Department of Mechanical and Mechatronics Engineering

MTE 482

MoBIUS: An Alternative to Laptop Touchpads

Group 41
Nisarg Bhavsar, 20624568
Xiyuan Chen, 20611692
Shilpan Shah, 20603389
Lawrence Wu, 20602816

March 17 2020

203 Lester St.
Waterloo, ON
N2L 0B5
March 26, 2020

Jan Huissoon
Department of Mechanical and Mechatronics Engineering
University of Waterloo
200 University Ave W.
Waterloo, Ontario, N2L 3G1

Dear Professor Jan,

This report, entitled "MoBIUS: An Alternative to Laptop Touchpads", was prepared for MTE 482 at the University of Waterloo. Based off of our previous report for MTE 481, the design for a device aiding users in controlling their laptop's cursors in a space-confined area is to be finalized and implemented. The purpose of this report is to detail the design, construction, and testing that was done for the final product.

Following the needs assessment, problem formulation, constraints and criteria, and proposed design from last term, several modifications were done for the final design to reduce size and unreliability in the end product. This included the replacement of capsenses with buttons and the reduction of software from two separate applications to one. Construction and manufacturing was done through a Canadian PCB manufacturer and WatIMake's 3D printers, and despite problems with programming the board and getting Bluetooth to properly transmit, these issues were resolved in time for the project deadline. MoBIUS received a positive response from those we showed it to, despite suffering delays and being overbudget, and future development should focus on correcting the PCB and properly implementing gesture recognition capabilities.

This report was written entirely by us and has not received any previous academic credit at this or any other institution. We would like to acknowledge the support of Professor Jan, who consulted on our initial design, as well as David Perna, who greatly assisted us with troubleshooting our hardware.

Sincerely,

Nisarg Bhavsar 

Shilpan Shah 

Simon Chen 

Lawrence Wu 

Executive Summary

As the usage of laptop computers grows, so too does the amount of people using them away from conventional desks and tables. This leads to scenarios where possessing a laptop is advantageous, such as being able to get a head start on the day's work while commuting on the bus, but the user experience while working in such a space-constrained is not always ideal. This is especially true when there is not enough room for a conventional mouse which, for many people, is the preferred choice of input device over a touchpad. MoBIUS is an alternative input device that more closely mimics the operation of a conventional mouse to enhance the user experience when in a space-constrained situation.

The most important constraint of MoBIUS is the ability to operate without needing any additional flat surface. Potential solutions were judged based on criteria ranging from size, cost, and battery life. Three potential solutions were created: a trackball glove, a capacitive keyboard cover, and a ring-based wearable. The latter solution was chosen and expanded into a final design resembling a "half-sleeve" that rests on the user's finger.

The critical hardware components include the microcontroller (MCU), inertial measurement unit (IMU), and battery. Selecting these components were mainly based on compact size and longer battery life, and the original design that called for the usage of capsenses had to be altered due to concerns over size and complexity. The casing of MoBIUS was 3D printed in ABS plastic, and the software takes full advantage of the existing functionalities in the Bluetooth protocol to advertise, pair, and connect with laptops as a mouse-related device. Efforts were made to go beyond the original design and integrate gesture recognition into MoBIUS, but ultimately the project timeline did not allow for a satisfactory implementation of this.

Construction and testing of the final product was delayed due to multiple factors, including the coronavirus outbreak which forced PCB manufacturing to be done at a much more expensive Canadian company. Further issues arose during the soldering process, most notably the MCU refusing to connect to the programmer and the discovery that BLE transmission required an external clock that the PCB was not designed to have. Fortunately, these were resolved in time for the deadline. The final product was generally well-received by test users and many were impressed by the device's small size and easy integration with their laptop. MoBIUS was able to meet many of the constraints and criteria that were laid out at the start of the project, and future development should focus on correcting the PCB layout as well as improving the work done for gesture recognition.

Contents

List of Figures

List of Tables

1 Introduction and Background

Laptop computers have grown ubiquitous due to their portability and affordability, and for some consumers, especially students and tech workers, they have almost supplanted traditional desktop computers entirely. People are now able to work on their computer in a variety of spaces that extend well beyond the traditional desk environment, thereby increasing their productivity. For instance, it is not uncommon to see people using their computers while they are taking the bus or subway to and from work; the daily commute is regularly cited as one of the biggest wastes of time in the working day, and having the ability to get a head start on the day's tasks or clean up a project on the way home is a great way to counteract this.

1.1 Needs Assessment

The portability of laptop computers opens up unique usage opportunities for their owners, but the associated user experience is not always the best. In particular, users are almost always limited to a touchpad when it comes to cursor input and control. Though some manufacturers have experimented with novel solutions such as laptop-tablet hybrids (i.e. Microsoft's Surface Book) and joystick-like studs in the middle of the keyboard (i.e. Lenovo's TrackPoint), most laptops follow the model set by Apple's Macbooks with a single buttonless touchpad below the keyboard. This form of input is certainly usable, but the quality of touchpads differs from manufacturer to manufacturer - many of them fall flat of the Apple touchpads that they attempt to mimic - and many people still prefer the comfort and speed of a regular mouse whenever they can use one. Thus, a problem arises when one wishes to use their laptop in a setting such as a bus or subway where there is not enough room for a conventional mouse. The prevailing attitude is to simply tolerate the touchpad, regardless of the user's preferences, or attempt to use a mouse within the limited flat surface available at the lower-right corner of the laptop. Both of these methods are sub-optimal, and there are not a lot of devices available that give consumers a dedicated alternative to the touchpad when they are in these space-constrained situations.

1.2 Problem Formulation

The aim of this project is to develop an input device for users to quickly and comfortably manipulate the cursor on their laptop in space-constrained situations. It will emulate a conventional mouse experience more closely than a touchpad, thereby improving the user's experience when working on their laptop in a variety of areas.

1.3 Constraints and Criteria

The most basic constraint for the device is that it can operate without needing any additional flat surface. This is self-explanatory, as its entire purpose is to serve as a form of cursor input and control in places that lack the space for a conventional mouse. However, this constraint does not exclude on-board solutions that use the area already taken up by the laptop. For example, a device that utilizes the screen or keyboard is still valid since it does not increase the total space required.

The device must also be able to function within a 1 metre radius of the user's laptop. This is not expected to be a difficult constraint to meet; given the fact that the user should be in front of their laptop at all times, a 1 metre radius should be more than enough for regular usage.

A final constraint is that the device's power source must be sufficient for a minimum of one hour of continuous cursor control. While this length of time is on the low side, it is considered suitable for the purposes of prototyping and covers at a minimum most cases where the user would be working on their laptop in a space-constrained area.

When it comes to the size of the device, a smaller design is preferred in order to complement the portability that is expected when using a laptop. The size should be less than or comparable to a typical computer mouse since it would be foolish to build an input device that is bulkier than what already exists.

Most existing products that fulfill a similar need to this project are sold for about $150. Some of these existing products are discussed in more detail in the following section. Thus, it is preferred that the per-unit manufacturing cost of the device is less than $100.

To further improve user experience, it is preferred that the device is easy to learn and get accustomed to. A new user should be able to effectively use the device within 15 minutes of picking it up for the first time.

Another important part of user experience relates to latency, which can be described as the time it takes for a user action to translate to a visible response on the laptop screen. 100 milliseconds is commonly cited as the maximum amount of delay before the user begins to notice and interpret a sluggish response, so the device should keep its latency below 100 milliseconds [3].

As mentioned earlier, the device must meet the constraint of a minimum of one hour of continuous cursor control before running out of power. Of course, it is preferred that this length of time on one charge be as long as possible.

The last criterion for the device is the effective input area that the user can operate in. This specifically deals with how easily the user can switch between typing and cursor control; if a design allows the user to switch almost anywhere they want, then it will be preferred over a design that forces the user to move their hand away from the keyboard. While this behaviour is not exclusive to the scenario of using a laptop in a space-constrained area - for instance, Lenovo's TrackPoint is designed to let the user keep their fingers on the home row of the keyboard - it can further improve the user's experience with the device.

1.4 Design Review

1.4.1 Potential Solutions

Three designs were suggested to begin with: a trackball glove, a capacitive keyboard cover, and a ring-based wearable. The first design was based off traditional trackball mice where there are built-in encoders that convert the rotational motion of the trackball to cursor movement. In the glove-based design, the trackball is placed at the wrist of the user so they can exploit the free area outside of the keyboard on their laptop to manipulate cursor movement while typing or resting their fingers on the keyboard.

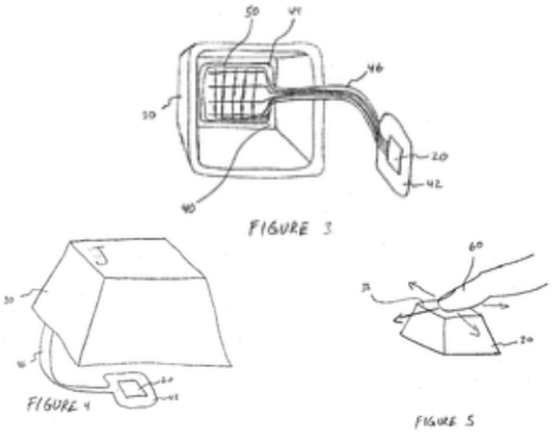


Figure 1: Capacitive keypad patent [1].

The capacitive keyboard cover design was created during the course of patent research. Patent US20040041791A1, shown in Figure 1, describes a capacitive touch array being attached to the underside of a single keypad. Positional information about the user's finger can be determined based on the activation response of the capacitive touch array, which can then be used to move the cursor of the computer. The design would expand on this patent to have a flexible keyboard cover acting as one massive touch array, allowing the user to provide input anywhere they wanted.

Figure 2: The Mycestro.

The third design was inspired by the Mycestro, shown in Figure 2, which is marketed as a "3D wearable, gesture-control mouse" with emphasis on ergonomics and the ability to "work desk-free"[4]. Thus, it is one of the best examples of an existing product that gives users an alternative to the touchpad and it is branded as "the only wireless mouse on the market that's wearable". The Mycestro tracks the motion of the user's finger with an IMU and directly translates it to cursor movement on the screen, and three physical side buttons are used for left, middle, and right clicks; scrolling is performed using an embedded touch array below the buttons.

The ring-based wearable design is akin to the Mycestro, but in a smaller and lighter form. To do this, gesture recognition would be used to remove the need for large physical buttons, and Bluetooth Low Energy would be used over regular Bluetooth to help preserve battery life and allow the option of using a smaller battery. The ring would also directly connect with the laptop's Bluetooth receiver instead of a dongle to further reduce the amount of hardware involved. The main challenges with a solution like this would be keeping battery life to acceptable lengths and the miniaturization of hardware due to size being a significant limiting factor.

1.4.2 Evaluation of alternative designs

	Size	Manufacturing Cost	Ease of Use	Latency	Battery Life	Effective Input Area	Total (weight * rank)
Weight	5	2	4	3	5	5	-
Trackball Glove	3	3	5	4	4.5	3	91.5
Keyboard Cover	2.5	1.5	3.5	4	4	4	81.5
Ring Based System	5	2.5	5	4	2.5	5	99.5

Table 1: Decision Matrix.

