Neural Network
RPi – Raspberry Pi

References

[1] "Viewing a Model | SketchUp Help." https://help.sketchup.com/en/sketchup/viewing-model (accessed Feb. 27, 2023).

[2] "3D Graphics View - GeoGebra Manual." https://wiki.geogebra.org/en/3D_Graphics_View (accessed Feb. 27, 2023).

[3] "Using the Camera - 2020 - SOLIDWORKS Visualize Help." https://help.solidworks.com/2020/english/Visualize/t_using_the_camera.htm (accessed Feb. 27, 2023).

[4] B. Shizuki, T. Hisamatsu, S. Takahashi, and J. Tanaka, "Laser pointer interaction techniques using peripheral areas of screens," in *Proceedings of the working conference on Advanced visual interfaces*, New York, NY, USA, May 2006, pp. 95–98. doi: 10.1145/1133265.1133284.

[5] T. Wada, M. Takahashi, K. Kagawa, and J. Ohta, "Laser pointer as a mouse," in *SICE Annual Conference 2007*, Sep. 2007, pp. 369–372. doi: 10.1109/SICE.2007.4421010.

[6] "Buy | Senso.me." https://senso.me/order (accessed Feb. 27, 2023).

[7] "Store | Hi5 VR Glove." https://hi5vrglove.com/store (accessed Feb. 27, 2023).

[8] "Buy Now - MANUS." https://www.manus-meta.com/buy-now (accessed Feb. 27, 2023).

[9] USB Implementers' Forum, "Device Class Definition for Human Interface Devices (HID)," Firmware Specification 1.11, May 2001. Accessed: Mar. 03, 2023. [Online]. Available: https://www.usb.org/sites/default/files/hid1_11.pdf

[10] E. Ayodele, S. A. R. Zaidi, Z. Zhang, J. Scott, and D. McLernon, "A review of deep learning approaches in glove-based gesture classification," in *Machine Learning, Big Data, and IoT for Medical Informatics*, Elsevier, 2021, pp. 143–164. doi: 10.1016/B978-0-12-821777-1.00012-4.

[11] B. Fang, F. Sun, H. Liu, and C. Liu, "3D human gesture capturing and recognition by the IMMU-based data glove," *Neurocomputing*, vol. 277, pp. 198–207, Feb. 2018, doi: 10.1016/j.neucom.2017.02.101.

[12] J. Nielsen, "Response Times: The Three Important Limits," in *Usability Engineering*, 1st ed., Morgan Kaufmann, 1993. Accessed: Mar. 01, 2023. [Online]. Available: https://www.nngroup.com/articles/response-times-3-important-limits/

[13] I.-J. Ding, N.-W. Zheng, and M.-C. Hsieh, "Hand gesture intention-based identity recognition using various recognition strategies incorporated with VGG convolution neural network-extracted deep learning features," *J. Intell. Fuzzy Syst.*, vol. 40, no. 4, pp. 7775–7788, Apr. 2021, doi: 10.3233/JIFS-189598.

[14] X. SHI, Z. Chen, H. Wang, D.-Y. Yeung, W. Wong, and W. WOO, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," in *Advances in Neural Information Processing Systems*, 2015, vol. 28. Accessed: Mar. 02, 2023. [Online]. Available: https://proceedings.neurips.cc/paper/2015/hash/07563a3fe3bbe7e3ba84431ad9d055af-Abstract.html

[15] M. Hopkins, B. Lattimer, and E. Graff, "Introduction to Human Interface Devices (HID) - Windows drivers," Mar. 19, 2022. https://learn.microsoft.com/en-us/windows-hardware/drivers/hid/ (accessed Mar. 03, 2023).

[16] USB Implementers' Forum, "HID Usage Tables for Universal Serial Bus (USB)," Technical Specification 1.4, Jan. 2023. Accessed: Mar. 03, 2023. [Online]. Available: https://usb.org/sites/default/files/hut1_4.pdf

[17] "Linux USB HID gadget driver — The Linux Kernel documentation." https://www.kernel.org/doc/html/next/usb/gadget_hid.html (accessed Mar. 03, 2023).

