FULL MOVEMENT GAMING MOUSE

By

Drake Bernhard
Michael Bindon

Final Report for ECE 445, Senior Design, Spring 2018
TA: Anthony Caton

2 May 2018
Project No. 23

Abstract

This project details motivation for and creation of computer mouse with an analog and digital joystick for use by the thumb alongside with the typical inputs of a mouse. Though the team experienced difficulty with implementing some of the design, the project achieved promising successes in individual portions of the design that will carry over to future designs.

Contents

1. Introduction

The objectives of this project are to create a personal computer gaming mouse that can be operated using only one hand, has the complete functionality and intuitiveness of a normal mouse, and adjustable sensitivity for all users. Other PC gaming mouses on the market typically have a two dimensional joystick. As a result, the desired direction of the joystick is corrupted by the thumb's arc-like movement [6]. However, the mouse in this project preserves the true desire of movement by placing the user's thumb on a guided joystick. The joystick is made up of analog slider in the horizontal direction and digital switches in the vertical direction. For this reason, we believe our concept is superior to existing full direction mouses available today.

For a successful full movement gaming mouse, the product will have a control unit, optical sensor, mechanical input in the form of a joystick, and power source coming from the user's computer. In the upcoming sections the individual design modules of this project will further explored. These modules include the joystick, input switches, optical sensor, microcontroller module, and USB communication. Afterward the design modules and their verifications will be explained.

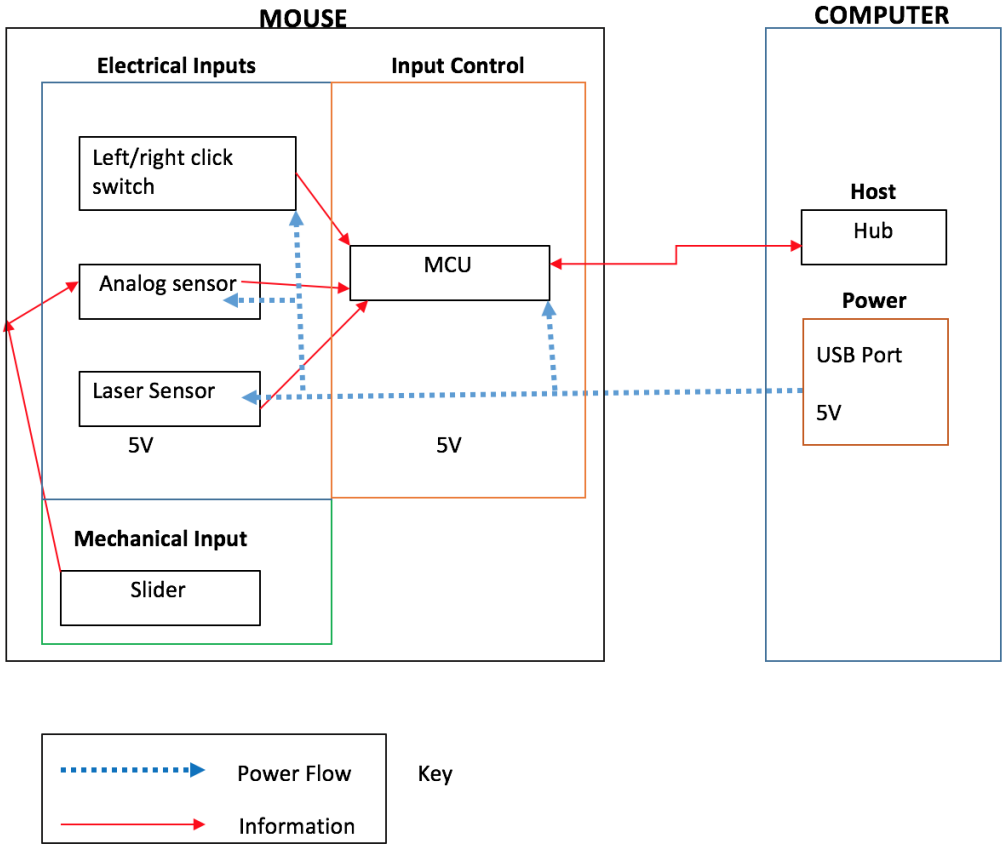

Figure 1 Block Diagram of Major Module Components

2 Design

Our project had to keep a constant balance between size, complexity, comfort, feel, and performance. Doing so requires careful planning at all stages in the design. This section will discuss the full design process for each module individually while making note of design choices that impacted multiple blocks.

2.1 Joystick

The joystick is the piece of the design that addressed our motivating problems directly. As such it was very important to achieve a good design of this component, especially since the joystick will interact very intimately with the end user.

The joystick design requirements were directly motivated by the high-level requirements derived from our original problem statement. We wanted to provide a range of motion, as opposed to just digital directions, that will also be guided so as to give a more true response to natural motion of the thumb. In addition to this the joystick, combined with the mouse body, must be comfortable enough to allow for multiple hours of almost uninterrupted use in order to be a practical mouse for gaming sessions and everyday use.

These general requirements were the starting point for researching how we would make and install our joystick. We researched many options but eventually agreed that a linear potentiometer mounted on a 3-position rocker switch would give the characteristics we were after. The linear potentiometer would have to be very compact as we estimated the thumb would have no more that 20mm of travel at an absolute maximum. The potentiometer would also have to not be too stiff to avoid it taking too much force to move. In addition, ideally the potentiometer would have some notch to signify it was in the central position. We found a potentiometer that fit these requirements perfectly. The other component was a 3-position rocker switch which would provide left, right, and resting positions and would directly replace the ‘A’ and ‘D’ keys that a user would usually use to navigate in a game. These components would also have to be joined together through a 3D printed part (and super glue) in a way that could be integrated into the side of a computer mouse without protruding too much to where it would cause discomfort.