Table 1 shows the decision matrix used to evaluate the three solutions. Each criteria is given a weighting from 0 to 5 and each solution can score 0 to 5 for a criterion. In general, size, ease of use, battery life and effective input area are given the highest weightings since these are the core design targets the design solutions are trying to achieve. A desired solution should be small and long-lasting while allowing users to freely manipulate cursor movement without being restricted by their surrounding space. Manufacturing cost is weighted the least since it is not the top priority with the product only being in development and prototyping phase. By comparing the decision matrix, it can be seen that the ring-based solution has the highest overall score of the three proposed solutions. The ring-based solution is the smallest solution and provides a high degree of freedom for user input while still being easy to use. This is despite trade-offs in battery life and manufacturing cost; the battery size is more restricted and electrical components are more expensive when in small sizes.

1.5 Proposed design

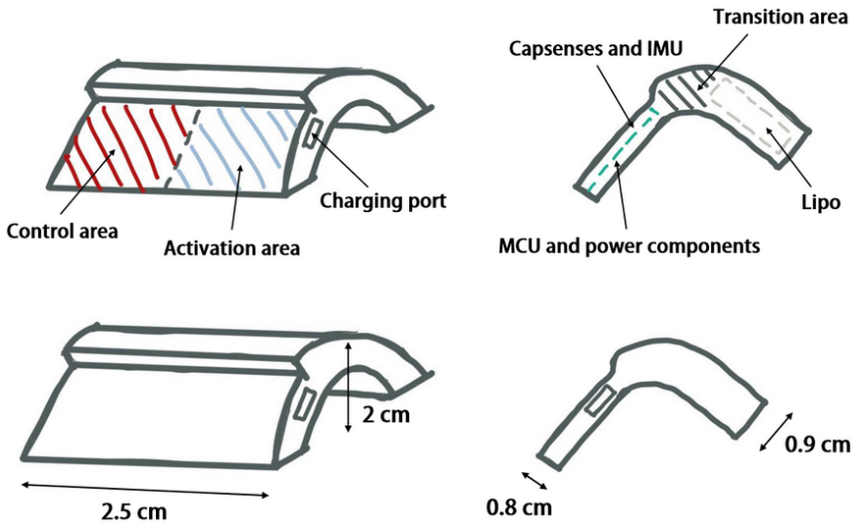


Figure 3: Sketch of final device design [2].

Figure 3 shows a simple sketch of the final proposed design. While the original idea was deemed a "ring-based" solution, it was later determined that making the device as small as a literal ring would pose significant difficulties with manufacturing and battery life. Thus, the design was enlarged to be more of a "half-sleeve" that would rest on the user's finger and be secured by a strap (not pictured). One side would contain the electrical components and sensors, including the IMU that would read 3D movement to be mapped to 2D cursor movement, while the other side would contain the battery. The "transition area" in between would contain the wiring between the battery and the PCB and join the two sides together. The device would also have two distinct areas of user control: an "activation area" that the user would touch to start and stop manipulating the cursor, and a "control area" that would be used for actions such as clicking.

While this proposed device bears some physical similarities to the Mycestro, it takes active steps to improve upon the Mycestro's design. The removal of physical buttons allows the size of the device to be reduced, which also corresponds to a lower hardware and material cost. Having the device connect directly to the computer's Bluetooth receiver rather than through a Bluetooth dongle is also an improvement; given the current trend of laptops being manufactured with fewer and fewer ports, eliminating the need for a dongle is a significant improvement for the user.

Last but not least, the device was named "MoBIUS", derived from the image of a Mobius loop and the first two letters of "mouse".

2 Electrical Design

Figure 4 below shows the high-level electrical architecture with various blocks to be considered when designing the electrical circuit/system of MoBIUS. More detailed schematics and bill of materials (BOM) can be found in the Appendix.

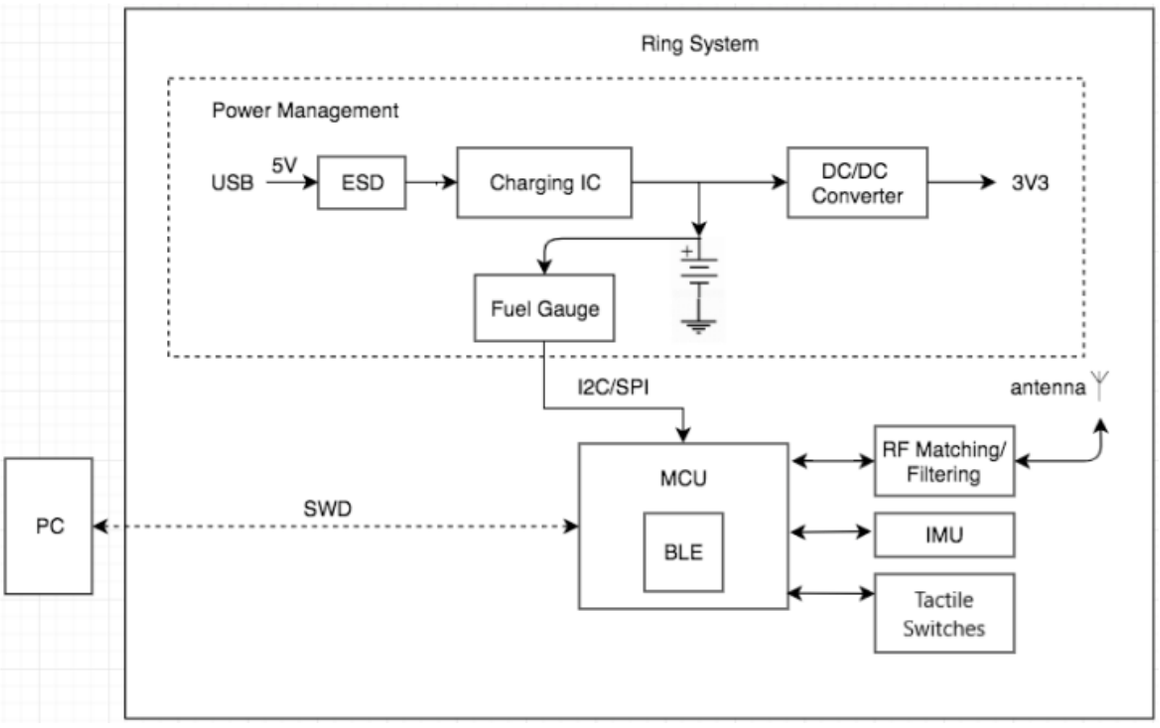


Figure 4: Electrical block diagram [2].

The design generally consists of two major functional blocks: a power management block and a system processing block. The power management block regulates and supplies power to the entire system. It receives an input voltage from external supplies through a micro USB port and to a charging integrated circuit (IC) that controls the charging rate of a 100 mAh lithium polymer (LiPo) battery. Meanwhile, a fuel gauge will be used to continuously monitor the state-of-charge (SoC) of the battery. During normal operation, the battery voltage is stepped down by a low dropout (LDO) regulator to output a steady 3.3V to power the rest of the system, including the micro processing unit (MCU) and other necessary peripherals. In addition, an optional ESD component may be used to provide electrostatic discharge protection to the system depending on the electrostatic discharge (ESD) ratings of the components.

Regarding the system processing block, the MCU would be reading data from the IMU and continuously check the states of the tactile switch array to detect user gesture inputs, then process, package, and transmit data through its integrated RF layer to a receiving laptop. Moreover, an integrated on-chip RF matching and filtering circuit is used to maintain good signal integrity during data transmission.

The system is programmed using the Serial Wire Debug (SWD) protocol. This protocol is chosen due to its programming and debugging capabilities, as well as its advantage of being space efficient in PCB layout as it only requires 2 signal lines for programming and debugging.

2.1 Touch Sense Array Replacement

Compared to the initial design proposal, the major modification that occurred for the final product was the replacement of the capsense-based touch sense array with a tactile switch array. After designing the linear diplexed capacitive slider, it was found that its footprint takes up about 40% of the entire board size. While it is possible to fit all the other components along with the capacitive slider in a compact form onto the PCB, all of these layouts would inevitably result in negative interference between either the RF or the capacitive slider with other on-board electrical components. Because of the board size restrictions, component clearance requirements of the capacitive slider and the RF module would be difficult to achieve. This would lead to either faulty position detection and interpolation on the capacitive slider due to extra parasitic capacitive loading from neighboring electrical components to the slider, or degraded performance of the RF module due to insufficient RF ground size. Thus, in order to properly implement the capacitive slider while maintaining optimal performance conditions for all components, the only option is to increase the PCB board size, which leads to another issue discussed below.

There are mainly three possible ways to increase the effective PCB size to make room for the capacitive slider: directly expanding the area of a rigid PCB, making a flexible area for the slider on a Flex-PCB, or make a second rigid PCB solely for the slider and link it to the main board through connectors. Directly increasing the rigid PCB area is not an option due to the hard size constraints of the product, and the other two methods directly relate to dramatic increases in production time and cost. These are issues that cannot be ignored due to project time and budget constraints. In addition, it may be unreasonable to start with such a high development cost for a prototype that needs further testing and verification.

For all the above reasons, it was decided to replace the capacitive slider, or the capacitive touch array, with an array of tactile switches and buttons to compliment the IMU for detecting motion-based inputs from the user.

2.2 Final Component Selections

The main system specifications includes the followings:

- Current consumption: 15.4 mA
- Input Voltage: 4.2V - 5V
- System Operating Voltage: 3.3V
- Bluetooth and motion measurement capabilities

	STM32WB55xx	STM32L4x2
Size	8 x 8 mm	5 x 5 mm
Processing Power	• Arm 32-bit • Processing: cortex M4, 64 MHz • Radio: cortex M0	• Arm 32-bit • Processing: cortex M4, 80 MHz
Current Consumption	600nA Standby, 8.1mA with RF	28nA Standby, 2.3 mA (operating)
Flash	1 MBytes	256 KBytes
Input voltage range	1.71 V to 3.6 V	1.71 V to 3.6 V
Supported Protocols	I2C x 2, SPI x 2, USB 2.0 FS, Integrated RF Transceiver (e.g BLE V5.0), USART x 1	I2C x 2, SPI x 2, USB 2.0 FS, USART x 3
Other	6 ~ 18 capacitive sensing channels, integrated SMPS. Integrated BLE	CAN bus. Requires separate BLE chip.

Table 2: MCU options [2].

Tables 2, 3, 4 (the latter two are on the following page) show the summary of a list of components that are critical for system operation. As a result of component comparison with the tables and research online, an LDO with 250 mA supply current and 3.3V output voltage, a 70 mAh ASR00011 LiPo battery, an STM32WB MCU with integrated BLE, and an LSM9DS IMU with 9 Dof are selected. However, due to international battery shipping restriction, the ASR00011 battery had to be replaced with a slightly bigger LiPo battery of 100 mAh. See Bill of Materials in the appendix for a detailed final list of selected components.

