ECE445 Team 44 Final Report

By:

Amanda Favila

Asher Mai

Lauren Wilcox

TA: Sainath Barbhai

Professor: Viktor Gruev

May 2023

Contents

1. Introduction *1*

1.1 Block Diagram 1

1.2 High-Level Requirements 2

2. Design *3*

2.1 Design Procedure 3

2.2 Design Details 5

3. Verification *7*

4. Cost & Schedule *8*

5. Conclusions *11*

5.1 Accomplishments 11

5.2 Uncertainties 11

5.3 Future Work 11

5.4 Ethics & Safety 11

References *12*

Appendix *13*

1. Introduction

One of the biggest transitions in society in the past few years has been the shift to working from home. This necessitates more accessible technology to allow people with certain disabilities or handicaps to do their remote work efficiently and comfortably on their computers. We have decided to create a cost-effective head-controlled mouse device that will allow the user to move and click the mouse cursor on the screen with their head movements. This device is especially helpful for those with arthritis, lupus, or other conditions that make it difficult to use a traditional mouse. Although there are other similar technologies on the market such as Smyle Mouse, they often require a camera setup, and external mechanical setup, and are generally expensive.[1] Our device will be lower cost, Bluetooth operated for ease, and just as effective.

1.1 Block Diagram

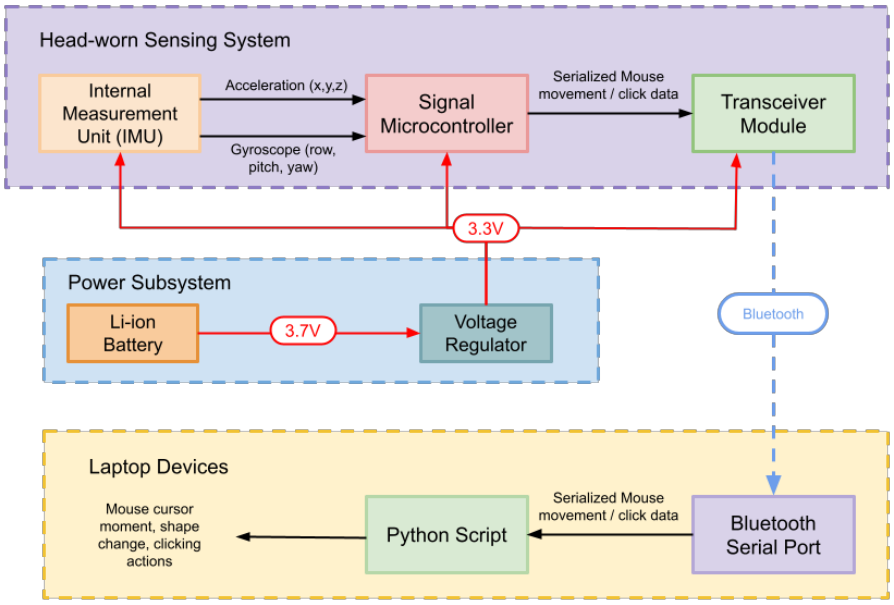

Figure 1: Block Diagram

Figure 1 shows the final block diagram used for our project with three main blocks: head-worn sensing system, power subsystem, and device system. The head-worn system includes the inertial measurement unit (IMU), the microcontroller, and the Bluetooth transceiver module, the power subsystem contains the battery and voltage regulator, and finally, the device system contains the Bluetooth serial port and a Python script to translate and apply the mouse displacement data. The IMU is the part that will collect the data on the user's head movement based on acceleration and gyroscopic data. These values will be sent to the microcontroller using SPI protocol to be processed into serialized mouse movement/click data, then sent to the Bluetooth transceiver module. On the device, the Bluetooth data will be received through its serial port, and the Python script will read this data and map it into pixel displacement and click functionality for the mouse cursor. All the parts on the device can operate with 3.3 V, so the power subsystem contains a voltage regulator which will reduce the voltage output of the battery from 3.7 V.