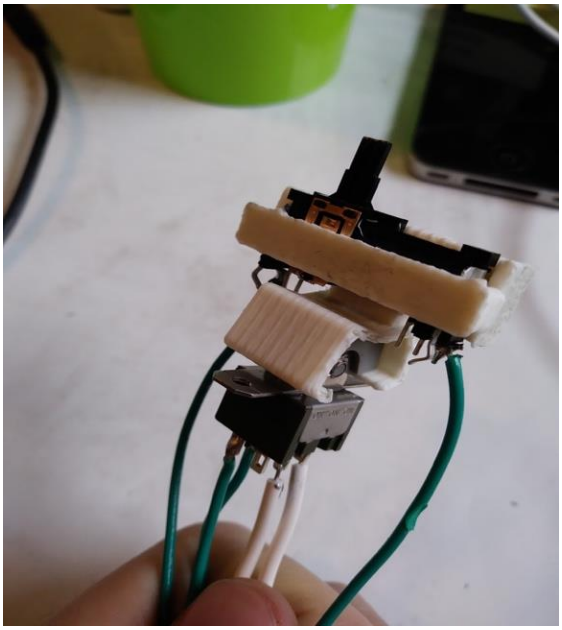

Figure 2 Rocker/Joystick Combination

The analog portion of the joystick is generated by the linear potentiometer through a voltage divider. This voltage divider required careful selection of the fixed resistor to be used. A resistance that is too high would cause too small of a change in voltage across the variable resistor. This could make it difficult to distinguish precisely where the linear potentiometer was at, or could require accuracy at maximum resolution which could add time to code execution as well as present the opportunity for noise to cause jitter in the value returned from the ADC.

Alternatively, a resistance that is too small would cause extremely non-linear voltages on the analog input pin with respect to the position of the potentiometer. The optimal resistance is the highest fixed resistance value that still provides enough voltage variability to have the necessary resolution. The lower the resistance the more linearity issues there will be which can cause issues with feel. Doing calculations showed that having a fixed resistance of anywhere from 1 to 1.5x the variable resistors maximum value would be ideal. The circuit schematic below shows resistor R1 as the voltage divider for the analog slider.

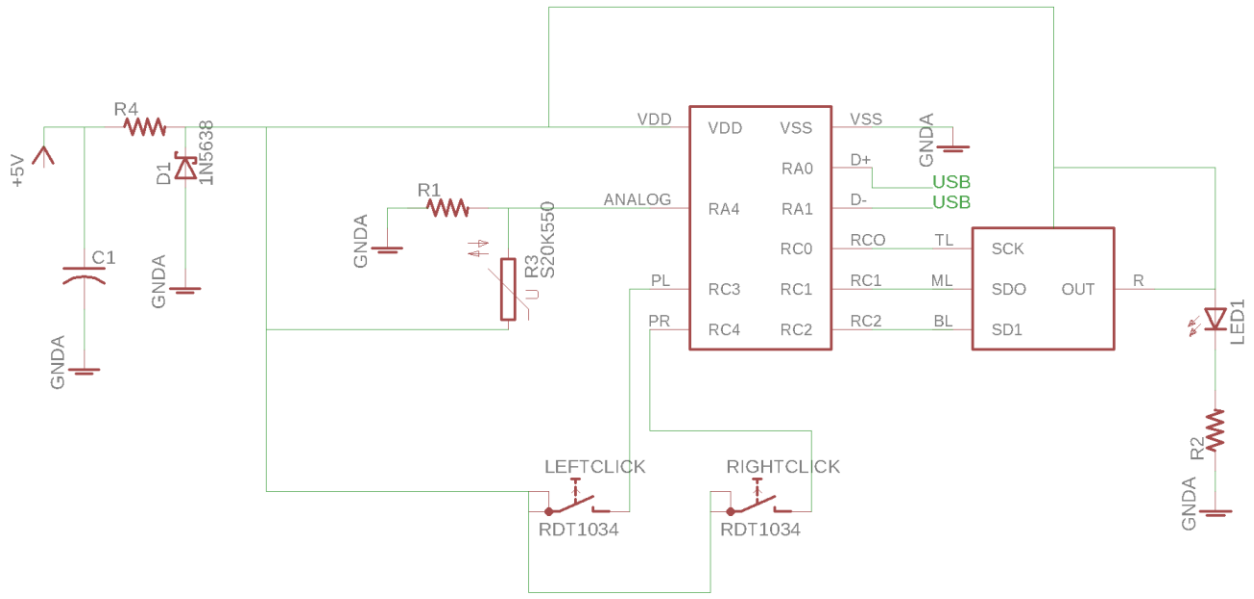

Figure 3 Simple Circuit Schematic

While we did design to provide better linearity it was also important to recognize that some correction should be done on our input data in order to map our analog voltage to the response that we would send to the host computer. Since our analog voltage would range from 2.5-5V with the designed circuit, we needed a way to map these voltages from -127 to +127. It is also important to have a fairly wide 'dead zone' of voltage values that would all cause a movement value of 0 to be sent to the host computer. This would correspond to the center mouse position and immediately touching the center catch on either side. A similar concept would be applied to both extremes of the mouse slider causing mapping to maximum positive and maximum negative values. The remaining voltage levels could then be mapped using a formula such as one shown in figure 4.