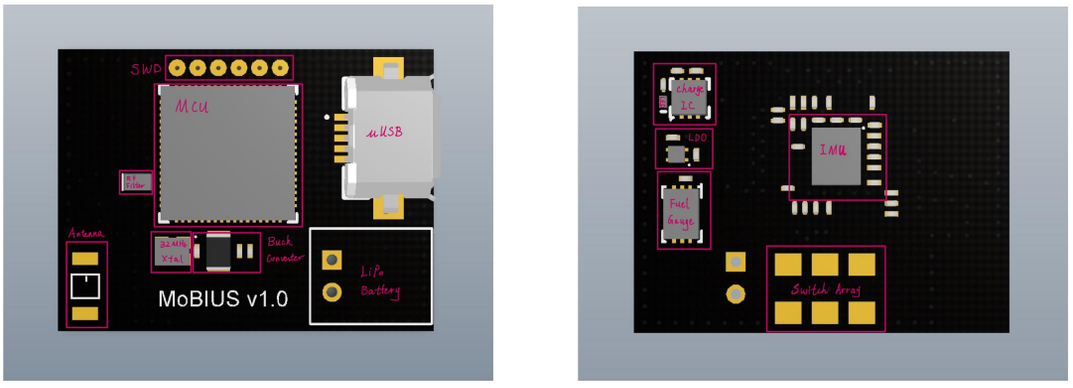
	LSM9DS1	LSM6DS3	BNO055
Size	3.5 x 3.1 mm	2.5 x 3 mm	3.8 x 5.2 mm
Supply voltage	1.9 ~ 3.6V	1.71 ~ 3.6 V	2.4 ~ 3.6V
Current consumption	• Accel: 600 uA • Gyro: 4 mA	Accel + Gyro: 1.25 mA	12.3 mA
DoF	9-axis	6-axis	9-axis
Data Type	16 bit	16 bit	16 bit gyroscope 14 bit accelerometer
Protocols	SPI/I2C	SPI/I2C	I2C/UART
Other	Integrated ML core	Integrated ML core	On board sensor fusion

Table 3: IMU options [2].

	Product No.	Size	Thickness	Voltage	Capacity
Coin cell	ML-2020/V1AN	2 cm x 2 cm	2 mm	3 V	45 mAhr
Lithium Polymer	LP221220	1.2 cm x 2 cm	2.2 mm	3.7 V	25 mAhr
	LP221619	1.6 cm x 1.9 cm	2.2 mm	3.7 V	30 mAhr
	LP242025	2 cm x 2.5 cm	2.4 mm	3.7 V	75 mAhr
	ASR00011	1.6 cm x 1.5 cm	5 mm	3.7 V	70 mAhr
Curved Lithium Polymer	LPCV381447	3.08 cm, 3.46 cm (inner, outer) Radius	3.8 mm	3.8 V	235 mAhr

Table 4: Battery options [2].

2.3 Final PCB Overview



(a) PCB top view

(b) PCB bottom view

Figure 5: Final PCB layout.

Figure 5 above shows the final PCB design for MoBIUS (refer to Schematics in the Appendix for more detailed hardware implementation). Its dimensions are: 2.3 cm x 1.7 cm x 0.8 cm, which is about the size of a single phalanx, or finger joint. The 6 SWD programming pad at the top of the board creates a simple interface for uploading firmware to the MCU. During normal operations, the LiPo powers the entire system; during charging, external power comes in from the micro USB to charge the LiPo battery, whose status is indicated by a red LED near the charging IC, and powers the rest of the system. A 32 MHz crystal is used to supply clock sources to the system, ensuring stable performance for various communication lines such as the I2C communication between the IMU and the MCU and the BLE data transfer between the MCU and the client laptop.

3 Mechanical Design

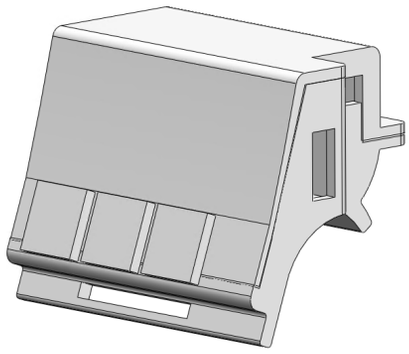


Figure 6: Final casing design.

As seen in Figure 6, the basic shape of MoBIUS's casing is similar to what was originally proposed: a long curved piece of plastic that rests on the user's finger and is secured by a strap. However, after the hardware components were modified, it was necessary to adjust the specific dimensions of the casing. Most noticeably, because the capsenses were replaced by offboard buttons, there was no need to place the PCB on the opposite side from the battery. A much simpler arrangement with the PCB stacked on top of the battery was the result, and though this increased the height of the casing, it gave much more freedom in regards to the exact shape and curve since most of the electrical components were concentrated in the centre of the device rather than being spread out like in the original design.

To accommodate as many users as possible, the curve of the casing itself was designed based off an MIT study that found the average width of the human index finger to be 1.6 to 2 cm [5]. For extra padding, the base radius of curvature was selected to be 2.2 cm. The arc length was kept to a minimum to avoid wrapping extensively around the user's finger and thereby limiting the sizes that it could accommodate, and it is approximately equivalent to a third of a circle before transitioning to tangential lines. Meanwhile, a Velcro strap ensures the adjustability of the fit around the user's finger and provides a comfortable cloth-like feel on the skin as opposed to hard plastic.

The casing is designed to be manufactured in two halves, top and bottom, which are secured together using 2 mm diameter screws and nuts. The back of the casing has a mounting rim for this purpose, and at the bottom are two slots that the Velcro strap slides through. Finally, there are holes to accommodate the switch, USB port, and charging LED, illustrated by Figure 7 on the next page. Dimensioned drawings are available in the Appendix under As-built Drawings.

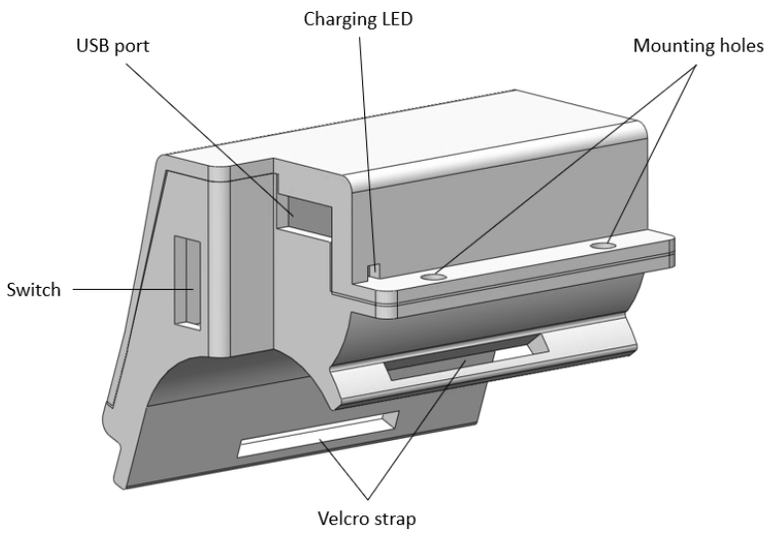


Figure 7: Back view of casing with labels.

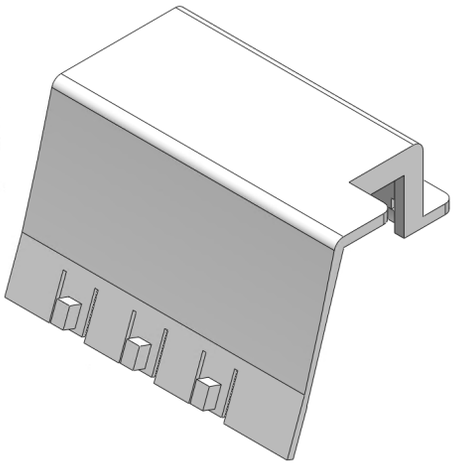


Figure 8: Casing top with exposed button cover.

The most notable feature of the final design is the integrated button cover. MoBIUS's casing has three thin flaps cut into it, each corresponding to one button and designed to be flexible enough to be able to bend and press down on the button when the user wishes to. The larger cover pieces that the user presses on are glued to these flaps rather than the buttons themselves, which makes assembly of the device much easier and avoids the possibility of wasting hardware components due to a design change in the cover.

4 Software Design

For MoBIUS to function properly, the required software can be split into two parts. The first part is the embedded software that runs on the MCU, which allows it to communicate with the IMU using the I2C communication protocol. Reading, filtering, and interpreting the IMU data into cursor movement and actions are all the responsibility of the embedded application, along with handling input from buttons and carrying out Bluetooth advertisement and transmission. The second part of the software is on the laptop that MoBIUS connects to; it needs to be able to carry out the Bluetooth pairing and connection process and manipulate the cursor based on the data that is received over Bluetooth.

The original design for MoBIUS's software called for two applications to be written: a sender application on the MCU that takes care of the embedded software, and a receiver application on the laptop that takes care of cursor movement. However, it was discovered after further research that the concept of the receiver application could be entirely replaced by the built-in Bluetooth functionality of the user's laptop without any additional software. In Bluetooth terminology, peripheral devices like MoBIUS that advertise themselves are called "servers" while devices that search for and connect to them are called "clients" [6]. A server includes specific information in their advertisement that tells clients what kind of services they can provide, which can be anything from biometric sensing to location tracking, and this information gives clients a basis as to how they should interpret the data they receive from the server. With MoBIUS, the most relevant service is the Human Interface Device (HID) service, which - after being paired and connected - allows the client laptop to automatically recognize MoBIUS as a mouse-related device and correctly manipulate the cursor based on the received data. Thus, the project only needed to be concerned with the embedded software on the MCU.

Software development was done using the STM32CubeIDE, an integrated development environment provided by ST for use with its microcontrollers. Among other things, the CubeIDE is capable of generating boilerplate code based off settings in an .ioc file, which includes setting up pin and clock configurations for the MCU. This code generation significantly reduces the time needed to get the MCU working and allows developers to focus on code unique to their application.

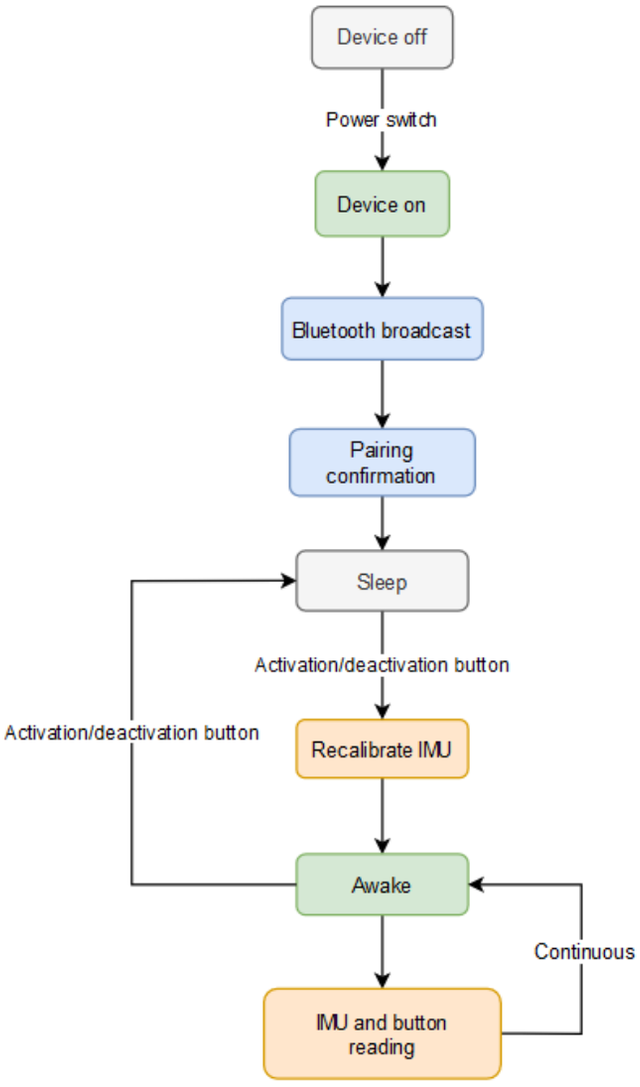


Figure 9: Power states.

Figure 9 shows the state machine that is implemented on the MCU. It is intended to allow MoBIUS to cycle between sleep and awake states based on user input in order to conserve battery life. When awake, the device continuously polls the IMU and handles button input for the HID updates that it will send to the laptop. When the user no longer needs to use the cursor, they can choose to deactivate the device which stops the Bluetooth transmission and puts it back into the sleep state.

