W.R.I.S.T.

Edward Lu, Joanne Park, Anushka Saxena

Department of Electrical and Computer Engineering,
Carnegie Mellon University

***Abstract*—Immersive 3D modeling interfaces, such as hologram pyramids or head-mounted AR displays, require new and novel input devices for high-fidelity interaction. Current trackpad technology on laptops constrains the user to their displays, restricting mobility. In this paper, we plan to solve this issue by building a system that will allow users to use their forearm as a trackpad to manipulate the view of a 3D model. Our system, called W.R.I.S.T., will contain a wearable device with a small footprint, include a 3D hologram pyramid, and have a wireless interface with low latency communication. We plan to integrate tapping, swiping, and pinching gestures to make interacting with 3D models a more immersive experience.**

***Index Terms*—3D Modeling, Hologram, Sensing, Wearables, Wireless**

I. Introduction

The rise of digital environments will improve the way professionals give presentations. Presenters want to be able to interact with their audience while also effectively engaging with the content on their screens. For instance, when university professors give lectures, they typically walk around the room while projecting content onto a 2D display. However, in order to interact with the content on the screen, they ultimately have to walk back to their computers, disrupting the flow of the lecture. This simple solution to this is to use devices like clickers. However, what if the content was more complex, like a 3D model that requires more than just a click to interact with? There needs to be a new input device that can be used to interact with 3D models while a presenter is mobile.

We present W.R.I.S.T.,[1] a system aimed to enable a user to use their forearm as a touch interface to control the view of a 3D model. W.R.I.S.T. allows for mobile, wearable sensing using a surface that is nearly ubiquitous to all human beings, our skin. Our bodies are always with us, no matter where we go, so it is natural to use it as an interface for computing. W.R.I.S.T. includes two main components: a wearable device with a small, compact form factor (about the size and build of a standard Apple Watch), and a 3D hologram pyramid to view a 3D model. Most of the graphical interfaces of W.R.I.S.T. are Web-based, able to run in the browser of nearly all devices that have one. W.R.I.S.T. is an accessible, small, and intuitive interface for presentations that require interacting with 3D models in a remote manner.

[1] Wide Range ImmerSive Teaching

II. Use-Case Requirements

When a user puts on W.R.I.S.T. device, their forearm will transform into a surface capable of recognizing two gestures: swiping and pinching. Each gesture is mapped to two types of manipulations of a 3D model: rotations and scaling.

For a swipe, a user should be able to point their finger and tap the skin on the forearm of the arm wearing the W.R.I.S.T. wearable and swipe across in any direction. This gesture should cause a 3D model to rotate in proportion to the direction and displacement of the user's finger swipe. For instance, if a user displaces his or her finger by 10mm, we expect the model to rotate approximately 5 times less than if the user displaced their finger by 50mm.

For pinching, a user should be able to tap two fingers (thumb and index or index and middle) on their forearm and pinch those two fingers either together or move the two fingers apart; these correspond to shrinking and growing the 3D model. All of the sensor data will be transferred wirelessly through WiFi using the MQTT protocol. Processing gestures will happen on a NVIDIA Jetson Nano.

The 3D model will be displayed on a 3D hologram pyramid made using four acrylic sheets and a standard screen. The model should appear to be holographically rendered onto the acrylic sheets. A user should not be able to notice a large delay between their gestures and the actions done to the 3D model. Additionally, the pyramid should be able to be displayed on any standard Web browser.

III. Architecture and Principle of Operation

With these requirements in mind, our solution is to build a small device as seen on the screen that is slightly bigger than a smart watch. After turning on, the user will be able to move their fingers in front of the device on their arm as if they are moving their fingers on a trackpad. These gestures will be algorithmically translated into a geometric transformation and sent to the scene via a web application and be projected onto the hologram pyramid.

