Magic Mice

Author: Bradley Zhou, Jade Wang, Jenny Han: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—**Magic Mice is a projector computer attachment that displays a Windows computer screen on an indoor wall. With a custom pen device that controls mouse movement and the computer webcam video feed that detects hand location and customizable hand gestures, Magic Mice simulates the functionality of a smartboard. It allows the user to mimic mouse movement and keyboard commands that are custom mapped to hand gestures. Magic Mice is an affordable collaboration tool for team workers in educational, professional, entertainment settings, etc., as it allows users to draw on sketch websites, navigate browsers and file systems, and more.

***Index Terms*—**Hand Gestures, Laptop, Machine Learning, Open CV, Projection, Sensors

I. Introduction

IN our increasingly online world, amplified by a pandemic, much of our lives and communication with others has become solely virtual. Many companies have switched to an entirely work-from-home system, meaning all communication happens over video calling platforms. Most education systems have also been forced to be conducted online, making the learning process even more complicated for everyone, teachers and students alike. Our world today is dependent on these communication platforms. However, these technologies have their limitations. Zoom, for example, has a drawing functionality that allows people to whiteboard, brainstorm, and discuss ideas on video calls. Teachers in wealthy areas will also use technologies like iPads, Surface Pros, or Smartboards to write and teach to their students through video calling platforms. However, not all organizations can afford to supply $1000 devices to every staff member.

With Magic Mice, the process of sharing information and effectively collaborating in our online world will be more accessible and cost effective. Magic Mice is a Windows computer projector attachment that allows users to display their screen content onto a wall in an indoor room with low light. Users can then interact with the screen through a custom sensory pen, which mimics mouse movement, and hand gestures, which are custom mapped to keyboard shortcuts. Magic Mice also includes a GUI that allows the user to calibrate the system to work at any distance between 5-10 ft, add new hand gestures, and customize them to map to keyboard shortcuts. and Magic Mice allows users to interact with the projected wall via the pen with a delay of 20 ms, and via hand gesture with a delay of less than 50 ms. It is also able to recognize up to 10 different hand gestures at once with an accuracy of 17 out of 20 times. Each gesture can be mapped to any valid keyboard shortcut on a Windows operating system.

II. Design Requirements

Our design requirements are defined in Table 1 to ensure the project's functionality and a reasonable user experience. The requirements are split into three main categories for each component of our project: pen, gesture recognition, and the projector system. The first component, the pen, is defined by its location being calculated accurately enough such that the mouse location, when viewed on the projected screen, is 1 inch away from the user's actual pen location. The pen clicks must also have a response time of 50 ms or less, and a polling rate of 100 Hz. The combination of these requirements will allow for a seamless mouse experience for the user. There will be no noticeable lag and the user will be able to use the pen to a degree of accuracy such that they can draw and write legibly. The second component, the hand gesture recognition, is defined by a 95% accuracy rate in recognizing hand gestures. The user must also be able to remove hand gestures and add custom hand gestures with the same amount of accuracy. The third component is the projector system itself which is defined by the projector being calibrated 5-10 feet away.

To test the pen's location accuracy when drawing on the projected screen, we will be using sketch.io, a sketching website on the Chrome browser. We will proportionally calculate the equivalent of 2 inches on the projected screen based on the computer screen's dimensions and projector's distance from the wall. We will then draw a square and a circle using the pen and measuring the tip of the pen's distance from where the drawn point on screen actually was. We will be doing this in 0.5 ft increments from 5 to 10 ft. To test the pen's button click response time, we will be utilizing timing code that measures from the point in which the button itself was clicked on the pen device, and the point in which the associated command was executed, such as the PyWin32 mouse command or the calibration code. To test polling rate, we have the program count how many Arduino IMU readings it received through PySerial in a fixed time interval. To test response time, we have the Arduino record a timestamp and then send a piece of data to the computer. A python program will be waiting to receive this piece of data and upon receiving it, will send another piece of data back to the Arduino. The Arduino records another timestamp upon receiving this data and taking the difference between the first

and second timestamps will give us an upper bound on the round trip time of communication. Dividing by two will give us a rough estimate of response time.

We'll be testing our gesture recognition algorithm by verifying our computer camera input maps to the correct hand expression. We'll be splitting our original marked data set into a 20 to 80 for a test to train dataset ratio. We will be running the trained model on a 20% test data set and strive for an accuracy of >85%. As for new hand gestures that aren't defined as default gestures already, we'll verify that the system is correct 17 out of 20 times. We will do this by taking 10 videos of 10 different people making the hand gesture. These videos will be inputs into the algorithm and would need to be matched correctly 9 times. We will repeat this for 10 hand positions outside of the default. The process of 10 different people doing the same hand movement for the test will give enough variation to test the accuracy of the system.

As for our projector, we'll be testing all distance ranges from 5-10 ft in 0.5 ft increments to ensure calibration works at all reasonable distances. We'll be making sure that a 2px thick vertical line has clearly depicted edges on the projected wall. 2px was chosen because it was thin enough to potentially be blurry in a low-resolution projection screen but thick enough such that the user would be able to see it. If the user does not see this 2px stroke, then it will be difficult to see.

