MTB Suspension Tuning DAQ
Final Design Review (FDR)

By: Dylan Ruiz

dyruiz@calpoly.edu

Ronan Shaffer

rmshaffe@calpoly.edu

John Ringrose

jringros@calpoly.edu

Theo Philliber

phillibe@calpoly.edu

June 6th, 2022

Sponsor:

Professor Joseph Mello

Mechanical Engineering Department

California Polytechnic State University

San Luis Obispo

Table of Contents

1. Design Updates

Since CDR, there have been three main design changes: utilize the original DAQ housing and microprocessor, redesign the main hub printed circuit board (PCB), and an updated auxiliary sensor housing for the addition of the hall effect sensor.

Utilizing the Original Microprocessor

Since CDR, the intent was to integrate a Renesas Electronics 176 Pin microprocessor in order to have more pin connections that the original STMicroelectronics 64 Pin microprocessor. This large increase in number of pins was not a team decision but was due there being a chip shortage. After incorporating the Renesas Microprocessor into the electrical circuit design on Eagle, we held a design review with Cal Poly professor and mechatronics expert, Charlie Refvem. During the review, we learned that incorporating this microprocessor would lead us to more work than we intended. To our advantage, Charlie had two spare microprocessors that were the exact same as Steven Wahl had used with his design (STM32F205RGT6) and were able to just have enough pins on the microprocessor to connect the additional sensors.

Redesigned Main PCB

With the original microprocessors in the hands of the team, we had to redesign the main PCB for a second time. The Renesas microprocessor had a 25.5mmx25.5mm footprint so when we changed back to the smaller STM microprocessor (10mm x 10mm) the original board form factor was a possibility. By reorganizing components of original board design and adding the new components for the additional sensors, we were able to maintain the same board size as the original device. This allowed us to use the same main DAQ aluminum housing which saved the team from having to perform design work on an updated housing. By using the original microprocessor, we were also able to use the original firmware designed for the STM pinouts.

Updated Auxiliary Sensor Housing for Hall Effect

After the completion of the critical design review, the team still did not have a design for incorporating the hall effect to the front wheel auxiliary sensor. Once the new boards were manufactured, and we were able to start hand soldering the electrical components onto the boards, we could measure the final dimensions of the hall effect in reference to the PCB. Figure 1 below shows a picture of the board for reference.

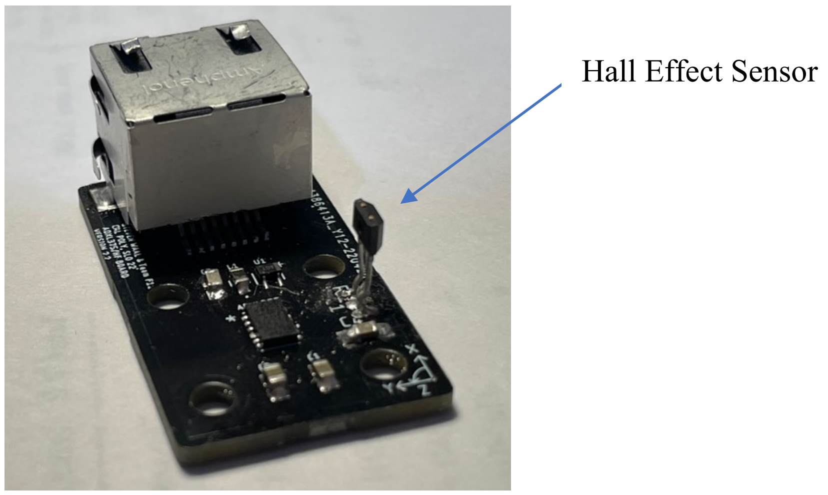


Figure 1. Auxiliary Sensor Board with Hall Effect

The Hall effect sensor is indicated by the red arrow. Since the sensor stands proud of the board by 10mm, the team added a slot in the sensor housing to incorporate the sensor. Figure 2 shows a screenshot of the final CAD model of the auxiliary sensor housing with an added slot for the hall effect sensor.

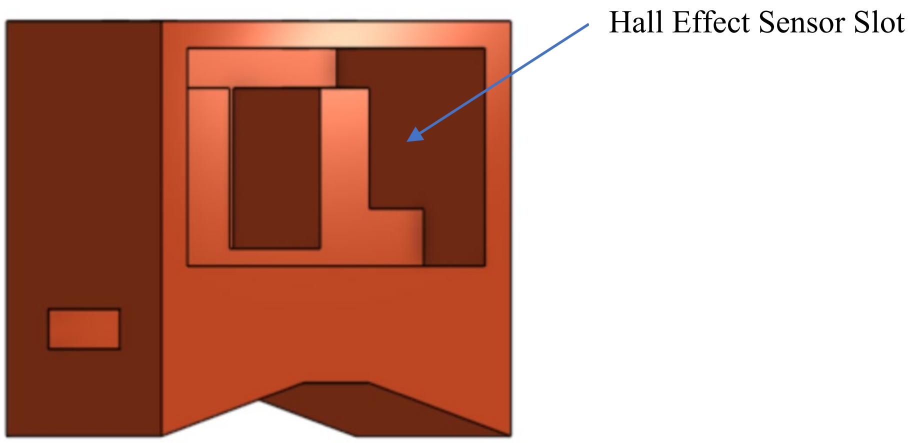


Figure 2. Auxiliary Sensor Housing with Hall Effect Sensor Slot

2. Manufacturing

The following section discusses the manufacturing processes that was executed to complete the verification prototype. The processes led to the development of the new data acquisition system.

2.1 Procurement

Our part procurement process was largely accomplished through purchasing from online vendors. All surface mounted (SMT) electrical components that will be soldered to the PCBs, including resistors, capacitors, batteries, switches, LEDs, and even the aluminum enclosure that houses the main unit were purchased from Digikey. We chose this vendor because of their exceptional variety and quantity of parts available, which is critical during supply shortages. We were able to save money by buying in bulk and consolidating most of our components into one shipment. We selected our components based on our system requirements and how compatible they are with the previous version of the DAQ to minimize redesign efforts and potential errors.

We ordered our PCBs and the stencils that we used to apply solder paste to the boards from JLC PCB because of their cheap costs, quick turnaround time, and reputation when it comes to quality manufacturing.

2.2 Outsourcing & Manufacturing

The outsourcing of our components begins with the manufacturing of the Main DAQ PCB, Accelerometer + Hall Effect PCB, and Accelerometer only PCB. These 2-layer PCBs were manufactured from JLC PCB. In conjunction with this order, we also requested to have stainless steel PCB stencils manufactured. These stencils allowed us to skim solder paste onto the PCBs. A zoomed in picture of the main stencil is shown in Figure 3.

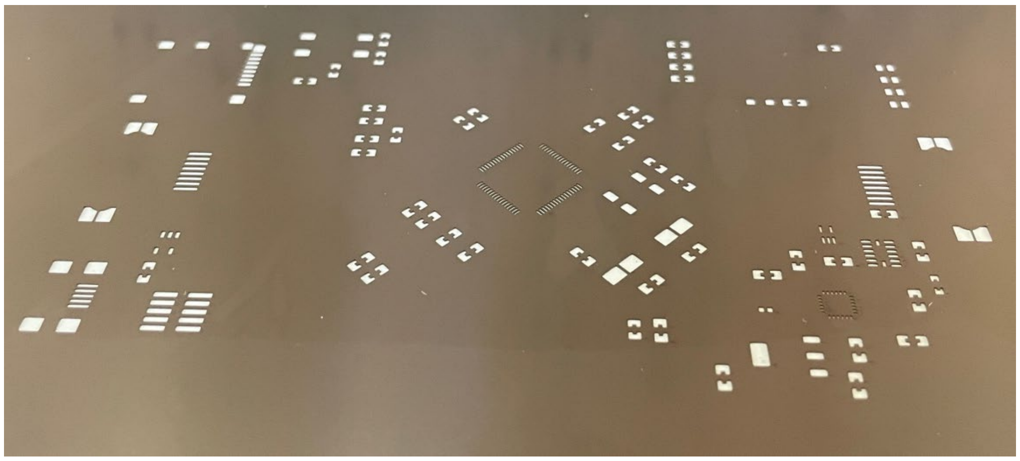


Figure 3. Main DAQ Stencil close up

In order to have these PCBs and Stencils manufactured, we submitted the necessary Gerber files to JLC. Gerber files describe the layout and properties of the PCB as defined by our CAD model. Below are pictures of the four different boards manufactured by JLC.

