ELECENG 4OI6A FINAL REPORT

UNIDENTIFIED FLYING MOUSE

AN ERGONOMIC COMPUTER MOUSE FOR INDIVIDUALS LACKING HAND DEXTERITY

Instructor:

Dr. Ameer Abdelhadi

Team Members:

Luke West – westl5 – 400310803

Matthew Yu– yum77 – 400322243

Ryan Xu – xur76 – 400309996

George Gill – gillg62 – 400327563

Date: April 21st, 2025

ACKNOWLEDGMENTS

The UFM Team would like to sincerely thank PixArt Imaging Inc. for generously providing us with samples of their proprietary sensors and offering continuous technical support throughout our project.

We are grateful to Dr. Li and Dr. Abdelhadi for their guidance and leadership in facilitating the capstone course and regular meetings.

Lastly, we appreciate Bart Trzynadlowski for his publicly available PAJ7025R2 demo repository on GitHub, which served as a valuable reference during our development process [1].

Contents

1. INTRODUCTION

The computer has become a staple household tool for everyday activities, ranging from education to entertainment. Due to the simplistic approach of input devices, the operation of a computer is second nature for most of the population. However, individuals who suffer from upper-limb amputations or damage centered around the hands can find these traditional input devices almost impossible to use. With computers remaining integral to society, a blank space must be filled for this demographic.

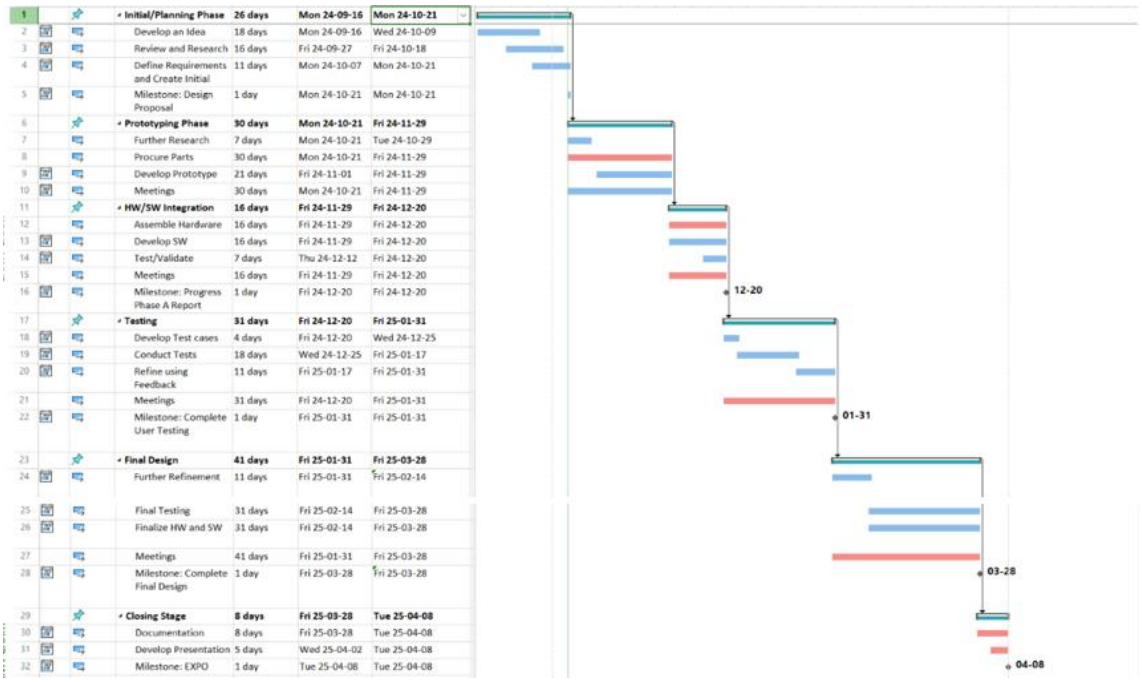
Our project, the Unidentified Flying Mouse (UFM), presents an innovative method of reimagining the conventional mouse into a sleek wristband that accurately detects wrist and arm movements and transforms these commands into digital inputs. It utilizes an IR camera and IR LED for accurate spatial tracking, an IMU for precisely detecting unique wrist movements, and a haptic module for tactile feedback, all operating on an ESP-32 Fire Beetle microcontroller. The product also offers a real-time customization option through a custom GUI to further enhance usability. Allowing users to adjust sensitivity and angles, and switch from left-handed to right-handed mode. The UFM is an efficient alternative to traditional mouse input devices, enabling greater independence through an ergonomic design.

Despite technological advances, the market remains underrepresented with assistive peripherals for individuals with disabilities. Those who tap into this still require physical inputs, like a touch. The UFM addresses this need by providing a non-contact solution that leverages natural wrist movements, redefining physical inputs such as clicking for an intuitive and ergonomic alternative.

Engineers have developed promising solutions in the past, which we have researched to refine our design. For instance, engineering students from De La Salle University developed a similar wireless mouse, as well as a prosthetic hook mouse from the University of Pittsburgh. Analyzing both the benefits and drawbacks of each, we were able to refine our design requirements. It also helped us make strategic decisions to address the shortcomings of these past designs, for example, using IR tracking for the scrolling rather than relying on the IMU only, as the IMU is prone to latency delay, or making the design ambidextrous to increase the demographic reach.

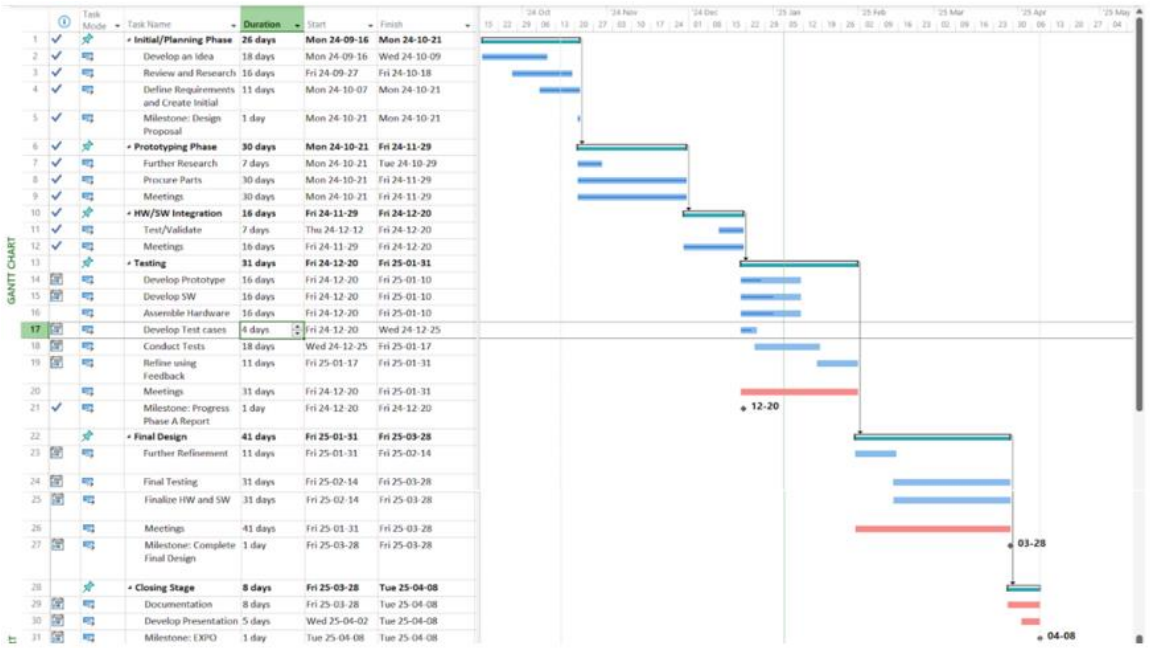
2. PROJECT GOALS, PLANNED APPROACH & EARLY DESIGNS

The Gantt chart below presents the project's progress from Sept 2024 - April 2025, displaying key tasks, and milestones. The project was split into six phases, ranging from initial planning to final design. As the system's development progressed, our timeline changed due to some setbacks. For example, around Dec 2024, we had major issues connecting with the Pixart sensor, due it's unconventional profile. This required us to fabricate a PCB to interface with the sensor, hindering the initial prototype development. In the second half of the timeline, we also had some design changes to the SW, which pushed back the final development. However, to compensate for this delay, we performed testing alongside refining the design as it was developed. The final Gantt chart lays out the entire project timeline, illustrating key changes and challenges that ultimately shaped the final design.



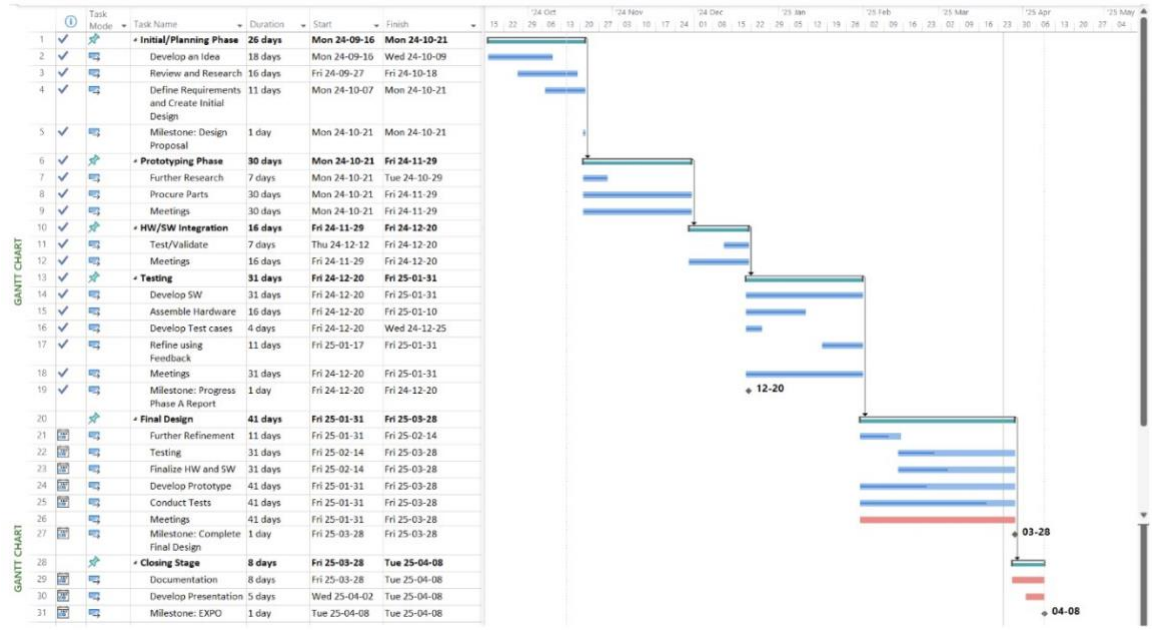
1		Initial/Planning Phase	26 days	Mon 24-09-16	Mon 24-10-21
2		Develop an Idea	18 days	Mon 24-09-16	Wed 24-10-09
3		Review and Research	16 days	Fri 24-09-27	Fri 24-10-18
4		Define Requirements and Create Initial	11 days	Mon 24-10-07	Mon 24-10-21
5		Milestone: Design Proposal	1 day	Mon 24-10-21	Mon 24-10-21
6		Prototyping Phase	30 days	Mon 24-10-21	Fri 24-11-29
7		Further Research	7 days	Mon 24-10-21	Tue 24-10-29
8		Procure Parts	30 days	Mon 24-10-21	Fri 24-11-29
9		Develop Prototype	21 days	Fri 24-11-01	Fri 24-11-29
10		Meetings	30 days	Mon 24-10-21	Fri 24-11-29
11		HW/SW Integration	16 days	Fri 24-11-29	Fri 24-12-20
12		Assemble Hardware	16 days	Fri 24-11-29	Fri 24-12-20
13		Develop SW	16 days	Fri 24-11-29	Fri 24-12-20
14		Test/Validate	7 days	Thu 24-12-12	Fri 24-12-20
15		Meetings	16 days	Fri 24-11-29	Fri 24-12-20
16		Milestone: Progress Phase A Report	1 day	Fri 24-12-20	Fri 24-12-20
17		Testing	31 days	Fri 24-12-20	Fri 25-01-31
18		Develop Test cases	4 days	Fri 24-12-20	Wed 24-12-25
19		Conduct Tests	18 days	Wed 24-12-25	Fri 25-01-17
20		Refine using Feedback	11 days	Fri 25-01-17	Fri 25-01-31
21		Meetings	31 days	Fri 24-12-20	Fri 25-01-31
22		Milestone: Complete User Testing	1 day	Fri 25-01-31	Fri 25-01-31
23		Final Design	41 days	Fri 25-01-31	Fri 25-03-28
24		Further Refinement	11 days	Fri 25-01-31	Fri 25-02-14
25		Final Testing	31 days	Fri 25-02-14	Fri 25-03-28
26		Finalize HW and SW	31 days	Fri 25-02-14	Fri 25-03-28
27		Meetings	41 days	Fri 25-01-31	Fri 25-03-28
28		Milestone: Complete Final Design	1 day	Fri 25-03-28	Fri 25-03-28
29		Closing Stage	8 days	Fri 25-03-28	Tue 25-04-08
30		Documentation	8 days	Fri 25-03-28	Tue 25-04-08
31		Develop Presentation	5 days	Wed 25-04-02	Tue 25-04-08
32		Milestone: EXPO	1 day	Tue 25-04-08	Tue 25-04-08

Figure 1: Initial Gantt Chart



	Task Mode	Task Name	Duration	Start	Finish
1		**Initial/Planning Phase**	**26 days**	**Mon 24-09-16**	**Mon 24-10-21**
2		Develop an Idea	18 days	Mon 24-09-16	Wed 24-10-09
3		Review and Research	16 days	Fri 24-09-27	Fri 24-10-18
4		Define Requirements and Create Initial	11 days	Mon 24-10-07	Mon 24-10-21
5		Milestone: Design Proposal	1 day	Mon 24-10-21	Mon 24-10-21
6		**Prototyping Phase**	**30 days**	**Mon 24-10-21**	**Fri 24-11-29**
7		Further Research	7 days	Mon 24-10-21	Tue 24-10-29
8		Procure Parts	30 days	Mon 24-10-21	Fri 24-11-29
9		Meetings	30 days	Mon 24-10-21	Fri 24-11-29
10		**HW/SW Integration**	**16 days**	**Fri 24-11-29**	**Fri 24-12-20**
11		Test/Validate	7 days	Thu 24-12-12	Fri 24-12-20
12		Meetings	16 days	Fri 24-11-29	Fri 24-12-20
13		**Testing**	**31 days**	**Fri 24-12-20**	**Fri 25-01-31**
14		Develop Prototype	16 days	Fri 24-12-20	Fri 25-01-10
15		Develop SW	16 days	Fri 24-12-20	Fri 25-01-10
16		Assemble Hardware	16 days	Fri 24-12-20	Fri 25-01-10
17		Develop Test cases	4 days	Fri 24-12-20	Wed 24-12-25
18		Conduct Tests	18 days	Wed 24-12-25	Fri 25-01-17
19		Refine using Feedback	11 days	Fri 25-01-17	Fri 25-01-31
20		Meetings	31 days	Fri 24-12-20	Fri 25-01-31
21		Milestone: Progress Phase A Report	1 day	Fri 24-12-20	Fri 24-12-20
22		**Final Design**	**41 days**	**Fri 25-01-31**	**Fri 25-03-28**
23		Further Refinement	11 days	Fri 25-01-31	Fri 25-02-14
24		Final Testing	31 days	Fri 25-02-14	Fri 25-03-28
25		Finalize HW and SW	31 days	Fri 25-02-14	Fri 25-03-28
26		Meetings	41 days	Fri 25-01-31	Fri 25-03-28
27		Milestone: Complete Final Design	1 day	Fri 25-03-28	Fri 25-03-28
28		**Closing Stage**	**8 days**	**Fri 25-03-28**	**Tue 25-04-08**
29		Documentation	8 days	Fri 25-03-28	Tue 25-04-08
30		Develop Presentation	5 days	Wed 25-04-02	Tue 25-04-08
31		Milestone: EXPO	1 day	Tue 25-04-08	Tue 25-04-08

Figure 2: Phase A Gantt Chart



	Task Mode	Task Name	Duration	Start	Finish
1		**Initial/Planning Phase**	**26 days**	**Mon 24-09-16**	**Mon 24-10-21**
2		Develop an Idea	18 days	Mon 24-09-16	Wed 24-10-09
3		Review and Research	16 days	Fri 24-09-27	Fri 24-10-18
4		Define Requirements and Create Initial Design	11 days	Mon 24-10-07	Mon 24-10-21
5		Milestone: Design Proposal	1 day	Mon 24-10-21	Mon 24-10-21
6		**Prototyping Phase**	**30 days**	**Mon 24-10-21**	**Fri 24-11-29**
7		Further Research	7 days	Mon 24-10-21	Tue 24-10-29
8		Procure Parts	30 days	Mon 24-10-21	Fri 24-11-29
9		Meetings	30 days	Mon 24-10-21	Fri 24-11-29
10		**HW/SW Integration**	**16 days**	**Fri 24-11-29**	**Fri 24-12-20**
11		Test/Validate	7 days	Thu 24-12-12	Fri 24-12-20
12		Meetings	16 days	Fri 24-11-29	Fri 24-12-20
13		**Testing**	**31 days**	**Fri 24-12-20**	**Fri 25-01-31**
14		Develop SW	31 days	Fri 24-12-20	Fri 25-01-31
15		Assemble Hardware	16 days	Fri 24-12-20	Fri 25-01-10
16		Develop Test cases	4 days	Fri 24-12-20	Wed 24-12-25
17		Refine using Feedback	11 days	Fri 25-01-17	Fri 25-01-31
18		Meetings	31 days	Fri 24-12-20	Fri 25-01-31
19		Milestone: Progress Phase A Report	1 day	Fri 24-12-20	Fri 24-12-20
20		**Final Design**	**41 days**	**Fri 25-01-31**	**Fri 25-03-28**
21		Further Refinement	11 days	Fri 25-01-31	Fri 25-02-14
22		Testing	31 days	Fri 25-02-14	Fri 25-03-28
23		Finalize HW and SW	31 days	Fri 25-02-14	Fri 25-03-28
24		Develop Prototype	41 days	Fri 25-01-31	Fri 25-03-28
25		Conduct Tests	41 days	Fri 25-01-31	Fri 25-03-28
26		Meetings	41 days	Fri 25-01-31	Fri 25-03-28
27		Milestone: Complete Final Design	1 day	Fri 25-03-28	Fri 25-03-28
28		**Closing Stage**	**8 days**	**Fri 25-03-28**	**Tue 25-04-08**
29		Documentation	8 days	Fri 25-03-28	Tue 25-04-08
30		Develop Presentation	5 days	Wed 25-04-02	Tue 25-04-08
31		Milestone: EXPO	1 day	Tue 25-04-08	Tue 25-04-08

Figure 3: Phase B Gantt Chart

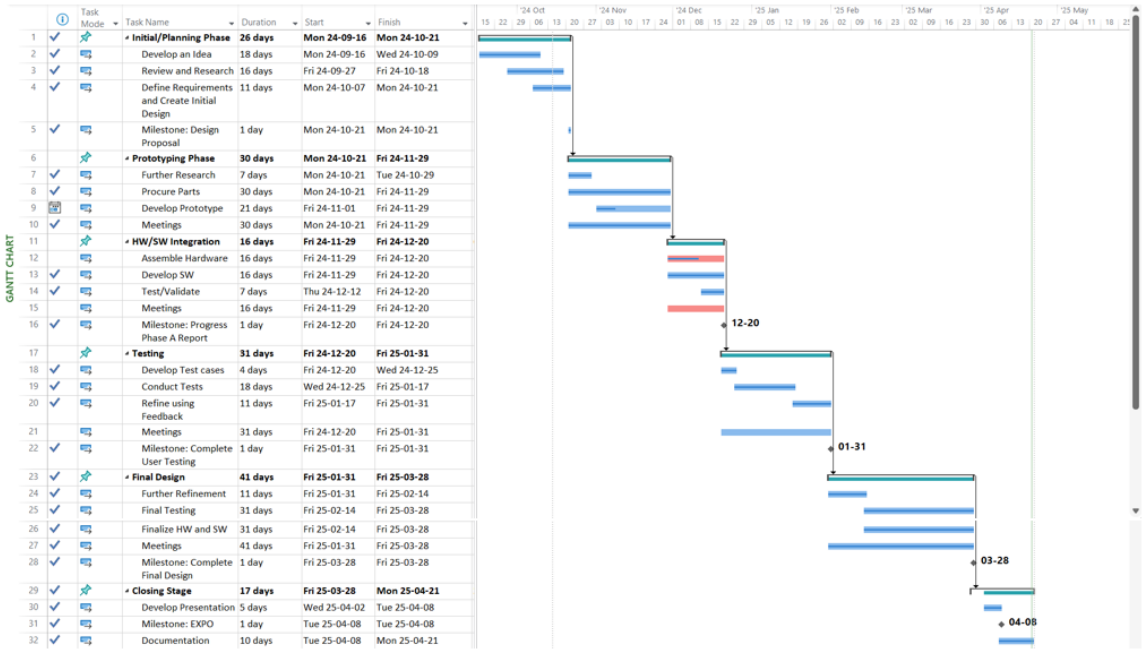


Figure 4: Final Gantt Chart

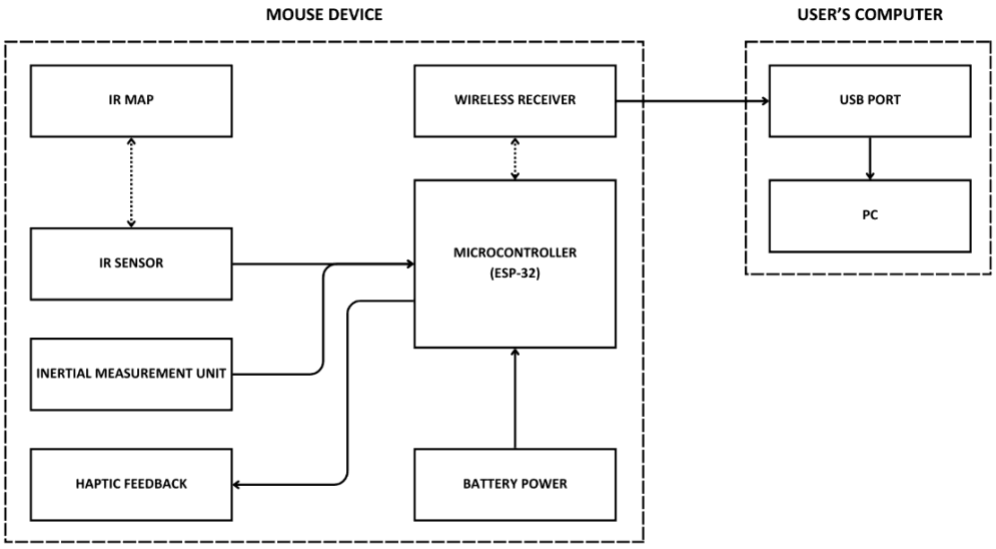


Figure 5: Initial Block Diagram

Looking at the first proposed block diagram, the general design stayed very similar. There were a few minor changes such as the method for mouse cursor movement being much less complicated and using Bluetooth rather than an external wireless receiver.