One big change to the original block diagram was the kind of transceiver we picked. Originally, we planned to use a 2.4 GHz RF transmitter with a USB plug-in receiver, but ultimately opted against it due to the added complexity of needing another printed circuit board (PCB) for the receiver. With Bluetooth, we only needed one transceiver module mounted on the device and it increases the ease of use of the entire device. Additionally, we planned to program the Bluetooth as a human interface device (HID) so that the laptop device would automatically recognize the device as a mouse and interpret the serial data as mouse displacements. However, we were not able to implement this feature, so we added a Python script that is run on the laptop to process the displacement and click data received from the Bluetooth serial port. The script is necessary to implement the cursor movement and clicking functionality.

1.2 High-Level Requirements

Our project needed to meet several performance requirements for it to be a suitably functioning device that is operable for users.

1. The device must have a successful calibration sequence that calculates appropriate distances and speeds for the cursor to move based on the user's specific head movements.
2. The device must be able to accurately move and click the mouse cursor based on the user's head movements. This means that when the user moves their head up, down, left, and right, the mouse cursor will move up, down, left, and right, respectively.
3. The device must be able to be used on both Macs and PCs.
4. The device must utilize user-adjustable sensitivity that can map different cursor speeds to the same head rotation speed.

Requirements 1, 2, and 4 were largely based on the performance of the IMU and the Python script. Calibrating the IMU data and fine-tuning the mapping from degrees of rotation to pixels moved on the screen were essential to ensuring the success of these requirements. To keep this device as accessible as possible to both Macs and PCs, we implemented the data communication with Bluetooth and made sure to not use Mac/PC-specific notations or functions in our Python script.

2. Design

The overall design of our device features several components: Power Supply, Head-worn Sensing Device mounted on a hat and a Python program on the user's PC or Mac that receives the Bluetooth signal and executes the mouse movement and clicking actions. The Head-worn sensing device includes an Internal Measurement Unit (IMU), a Bluetooth transceiver, a microcontroller, a battery, and a USB-C port. The user may use a power bank that sits on the table and plug it into the USB-C port to charge or to keep it powered while using the device.

2.1 Design Procedure

We analyzed many considerations for several parts of our design. The first consideration we had was to design a product that is camera-free. Cameras could be useful for tracking precise head movements, detecting eye-winking for clicking actions, as well as where the eyes are looking for precise mouse cursor movements. We chose an approach that would require no cameras, despite the added sensing capability, to preserve users' privacy. Many users don't like having the webcam turned on or uncovered because there is a risk of hackers having access to the footage. So, we decided to use an IMU alone for all the sensing purposes.

There are many different types of IMUs to choose from, which mostly differ in degrees of freedom. We decided on an IMU that has 6 degrees of freedom that combines a 3-axis gyroscope and a 3-axis accelerometer. The gyroscope provides rotational velocity data that are in the units of degrees per second. This data is very useful for providing us with head rotation information, such as nodding, lifting of the head, turning left, and turning right, all of which decide which direction we will move the mouse and by how much. The accelerometer data provides us with information on how the device is oriented and which way gravity is pointing. This information is important for us to implement the "tilt head to click" functionality because using the gyroscope alone for tilt detection is ambiguous, e.g. the action of tilting your head to the left and then the action of tilting back to the neutral position could be misidentified as left and then right tilt without accelerometer data. Some IMUs have 9 degrees of freedom. In addition to the gyroscope and accelerometer, these IMUs have a 3-axis magnetometer. Magnetometers help determine how the devices are oriented about the North Pole. We decided that we do not need to know this information because we always assume that the user is sitting in front of the Mac or PC facing the screen, and we only need to program the device relative to the screen and not relative to the North Pole.

A third design decision we had was to use a Bluetooth Transceiver instead of a pair of 2.4GHz Wireless Transceivers. We realized that all modern Mac and PC devices that our device interacts with have built-in Bluetooth modules. Bluetooth modules can be paired with Mac and PC devices in their Bluetooth settings easily, and it does not require anything to be plugged in. The lack of the need for a USB receiver frees up the user's USB ports so that they can be used for other peripherals, and it also has the advantage of not having an extra piece of small item that the user must keep track of and not misplace accidentally. We also considered the option of communicating through Wi-Fi on the user's Mac and PC. However, most Mac and PC devices cannot connect to multiple Wi-Fi networks at once, so connecting their devices to our Head-Controlled Mouse device would prevent the users from accessing the internet.