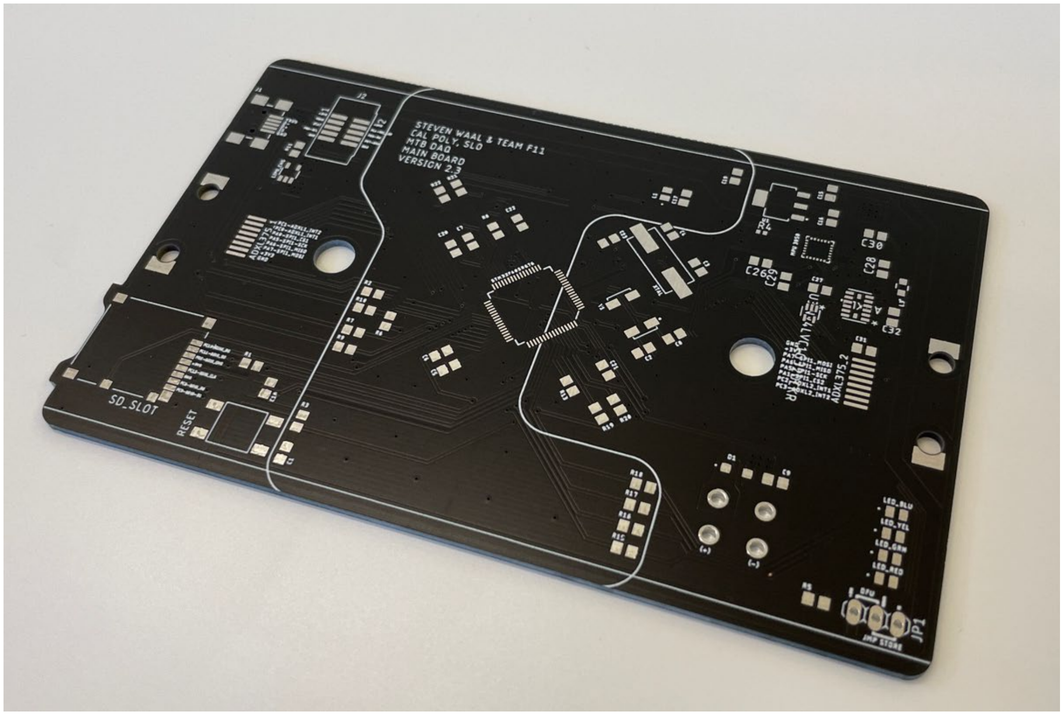


Figure 4. Front of Main DAQ PCB

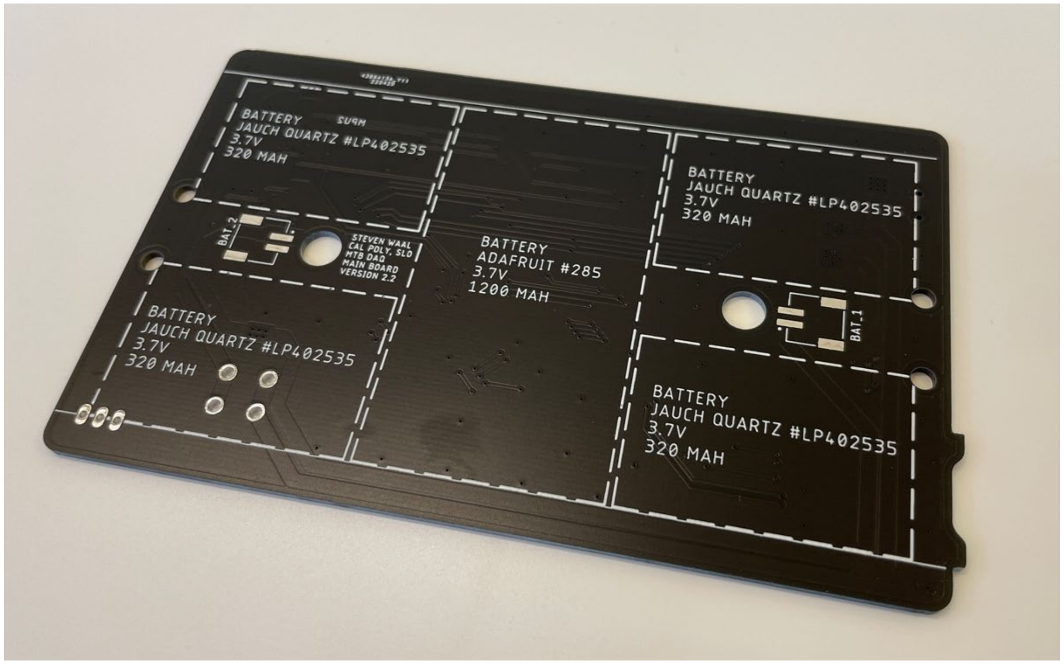


Figure 5. Back of Main DAQ PCB

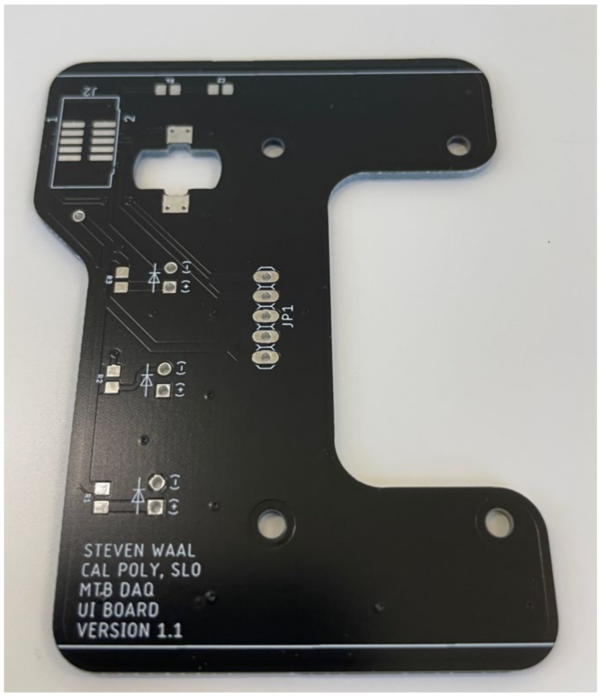
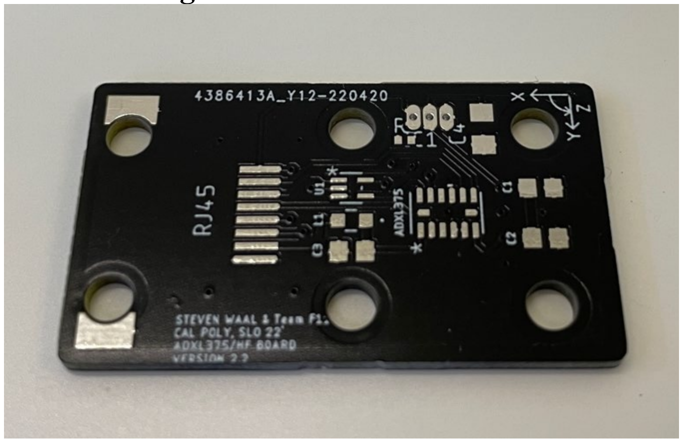


Figure 6. User Interface PCB



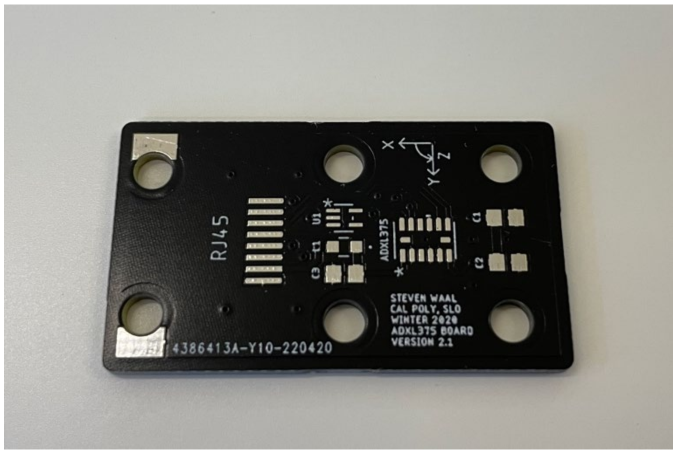


Figure 7. Auxiliary Sensor Boards PCB

In addition to manufacturing the PCBs, we needed to produce more housings for the components. For the main DAQ unit, we purchased an aluminum enclosure (see Figure 8). This enclosure had to be modified in two ways. The faceplate needed to have holes cut out for the LEDs, display, and bolt holes. Second, the side panels will be 3D printed to allow the cables and SD card to be inserted.