The electrical schematic designed on KiCAD outlines the configuration of the design, with a microcontroller and other components. This schematic has seen three iterations, from the initial development in Oct 2024 to the final design in March 2025. The preliminary schematic is split into two modules: wrist-mounted and table-mounted. The wrist-mounted enclosures hold the Pixart sensor, the microcontroller, two haptic motor drives and motors, IMU, and a battery. The microcontroller interfaces all the components, acting as a central node that communicates through various GPIO pins. The table-mounted is simply an LED array powered by an external power source and controlled by a switch.

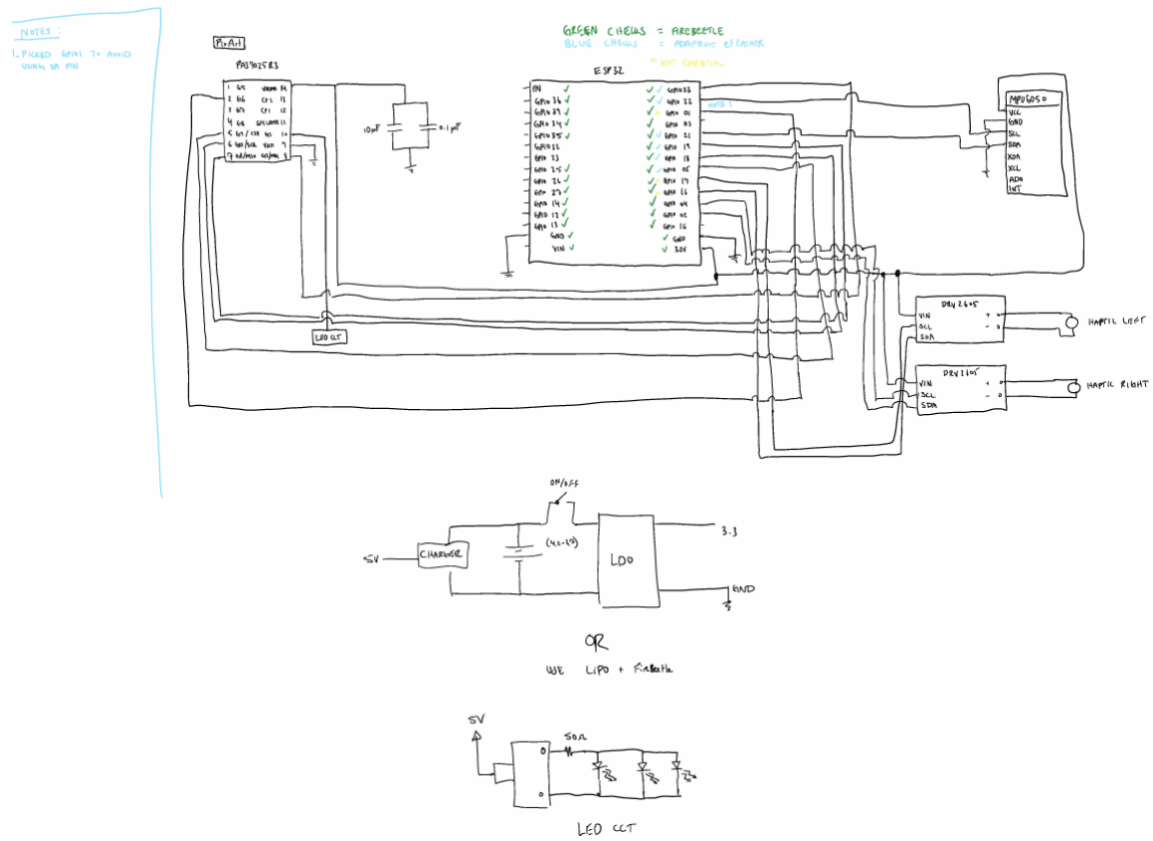


Figure 6: Initial Schematic

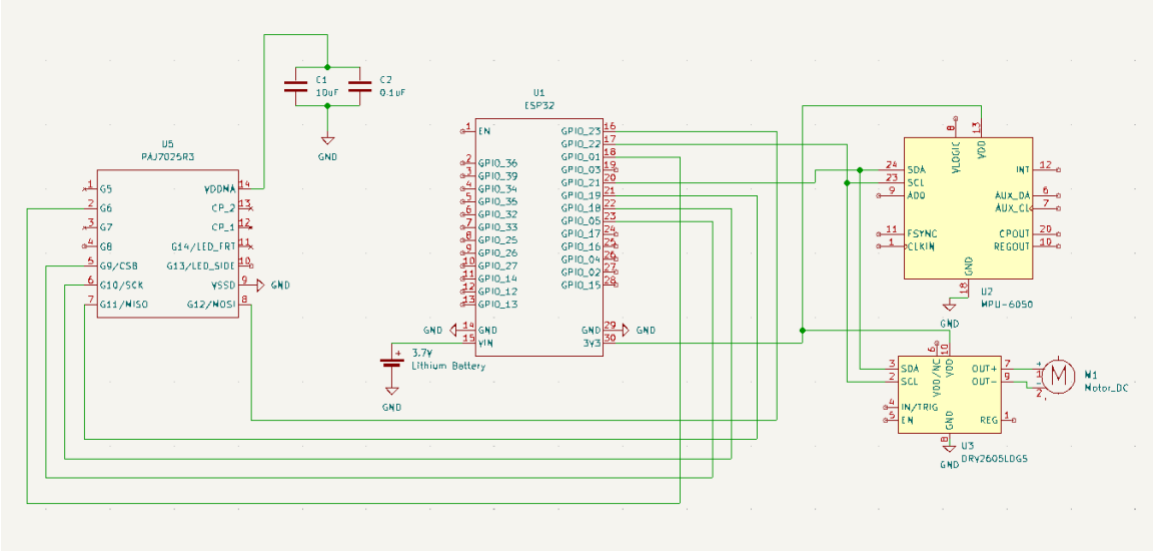


Figure 7: Phase A Main Schematic

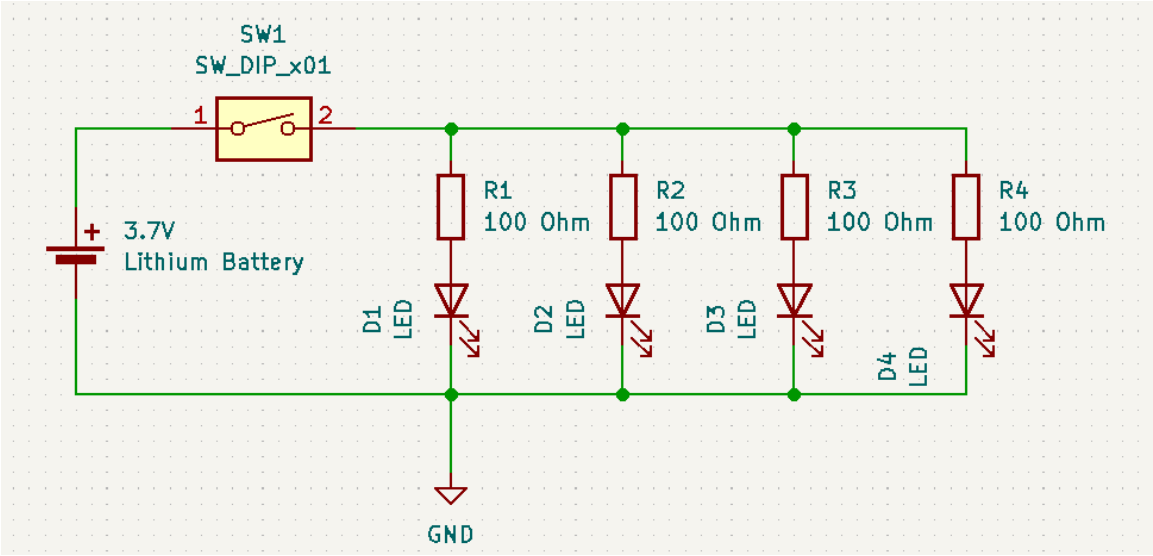


Figure 8: Phase A LED Circuit

The final schematic alters both modules slightly, as seen in the Figures below. The wrist-mounted aspect finalizes the microcontroller as the FireBeetle 2 ESP-32E due to its onboard voltage regulator. Additionally, we removed a motor driver and motor due to redundancy, which also lessened the size and weight. The table-mounted design was simplified to a singular LED, 100-ohm resistor, a switch, and a double A battery, as we found a single LED was sufficient for our tracking application.

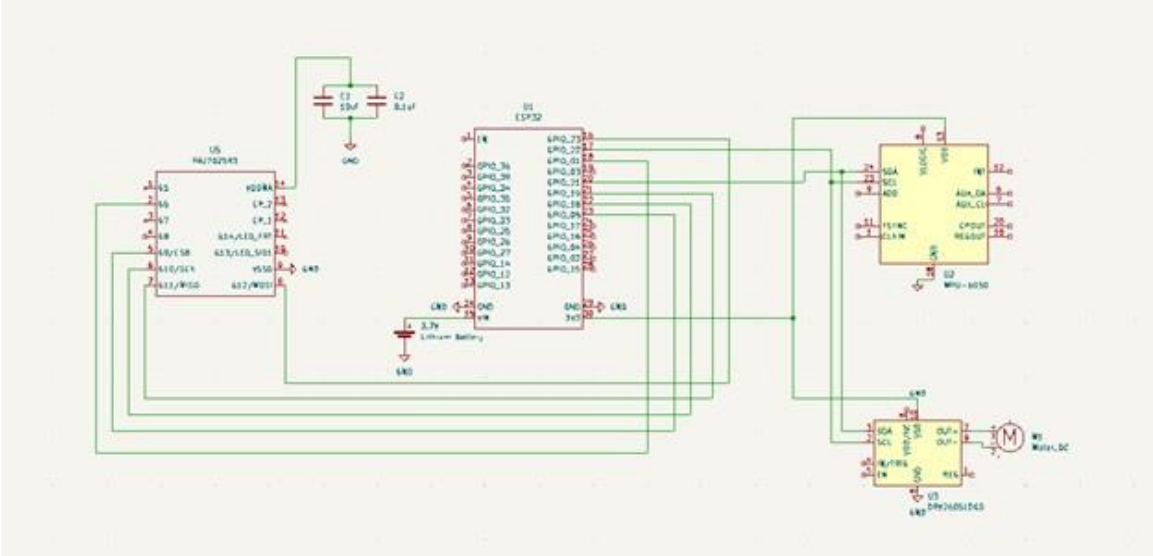


Figure 9: Final Main Schematic

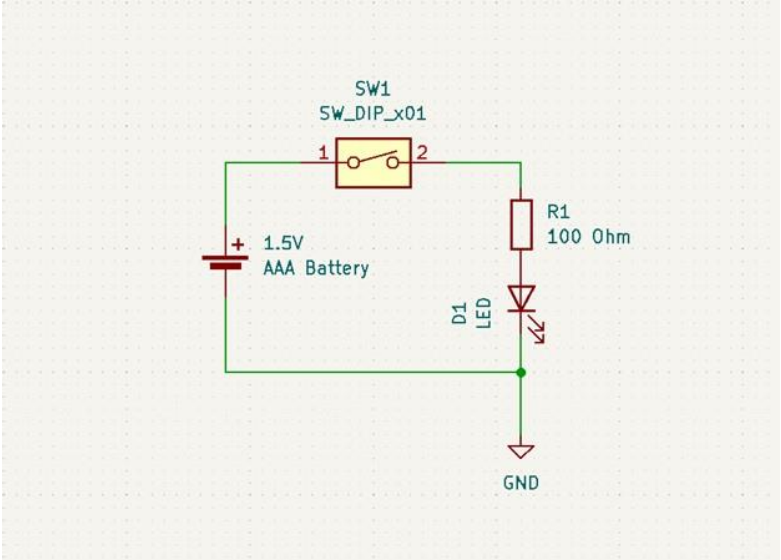


Figure 10: Final LED Circuits

2.1 Initial Goals and Markers

The goal of the project was to create a new and ergonomic computer input device. Firstly, we predefined requirements: This device was to be lightweight, compact, and operate hands-free, requiring no physical input. Additionally, we wanted to ensure the system was easy to set up and ready to use out of the box. We also wanted to not limit the user's natural wrist movements, avoiding any body joints. For the software aspect, the device needed to provide reliable motion tracking with little to no time lag, along with the basic features of a mouse, such as clicking. We also wanted to promote a standalone system, operational,

using a battery for around 8 hours, without requiring wired connections. These core requirements were referred to multiple times for the decisions that drove the design process.

2.2 Mouse Cursor Movement & Functionality

The design behind each mouse function proved challenging, and each went through multiple iterations over the course of the project. In the initial stages, our team planned a much more complex method to obtain mouse cursor movement and other mouse functions than what was used in the final product. The idea was to perform real-time monocular tracking of the user's wrist in 3D space using multiple IR LED tracking points. The advantage of this in-depth method is that wrist rotation could be isolated from arm movement which is co-planar to the desk. This would allow gesture recognition without needing to pause and turn off input from the IR. In theory, this would more closely match the performance of a regular computer mouse and provide a much better user experience.

2.2.1 3D Monocular Tracking Working Principle

The two main hardware components used for 3D tracking would have been the PixArt PAJ7025R3 MOT (multiple objects tracking) sensor, and a cluster of multiple 850nm IR LEDs in a known dimensional configuration. The MOT sensor contains a camera with an 800-900nm IR pass filter as well as built in hardware to track up to 16 objects. It assigns a number to each object and returns the coordinates of each point to the ESP-32 using SPI.

To obtain cursor motion, the 3D coordinates of the sensor relative to the cluster of LEDs must be obtained using only the 2D point coordinates obtained from the MOT sensor, and the intrinsic parameters of the sensor's camera. To accomplish this, multiple approaches were researched and explored. There are two projects documented on GitHub using the same MOT sensor which were used as main inspirations for this project: Retrosphere, which uses a stereo setup built into glasses to track reflective spheres for various VR and AR applications, and a 6dof demo by Bart Tryznadlowski which uses a monocular setup to model motion of a real ping pong paddle in 3D [1][8].

The main advantage of a stereo setup is that depth information is easier to obtain and more accurate, leading to more accuracy in the 3D tracking. In our project, a monocular approach was explored more thoroughly, because the hardware setup is less complex and simpler to mount to a human wrist. Rather than using the known distance between two cameras, the known dimensions of a cluster of LEDs is used to determine camera pose in 3D. There are many approaches for achieving this that were explored, but eventually, the PnP (Perspective-n-Pose) solver in OpenCV was chosen [9].

This algorithm takes a set of known world coordinates (in our case the point cluster), a set of 2D points as seen in the camera, and the camera's intrinsic matrix and distortion coefficients. Then, it computes the rotation and translation required to correspond the 2D

camera points to the 3D world points using the Levenberg-Marquardt algorithm. This can then be used to recover the camera's pose in the world frame. The camera's pose and position in space could then be analyzed for specific gestures to perform cursor motion or other mouse functions.

2.2.2 Python Simulation of Cursor Motion Using PnP in OpenCV

To test the viability of 3D monocular tracking for the project, a simulation was written in Python using OpenCV. The camera was simulated using an intrinsic matrix created from the datasheet parameters of the PixArt PAJ7025R3. A 3D point cluster is created, and the points are projected onto a 2D plane using the $cv.projectPoints()$ function. Then, the camera pose is recovered using OpenCV's PnP (Perspective-n-Point) solver. The x and y coordinates of the camera position are then used for cursor movement, and a simulated cursor is plotted on a 1920x1080 screen. As an added feature, a function adding random gaussian noise to the camera points was also created to simulate real word noise and monitor its effects, but for the plots shown below, noise was neglected to show a clean result.

Various point clusters were tested for accuracy of recovering the camera pose. At first, a four-point rectangular pattern like that in Bart Tryznadlowski's project was used [1]. Although the points being coplanar does decrease depth information of the object, this method offered the advantage of a simpler setup that involving less LED occlusions (when one LED covers another LED) to consider. Because of this, this method was the first method explored in the physical implementation, as it was simplest to test. An example of a simulation using the rectangular structure is shown below.

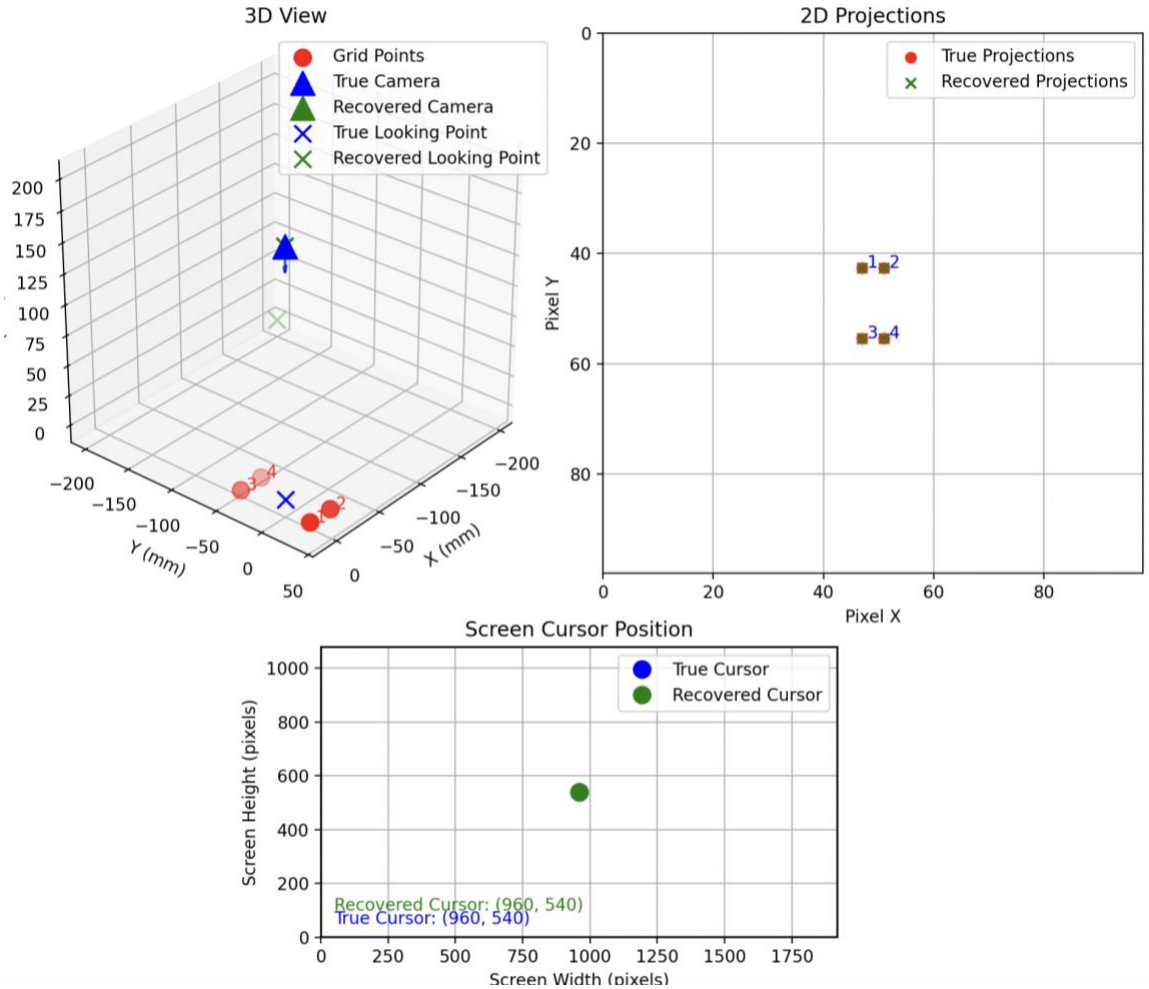


Figure 11: Simulated Monocular Tracking Using a Rectangular LED Configuration

To improve accuracy, a six LED structure was also generated with a coplanar triangular base, and varying LED heights and locations. A nine LED structure was also explored, using 5 coplanar points as the base and 4 other points with varied height, however the six LED configuration is preferred as the build complexity is reduced.

