ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

Any-Surface Computer Stylus

Team #86

ETHAN FORSELL
(ethanef2@illinois.edu)
JOHN BLEDSOE
(johndb3@illinois.edu)
ALEX CAMAJ
(acamaj2@illinois.edu)

TA: Manvi Jha

May 6, 2026

Abstract

Over the course of a semester, we created a fully flushed out product: The Any-Surface Stylus. This final report covers the full design cycle of our senior design project. It goes over the product motivation, design process, both hardware and software implementation, as well as debugging and final testing. Throughout, we talk about our design decisions, how we ended up with our final product, and make note of shortcomings and future improvements.

Contents

1 Introduction

1.1 Problem Statement and Motivation

Today, we rely heavily on digital documents and texts over their traditional paper counterpart. Digital documents are revolutionary in their ability to be sent to many people almost instantly. However, technology for people to edit these documents as free as a regular pen and paper has been slow to come. Most people are forced to use a computer mouse to draw with point and click workflow. One solution is to have a stylus that you can use on the computer screen itself, but this requires you to already have a touchscreen device. Another option is to carry around a digital drawing pad and pen, but this is bulky and often overkill for simple document editing tasks.

Our group set out to fill a simple niche: to bridge the gap between paper and digital documents and increase people's ability to write in digital formats. To do so we set out to design a stylus that can work on nearly any surface to write on a computer like one would a regular paper.

1.2 Our Solution and Requirements

In order for our stylus to be an effective digital writing tool, we had to make sure it had a couple of featured in particular. We wanted our stylus to be able to connect to any computer and work just as easily as pulling out a regular pen and writing. We also wanted our pen to be precise and accurate in movement, be able to track movement when lifted, and be as lightweight as a regular pen. Additionally, since it would be moving the cursor of the computer, we wanted it to have the functionality of a regular computer mouse as well, giving the user a fully capable tool. Before designing we formalized these requirements into three high level requirements:

1. **Writing speed test:** We want our pen to be low latency enough to write as fast as a traditional pen and paper. We will quantify this by saying we want our stylus writing time for a given text to not increase by more than 25% of the time it would take with pen and paper.
2. **Pen Writing Range:** In order for our stylus to work like a regular pen, we need the cursor to move along with the stylus when you pick it up to maintain spacial continuity with the page. We would like the combined sensors to have enough spatial range to track the paper when the stylus tip is lifted at least 0.5 inches while keeping the optimal (non jittery) movement when close to the page.
3. **Ergonomics and easy of use:** We want out product to be easy to use and maneuver like a traditional pen or pencil. To accomplish this we are going to keep the stylus part of our product under 30 grams. This should allow for the expected ease of movement needed in a writing utensil.

Figure 1: Visual Aid

1.3 Subsystem Overview

The development of this vision brought us to the final system design shown in the block diagram in Fig. 2. The system consists of two main units and can be further broken down into three subsystems. One unit houses the various sensors and user input tools while the other receives the signals from these sensors and translates them into usable USB HID signals. The units would be physically divided, with the stylus containing all the sensors and the processing unit containing the larger processing components, keeping the stylus package small and lightweight. A visual aid of the initial concept is provided in Fig. 1.

Looking closer at the two units, we can break them down into three different subsystems: sensing, control, and power. The sensing subsystem is within the stylus itself and gathers information to send to the processing subsystem. The processing subsystem is between the host device (computer) and the sensing subsystem processing the sensor information. Both of these subsystems are powered by the power subsystem. We needed to spread the power subsystem across both units as there are components on both sides that need different voltages. These subsystems are visualized in Fig2.