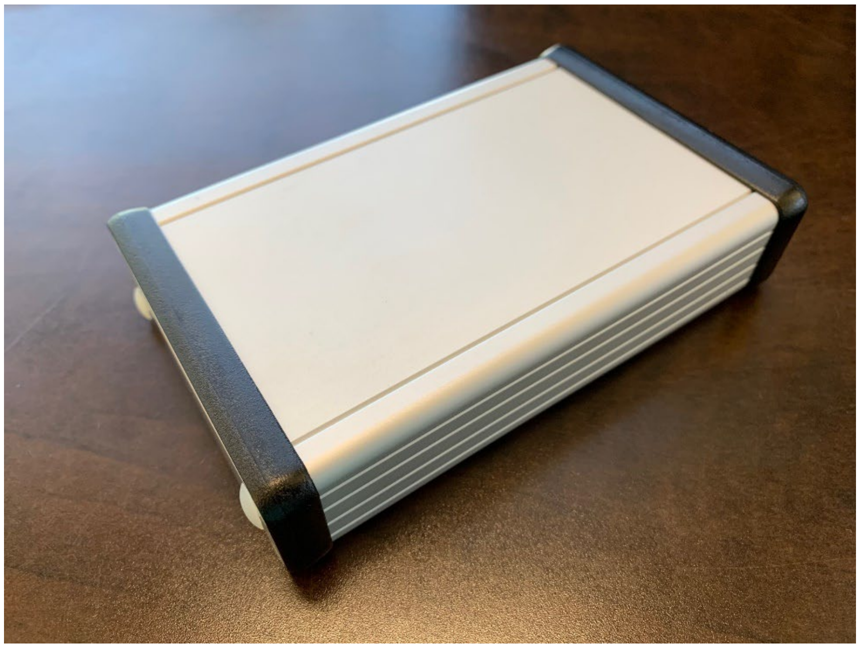


Figure 8. Blank Aluminum Enclosure for Main DAQ Unit

In order to machine the faceplate, we 3D printed a stencil showing all the features we needed to cut into it. This was placed over the faceplate and drilled through for the circular holes. The square display hole was traced through the stencil, and then the corners were drilled out and the sides cut with an angle grinder. Finally, the corners and sharp edges were cleaned up with files. The stencil is shown in Figure 9, followed by the machined faceplate in Figure 10.

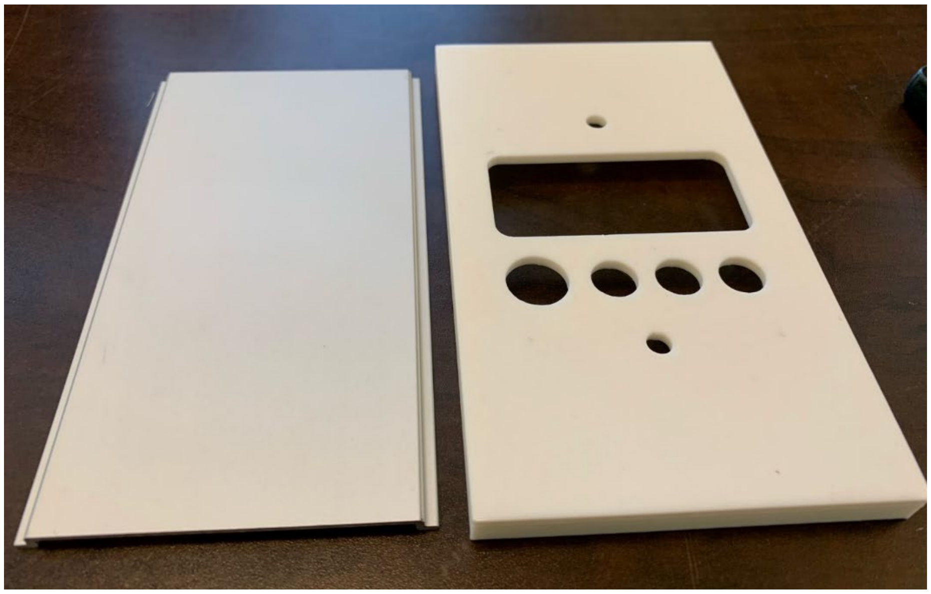


Figure 9. DAQ Housing Faceplate with 3D-Printed Manufacturing Stencil

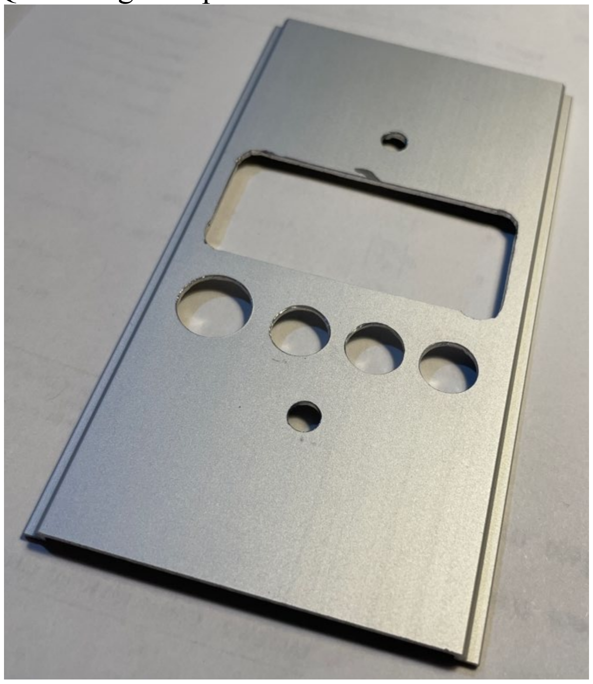


Figure 10. Machined Faceplate

2.3 Assembly

The assembly process for our system includes adding all the electronic components onto the PCB's making them effectively complete. To add all the SMT components to the PCB's we simulated a reflow soldering process using a hot plate in place of a reflow oven.

First, we oriented the stencils on top of the PCBs, making sure the slots were perfectly aligned with the pads on the PCBs. Next, we applied solder paste to the top of the stencil slots, scraping off the excess paste with a putty knife to ensure the paste was applied evenly.

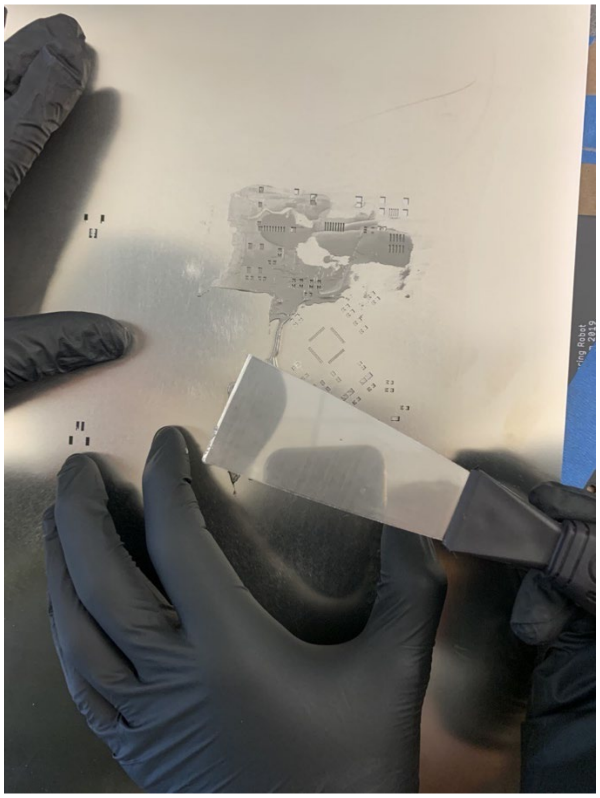


Figure 11. Applying Solder Paste Using the Stencil

After the paste was applied, we removed the stencil carefully, without disturbing the paste applied to the PCB. Then we placed all of the SMT components carefully atop of the solder paste.

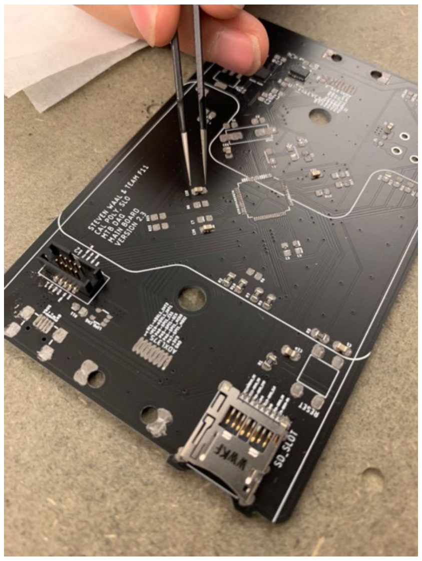


Figure 12. Placing all Components on the Applied Solder Paste

Using the hot plate, we simulated the reflow process by heating up the board and melting the solder within the solder paste to create viable connections between our components and the board.

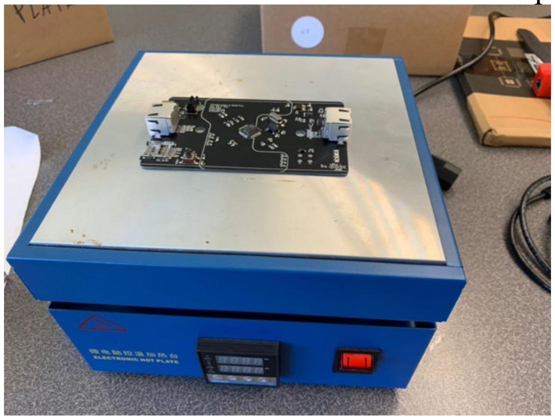


Figure 13. Using Hotplate to Simulate Reflow Oven

During this assembly process, we considered hand soldering all the components onto the boards, however this posed a high risk. Hand soldering SMT components onto a board can be challenging without the right equipment, and one mistake can make the entire board non-functional. Due to these reasons, we decided to pursue other options first and leave this as a last resort. Next, we planned to simulate the reflow process using a heat gun. This method is primarily used to rework certain components on the board and would have been inefficient for our use as we are assembling the entire board which consists of many components. Finally, without the use of a reflow oven, we decided to use a hot plate. This would allow us to control the temperature of the entire board rather than certain segments. Using a hot plate proved to be the most efficient method to assemble our PCBs. Charlie Revfem assisted our team in applying the hot plate method for our PCB assembly.