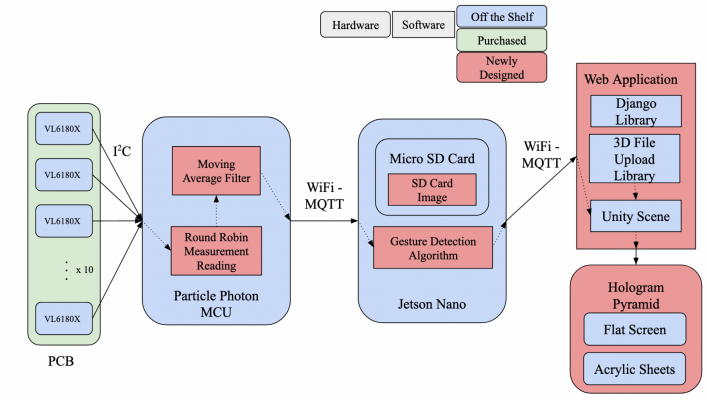

Fig. 1: High-Level Block Diagram

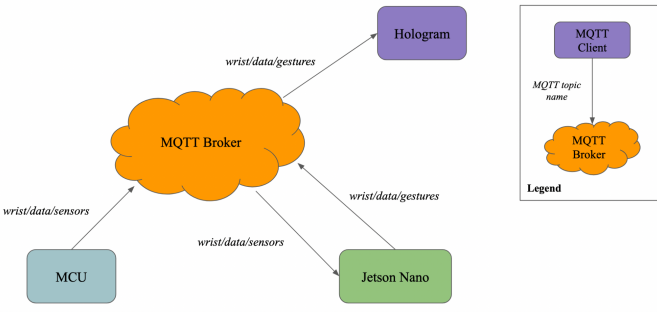

Fig. 2: MQTT Data Flow

There are five main blocks: PCB, MCU, Jetson Nano, Web Application, and Hologram Pyramid. You can refer to figure 1 for a block diagram of our application. The PCB involves the sensors used for distance detection. The MCU controls what sensors are being detected. The Jetson Nano provides gesture recognition computation. The web application takes in the gesture and coordinates associated with the gesture. The hologram pyramid projects the 3D model. This is also the order of how data will be sent.

IV. Design Requirements

A. *Engineering*

The most important requirement is portability. Therefore, the weight and dimensions of the W.R.I.S.T. wearable should be about the same as those of a typical Apple Watch. We aim for the wearable to be less than 100g, less than 65mm by 65mm, and cost less than $150. The heaviest Apple Watch available is about 41.7g, so we aim to be about double that weight at 100g so our device would feel light and comfortable on a user's wrist. We purposefully aim much higher, since we are not creating custom SoC's like Apple, so we want a lot of slack for our weight requirement to account for the off-the-shelf batteries and boards we will use.

With the evolution of visual technologies, we anticipate the next wave will be ubiquitous hologram technology [2]. However, due to expense and lack of availability, holographic projectors have a long way to go before being more common. Instead, we propose the use of a holographic pyramid that will be able to deliver the same effects of 3D models. To design this pyramid, we followed the designs of Pepper's Ghost, a popular optical illusion technique to project objects that aren't in direct view [3]. Each side of the pyramid is an isosceles trapezoid that is connected to each other by the two slanted sides. The two main geometric considerations are a) the smaller angle in the trapezoid is 45 degrees, and b) the ratio of the top and bottom side of the trapezoid is 1:9. The pyramid will be placed on top of a monitor or flat screen that will lie on a table and will be provided by the user.. 22" monitors are popular, which means the monitor is approximately 20" by 10". If we divide the shorter side by 3, we get the size of the smallest size of the trapezoid. Therefore, we estimate that the pyramid size should be 3 ⅓" and the longer size be 30". The material of the pyramid will be acrylic, which is extremely cheap, so we can test out different sizes of the pyramid to achieve the optimal effect.

B. *Gesture Classification*

The intended gesture of the user and the gesture that our classifier identifies should match about 90% of the time for swiping and 75% of the time for pinching. This means we can tolerate 1 failure out of every 10 tries for a swipe and 1 failure out of every 4 tries for a pinch. The reason why we have different accuracies for swiping and pinching is because we predict, based on our initial algorithm and testing plans, that there might be pinching actions that may be detected as swipes. For instance, pinching perpendicularly to the sensors will likely be detected as a swipe gesture.

C. *Gesture Accuracy*

Furthermore, we want the scaling of our transformations to be correct. Due to potential overlap of the sensor readings when a user's finger is farther away from the sensors, we predict that the accuracy of our sensor readings may drop. We plan for an 85% difference between the true displacement of the finger paths when compared to the measured displacement predicted by our algorithm. We expect a somewhat low accuracy here since it will not take much away from user experience if the 3D model does not transform *exactly* proportional to their finger displacement.