Varying the heights of the LEDs in the cluster helps provide depth information, but also decreases the range of motion, as occlusion occurs more frequently. Another thing to consider with the LED structure is that at lower height differences, it is more difficult to physically build, since our budget and build quality requires high error tolerance. Shown below is an example where the camera is directly above the Six LED structure:

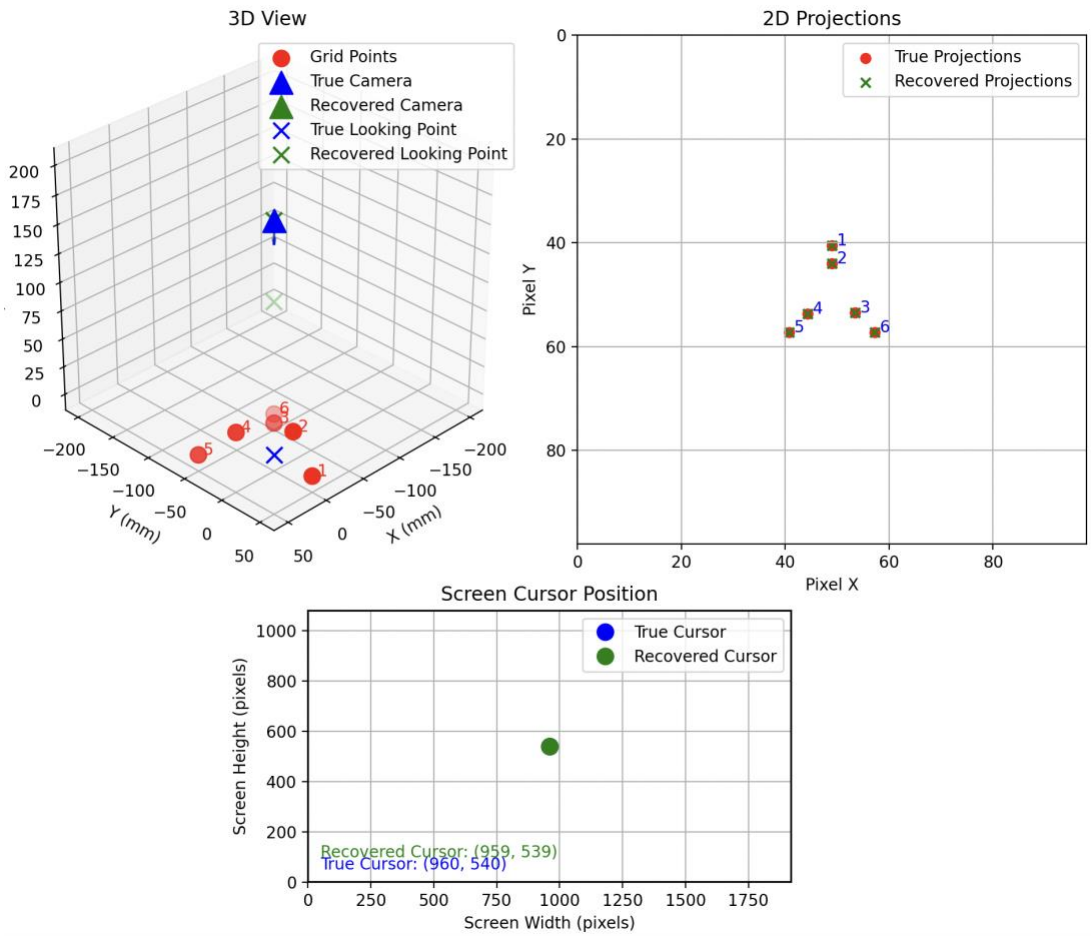


Figure 12: Simulated Monocular Tracking Using a Six LED Configuration

2.2.3 Mouse Function Rework

After simulating in Python, our group progressed to the hardware testing stage. Our first step was to receive the object coordinate data from the PixArt IR sensor using SPI. Firmware originally written by Bart Tryznadlowski in Arduino for his 6 degree of freedom tracking demo project was modified for our own purposes to interface with the sensor [1]. Using the Visual Studio Code Extension Platform IO, the code was properly formatted to be flashed on the ESP-32 microcontroller. It was thoroughly cross-examined with the data sheet, commented in detail and modified to ensure proper interfacing with the sensor for our purposes.

After successfully sending serial data to the PC, the next step in achieving mouse function was creating a temporary Python script to visualize the LEDs for better testing. The script plotted the coordinates received in real-time, providing an easy way to get a feel for the sensor's field of view and range. An example of this initial test setup is shown below.

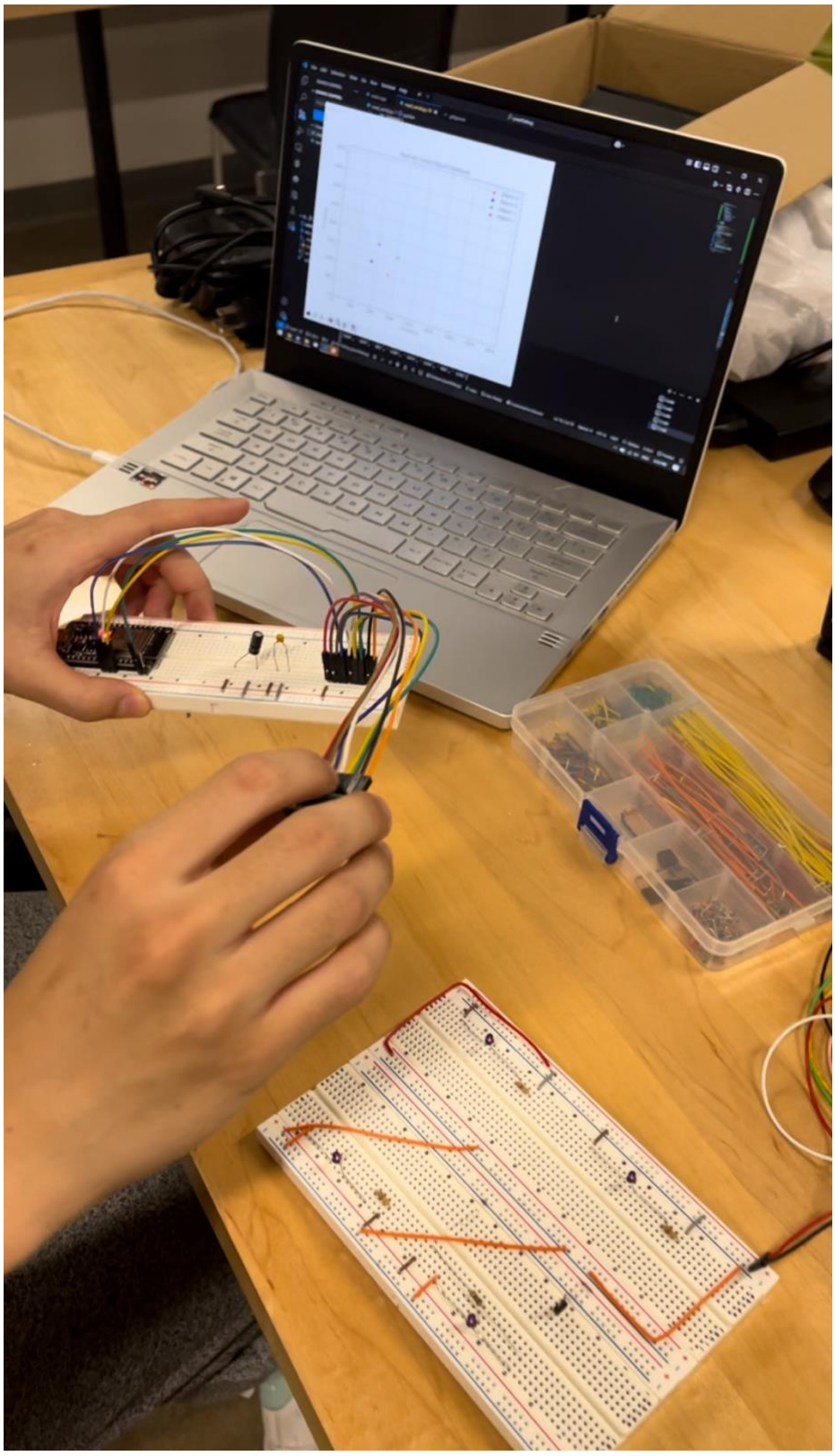


Figure 13: Python Visualization Test of PixArt Sensor Data

Since this visualizer was also written in Python, the simulation code was easily integrated into the visualizer, and attempts were made to plot a mouse cursor along with the LED points. It was at this stage that our group came across some major roadblocks which radically changed the design of the project and its working principle. After identifying our biggest roadblocks, we came up with a plan to solve them. This new plan and its final implementation are covered in section 3, and details behind each problem and how we decided to solve them is covered in section 4.

2.3 Software Integration

The initial integration called for embedded software on the ESP-32-WROOM. This meant we needed a platform and the correct toolchain to interact with the micro controller. PlatformIO was selected due to its high versatility and ease of use with large amounts of existing board data, and code analysis tools. Methodologies were to be implemented in C++ using the principles discussed above in section 2.2.1. We also planned to implement BLE and mouse functionalities using existing open-source libraries with the help of PlatformIO. The nature of this implementation meant that the final product will be a out-of-the-box experience for the user, requiring no extra installation of software on their computer. The device will function as a traditional HID mouse device from the perspective of the user's device regardless of platform or OS.

2.4 Physical Implementation

For the physical design, we first predefined some key requirements that the design should meet. These included ensuring the design is comfortable, secure, lightweight, and safe. These requirements drove each design decision to ensure the most optimal configuration. For example, the most important factor was comfort, as the design should account for prolonged use without disrupting natural wrist movements. Additionally, the size and weight of the design were considered. The wristband should accommodate wrist sizes of 14 - 22 cm and be securely mounted to the size of the user's wrist. We aimed to constrain the total weight to less than 150 grams, as the typical watch is around 120 grams.

The physical design was created in two major segments: the wristband itself and the enclosures that would securely house the components.

Preliminary wristband designs envisioned a clamping method similar to an Apple Watch. This method provides a secure fit and is universally recognized. The initial enclosure design was a simple two-piece assembly fastened using M2 screws and heatsinks present in both the primary and camera enclosures. In order to minimize the footprint of the main enclosure, the ESP32 was stacked on top of the other components, resting on four pillars. The IMU, motor, and driver rest below in predefined locations. There are also multiple holes for connecting to the microcontroller and a hole for the motor to contact the user's skin directly. The initial camera enclosure held the first iteration PCB, using friction to

hold the PCB securely in the two-box assembly. Lastly, the battery was positioned on the lid of the main enclosure, sitting on top of the rest of the components.

For the table-mounted components, a simple box was created for the standalone LEDs, with four holes set at 100mm apart.