With the hot plate, we attempted to follow the solder paste manufacturer's provided thermal profile [1]. We used this as an approximate goal since we didn't have the sophisticated equipment required to precisely control the board's temperature.

3. Design Verification

The following section discusses the results of the testing conducted on the MTB DAQ verification prototype. The original specifications designed during the SOW are revisited to determine if the new device passed or failed. Then the limited metric testing is discussed to answer whether the device can provide suspension tune feedback.

3.1 Specifications

Our team prepared multiple specifications that were important for our product's design based on our initial plan. However, as our team decided to prioritize the functionality and metric testing of our system, many of these specifications were not able to be met. The following specifications are referenced within the DVP&R Table in Appendix F.

Main Hub Size

We verified the Main Hub size using calipers. Since our new design of the main hub was not altered from the original design, despite the additions of the new accelerometer and gyroscope sensors, our Main Hub dimensions are within the scope of our specification, being around 5"x3"x1" in size.

Sensor Housings Size

It is important that our sensor housings do not obstruct any of the bikes' mechanical parts or the user's natural pathway when operating the mountain bike. With this in mind, our team decided to keep the sensor's housings limited to 1.5"x1.5"x1.5" if possible. Our final design was a sensor housing that was 1.7"x1.2"x1.06". Although these dimensions do not match our initial projection, we determined that this size was sufficient for avoiding obstruction while also providing enough protection for the sensors. These measurements were also conducted with calipers.

Weight

The weight of our entire system, including the main hub, two external sensors, housings and cables were measured on a scale. The weight of the entire system is 685g. The weight specification requires the whole system to be less than 500g. It is desirable to add as little weight as possible to the mountain bike, as the system is meant to enhance the performance of the mountain bike. A weight similar to that of a full water bottle was desired to justify its use for customers.

Cost

Our design team initially decided to limit the cost to produce this device to be less than $150. This specification was decided on when our goal was to create a customer ready product. However, since our scope has changed, the cost to produce the device is no longer relevant as most of the cost comes from multiple shipping orders.

Battery Life

The battery life was tested during our metric testing. We set a time limit of at least one hour of operation in our specifications for our design. This was set with the intention of it being used for testing sessions that could last at least one hour with no available power sources nearby. Our team was able to use it for a little over two hours while testing, having no problems with data corruption from the device with low battery.

Ingress Protection

Due to the change in scope, ingress protection for our device was not designed or tested. Our main device's main purpose was to use for metric testing and would fail this test if it were performed with our current design.

Foolproof

This test focused solely on the use of the Main DAQ system itself, not understand or post-processing the data. Although our team believes it is important to be able to understand and post-process the data to achieve results within testing, it is required to have previous knowledge with the computing software MATLAB.

Maximum Recording Storage

This test solely relies on the SD card used. Since our design implements an external storage unit (SD Card), the amount of data it can store is varied by the resources of the user. Our design does not have a fixed storage amount and easily exceeds or design specification of storing 8gb or more. Our team used a 16gb SD card for our testing and had no problems with the memory being full.

Mounting Universality

This specification ensures the possibility of using the DAQ device on all types of full suspension MTBs. Upon testing this specification, our team encountered an unexpected problem that resulted in our design to fail this specification. While all MTB's have a water bottle bosses that are standard in spacing, the position of the water bottle bosses can vary. This resulted in our test to fail, as we were unable to insert the ethernet cables into the Main Hub after attaching it to the MTB due to the obstruction of the frame.

Aesthetics

This specification was also not able to be testing and lost its importance once we switched from the primary goal of creating a customer ready product. Since the design of the new device was primarily to test for metrics when optimizing suspension tuning settings, the aesthetics of the design was no longer important and not within the scope of our project.

Suspension Tuning Recommendation

This specification is out of our project's scope as we are primarily using the device to validate the metrics used to create a suspension tuning recommendation. Our device will no longer give a tuning recommendation itself and is not designed to perform that desired action. It will, however, be used to collect data that can be used to interpret what settings to change based on our metric testing.

3.2 Tests and Results

Metric Testing

Metric testing is the main test that evaluates the performance of the MTB DAQ. Since the overall goal of the project is to quantitively provide suspension tune recommendations, we had to gather as much data on different suspension settings. There are three suspension tune recommendations the DAQ can provide feedback on; stiffness, compression damping, and rebound damping. As a team, we concluded a proven suspension tune recommendation must be backed by large amounts of data. This data was gathered while a rider rode the same trail on many different suspension tune setups. This includes modifying each one of the settings individually as well as at the same time. The validation of a tune would be based on speed and qualitative feedback. Speed meaning how fast did the rider get from point A to point B. Qualitative feedback from the rider would be their opinion on how it felt and what it was doing on the trail. With these two validation points, the metric tested during these trails would hopefully backup the faster speed and better rider feedback with some sort of sensor rates.

The first tested metric was a fork handlebar transmissibility test. This test had one accelerometer unit positioned on the handlebar and one accelerometer unit positioned on the fork lower. During the first day of testing, we looked at different fork stiffnesses only. The fork used during this testing was a Fox 38 with air pressures ranging from 80-100 psi with 5 psi intervals. This range of pressures was decided on since the recommended pressure by fox was 90 psi for the rider. We assumed that with this starting point recommendation, the fork stiffness would not be changed more than ± 10 psi. The testing was performed on the first upper segment of Shooters on the Cuesta Grade. This trail is roughly 1 minute long and features high speed flat corners, chattering sections, and one small jump. The team collected 2 trials per fork stiffness, totaling up to 10 runs. For post processing the data we first took the rms value of the fork and handlebar at a time step of 1ms. This plot ended up showing no clear distinction run-to-run. We then increased this time step until we could identify spikes or reductions in amplitudes. The final time step we arrived at was ~2 sec. The 80 psi and 100 psi data are shown below in Figure 14.

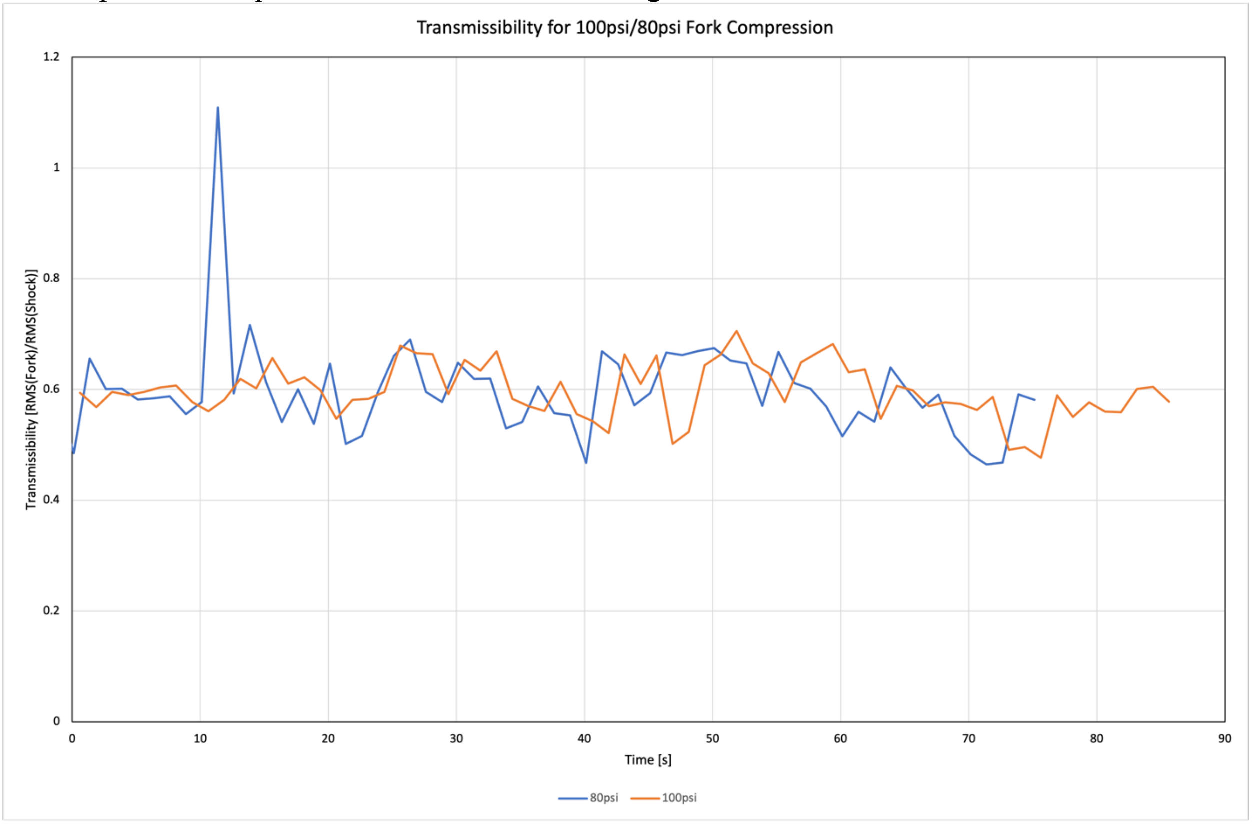


Figure 14. Fork-Handlebar Transmissibility Plot of 80psi and 100psi Fork Stiffness

There were a few things to take away from this testing. First, we needed to identify when the trail begins on the plots. This could be as simple as introducing a small feature at the beginning of trail that should show a spike on the plot. Secondly, it seems that the fork-handlebar transmissibility metric may not be the valid way to provide a suspension tune as we did not see dramatic changes in the data collected.