D. *Latency*

We also want these gestures to be translated to almost be instantaneous, and we are aiming for a benchmark set by past projects that also required similar technologies [1]. We plan for a 100ms delay between when a user finishes his or her intended gesture to when the 3D model is transformed. This will give about a 10FPS update rate of our 3D scene. While 10FPS is very low for rendering, it does not distract away from user experience. However, we expect to be able to achieve much greater than 10FPS.

V. Design Trade Studies

A. *Sensors*

We considered many different types of sensors when considering the PCB design. For our purposes, we chose the VL6180X (a distance sensor) amongst all the other sensors because of multiple factors. Some of the other sensors we were considering were: pressure sensors, gyro sensors, and

visual sensors. We deemed that these other sensors all affected user experience in the scope of our project. Gyro sensors require more extreme hand gestures which would cause discomfort for the user. Pressure sensors would take away the portability aspect of our design. Because pressure sensors rely on a defined medium to be attached to, it mimics a traditional trackpad. Finally visual sensors need to have the subject in front of a camera in order to detect movement. This would prevent the user from moving outside a fixed range of space.

Even amongst distance sensors, we chose the VL180X because of its short range of 20cm. We did not need a sensor with longer than 40cm distance, and we actually thought having a shorter range would provide more accurate readings. We decided 20 cm would be an okay size since it gives enough space for a user to do a gesture, but also not too large since the human forearm, on average, is only 40 cm. We included a chart below that lists out the different distance sensors we looked at and their feature comparison [4].

B. *Communication*

We decided to not use a wired approach to data communication, since it would defeat the purpose of our project and its goals of mobility. Initially, we were deciding on using Bluetooth for communication, but the BLE device we had was a bit hard to get working to prototype, so we shifted to WiFi. Since we want to integrate with Web technologies, WiFi seemed like the better option. Due to its ubiquity and long range, we decided to work with WiFi. We decided on the Particle Photon for our microcontroller, since it is easy to program and comes with WiFi built-in. For the communication protocol, we were deciding between the Particle Cloud's pub-sub system, HTTP requests, or MQTT. For our case, MQTT seemed like the best fit. The Particle Cloud is mostly a black box, and we wanted to have some more control over deployment. Additionally, HTTP requests are not as fast as MQTT and are typically used for more document-based data rather than raw sensor data. MQTT, which is popular for IoT applications, seemed to allow high throughput communication and excel at sending small quantities of data at a high rate. MQTT also allows for high scalability, as any device can simply subscribe to an MQTT topic and either a stream of our sensor array measurements or our detected gestures and finger coordinates.

	VL6180X	VL6180V1	VL53L4CD	VL53L0CX	VL53L1CX	VL53L3CX	VL53L4CX	VL53L1CB	VL53L5CX
	Proximity and ambient light sensor	Proximity sensor with low power consumption	Proximity sensor with high accuracy	Ranging sensor	Ranging sensor programmable FoV	Ranging and multi-target sensor	Short to long ranging, multi-target sensor	Long Distance and multi-target sensor	8x8 multizone sensor
Part number	VL6180XV0NR/1	VL6180V1NR/1	VL53L4CDV0DH/1	VL53L0CXV0DH/1	VL53L1CXV0FY/1	VL53L3CXV0DH/1	VL53L4CXV0DH/1	VL53L1CBV0FY/1	VL53L5CXV0GC/1
Max distance	20 cm	60 cm	130 cm	200 cm	400 cm	500 cm	600 cm	800 cm	400 cm
Close distance detection	•	•	•	•		•	•		
Multi-target detection						•		•	•
Multi-zone								•	•
Programmable FoV					•			•	
Lower Power mode	•	•	•	•	•				•
Ambient Light Sensing	•								

Fig 6. Comparison chart of different distance sensors taken from ST [4]

C. *Web application*

We thought about multiple web application frameworks, such as Symfony, Express, Ruby on Rails, and Django. However, amongst these, we decided to use Django. We mainly chose Django because our members were most familiar with it and because Django is used using Python. It also provided all of the basic functionalities we needed. We only needed an interface where we could store user data and serve our Unity application, which we could do with Django.