- Center dead zone region = 3.15V— 3.45V (~6.5 - 7.5 mm)
- Up direction sensitivity of joystick = $\frac{5V-3.45V}{127-0} = 12.2\ mV/bit$
- Down direction sensitivity of joystick = $\frac{3.15V-2.5V}{0--127} = 5.1\ mV/bit$

Figure 4 Analog Voltage Digital mapping Formula

There was one more consideration that use of the microprocessor's analog input necessitated. This was the acquisition time of the analog to digital conversion. Initially we had some concern that due to our high impedances connected to our analog input pin the internal capacitors used by the ADC would take too long to change up (several ms). We found that this was a valid concern, but further research showed that this excessive amount of time would only occur during the initial analog setup and acquisition since after this point the capacitors would already be charged to either the correct level or close to it. We also found that given our voltage divider we would not need all 10 bits of output in order to have the resolution we were after. Our example calculation is shown in the equations below.

$$T_c = -C_{hold}(R_{lc} + R_{SS} + R_S) * \ln(1/2047) \quad (1)$$

$$= -13.5pF * (1k + 7k + 60k) * -76.24 \quad (2)$$

$$= \sim 7\ ms \quad (3)$$

2.2 Right and Left Click

The other physical device that the user will feel and interact with is the mouse buttons. These buttons should have a satisfying click as well as be able to hold up to long term use and rapid bursts of clicks. One additional constraint that we had to keep in mind was the size of the mouse buttons and how they would be pressed from the user pressing down on the casing above the buttons [4]. We looked into options for the mouse buttons that were designed and used largely for mice. The buttons also should be mechanically debounced properly to help improve stability of the mouse.

2.3 Position Sensor

For our optical sensor there are two major technologies, laser and optical. Each one scans the ground below the camera and keeps relative position. Laser technology can have good readability on a wide variety of surfaces that would give optical sensors trouble. We decided to choose the optical sensor to avoid additional circuitry and lighting that would be required for the laser sensor. We weren't concerned with the surface since our mouse is begin designed for desktop usage, especially for gaming. We also felt comfortable since this was a wired mouse which is much less likely to be carried around.

A major criterion in selecting the sensor was lens. The lens needs to be aligned very tightly with the camera hole. The lens also needed to be built for a very close range. We had a very difficult time finding lenses that would work well with the sensor. Most optical sensors have a recommended lens however most of these lens were not available to be purchased. We also had to take the positioning of the optical sensor in the mouse into account. It is important that the optical sensor is fairly central on the underside of the mouse.

We also had to make sure that the interface would work. Most products had either SPI or I2C interface. We had to check that the microcontroller would be compatible with serial interface.

A final point of consideration was DPI (dots per inch). DPI is a measure of spatial resolution corresponding to how many counts a movement of one inch would be expected to cause to accumulate. In selecting a optical sensor we wanted to have enough hardware capability to be able to have our mouse be able to use smaller movements to be able to traverse computer monitors, even high resolution and dual monitors. For example, a high resolution computer screen is typically 2560x1440. This means that it would take about 5000 counts to move completely across 2 screens. Our sensor should have a DPI of at least 1000 to allow for a typical five inch range of movement to allow the mouse cursor to move completely across dual monitors.

We eventually selected the CJMCU-110 optical position sensor. This sensor uses SPI interface, has options for 400 or 1600 DPI, and we were able to find a lens that would work with the sensor.

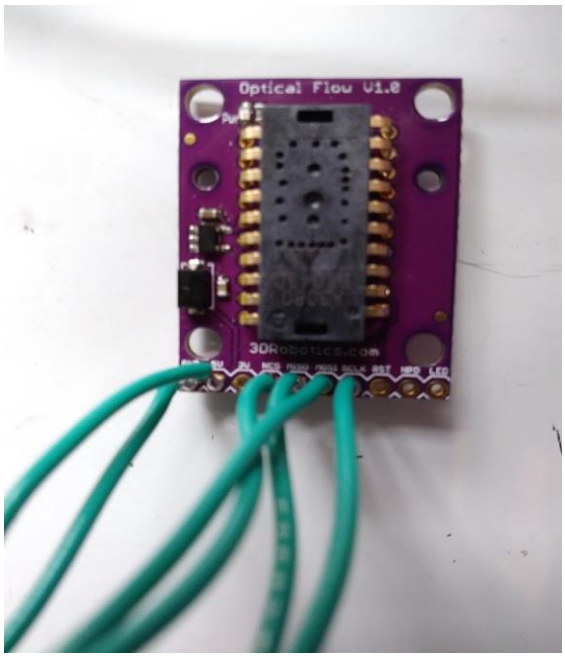

Figure 5 CJMCU -110

2.4 Microcontroller

The microcontroller implemented in the mouse is the PIC16F1459. This PIC was chosen for three main reasons: cost, USB implementations, and number of inputs. The microcontroller is considerably cheaper than an FPGA. Additionally, it can handle 5 V inputs, has pull-up resistors, and internal 48 MHz clock for USB. This chip also has an analog input, serial protocol interface, and 20 pins [7].

The main responsibility of the PIC16F1459 is to handle the inputs from the SPI, joystick, left and right clicks. With the joystick analog slider, the MCU must also create a center dead zone from 3.15 V-3.45V and linearize each region on either side of the dead zone. For the communication between host and the mouse, the PIC must be able to respond to the host interrogation of the device and supply the necessary information when needed.