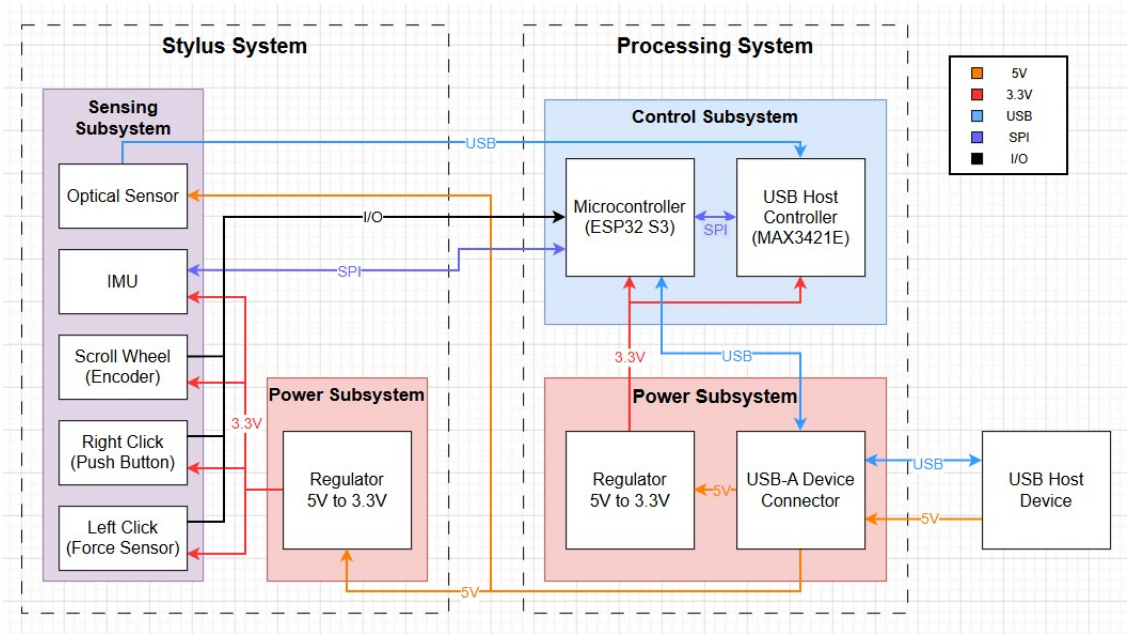


Figure 2: Block Diagram

2 Design

2.1 Component Selection

Force Sensor: There are many ways to detect if the stylus is making contact with the surface: a force sensitive resistor (FSR), pushbutton, hall sensors. We chose to use a force sensitive resistor as they are flat, lightweight, and simple making them perfect for use in our stylus. A force applied to the pad of the resistor reduces its resistance according to the curve shown in Fig. 3 [1]. To choose the best one for our design, we measured the force typically exerted while writing and made sure that the FSR would lose most of its resistance in that range. This was tested and verified upon getting the actual FSR. By tuning a voltage divider we can connect the sensor to a ADC pin on the ESP32 and convert to a serial value that can be used and tuned to our needs. A sponge-like piece on the tip will act to disperse the force of a push across the entire pad of the resistor.

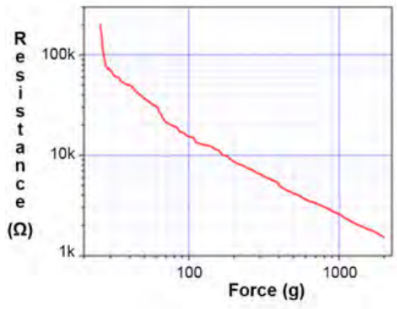


Figure 3: Force Resistance Curve - FSR

Optical Sensor: We opted to use a common computer mouse sensor, the MX8733B. This optical sensor implements many of the functions we need in a very compact case. It also has the advantage of being very inexpensive and has a wider 2mm range of operation before it detects liftoff, making a slightly mobile tip possible. It also processes the sensor data into USB differential signaling and contains on on-board LED driver [2].

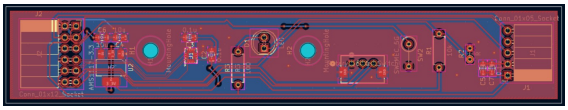
Microcontroller: The ESP32 was chosen to provide the control to the circuit. It has a large number of pins for connecting all the sensors and buttons. It is compatible with analog signals, SPI communication, has many GPIO pins, and differential USB output. The processor is plenty fast and capable for the software implementation. Additionally, the dev-board also comes with the voltage regulator to handle power conversion between 5v and 3.3v.