VI. System Implementation

A. *W.R.I.S.T. Band*

The W.R.I.S.T. Band will consist of two main components: our custom-designed PCB for housing ten VL6180X sensors connected through a single I^{2}C bus and a WiFi-enabled microcontroller board.

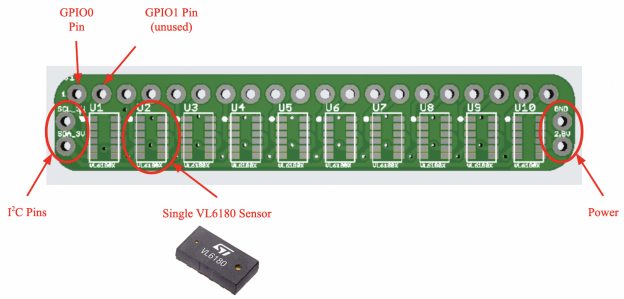

Fig 3.: PCB Diagram

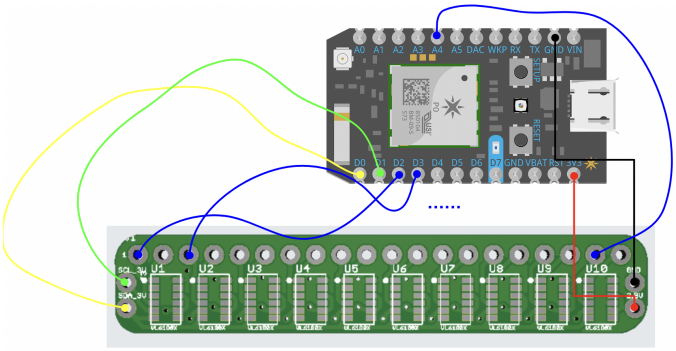

Fig. 4: Wiring Diagram

Our PCB contains a sensor array of ten VL6180X Time-of-Flight distance sensors arranged in a straight line. You can refer to figure 3, 4 for a visualization of our custom PCB design and its wiring. The VL6180X used I^2C for communication and can measure up to about 200mm. Additionally, they contain two GPIO pins: one for turning on and off the device and another for generating an interrupt when sensor data is available. A custom PCB allows us to place the ten sensors in a way so that the sensor array is compact, since we need the sensors to be as close as possible for our algorithm to correctly classify swipes and pinches with as much granularity as possible. We chose ten sensors because ten was the largest number of sensors we could fit into an array before the PCB became longer than the length of one of our wrists.

We choose a Particle Photon WiFi-enabled microcontroller for sensor data collection and transfer. The Photon has 72MHz clock speed, a Real Time Clock, and a Cypress Wi-Fi chip. To collect sensor data, the Photon turns on a single VL6180X sensor and reads data from it and repeats this in a round-robin fashion, storing sensor data into a 10 element, one byte array. Since the VL6180X reads up to 200mm, its measurements can easily fit into 8 bits. We choose to read in a round-robin way due to IR interference across sensors. In fact, the VL6180X sensors read distance measurements from a 25° cone in front of it.

The PCB, Photon, and battery will all sit in an enclosure attached to a wristband. The sensor array will stick out perpendicular to the wrist, while the Photon and battery will lay flat on the wrist. When a finger is tapping on the forearm, some sensors will pick up an object, read its distance, and our algorithm should be able to detect that object given the sensor data. The Photon will read all the sensor data, pack it into an array, and send through MQTT for processing.

B. Middleware and Gesture Recognition

We have two edge servers on our system: Mosquitto MQTT Broker and NVIDIA Jetson Nano. The Mosquitto MQTT Broker will be used to shuffle around data through MQTT topics. Our topic structure is detailed in figure 2. We will have topics for data flow and topics for control and discovery. The Jetson controls both the wearable device and the hologram, and neither will start the W.R.I.S.T. data flow until the Jetson is ready.

The Jetson will perform data preprocessing and gesture recognition. For MVP, we're aiming to perform four gestures: left elation, right rotation, zooming in, and zooming out. We chose these four gestures because they serve as the basis of other transformations such as any direction rotation and translation.