2.4.1 USB Module

The design of the USB module follows the USB protocol which is shown in the software flowchart below. For the mouse to communicate with the host computer, the PIC16F1459 needs to be able to respond to the host's request in a timely manner in addition to the correct sequencing. First the host recognizes the mouse is connected when the data lines are no longer connected to the ground and the mouse has a pull resistor on the D+ line. Next the USB resets

the mouse because the host knows the state of the registers after the reset. In the third step, the host request the size of the device descriptor over a temporary address. Again the computer will reset the mouse so it knows the mouse's state after the reset. Afterwards, a permanent address is assigned to the USB device. Finally, in the upcoming steps the receives the entire device descriptor which describes the serial number, manufacturer, and product [1]. The configurator describes the power of the device, self-powered, and number of interfaces. Next the interface describes the mouse as a human interface device and the number of endpoints. Finally, the endpoints describe the data going from the mouse to host. [3]

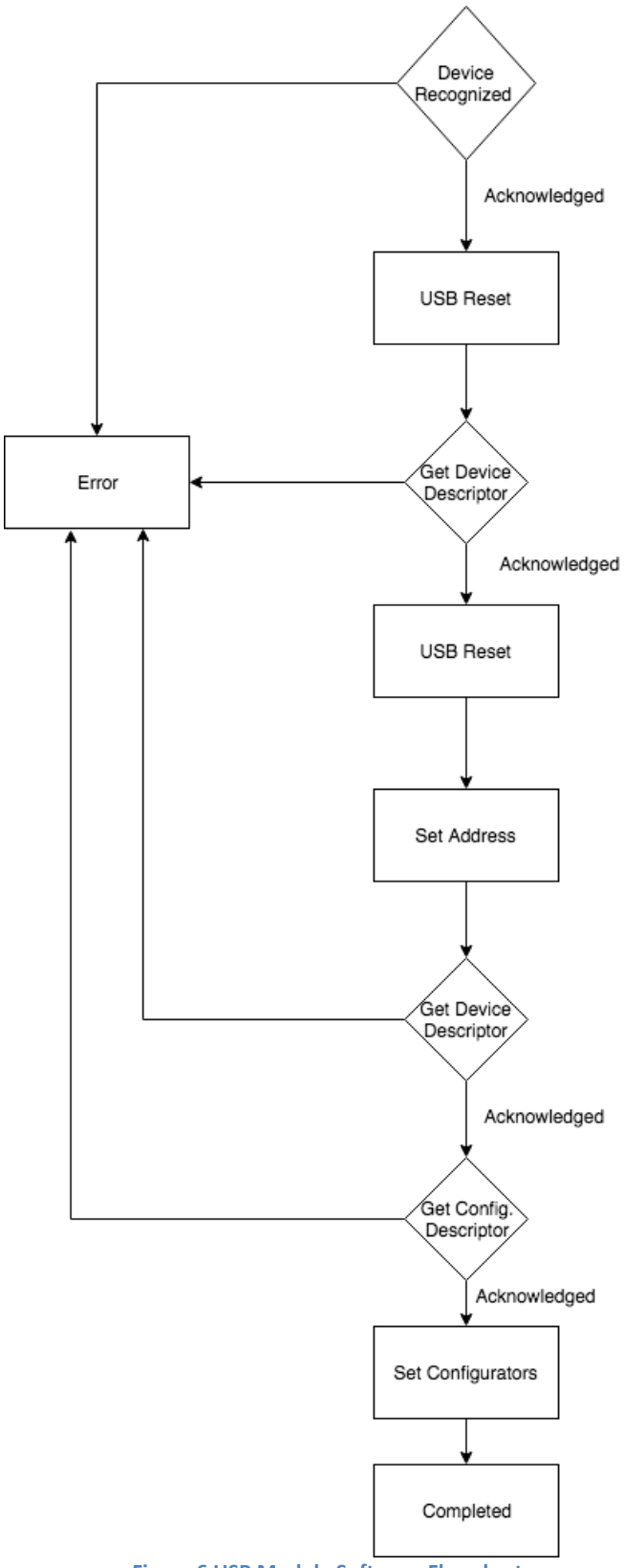

Figure 6 USB Module Software Flowchart

2.5 Physical Design

Our mouse had a few criteria in regard to the physical design in order for our final product to be successful. The biggest of these was general feel of the mouse, for example having buttons that weren't too soft or too stiff. The joystick also needed to be in a tightly defined position that will allow for the thumb to reach it as well as move around comfortably. Our team had access to a 3D printer and we leveraged that to iterate through prototypes quickly. As opposed to many groups, whose 3D printing needs include a box structure, or a simple mount, our group needed to make small adjustments to the general mouse body to get the best comfort as well as achieve very tight tolerances on our joystick molding. These requirements made external 3D printing options and the campus printing services undesirable. While doing the physical design we had to ensure that we would be able to have enough internal space for all the separate components to fit. This space constraint motivated our PCB design to try to remain small as possible and eventually pushed us to use multiple separate hardware pieces that could be made to fit more flexibly in the space inside the mouse.

3. Design Verification

3.1 Joystick

To verify that the joystick slider was communicating with the MCU a simple test was performed. An LED was set as output that would only light up if the hex value from the analog slider was greater than its midpoint value of 0xA8. From this test, when the slider was moved below the midpoint the LED turned off and when it was above it the LED turned on. Therefore, the analog input of the slider was able to be successfully read by the microcontroller.