A fourth design concern we had was the voltage regulator. We needed a voltage regulator to ensure that our ICs received an input voltage with as little variations as possible so that our device has constant performance. Coincidentally, the voltage regulator that we used in the soldering assignment (the LDK120DM33R), which has an output voltage of 3.3 V, worked in our case. 3.3 V fell within all desired input voltage ranges for the rest of our components. The input voltage of the microcontroller is 1.8V - 5.5 V. The operating voltage of the IMU is 1.71 V – 3.6 V. The required voltage for the Bluetooth transceiver is 3.3 V – 5 V. The desired voltage of the ISP header is the same as that of the microcontroller. This was a plus because it meant that we did not have to worry about using any voltage dividers or level shifters in our design, so it simplified our work.

For the power supply, we aimed to reduce the amount of weight that the user must carry on their head, so we decided to use a small Li-ion battery that is lightweight and does not cause discomfort over extended periods of usage. We also wanted this power supply to be rechargeable, so that the user would not have to keep buying new batteries and replacing them. Because we went with a lithium-ion battery, it was important to ensure user safety by making sure that the operating temperature of the battery allowed for the device to be used at around room temperature, since most people use computers and such devices inside. Room temperature is around +20 degrees Celsius, and the operating temperature of the battery is -20°C to +60°C, so it safely fell within the accepted range. We also added a secondary method of power to our device, a USB-C charging port. This was out of concern for convenience as well, since most people already have a USB-C charging cable lying around. It can be connected to a power bank that does not sit on the user's head, so there would be no added weight in this case. When choosing these two components, we needed to make sure that their voltages fit into the input voltage of our chosen voltage regulator. The rechargeable battery has a nominal voltage of 3.7 V, and the USB-C charging port has a nominal voltage of 5 V. The input voltage of the voltage regulator is 1.9 V – 5.5 V, so these components are all compatible with each other.

We also considered many factors as to what microcontrollers we need. We want to choose a microcontroller that is as cheap as it can be while still meeting our needs. We do not have a lot of processing that needs to be done, so the clock rate does not need to be high. We also don't have large arrays of data that need to be stored in memory. We needed enough pins to communicate with the IMU and Bluetooth Transceiver. The microcontroller needs to support SPI protocol to communicate with the IMU, and it needs to have RX and TX serial communication pins to send data through the Bluetooth Transceiver. We eventually decided on ATmega328P, which is the same chip found on Arduino UNO that we can use to prototype our design. This device has 16 KB of RAM and 20MHz of clock speed, and it costs only $2.79, so it was the best choice that is enough for our needs while still being cheap.