4.1 Bluetooth

Bluetooth functionality requires a series of consistent repeated actions - for instance, repeatedly broadcasting an advertisement for clients or sending updated data at fixed intervals. For this reason, MoBIUS's embedded software uses several timers that are each linked to a specific event. When the device is turned on, one timer broadcasts an advertisement every 80 milliseconds until a connection with the client laptop is established or a minute passes; the 80 milliseconds is the recommended interval for "fast" advertisement. Meanwhile, another timer is responsible for sending a cursor update to the client every 50 milliseconds while the device is active. This time interval is much shorter in order to reduce the latency experienced by the user.

The Bluetooth protocol also includes support for secure connections between devices, and servers may be set up to give an authentication prompt to clients who are attempting to pair with them. However, most Bluetooth peripherals that people are familiar with do not include this feature, opting instead for a "Just Works" authentication model where pairing is a one-click process [7]. Given that the only data it works with is related to cursor movement, it was decided that MoBIUS would also use Just Works authentication for simplicity and convenience.

4.2 Button Interrupts and Debouncing

Besides timers, interrupts are the other driving force behind MoBIUS's software. Each button triggers an interrupt on its rising and falling edge - in other words, when the user presses and releases - which are interpreted differently based on the button. For the two buttons that are linked to left and right clicking, referred to as SW1 and SW3 respectively, each interrupt changes a boolean value that represents if the button is currently being pressed or not. The corresponding left and right click values that are sent with the HID update are based off these booleans, and this allows the user to perform actions that require a button to be held such as clicking and dragging.

The button that controls switching between tracking and scrolling modes, referred to as SW2, also has a boolean representing if it is being pressed or not. However, it has some additional logic to it. Switching between tracking and scrolling requires the user to double click SW2, as opposed to a single click which activates or deactivates the device, and thus there must be a way to differentiate between the two. Intuitively, a double click can be recognized as the user pressing SW2 shortly after releasing it. As a result, whenever an SW2 interrupt is received, a timestamp of the most recent instance of SW2 being pressed or released is updated based off the boolean. If the current interrupt is for a press and it has arrived within 100 milliseconds of the last release, it is considered a double click and causes a mode switch.

Interrupts are not flawless, however. Due to their mechanical nature, it is possible for a button to rapidly transition between its open and closed states during a single press or release, which causes multiple interrupts to be sent to the MCU. This phenomenon is known as bouncing; consequently, debouncing is a common software technique used to address this problem. After an interrupt is received, any subsequent interrupts that occur within a certain timespan will be ignored, preventing the possibility of rapid changes in behaviour due to multiple interrupts triggering one after another. MoBIUS's software implements debouncing for each button by recording the timestamp of the last interrupt that was considered to be valid. Any interrupts that originate from that same button in the next 20 milliseconds are ignored.

4.3 IMU Calibration

The movement of the cursor is based upon careful readings of the IMU. Acceleration and gyroscope data is used to perform the movement of the cursor, scroll along a webpage and detect well-defined gestures. The IMU is placed onto the user's finger, and the user switches between typing on the keyboard and moving the cursor. During use, the user's hands are to be kept on the keyboard; however, the orientation of the fingers can change. It is a function of the last word typed by the user, the last interaction with the keyboard, and hence is unpredictable. To ensure the same level of control is achieved every time the user switches from typing to manipulating the cursor with MoBIUS, a calibration of the IMU is performed at each switching instance.

It is noted from before that a well-defined threshold for the perception of zero-latency in most human-computer interaction is roughly 100 ms. Actions initiated by the user which are acted upon by the computing interface within 100 ms are perceived to occur immediately after initiation. This was found to be especially true for tasks requiring hand-eye coordination, such as that of having fine control of the cursor.

Multiple calibration procedures were attempted. A simple yet effective calibration was found to simply read acceleration and gyroscope data upon state transition, calculating a weighted average. This weighted average is then used to offset subsequent readings to allow the user's finger to be at a relative origin regardless of absolute orientation. A weighted average was used to take into account already present accelerations when the state transition is initiated. The weights skew more recent readings of acceleration and orientation more heavily, leading to a better calibration value experimentally.

4.4 IMU Filtering

A series of different filters were tested for the IMU data channels. The goal of this filtering was to reduce the jitter associated with the acceleration channels and the associated drift. The drifting of the IMU was rectified through the quick calibration process explained above.

A moving average filter was tested, which reduced the jitter to an acceptable level while being very simple to maintain. A circular buffer was implemented wherein new IMU readings would overwrite the oldest data value in the buffer. An efficient macro was implemented to take the average of the buffer in order to achieve the required minimal processing delay.

4.5 Gesture Recognition

To add further functionality to MoBIUS, gesture recognition was attempted. It was proposed that gestures could be used to control higher level functions that the user can customize, such as switching tabs in a browser. The high level idea is to capture a certain amount of time-series data to allow for a block of data to be classified into one of four gestures: tap, double tap, right swipe, or left swipe. It is imperative that the classification of the gestures does not interfere with the low-latency transmission of IMU data for cursor manipulation.

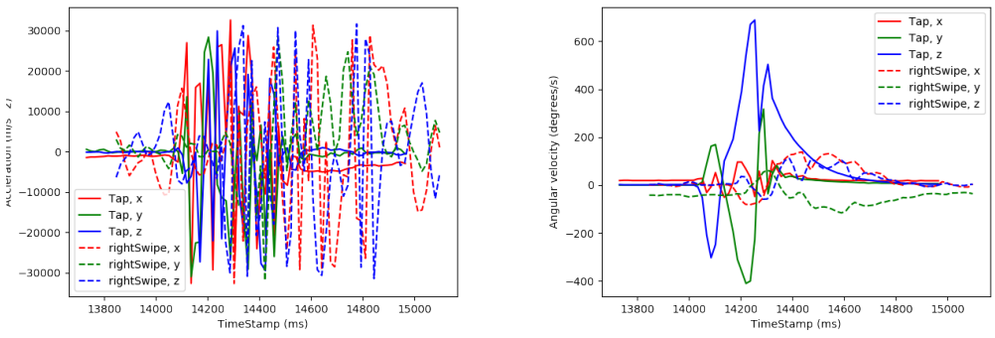


Figure 10: Representative IMU Training Data

Representative data can be seen in Figure 10. Gesture data was captured using the final prototype of MoBIUS, ensuring minimal slippage on use by trial participants. Participants were requested to perform tap, double tap, right swipe and left swipe gestures in a 1 second data capture window.

Figure 10 shows the acceleration and angular velocity data captured from the IMU for a tap and right swipe gesture performed by the same participant. It can be observed that that the acceleration data is very noisy, the right swipe gesture having significantly more peaks and troughs. This is countered by distinctly different profiles of angular velocity between the two gestures. This initially provided the basis by which gestures can be differentiated.

Keeping latency in mind, it was experimentally determined that 70 samples of consecutive IMU samples were needed to classify gestures. One data point is therefore 70 samples of 6 channels, resulting in a data point having 420 dimensions.

In an effort to reduce the dimension of the data, principle component analysis was used. Principal component analysis converts a set of data points into a set of linearly uncorrelated variables called principle components. This serves to reduce the number of dimensions of each data point, while simultaneously maximizing variance along each of the remaining dimensions.

The scikit-learn API [8] was used, and its implementation of principle component analysis was leveraged.

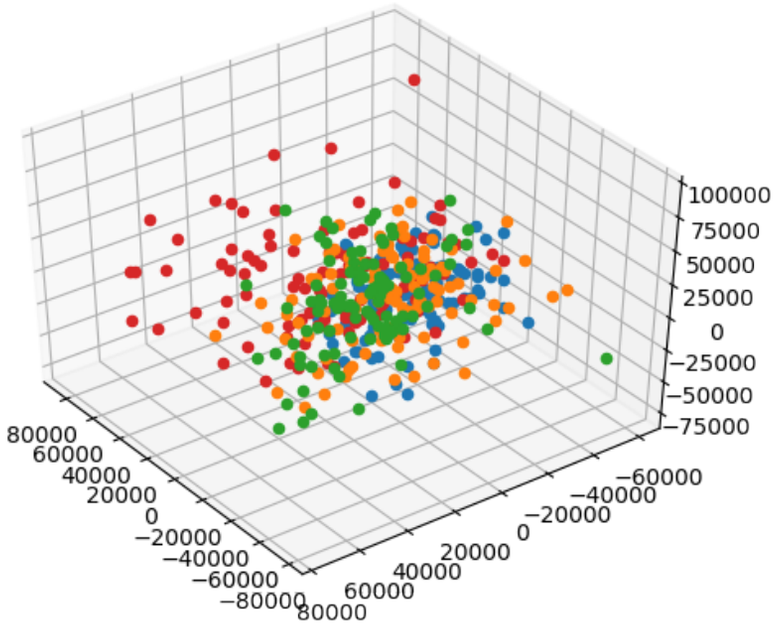


Figure 11: Reduced Dimensional Gesture Training Data

0.02084204	0.02038188	0.01883709

Table 5: PCA Variance Ratios

Figure 11 shows the reduced dimensional training data. The dimension of the training data was reduced from 420 to 3, in an effort to reduce the amount of complex processing done on the MCU. The variance ratio for each of the three principle components can be seen in Table 5. It can be seen that the variance ratios are similar, and small. This does not bode well for being able to differentiate between gestures. A small variance ratio indicates a small variance of data along a particular principle axis. This can be more clearly seen in Figure 12.

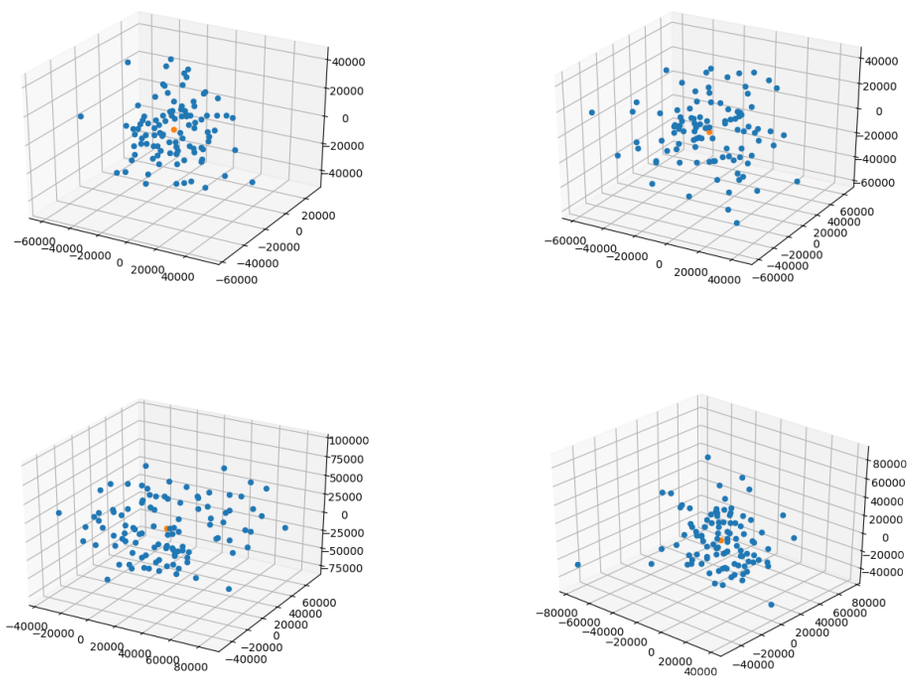


Figure 12: Reduced Dimensional Gesture Training Data

Figure 12 shows 4 sub-graphs, each with a single gesture's reduced dimension training data-set. The center of mass of the data-sets are also shown in Table 6. It can be seen that there is a significant difference between the positions of the center of masses of the four different gestures.