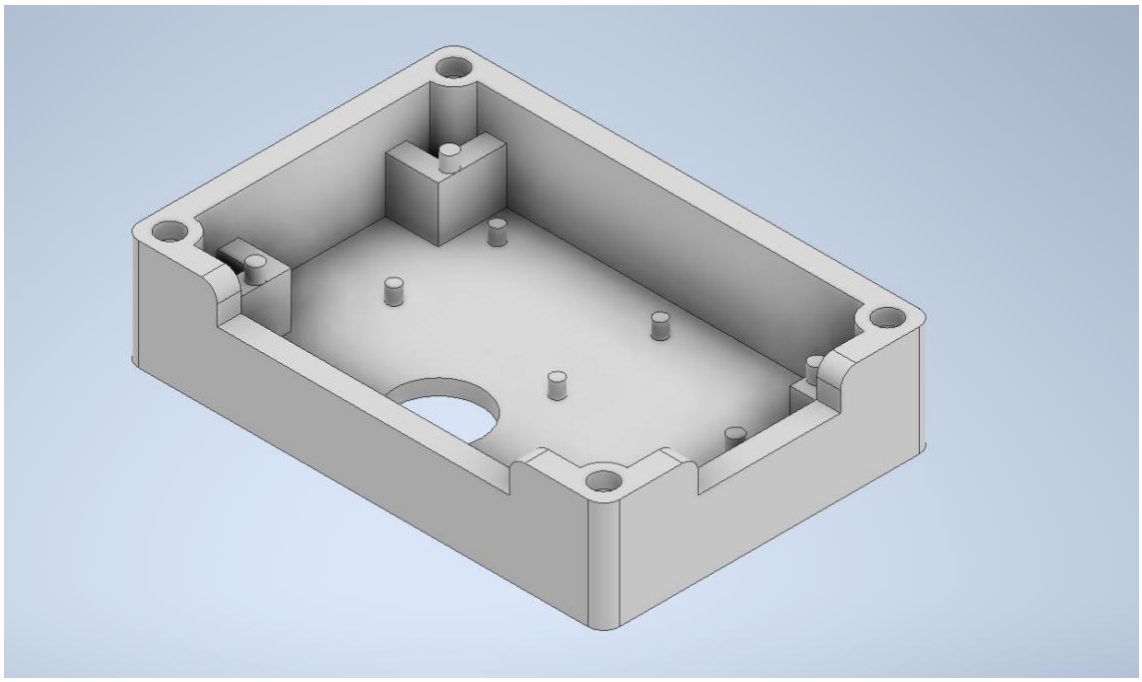


Figure 14: CAD of Wrist Enclosure

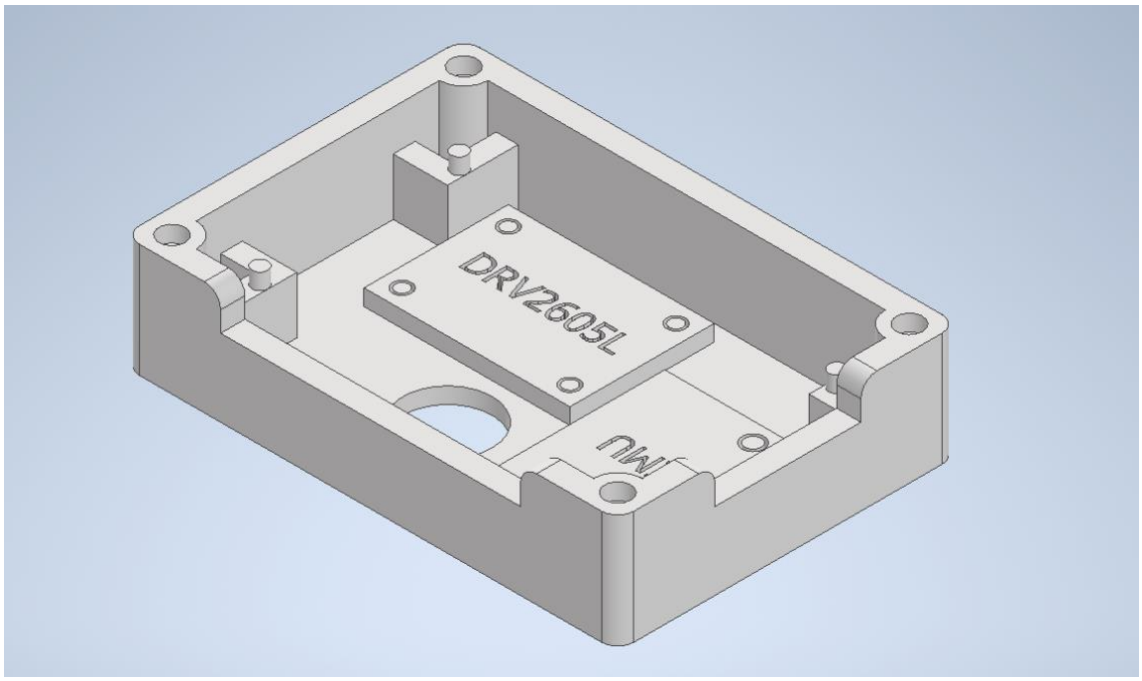


Figure 15: Initial Wrist Enclosure with IMU and Haptic Motor Driver

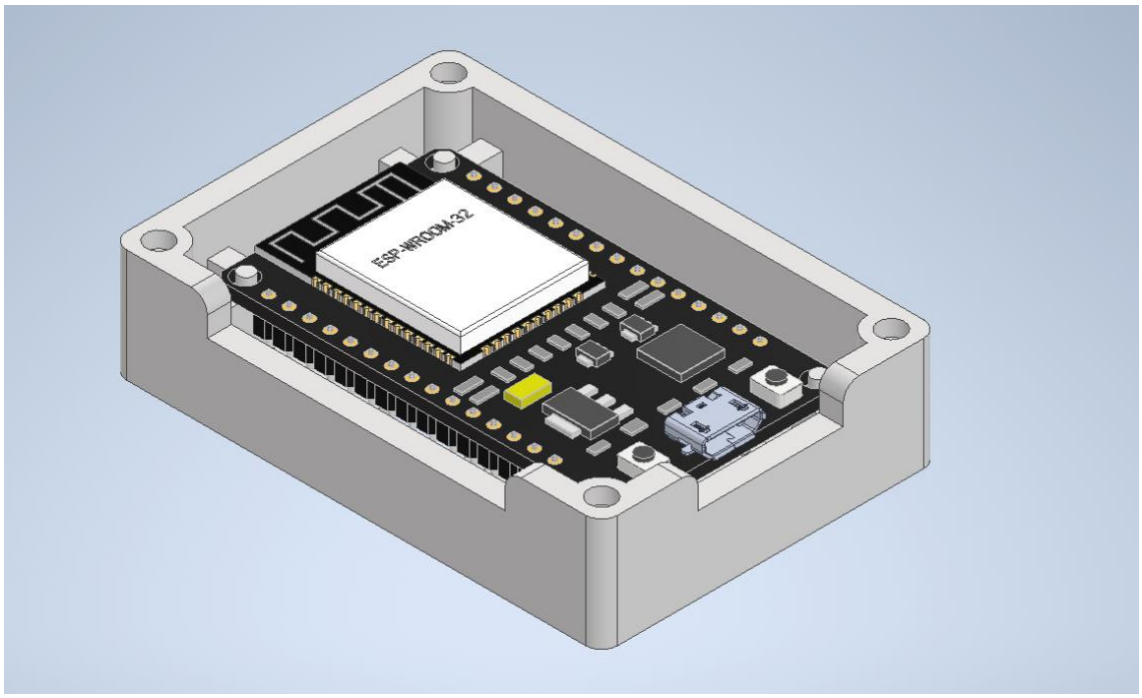


Figure 16: Initial Wrist Enclosure with IMU, Haptic Motor Driver & ESP-32

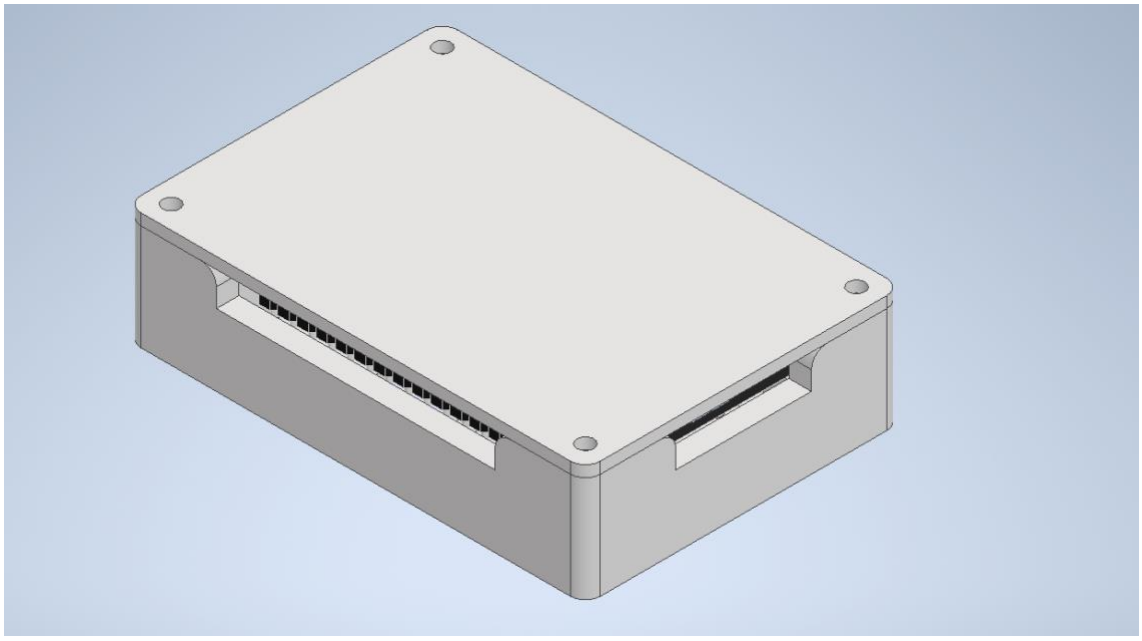


Figure 17: Fully Assembled Initial Wrist Enclosure

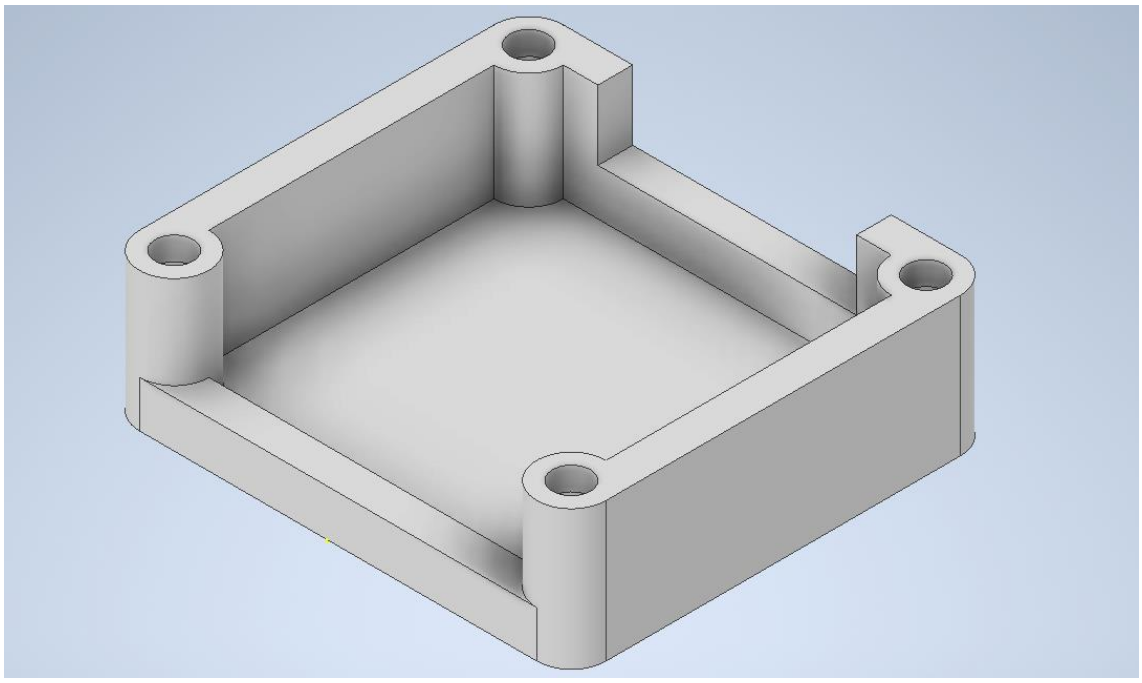


Figure 18: Initial CAD of Camera PCB Enclosure

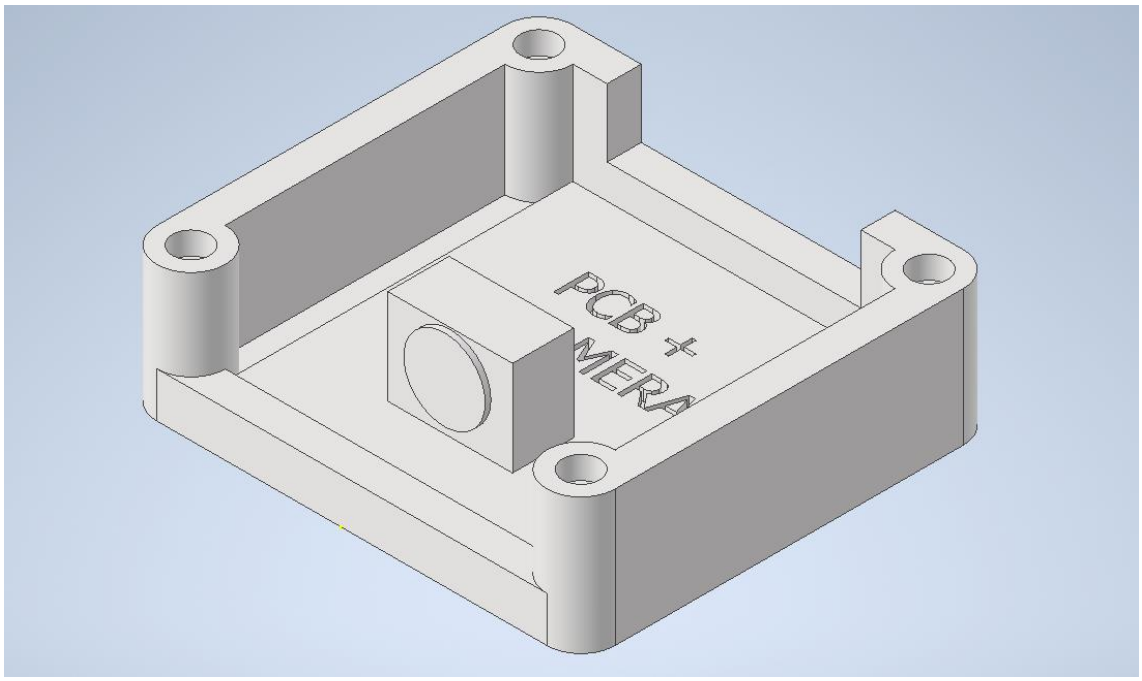


Figure 19: Initial Camera Enclosure with PCB

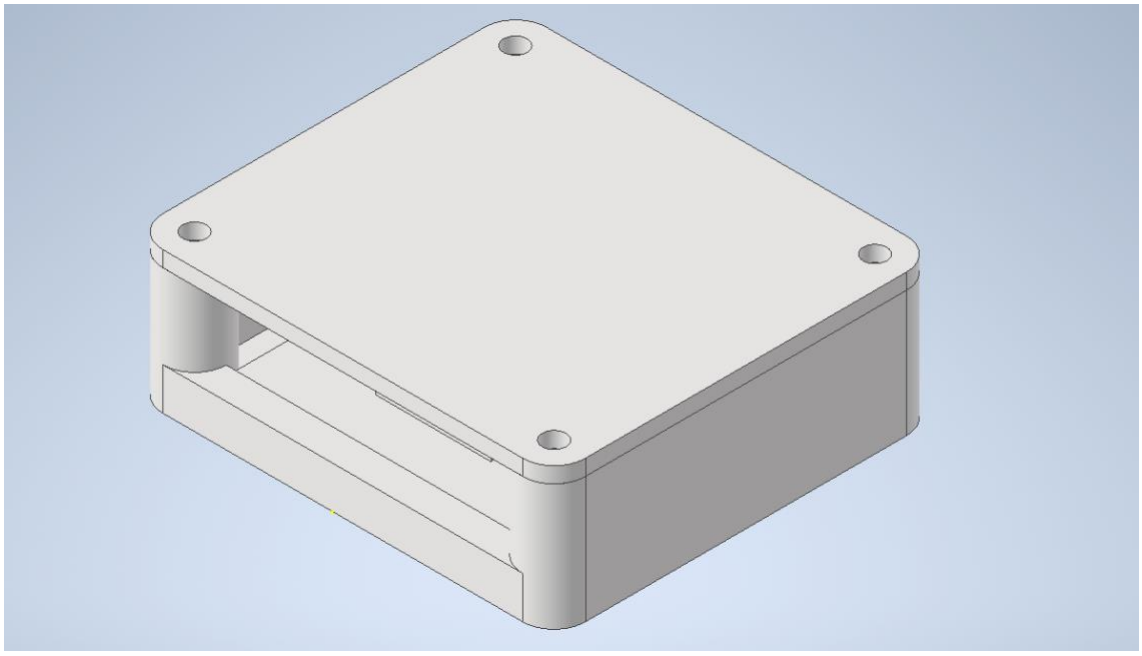


Figure 20: Fully Assembled Initial Camera Enclosure

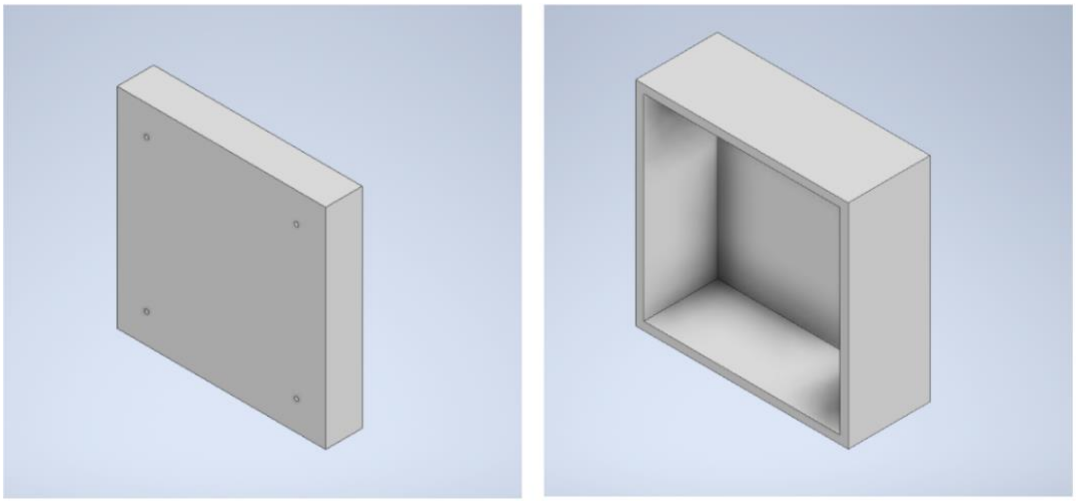


Figure 21: CAD of IR LED Circuit Enclosure (Base on Right, Lid on Left)

After careful consideration, this design was altered for various reasons. Firstly, the wristband was changed to a compression sleeve due to its multiple benefits. Since the compression sleeve can be easily mounted using a single hand. Additionally, the compression sleeve is very secure, so any wrist movement will not disrupt the sensors and components. We also incorporated velcro straps for more versatility on the sleeve. We found that having the enclosures permanently bonded to the wristband eliminated the flexible profile of the compression sleeve and instead became very rigid. The velcro also offers the user more freedom to choose the locations of where the enclosures are to be mounted.

The enclosures also saw significant alterations due to the newly fabricated PCBs. For the main enclosure, we followed the same idea of stacking the components to minimize the footprint, but rather than using friction to secure the components, M2 fasteners and heat sinks were leveraged inside to mount the PCB to the floor of the enclosure, heavily decreasing the risk of the components being disrupted while in use. Similarly, M2 fasteners were utilized again in the camera enclosure to mount the camera PCB securely. Due to the PCB, the enclosure sizes were also drastically reduced to a more compact option. Next, the placement of the battery was repositioned. The original position on the lid of the main enclosure was suboptimal, as the battery was extensive and thus increased the width of the design. This also made all of the weight centred at a single point. Instead, to balance the weight, we decided to fabricate a third enclosure to encase the battery, which is located on the opposite side of the camera. This provides much more balance and shifts the weight to increase the ergonomic ability.

The LED circuit box was reduced because a single LED was needed. Following the same approach as a two-piece box assembly.