Component	**Metric**
Pen	- 1 inches when used with a projector - 50 ms button clicks response - 100 Hz polling rate
Gesture Recognition	- 85% accuracy - Add/remove new gestures
System/Projector	- Calibration for distances of 5-10 ft

TABLE I. Design Requirements

III. Architecture and/or Principle of Operation

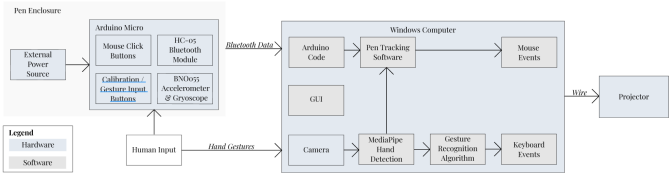

Fig. 1. Overall System Block Diagram

Our system takes in two main methods of human input, hand gestures and control of a custom sensory pen. The system is also controlled via calibration and gesture name and macro input in the GUI. As shown in Figure 1 above, the sensory pen is made of an external power source and several Arduino parts that are used to sense inputs. There are three push button switches. Two are for left and right mouse clicks, and the third one doubles as orientation calibration and gesture input. There is a BNO055 IMU to sense accelerometer and gyroscope coordinates for our pen tracking algorithm, and a HC-05 Bluetooth module that communicates all of the sensor data to the Windows computer, where all data processing is done.

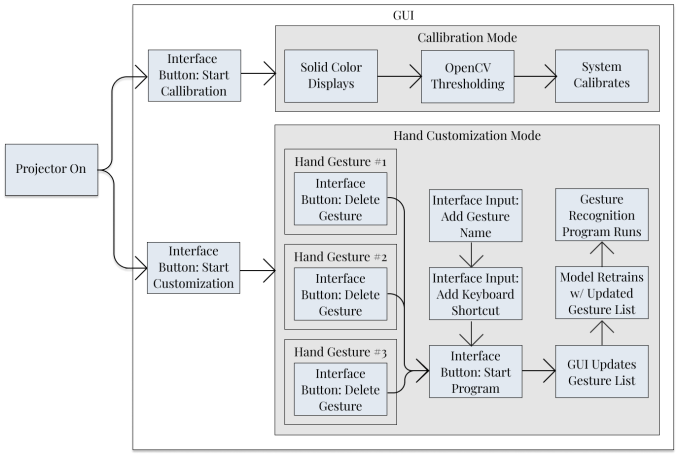

Fig. 2. GUI System Flow

The sensor data is initially read as Arduino data every 200 ms and communicated to the computer through an HC-05 Bluetooth module. This data is read in through Arduino Code that aggregates all of the sensor data and sends it to the pen tracking software as a dictionary of Python values. This software also takes in hand location coordinates from the Media Pipe Hand Detection algorithm. This is an additional input into our design of the pen tracking algorithm since the design report. Using these two inputs, the pen tracking software's algorithm then takes a weighted approach to output respective mouse control of location, left, and right click button actions as well as calibration and gesture input events. Additionally, hand gesture image inputs are captured through the Windows webcam video feed. The same Media Pipe Hand Detection algorithm outputs isolated frames of the user's hand and sends it to the gesture recognition algorithm. This algorithm processes these inputs, detects which hand gesture is being displayed, and maps them to their respective keyboard shortcuts. Pen mouse control outputs and keyboard shortcuts are effectively executed on the Windows computer. These actions are then reflected onto the projector screen through a physical wire connected to the computer.

The projector system is calibrated through a Windows GUI application. Before the custom sensory pen can be used, the projector must be turned on and placed at the desired distance away from the wall the laptop's screen will be projecting to. The calibration mode is activated through clicking on one of the interface buttons on the GUI. The program will then display a solid color (default red) full screen on the projector. The webcam should be pointed towards the projection from the projector and will use OpenCV thresholding to determine what portion of the webcam's view corresponds to the projection from the projector. This allows us to more accurately map the user's hand position.

Furthermore, the GUI has a hand gesture customization

mode in which the user can set custom hand gestures to map to any keyboard shortcut. As shown in Figure 2, once the customization mode is entered via another interface button, the user can delete any of the listed hand gestures, as well as type in a new name and keyboard shortcut should they choose to add one. Once they click the 'start' button, the GUI will automatically take them into the training mode of the gesture recognition software. The user must make their custom hand gesture for approximately 5 seconds while holding the gesture input push. The model will then retrain with the updated gesture list.