The next session took place on May 31st, 2022 and looked at fork handlebar transmissibility again. Max rode the bike during this set of runs. The suspension component being altered was rebound.

We did 6 total runs, this time identifying the start of the trail by riding over a 2”x4” block to signify a peak in the data. From there we could chop the data collection at the actual start and end of the trail. Here is a plot of the transmissibility for the fastest rebound setting, lowest rebound setting, and Fox Factory rider recommendation rebound (for Max’s rider weight).

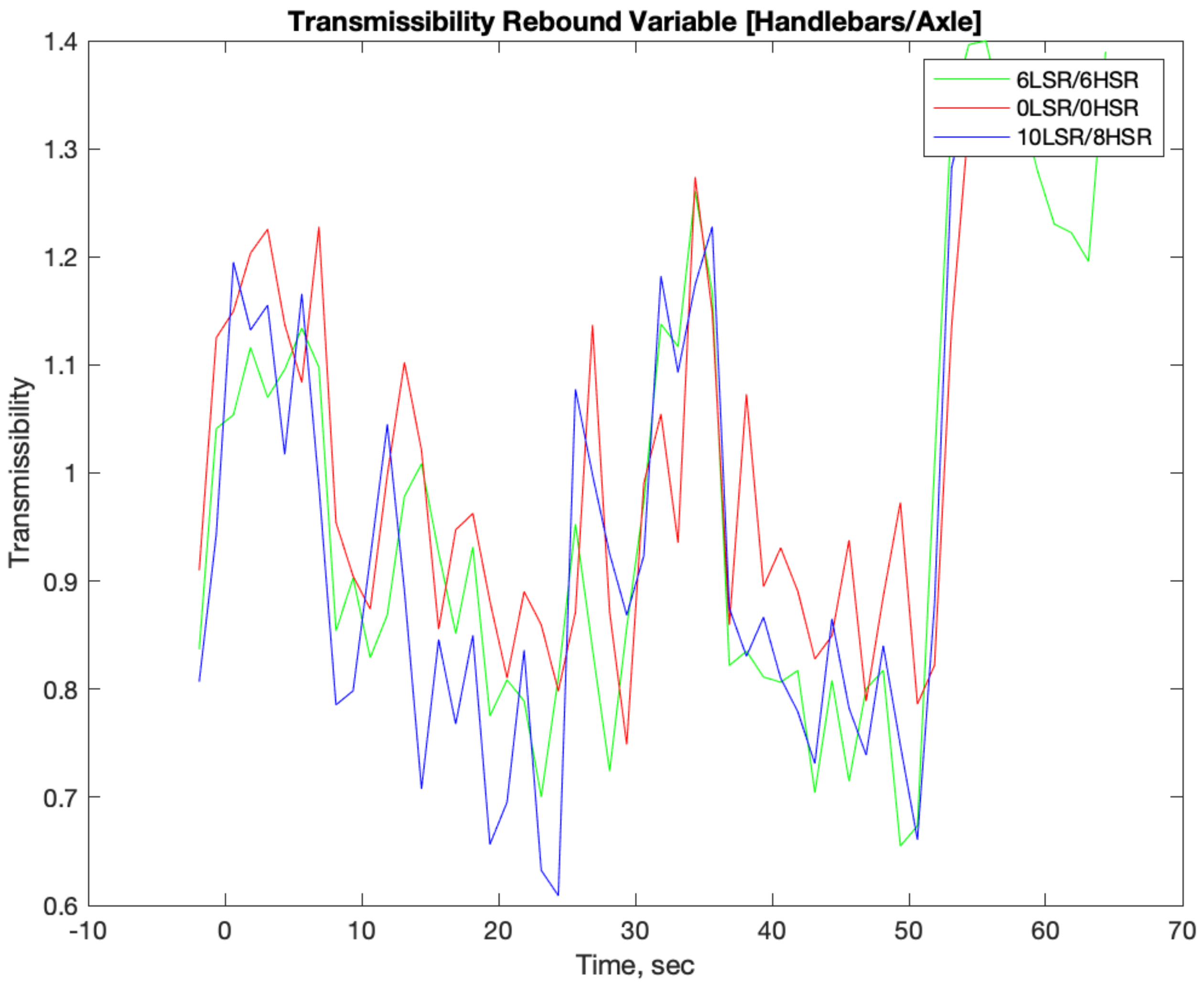


Figure 15. Fork-Handlebar Transmissibility Plot of Slow, Middle, and Fast Rebounds

Figure 15 looks at the RMS values of the fork and handlebar accelerations, but the data was post processed with a time step of ~1.2 seconds. By looking at the graph, we can see that there is a large difference in transmissibility between different setups. Less intuitive, but the setting with the least number of rebound clicks (slowest rebound) showed the largest transmissibly. We took the post processing one step further and took the rms of these data sets with a time step of the entire length of the trail. Figure 16 shows the results.

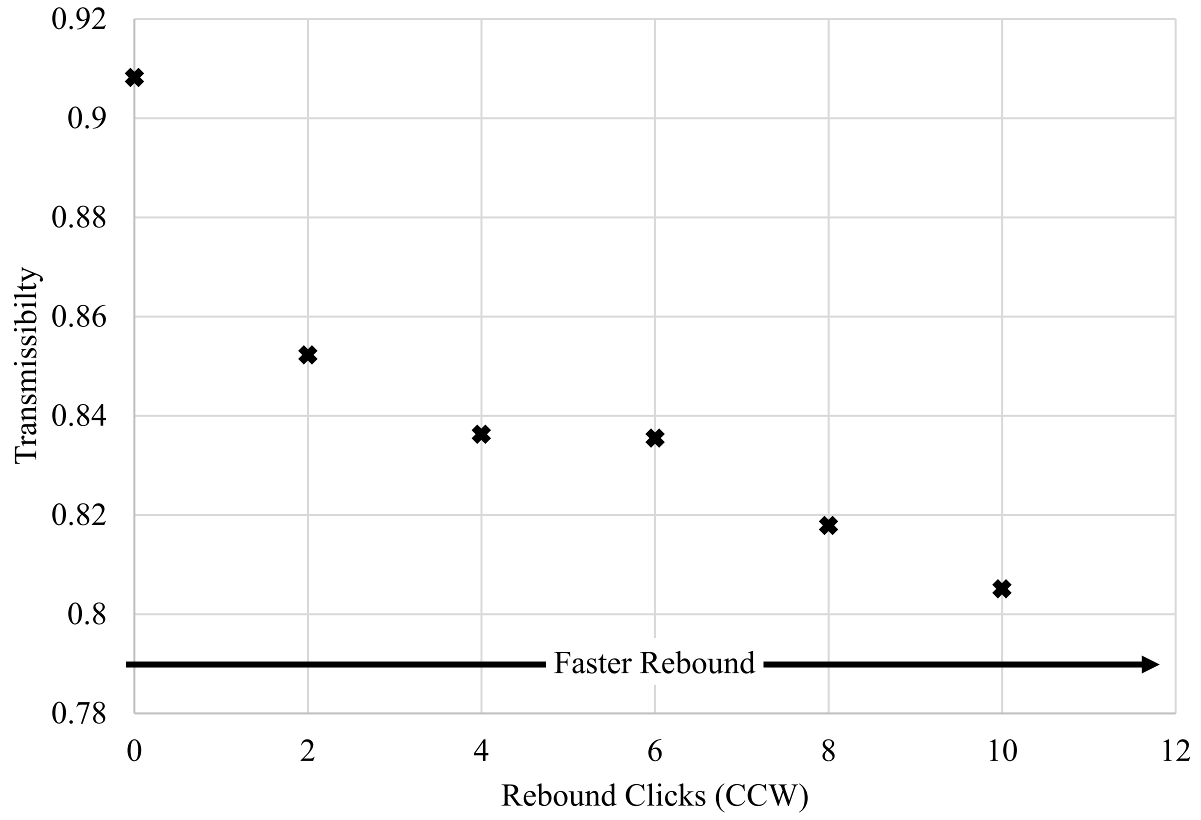


Figure 16. Transmissibility of Entire Run with Varying Rebound

From Figure 16, we can see that as the rebound setting in the front fork increases this transmissibility metric decreases. The results of this testing are contradictory to the qualitative feedback we recorded from Max. The optimal rebound for Max's fork was 6 clicks and two clicks in either direction felt very similar, but as the setting went in either direction the suspension started to feel unstable. In the faster rebounds, the fork felt stiffer and had less traction, whereas the slower rebounds felt too soft and still did not have traction. Our conclusion on the trend of decreasing transmissibility with faster rebound settings is that with faster rebounds the accelerometer on the fork lower was able to accelerate more. The equation below shows the transmissibility function and the variables changing due to faster rebound.

$$\downarrow Transmissibilty = \frac{RMS(Handlebar\ Acceleration)}{RMS(Fork\ Lower\ Acceleration)} \uparrow$$

Accelerometer Calibration Verification

Due to time constraints and hardware issues, this test did not end up being performed. The planned procedures are laid out below for future groups who wish to perform such a test.