Having selected the components, we then had to verify that they would meet our expectations. For our joystick we wanted a movement range of 10-20mm of travel in the analog forward/backward axis (the potentiometer) and anywhere from 5 to 20 mm of travel in the left/right discrete direction. Both of our selected components were measured to confirm that they would meet these criteria after being joined together. The rocker switch has 10 degrees of rotation in either direction. By having the rocker on bottom we extended the travel to about 5mm which achieved a very nice feel. Also the linear potentiometer had 13.5mm usable travel, which we found to be very comfortable.

Given our choice in inputs to make up the joystick we had to leave 3 pins on our microcontroller dedicate to the joystick. Our joystick also placed the constraint on the microcontroller that it would have to have support for analog input pins. We also needed to confirm that we would have good, repeatable response from movements on our analog slider. We determined previously we felt we could notice a smallest movement distance of 1 mm. With that we verified that we would be able to detect this small of a distance. We took data of the slider's position, resistance, and output voltage of our voltage divider. This data is compiled below in figure 7. As stated previously, we knew that we would be adjusting the results of our ADC to a few different ranges in order to provide output that would better align with expectations. This was implemented in software after confirming that we were able to detect small enough movements accurately.

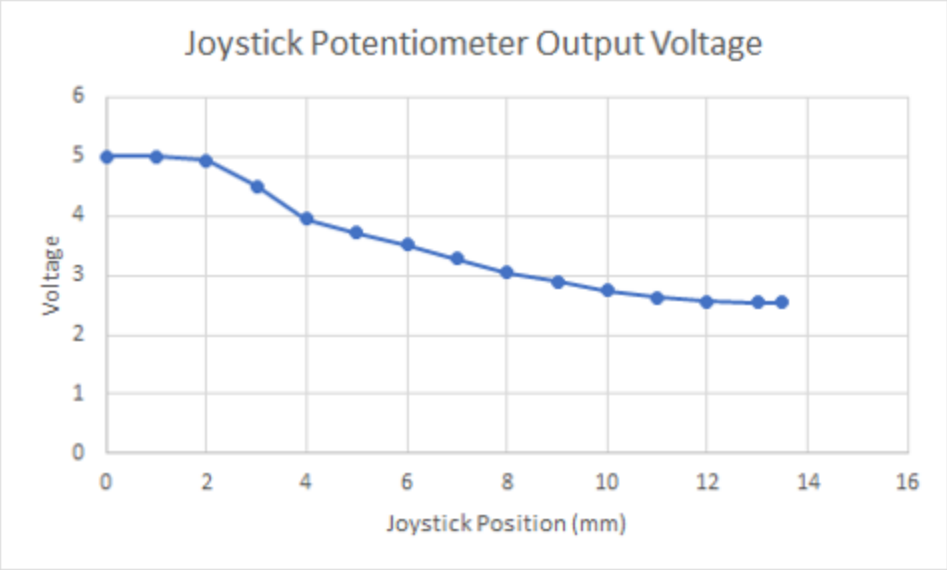

Figure 7 Voltage of Linear Potentiometer vs. Position

3.2 Left and Right Click

The switches of our circuit, which include the left/right click buttons and the up/down rocker switch in our joystick, were tested using a simple LED test to verify the microcontroller was reading the information. When a switch was pressed the LED would turn on; otherwise, the LED would remain off. This test went according to plan and the data read between the PIC16F1459 and digital switches were verified. We also managed to confirm that our hardware could handle 10 state changes per second through viewing settling times of state changes as in figure 8.