[18] "Linux USB gadget configured through configfs — The Linux Kernel documentation," Apr. 25, 2013. https://www.kernel.org/doc/html/next/usb/gadget_configfs.html (accessed Mar. 03, 2023).

[19] "libusb/hidapi." libusb, Mar. 02, 2023. Accessed: Mar. 03, 2023. [Online]. Available: https://github.com/libusb/hidapi

[20] S. Lee, K. Park, J. Lee, and K. Kim, "User Study of VR Basic Controller and Data Glove as Hand Gesture Inputs in VR Games," in *2017 International Symposium on Ubiquitous Virtual Reality (ISUVR)*, Jun. 2017, pp. 1–3. doi: 10.1109/ISUVR.2017.16.

[21] K. Tanjung, F. Nainggolan, B. Siregar, S. Panjaitan, and F. Fahmi, "The Use of Virtual Reality Controllers and Comparison Between Vive, Leap Motion and Senso Gloves Applied in The Anatomy Learning System," *J. Phys. Conf. Ser.*, vol. 1542, no. 1, p. 012026, May 2020, doi: 10.1088/1742-6596/1542/1/012026.

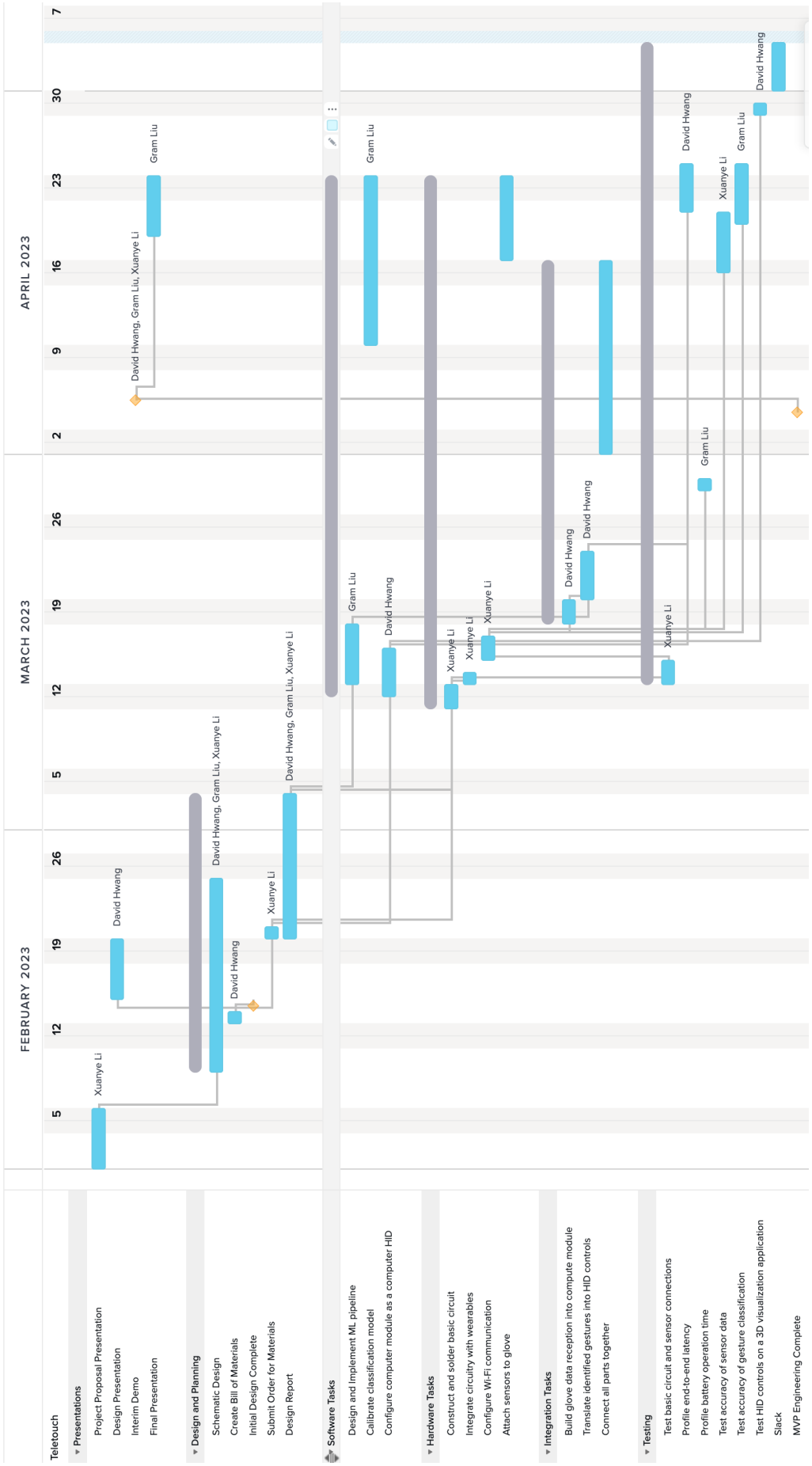
FEBRUARY 2023
MARCH 2023
APRIL 2023
5
12
19
26
5
12
19
26
2
9
16
23
30
7
Teletouch
Presentations
Project Proposal Presentation
Design Presentation
Interim Demo
Final Presentation
Design and Planning
Schematic Design
Create Bill of Materials
Initial Design Complete
Submit Order for Materials
Design Report
Software Tasks
Design and Implement ML pipeline
Calibrate classification model
Configure computer module as a computer HID
Hardware Tasks
Construct and solder basic circuit
Integrate circuitry with wearables
Configure Wi-Fi communication
Attach sensors to glove
Integration Tasks
Build glove data reception into compute module
Translate identified gestures into HID controls
Connect all parts together
Testing
Test basic circuit and sensor connections
Profile end-to-end latency
Profile battery operation time
Test accuracy of sensor data
Test accuracy of gesture classification
Test HID controls on a 3D visualization application
Slack
MVP Engineering Complete
Xuanye Li
David Hwang
David Hwang, Gram Liu, Xuanye Li
David Hwang
Xuanye Li
David Hwang, Gram Liu, Xuanye Li
Gram Liu
David Hwang
Xuanye Li
Xuanye Li
Xuanye Li
David Hwang
David Hwang
Xuanye Li
David Hwang
Gram Liu
Xuanye Li
Gram Liu
David Hwang
David Hwang, Gram Liu, Xuanye Li
Gram Liu