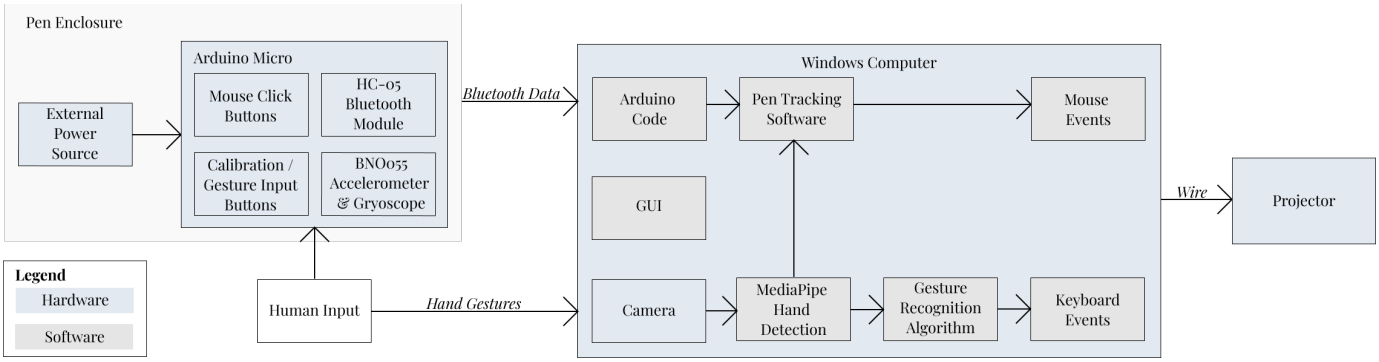

Figure 3. Enlarged Overall System Block Diagram

IV. Design Trade Studies

A. Arduino Model

The model that we ultimately decided to use is the Arduino Pro Micro. We considered various models. We first looked at the de facto standard for many Arduino projects: the Arduino Uno. The limiting feature of this board is the fact it uses the ATmega328 microcontroller. This microcontroller does not have any native USB capability. USB capability is important because it allows the Arduino to be recognized as a mouse/keyboard input device. This gives us great flexibility and more options in actually creating the mouse movement events to pass through to the operating system. While the Uno does not support USB capability, the Arduino Leonardo and its ATmega32u4 does. In fact, all of Arduino's boards with the ATmega32u4 microcontroller have this desired functionality. Though the Leonardo has all the desired technical functionality we needed, it did not have the ergonomic qualifications we were looking for. Having a width of 53.3 mm, the Leonardo was much too wide to comfortably shape into a pen/stylus form. The width of an average pencil is 6 millimeters. Of course, our pen/stylus will be wider than a pencil but holding a component the size of nine pencils in one's hand is excessively cumbersome. We then looked to the Arduino Pro Micro. The Arduino Pro Micro has the same technical specs as the Arduino Leonardo; they both use an ATmega32u4 microcontroller running at 16 MHz, have 32 KB of memory, and 2.5 KB of RAM. Having all the same technical functionality, including the desired USB capability in a much more compact size and a width of only 18 millimeters checked all the boxes we needed for a main board. The width is only three pencils, which sits significantly more comfortably in the average person's hand compared to the wider Arduino Leonardo.

B. Accelerometer/Gyroscope Model

We were considering two models of accelerometer and gyroscope sensors -- either the BNO055 or the MPU6050. Both were sourced from Adafruit. The BNO055 is more fully featured, having an accelerometer, a gyroscope, and a magnetometer compared to the MPU6050 having just an accelerometer and gyroscope. Additionally, the BNO055 reports multiple forms of sensor data (calibrated, uncalibrated, raw ADC values). We considered the MPU6050 because it had
a simpler interface and is significantly cheaper, which is one of the points we wanted to emphasize about our project. Ultimately, we decided on the BNO055 because cost is secondary relative to the other aspects of our project and having the most data to work with will help immensely in development. Our algorithms require us keeping track of various different pieces of information about the pen. The BNO055 was more complex but it had a wider variety of data points for us to work with. The BNO055 reports absolute orientation, which is fundamental to a lot of our calculations. Some of the additional forms of data reporting already combine some of the readings in ways that are helpful to calculating these pieces of information.

C. Form Factor

The form factors we considered the most are a pen/stylus shape (like an apple pencil) or a glove with the user's fingers being the main point of tracking. The glove is a more novel and unique design that is more ergonomic as it allows the user to have his/her hands/fingers fully relaxed compared to having to actually stress muscles to grip a pen, which could lead to fatigue. A pen shape is more classic and there are many preexisting analogous examples, such as an apple pencil or a smart board pen. We decided to go with a pen shape due to the simpler implementation, the lack of ubiquity for a glove shape (people have different sized hands), and the fact that a pen can allow for more natural drawing click/drag functionality, which is useful in scenarios such as drawing.

D. Pen Tracking Software Architecture

One of the biggest points of contention we had discussed was how to organize the software functionality. We were considering three different architectures for our pen tracking software. The high-level ideas we had to work around are the reading of the accelerometer sensor data, the processing of that data to make a decision on where to move the mouse, and the actual translation of that decision to an OS mouse movement event. For each of these pieces of functionality, we had to decide whether we wanted the Arduino and its microcontroller itself to be responsible or the user's computer and its much more powerful CPU to be responsible. The three options we considered are listed below:

Option 1: Have the Arduino itself perform all of these functions. The Arduino will internally keep track of the cursor's position, read the sensor values directly from the accelerometer attached to it, update the internal representation of the cursor's position, and directly translate that into a mouse movement event that is then sent directly to the user's computer via USB.

Option 2: This option is similar to the first option except for the fact we will have two Arduinos instead of one. The purpose of the second Arduino is to act as a Bluetooth receiver. One Arduino will be used as the pen/stylus, and it will wirelessly (via Bluetooth) send its sensor data to the second Arduino. The second Arduino will be plugged into the user's computer via USB and will be the one that internally keeps track of the cursor's position as well as using the sensor data received over Bluetooth to update the cursor's position in real time. This Arduino also has the responsibility of turning the cursor position movements and translating them into OS events that are then sent via USB.