Inertial Measurement Unit: To track movement when the stylus is lifted, we decided to use an IMU. We specifically chose the BMI160 because it is lightweight, compact, and was configurable for both acceleration and gyroscopic movement. We took advantage of the gyroscopic tracking for our final implementation of aerial movement. One unforeseen challenge of this IMU was the size of it. It was very hard to solder the pins onto the PCB. It would have been beneficial to sacrifice some of the size and weight to have an easier

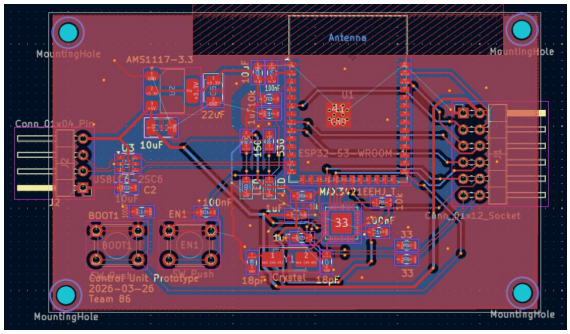
time soldering it on.

2.2 Physical Design

PCBs: Our PCBs need to work not only electrically, but also geometrically to fit inside the stylus and make sure all of the components are in the correct spot. The design for the control unit was just a standard PCB design, but the design for the stylus unit needed to be an elongated rectangle with all of the wiring for the buttons in an accessible location near the bottom. We also had to account for components that would be mounted off of the PCB and simple connected with wires.



(a) Stylus Unit PCB



(b) Control Unit PCB

Case: The physical design required coordnation of all the parts and geometries. The stylus would be built of two main plastic pieces made custom through a 3D printer. The body would hold the stylus PCB and the tip would house the optical sensor and lens at the correct height. In the final modifications we had to alter the tip due to printing issues. The CAD model is shown in Fig. 5, the yellow piece is the main body and the light blue piece is the tip. The FSR sits between the back of the tip and the cross-piece in the body.

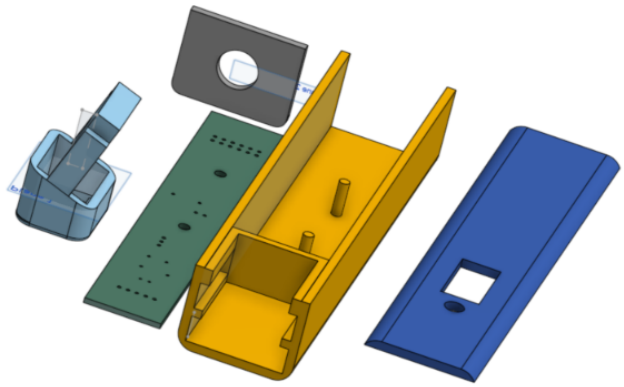


Figure 5: Stylus CAD Model

2.3 Software Design

The software will be running on an ESP32 which will be responsible for interfacing with the 5 sensors in the pen, computing cursor movement, and communicating over USB to the host device. This firmware for this will be written in C using the ESP-IDF. We will take advantage of the dual core functionality of this chip and use one for processing sensor info and the other for USB interfacing. They will communicate using an RTOS queue where the sensor core will publish its data to a queue and the USB core will subscribe and link that to the USB host device.

2.3.1 Sensor Processing Firmware

The sensing core will function using a timer interrupt every 8ms to reach the necessary timing restraints for a smooth cursor movement. Depending on the current state we will read either the optical sensor or IMU output followed by the FSR, push button, and scroll wheel.

States: The pen will have a three state framework that allows it to perform in multiple environments.

INITIALIZE: All sensors will be initialized and the link between the host and the device will be established.

SURFACE: This state is the state after initialization finishes and the FSR is above the threshold. The optical sensor is in control of the cursor movements when in this state.

AIR: This mode is entered when the FSR reading is below the threshold and the IMU takes over the control.

Transitions:

- INITIALIZE$\rightarrow$ SURFACE OR AIR: After power up and all the sensors have been initialized the state will then transition to either SURFACE or AIR depending on the FSR pressure level.
- SURFACE $\rightarrow$ AIR: This happens when the pen FSR pressure transitions from above to below the threshold. The control then goes from the optical sensor to the IMU.
- AIR $\rightarrow$ SURFACE: This happens when the pen FSR pressure transitions from below to above the threshold. The control then goes from the IMU sensor to the optical sensor.

2.3.2 USB HID

The ESP32 has a naive TinyUSB stack that is included within the ESP-IDF framework. This allows the device to present itself as a standard HID mouse class device.