TABLE VI. BILL OF MATERIALS

Item	Description	Quantity	Price (per unit)	Subtotal	Used
Short flex sensor	3.054" long	15	$7.16	$107.40	☑
ESP32-C3 RISC V		2	$9.95	$19.90	☐
ESP32-S3		2	$19.95	$39.90	☐
Coral USB Accelerator	Edge TPU coprocessor comp	1	$59.99	$59.99	☐
Raspberry Pi 4 8GB	ECE Inventory	1	$0.00	$0.00	☑
MPU-6050	IMU (Accelerometer + Gyro)	2	$13.00	$26.00	☐
Gloves	Stretchy gloves (pack of 24	1	$12.99	$12.99	☑
Lipo battery charger	USB charger for lipo batterie	2	$5.95	$11.90	☐
Lithium Ion Battery	3.7V, 2000mAh	2	$12.50	$25.00	☐
USB/Power Splitter	USB-C/Power splitter for Ras	1	$3.95	$3.95	☑
USB-A to USB-C cable	USB-A to USB-C 3.0 data an	2	$4.95	$9.90	☑
Power Supply (RPi)	15W Power Supply for Raspb	1	$8.00	$8.00	☐
Perfboard	Pack of 10	2	$5.00	$10.00	☑
Slide switch		5	$0.95	$4.75	☐
Indicator LED	Active indicator LED (Pack o	2	$3.95	$7.90	☐
4-pin to male headers ca	For MPU 6050	2	$0.95	$1.90	☑
Power indicator LED	Power indicator LED (Pack o	1	$3.95	$3.95	☐
Arduino Nano IoT	Microcontroller with Internet	2	$24.00	$48.00	☑
LiPo battery adapter bre	Switched JST-PH 2-Pin SMT	2	$2.50	$5.00	☐
LiPo battery direct recei	JST-PH 2-Pin SMT Right Ang	2	$0.75	$1.50	☐