Option 3: We only have one Arduino, and its only responsibility is to send sensor data over to the user's computer via Bluetooth. The user's computer will be running a user space program that will perform the functionality of keeping track of the cursor's position, receiving the accelerometer data, using the data to update the cursor's position, and finally translating that to an OS event to move the mouse.

Option 1 was the first one that we considered but we had to pivot away from it due to the restriction of being limited to wired USB. While it leads to a logical separation of concerns (all the code related to the pen movement is run on the pen itself) and supports the idea that the pen itself is a standalone input peripheral, it did not give us the Bluetooth wireless functionality we specified in our requirements. Option 2 was what we considered next. This architecture is commonly seen among wireless mice and keyboards that have a USB dongle required to use. This still gives us the benefit of having a logical separation of concerns but has a higher development complexity. There are more points of failure with this approach. Another downside to this method is the higher response time due to having another point of connection for data communication. Option 3 has multiple benefits. The biggest benefit is that the user's computer would be the system running most of the more intensive parts of the code. Computers (on the low end) have a 2.0 GHz CPU, which is a clock speed of over 100x greater than the Arduino's microcontroller. Given that we are aiming for a 125 Hz polling rate, this means that each update (reading in sensor values and updating cursor position) will have to be done in 8ms. Having the most powerful CPU possible running the hot path of the code will ensure that we can reach this polling rate requirement. A CPU running at 2.0 GHz means that each update will have

$$2 * 1024 * 1024 * 1024 / 125 = 17.18 \text{ M} \quad (1)$$

17.8 million clock cycles to perform all the calculations necessary. Additionally, because we have multiple subsystems at place (pen tracking and gesture recognition), consolidating all the code to run on the same platform is also a benefit, though this benefit is more subjective and one that we decided we valued. Ultimately, we decided to proceed with option 3 due to the aforementioned benefits.

E. Gesture Recognition Machine Learning Library

We first considered using OpenPose, a CMU developed open source tool to detect body positions in real-time. We considered this because one of our team members has experience working with the tool in research projects and supports our real-time requirements. As we researched more, there were some roadblocks. First, OpenPose is very computationally intensive, requiring a powerful NVIDIA GPU to efficiently work in real-time. Given that our platform is meant to run on laptops, our workaround would have to be sending the video feed over to a remote machine that is running OpenPose, which although possible, can have reliability issues depending on the user's network. Another issue is that OpenPose exists more as a standalone tool and integrating its output into ML models to actually detect gestures is unintuitive. Given all these limitations, we looked to OpenCV next, which is more lightweight. Upon research, we discovered that while OpenPose provides much more detail, the extra detail is not necessary, as gestures can be reduced to shapes that can be extracted from grayscale images. OpenCV sacrifices data detail for much more portability. OpenCV also provides a simpler interface for integrating with machine learning: reducing the image to just grayscale sequence of pixels and passing that information through to neural networks is a very common approach for recognition models (handwriting shape, etc). We ultimately decided these features were more important, so we moved forward with OpenCV.

V. System Description

Hardware System Description

A. Arduino Pro Micro

The Arduino Micro serves as the main microcontroller board of the pen that connects all of the Arduino sensors and components together. It was chosen for its small and manageable size and laptop mouse compatibility. The

Arduino Micro is based on the ATMega32U4 board with built-in USB communication, allowing the Micro to appear to a connected computer as a mouse.

B. BNO055

The *BNO055* is an accelerometer, gyroscope, magnetometer, and orientation sensor. It was chosen for its Arduino compatibility and manageable size. The sensor also encapsulates four sensor functionalities in one. We experimented with all the sensor readings to find an accurate algorithm for distance. Finally, we came up with one that combined an accelerometer, gyroscope, and OpenCV information. The accelerometer is a 3-axis accelerometer that gives us coordinates about which direction is down through measurement of gravity. The gyroscope measures data on how fast the sensor is being twisted around. This combination of these data types allows for calculation of how our sensor pen is moving through space.

C. Buttons

The pen has three switch push buttons in use. The button on the top side of the pen mimics the 'left click' on a computer mouse. The button in the middle of the pen mimics the 'right click' on a computer mouse. The button on the bottom is calibration mode for the projector distance. For debugging purposes, clicking top and middle buttons releases the cursor control from the pen to the computer mouse. All clicks of the buttons and push/hold of the buttons are detected.

D. HC-05

The HC-05 Bluetooth module was selected as the mode of communication between the pen and software program. The Bluetooth module aggregates all of the sensory data through an Arduino script and sends all of the data to the laptop via Bluetooth communication. Bluetooth served as a simple way for us to avoid using wires for a better user experience. This specific module was also selected for its Arduino compatibility and small size. We found later on that the HC-05 was incompatible with Mac, so we switched to using a Windows computer.

Software System Description

Python and C++ will be the main languages we will be using for our project. This is because Python has support for all the features we plan to implement. For the pen output, our Python program will read in the Bluetooth data using the PySerial library, a fairly commonly used Python Bluetooth API. The system will need to decipher what to do based on the click type, such as hover, long press, scroll, click, and perform that action at the correct corresponding location of the computer. The system also will use OpenCV and machine learning to detect hand gestures. These hand gestures will map to customizable user macros. After this, the system will update the projector to update the display that the user sees. Figure 4 describes the system specification for the software component of our design.