We're using a coordinate plane paradigm where the x-axis represents the direction along the user's arm and the y-axis represents the direction along the sensors. We can place coordinates where fingers are placed and compare the relationship between the points every n number of milliseconds in order to a) accurately predict which gesture is being performed and b) quantify the translation. This n is currently undefined but will depend on how the latency will be optimized. These gestures are displayed in figures 7-10 on page 7.

We will identify what is a finger depending on what's the farthest the sensors go. Depending on if a finger crosses that threshold, an LED light for that sensor will go off to inform the user that their finger isn't being detected. This depends on how the sensors we ordered behave, but we predict that we'll have to set a 10 centimeters limit for movement along the x-axis.

We will set a threshold for the measurements to accurately detect what gesture is occurring. For example, if the differences between the y-coordinates are greater than m millimeters, then we can predict that a zooming action is occurring rather than a rotating action.

C. 3D Hologram Pyramid

The 3D hologram will be presented via a web application and a hologram pyramid. We are planning to use Django for the framework of our web application. Right now we are planning to use the MQTT protocol for data transfer between the Jetson Nano and the web application. We are going to use Unity for manipulation of the 3D object and formatting. Unity has a WebGL option that will allow us to build our content as JavaScript programs that will run our Unity application on a web browser. The Jetson Nano will transmit data in json format that has key-value pairs of:

```
{"gesture": String,
"xs": IntegerArray,
```

```
"ys": IntegerArray,
"timestamp": Float}
```

This will then be parsed by the Javascript on the web application. It is possible to call Unity functions via Javascript function calls with the Unity instance. We will create Unity functions that will handle the incoming stream of information and perform the appropriate modification to the Unity scene and 3d object.

To go more in depth about the hologram pyramid, we will be using four acrylic trapezoids glued together to create our pyramid. We based our trapezoid dimensions based off the angles from Pepper's Ghost Experiment, which is a popular design mechanism for optical illusions. Unity will display four perspective views of the 3D model (*figure 5)*. The white lines are not actually displayed, and are drawn in figure 5 to show where the pyramid base will be in respect to this image. The shorter base edge of the trapezoids will go along the white square, and the lines coming out of the corners of the square represent the edge between two trapezoids.

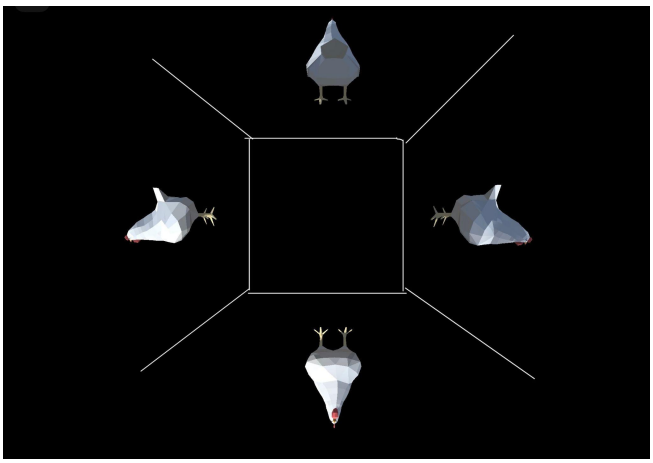

Fig. 5 Unity View of 3D model

VII. Test, Verification and Validation

Our testing will be done with a user study of 5 users. Each user will put on a W.R.I.S.T. band and perform 10 actions. Trail 1 will be swiping 10 times. Trail 2 will be pinching in 10 times. Trail 2 will be pinching out 10 times. We will record what our classifier predicts the user's gesture for each trial.

A. Engineering

To keep track of the portability of our design, we will measure and keep track of weight, dimensions, and cost of our final prototype. We will ask the users in our user study and keep track of whether the weight is too heavy or if the device is too big/small for someone's wrist.

B. Gesture Classification

To measure the correctness of our gesture detection algorithm, we will have each user swipe 10 times, pinch in 10 times, and pinch out 10 times. Each time, we will record what the user is doing and what our classifier detected. To quantify our accuracy, we will take (# times we correctly classified gesture) / (# times the user performed a gesture).