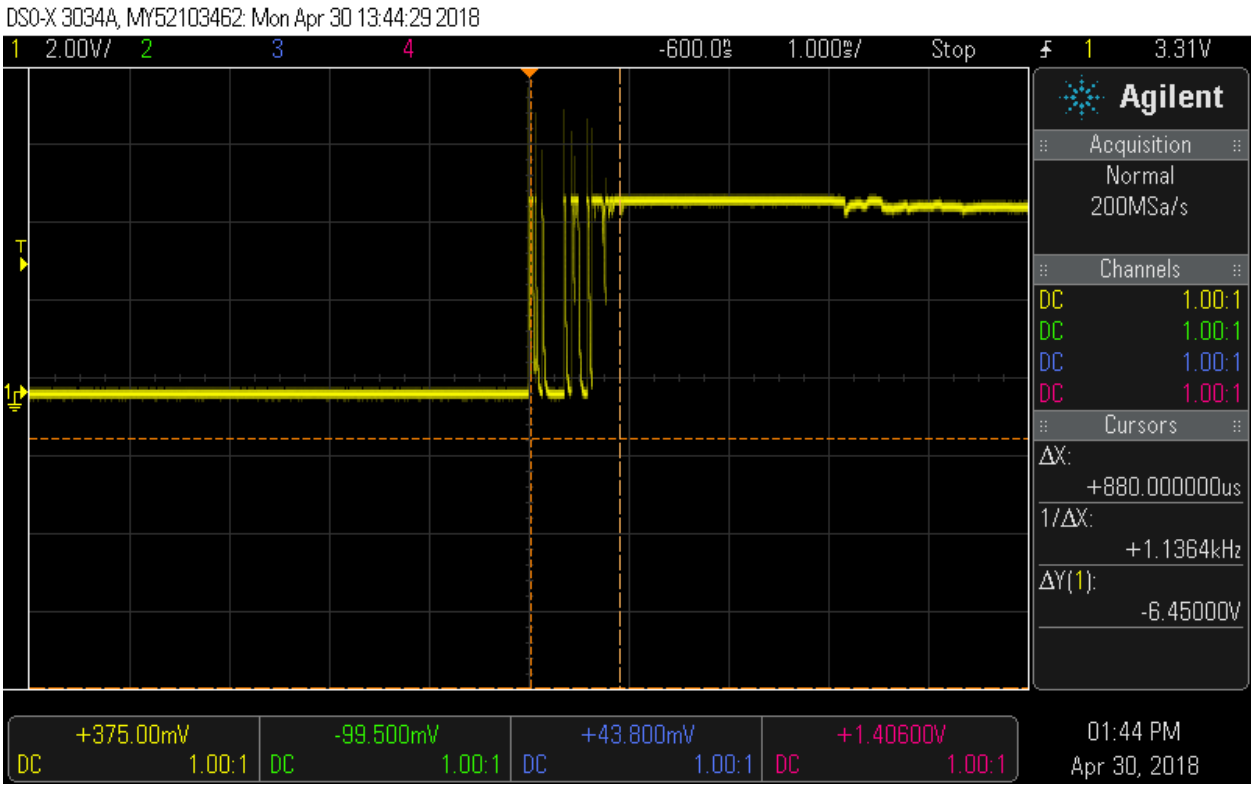

Figure 8 Rise Settling Time for Mouse Click Buttons

While we could confirm the hardware was capable, due to difficulty with USB implementation we were unable to confirm that the high click rate could be recognized during full product execution. Specifically, we were unable to test the update rate of our buttons while the microprocessor was also handling SPI communication, performing analog reads, and handling USB communication.

3.3 Optical Sensor

The optical sensor selected was tested to work with the Arduino SPI library and then tested and confirmed to be compatible with the microprocessor. Figure 9 below shows an oscilloscope capture of the 4 wires used for SPI communication between the optical sensor and the microprocessor. In the figure the top waveform is the SPI clock, second is the MOSI, third is MISO, and bottom waveform is the slave enable. In this capture the host (microprocessor) pulls down the slave enable line, and starts the SPI clock. On each positive edge of the clock the slave will read the value of MOSI [2]. In the first byte transmitted this is read as 0x0A, which informs the slave that it is requesting a read of register 0x0A. After the clock is disabled to give the slave

time to prepare, the slave transmits the value 0x09 back over the MISO data line. This is the expected response as register 0x0A is a configuration register of the optical sensor that contains 0x09 by default.

Once we were able to establish communication worked we went on to test the accuracy of the relative position tracking. We were able to prove functionality through reading the XPos and YPos registers to prove that values would change as movement occurred. Unfortunately, we had difficulty getting a stable lens alignment that would allow us to confirm the DPI accurately.

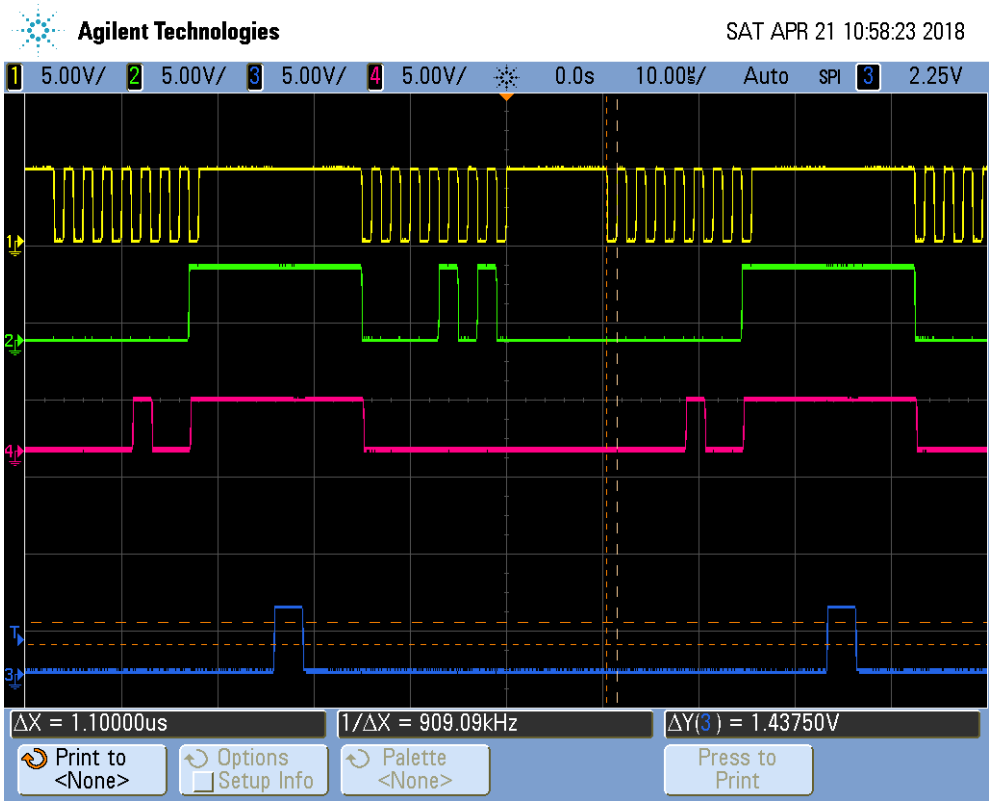

Figure 9 SPI Verification

3.4 Microcontroller

To first verify the microcontroller, PIC16F1459, could be programmed successfully a blinking LED program was the first test performed. This test went accordingly and a blinking LED was visible. This test also helped better understand the compiler, XC8, on MPLAB X IDE before attempting more complicated functions. The communication between the PIC16F1459 and the other inputs such as the switches, analog slider, and serial protocol interface are explained in the sections above.