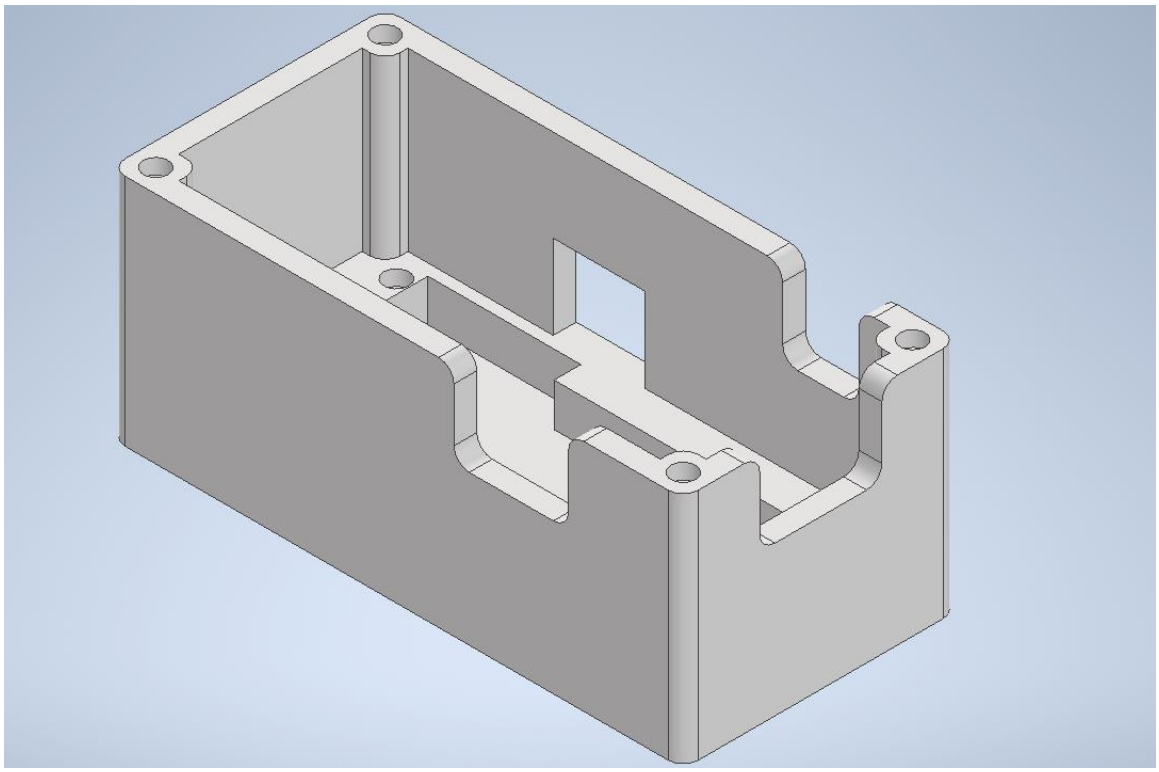


Figure 22: Final CAD of Wrist Enclosure

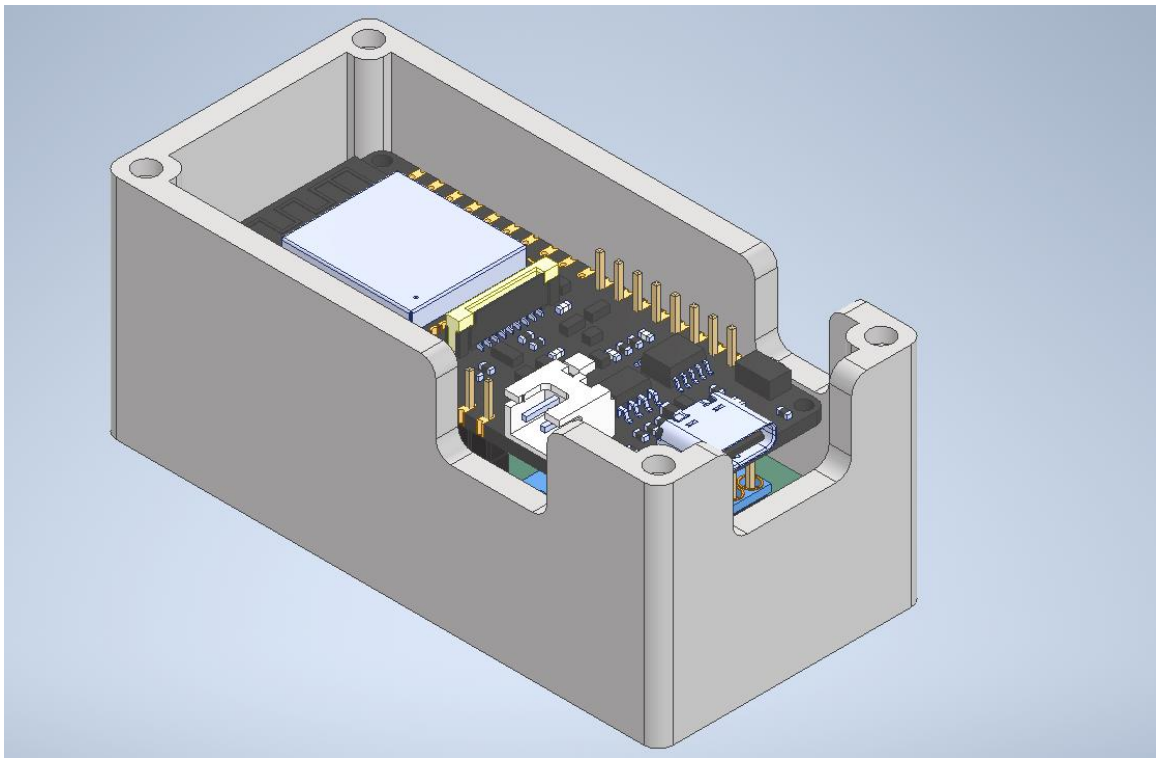


Figure 23: Final CAD of Wrist Enclosure with Components

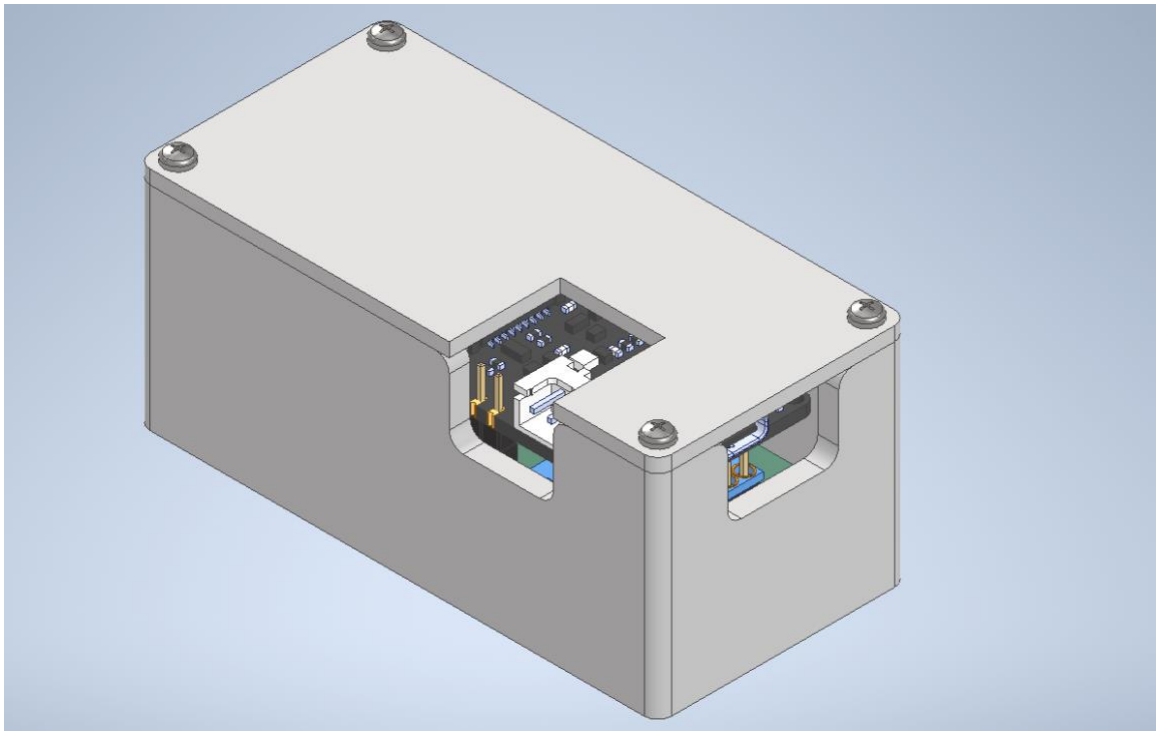


Figure 24: Final CAD of Wrist Enclosure Fully Assembled

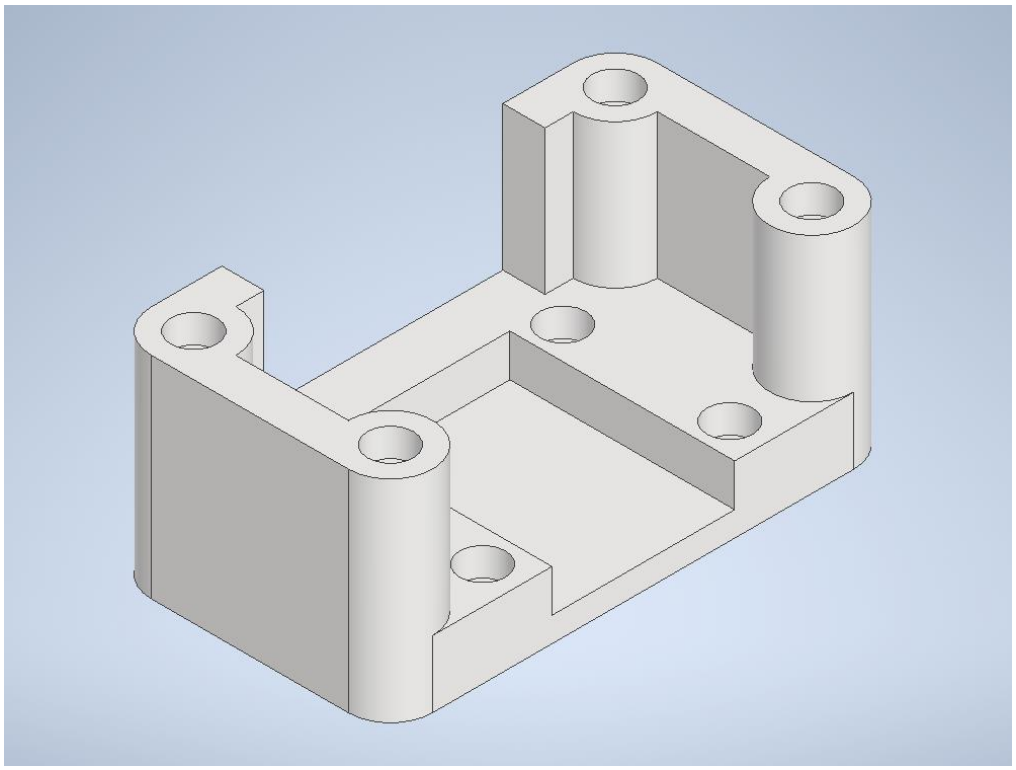


Figure 25: Final CAD of Camera Enclosure

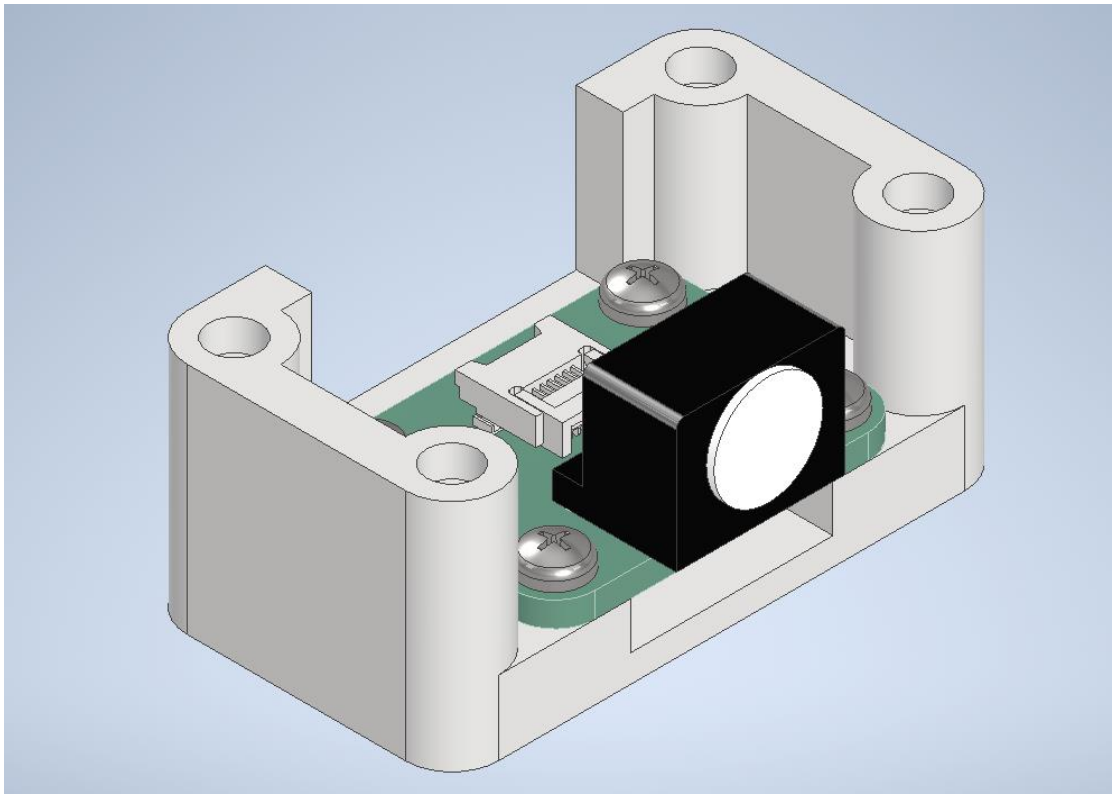


Figure 26: Final CAD of Camera Enclosure with Camera PCB

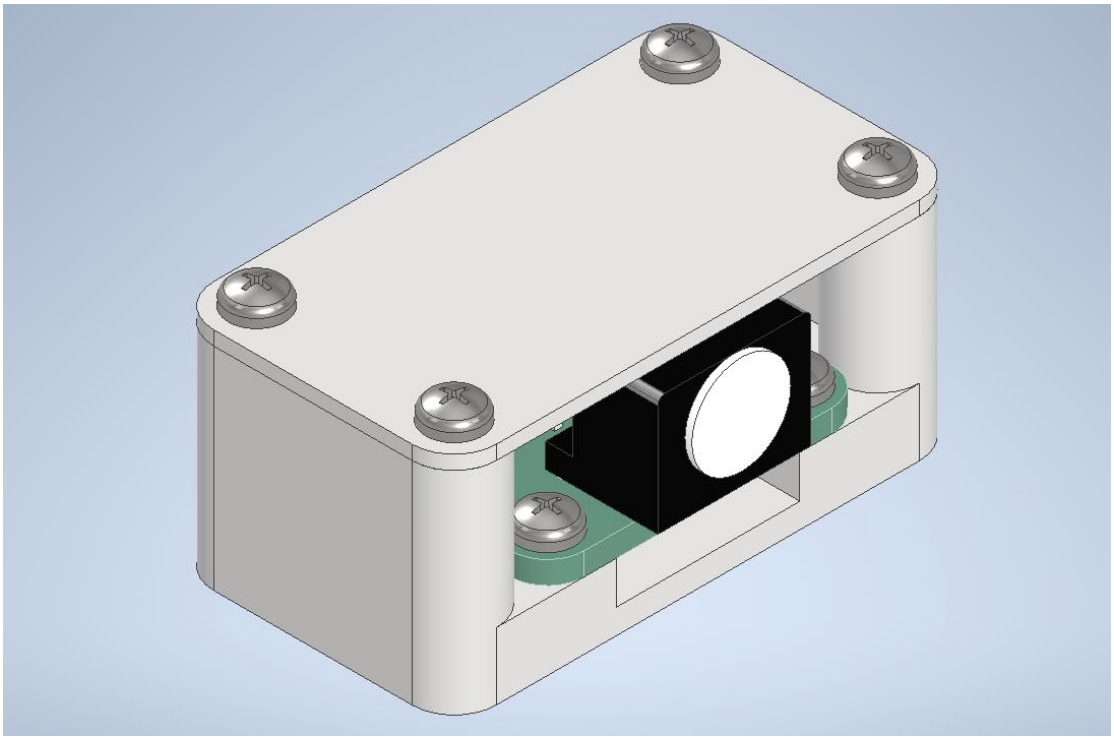


Figure 27: Final CAD of Camera Enclosure Fully Assembled

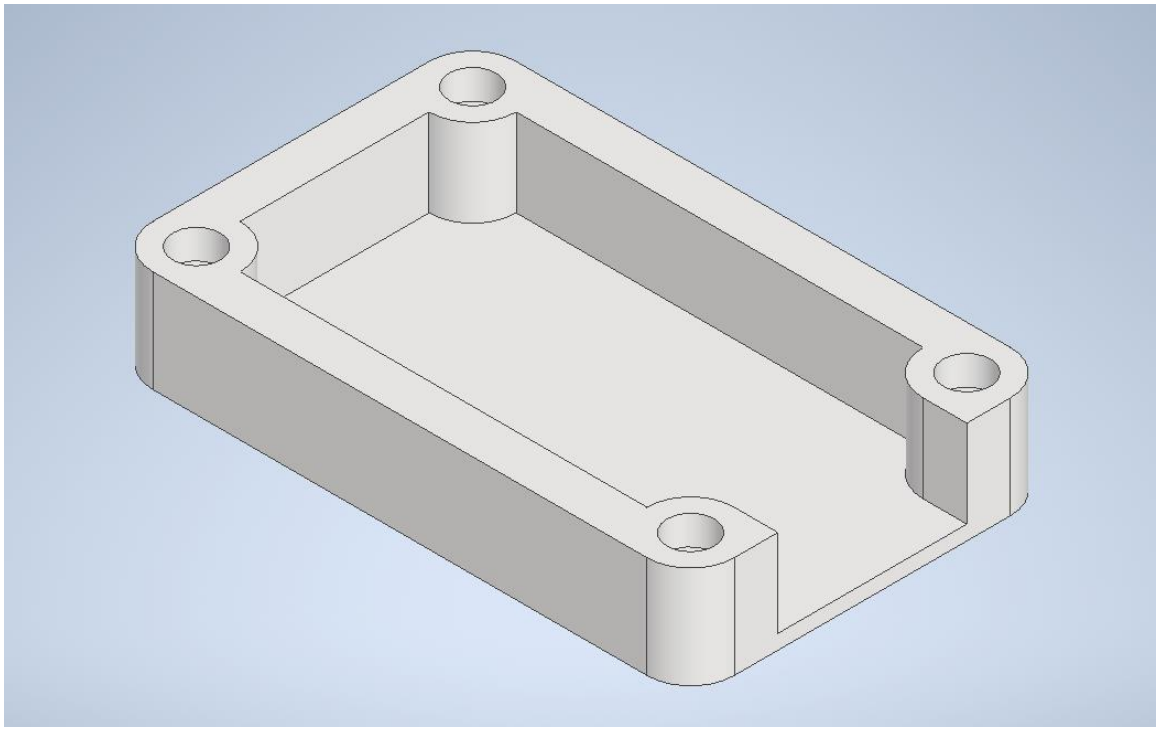


Figure 28: Final CAD of Battery Enclosure

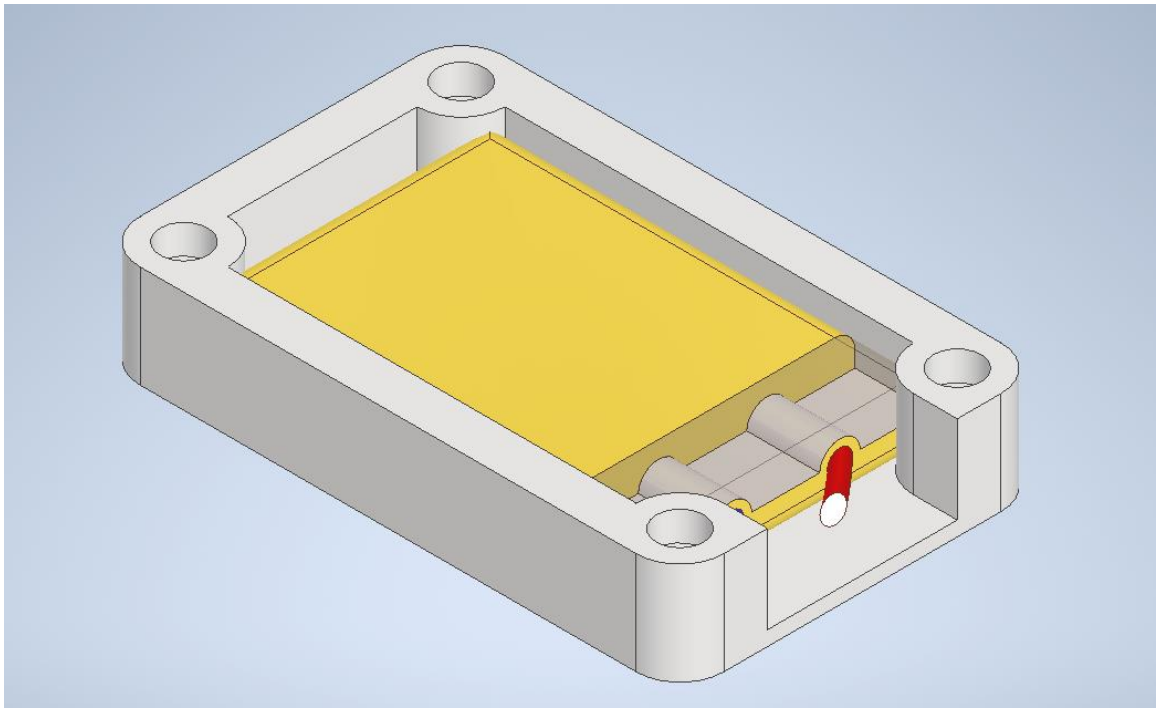


Figure 29: Final CAD of Battery Enclosure with Battery

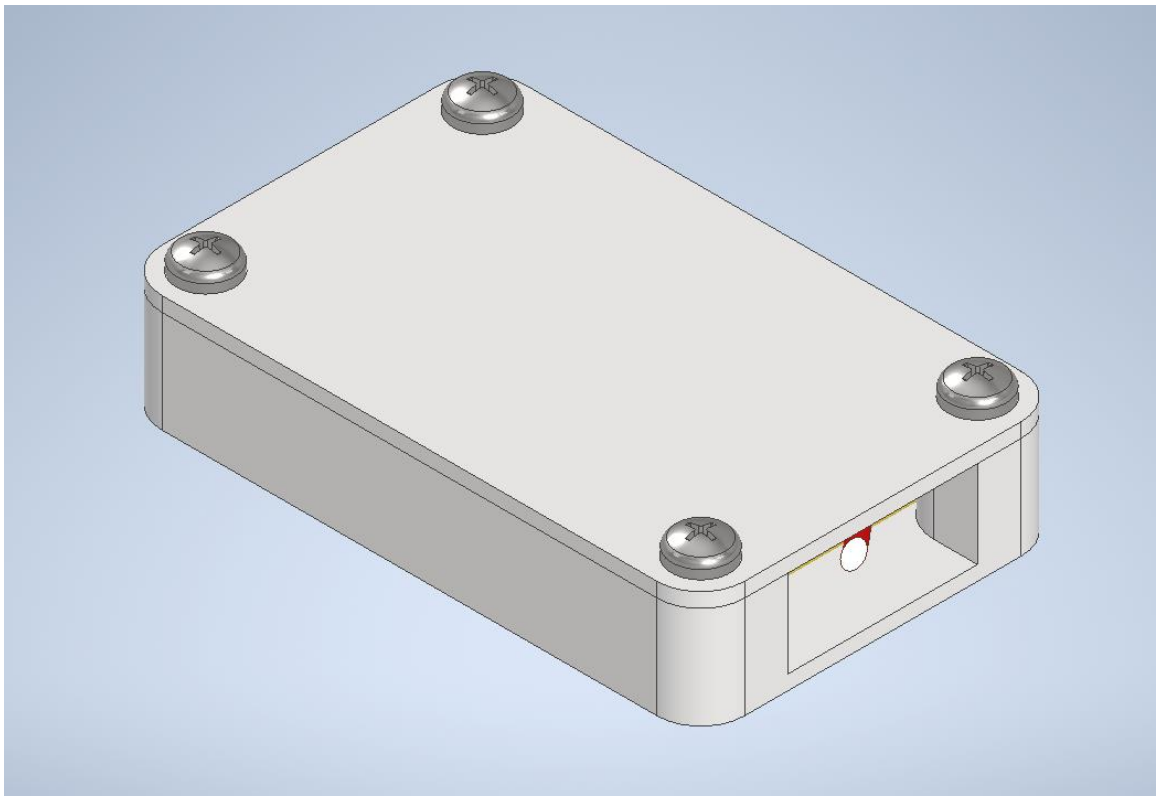


Figure 30: Final CAD of Battery Enclosure Fully Assembled

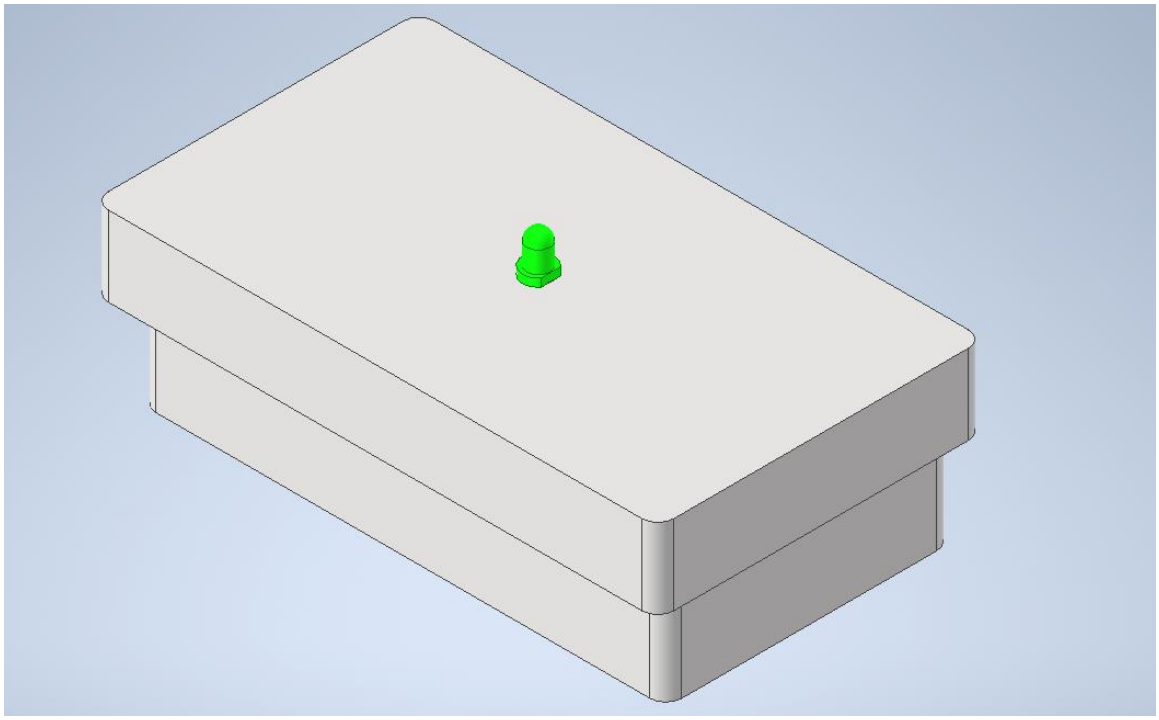


Figure 31: Final CAD of LED Enclosure Fully Assembled

2.5 Hardware Iteration

A critical component of our capstone project involved the integration and iterative development of the hardware required to interface with the PixArt PAJ7025R3 image sensor. Our goal was to design and manufacture a robust, miniaturized, and wearable system capable of acquiring reliable optical motion data from the user's forearm movements. This required multiple cycles of printed circuit board (PCB) development.

2.5.1 Motivation & Initial Challenges

The PixArt PAJ7025R3 is a low-resolution optical navigation sensor designed for gesture recognition and motion tracking. Despite its potential for integration into wearable systems, this sensor posed significant design and prototyping challenges. Its unique footprint and lack of a standardized breakout board meant that no commercial development modules existed at the time of our design. Furthermore, its documentation was limited in way to connect and interfaces with the sensor.

Our initial attempt to interface with the sensor involved manually soldering jumper wires to the gold finger contacts. This approach did not work at all and the soldering heat caused damage to the sensor. We also reached out to several external vendors to inquire about potential breakout board or custom connector fabrication, but the lead times and costs were not feasible within our project deadlines.

After damaging a sensor, it quickly became apparent that designing a custom PCB was the most effective and scalable path forward.

2.5.2 First Iteration: PixArt PCB

The main goal of this pcb was to interface with the PixArt PAJ7025R3 sensor. To do this, a custom footprint was created based on its datasheet mechanical specifications. This allowed precise pad alignment with the sensor's gold fingers for soldering. For this iteration, our group was not entirely sure which pins were needed for our use case so the pcb routed all sensor pins to a 9-pin through-hole header, making it easy to connect to breadboards and microcontroller development boards during testing. Labels were added for every single pin to aid in debugging.

The first iteration was successfully fabricated, and we were able to communicate with the sensor. Although the design proved functional, it revealed several areas for optimization in size, component selection, and number of pins needed.

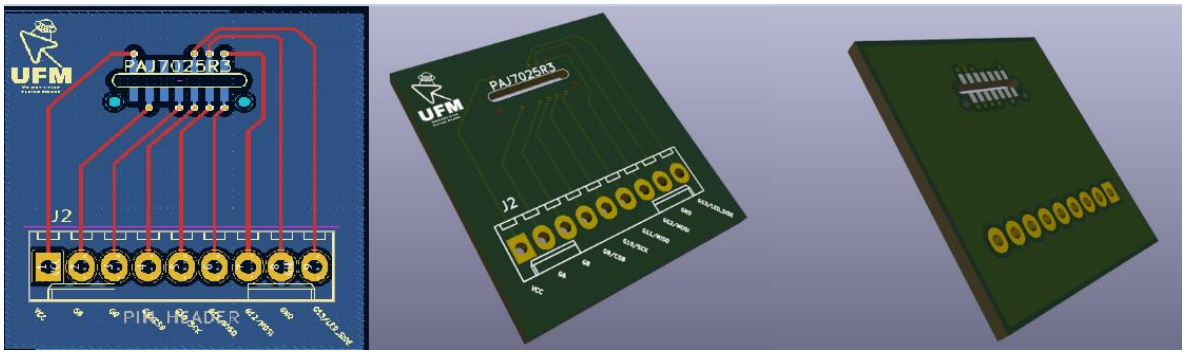


Figure 32: First Iteration PixArt PCB KiCAD Model

2.5.3 Second Iteration: Pixart PCB

Our team identified key areas for improvement from our experience interfacing with the sensor on the first iteration PCB. After testing, we found out that only six of the nine pins were required for our project to function: VCC, GND, MISO, MOSI, SCK, CSB. This allowed us to reduce the size and complexity of the board layout.

The datasheet also recommends the use of local decoupling capacitors. While not mandatory, we decided to add the two capacitors in case of power ripple and supply instability. We then added four non-plated through holes (NPTH) for mounting the board inside a 3D-printed shell. This choice ensured that the sensor PCB could be securely fixed in position without introducing any risk of creating unintended electrical connections or ground loops.

The through-hole pin header from the first design was too bulky and vertically intrusive for wearable applications. In its place, we introduced a **Flat Flexible Cable (FFC)** connector. This offered several advantages:

- Reduced Z-height for easier enclosure integration.
- Lighter and more flexible than ribbon cables or wire bundles.
- Maintains a reliable mechanical connection with minimal risk of intermittent contact.

We evaluated several locking mechanisms (e.g., slide-lock, flip-lock, rotary-lock), ultimately selecting a rotary-lock + back-lock hybrid connector due to its combination of mechanical strength and compact form factor.

These updates reduced the board area by over 30%, from an initial **30 mm × 30 mm** to a tighter **22 mm × 22 mm.**

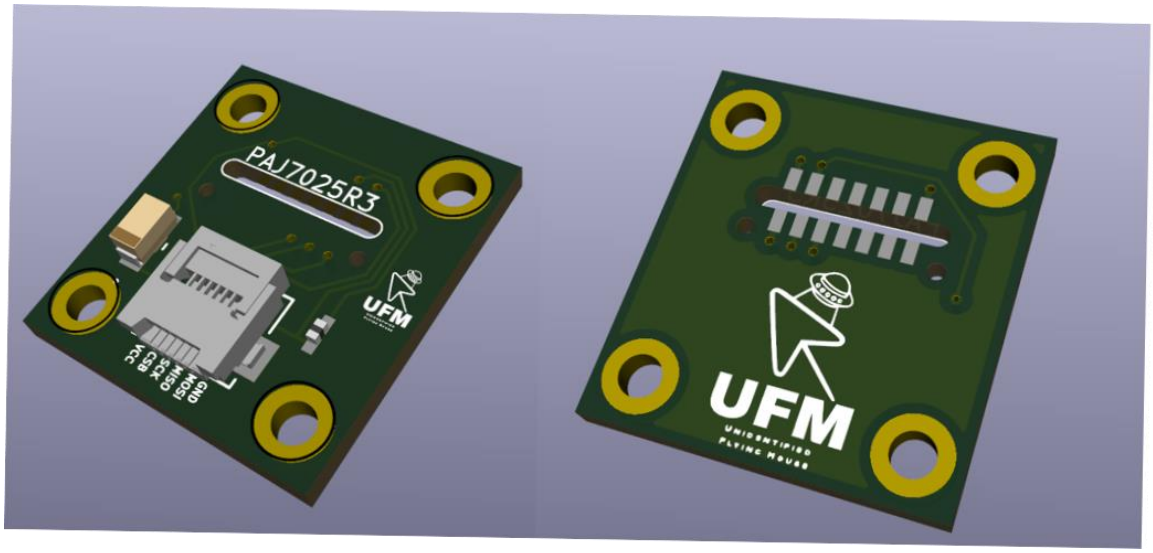


Figure 33: Second Iteration PixArt PCB KiCAD Model

2.5.4 Third Iteration: Pixart PCB

Through conducting more research about PCB design and working with the sensor more **we concluded that more improvement could be made. Specifically, the sensor's lens sat too far recessed under the housing lip, which introduced occlusion and reduced the effective field of view.**

In the third and final iteration, we re-optimized the board layout with the following changes:

- Dimension Reduction: The PCB dimensions were minimized to **22.550 mm** × 16.850 mm, nearly 50% reduction compared to the first version.