The HID report descriptor defines the mouse with three buttons left, right, and middle click, a 16 bit signed field for the X and Y displacement, and an 8-bit signed vertical

scroll field. The report will be 4 bytes total. We will map this to our buttons using the framework:

- left click: FSR pen-down state
- right click: push button state
- X, Y displacement: outputs of state machine (either optical or IMU displacement)
- Scroll: scroll wheel accumulator value

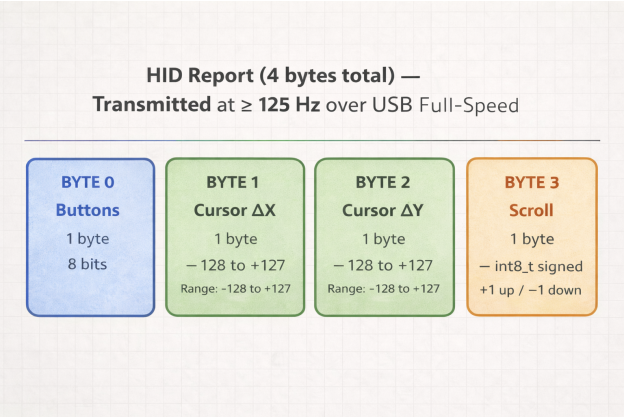


Figure 6: Report Layout

The USB Core will run the TinyUSB task continuously. Whenever there is a new packet that is available in the RTOS queue, published from the sensor core, the USB Core will dequeue it and reports the queue over USB to the host device. This must be submitted at a rate of 125 Hz in order to meet baseline cursor smoothness.

2.4 Tolerance Analysis

Power Requirements: Power is supplied only through a USB cable. This power is nominally 5v but is guaranteed to range between 4.45-5.25V minus the voltage drop across the cable, which should be negligible at the low currents being drawn.

The 5v from the USB connects to a regulator that drops the voltage to 3.3v $\pm$ 0.3v. The AMS1117 low dropout voltage regulator is used to provide a steady 3.3v output from the 5v input. This regulator has an operating output range between 3.2 and 3.3[v] which is well within the tolerances of the devices used.

The optical sensor (MX8733B) requires a 4.5-5.5v operation. So it can be connected directly to the USB provided 5v. It should draw maximum of 15mA during operation and 7.5mA when idle. This is not an issue for any USB hosting device. The LED is driven from a circuit internal to the optical sensor IC so the power draw is included in the 15mA current [2].

The ESP32 operates at 3.3v and 44mA idle and up to 180mA during peak activity including wifi-transmission [3]. Because we are not using the wireless features the current

spikes should be limited. This means that even at the highest power draw expected from all components, the regulator is under rated power and should not overheat.

Optical Sensor: The optical sensor has an operating distance. It first starts working when the bottom of the lens is 2mm off the surface and is unable to detect movement when lifted 4mm off the surface, giving a 2mm operating range. The sensor works most reliably when the surface is illuminated at an angle from a single light source. The stylus will need to block ambient light and supply its own for the best performance.

IMU: The IMU is responsible for sensing the less precise movements when the stylus is too far from the surface for the optical sensor. It uses tilt and acceleration data which is translated to XY movement. The sensor contains analog to digital conversion within the IC which is available in single nano-second time scales [4]. the SPI data must be translated to XY mouse controls within the microcontroller unit, which takes micro-second time scale. Even putting the delays together the sensor should react plenty-fast to feel instant to a user. The combination of tilt and acceleration may require some skill to use but with proper tuning should be able to move the cursor with the pen lifted off the table.

2.5 Design Alternatives

Bluetooth Connection: The two subsystems are connected via physical wires. In the early stages of the project we discussed the alternative of using a bluetooth system but decided the wired would be better. A wireless system adds a lot of complexity to the device, especially the stylus itself. A wireless system would require the addition of a processing system, antenna, and battery to the stylus. On top of the added size, creating a wireless system decreases the pens reliability by adding significant latency up to 100ms, requiring the need for charging, and adding the need to establish a wireless connection prior to operation.

Optical Flow Sensor: The cornerstone of the project is the devices' need to sense the precise movements needed for writing. Ultimately, we landed on using a two sensor system: a mouse optical sensor and a 6-axis IMU. The traditional computer mouse optical sensor is desirable because they work at distances expected from a stylus and at 1000 DPI (Dots Per Inch) it is able to detect all the precise movements required for writing. The IMU motion detection is perfect because it is able to provide input regardless of the stylus position or orientation. Before landing on these sensors for a two-sensor system we researched using other optical flow sensors. These devices are typically used in drones to detect movement by comparing images of the ground below them, similar to how a mouse works. However, there are no optical flow sensors commonly available that have an operational range between the distances needed for a writing tool.