C. Gesture Accuracy

To measure the accuracy of our gesture detection algorithm, we will measure the true, actual displacement of various users' swipes and pinches and compare that to our calculated displacement. We will perform 10 trails with 5 users and have them each swipe and pinch while wearing a W.R.I.S.T. wearable and count the number of correct detections. Each user will swipe and pinch 10 times. To quantify our accuracy, we will take the percentage of (actual displacement of finger) / (measured displacement of finger).

D. Latency

To measure the latency of our system, we will timestamp every incoming and outgoing packet of data. We will measure the latency from our sensors to MQTT to the Jetson, and the latency from the Jetson to the hologram Web application. We will use the Unix timestamp with millisecond accuracy for this. To find the latency, we will subtract our timestamps (current Unix timestamp - Unix timestamp of packet). We will find the time difference (in milliseconds) of *n* packets over one minute of use and take the average for a final number.

VIII. Project Management

A. Schedule

Our tentative schedule is attached on page 9.

B. Team Member Responsibilities

Even though there are five blocks, there are six main parts of the capstone. Edward will be working on board creation and communication. Joanne will be working on the Unity web application and hologram, and Anushka will be working on the gesture algorithm and prototyping. Although each member on our team is specialized in at least one part, we all work together on each block since there are a lot of dependencies from one block to another.

Our schedule is seen in figure 12.

C. Bill of Materials and Budget

Bill of Materials is seen in Figure 11.

D. *Risk Mitigation Plans*

We are planning to create a custom PCB to accommodate our wristband size. However due to the term of our project, the shipping of the PCB after design is one huge risk factor we had to consider. It would be hard to have multiple iterations of our PCB because of time constraints. We planned to mitigate this risk by first modeling our PCB off an already existing PCB. We also got a professor to check our PCB before we ship it off.

IX. Related Work

There are many portable trackpads out there, such as the Apple Magic Trackpad, Logitech Wireless Touchpad, which could provide a similar gesture recognition function as our wearable watch. However none of the existing trackpads out in the market right now has a design where there is no actual physical medium for a user to perform gestures on. Our wristband makes use of distance sensors to detect gestures made on our arm, thus adding a level of mobility and portability that differs our product from the existing trackpads.

We are designing a hologram pyramid to visualize 3D objects and manipulate them using gestures taken by our wristband. When compared to the other existing 3D model viewing technologies such as Autocad, Unity, three.js, many of them require the user to view from a display. They are limited to where they can view this 3d model. We provide this hologram pyramid as a way for users to view their model from any space and also be able to walk around it and manipulate its view using our wristband. Thus also adding another level of uniqueness from the existing trackpad technologies out there.

X. Summary

We are hoping that this project will change the immersive technology field in the future and to improve how we interact with computer graphics and modeling. We are changing the field of digital environments that will improve how professionals give presentations. Presenters want to be able to interact with their audience while also effectively engaging with the content on their screens. Professors who prefer moving around the classroom, engineers who want to provide a more interactive experience to stakeholders, and doctors who want to inspect MRI imaging in a more practical sense will be able to use our product with ease and add value to their daily work.

Glossary of Acronyms

MQTT – Message Queuing Telemetry Transport
OBD – On-Board Diagnostics
RPi – Raspberry Pi

References

[1] Deber, Jonathan, et al. “How Much Faster Is Fast Enough?: User Perception of Latency & Latency Improvements in Direct and Indirect Touch.” Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, 2015, pp. 1827–1836.

[2] Team ISTE. “8 Classroom Uses for Holographic Technology,” ISTE, 2015, https://www.iste.org/explore/ISTE-blog/8-classroom-uses-for-holographic-technology.

[3] “Pepper’s Ghost: Holgoram Illusion,” Science World, https://www.scienceworld.ca/resource/peppers-ghost-hologram-illusion/.

[4] “Time of Flight Sensors,” ST, https://www.st.com/en/imaging-and-photonics-solutions/time-of-flight-sensors.html.