- Trace Clearance: We enforced tighter trace clearance ensuring that all traces and components were placed together as tightly as possible to optimize space.
- Front Occlusion. To eliminate the occlusion, we experience with the first PCB iteration, we shaved down the area between the front of the PCB and the camera.

This final version fully resolved the occlusion problem and offered a mechanically and electrically robust platform for sensor integration.

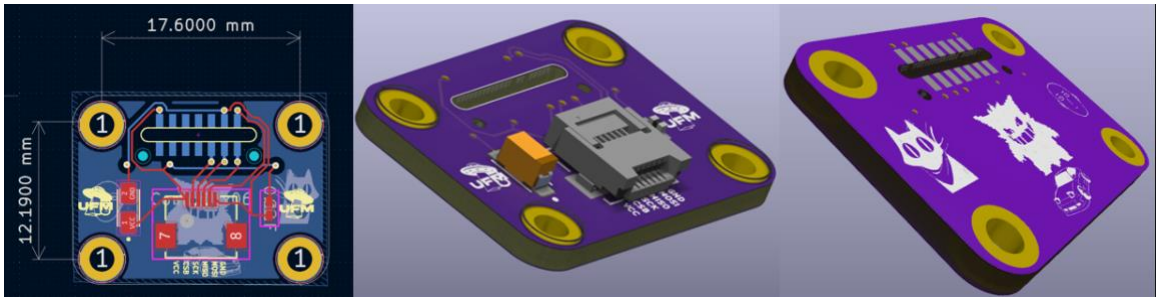


Figure 34: Third Iteration PixArt PCB KiCAD Model

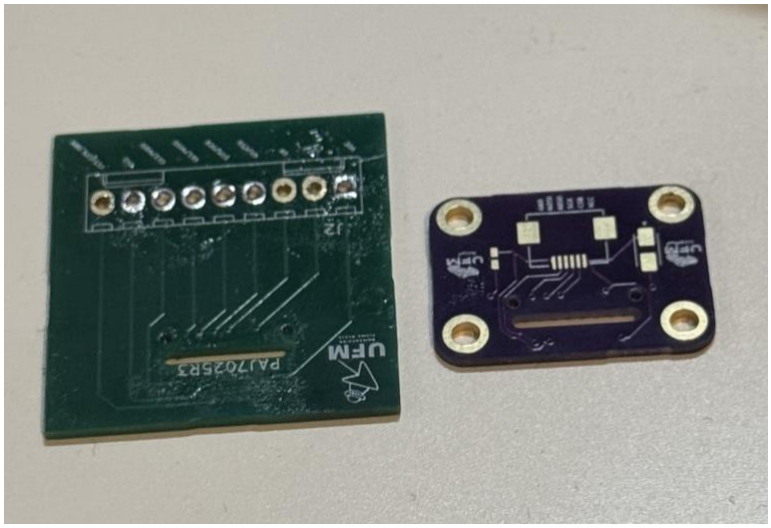


Figure 35: First Iteration vs Third Iteration PixArt PCB

2.5.5 Main PCB: Modular Stack Architecture

In parallel with sensor board development, we also designed a second PCB to host the remainder of the system electronics, including:

- A **FireBeetle ESP32 microcontroller** for sensor interfacing and Bluetooth data transmission.
- An **MPU6050 inertial measurement unit (IMU)** to detect wrist orientation and movement.

- A **motor driver** for connecting haptic vibration motors.

Due to time constraints and limited access to advanced PCB assembly services, we used through-hole pins and surface mount female pin headers to implement a stacked module architecture. The FireBeetle MCU was mounted directly above the IMU and motor driver board. This approach allowed for:

- Efficient use of vertical space.
- Easy access to components through socketing the MCU
- Compact design
- Easy mount to 3d printed enclosure

To optimize the design and avoid component collisions, meticulous planning was essential. Our goal was to ensure the PCB's size matched that of the largest component, the MCU. As a result, we had to experiment with various component configurations to achieve the best arrangement.

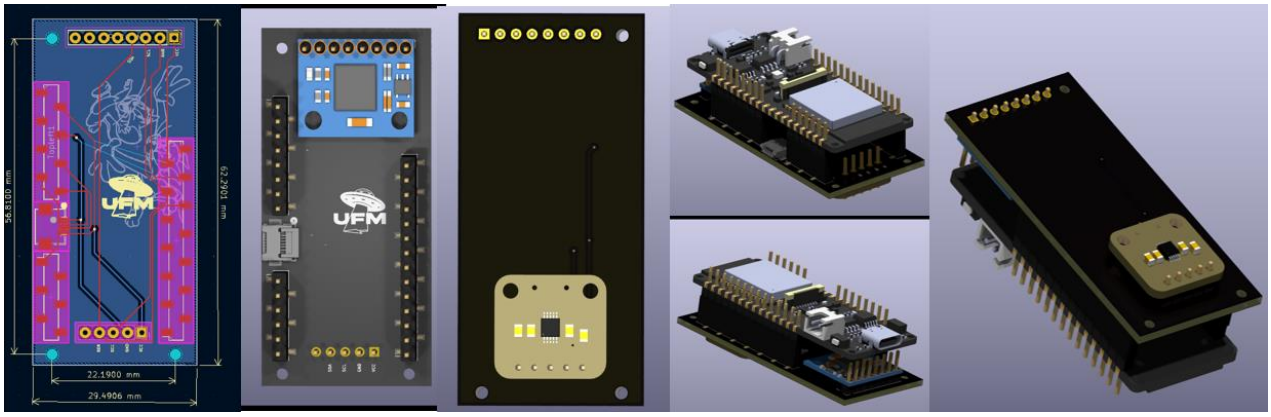


Figure 36: Main PCB KiCAD Model and Modular Stack Assembly

2.6 GUI: UFM Configurator

To give end users an intuitive way to configure our wearable mouse—and to avoid the cumbersome manual build-and-flash process—we explored several desktop-GUI approaches. Below we detail each major iteration, the technical rationale, challenges encountered, and the final solution we adopted.

In our first attempt, we set out to build a small Windows app in C++ using Qt Designer. The reason we chose this was because C++ is faster and widely used in most I/O GUIs. The idea was to let users pick a compiled .bin or .hex file and flash it to the device over USB. Although it worked, everyone needed Visual Studio, the Qt SDK, and PlatformIO plugins just to build and test. On top of that, users still had to rebuild firmware whenever they wanted to change a setting.

Next, we switched to Python with PyQt6 and PyInstaller. Writing the interface in Python was much faster, and we automated PlatformIO so users didn't have to run commands by hand. But bundling the Python runtime, Qt libraries, and PlatformIO modules into an .exe that was over 200mb. Startup took over 10 mins on some computers, and antivirus tools often flagged the huge file.

Finally, we simplified everything. We updated the firmware to accept plain-text commands like "sensitivity=0.1" over UART . On launch it scans for our device, shows simple controls for hand mode, sensitivity, and click angles, then sends those ASCII commands directly—no firmware files, no IDE, no flashing. The result is a compact executable that starts in under a second and lets users tweak settings on the fly.

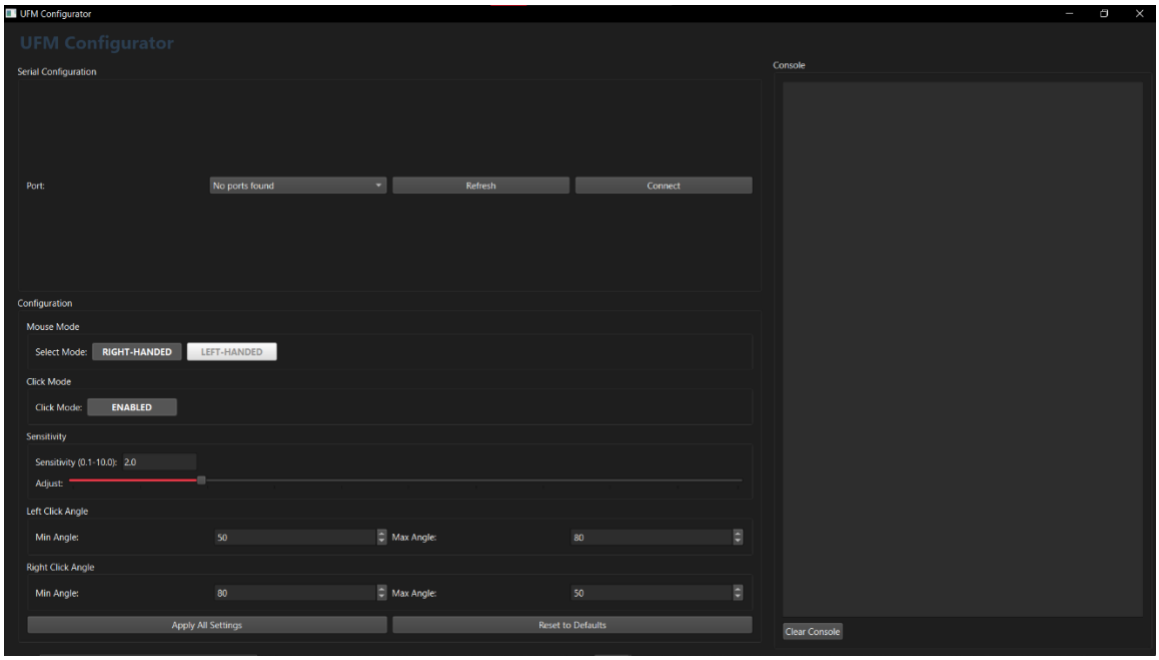


Figure 37: UFM Configurator User Interface Application

2.7 Component Selection

2.7.1 PixArt IR Multiple Objects Tracking Sensor

At the very beginning of the project, our team knew the general functions we wanted, but did not know about all the technologies available to achieve them. We decided to reach out to a well-known computer mouse sensor manufacturer PixArt Imaging Inc., through email. After describing our problem and the scope of our project, the PixArt team provided us with a suggested sensor to use and offered to send us two free samples to facilitate testing.

The PAJ7025R3 is an integrated CMOS image sensor and processor that has object tracking capabilities. It tracks the LED with a 141-degree field of view, and returns its coordinates as seen in the sensor using SPI protocol to our ESP-32 microcontroller.

2.7.2 IR LED

The PixArt PAJ7025R3 has an infrared filter whose passband ranges from 800nm to 900nm wavelengths of light. We chose the OSRAM SFH-4356 IR Emitter as it emits light at 850nm, which is in the middle of the IR sensor's passband. An added advantage to using an IR LED is that most of the light is invisible to the human eye, making the tracking reference less distracting to the user.

2.7.3 IMU

For our purpose, the IMU detects complex wrist movements through wrist movements. This meant we needed a device to capture rotational and linear velocities (6 DOF). Various IMUs on the market could serve this purpose. However, we selected the MPU 6050 for its affordability and established performance, as it is widely used in numerous other projects.

2.7.4 Microcontroller

In our design, the microcontroller acts as the central brain, facilitating and receiving commands from the various sensors to perform actions. We headed towards an ESP-32 development board due to its multiple features. These include an onboard transmitter, wireless Bluetooth capabilities, and a low-power functional mode to improve battery life. Additionally, we targeted a board that had a small profile, due to the nature of wanting our design to be minimal. Initially, we settled for the ESP-WROOM-32, which included all the features, as well as multiple GPIO pins. As the project progressed, we realized our microcontroller did not have an onboard voltage regulator, meaning it could not support an external power source. The ESP-32 Firebeetle was then chosen, as it supported all the same features but also included an onboard voltage regulator, around the same size as the previous board.

2.7.5 Haptic Feedback

The haptic feedback module is constructed using the DRV2605L motor driver and a low-powered motor. Since one of our goals was to maximize the battery life, we selected this combination over other motors. The driver was also attractive as it offers a wide range of motor buzzing patterns, which simplified the requirement for PWM controls.

The motor itself is the Vybronics VCLP1020B002L, which is a low current coin sized haptic vibration motor. It was selected as it consumes only 30mA and is small and lightweight. The vibration force is less than other haptic motors, at only 0.75 G, since each GPIO of the ESP-32 is only rated for a max current of 40mA, sizing for the proper electrical rating was more important.

2.7.6 Wristband

For our wristband, we decided to use a compression sleeve, equipped with Velcro straps for each of the enclosures. As mentioned above, we settled on this type of clasping mechanism as it best suited our physical requirements. The compression sleeve provides a very snug fit but does not limit the user in any fashion. Additionally, due to the elastic material, the sleeve can be easily equipped using a single hand and supports various sizes of wrists. The material is also breathable and can be worn for a prolonged period without any irritation.

The enclosures were designed on Autodesk Inventor and printed using PLA filament. PLA is beneficial as it provides a balance between being lightweight and sturdy enough to shelter and protect the internal components.

2.7.7 Capacitor Selection

- Ceramic Capacitors
    - **Pros**: Compact, low cost, excellent frequency response; ideal for decoupling due to low ESR.
    - **Cons**: Limited capacitance range; capacitance affected by voltage (DC bias effect).
- Tantalum Capacitors
    - **Pros**: High capacitance in a small form factor; stable performance; low leakage current; good for bulk storage.
    - **Cons**: More expensive; sensitive to voltage spikes, which may cause failure.
- Electrolytic Capacitors
    - **Pros**: High capacitance; low cost; suitable for bulk storage.
    - **Cons**: Larger size; shorter lifespan; higher ESR.
- Film Capacitors
    - **Pros**: Highly stable and reliable; excellent tolerance.
    - **Cons**: Bulky and costly; not ideal for compact designs.

Final Choice:

- 0.1 µF **ceramic capacitor** for high-frequency noise decoupling due to low ESR and fast response.
- 10 µF **tantalum capacitor** for bulk storage, chosen for its compact size and stable characteristics.

2.7.8 Connector Selection: FFC vs. FPC

To improve compactness and weight, we transitioned to flat flexible connectors, comparing FFC and FPC options:

- **Flat Flexible Cables (FFC)**
  - **Pros**: Low profile; flexible; cost-effective; widely available.
  - **Cons**: Limited durability; prone to damage with improper handling.
- **Flexible Printed Circuits (FPC)**
  - **Pros**: More durable; better electrical performance; customizable.
  - **Cons**: Higher cost; requires precise manufacturing.

Final Choice:

- Selected **FFC** for its balance of simplicity, affordability, and suitability for lightweight applications.

2.7.9 FFC Connector Locking Mechanisms

The team evaluated several FFC locking mechanisms to ensure secure connections under movement and vibration:

- Back Lock
  - **Pros**: Secure connection; straightforward to implement.
  - **Cons**: Difficult to access in tight spaces.
- Flip Lock
  - **Pros**: Easy to use; quick assembly.
  - **Cons**: Bulky; prone to wear over time.
- Flip Lock + Back Lock
  - **Pros**: Combines security and convenience.
  - **Cons**: Larger footprint may impact compactness.
- Latch Lock
  - **Pros**: Very secure; prevents accidental disconnects.
  - **Cons**: More force needed; may increase size.
- Rotary Lock
  - **Pros**: Extremely stable; resistant to vibration; strong mechanical engagement.
  - **Cons**: Requires more effort to engage/disengage.
- Rotary Lock + Back Lock
  - **Pros**: Enhanced stability and safety; resists vibration and accidental pulls.
  - **Cons**: Bulkier; may require more assembly effort.

- Slide Lock
    - **Pros**: Quick and easy to operate.
    - **Cons**: Less secure in high-vibration environments.

Final Choice:

- Selected **Rotary Lock with Back Lock** for its robust, vibration-resistant connection and compact design which is ideal for wearable applications.

2.7.10 Mounting Hole Type: NPTH vs. PTH

We evaluated through-hole types for PCB mounting purposes:

- Non-Plated Through-Hole (NPTH)
    - **Pros**: Cost-effective; simpler to manufacture; suitable for mechanical use.
    - **Cons**: No electrical connectivity.
- Plated Through-Hole (PTH)
    - **Pros**: Supports electrical connectivity and structural reinforcement.
    - **Cons**: More complex and costly.

Final Choice:

- Selected **NPTH** for mounting holes as only mechanical support was required.

3. ACTUAL IMPLEMENTATION

The Final design of the UFM is shown below. It features a compression sleeve wrist band with three 3D printed enclosures. The final design and how all components work together in the system is shown below.

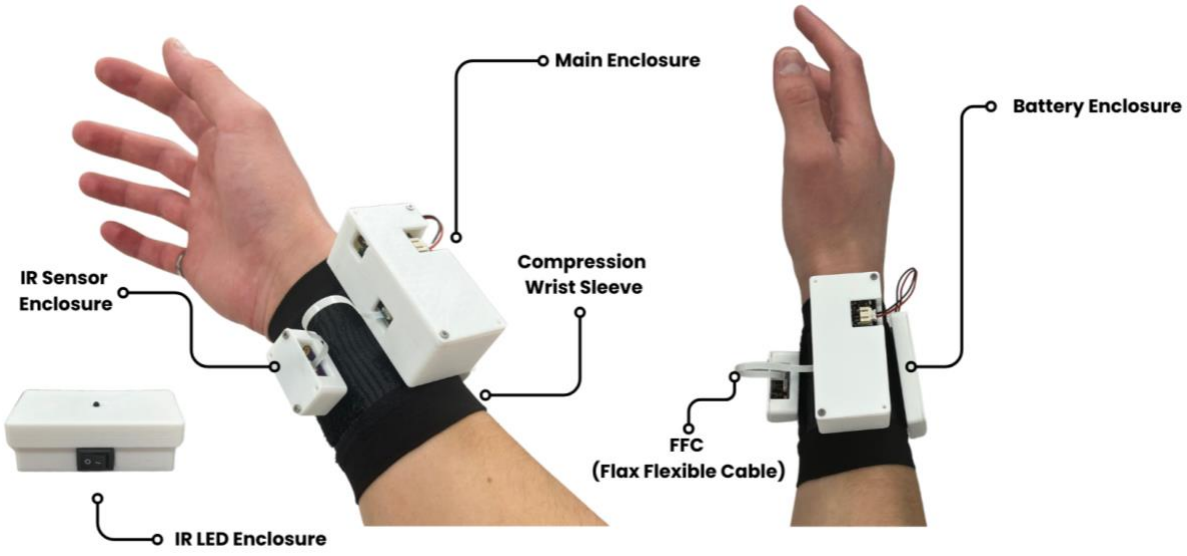


Figure 38: Final Device Assembly

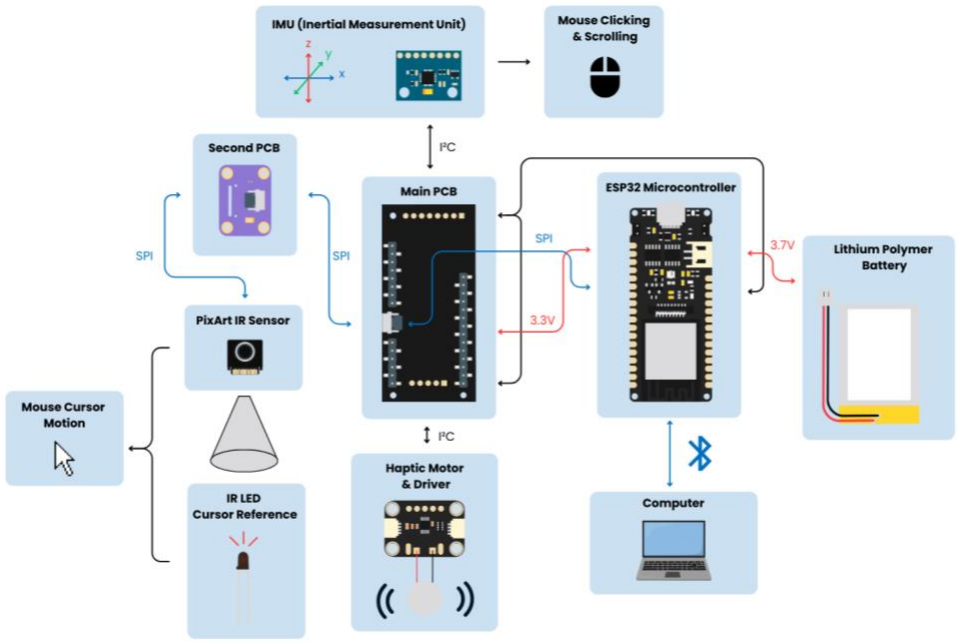


Figure 39: Final System Block Diagram

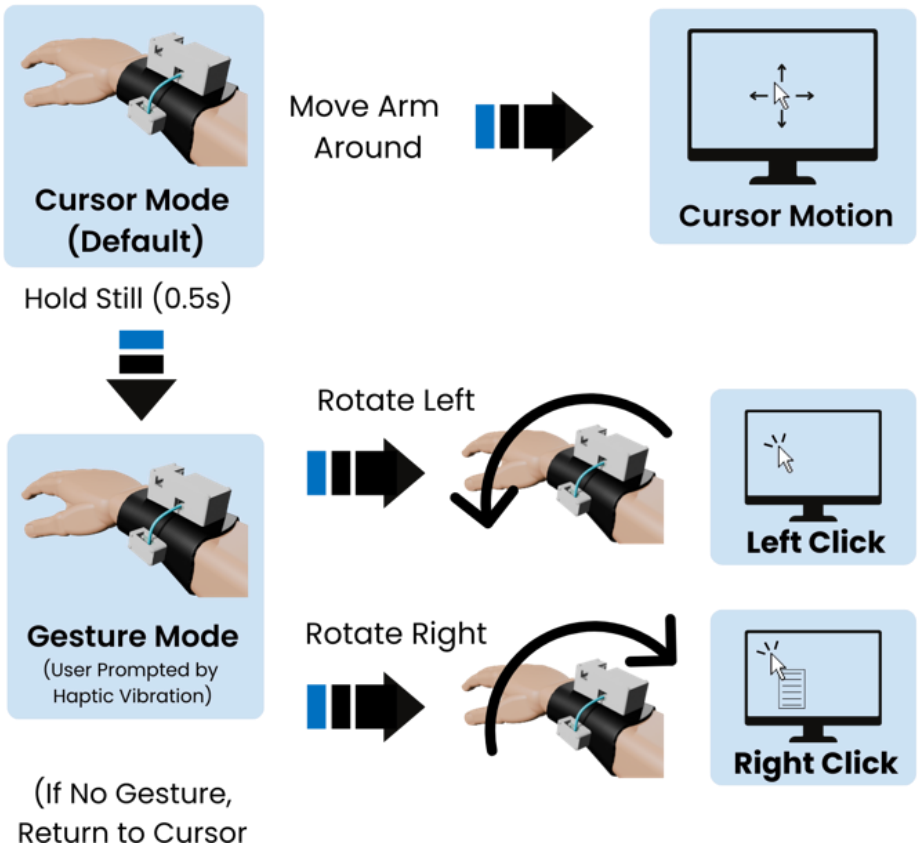


Figure 40: UFM Functions

The UFM can perform accurate, low-latency mouse cursor movements, as well as left and right mouse clicks. Mouse cursor motion is achieved by moving the arm relative to the reference LED. Clicking is performed by pausing briefly, then performing a rotation to the left or to the right.