STM32 Microprocessor: While selecting components, we realized that we had an issue connecting up the optical sensor with the ESP32. The optical sensor sent out differential USB data, but the ESP32s3 only has one USB differential data port which we needed for communication with the computer. To solve this issue we considered switching to an STM32 microprocessor with more USB differential data ports. However, after some consideration we decided instead to use a MAX3421E chip to convert the differential USB

signal to SPI and send it to the ESP32 that way. Our main reasoning for this was that the STM32 would be overkill for our design in terms of performance and complexity and it would be easier add the MAX3421E chip rather than redesign for the STM32.

3 Cost

3.1 Materials Cost

Table 1: Bill of Materials

Part Name	**Qty**	**Cost (total)**	**Description**
ESP32-S3	1	$6.00	Microprocessor
MAX3421	1	$12.29	USB peripheral controller
ECS-120	1	$0.39	12 MHz crystal oscillator
4518-USBLC6	1	$0.39	USB ESD protection diode
USBC-31	1	$0.46	USB-C port
SS14	1	$0.44	Schottky diode
T491B106K010AT	1	$0.66	10 μF tantalum capacitor
BMI160	1	$4.17	6-axis IMU
34-00004 FSR	1	$7.12	Force-sensitive resistor
MX8733B	1	$1.00	Optical sensor + lens
EC10E	1	$0.92	Rotary encoder
PTS636	3	$0.63	Tactile push button
N/A	3	$0.22	Indicator LED
AMS1117	2	$0.20	5 V $\rightarrow$ 3.3 V LDO regulator
N/A	15	$0.20	Capacitors (assorted)
PCB Control Box	1	$5.00	Custom PCB
PCB Stylus Pen	1	$5.00	Custom PCB
Wires	–	$1.00	Connection wires
3D printed case	1	$5.00	Enclosure (PLA)
	Total	**$51.09**	

3.2 Labor Costs

Each team member spent approximately 5 hours per week over the 16-week semester. With 3 team members, this comes to 480 total person-hours. At the average ECE graduate salary of $80,000 ($\approx$ $38 per hour), the projected labor cost is:

$$\text{Total} = \frac{\$}{\text{hour}} \times 2.5 \times \text{team members} \times \text{hours per member}$$

$$\$22{,}800 = 38 \times 2.5 \times 3 \times 80$$

3.3 Schedule

Table 2: Project Schedule and Task Assignments

Week	**Task**	**Person**
Feb 23 – Mar 1	Order parts	Everyone
	Design document	Everyone
Mar 2 – Mar 8	Test sensors	Everyone
	Prototype	Everyone
	Design review	Everyone
	Work on PCB design + order	Everyone
Mar 9 – Mar 15	Design and print housing	Everyone
	Pass audit	Everyone
	Breadboard demo	Everyone
	Team evaluation 1	Individual
	PCB order	Everyone
	Software implementation	Everyone
Mar 16 – Mar 22	Spring break	Everyone
Mar 23 – Mar 29	Last PCB order	Everyone
	Assembly	Everyone
Mar 30 – Apr 5	Individual progress report	Individual
	Final assembly	Everyone
Apr 6 – Apr 12	Progress demo	Everyone
	Team contract assessment	Everyone
Apr 13 – Apr 19	Bug fixes + verification	Everyone
Apr 20 – Apr 26	Mock demo + presentation	Everyone

Week	**Task**	**Person**
	Final tweaks	Everyone
Apr 27 – May 3	Final demo + presentation	Everyone