2.2 Design Details

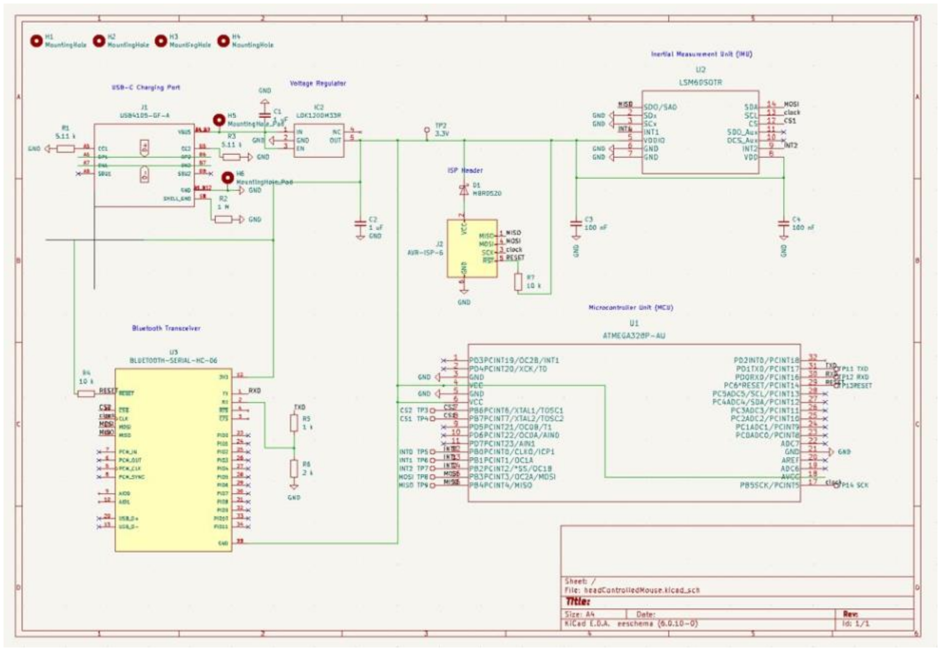

Figure 2: PCB Schematic

Figure 2 shows our final schematic design of the system. The two sources of power are shown in the top left. The first means of powering the device is the USB-C charging port, USB4105. Reference [3] was particularly useful in wiring up the connector with two 5.11 kΩ pull-down resistors as well as one 1 MΩ pull-down resistor. The second means of powering our device was our rechargeable battery pack, so we included H5 and H6, two mounting holes, at VBUS and GND respectively, which would be used to connect the battery terminals to the circuit.

The other part of our power subsystem was the LDK120DM33R voltage regulator. As per the datasheet, the input and output both receive 1 µF filter capacitors. We gave the enable pin the same input voltage since we always wanted the voltage regulator to be operating, so it needed to have a constant high signal. The output of the voltage regulator is the input to the rest of our ICs.

The LSM6DS0TR Inertial Measurement Unit is located in the top right of our schematic. The four SPI pins, MISO, MOSI, clock, and CS1 were connected to our SPI protocol. We also connected the two interrupt pins to the microcontroller in case we needed them. The rest of the pins were grounded or not connected as needed, and the input voltage was given a 100 nF filter capacitor per the datasheet's requirement.

The AVR-ISP-6 ISP header was needed to program our microcontroller. The same 4 SPI wires were connected.

The HC-06 Bluetooth transceiver again used a reset pin and the 4 SPI wires, the same way as the ISP header. In addition to those, there were Tx and Rx pins. The transmission pin of the Bluetooth transceiver was connected to the receiving pin of the microcontroller, and vice versa, resistors of values 1 kΩ and 2 kΩ as requested by the datasheet.

Many test points were added throughout the circuit so that we would be able to inspect the signals at various points while testing the PCB. This included a test point for the raw input voltage, regulated input voltage, and all SPI wires, interrupts, chip selects, and resets.

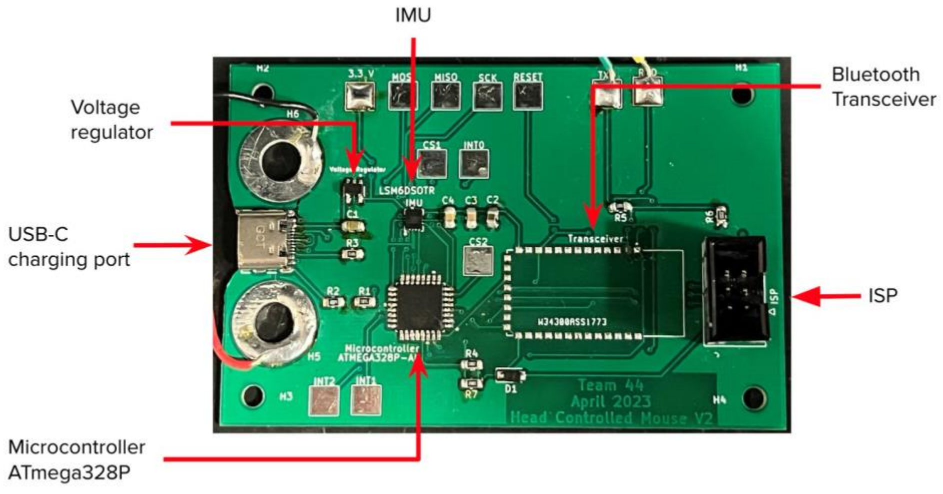

Figure 3: Physical PCB

Figure 3 shows our PCB with all the components soldered on. The only component not soldered on was the Bluetooth transceiver, because we ended up changing the exact device, we were using so we had to simply solder wires onto the pins we needed and connect those to a breadboard with our new transceiver.

3. Verification

The overall device has been tested by simulating how the user would interact with it in the real world. We can verify that the device works with both Macs and PCs. We find that the mouse can move to where the user wants it to with high accuracy. The device can stay still in the same place without much vibration when the user stops moving their head. Tilting to click can be accomplished precisely by holding the cursor in one place for one second, and it won't move for another second, during which the user can tilt left or right to perform a left click or right click. The whole system works with a speed of 25 Hz on average.