3.4.1 USB Protocol

To verify if the mouse was adhering to USB protocol the program WireShark was used. The program itself monitors the packets sent and received from the host perspective. Initially, the test proved that the host couldn't get the device descriptor from the mouse. After, testing the hardware for the inrush, data line voltages, and ensuring current draw were according to USB standards, the problem was determined to be the internal oscillator not operating at 48 MHz. The clock not correctly operating would explain figure 10, our first captures using WireShark to test our microcontroller. The host computer was able to recognize the device was plugged and reset it. However, after interrogating the mouse for the device descriptor no input was received because the host's clock and the mouse's clock could not synchronize. After including an external oscillator in the circuit, WireShark demonstrated the device descriptor was able to be read along with the configurator and interface descriptors. Therefore, the USB protocol was implemented on the microprocessor correctly [3].

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	1.1.1	host	USB	28	URB_INTERRUPT in
2	0.555991	1.1.1	host	USB	28	URB_INTERRUPT in
3	1.180032	1.1.1	host	USB	28	URB_INTERRUPT in
4	1.803982	1.1.1	host	USB	28	URB_INTERRUPT in

Figure 10 USB Descriptors with Internal Oscillator

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	1.25.1	host	USB	28	URB_INTERRUPT in
2	0.055933	1.25.1	host	USB	28	URB_INTERRUPT in
3	0.120899	host	1.5.0	USB	36	GET DESCRIPTOR Request DEVICE
4	0.121233	1.5.0	host	USB	46	GET DESCRIPTOR Response DEVICE
5	0.121235	1.5.0	host	USB	28	GET DESCRIPTOR Status
6	0.121258	host	1.5.0	USB	36	GET DESCRIPTOR Request CONFIGURATION
7	0.121504	1.5.0	host	USB	37	GET DESCRIPTOR Response CONFIGURATION
8	0.121505	1.5.0	host	USB	28	GET DESCRIPTOR Status
9	0.121521	host	1.5.0	USB	36	GET DESCRIPTOR Request CONFIGURATION
10	0.121997	1.5.0	host	USB	62	GET DESCRIPTOR Response CONFIGURATION
11	0.121999	1.5.0	host	USB	28	GET DESCRIPTOR Status
12	0.122044	host	1.5.0	USB	36	SET CONFIGURATION Request
13	0.122545	1.5.0	host	USB	28	SET CONFIGURATION Status
14	0.122560	host	1.5.0	USBHID	36	SET_IDLE Request
15	0.122691	1.5.0	host	USB	28	GET INTERFACE Status
16	0.122714	host	1.5.0	USBHID	36	GET DESCRIPTOR Request HID Report
17	0.123335	1.5.0	host	USBHID	80	GET DESCRIPTOR Response HID Report
18	0.123336	1.5.0	host	USB	28	GET DESCRIPTOR Status

Figure 11 USB Descriptors with External Oscillator

4. Costs

The final cost of the project sums up to $12,644.85. However, most of this cost comes from the labor involved with the design process and implementation. Only 0.355% of the costs derives from the parts.

4.1 Parts

For the prototyping of the project we ordered and tested more parts than went into our final design. We tried a few different switches, buttons, sliders, and even optical sensors along the way. In the end however we used a total of $44.85 worth of parts in our finished prototype. A final version would likely able to be built far cheaper through bulk ordering prices and buying the mouse shell from another source. It should also be possible to find a cheaper rocker switch if production quantities make pursuing alternate components worthwhile (1000+ units).

Table 1 Parts and Cost

Manufacturer /Part Number	Description	Purchased From	Quantity @ Cost
CJMCU-110	Optical Flow Sensor Module	Ebay	1 @ $15.40
Alps RS15H113CA05	Slide Potentiometer 15mm	Mouser # 688-RS15H113CA05	1 @ $2.55
Sparkfun BOB-12700	USB A Female Breakout board	Mouser # 474-BOB-12700	1 @ $3.95
NKK Switches M2024TNW01-DH	Rocker Switch ON-ON-ON	Mouser # 633-M2024TNW01-DH	1 @ $8.18
Microchip Technology **PIC16F1459-I/SS**	8 bit MCU, USB, SPI, smnt	Mouser # 579-PIC16F1459-I/SS	1 @ $1.95
	Printed Circuit Board	ECE PCBway order	10 for $50
	3D printed casing	Drake Bernhard	~$5
Common Parts	LED, Click Buttons, resistors	ECEB Parts Bins	~$3

4.2 Labor

The labor turned out to be the largest cost in the creation of the mouse. It was predicted that we would spend about 12 hours per week on this project over 12 weeks. However, that average was closer to 14 hours per week. Additionally, we valued our labor at $30 per hour for this type of work and the qualifications of the group. The final cost of labor is shown in the equation below. The actual cost of the labor is $1800 more than expected at the original design document. This can be explained by the errors experienced in the project that weren't easily resolved and required more time than expected.

$$\$30*12*2.5*14 = \$12,600 \quad (4)$$

5. Conclusion

5.1 Accomplishments

For this project, the microcontroller was able to handle the analog data from the joystick slider, all inputs from the switches, and the serial protocol information. Additionally, the USB protocol was accomplished by using an external oscillator instead of the internal oscillator inside the PIC16F1459. Finally, the physical design was 3D printed and tested with a focus group of 10 people and passed their comfort standards.