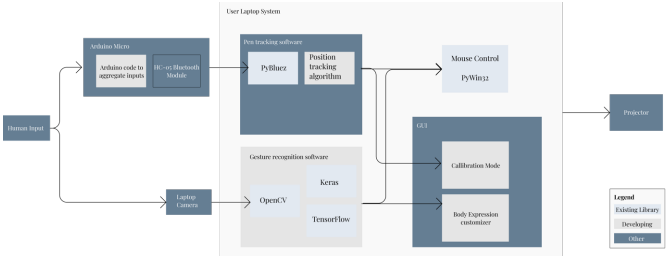

Figure 4. Software system specification overview

A. Gesture Recognition

The gesture recognition software will be used to input the user's webcam images to specific gestures. We will do this by using OpenCV to identify hand gestures. Additionally, we will be using Keras and Tensorflow to make use of convolutional neural networks, a common type of machine learning for mapping of image inputs to an output variable, which in our case will be hand positions. Then, these hand positions map to user macros, such as browser page refresh, or going back a page. These will also be implemented with the same PyAutoGUI library.

B. Pen

Finally, the pen will also have a software component as this is where the processing of our pen's sensor data to output location is done. For the pen, we will have Arduino code written in C++ that aggregates all of the collected data input. This includes left and right button clicks, accelerometer data, and gyroscope data. The HC-05 module will then take all of the data and send it to the computer via Bluetooth. Our program then uses a weighted approach of several inputs to output the pen's location. It considers the inputs of the hand's location detected by the same MediaPipe library used in the gesture recognition algorithm, the location of the tip of the pen relative to the center of the hand computed with gyroscope data, and distance data computed from applying a Kalman filter and double integrating the accelerometer data.

VI. Test and Validation

A. Pen buttons/IMU/Bluetooth

These were tested and validated by sending the IMU/button state data over to the computer over Bluetooth and writing a python program to read in the data and see if it matched with the data that was sent.

B. Pen tracking algorithm and calibration

These two components were tested concurrently because they had dependencies from each other; the algorithm had variable scale parameters that would be changed depending on the resolution of the projector/screen as well as the projector/screen size. We tested these functionalities by running our calibration and then making general pen movements with the projector at multiple different distances. We then looked at the error between the location of the tip of the pen and the location of the cursor and measured them to see that they did not exceed an acceptable error range (which we decided to be 3% of the screen size). For example, if the projector screen was 60 inches diagonally, we wanted the cursor to be no more than 2 inches away from the tip of the

pen.

C. Polling rate and response time

To test polling rate we have the program count how many Arduino IMU readings it received through PySerial in a fixed time interval. To test response time, we have the Arduino record a timestamp and then send a piece of data to the computer. A python program will be waiting to receive this piece of data and upon receiving it, will send another piece of data back to the Arduino. The Arduino records another timestamp upon receiving this data and taking the difference between the first and second timestamps will give us an upper bound on the round trip time of communication. Dividing by two will give us a rough estimate of response time.

D. Gesture recognition

To test gesture recognition, we trained three default gestures (open hand, fist, thumb right) and proceeded to try and use that gesture 20 times for each of the default gestures. We then count how many times that gesture was correctly recognized by our model.

VII. Project Management

A. Schedule

Figure A1 of the appendix shows our schedule for the project. In this schedule, we broke down our tasks to smaller pieces. We left time buffers in the schedule for error and testing. Also, we tried to combine simultaneous elements at the same time.

B. Team Member Responsibilities

There are a few key hardware and software elements in the design of Magic Mice: the pen, the hand gesture recognition, and the overall system. Since these three elements can be done mostly independent of each other and then fitted together at the end, we decided to split tasks this way. Each team member is responsible for the capability of a specific section, the success of that element is checked by another team member, and everyone helps in the case the team member has issues bringing the design to fruition. This way, a team member should be a specialist in a specific area of the project. Since hand expressions are the more complex element of the project, we split the testing more finely for this element.

Jade worked on the gesture recognition software. She helped create the model for the gesture recognition and training mode of the project. She did research on how to best go about the gesture recognition software. Finally, seh tested the gesture recognition and pen components.

Jenny was in charge of the overall system requirements and GUI. This means she helped make sure all the input elements communicate with each other and update the screen accordingly. She also relayed information to the projection on the wall. Jenny also helped automate the gesture recognition algorithm and made it smoothly link to our GUI. Jenny worked on the pen location tracking as well by testing different solutions, researching accelerometer-based distance calculation papers, and testing the pen accuracy.

Bradley was in charge of assembling the hardware elements for the pen and writing the pen tracking and calibration software. He was also in charge of testing that the system updates according to inputs given. Also, he was in charge of testing hand expressions.

C. Budget

The spreadsheet of materials and costs can be found at the end of the report in Figure A2 of the Appendix.

D. Risk Management

We anticipate that parts will be broken as we iterate on our hardware design. Thus, we have ordered several backup parts in case some break. Additionally, we will be using version control and Github to manage our iterative changes and collaborative efforts.

VIII. Ethical Issues

Like most emerging technologies, the older generation that is not as technology literate might not be able to use our product to its fullest capacity. Though we tried to make the functionality as intuitive as possible (gestures and pen), the intuition is only developed by people who have experience using other forms of technology. If the user was disabled and had vision impairment, a motor control disability, or etc, it would be challenging to use this product as well. This project depends heavily on being able to perform the hand gestures and fine motor skills to control the pen.