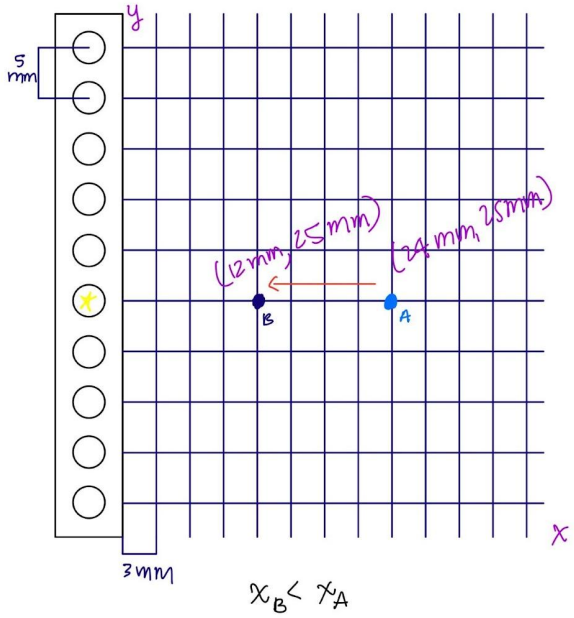

Fig. 7: Left rotation: the x coordinate after n ms is less than the previous x coordinate.

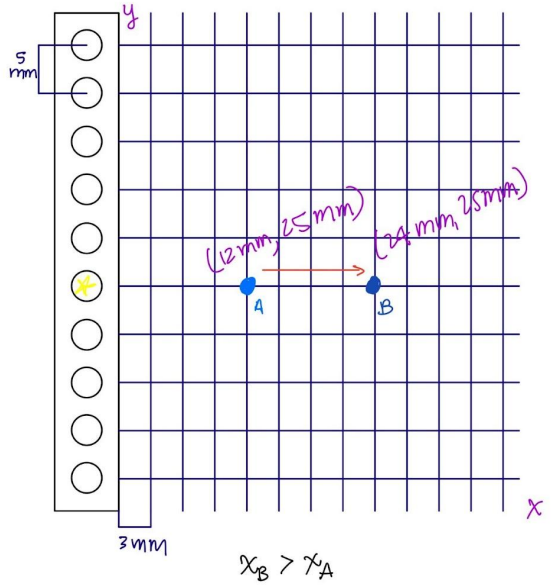

Fig. 8: Right rotation: the x coordinate after n ms is greater than the previous x coordinate

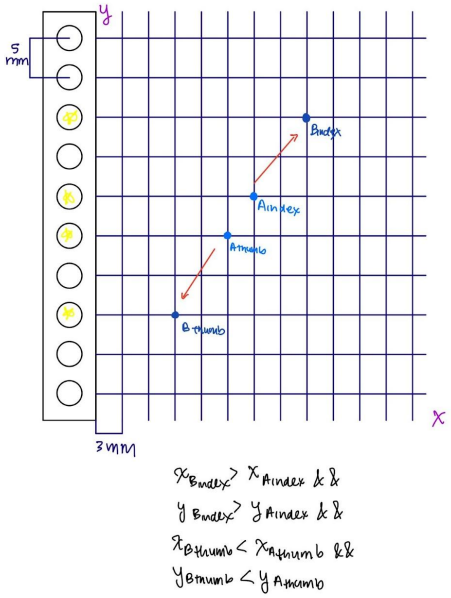

Fig. 9: Zooming out: both x- and y-coordinates of the finger nearest to the sensors after n ms are less than the previous x- and y-coordinates respectively, and both x- and y-coordinates of the finger farthest to the sensors after n ms are greater than the previous x- and y-coordinates respectively.

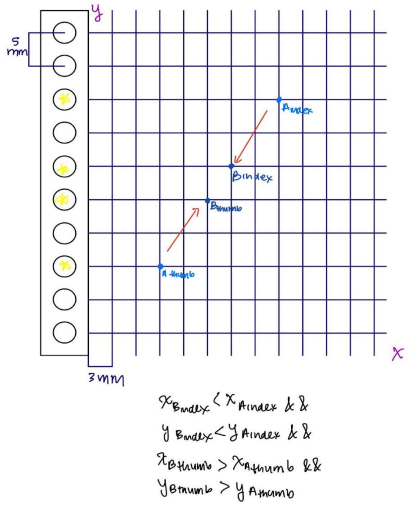

Fig. 10: Zooming in: both x- and y-coordinates of the finger nearest to the sensors after n ms are greater than the previous x- and y-coordinates respectively, and both x- and y-coordinates of the finger farthest to the sensors after n ms are less than the previous x- and y-coordinates respectively.