We verified the IMU to meet most of the requirements described in Table 3 in the Appendix. The IMU can detect head movements up, down, left, and right and detect the speed at which they move in those directions. The user can move from one side of the screen to the other without moving more than 45 degrees. We are also able to use a programmed timer to measure the latency of the IMU to less than 0.01 seconds.

The microcontroller meets some of the requirements described in Table 4 of the Appendix. The microcontroller is verified to be able to communicate through SPI protocols on the development board version of the chip. However, we failed to verify that it communicates with the IMU using SPI protocol because we are reading all 0.0 data for the 6 axes of the IMU.

The Transceiver verification in Table 5 has been changed due to changes in our design. We originally planned on verifying that it can communicate with 2.4 GHz of wireless signal, but we changed the Transceiver to Bluetooth, and we can verify that it is indeed able to send data from the Microcontroller to the Mac and PC devices and be shown in the Serial Monitor of Arduino Application.

Verifying the operation of the power subsystem was rather straightforward. We utilized a voltmeter on both the breadboard and the PCB to measure the voltage at different points in the circuit. On average, the voltage of our rechargeable power supply was measured to be 3.7 V, the voltage of the USB-C charging port was measured to be 4.9 V, and the output of the voltage regulator was measured to be 3.33 V.

4. Cost & Schedule

We calculated the labor costs from the following formula. Cost of labor: $30/hr ╳ 12 hrs/week ╳ 10 weeks ╳ 3 members = $10,800

Throughout the design and building process, we changed our minds on which parts to use for certain subsystems, such as the Bluetooth transceiver, so the catalog of parts we used changed a multitude of times. Figure 2 shows our itemized part list for the final design including prototype boards and extra small circuit parts like resistors and capacitors.

Table 1: Parts List

Description	Manufacturer	Quantity	Ext. Price	Link
Baseball Cap		1	$10.00	
LDK120M33R (Voltage Regulator)	STMicroelectronics	1	$0.91	Link
LSM6DSOTR (Inertial Measurement Unit)	STMicroelectronics	1	$6.60	Link
ATMEGA328P-AU (Microcontroller)	Microchip Technology	1	$2.86	Link
LP523450JU+PCM+2 WIRES 70MM (Battery)	Jauch Quartz	1	$11.30	Link
USB4105-GF-A (USB-C charging port)	GCT	1	$0.81	Link
RC0402FR-075K11L (5.11 kOhms resistors)	YAGEO	10	$0.15	Link
SEN-18020 (IMU breakout board)	SparkFun Electronics	1	$12.95	Link
CRCW02011M00FKED (1 MOhms resistors)	Vishay Dale	10	$2.34	Link
GRM033R61A104ME15D (0.1 mF capacitors)	Murata Electronics	10	$0.10	Link
GRM033R60J105MEA2D (1 mF capacitors)	Murata Electronics	10	$0.32	Link
HC-06 Bluetooth Transceiver	HiLetgo	2	$15.49	Link

Once all the parts got in and we constructed our final prototype, we calculated the total cost of the project as follows:

Total Cost of Project: $63.83 (Parts) + $10,800 (Labor) = $10,863.83

The cost of the device itself (not including the prototyping parts that we ordered for testing) is calculated to be $43.14 which is significantly lower than other hands-free computer mouse devices on the market. This ensures better accessibility for users who need a device such as this but do not have the financial means for the more expensive products.

The scheduling of our work as a team was split up over the following 12 weeks of the semester. Although we aimed to follow our original schedule as created in the Design Document closely, other commitments and a delay in parts delivery shifted the timeline. Table 2 shows the updated final schedule of our workload throughout the semester.

Table 2: Semester Work Schedule

Week	Deliverables	Division
2/13	Finish the Design Document and Team Contract.	All
	Purchase all parts listed in the Design Document as well as anything needed for prototyping.	All
	Prepare for the Design Review.	All
	Draft schematic and PCB.	Lauren
2/20	Learn how to use and communicate with the wireless USB receiver.	Asher
	Start planning how data processing will be done (converting accelerometer and gyroscope data to mouse location).	Amanda
	Design Review and PCB review with Instructor and TAs. Purchase all needed components.	All
	Make any necessary changes to the PCB.	Lauren
2/27	Set up a program project so we can start writing code.	Asher
	Plan how the sensors will be securely attached to the hat.	Amanda
	Pass PCB audit and place order by Tuesday 3/7.	All
	Complete Teamwork Evaluation I by Wednesday 3/8.	All
3/6	3D print device enclosure	Lauren
	Research Bluetooth communication.	Asher
	Begin writing data processing code.	Amanda
3/13	Spring Break	All