MEMS accelerometers have a certain amount of inaccuracy inherent to their design. That is, the accelerations they read may not accurately reflect the true accelerations they undergo. To get an idea of how accurate our ADXL375 accelerometers are, we tested them with known accelerations to compare the data to.

Providing a constant known acceleration to the sensors is a somewhat difficult task. Instead, we used the known acceleration of gravity to get an idea of how the sensors perform. By positioning the sensors at different known angles, we can compare the output of the sensors to the theoretical value. To position the sensors, we created a fixture (Figure 17 below) with slots cut at different angles. For each 15-degree increment, we averaged 30 seconds of data from each axis, and compared them to the expected values. Finally, we took the average and standard deviation of the error at each angle and calculated them to 99.7% uncertainty (3-sigma confidence interval).

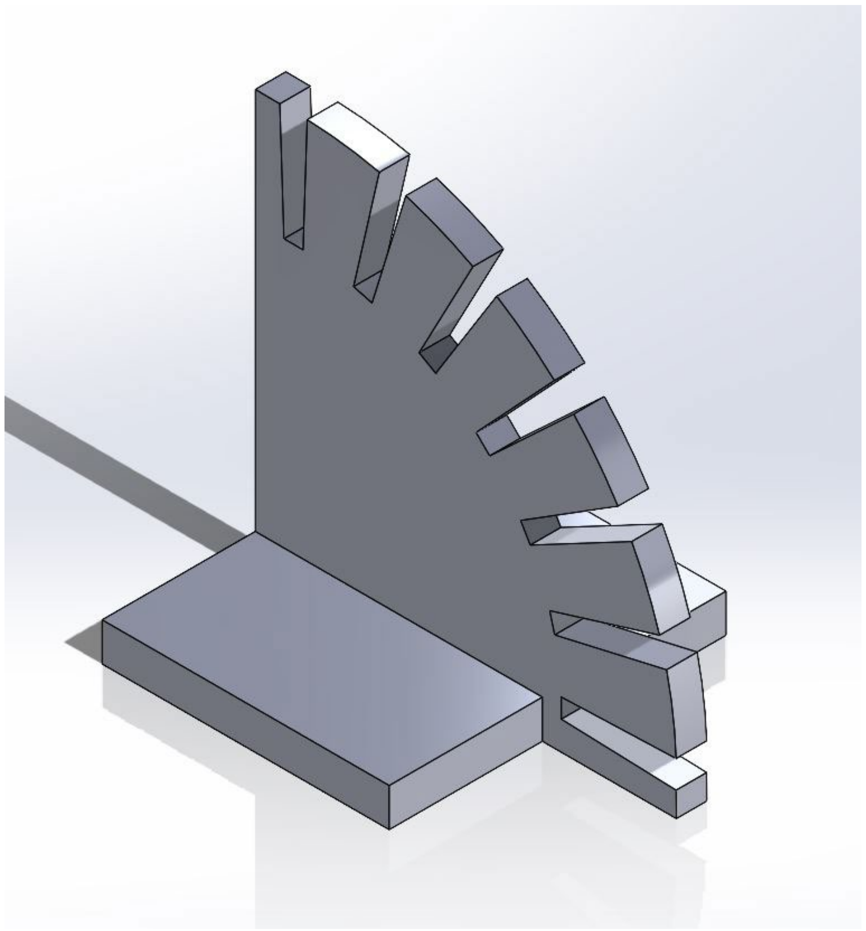


Figure 17. Calibration verification fixture. Accelerometer board would be inserted into the angular slots, cut at 15° intervals from 0° to 90°.

From this calculation, we would get an observed measurement error. This would be compared to the expected measurement error, based on the accelerometer manufacturer's posted uncertainties and the uncertainty of the test rig.

3.3 Challenges and Recommended Testing

Debugging and Troubleshooting

Identifying and solving problems with the original device proved to be a greater challenge for us than we anticipated. Some issues, such as broken soldered connections for buttons, caused bugs that were intermittent and sporadic. This unpredictable behavior made it harder to pinpoint the problem. Another issue that took our team some time to identify and fix was an inadequate power supply, which was probably a result of depleted batteries connected in parallel so that when they were unequally charged after some use they would drain into each other, so they would discharge faster than expected. Buying new batteries seemed to fix this issue, as well as disconnecting the power circuit when not in use.

Procuring Materials and Components

Cal Poly has limited resources for PCB manufacturing and assembly, and electronic components are in short supply globally. This made it difficult to buy replacement components and locate tools and hardware on campus that we needed to use during assembly. There was also a learning curve for our team, as no one had prior experience working with PCBs. Because of this, it took our team longer than anticipated to put together our verification prototype. Charlie Refvem was a big resource for our team and provided us with electronic components and tools we needed to assemble the PCBs that we otherwise would not have had access to.

Metric Testing Challenges

Limited time and logistics, and isolating suspension effects were some of the challenges we faced when testing the original device once it was up and running. We wanted to get realistic data with our system, which meant riding an actual mountain bike trail segment repeatedly, as opposed to some artificially constructed course closer to campus or home. To conduct this testing, we needed to drive up to Cuesta Ridge, set up the bike and DAQ, and ride down a section of trail and pedal back up for each run of data collection. This ended up taking 3+ hours for 10 runs just varying a single suspension parameter, so it's very time intensive. Once we have collected some data and identified some trends, it is not always clear if the cause of the trend is due to the change in suspension settings or something else, like the rider getting more comfortable on the trail and taking a better line or becoming more tired and riding slower. There are many factors that affect the end result. Additionally, there are many different methods to process the data, and it is not always obvious how to analyze it to expose differences between runs and make sense of the behavior.

Recommendations for Testing

First and foremost, we recommend an abundance of testing to collect data on the effects of each individual suspension parameter on the bike's behavior. Collecting a wealth of data with a variety of settings, riders, bikes, and trails will help clarify trends in the data and isolate the effects from specific inputs. It is helpful to have identifiable features that leave distinctive data points as it makes it easier to compare results between runs. This can be difficult for certain trails, so choosing a simple trail that can be ridden in around one minute would be optimal. This trail should have different known significant features that can easily be identifiable when looking at the data. Also, when starting the trial runs, implement something to roll over to indicate when you start the actual trial run. This allows you to identify the start of each run through data processing rather than quickly starting the trial right after you press the button.

4. Discussion and Recommendations

From this design project, we learned that root cause analysis and troubleshooting electronics hardware is difficult and time consuming. When there are dozens of components in a few square inches of circuit board and hundreds of lines of code, there are a multitude of potential underlying causes that might be contributing to the buggy behavior. We also learned that designing and assembling the PCBs takes painstaking time and effort as well as technical skills that we had not practiced until we assembled our final prototype. This is a crucial step in the manufacturing process because one faulty soldered connection can disrupt the functionality of the entire system.

To continue this design, we would fix the board design to add any missing traces and fix any soldering mistakes with the existing components so that every sensor is fully functional. We would also conduct more testing and collect as much data as possible to have more room for developing metrics and identifying trends that result from suspension changes.

If we were to continue refining this design to meet the needs of the customer, we would make the system more user friendly to set up and operate. Specifically, we would redesign the UI to include a battery charge level indicator, move the record button to the handlebars for easier access, and recess the power switch below the surface of the housing to prevent inadvertent switch flips. We would also use batteries that are more standard and safer, such as 18650 cylinders, to avoid connecting batteries of varying capacities in parallel. Furthermore, we would improve the housings to eliminate openings to the interior that could allow dust and water to enter the same space as the electronics and incorporate another mounting option for the central unit, so the rider does not have to choose between bringing a water bottle or the DAQ on a ride. Finally, we would include Bluetooth modules to transmit and receive data wirelessly, so the rider does not have to fidget with messy ethernet cables and zip ties.