Tap	-8795.3851308	10268.10047875	-2075.70910648
Double Tap	-4537.65757445	-4326.99345011	-3129.26264561
Right Swipe	-8213.47668195	8618.31328849	4496.40978954
Left Swipe	20723.40237662	5928.50459427	744.5030303

Table 6: Center of Masses of Reduced Dimensional Gesture Training Data

Instead of relying solely on the PCA to segment out the gesture classes, it was decided to attempt to use a euclidean distance measure to classify the gestures. The idea is to capture 420 samples, reduce the dimensionality to 3 (a quick operation using the already trained PCA), and compute the euclidean distances from the new data point to each of the gesture's center of masses shown above. The closest center of mass is used to classify the data point. The euclidean distance formula is shown in Equation 1. A measure of the similarity of the data points to the center of masses of the gestures is shown in Equation 2.

The training data had 100 instances of each gesture. Another validation dataset was used, which had 100 instances of each gesture.

$$d_{p_1,p_2} = \sqrt{\sum_n (p_{1,n} - p_{2,n})^2} \tag{1}$$

$$ds_{p_1,p_2} = \frac{1}{1 + d(p_1, p_2)} \tag{2}$$

Table 7 shows the confusion matrix resulting from the use of the euclidean distance as a method of classification. The rows represent the actual label of the class, the columns represent the classified label based on the distance similarity measure described above. It can be seen that tap and double tap can be readily discriminated from each other as the input tap validation gestures were classified correctly 65% of the time. Double tap gestures were classified correctly 67% of the time, while right swipe and left swipe gestures were classified correctly 69% and 68% respectively.

	Tap	Double Tap	Right Swipe	Left Swipe
Tap	65	17	10	8
Double Tap	9	67	14	10
Right Swipe	2	1	69	28
Left Swipe	5	8	19	68

Table 7: Euclidean distance confusion matrix.

The ability to discriminate between the training and test gestures is promising. However, the gesture recognition was not implemented into the firmware of the MoBIUS. User testing revealed that the misclassification of gestures resulted in undesirable complications affecting the movement of the cursor.

Firstly, the lag in cursor movement was noticeably increased from below 100 ms to roughly a point wherein the cursor delay was noticeable. Additionally, when a gesture was detected, the cursor would often be past the point at which the gesture was to be applied. For example, if the tap gesture was mapped to left clicking, it was difficult for the user to perform the tap gesture and have it correctly recognized without also moving the cursor. This is due to the same input device being used for multiple purposes at the same time. As such, gesture recognition and mapping to higher level input directives was removed from the final prototype. Snippets of the final code can be found in the Appendix under Firmware Excerpts.

5 Construction and Manufacturing

5.1 PCB Manufacturing

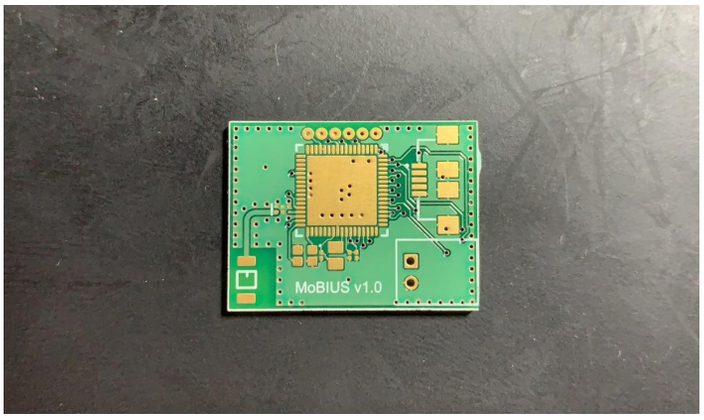


Figure 13: The manufactured PCB.

The process of obtaining a custom PCB is as simple as contacting a manufacturer, sending them the relevant schematics, and waiting for them to fabricate and ship back the board. Unfortunately, the winter of 2020 proved to be an especially bad time to place a PCB order due to the outbreak of the coronavirus in China. With Chinese-supplied companies unable to provide an estimate of how long their turnaround times would be, many of the cheapest options for PCB manufacturing were unavailable for the term. While there are alternatives based in North America, they are typically much more expensive than their Chinese counterparts. There was little choice, however, and after contacting several companies and asking for quotes on the expected price and turnaround, it was decided to use Candor Industries, a Toronto-based manufacturer, for the PCBs due to their close proximity to Waterloo.

5.2 3D Printing

The casing was manufactured using WatIMake's 3D printers and ABS plastic filament, shown in Figure 14 on the next page. While the first prototype was done in PLA plastic, the switch to ABS was made soon after due to ABS's increased flexibility and durability compared to PLA. However, ABS has an increased tendency to warp compared to PLA, so greater care needed to be taken with the quality control process. This largely consisted of visual inspection for warping and a mock assembly of the casing; areas that had difficulty fitting together were first sanded down, and if they still had trouble meeting the design, a reprint was required. The STL files used for 3D printing were also modified from the original design to include tolerances around sensitive areas such as holes and mating surfaces.

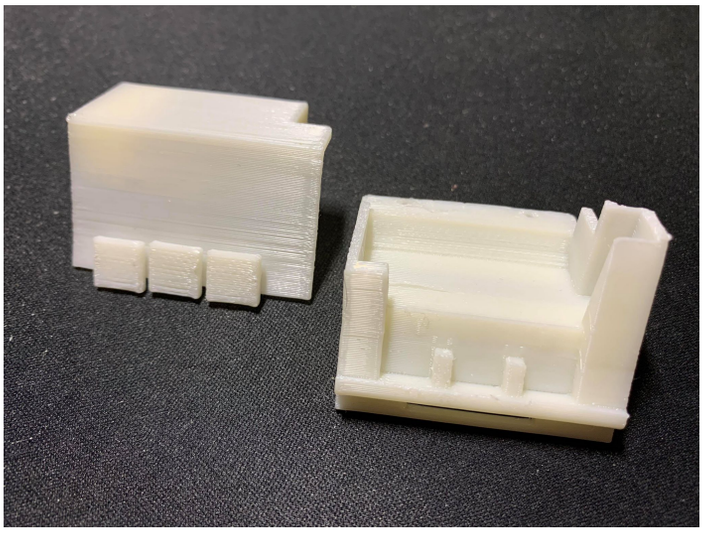


Figure 14: The printed casing halves.

In a bid to simplify the construction process, the team experimented with printing the top half of the casing with the button covers already included rather than having them as separate pieces that would be glued on after. Unfortunately, this led to the area around the button covers being much too complex for the printer to efficiently handle, and the added support material was almost impossible to remove without damaging the thin flap mechanism. Thus, all remaining prints were done according to the original design with separate button covers.

5.3 Soldering

Soldering was generally performed through reflowing components on the PCB with hot air since the majority of the components (resistors and capacitors) chosen are on the micro-scale level (mainly 0201 packages, which are 0.6 mm x 0.3 mm). Tools such as a home-made microscope (which is using the magnifying glass function on an iPhone), soldering iron, solder paste and flux were used to help smoothen the soldering process and soldering reworks. Due to the impact of the coronavirus as mentioned earlier, a stencil was not attainable at a reasonable price, which dramatically increased the difficulty of soldering and debugging.

Figure 15 on the next page shows the PCB after all components were soldered onto it, placed next to a quarter for scale.

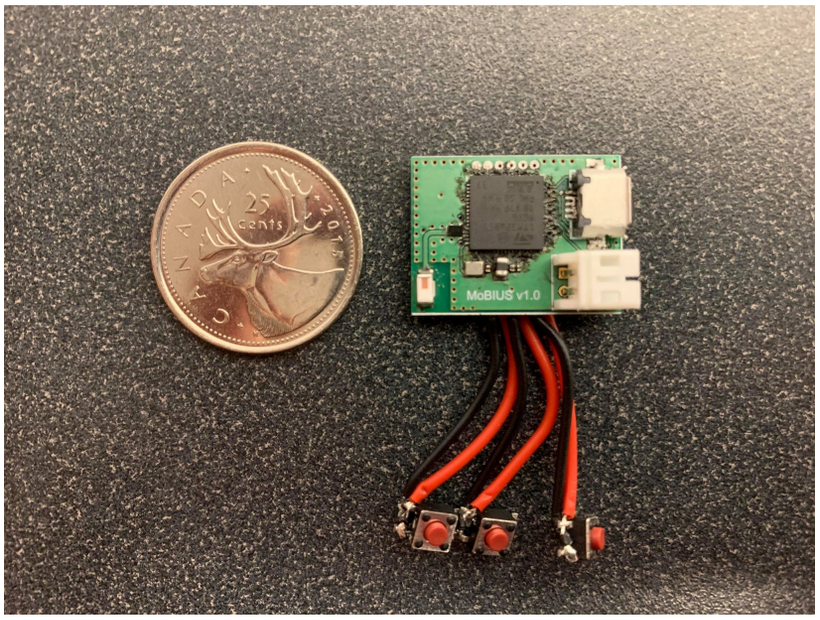


Figure 15: PCB with soldered components (quarter for scale).

5.4 Hardware Testing

Testing was performed as each functional block on the PCB was soldered. The power output was 3.3V as expected, and the LiPo was charging with a lit up LED indicating its status. However, several hardships were experienced while testing and debugging the MCU.

Initially, the firmware was not able to upload to the MCU due to the inability to detect the MCU from the PC side. While no signs of abnormality was found from extensive short testing on every single MCU pin, the NRST programming pin was measured with a low voltage, indicating that there was some connectivity issues on the power lines causing the GPIO not to be properly powered up. For this reason, the MCU was resoldered with solid connections at each pin.

Then, with NRST properly pulled high, the MCU still was not able to be detected by the programmer. By inspecting a reference Nucleo Board using the same MCU and some cross validation, the MCU was proven to be not faulty and it is found that the SWCLK pin should also be pulled high. This showed that some internal GPIO lines were not powered up properly. This was confusing, since the soldering issue was solved with all power pins receiving 3.3V. Through extensive research, it was found that the buck converter needs to be soldered on (which was not soldered on at the time due to modular testing, and was not thought to be necessary) to power the internal SMPS. As a result, soldering on the external buck converter fixed the issue, making the MCU programmable.

The last issue was related to the BLE, including its ability to properly advertise and connect to a PC client. Advertisement issues were solved through software configurations, while connectivity became a big problem due to a design requirement not mentioned anywhere in the MCU product datasheet. As stated in the datasheet, 4 clock sources, which includes the high speed external crystal (HSE) used in this design, can be used to wake up the BLE for transmission. However, through the help of online forums and research, it is discovered that only the low speed external clock (LSE) was accurate enough to hold a stable BLE connection between the MCU and a PC client while all other clock sources were only sufficient for advertising. Since the LSE was not part of MoBIUS's planned hardware design, an LSE had to be ordered and externally wired to the PCB. This was an extremely difficult task to do as the LSE (1.2 mm x 2.05 mm in size) along with two 0402 loading capacitors (1 mm x 0.5 mm in size) had to be wire-soldered on to two MCU pins of 0.2 mm pitch. In addition, the LSE is very load sensitive and any additional stray capacitance from a "long" enough wire and thickness (eg, 1 cm length 0.254 mm thickness) would cause the LSE fail to start. Fortunately, this fix came out to be successful.