5.2 Uncertainties

After troubleshooting the errors in the USB protocol and coming to the conclusion the internal oscillator was not operating at 48 MHz, an external oscillator was ordered. However, the external oscillator didn't arrive in enough time to integrate the SPI and analog slider functions with the polling of the host computer. These input functions do need a longer time to gather the information and process it in the microcontroller. As a result, the polling rate of these inputs are still uncertain. For example, the SPI takes 110 µs which is almost one half of a USB cycle. Therefore, it possible to complete a SPI read within one cycle and the analog read is considerably much less time. An uncertainty still remains because these functions have yet to be tested with USB, but the timing can be theoretically implemented, especially through better use of interrupts.

5.3 Ethical considerations

We do not anticipate that the development of our product would unleash any possibilities of misuse that did not already exist with other computer mice. The largest ethical concern is that we must consider existing products that are similar and be confident that our product is sufficiently differentiated and unique so as to not upset any existing works which would violate number seven in the IEEE Code of Ethics. [5]

5.4 Future work

The upcoming plans for this project includes redesigning the printed circuit board to include a space for the external oscillator. Next, the analog slider and SPI would be implemented on the software side to adhere with USB protocol. Finally, a graphical user

interface could be created for the mouse. The GUI would allow more adjustability for an individual user and allow them to bind buttons to their desire and set a sensitivity threshold.

References

[1] Universal Serial Bus. [Online]. Available: http://www.usb.org/developers/hidpage/HID1_11.pdf. [Accessed 12-March-2018]

[2] Serial Peripheral Interface (SPI). [Online]. Available: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi. [Accessed 14-Feb-2018]

[3] USB Protocols. [Online]. Available: http://www.beyondlogic.org/usbnutshell/usb3.shtml. [Accessed 16-Feb-2018]

[4] How many clicks the best gaming mice can handle. [Online]. Available: https://mybroadband.co.za/news/gaming/208870-how-many-clicks-the-best-gaming-mice-can-handle.html. [Accessed 16-Feb-2018]

[5] 7.8 IEEE Code of Ethics. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed 17-Feb-2018]

[6] This '4D' mouse has a joystick, but that doesn't make it good. [Online]. Available: https://www.pcgamer.com/this-mouse-has-a-joystick-but-that-doesnt-make-it-good/.[Accessed 19-Feb-2017]

[7] PIC16 Microcontroller Datasheet. [Online]. Available: https://www.microchip.com/wwwproducts/en/PIC16F1459https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed 4-Mar-2018]

Appendix A Requirement and Verification Table

Table 2 Requirements and Verifications

Project Area	Requirement	Verification	Verification Completed
USB	Zener diode within mouse's circuit will protect 5.25V over voltage	A. Multimeter probes will be placed across the Zener diode B. Test voltages from 4.75-5.25V C. Multimeter indicates minimum voltage drop above 5.25V	Y
USB	Clock rate is 48MHz	A. Oscilloscope is placed from clock out pin to ground to measure clock frequency	N
USB	Inrush Current never more than an amp?	A. Oscilloscope is placed across VDD to ground with 1 Ω in between	Y
USB	D+ D- 3.3V	A. DMM placed across D+ to ground and D- to ground	Y
USB	A 2 µF will be placed from source to ground to protect transient voltage drop no more than 330mV	A. Oscilloscope will be placed across source and ground to measure voltage B. Voltage will be measured for 1 second after plugging the device into the active power supply C. Oscilloscope should indicate voltage no more than 330mV	Y
Joystick	The slider will be able to move 30mm for comfort	A. Measure 5 other people's thumb mobility to confirm comfortability of 20mm range	Y
Joystick	Rocker switch is able to distinguish rest/forward/backwards	Simple circuit to prove that the rocker positions are unique.	Y
Joystick	The slider will be able to distinguish movement of 1mm	Analog voltage measurements of finished voltage divider by MCU after small measured movements of the slider.	Y
Buttons	Click buttons register when they are pressed	A. Digital multimeter will be placed across the right/left click switches connected to 5V source B. When switches are closed multimeter will read 0V and when open the multimeter will indicate 5V drop.	Y
Buttons	Click buttons will need to be able to register at least 10 distinct clicks per second.	Use of counting circuit coupled with servo instructions for precise button clicks and data confirmation.	Y
Optical	Less than 15ms of latency with our mouse	Precise measurement is possible on finished product through having mouse	N

		view a changing computer screen and logging mouse response.	
Optical	Optical position sensor will have accuracy of 650 DPI	Test with Arduino to ensure that proper total counts received to corresponding to 1”, 2”, and 3” movements over multiple trials.	N
MCU	MCU acknowledges and handle input from Analog and digital pins	Will be confirmed through tests of discrete pushbutton/switch signals as well as analog voltage levels which will be confirmed with a multimeter.	Y
MCU (Attached)	MCU is able to handle SPI communication	Will be confirmed through interfacing known working parts with the MCU.	Y
Computer	The computer will cooperate with common driver protocols	A. The MCU will be programmed as a common driver initially and connected to the computer. B. The computer will be able to recognize the device plugged into its port	Y