For a visual demo and explanation of how the UFM works, our project video can be accessed at the following link: https://www.youtube.com/watch?v=52ZPP2TyDK4.

3.1 Hardware Implementation

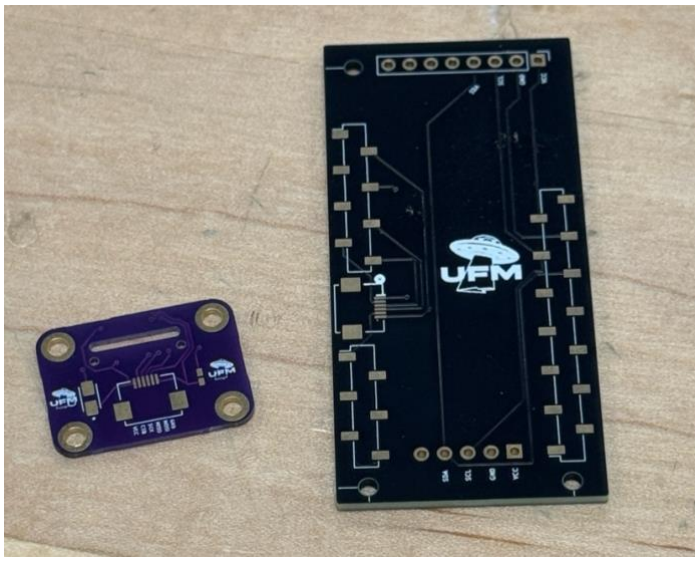


Figure 41: Final PixArt PCB and Main PCB without Components

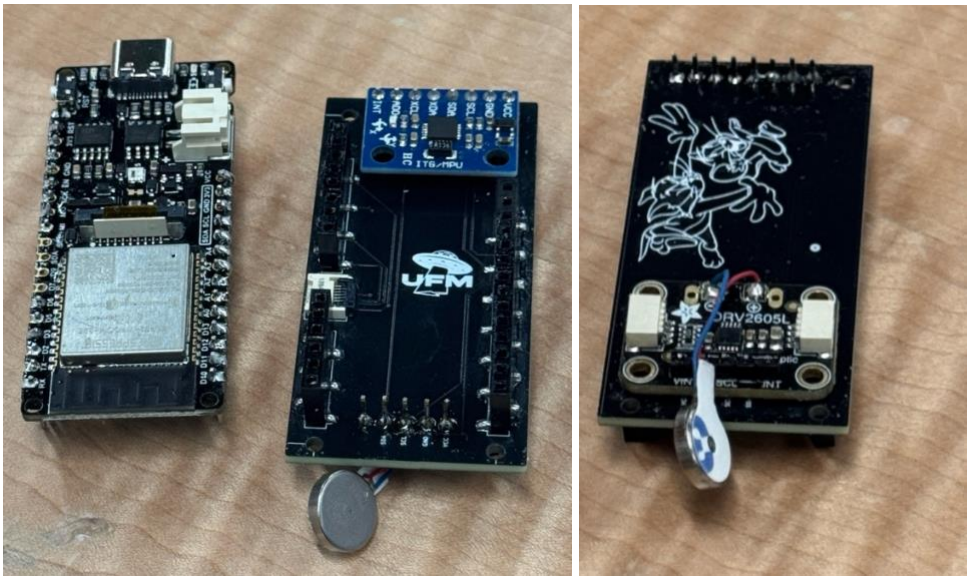


Figure 42: Modular Stack Architecture (Shown Left to Right are ESP-32 Firebeetle2, Main PCB Top View, Main PCB Bottom View)

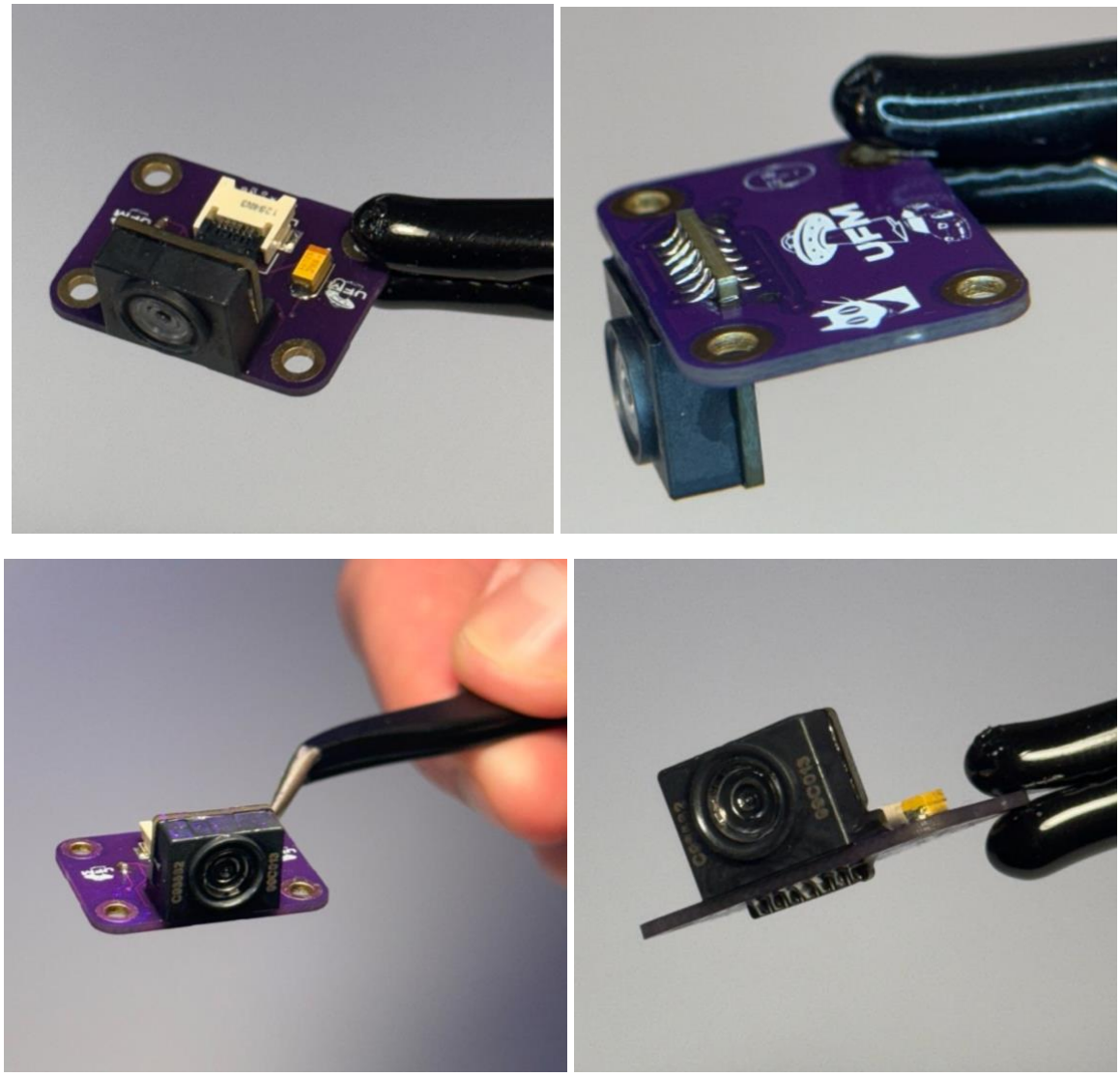


Figure 43: Various Views of Final PixArt PCB

3.2 Software Iterations

The software aspect of the project went through several overhauls, due to unforeseen design and implementation challenges, along with other technical difficulties.

3.2.1 On-board 3D Spatial Tracking

This is the initial software implementation of the device as listed in section 2. Utilizing a PnP solver, the program will solve for the positions and orientation of the sensor in 3D space. This implementation, however, had its challenges. It requires large amounts of floating-point computation, something the processor onboard the ESP32 was not able to

handle in real-time. This approach resulted in a lack of responsiveness for the user, and unusually high-power consumption.

3.2.2 Offloaded 3D Spatial Tracking

The challenge faced in section 3.1.1 turned us towards moving the computational overhead to the device that the mouse is connected to. Typically, a Windows or Mac computer has the amount of processing power to compute these floating-point calculations in real time. Modern processors' increasing number of parallel cores and threads also allows minimal interference with computers' overall usage performance. However, this approach came with another set of problems. For one, this implementation meant that the software for our device had to be developed and maintained in multiple different fronts for different platforms, OS, and even devices. It removes the advantage of making our device HID compliant, which isolated our development from the hassle of managing different system calls and the set of problems it could come with. This approach was abandoned in the end when we've realized that we were headed in a direction further from our original vision.

3.2.3 2D Relative Planar Tracking

Finally, we've made the decision to once again migrate the entire code base to the ESP32. This time, we've massively reduced the complexity of the calculations. By only tracking the relative 2D positioning of the sensor, we've eliminated the need for a PNP solver for the cost of slight inaccuracies when using the device for a long period of time. This tradeoff was worth it, since the phenomenon called "mouse drift" already exists in all mouse devices that people use every day. Absolute positioning are often used for devices with much higher accuracy requirements, such as VR/AR applications, and drawing tablets for digital art. The common computer mouse device didn't have this constraint, and relative motion proved to be sufficient for everyday usage. The implementation was finalized in only 30% of the memory usage of our ESP32 board. The program performed much more swiftly and efficiently. The noticeable input lag has been drastically decreased, and overall responsiveness was much more pleasant for the user. This implementation was achieved by taking a positional value of a single object every frame and comparing the current position with the previous frame's position to generate a delta. This delta was scaled to the screen resolution and multiplied by a sensitivity parameter that can control the sensitivity of the mouse. A user can then simply find the device in BLE pairing mode and pair with the device to allow a seamless user experience.

3.3 Expenditure

The total cost of the project is broken down in the following table.

Item	Amount	Cost	
ESP-WROOM-32	x2		$22.59
Haptic Motor	x5		$27.03
Motor Driver	x2		$26.40
Battery Spring Clip AAA	x6		$10.38
LiPo Battery	x1		$21.46
0.1 uF Capacitor	x50		$0.51
10 uF Capacitor	x10		$3.72
FFC Connector	x12		$22.35
FireBeetle ESP32-E	x2		$33.88
FFC Cable 6pos 0.5mm pitch	x6		$27.80
Surface Mount Connector 8 Pos	x6		$11.12
Surface Mount Connector 6 Pos	x6		$8.54
Surface Mount Connector 5 Pos	x6		$7.32
Surface Mount Connector 3 Pos	x18		$16.27
IR Emitter	x10		$37.00
Poster	x1		$80.11
Velco	x1		$12.42
Compression Wrist Sleeve	x1		$14.68
PixArt PAJ7025R3 Sample Kit	x5		$117.80
PixArt Test PCB	x5		$24.61
Redesigned Pixart PCB + Main PCB	x10		$135.60
Heat Resistant Tape	x1		$14.68
		Total	$682.32

The original estimate for the project at the proposal stage was around $120 CAD. This is around six times less than the actual amount spent of $682.32 CAD. There are several reasons for this, such as shipping costs, damaged components, several PCB iterations and a minimum number of components needing to be purchased (ex. We needed to purchase five PixArt sensors, while the final design only uses one, JLC PCB requires a minimum number of 5 PCBs to be manufactured, etc.).

4. CRITICAL PROBLEMS SOLVED

4.1 Tracking Complexity

The tracking complexity of the software implementation was a continuing issue of this project. The scope of this aspect was always loosely defined and required a solution to a problem with many approaches. The issue comes with our initial assumption of needing 3D spatial tracking for the device to work. As per our design goal, we've planned to implement mouse click functionalities with rotational movement. This input would give an angular positional offset to the cursor's position on the screen. We thought the only way to combat this was to retrieve the spatial position and orientation of the sensor, to correct positional offset and isolate the cursor movement from the user's arm's angular movement. This approach required large amounts of floating-point calculations, too much for an ESP32 to perform in real time at a usable refresh rate. This problem was only solved when we thought of an alternative input to mouse clicks. By holding the cursor still to enter a separate operating state that disabled the mouse movement, we can allow any gesture to be input before re-enabling the cursor movement. This meant that we no longer need to correct the offset created by the arm rotation. With this constraint removed, we were able to implement a much more computationally simple implementation in 2D tracking. This was a major breakthrough in the project, as it showed us for the first time, a natural and responsive user experience with minimal input delay, intuitive user experience via the simple Bluetooth connection, and minimal tracking inaccuracies, due to the average person's existing experience operating a common computer mouse.

4.2 Acquiring More Sensors

As mentioned in part 2.5.1 Motivation & Initial Challenges, after damaging a sensor, we were left with only a single working unit, which was not enough to meet our goal of demonstrating at least two functional wrist mice. When we contacted PixArt to acquire more sensors, they informed us that due to a rework in their manufacturing operations, the earliest they could ship replacements would be at the end of February. This delay left us with very little time for further development and testing. Since the sensor is not commercially available, we explored other options and found only a few vendors offering it for prototyping. After numerous emails and phone calls, we eventually found one vendor that had a pack of five sensors in stock. However, this vendor only sold to registered businesses, so we had to involve McMaster staff to help us complete the purchase. After several more emails and roughly two weeks of waiting, the sensors finally arrived, allowing us to continue development with a more robust and reliable setup.

4.3 PCB Soldering and Assembly

One of the most critical technical problems we faced involved soldering and assembling the main PCB and camera module, particularly due to the extremely small footprints of components like the fine-pitch camera connector and tiny surface-mount capacitors. Early

in the process, we experienced persistent solder bridging with the 0.05 mm pitch connector, despite using the recommended method of applying solder paste with a stencil and hot plate. Even minor misalignments of the stencil resulted in uneven paste distribution, causing shorts between adjacent pins. Initially, these faults were difficult to detect and only became apparent during post-soldering continuity checks with a multimeter. After diagnosing the problem, we discovered that the solder paste provided by Thode Makerspace was not fine enough for such precise work. Our solution involved applying the stencil, heating the board, then manually inspecting and removing excess solder with a solder wick—though this carried its own risks, including potential damage from repeated heating cycles. The challenges continued when assembling the camera PCB. Because the sensor protruded from the back, reheating on the hot plate was no longer an option, meaning the soldering had to be flawless on the first try. At one point, we accidentally bridged the connector pins and attempted to remove the sensor with a hot air reflow gun, but due to our limited experience, we damaged the module beyond repair. Additionally, soldering the camera sensor required an unconventional and delicate “L” motion technique due to its tightly spaced gold fingers. Applying too much solder caused bridging, while too little failed to make a connection. The process was made even more difficult by the strict constraints outlined in the datasheet, which limited soldering time to two seconds and temperature to 300°C. Despite all these challenges, through trial, error, and refinement of our techniques, we were able to successfully mount all necessary components and bring both PCBs to full functionality.

5. CONCLUSION

The final version of the UFM is a polished prototype that successfully addresses the problem that our group set out to solve at the onset of the project: providing a more accessible alternative to the typical computer mouse design for those who have difficulty performing repetitive finger and hand movements. It performs the two main functions of a computer mouse, namely mouse cursor movement and clicking, while keeping all functional movements exclusive to the wrist, arm, or shoulder of the user. The compact and ergonomic design provides a sleek accessory that can be worn comfortably for a long time, and a GUI that allows user flexibility. While there is still room for improvement in the design, this project outlines the possibilities and opportunities in creating assistive devices.

6. FUTURE PLAN

The UFM is a more accessible option than an ordinary computer mouse for those with limited hand dexterity or pain in the fingers. However, its overall performance is less than the average mouse, leaving lots of room for improvement.

We would like to flush out the tracking range of the mouse. A common problem we saw at the demo where we had many guests try out our device was that a lot of participants would instinctively move the mouse too far away from the single tracking sensor. This could theoretically be easily fixed by having a bigger grid of LEDs instead of a single one. However, during development the mouse had some minor issues with this approach, but a fix a well within the scopes of our abilities given more time.

We would also like to increase the performance and responsiveness of the device. Currently, the PixArt sensor we chose works well enough for very general purposes such as simply navigating the OS. The performance of the device starts becoming an issue when more accurate and faster tracking is needed, such as in a gaming context. We would like to incorporate the device to work at much higher refresh rates than the current 200hz, and move on to a dedicated wireless dongle system instead of using Bluetooth.

6.1 Additional Mouse Functions

The first and most impactful changes which could be made are the inclusion of more mouse functions. At present, the UFM can only provide mouse cursor motion and clicking. However, the typical computer mouse has many more functions which could be implemented. By adding more functions, UFM has the potential to be a like-for-like replacement for the traditional mouse in terms of functionality.

One such function is mouse wheel scrolling, which was investigated by our team. We successfully implemented this function in isolation by capturing a tilt gesture using only the IMU. However, when attempting to integrate it into the system, there were several issues which arose, such as interference with clicking gestures, switching back to cursor mode too early, or difficulty returning to clicking mode. Given more time to debug and investigate, it is likely these problems could have been solved, leading to an additional function which is quite commonly used.

Other functions which were not investigated in detail, but discussed are a held click for a duration (such as for click-and-drag operations), double clicking, scroll wheel clicking, or programmable auxiliary functions present on many modern mice in the form of extra buttons. The addition of some these functions would likely be possible using gestures from the IMU, but the time constraints forced our team to focus on the core use case instead and neglect these additional functions.

6.2 Hardware Improvements

The next improvement would be to the hardware, and assembly of all the components. At present, the highest power component of the project is the ESP-32 Firebeetle. According

to its specs, it consumes an average of 80mA of current, which is much more than the sensors, who operate at around 6mA for the PixArt Sensor, and 4mA for the IMU. Because of the initial thoughts to do 3D pose estimation, our group chose a microcontroller with a decent amount of computing power. After this approach was changed, and computation on the microcontroller was heavily simplified, other controllers might be more suitable to the current functions. With more time to research, a new microprocessor which consumes less power than the ESP-32 could be chosen. As a result, this would allow for a smaller battery and provide an increase in the unidentified flying mouse's battery life.

6.3 More Compact Assembly

Another improvement would be to the overall assembly. The current size is compact, but this could be improved. The main constraint to the size is the ESP-32 Firebeetle2. This is a development board which contains all the circuitry required for level shifting, charging, and properly interfacing with GPIOs. However, since our device does not use all the GPIOs on the controller, a more compact design would be possible by designing a custom PCB from scratch. This would provide the opportunity to house all the components in one compact module, rather than three. This would reduce both the size and complexity of the device, making it more convenient to put on, and less cumbersome to use. Our group avoided this option, as the complexity and time required to design a custom board with the controller was much higher risk and had the potential to be too complex for the project. Given a longer timeframe however, this would lead to a better design overall.

7. SELF-ASSESSMENTS

7.1 Luke

My first major contributions to the project were in the early design stages for component selection. I selected the ESP-32E Firebeetle 2 as our final microcontroller as well as sized the LiPo battery with input from George to determine an ideal capacity of battery for high battery life while still being lightweight. I selected the low-current haptic vibration motor as well as the DRV2505L haptic motor driver to generate click-like vibrations. I also worked with George to finalize the design for our initial schematic of all components together.

After damaging one of our two free samples received from PixArt, I did research on alternative options. However, we determined that PixArt was still our best option, so I called and emailed different vendors and was able to order a sample size of 5 sensors and through McMaster's ECE department (the vendor would only sell business to business, not to general customers).