Fig. 11: Bill Of Materials

Item Name	Part #	Manufacturer	Quantity	Price	Description	Total
Custom PCB	n/a	JLCPCB	1	$4.00	Custom-designed Printed Circuit Board	$4.00
Distance Sensors	VL6180X Time-of-Flight distance sensors	STMicroelectronics	10	$2.95	Proximity sensor and ambient light sensing (ALS) module	$29.50
Microcontroller Board	Particle Photon	Particle	1	$0.00	WiFi-enabled board with STM32 ARM Cortex M3 microcontroller (personal device)	$0.00
Resistors and Wires	10K Ohm	n/a	12	$0.00	10K Ohm Resistors and Wires (scrounged from lab spaces)	$0.00
NVIDIA Jetson Nano	B01	NVIDIA	1	$0.00	GPU-enabled development kit (personal device)	$0.00
Acrylic Sheets	n/a		2	$7.50	Clear plastic sheets to cut trapezoids out of for hologram pyramids	$15.00

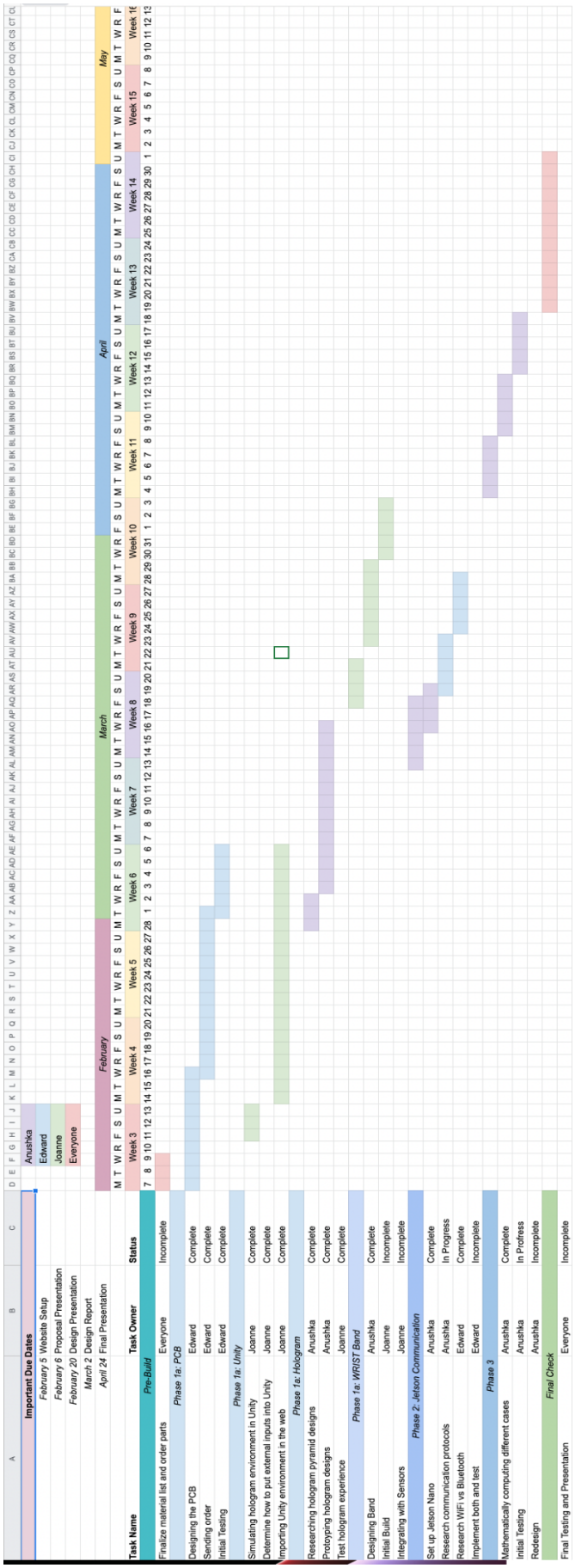

Fig. 12: Tentative Schedule for Capstone