6 Final Testing and Performance

The final product was tested with trial runs that simulated the expected workflow of the user. Basic Bluetooth functionality was the first to be tested. The device was powered on and the test laptop's Bluetooth menu was checked to see if it could detect MoBIUS's advertisement. Following this, the pairing process was initiated and it was checked if the laptop could connect successfully to MoBIUS and recognize it as a mouse-related device. At this point, MoBIUS was also power cycled off and on again to see if it could automatically reconnect to the paired laptop. After Bluetooth testing was finished, the onboard electrical components were checked starting with the buttons. A typical trial consisted of the following steps:

- Press SW2 to activate the device and start cursor tracking
- Move the device around to check IMU functionality
- Press and/or hold SW1 and SW3 to test left and right clicks and click-and-drag
- Double click SW2 to switch to scrolling mode
- Move the device up and down on a webpage to check scrolling functionality
- Double click SW2 to switch back to tracking mode
- Press SW2 to deactivate the device

The range and strength of the BLE transmission was also tested to make sure there was no interference from the casing or other sources. Fortunately, it was found that even when enclosed inside a box a good 2 metres away from the laptop, MoBIUS was still able to properly connect and communicate. The constraint of a minimum of one hour of continuous cursor control was also tested by leaving the device on for extended periods of time, checking that it remained paired and connected all throughout. While a full run that completely depleted the battery was not performed due to time constraints, the minimum requirement was easily met, and it is predicted that it would take at least 6.5 hours before the battery would be fully depleted.

Several participants were asked to try the final product for feedback on the user experience. The reception was generally positive with most people saying they were impressed with how small the final device managed to be and the smooth integration with the existing Bluetooth functionality on their laptop. They were also able to learn how to use the device within a short timespan, though the prevailing criticism was the still somewhat sensitive nature of the cursor movement, especially when trying to make extremely small adjustments. However, for general usage, MoBIUS still performed as required and expected.

7 Schedule and Budget

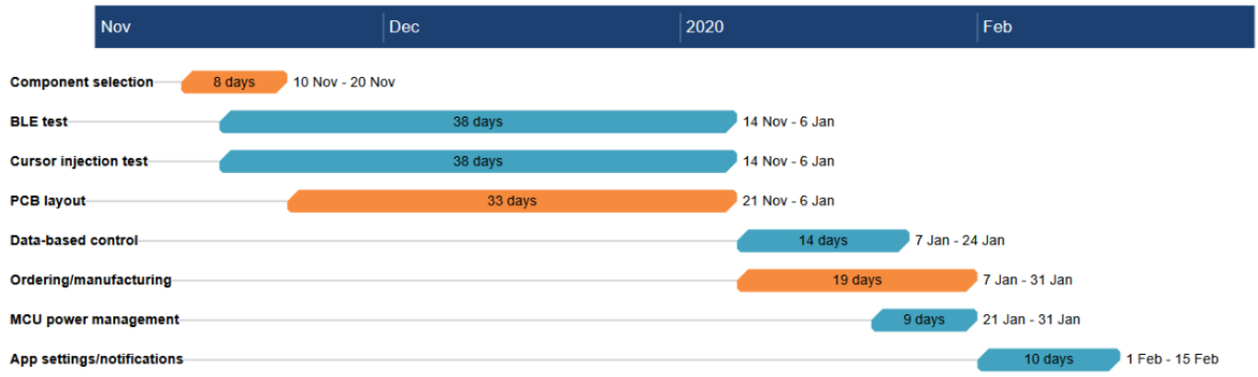


Figure 16: Project Timeline [2].

Figure 16 shows a general timeline of the originally planned project's stages, with hardware-related work in orange and software-related work in light blue. The original goal was to finish preliminary tests and planning before the start of the winter term; hardware component selection and the PCB layout schematic would be done before then along with testing of BLE connectivity and the manipulation of a laptop cursor through software. After this, the PCB and other hardware components would be ordered and assembled while the software team worked on controlling the cursor based on sensor data during the first half of the winter term. Finally, steps would be taken to optimize the MCU's power consumption and put the finishing touches on the application running on the laptop. The schedule aimed to complete all tasks by the middle of February, thus leaving several weeks of leeway for any unexpected events that may occur.

However, the project was behind schedule due to several delays. End-of-term assignments and project deadlines, as well as final exams, together with Christmas break and back-to-school preparation pushed back the time scheduled for various tasks significantly. For this reason, only a rough hardware design, component selection, and some basic BLE functionality testing on a development board were accomplished by the end of January. Then, the hardware schematics and PCB layout were quickly revised and carried out and tailored to meet manufacturing capabilities while more high level software was developed in parallel.

Due to the unexpected outbreak of the COVID-19 virus, cheap manufacturing with reasonable lead times were unavailable, leaving no options but to seek solutions with North American manufacturers and taking a hit on manufacturing time and cost, thus creating the second major delay on project task deadlines. Fortunately, the majority of the software was developed and tested on a development board and was ready to be flashed when the PCBs arrived. However, the hardware construction did not go as smooth as expected, which resulted in spending extra time in troubleshooting MCU programming and BLE connection problems. The difficulties in hardware troubleshooting took longer than anticipated, causing the third delay in project scheduled deadlines. Luckily, the complete software and the mechanical components were ready after all hardware issues had been debugged. As a result, the product was fully tested and complete before the presentation deadline.

Bill of Materials			
Hardware Components			
Digikey Part Number	**Description**	**Cost ($)**	
497-14946-1-ND	IMU IC	6.76	
1568-1257-ND	IMU Sensor Module	16.95	
497-18488-ND	MCU-STM32WBxx Series	8.27	
Mouser# 406-ASR00011	ASR00011 Lithium Polmer Battery	3.49	
497-19088-1-ND	RF Matching	0.33	
	Low Dropout Regulator	0.48	
	Charging IC	1.3	
	Fuel Gauge	4	
PCB Manufacturing			
Manufacturer	**Description (layers, drill size, trace width, etc)**	**Cost ($)**	**Lead Time**
PCBWay	4 layer, 0.3mm drills size, 4 mil trace width	334	15 days
	1 mm thickness, Burried/ Blind Vias, Immersion Gold		
PCBWay	2 layer, 0.3mm drills size, 4 mil trace width	92	10 days
	1 mm thickness, Burried/ Blind Vias, Immersion Gold		
Total Max Cost		375.58	

Table 8: Predicted cost of materials for 5 units [2].

Table 8 on the previous page breaks down the original estimated budget, which is calculated by aggregating the individual component costs in the hardware Bill of Materials (BOM) as well as the PCB manufacturing cost. In terms of hardware component cost, all necessary components were sourced from either DigiKey or Mouser (costs for each component are listed in 8). As for manufacturing cost, there are many manufacturers, local and international, that can be selected. However, PCBWay will be used for estimating manufacturing cost since the price does not vary too much for manufacturers similar to PCBWay.

The cost of manufacturing a PCB highly depends on the number of layers of the PCB, the minimal trace width and the spacing between each trace, and the drill hole sizes of the vias. Since the device should ideally be as small as possible, without pushing the manufacturing limits, the smallest drill size is decided to be 0.3mm and the smallest trace width is 4 mil. Ideally, the PCB should be designed with 4 layers to provide optimal performance. However, this might be optimized to two layers if needed since the manufacturing cost and lead time increases drastically with the number of layers in a board. The trade off for this is performance and potentially larger overall size of the PCB. With these design considerations in mind, there are roughly two estimates for PCB manufacturing cost; $334 for a 4-layer board and $92 for a 2-layer board. The total cost then becomes $375.58 for the 4-layer boards and $133.58 for the 2-layer boards.

While the cost of electrical components were similar to the anticipated expenses as the table above, the actual manufacturing cost increased significantly (refer to Bill of Materials in the Appendix for a more detailed cost breakdown of the final component list). As mentioned earlier, due to the impact of the COVID-19 virus on the cheaper Chinese manufacturers, North American manufacturers were the only option. However, they produce PCBs at higher costs, at least 10x the cost per board under the same manufacturing requirements, and can dramatically increase as the required lead time decreases. Due the delays in scheduled project deadlines, the PCB needs to be made quite urgently, thus driving the manufacturing cost higher. In the end, the cheapest deal found was with Candor Industries located in Toronto with a cost of $590.99 for 5 boards.

8 Conclusion and Recommendations

To improve the user experience when working on a laptop in a space-constrained situation, MoBIUS was created as an alternative input device to the touchpad. It was designed to operate without an additional flat surface, within 1 metre of the laptop, and for a minimum of 1 hour before requiring a recharge. After judging potential solutions on the criteria of size, cost, ease of use, latency, battery life, and effective input area, a "ring-based" wearable solution was initially chosen and later developed into a final design resembling a "half-sleeve" resting on the user's finger. The casing of MoBIUS was includes a built-in cover for the buttons and 3D printed with ABS plastic due to its mechanical properties, and it was designed to physically accommodate as many different users as possible.

Initial selection of the MCU, IMU, and battery were carried out according to their importance in the device. The STM32WB55xx MCU, LSM9DS1 IMU, and ASR00011 LiPo battery were selected due to their combined ability to minimize size and maximize battery life. Bluetooth Low Energy allows the device to send data to the laptop and ultimately control the cursor through a combination of IMU and button data. The original design called for capsenses to be used, but the resulting increase in size and hardware complexity was deemed to be too much for the scope of the project.

Thanks to the Bluetooth protocol's inherent ability to recognize mouse-like devices and automatically interpret their data as cursor-related actions, MoBIUS only required an embedded application for the MCU to be developed. The software implements Bluetooth advertisement, Just Works authentication, and HID service along with button interrupts and debouncing. IMU data is used to calibrate the device upon each activation instance, and the project also experimented with using gesture recognition for more advanced actions that the user could perform. Unfortunately, a satisfying and reliable implementation was not obtained in time, and thus the original firmware was kept for the final product.

Construction and manufacturing was challenged by the coronavirus which forced PCB fabrication to be done with a Canadian-based company at increased cost. Further difficulties arose when the MCU would not be recognized by the programmer and it was discovered that successful BLE transmission required an LSE clock that was missing from the original schematics. Fortunately, these problems were fixed in time, and the final product was typically well-received by test users.

MoBIUS was able to meet many of the constraints and criteria that were laid out at the start of the project despite delays and increased costs. It can operate without any extra flat surface, function within a 1 metre radius of the laptop, and last for over an hour off of one battery charge. Though the final cost per unit was over the preferred amount of $100, MoBIUS is still small, easy to use, and gives the user a high degree of freedom when controlling the cursor.

It is recommended for any future projects that electrical-related work - hardware selection, components ordering, and PCB manufacturing - be done as soon as possible. Lead times were the primary bottleneck experienced during this project, and in hindsight a much better plan would have been to place orders shortly after the Christmas break. As well, more extensive research regarding BLE would have possibly revealed the issue regarding the LSE, which would have further saved time and money. Thus, future developments on MoBIUS would first focus on correcting the PCB and properly adding in the connections and pads for the LSE. Once this is done, The next step would be to improve the earlier attempts with gesture recognition and properly implement it for even more user flexibility.