If we were to build this prototype again, we would outsource the assembly of the PCBs to streamline the manufacturing process and minimize any soldering mistakes that could result from tediously soldering by hand every component onto the board. We would also like to test the final circuit with a bed of nails to detect voltages and currents at many different grid points to ensure power and signals are being sent where they should be sent.

To produce a high volume of devices at a reasonable price and in a time efficient manner, we would outsource the soldering/assembly process to a facility who specializes in PCB production and has the means to mass produce high quality, reliable boards. We would also buy components in bulk from suppliers to lower the price per component.

Our team recommends using this design to continue testing potential metrics that will lead to the development of an algorithm that can be used to suggest optimal suspension tuning settings. Using the User Manual in Appendix E, anybody should be able to use this design efficiently as our team covers known bugs and issues with the device.

5. Conclusion

In retrospect, this project was not the best fit for a Cal Poly Senior Design Project, given the structure of the class and timeline of deliverables. The emphasis on problem definition and ideation in the first quarter is better suited for teams starting a design from scratch, and we feel that we could have achieved more of our goals if we had been able to test, troubleshoot, and refine the existing design earlier in the year [2].

In the end, we were able to fully debug the original device and collect good data with a mountain bike on the trail. We designed and built a partially functional data acquisition device, capable of collecting data with our two auxiliary accelerometers. We also redesigned and manufactured sensor housings that are more universally compatible with different bikes. Finally, we analyzed data that we collected to start identifying trends and experiment with post-processing methods.

We did not achieve a fully functional device capable of collecting data from all three accelerometers, the gyroscope, and the hall effect speed sensor. We also did not fully develop and refine enough metrics to robustly process data and draw meaningful conclusions. We attribute these shortcomings to our mismanagement of time at the beginning of the year, when we should have been prioritizing testing and hardware troubleshooting. However, we feel that another reason we failed to achieve these end goals is the ambitious scope of this multi-faceted project. Considering the experience of our team coming into this project and the resources and guidance available to us throughout the year, the mismatch between this project's needs and the course's structure and requirements, and the miscommunication about critical flaws in the original device, we feel we were not adequately prepared to successfully complete our goals that we set in the beginning of the year.

If we were to do this project over again, we would refine our goals to focus on getting the hardware and firmware right to build a fully functional data acquisition system from scratch. We would only move on to metric testing, development, and data processing afterwards, or separate this section into its own project entirely.

References

[1] Solder Paste Data Sheet - https://www.chipquik.com/datasheets/SMD291AX.pdf

[2] S. R. Waal, “A Quantitative Approach for Tuning a Mountain Bike Suspension”, M.S. thesis, California Polytechnic Univ., San Luis Obispo, CA, 2020.

Appendix

A - Main DAQ Python Code

```
# MAIN
#========================================================================
# @file main.py
#========================================================================
# ABOUT:
# This code runs a simple data acquisition system on the MTB-DAQ v2.2
# board running Micropython PYBv1.1 version 1.12 firmware. The code
# records data from three ADXL375BCCZ accelerometers at 1600Hz over SPI and
# one MPU-3050 gyroscope and writes the data to binary file on a Micro SD card.
#========================================================================
# WRITTEN BY: Steven Waal / Updated by Team F11
# DATE: 04.12.2022
#========================================================================
# NOTES:
# 05.28.2019 - File created (SRW)
# 06.07.2019 - Updated to use SPI bus no. 2 instead of no. 1 (not sure why
#              but for some reason SPI bus no. 1 is not working properly)
# 01.29.2020 - Implemented RTOS system
# 02.12.2020 - Implemented display and SD card tasks.
# 02.12.2020 - Having trouble with timing. I can't get faster than 5msec
#              between measurements. I tried moving the record state for
#              each task to the very top (to minimize if statements it
#              takes to get there) but that didn't do too much.
# 02.19.2020 - Implemented code to support ISR
# 03.10.2020 - Changed over to PyBoard. Update pinouts and code to check
#              presence of micro SD card.
# 05.23.2020 - Adapted code to work with MTB DAQ v2.2 main board.
# 06.11.2020 - Final comments added, code cleaned up
# 04.12.2022 - Added third accelerometer and gyroscope
#========================================================================
# COPYRIGHT:
# @copyright This program is copyrighted by Steven Waal and released under
# the GNU Public License, version 3.0.
#========================================================================




#========================================================================
# MISC. NOTES
#========================================================================
# - There is a completely blank file called "SKIPSD" that is loaded onto
#   the board. This prompts the microcontroller to boot from the
#   the internal flash instead of the SD card when an SD card is inserted
#   before powering on.
```