3/20	Redesign PCB	Lauren
	Continue working on SPI protocol and Bluetooth communication.	Asher and Amanda
	Pass second round PCB audit and place order by Tuesday 3/28.	All
	Complete individual progress reports by Wednesday 3/29.	All
3/27	Order IMU breakout board, extra Bluetooth part, and resistors/capacitors	Amanda
4/3	Order final PCB	All
	Implement project on breadboard and calibrate IMU data	Amanda
	Test Bluetooth connection, attempt HID programming	Asher
4/10	Complete Team Contract Fulfillment by 4/14.	All
	Solder on parts to PCB	Lauren
	Start testing device with PCB	All
	Prepare for Mock Demo.	All
4/17	Give Mock Demo during the weekly TA meeting on Tuesday 4/18.	All
	Prepare for Final Demo and Mock Presentation.	All
	Give Final Demo to instructor and TAs.	All
4/24	Give Mock Presentation with TAs. Prepare for Final Presentation.	All
	Give Final Presentation to instructor and TAs.	All
	Complete final paper by Wednesday 5/3.	All
5/1	Turn in Lab Notebooks by Thursday 5/4.	All
	Complete Lab checkout with TA and attend award ceremony on Thursday 5/4.	All

5. Conclusions

5.1 Accomplishments

Ultimately, we were able to accomplish all the high-level requirements for our head-controlled mouse. Specifically, the device can accurately map the user's head rotation to a displacement of the cursor in the x and y directions on the screen. Additionally, the user can click the cursor with a slight tilt to the left or right with ease. We verified that the device is operable on both Macs and PCs and that the user can adjust the speed of the mouse cursor on the screen with the speed of their head rotations. We were able to keep the weight of the device very minimal such that there would be no discomfort to the user after prolonged use and improved the power capabilities to include two methods of charging as previously mentioned. Altogether, we were able to create this head-controlled mouse in a user-friendly and reliably functional manner.

5.2 Uncertainties

We are not able to get our PCB fully working, which mainly is a result of the SPI pins not being connected properly between the IMU and the microcontroller. We tested the connections and determined that the Chip Select pin has been wired incorrectly.

We also find that although the Bluetooth Transceiver can establish a stable connection with Mac and PC after it is connected, it does not always connect successfully on the first try and requires the user to go into Bluetooth settings on the Mac or PC to forget the Bluetooth device and then try to reconnect.

5.3 Future Work

Although we were unable to fully implement the device on a PCB, the breadboard prototype worked very well and achieved all our predetermined high-level requirements. Therefore, after correctly integrating all the parts on a PCB we have further features we could implement with future work. Programming the Bluetooth transceiver as a HID would be a significant advance in the usability of the device as it would eliminate the need for the Python script, and thus make the device accessible to tablets and devices other than laptops. Additionally, a scrolling feature should be implemented along with other click-and-hold functionalities which can be done using head tilts in another axis of rotation. Finally, further progress could be made by creating a movement for common keyboard shortcuts such as copy/paste.

5.4 Ethics & Safety

In the development of our head-controlled sensor, our team placed great importance on ensuring the safety and ethical considerations of our device, as outlined by the IEEE Code of Ethics.[2] We paid close attention to head movements that may cause discomfort to users and took steps to avoid this. Our design includes a safe battery enclosure and a well-balanced weight distribution on the hat. We worked within a peer and faculty-reviewed system to ensure that our technical work is legitimate, safe, and makes realistic claims by section 5 of the IEEE code.[2] We welcomed feedback and criticism from all team members to improve our project. Additionally, we have undergone relevant CAD, lab safety, and technical training to ensure everyone's safety and to comply with campus policies. To mitigate privacy concerns, we implemented a camera-free design, tracking only the head's movement and processing it into mouse displacement data. This ensures that no personal data is breached.

To further prioritize user safety, we follow strict safety procedures and use enclosures to prevent contact with device parts. We use rechargeable batteries with a diode connected to the circuit to prevent any damage from incorrect battery insertion. We also provide instructions on properly fitting the hat to the user's head for optimal weight support and comfort.