Another ethical issue would be misapplication with people who are not familiar with technology and have issues learning new technology. People who aren't as tech savvy might have issues because it would require some knowledge of how to set up a projector, how to use a OpenCV application, and how to interact with a computer in a new way (pen + gestures). If someone already was not good with computers, our project would be harder for them to use. This means it is not as friendly socioeconomically or for people with different educational backgrounds.

Additionally, groups of people that we are less exposed to would most likely be vulnerable to misapplication. Since one of the technologies that we are using, Computer Vision, depends on a machine learning model that uses real-time camera input of a human body, we could possibly train the model on a dataset that is not diverse enough. Thus, it is possible that it would not work on certain groups of people.

Finally, another ethical issue would be hacking or performing incorrect user macros. If the algorithm was incorrectly determining hand gestures and performing unexpected macros in a projected screen on the wall, it could be embarrassing or inconvenient to the user who is presenting to other people. Additionally, if the macros were hacked to perform dangerous terminal commands it could be dangerous to the safety of the user's laptop information.

In the future, the ideal data we could collect would be habits of the user's movement. This way, we could use that

information to predict the actions of the user before they did anything. With this, it would improve the UI because the system could act on these predictions and save the user effort. Because we have only tested this on our teammates, it would be hard to collect enough data to make this a reality. Because we don't have this information, we are basing actions off of the current movement of people rather than predictions. Otherwise, if we had enough data of other people's behavior, we could create a prediction model with machine learning. The ethical issue in this case would be privacy. A user can feel uncomfortable by an algorithm memorizing and categorizing their movements. If an algorithm were able to do this, then they could predict what mood the user was in or anticipate their next move. This could inform corporations when the best time to advertise something to a user would be or otherwise manipulate the user's decision making when they are most susceptible to it.

IX. Related Work

Two analogous products that exist are SmartBoards and Microsoft's Surface Hub. The main benefits our project has over these competing products is cost, modularity, and portability. SmartBoards are on the order of $2500+ and Surface Hub's are on the order of $7000+. The only required hardware for our project is a projector, a webcam, and the pen itself. Users have the flexibility to use a projector that is as good or bad as they would like; projectors can be available for less than $100. Many users already have webcams built into their laptop devices, which is the most common use case with our project. The pen components themselves cost about $75, but that cost has the potential to be reduced by choosing components manufactured by cheaper electronic retailers. This is a significantly cheaper price for having a similar feature set of being able to use a pen/finger as an input device. The user also has the freedom to choose more expensive and featured projectors/webcams if they so desire to have a better experience. Because our display relies on a projector as opposed to a fixed screen, our project is significantly more portable as well. While our product is, of course, not as polished as those professional products, our main focus wasn't to try and replicate all the features; our focus was to replicate the most important features at a significantly lower price point while also providing room for user flexibility.

In terms of the pen tracking algorithm, we were able to look through multiple research articles and online resources about how others attempted to calculate distance through accelerometer and gyroscope readings. In terms of research articles, we found a few alphabet matching IMU calculations as well as methods to find the location through filters and double integration. Other than the papers, we also found a SciKit Kinematics library that took IMU data and translated it into distance calculation. Although we didn't specifically implement any of these resources, we used what we learned from all of these online resources to inform our software design for the pen location tracking algorithm.

X. Summary

In summary, our goal was to make collaboration among groups more accessible. Magic Mice is able to display a Windows computer screen onto any wall variable distance and calibrate accordingly. The user will be able to passively and actively interact with the projection using the pen or with customized gestures.

Being able to dynamically change the interface of a screen gives many benefits to an online world**.** From educators to entertainers to professionals, the Magic Mice allows people to create and share on their own terms.

XI. Glossary of Acronyms

IMU – Inertial Measurement Unit

XII. References

[1] Y. Wang, H. Li and G. Shan, "Acquiring the Distance Data with Inertial Measurement Unit in a Wearable Device for the Training of Hammer Throwers," *2018 14th International Conference on Computational Intelligence and Security (CIS)*, 2018, pp. 492-495, doi: 10.1109/CIS2018.2018.00117.

[2] Jamil, Faisal, et al. "Toward Accurate Position Estimation Using Learning to Prediction Algorithm in Indoor Navigation." *Sensors*, vol. 20, no. 16, 7 Aug. 2020, p. 4410., doi:10.3390/s20164410.

[3] Zrenner, Markus et al. "Comparison of Different Algorithms for Calculating Velocity and Stride Length in Running Using Inertial Measurement Units." *Sensors (Basel, Switzerland)* vol. 18,12 4194. 30 Nov. 2018, doi:10.3390/s18124194