My largest technical contribution came when working on the mouse cursor and functions. I did research on various types of 3D spatial tracking using cameras and used this research to develop and write the Python simulation for mouse cursor movement. I also performed initial testing with the sensor and modified Bart Trzynadlowski's Arduino code to load settings and obtain serial data from the PixArt Sensor [1]. This code was used as a base for the final code flashed to the ESP-32. As mentioned in the report, the 3D tracking method was eventually scrapped after testing with the hardware as it was too complex to perform on the microcontroller. I worked alongside Ryan to determine other options and helped write the simplified code for mouse cursor movement.

I made various contributions to the visual appeal of the project and created 2D visuals at each stage of the project (proposal, phase A/B, expo). The include the team logo, concept drawings, initial function visualization for the proposal, block diagrams, component breakdowns, etc. I designed and created the team's capstone expo poster and did voiceovers and physical demos to assist Ryan in the creation of our video.

I did some work on the final physical assembly of the device. I sewed the Velcro patches onto the compression wrist sleeve and soldered the LED reference circuit.

I provided a lot of administrative support for the project as well. I regularly checked on the team's progress and tried to help each member a little bit in each part of the project to better understand the bigger picture of our product and see what could be improved. I regularly scheduled team meetings and provided general support to team members (ex. Helping Matt with routing traces and ensuring proper PCB connections, giving feedback to George on 3D model designs, etc…).

I think our team dynamic worked well, and each member fell into a specific role. When it comes to reports, we always split the work evenly and each member did their share. I feel

my contribution to the team was satisfactory over the course of the project. As a percentage, I would mark my overall contribution as ~25%.

7.2 George

At the start of the project, I contributed by researching different methods of how we can achieve the fundamental purpose of our project. This meant researching different hardware and software components and weighing the benefits of each. Later, after the initial components had been determined, I assisted Luke with creating a schematic that showcased the preliminary design. This was translated into an actual schematic, where I utilized KiCAD software.

Next, I integrated the IMU and haptic module with our selected microcontroller. This was achieved by sifting through numerous libraries and online resources to better understand the components. I also researched different methods of capturing consistent motions, which made me dive into Kalman Filtering to use all 6 DOF to capture the pitch and yaw values. Also, writing code, merging all the necessary libraries for the initial prototype.

My largest contribution was on the physical modelling side of the wristband and enclosures. This included developing CAD models and assemblies using Autodesk Inventor. I also researched existing solutions targeted towards our intended demographic and used these as guidelines. Next, I created preliminary concepts of the enclosures, iterating along the way from suggestions from Matt, Luke, and Ryan. Matt and I did a lot of reworking and fit testing until the optimal design was finally completed. Additionally, I was also responsible for booking the 3d print stations, to keep the revisions constant.

I also assisted Matt in assembling the prototype. This included soldering components to the PCBs or soldering heatsinks into the enclosures. I also performed debugging to identify faults, such as continuity tests, to resolve any issues.

Lastly, I assisted Luke and Ryan in creating 3D visuals in Blender for the EXPO and video purposes. This included downloading 3D models of components, creating assemblies, and importing them into Blender. This also included creating 3D renders to capture our product in full display.

Overall, I enjoyed working alongside my team members. I got an opportunity to showcase my skills while also getting a chance to learn new ones. I feel that my contribution was satisfactory, coming at around 25 percent.

7.3 Ryan

In the beginning of the project, I was able to brainstorm ideas and lock in our vision for our project. I sourced key components such as the PixArt sensor, by establishing communication with company personnel, and gaining support from them via parts they've sent our way.

Software development and integration was my main and biggest contribution to the project. I am responsible for the final implementation of all of the mouse functionalities, including mode switching, clicking, and Bluetooth interfaces. I also assisted in the development stages of the software with Luke, through the different iterations that the software took over the course of the project.

I was also able to assist Matthew in the development of the GUI software. Since I worked very closely with the software of the mouse device itself, I was able to help Matthew with the development of the GUI.

Finally, I poured large amounts of hours into the presentation of our product. With the help of George's blender models and animation drafting, I was able to produce professional animations in 2D and 3D for our product advertisement. Together with Luke, we were able to direct, shoot, and produce a high-quality video of our product demo and technical breakdowns.

I felt our team dynamic and work distribution to be very effective and efficient. Every member of the team had a very distinct role that we each fulfilled, and we all contributed very meaningfully to the project. Overall, I think my contribution to the project was 25%.

7.4 Matthew

Hardware Development and Assembly

One of my major contributions to this project was in the design, development, and assembly of the custom hardware used in our wearable mouse system. From the beginning of the term, I took the lead on creating our custom PCB for the PixArt sensor, as well as a custom PCB to house the MCU, IMU, and motor driver using KiCad. I was responsible for creating schematic diagrams, designing custom component footprints, and building the PCB layout to fit within our system's physical and functional constraints.

Throughout development, I continuously revised and improved the design based on feedback from the team. This included changes like adding or removing certain components, adjusting footprint placement, and optimizing routing to make assembly easier and minimize errors. I also worked on selecting the right components such as capacitors, resistors, connectors, and other small electrical parts that fit within our design requirements and physical space limitations.

Once the design was finalized, I reached out to multiple PCB manufacturers to confirm that our board design could be fabricated correctly and within our project budget and timeline. I made sure that the custom footprints we created would meet the requirements for manufacturability, especially for components like the FFC connector and IMU, which require precision soldering due to their small pitch.

After receiving the boards, I performed hands-on assembly of the electronics. I soldered several critical components onto the PCBs, including:

- PixArt Sensor
- IMU (Inertial Measurement Unit)
- Motor driver and haptic motor
- Female Pin Headers
- LED circuit components, including the IR LED, power switch, and resistor

For surface-mount components, I worked closely with George to learn how to use hotplates and reflow techniques, gaining experience with solder paste application and proper heat timing to ensure consistent, strong joints without damage to sensitive parts. I also used tools like a solder sucker, solder wick, multimeter, and hot air rework station to remove and correct components, when necessary, especially for parts that were misaligned or bridged.

To ensure electrical reliability, I performed continuity tests across all key connections in the system. This was critical for identifying any shorts, cold solder joints, or broken traces before powering the board for the first time.

GUI Software Development

Another significant contribution I made was the development of the desktop GUI application that allows users to configure and interact with our wearable mouse device.

The goal was to build a simple, fast, and user-friendly configuration tool that would remove the need for users to manually flash firmware every time they wanted to change a setting. I explored multiple approaches, iterating through different deployment options to find the best approach.

- Initial Version (C++ and Qt Designer):
  I first built a Windows GUI using C++ with the Qt Designer framework. This allowed for high-speed execution and better performance with I/O interfaces. The GUI allowed users to choose, and flash compiled .bin or .hex files over USB. Although functional, this version had high setup requirements. Users needed to install Visual Studio, the Qt SDK, and PlatformIO, which was not practical for the average end user.
- Second Version (Python with PyQt6 and PyInstaller):
  To improve usability and speed up development, I switched to Python with PyQt6. This version automated most backend tasks like flashing firmware using PlatformIO and included a more intuitive user interface. However, when packaged with PyInstaller, the application became very large (~200 MB), slow to launch (10+ minutes on some systems), and frequently flagged by antivirus software. These issues made it unsuitable for deployment.
- Final Version (Command-based UART Control):
  I designed a completely new approach that simplified everything. With the help of Ryan, we updated the firmware so it could accept plain text ASCII commands sent over UART (e.g., Sensitivity=0.1). The GUI now scans for our device automatically on startup and provides basic controls to adjust parameters like

hand mode, sensitivity, and click angles in real time. No firmware flashing or extra tools are needed anymore. The result is a lightweight executable that starts in under a second, has no external dependencies, and gives users full control over their device settings.

This GUI not only makes our product easier to use but also makes it feel more polished and professional. It's a small application with a big impact on user experience.

Assembly and Integration

I also contributed to the mechanical side of the project. I worked alongside George to assemble the early and final 3D printed prototypes, helping fit the internal electronics within the enclosure. I contributed ideas on how to make the housing smaller, lighter, and easier to manufacture while still maintaining strength and usability.

Overall Reflection

Looking back on the entire capstone project, I believe my overall performance was satisfactory. I took initiative in multiple key areas such as hardware design, hands-on assembly, and software development for the GUI.

I put in effort not just to complete tasks but to improve them over time, whether it was by revising PCB layouts, refining the GUI to make it faster and more user-friendly, or ensuring that the final physical product was sturdy and presentable.

While I am proud of my contributions, I also recognize areas for growth. For example, I felt like I could have put more time into the initial design of the PCBs. This would have given the team more time to experiment with the sensor. I also believe that the main PCB that I designed could have been much smaller.

Estimated Total Contribution to Project: ~25%

8. REFERENCES

[1] B. Trzynadlowski, “PixArt PAJ7025R2 6dof Tracking Demo with Arduino and Windows,” GitHub. Accessed: Jan. 03, 2025. [Online]. Available: https://github.com/trzy?tab=repositories&q=PixArt&type=&language=&sort=

[2] “Types of Capacitors: Pros, Cons, & Applications,” Circuit Crush. Accessed: Jan. 16, 2025. [Online]. Available: https://www.circuitcrush.com/types-of-capacitors-applications/

[3] “What is the Difference Between FFC and FPC? ,” Quadrangle Products. Accessed: Jan. 16, 2025. [Online]. Available: https://www.quadrangleproducts.com/what-is-the-difference-between-ffc-and-fpc/

[4] “FFC vs FPC: Understanding Flexible Cable Technologies,” Newhaven Display International. Accessed: Jan. 16, 2025. [Online]. Available: https://newhavendisplay.com/blog/ffc-vs-fpc-understanding-flexible-cable-technologies/

[5] A. Brown, “Plated vs Non-Plated PCB Through Holes in PCB Desigs,” epec Engineered Technologies. Accessed: Jan. 16, 2025. [Online]. Available: https://blog.epectec.com/plated-vs-non-plated-pcb-through-holes-in-pcb-designs

[6] “What is a FPC/FFC Connector?,” Omron. Accessed: Jan. 16, 2025. [Online]. Available: https://components.omron.com/eu-en/products/connectors/board-to-fpc-connectors/board-to-fpc-connector_features?utm_source=chatgpt.com

[7] “ESP32 Thing Hookup Guide,” Sparkfun. Accessed: Jan. 16, 2025. [Online]. Available: https://learn.sparkfun.com/tutorials/esp32-thing-hookup-guide/hardware-overview#:~:text=The%20ESP32%20can%20pull%20as,while%20actively%20transmitting%20over%20WiFi.

[8] A. N. Balaji, C. Kimber, D. Li, S. Wu, R. Du, and D. Kim, “RetroSphere,” Proc ACM Interact Mob Wearable Ubiquitous Technol, vol. 6, no. 4, pp. 1–36, Dec. 2022, doi: 10.1145/3569479.

[9] “Perspective-n-Point (PnP) pose computation,” OpenCV. Accessed: Jan. 03, 2025. [Online]. Available: https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html

[10] B. K. P. Horn, “Closed-form solution of absolute orientation using unit quaternions,” Journal of the Optical Society of America A, vol. 4, no. 4, p. 629, Apr. 1987, doi: 10.1364/JOSAA.4.000629.

[11] S. Meers, K. Ward, and I. Piper, “Simple, robust and accurate head-pose tracking using a single camera,” Jan. 2006, [Online]. Available: https://ro.uow.edu.au/articles/conference_contribution/Simple_robust_and_accurate_head-pose_tracking_using_a_single_camera/27792744

[12] “Haptic Technology 101: A Beginner’s Guide to the Different Types of Vibration Motors,” TItan Haptics. Accessed: Jan. 16, 2025. [Online]. Available: https://titanhaptics.com/haptic-technology-101-a-beginners-guide-to-the-different-types-of-vibration-motors/

[13] University of Pittsburgh the McGowan Institute For Regenerative Medicine, “Prosthetic Hook Mouse for people with upper-limb amputations,” https://mirm-pitt.net/prosthetic-hook-mouse-for-people-with-upper-limb-amputations/ (accessed Oct. 20, 2024).

[14] L. M. Keita, B. Alcantara, and M. G. Salazar, “Wireless Mouse for people with upper limb
amputation,”://www.dlsu.edu.ph/wpcontent/uploads/pdf/conferences/ditech/proceedings/volume-3/paper-3.pdf (accessed Oct. 20, 2024).

APPENDIX A

A.1 GitHub Link

Main UFM Repository: https://github.com/westl5/UFM

- All code can be found in the *code* directory
- All PCB design information can be found in the *pcb* directory
- Other relevant documents such as datasheets, photos and videos can be found in the *docs* directory

APPENDIX B Capstone Expo Poster (See Next Page)

McMaster University | Electrical & Computer Engineering

UNIDENTIFIED FLYING MOUSE

ECE Group 10 (Jom & Terry) - George Gill - Luke West - Ryan Xu - Matt Yu

Motivation

Traditional computer mice are designed to give inputs to the computer from the user's hands. This makes it **difficult** for those with **limited hand dexterity**, those who have undergone **hand amputation** or those who were **born without a hand** to use the computer.

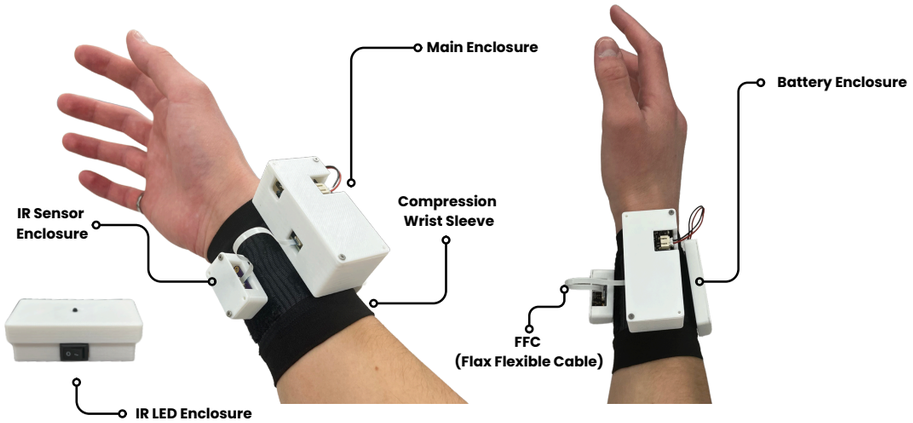
Objective

Design an **input device** which can be used as a computer mouse **without** requiring the **mouse hand**, with the goal of increasing computer accessibility for those who do not have the full use of their hands.

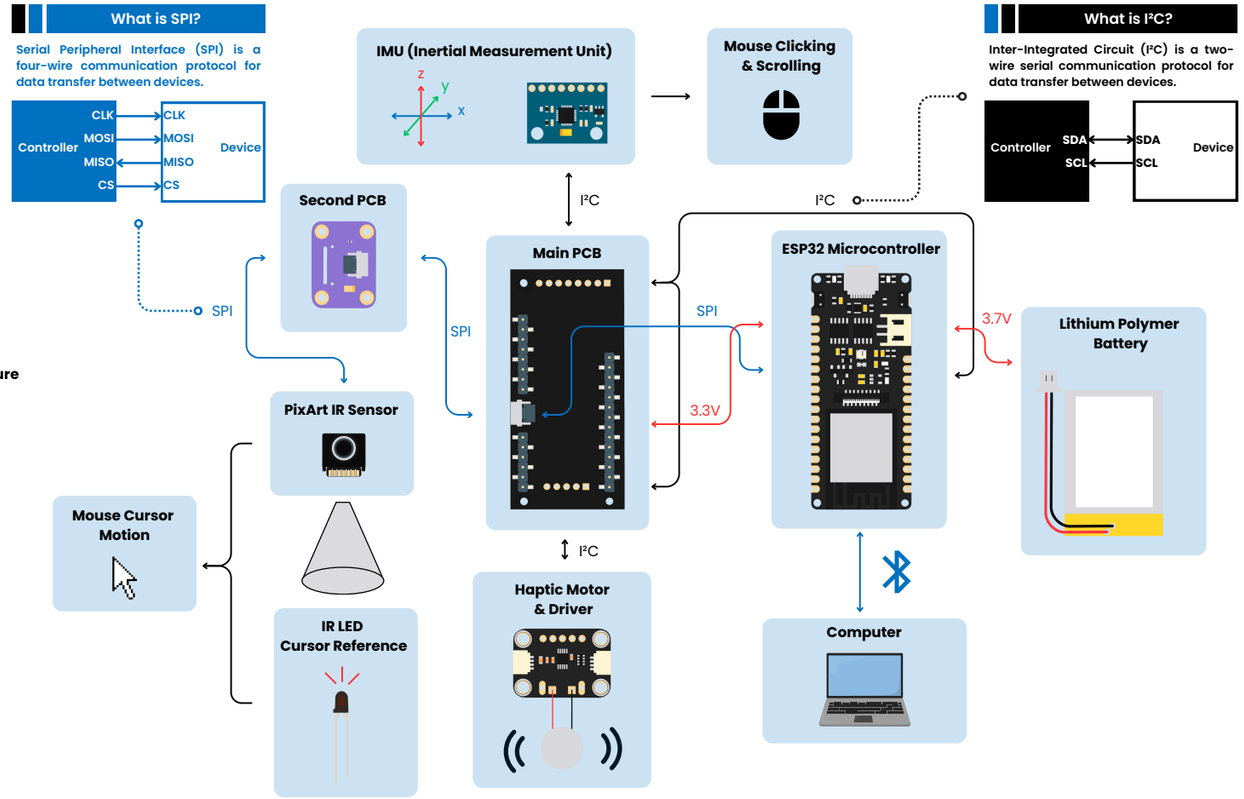
Our Solution

We have created a wrist-worn device which uses movements from the **arm** and **forearm** to perform computer mouse functions.

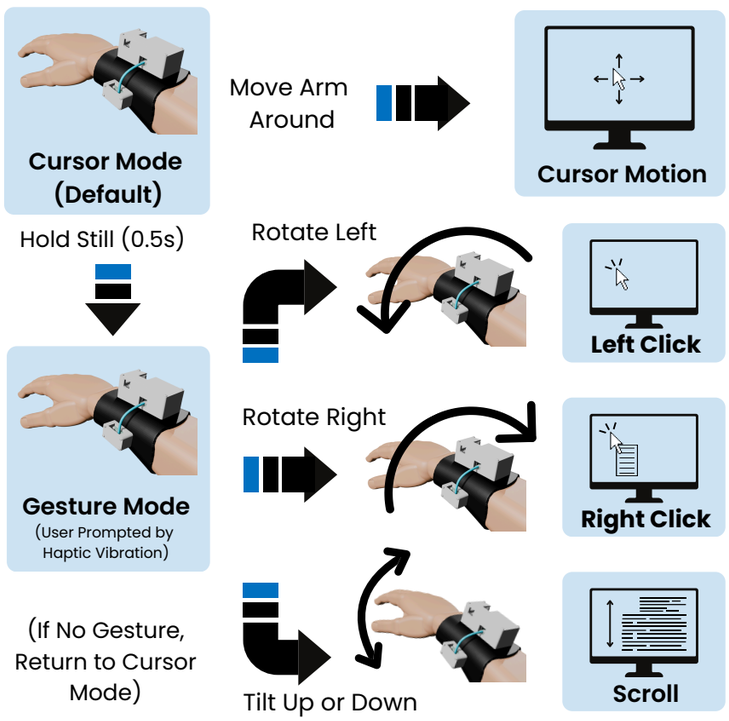
Our device uses an **IR LED** as a tracking reference, which sits in a compact enclosure on the user's desk.



System Overview



How Do You Use it?



How Does It Work?

- The PixArt **IR Sensor** Tracks an 850nm **IR LED** at 200fps and sends its **x-y coordinates** to the ESP-32 Microcontroller.
- Each coordinate is **compared** to the **previous** one and the **difference** is used to determine the **direction** and **magnitude** to of mouse movement to apply.
- If the mouse remains **still** for 0.5s, the user is prompted via **haptic vibration** to perform a gesture to click or scroll.
- An **IMU** measures **acceleration** and **angular velocity** to capture each gesture.
- The **mouse cursor** or **gesture** commands are sent to the PC by the ESP-32 over **bluetooth**.

Software Design Choices

- Codebase entirely contained to the ESP-32 meaning no extra software is needed from the user: Just **pair** with the device using **bluetooth** and the mouse is **ready to use**.
- Mouse movement calculation is very simple, allowing for low computation time and a very **responsive** mouse cursor.
- Cursor movement is separated from other functions with a wait time, limiting mis-inputs.

Physical Design Choices

- Small form factor and lightweight construction for a comfortable fit.
- **Compression sleeve** allows users with various wrist sizes to put-on the mouse using **only one arm**.
- Enclosures secured with **velcro** for easy **attachment** and **removal**.
- Battery power allows for **wireless** operation for an estimated **8hrs** with a 1100mAh LiPo battery.

References & Acknowledgements

[1] B. Trzynadlowski, "PixArt PAJ7025R2 6dof Tracking Demo with Arduino and Windows," GitHub. Accessed: Jan. 03, 2025. [Online]. Available: https://github.com/trzy?tab=repositories&q=PixArt&type=&language=&sort=

[2] University of Pittsburgh the McGowan Institute For Regenerative Medicine, "Prosthetic Hook Mouse for people with upper-limb amputations," https://mirm-pitt.net/prosthetic-hook-mouse-for-people-with-upper-limb-amputations/ (accessed Oct. 20, 2024).

[3] L. M. Keita, B. Alcantara, and M. G. Salazar, "Wireless Mouse for people with upper limb amputation,":://www.dlsu.edu.ph/wpcontent/uploads/pdf/conferences/ditech/proceedings/volume-3/paper-3.pdf (accessed Oct. 20, 2024).

- Special thanks to the **Thode Makerspace** for providing resources such as 3D printers and soldering equipment.

GitHub + Video