References

[1] Smyle Mouse. Head Mouse for Hands-free Computer Control.

https://smylemouse.com/

[2] IEEE Code of Ethics. IEEE Policies, Section 7 - Professional Activities.
https://www.ieee.org/about/corporate/governance/p7-8.html

[3] “How to add USB-C to your projects - PCB design tutorial,” PCBway. [Online]. Available:
https://www.pcbway.com/blog/PCB_Design_Tutorial/How_to_add_USB_C_to_your_projects.html#:~:te
xt=Using%20USB%2DC%20in%20your,with%20the%20micro%20USB%2DB. [Accessed: 27-Mar-2023].

Appendix

Table 3: IMU Requirements and Verifications

IMU Requirements	**IMU Verifications**
The IMU should be able to detect the differences in orientation based on the user's head movement, i.e. movement up, down, left, right, and head tilts.	We will display the data measurements generated by the IMU through an Arduino to verify that correct directions are being calculated.
The IMU should be calibrated to convert a 90° range of yaw movement to 100% width of any screens, and convert a 50° range of pitch movement to the 100% height of any screens.	We will verify that the mouse cursor can be moved from one side of the screen to the other without moving the head further than the degree constraints. This can be further verified by moving only 45° range of yaw and ensuring the cursor only moves 50% across the width of the screen.
The IMU should be able to measure the speed of the user's head movements.	We will use an Arduino to display the angular velocity data and confirm that the values are correct relative to the user's head movements.
The IMU should have minimal latency in sending data, less than 0.02 second.	When mapping the IMU data to the mouse cursor on a computer, we will time the difference between the user moving their head and the cursor moving on screen. We are electing to use SPI protocol rather than I2C, which is faster as it is a 4 wire protocol so data can be sent and received at the same time.

Table 4: Microcontroller Requirements and Verifications

Microcontroller Requirements	**Microcontroller Verifications**
The microcontroller must have enough input and output pins for our data. Input: Acceleration(X, Y, Z), Gyroscope(row, pitch, yaw Output: Mouse displacement(X, Y), Click(left, right)	We can verify that the microcontroller that we chose will have the correct number of I/O pins by consulting the datasheet. The datasheet states that there are 23 programmable I/O lines, which is enough for our data even using 4-wire SPI protocol.
We must be able to use a microcontroller on a breadboard as part of our prototype as well as on a PCB as part of our final design.	We have solved this requirement by using 2 microcontrollers. The one we have selected in this document will be part of our final design, soldered into the breadboard. For prototyping, we will use the microcontroller from an Arduino on a breadboard. It has

	the following datasheet: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-metro-mini.pdf The pinouts of these 2 microcontrollers do not exactly match up, however we can just wire the prototype and the final version differently.
The microcontroller must communicate in SPI standard so that it can communicate with our chosen IMU.	This is verified in the datasheet for the microcontroller we selected.

Table 5: Transceiver Requirements and Verifications

Transceiver Requirements	**Transceiver Verifications**
The transceivers must be able to communicate with the receiver at the same frequency, 2.4 GHz RF.	This is verified via the datasheet. We can also use an oscilloscope and send a single pulse in the frequency domain to measure at which frequency this pulse is sent at.

Table 6: Power Supply Requirements and Verifications

Power Supply Requirements	**Power Supply Verifications**
Supply a voltage of 3.7 V +/- 1.3 V.	We can verify that the power supply will output the desired voltage by measuring the voltage across the two terminals of the battery using a digital multimeter. We can verify that the power supply will output the desired currents to each of our components by using a shunt resistor and measuring the voltage across the resistor and using Ohm's law to determine the current.
Input/Output protection: What would happen if a user inserted a plug or battery backwards?	We can prevent users from plugging the battery in backwards by being very intentional with our wire colors, labeling, and instructions. However, if this were to happen, we will be sure to place a diode in the circuit, so that if the power supply is plugged in the incorrect way, no current will flow through the diode and we will have an open circuit.
Environment: Will your project function in hot or cold conditions?	According to the datasheet, the operating temperature of the power supply is anywhere from 0 degrees Celsius to 50 degrees Celsius, while still maintaining up to a 4.1 V charge. This verifies that our power supply will function in all typical room temperatures.