4 Verification and Results

4.1 Sensor Subsystem

Table 3: Sensor Subsystem Requirements, Verification, and Results

Requirement	**Verification**	**Quantitative Result**
The optical sensor must correctly output displacement in X and Y position with an error of no more than 5%.	Move the stylus pen exactly 100 mm along a ruler horizontally and read the cumulative delta-X count from the ESP32 terminal, converting using the sensor calibration (counts/mm). Confirm reported movement is within 95–105 mm. Repeat 5 trials and log results.	**PASS.**Trial results (X-axis): 99.4, 100.8, 101.2, 98.9, 100.3 mm. Mean = 100.12 mm; max error = 1.2% (well within $\pm$5%).
The optical sensor must stop reporting X/Y displacement when the pen is lifted 3.0 mm $\pm$ 0.5 mm from the writing surface.	Confirm displacement output on the serial monitor while moving the pen on the surface. Place a 3 mm spacer block under the tip and verify all reported counts are zero. Record sensor output in a data log.	**PASS.** On-surface motion produced non-zero ΔX/ΔY counts. With 3.0 mm spacer applied, 100% of 60 s of polled samples returned ΔX = 0, ΔY = 0.
When the stylus is more than 3 mm above the surface, the IMU must control the cursor in all 4 directions when tilted, and stop moving when held flat within 5° of horizontal.	With the pen held above 3 mm, tilt 45° in each direction (left, right, forward, backward) and confirm cursor moves accordingly. Return pen to within 5° of horizontal and confirm no cursor movement. Save trial videos and pass/fail data.	**PASS.** 4/4 directions registered correct cursor motion at 45° tilt across 5 trials each (20/20 trials). Cursor velocity scaled linearly with tilt angle. At $\leq$5° from horizontal, measured cursor drift = 0 px over 10 s hold (5/5 trials).
The push button must register a right-click event within 100 ms of the physical press and deregister on release, with no more than one click event per physical press.	Press and release the GPIO button 10 times. Record latency from GPIO falling edge to right-click HID event and the number of click-down events per press.	**PASS.** 10/10 trials registered exactly 1 click-down event per press. No bounce or duplicate events observed.

Requirement	Verification	Quantitative Result
The scroll wheel must register exactly 1 scroll event per detent and correctly distinguish upward from downward direction.	Open a serial monitor on the PC host. Move the scroll wheel up 10 detents, then down 10 detents. Verify monitor shows 10×"+1" followed by 10×"-1". Save data log.	**PASS.** Recorded sequence: ten consecutive "+1" events followed by ten consecutive "-1" events with no missed or duplicated detents. Direction encoding verified across 3 repeated runs (60 events total, 0 errors).
When the stylus is dragged across a surface under constant pressure above the FSR threshold, a left click must remain held without interruption over 100 mm of travel.	Wrap rubber bands around the pen tip to apply force above the left-click threshold. Drag the pen across the writing surface while monitoring the FSR analog output voltage and confirming continuous left-click on the host.	**PASS.** FSR threshold = 1000 out of te max ADC reading of 4096 . During 100 mm drag, FSR held below the threshold. The HID monitor showed left-button-down state continuously for the full 100 mm with 0 release events. Repeated over 5 trials.

4.2 Control Subsystem

Table 4: Control Subsystem Requirements, Verification, and Results

Requirement	Verification	Quantitative Result
The ESP32 must enumerate as a USB HID mouse within 5 seconds of being connected to a host device.	Connect the control box to a Windows machine and observe Device Manager. Measure time from USB plug-in to device appearance with a stopwatch. Repeat 5 trials.	**PASS.** Enumeration times across 5 trials: 1.42, 1.51, 1.38, 1.47, 1.55 s. Mean = 1.47 s (well below 5 s requirement). Device consistently appeared as "HID-compliant mouse."
When the stylus tip transitions from the surface to above 3.0 mm, control of the cursor switches from the optical sensor to the IMU.	On the host, use a serial monitor to verify that once the pen is in the air, cursor data originates from the IMU.	**PASS.** 10/10 lift trials switched cleanly from the optical sensor to hte IMU showing data readings only on one device at a time. 10/10 set-down trials switched back.
The microcontroller must record data from all 5 sensors and relay them over USB to the host with less than 5% data loss.	Run a firmware test that polls all sensors for 60 s and logs to host. Verify each sensor logged the same sample count for the full window. Interact with each input to confirm value changes.	**PASS.** 60 s capture at 125 Hz polling: All sensors output data for each 8ms interval. All inputs registered changes upon interaction.

4.3 Power Subsystem

Table 5: Power Subsystem Requirements, Verification, and Results

Requirement	**Verification**	**Quantitative Result**
The power converter must step down the 5 V input to 3.3 V $\pm$ 0.3 V for all subsystems.	Use a digital multimeter to verify the input voltage to each sensor rail is within range under load.	**PASS.** AMS1117-3.3 output measured at no-load = 3.31 V; under full load (165 mA) = 3.27 V. All sensor rails within 3.0–3.6 V tolerance.
The USB-A connector must be recognized when plugged into a USB host device.	Open Device Manager on the host and confirm the device enumerates as an HID with no "Device Not Recognized" error. Test on 3 separate hosts.	**PASS.** Enumerated successfully on 3/3 hosts (Windows 11, as "USB HID-compliant mouse.") No error messages observed.