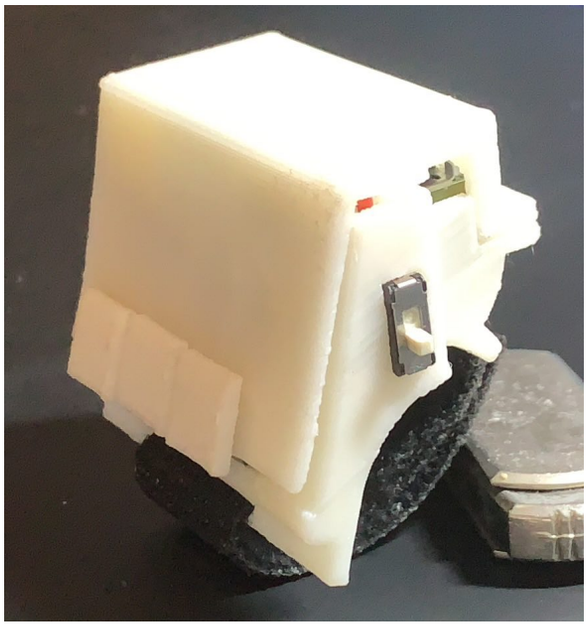


Figure 17: The final version of MoBIUS.

9 Teamwork Effort

This team is comprised of four students. For maximum efficiency, each team member was responsible for areas of the project that they either had the most expertise in or were the most qualified to carry out relative to other members. As well, the final report was worked on by the entire team.

Nisarg has experience with the software development cycle for embedded systems. He implemented and tested the calibration procedure for the IMU to enable cursor tracking and accurate manipulation. He read prevalent human computer interface research to decide the high level interface of MoBIUS, including the idea of a virtual joystick, an intuitive and simple mode for cursor manipulation. He also worked on the gesture recognition, gathering data and applying knowledge from MTE 549, Data and Sensor Fusion, to classify gestures from IMU data.

Simon has a strong background and experience in electrical hardware design, embedded systems, as well as experience in designing mechanical parts. He came up with the high level hardware system architecture and did the majority of the hardware component selection which was then used for implementation in the electrical schematics as well as the PCB layout. To ensure system performance, he minimized design errors by doing extensive RF antenna, linear capacitive sensing, and Bluetooth research from various application notes. Before sending out the PCB to manufacturing, he constantly checked up with the software and mechanical members in the team to see if there are any further changes to the PCB design so that the system will work as intended after putting everything together. When the board was ready for manufacturing, Simon worked with Lawrence to contact 10+ manufacturers to find the best offers that met the required manufacturing specifications while also being low-cost. Once the PCB had arrived, Simon did the complete soldering as well as the electrical testing and debugging for the board. He also worked closely with the software team to test MCU programming and BLE functions and debug unexpected electrical issues as different functional blocks were being soldered onto the PCB. In parallel to the electrical tasks Simon was performing, he helped the software team find certain component drivers and libraries, troubleshooting firmware MCU configuration settings during the initial software development stage, and helped Lawrence come up with the rough mechanical stack-up and casing design for MoBIUS.

Shilpan has experience working with software and hardware development. He is knowledgeable in RF hardware, image processing and working with Bluetooth technology. This was seen by helping with hardware design and research into various RF components and parts for the device and making design decisions and trade offs. As well, for software design, Shilpan performed IMU calibration tests and experimented with various filtering techniques for accurate readings to be used for smooth cursor control. Shilpan also created the poster for the symposium and sought feedback from other groups on improving the group's presentations and final product. Through key leadership and team player principles, he was able to help the group deliver the overall project on time while having fun.

Lawrence has a background in high-level software development, 3D printing, and CAD and graphics programs. As a result, he was responsible for designing the casing of MoBIUS with input from other members of the team, creating the relevant Solidworks files, and printing and assembling the casing at WatIMake. For MoBIUS's software, Lawrence went through ST-provided workshops and documentation to set up the embedded project in STM32CubeIDE and implement Bluetooth functionality. He also implemented the timer and interrupt system, button debouncing, and double-click and mode switch logic seen in the software. Lawrence was also responsible for creating and maintaining the website for MoBIUS and took charge for the final report, which included creating a rough outline, checking for formatting, and assigning specific sections for team members to write.

References

[1] Keyboard touchpad combination. https://patents.google.com/patent/US20040041791. Accessed: 2020-03-26.

[2] Shilpan Shah Lawrence Wu Nisarg Bhavsar, Xiyuan Chen. An Alternative to Laptop Touchpads, 2019.

[3] Nadine Franke Thomas Krems Josef. Attig, Christiane Rauh. *System Latency Guidelines Then and Now – Is Zero Latency Really Considered Necessary?* 2017.

[4] Mycestro - A 3D wearable, gesture-control mouse...and more! https://mycestro.com/. Accessed: 2020-03-26.

[5] 3-D Finite-Element Models of Human and Monkey Fingertips to Investigate the Mechanics of Tactile Sense. http://touchlab.mit.edu/publications/2003_009.pdf. Accessed: 2020-03-26.

[6] A Developer's Guide to Bluetooth. https://www.bluetooth.com/blog/a-developers-guide-to-bluetooth/. Accessed: 2020-03-26.

[7] Understanding Bluetooth Security. https://duo.com/decipher/understanding-bluetooth-security. Accessed: 2020-03-26.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

A Schematics

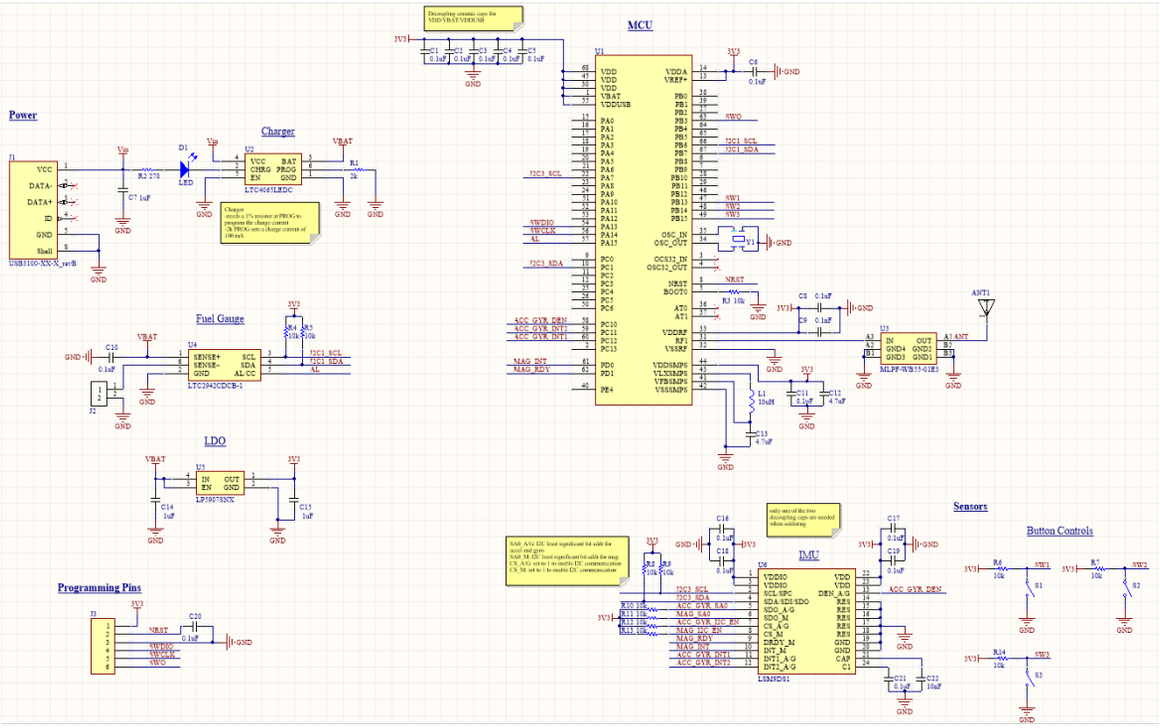


Figure 18: PCB Schematics.

B Bill of Materials

Index	Quantity	Distributor	Part Number	Manufacturer Part #	Description	Unit Price	Cost
1	1	Digi-Key	712-1006-1-ND	2450AT18B100E	RF ANT 2.4GHZ CHIP SOLDER SMD	0.86	0.86
2	1	Digi-Key	644-1354-1-ND	NX2012SA-32.768KHZ-EXS00A-MU00530	CRYSTAL 32.768KHZ 7PF SMD	1.53	1.53
3	3	Digi-Key	EG5353CT-ND	TL3305AF260QG	SWITCH TACTILE SPST-NO 50MA 12V	0.30	0.90
4	1	Digi-Key	2073-USB3100-30-ACT-ND	USB3100-30-A	MICRO B SKT, TOP MOUNT (REVERSE)	1.05	1.05
5	1	Digi-Key	754-2024-1-ND	APG0603SURC-TT	LED RED CLEAR 2SMD	0.55	0.55
6	1	Digi-Key	455-1719-ND	S2B-PH-K-S(LF)(SN)	CONN HEADER R/A 2POS 2MM	0.25	0.25
7	1	Digi-Key	490-4029-1-ND	LQM21FN100M70L	FIXED IND 10UH 100MA 600 MOHM	0.32	0.32
8	1	Digi-Key	ED11561-ND	854-22-006-10-003101	CONN SPRING PISTON 6POS SLD PCB	8.93	8.93
9	15	Digi-Key	490-10388-1-ND	GRM033C71C104ME14D	CAP CER 0.1UF 16V X7S 0201	0.04	0.56
10	3	Digi-Key	490-13219-1-ND	GRM033C81A105ME05D	CAP CER 1UF 10V X6S 0201	0.42	1.27
11	2	Digi-Key	490-13230-1-ND	GRM035R60J475ME15D	CAP CER 4.7UF 6.3V X5R 0201	0.49	0.98
12	1	Digi-Key	490-6146-1-ND	GRM033R71C101KA01D	CAP CER 100PF 16V X7R 0201	0.04	0.04
13	1	Digi-Key	490-14453-1-ND	GRM033R71C103KE14D	CAP CER 10000PF 16V X7R 0201	0.03	0.03
14	2	Digi-Key	490-10668-1-ND	GCM1555C1H100JA16D	CAP CER 10PF 50V C0G/NP0 0402	0.16	0.32
15	12	Digi-Key	P122414CT-ND	ERJ-1GNF1002C	RES SMD 10K OHM 1% 1/20W 0201	0.03	0.38
16	1	Digi-Key	P123271CT-ND	ERJ-1GNJ202C	RES SMD 2K OHM 5% 1/20W 0201	0.04	0.04
						Total	18.01
1	1	Mouser	511-STM32WB55RGV6	STM32WB55RGV6	RF System on a Chip - SoC	11.73	11.73
2	1	Mouser	511-LSM9DS1TR	LSM9DS1TR	IMUs - Inertial Measurement Units	8.70	8.70
3	1	Mouser	344-NX2016SA32S06465	NX2016SA-32M-EXS00A-CS06465	Crystals Crystals CRYSTAL 32MHZ 10PF SMD	0.91	0.91
4	1	Mouser	511-MLPF-WB55-01E3	MLPF-WB55-01E3	RF Wireless Misc RF Wireless Misc CSPG BUMPLESS	4.29	4.29
5	1	Mouser	584-C2942CDCB-1TMPF	LTC2942CDCB-1#TRMPBF	Battery Management Battery Gas	6.75	6.75
6	1	Mouser	584-C4065LEDCTRMPBF	LTC4065LEDC#TRMPBF	Battery Management Standalone 250mA Li-Ion Coin Cell Charger	4.17	4.17
7	1	Mouser	595-LP5907SNX-3.3NPB	LP5907SNX-3.3/NOPB	LDO Voltage Regulators 250mA Ultra-Lo Noise	0.64	0.64
						Total	37.19
PCB	1	Candor Industries	MoBIUS v1.0	N/A	Main PCB board	118.20	118.20
						Total	118.20
Casing	1	WatIMake	MoBIUS Casing	N/A	3D printed casing	1.86	1.86
						Total	1.86
						Total Sum	175.26

Table 9: Bill of materials for one unit.