XIII. Appendix

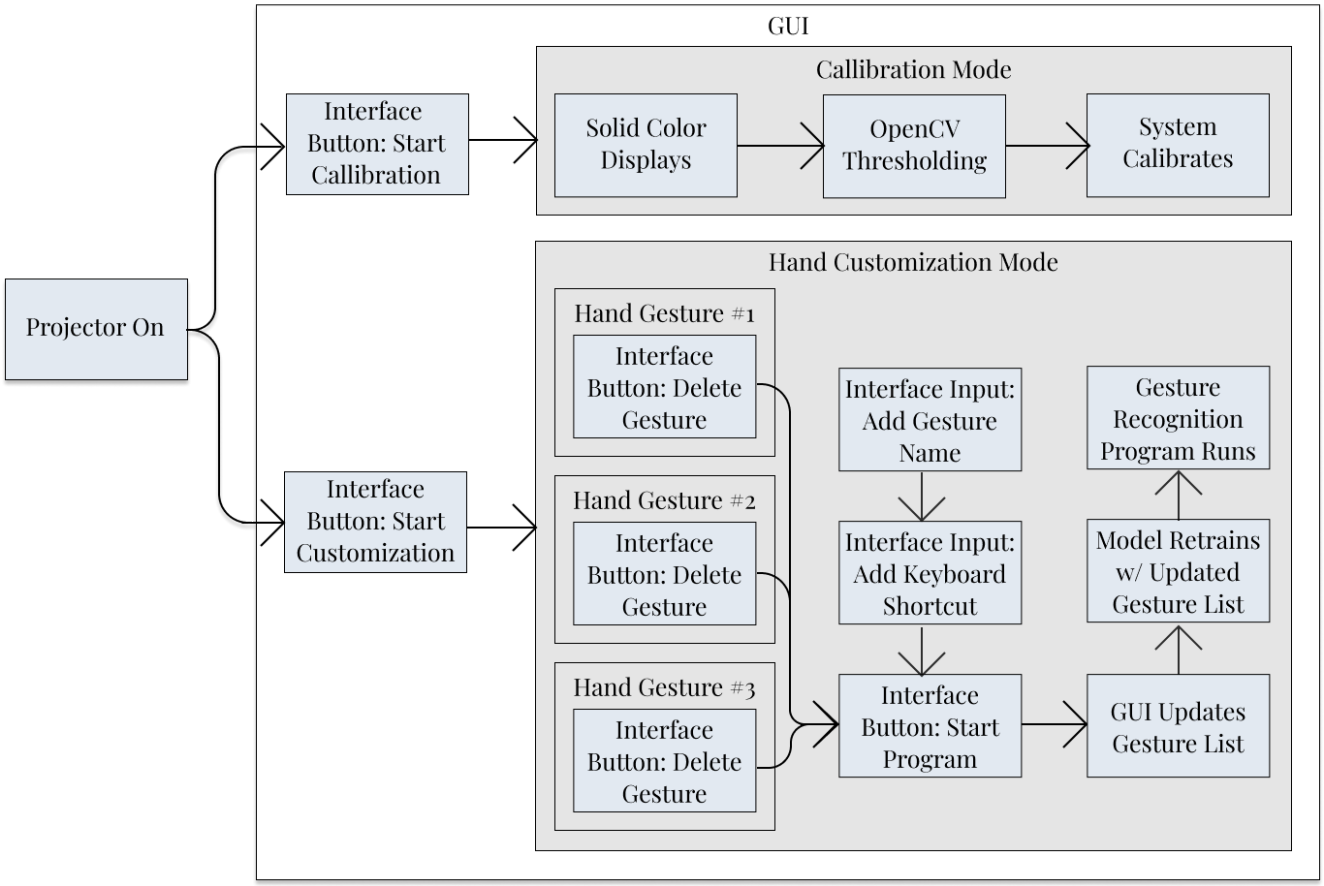

Fig. 2. GUI System Flow

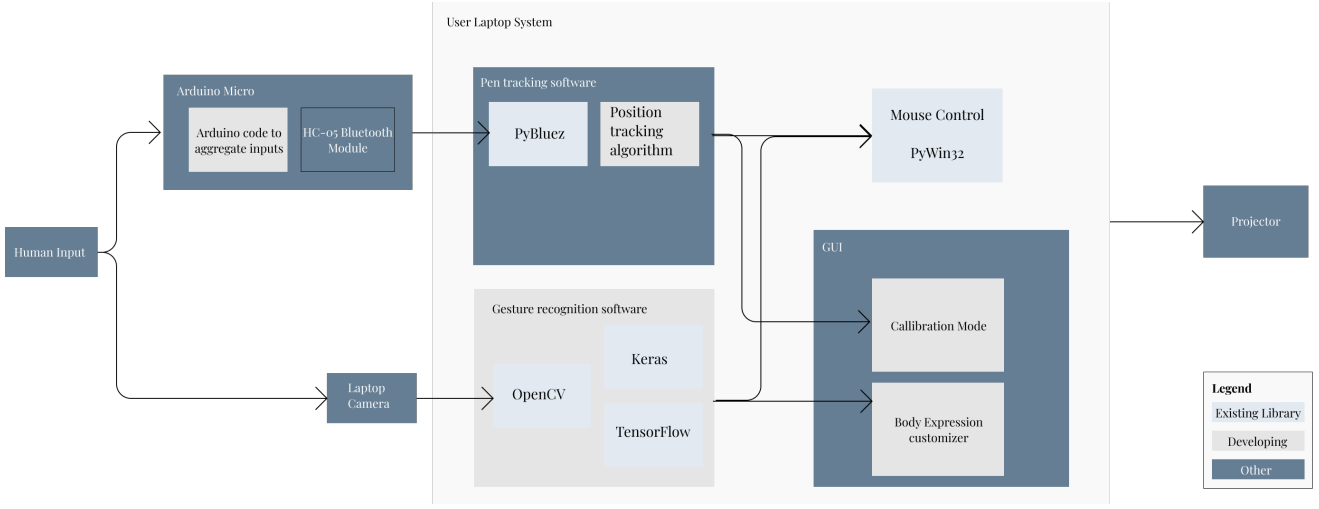

Fig. 4. Software system specification overview

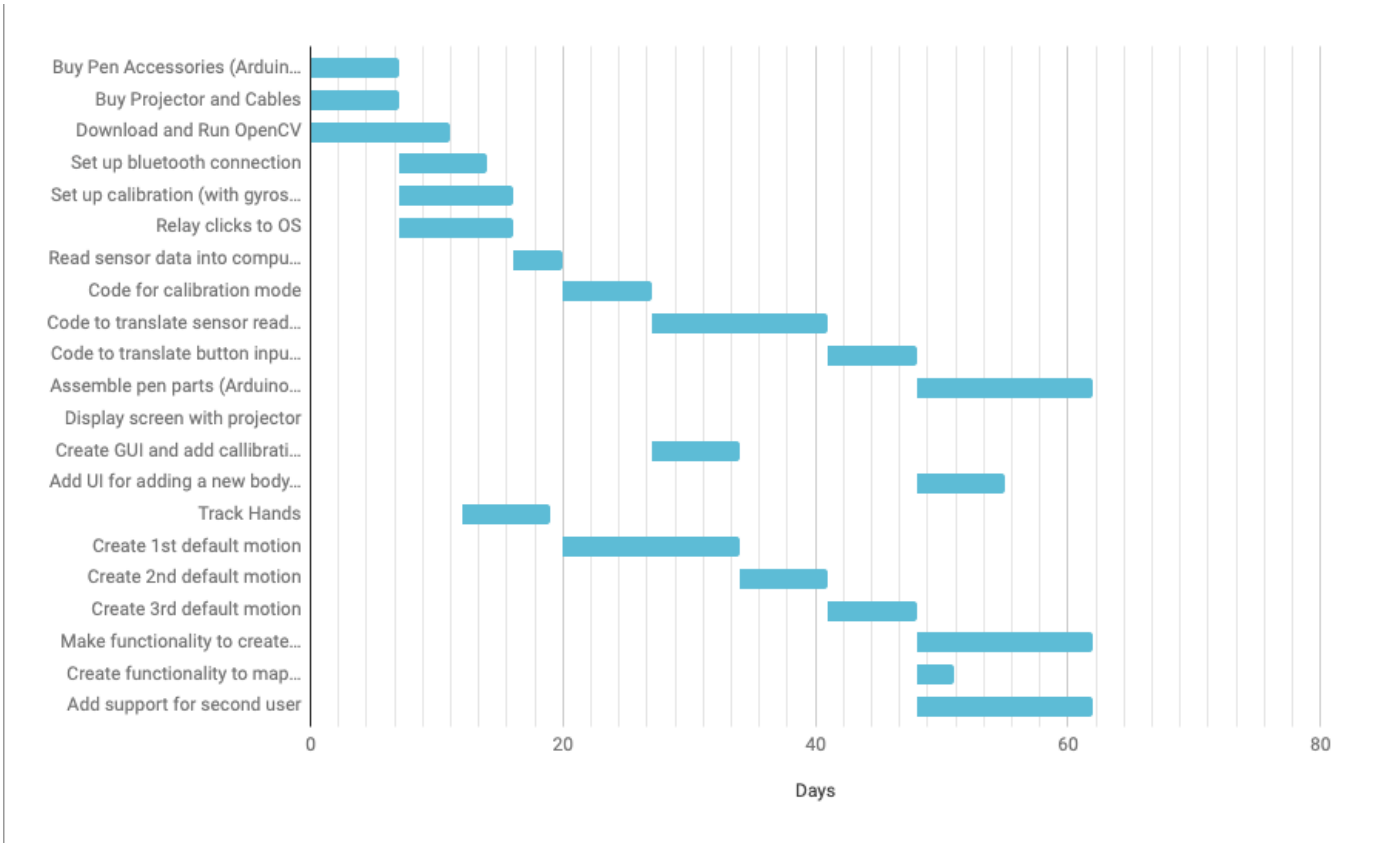

Fig. A1. Gantt Chart

Component	Link	Individual Cost	Quantity	Total Cost (tax + shipping)
Pen				
Arduino Pro Micro	https://www.amazon.com/Arduino-Micro-Headers-A000053-Controller/dp/B00AFY2S56	20.7	2	43.88
MPU6050 Accelerometer/Gyrosco	https://www.adafruit.com/product/3886	6.95	2	26.21
BNO085	https://www.adafruit.com/product/2472	34.95	2	75.4
HC05 Bluetooth module	https://www.amazon.com/HiLetgo-Wireless-Bluetooth-Transceiver-Arduino/dp/B071YJG8DR	7.99	2	16.94
Buttons	https://www.amazon.com/Gikfun-6x6x5mm-Switch-Button-Arduino/dp/B00R17XUFC	5.98	1	6.34
Wires	https://www.adafruit.com/product/4209	0.95	3	3.02
Breadboard	Sourced from ECE lab			
Projector	Sourced from ECE lab			
Total				171.79

Fig A2. Budget