5 Conclusion

5.1 Accomplishments

At the completion of our project we were able to successfully build a working stylus pen in a user friendly printed case. The final product resembled a prototype of what we would want the final version to look like. All of our first initial high level requirements and functionality was met with the exception of the weight of the pen. We got a working pen that plugged into any windows computer and enumerated as an HID device. The pen could then use both the optical sensor and IMU to control the cursor on the screen very smoothly with little latency. Then we also had the left click tip button as well as the right click push button working on the pen case. Luckily we had a scroll wheel built in to the case that allowed us to flawlessly scroll on the host device. All of the functionality of a mouse was implemented and worked well. Both PCBs worked without any issues or need of breakout boards.

5.2 Uncertainties

In terms of the results that were not what we wanted, the weight was the only thing. After having a finalized product we were left with a final pen weight of 45g which is higher than the proposed 30g maximum. This is something that was very minor to the functionality of our device but it does add user friendly characteristics to our device when the 30g maximum is met. For us the main issue getting to this weight was the 3D printed case. We did not have much experience or time to iterate and develop a case that was very optimized, but rather made a functional design. This allowed us to have a prototype that worked but lacked ergonomics. This resulting issue could be resolved with more work on CAD and several 3D prints to built something that is smaller and more compact but still user friendly. The PCB itself is very light but if we designed that to be smaller the case could be even smaller as well. Overall this was our only main unsatisfactory result in our final product.

5.3 Future Work

To start, we would take more time to develop a refined 3d case for the pen that would be lighter as well as feel better in the hand. The PCB for both the control box as well as the stylus pen could be smaller and lighter. In terms of the optical sensor we could find a smaller optical sensor and have a smaller tip so that it goes on the surface smoother. In software we can add macros to the IMU as well as the buttons to have more features for our pen. The IMU can also be optimized to move smoother for more complex cursor control. Bluetooth could have also been implemented to make our design complex. USB-C interfacing would be another good addition since that is the new standard for wired connection.

5.4 Ethical Concerns

This project aims to make digital writing more accessible and convenient through a stylus-based input device, with affordability and environmental responsibility as core design priorities — including the potential to reduce paper and printer use. We are committed to following the IEEE Code of Ethics (particularly Section I, codes 1, 2, and 5) [5] and the ACM Code of Ethics (codes 1.1, 1.6, and 2.1) [6] throughout development. The primary safety concern is the optical sensor's laser, which may require a longer focal distance than a typical mouse and could pose an eye-safety risk if misused; we will adhere to LIA laser safety guidelines [7] and follow good wiring practices to mitigate any low-voltage thermal or short-circuit risks. For privacy, the wired USB design provides a secure communication channel that prevents third-party interception, and no user data is stored on the device itself.

References

[1] *Fsr® 400 series data sheet*, https://cdn2.hubspot.net/hubfs/3899023/Interlinkelectronics%20November2017/Docs/Datasheet_FSR.pdf, Accessed: 2026-02-27.

[2] *Mx8733b usb single chip optical mouse sensor*, https://datasheet4u.com/pdf-down/M/X/8/MX8733B-LIZEElectronic.pdf, Accessed: 2026-02-27.

[3] S. Prabhu, *Insight into esp32 sleep modes their power consumption*, https://lastminuteengineers.com/esp32-sleep-modes-power-consumption/, Accessed: 2026-02-27.

[4] *Icm-40609-d datasheet*, https://invensense.tdk.com/wp-content/uploads/2022/07/DS-000330-ICM-40609-D-v1.2.pdf, Accessed: 2026-02-27.

[5] IEEE. "IEEE Code of Ethics," Accessed: Feb. 10, 2026. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.

[6] ACM. "ACM Code of Ethics," Accessed: Feb. 27, 2026. [Online]. Available: https://www.acm.org/code-of-ethics.

[7] LIA. "LIA Laser Saftey Information," Accessed: Feb. 27, 2026. [Online]. Available: https://www.lia.org/resources/laser-safety-information?srsltid=AfmBOorlHd2w8-x3POavt33yMtar3aLtTom_zEYMQNBG1uMVKjFXQS_X.