C As-built Drawings

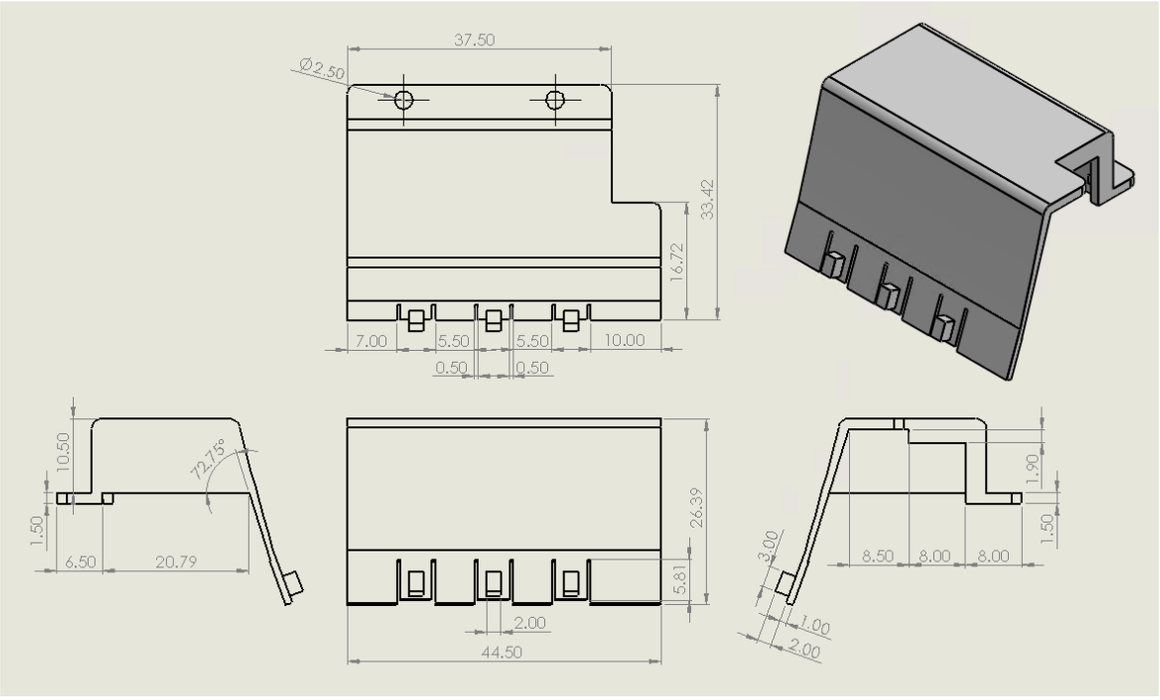


Figure 19: Casing top with dimensions.

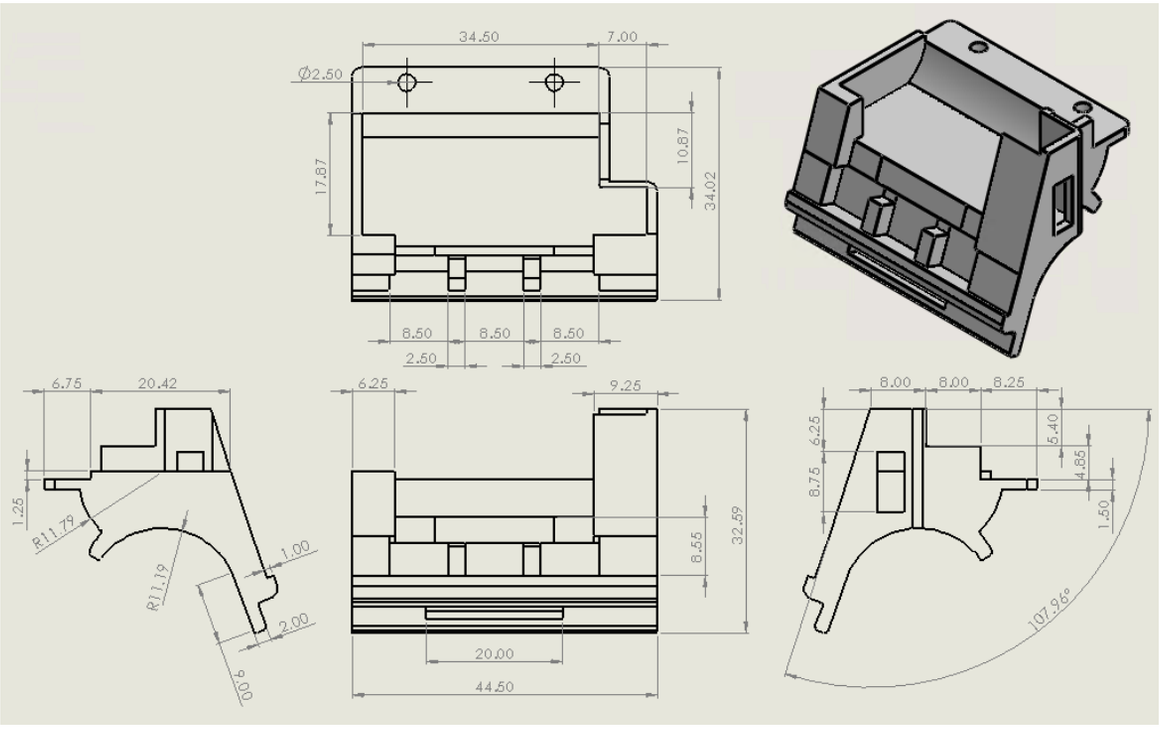


Figure 20: Casing bottom with dimensions.

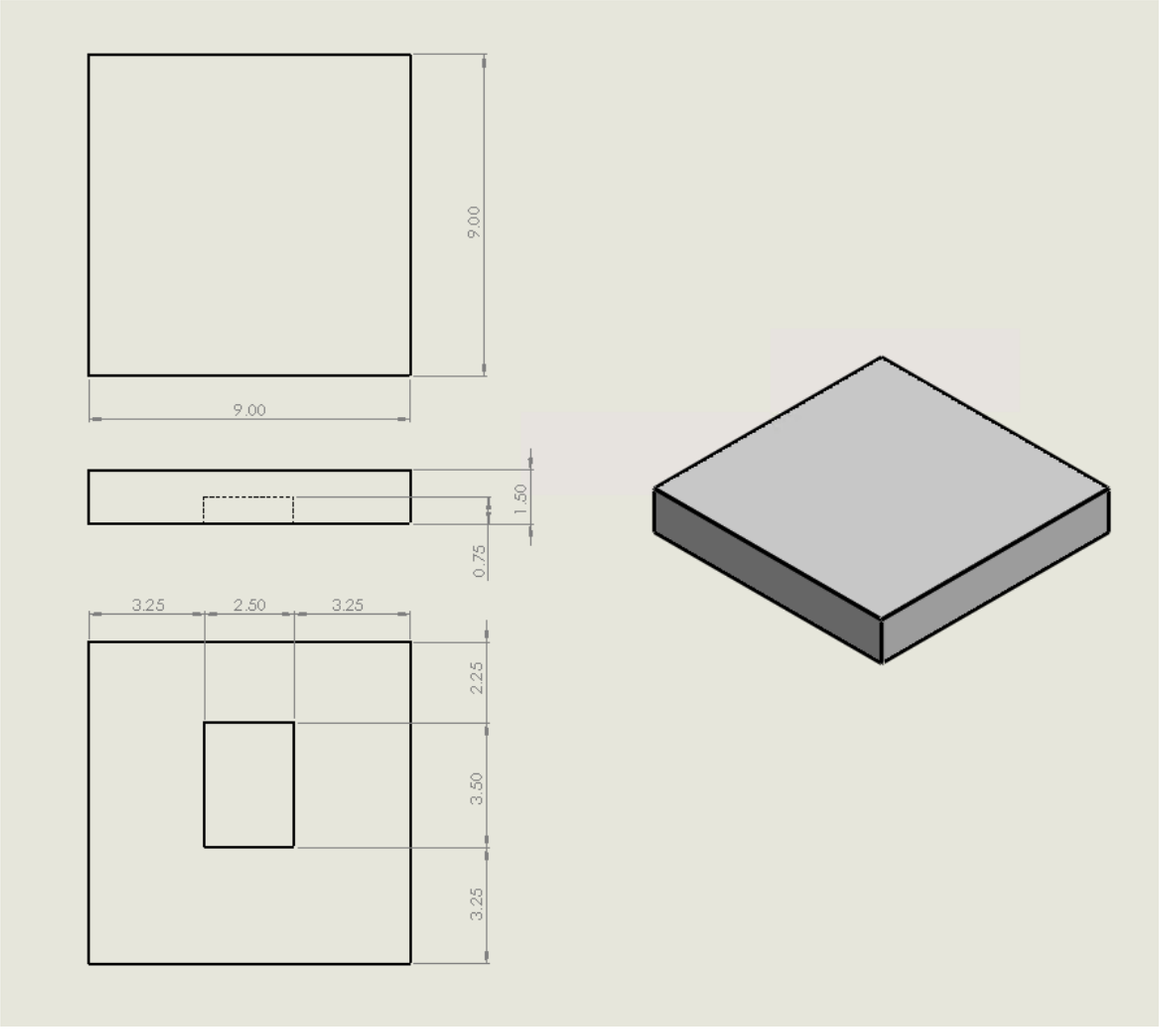


Figure 21: Button pad with dimensions.

D Firmware Excerpts

D.1 IMU Calibration

```
void calibration(void){
        int maxCalCount = 20;
        int cal_int = 0;
        int cnt = 0;

        while(cal_int < maxCalCount){
                        lsm9ds1_dev_status_get(&dev_ctx_mag, &dev_ctx_imu, &reg);
                        if ( reg.status_imu.xlda && reg.status_imu.gda ){
                                readIMUAcceleration();
                                readIMUAngularVelocity();
                                gyro_x_cal += acceleration_mg[0];
                                gyro_y_cal += acceleration_mg[1];
                                gyro_z_cal += acceleration_mg[2];
                                cal_int++;

                                acc_z = angular_rate_mdps[2];
                                if (cnt <= 20)
                                {
                                        tap_buffer[cnt++] = acc_z;
                                }
                        }
                }
                gyro_x_cal /= maxCalCount;
                gyro_y_cal /= maxCalCount;
                gyro_z_cal /= maxCalCount;

}
```

D.2 HID Update Cursor Position

```
static void HIDSAPP_Tracking_UpdateChar(void)
{
        tap = 0;
        HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_5);
        mouse_report_t mouse_report;
        uint8_t* report = (uint8_t *)& mouse_report;

        int8_t xMotion = gyro_x_output/10;
        int8_t yMotion = gyro_y_output/10;
        int8_t zMotion = acc_z_output;

        if (abs(xMotion) < 5){
                xMotion = 0;
        }
        if (abs(yMotion) < 5){
                yMotion = 0;
        }

        if( (abs(zMotion) > 122) && (abs(xMotion) < 5) && (abs(yMotion) < 5) ){
                tap = 1;
                xMotion = 0;
                yMotion = 0;
        }

        report[0] = leftClick;
        report[0] |= rightClick << 1;
        report[0] |= 0 << 2;
        report[1] = xMotion;
        report[2] = yMotion;
        report[3] = 0;

        HIDS_Update_Char(REPORT_CHAR_UUID,0,0,sizeof(mouse_report_t),(uint8_t *)& mouse_report);
}
```