1

```
#=========================================================================
# IMPORT MODULES
#=========================================================================
# Import modules for use in this file. Note that there are certain modules
# that automatically come with downloading micropython onto the board. To
# see a list of these modules, type help("modules") in the REPL. This will
# return a list of the available modules. Custom modules can be added by
# saving them as separate *.py files, uploading them to the board, and
# referring to them here.

# HARDWARE MODULES
from ADXL375_driver import ADXL375
from ht16k33_seg import Seg14x4
from MPU3050_Driver import MPU3050

# # GENERAL MICROPYTHON MODULES
import micropython, pyb, utime, gc, machine, os

# MISC.
from helperFunctions import clear_accel_buf
from helperFunctions import decode_data
from helperFunctions import twos_comp
from helperFunctions import get_ODR
```

```
#=========================================================================
# ALLOCATE MEMORY FOR ERROR REPORTS
#=========================================================================
# According to Micropython docs, "If an error occurs in an ISR,
# MicroPython is unable to produce an error report unless a special buffer
# is created for the purpose. Debugging is simplified if the following
# code is included in any program using interrupts."

micropython.alloc_emergency_exception_buf(100)
```

```
#===========================================================================
# DEFINE ACCELEROMETER PARAMETERS
#===========================================================================
# Determines how many data points are stored in the FIFO buffer before an
# an interrupt is generated. Maximum is 32.
FIFO_BUFF_COUNT     = micropython.const(20)
```

```
#===========================================================================
# DEFINE BUFFERS
#===========================================================================
CMD_RD       = bytearray((0b11110010, 0, 0, 0, 0, 0, 0)) # ADXL375_1.
# Command to read multiple bytes starting with X data
buf1_7       = bytearray(7) # Buffer of ADXL375_1. Make buffer large enough to
# read data from X, Y, and Z
buf2_7       = bytearray(7) # Buffer of ADXL375_2. Make buffer large enough to
# read data from X, Y, and Z
buf3_7       = bytearray(7) # Buffer of ADXL375_3. Make buffer large enough to
# read data from X, Y, and Z
buf4_7       = bytearray(7) # Buffer of MPU-3050.  Make buffer large enough to
# read data from X, Y, and Z
buf_HE       = bytearray((0b00000000)) # Buffer of hall effect sensor. Make
# buffer 1 byte to store how many passes of the magnet have occurred.
```

```
#===========================================================================
# CREATE PIN OBJECTS
#===========================================================================
# Create pin objects. Pins are labeled according to the MTB DAQ v2.2 main
# board schematic.
# SD chip detect pin
CD                      = pyb.Pin(pyb.Pin.cpu.A8, pyb.Pin.IN, pyb.Pin.PULL_UP)
# Enables the chip detect pin with internal pull up resistor

# ADXL375 1 chip select pin
SPI1_CS1                = pyb.Pin(pyb.Pin.cpu.A0, pyb.Pin.OUT)
# ADXL375 2 chip select pin
SPI1_CS2                = pyb.Pin(pyb.Pin.cpu.A1, pyb.Pin.OUT)
# ADXL375 3 chip select pin
SPI1_CS3                = pyb.Pin(pyb.Pin.cpu.B9, pyb.Pin.OUT)




# Record Button
REC_BTN                 = pyb.Pin(pyb.Pin.cpu.B3, pyb.Pin.IN)
# Record LED
REC_LED                 = pyb.Pin(pyb.Pin.cpu.C4, pyb.Pin.OUT)
# Interrupt Pin
ADXL1_INT1              = pyb.Pin(pyb.Pin.cpu.C0, pyb.Pin.IN, pyb.Pin.PULL_DOWN)
# Set internal pull-up resistor so we always know state of pin when not in use

# Callback function to store hall effect reading. Upon the magnet's presence,
# the interrupt will be triggered and run this function.

def HALL_EFF_CB(IRQ_src):
    global buf_HE
    buf_HE[0] = 1

# Hall Effect Interrupt Pin
HALL_EFF_INT            = pyb.ExtInt(pyb.Pin.cpu.C1, pyb.ExtInt.IRQ_RISING, pyb.P


# VLogic pin for
VLogic                  = pyb.Pin(pyb.Pin.cpu.C13, pyb.Pin.OUT)
VLogic.high()
#Create and initialize Gyro I2C object
Gyro_I2C                  = pyb.I2C(2, pyb.I2C.MASTER)
Gyro_I2C.init(baudrate = 40000)
utime.sleep_ms(100)



#==========================================================================
# CREATE DISPLAY OBJECT
#==========================================================================
#Create display object. Default to off.
Display = Seg14x4(machine.I2C(1))
Display.text('    ') # Clears display
Display.show()
```

```
#==========================================================================
# INITIAL CHECK IF SD CARD IS PRESENT
#==========================================================================
# If SD card is not present, program waits until user inserts one and flashes
# "SD" on the display.
# Once the SD card is present, the proper files and directories are created
# if they have been erased.

# If SD is not detected, flash "SD" on the display.
if CD.value():
    Display.text('  SD')
    Display.blink_rate(2)
    Display.show()

# Wait for user to insert SD card if not already done.
while CD.value():
    # Make display LEDs blink. This will hold up the program and ensure that
    # the SD card is mounted before continuing.
    utime.sleep_ms(100)

# Clear display once SD card has been inserted.
utime.sleep(3)
Display.text('    ') # Clears display
Display.blink_rate(0)
Display.show()

try:
    # Once user has inserted SD card, mount it.
    os.mount(pyb.SDCard(), '/sd')
except:
    Display.text('OFF ')
    Display.blink_rate(2)
    Display.show()

# Remake 'log' directory if it has been erased. If not, ignore error
try:
    os.mkdir('/sd/log')
except:
    pass

# Remake 'count' directory if it has been erased. If not, ignore error
try:
    os.mkdir('/sd/count')
except:
    pass

# Check if 'count.txt' was erased. If it was, remake it
try:
    file = open('/sd/count/count.txt', 'x') # The 'x' argument indicates that
    # if the file already exists, through an error

    file.write('0\n')
    file.close()
except:
    pass

# Change directory to SD card to prepare for writing files to it
os.chdir('/sd/log')




#===========================================================================
# CREATE SPI OBJECT
#===========================================================================
# Create SPI object in order to use the spi protocol

# Set baudrate to maximum of 5 MHz
# Set polarity and phase as specified by sensor datasheets.
spi_1 = pyb.SPI(1, pyb.SPI.MASTER, baudrate=5000000, polarity=1, phase=1,
                bits=8, firstbit=pyb.SPI.MSB)




#===========================================================================
# CREATE ADXL375 ACCELEROMETER OBJECTS AND CONFIGURE SETTINGS
#===========================================================================
# ***************************
# *****ADXL375 1 OBJECT*****
# ***************************
```

```
ADXL375_1 = ADXL375(spi_1, SPI1_CS1)
ADXL375_1.standby() # puts accelerometer in standby mode. this is necessary to
# configure it
# DATA RATE AND POWER MODE CONTROL
ADXL375_1.odr(ADXL375_1.ODR_1600HZ) # sets data rate to 1600 HZ
ADXL375_1.normal_power_mode()
# DATA FORMAT
ADXL375_1.spi_4_wire()
ADXL375_1.right_justify()
# SETUP INTERRUPTS
ADXL375_1.int_disable(ADXL375_1.Watermark_enable) # Make sure interrupts are
# disabled before configuring as per datasheet
ADXL375_1.FIFO_Mode_FIFO() # Configures the FIFO buffer to operate in FIFO mode
ADXL375_1.trigger_int1() # Configures the interrupt to pin INT1
ADXL375_1.interrupt_active_high() # Configures the interrupt to be active high
ADXL375_1.set_samples(FIFO_BUFF_COUNT) # Sets the number of samples before the
# watermark bit is set

# ***************************
# *****ADXL375 2 OBJECT*****'
# ***************************
ADXL375_2 = ADXL375(spi_1, SPI1_CS2)
ADXL375_2.standby() # Puts accelerometer in standby mode. this is necessary to
# configure it
# DATA RATE AND POWER MODE CONTROL
ADXL375_2.odr(ADXL375_2.ODR_1600HZ) # Sets data rate to 1600 HZ
ADXL375_2.normal_power_mode()
# DATA FORMAT
ADXL375_2.spi_4_wire()
ADXL375_2.right_justify()


# ***************************
# *****ADXL375 3 OBJECT*****'
# ***************************
ADXL375_3 = ADXL375(spi_1, SPI1_CS3)
ADXL375_3.standby() # Puts accelerometer in standby mode. this is necessary to
# configure it
# DATA RATE AND POWER MODE CONTROL
ADXL375_3.odr(ADXL375_3.ODR_1600HZ) # Sets data rate to 1600 HZ
ADXL375_3.normal_power_mode()
# DATA FORMAT
ADXL375_3.spi_4_wire()
ADXL375_3.right_justify()

# ***************************
# *****Gyroscope OBJECT*****'
# ***************************
Gyro = MPU3050(Gyro_I2C)
```

```
#===========================================================================
# GET FILE COUNT
#===========================================================================
print('GET FILE COUNT')
print()

countFile = open('/sd/count/count.txt', 'r')
last_line = int(countFile.readlines()[-1])
file_count = last_line
countFile.close()
```

```
#===========================================================================
# INITIALIZE RECORD LED AND CLEAR DISPLAY
#===========================================================================
print('INITIALIZE RECORD LED AND CLEAR DISPLAY')
print()

# Turn off record LED
REC_LED.value(1)
# Clear display
Display.text('    ') # Clears current text/numbers on display
Display.number(file_count) # Prints the desired number
Display.blink_rate(0)
Display.show() # Updates the display
```

```
while True:

    #===========================================================================
    # WAIT FOR INPUT FROM RECORD SWITCH
    #===========================================================================
print('WAITING FOR INPUT...')
print()

while REC_BTN.value() == True: # Wait for user to press button
    pass
while REC_BTN.value() == False: # Wait for user to let go of button
    pass
```

```
#===========================================================================
# BEGIN RECORDING
#===========================================================================
print('RECORDING')
print()

# Turn on record LED
REC_LED.value(0)
# Increment file count and save to count file
file_count += 1 # Increment file count
countFile = open('/sd/count/count.txt', 'a')
countFile.write(str(file_count) + "\n")
countFile.close()

# Update the display
Display.text('    ') # Clears current text/numbers on display
Display.number(file_count) # Prints the desired number
Display.blink_rate(0)
Display.show() # Updates the display

# Clear the accelerometer buffer. Make sure it is in standby mode first
ADXL375_1.standby()
clear_accel_buf(ADXL375_1)
ADXL375_2.standby()
ADXL375_3.standby()

# Create the data file.
file = open('data' + str(file_count) +'.bin', 'wb')
```

```
#===========================================================================
# RECORDING!
#==========================================================================
# Enable interrupts and start measuring!
ADXL375_1.int_enable(ADXL375_1.Watermark_enable)
ADXL375_1.measure()
ADXL375_2.measure()
ADXL375_3.measure()
HALL_EFF_INT            = pyb.ExtInt(pyb.Pin.cpu.C1, pyb.ExtInt.IRQ_RISING,
                                     pyb.Pin.PULL_NONE, callback=HALL_EFF_CB)

while REC_BTN.value() == True: # Wait for user to press button
    while ADXL1_INT1.value() == False: # If the INT1 pin is low, wait for
    # accelerometer to collect more data
        pass
    for i in range(FIFO_BUFF_COUNT): # Store values onto SD card in
        # 'log.bin' file

        #Read Accelerometers and gyroscope
        SPI1_CS1.low(); spi_1.send_recv(CMD_RD, buf1_7); SPI1_CS1.high()
        # Read ADXL375_1
        SPI1_CS2.low(); spi_1.send_recv(CMD_RD, buf2_7); SPI1_CS2.high()
        # Read ADXL375_2
        SPI1_CS3.low(); spi_1.send_recv(CMD_RD, buf3_7); SPI1_CS3.high()
        # Read ADXL375_3
        Gyro.i2c.mem_read(buf4_7, 0b1101000, 0x29)
        # Read Gyro


        #Write data bytes to SD card
        file.write(buf1_7) # Write data to log.bin (bytes 0-6)
        file.write(buf2_7) # Write data to log.bin (bytes 7-13)
        file.write(buf3_7) # Write data to log.bin (bytes 14-20)
        file.write(buf4_7) # Write Gyro data to log.bin (bytes 21-28)
        file.write(buf_HE); # Write hall effect data to log.bin (byte 29)

while REC_BTN.value() == False: # Wait for user to let go of button
    pass








#==========================================================================
# DONE RECORDING!
#==========================================================================
print('DONE RECORDING')
                                                                 10
```

```
print()

# Close data file
file.close()

# Turn off accelerometer
ADXL375_1.standby()
ADXL375_2.standby()
ADXL375_3.standby()
# Turn off record LED
REC_LED.value(1)
# Turn off hall effect interrupt
HALL_EFF_INT            = pyb.ExtInt(pyb.Pin.cpu.C1, pyb.ExtInt.IRQ_RISING,
                                     pyb.Pin.PULL_NONE, callback=None)



print('FINISHED')
print()

Display.text('    ') # Clears current text/numbers on display
Display.number(file_count) # Prints the desired number
Display.blink_rate(0)
Display.show() # Updates the display
```

B - Gyroscope Driver Python Code

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
""
    @file           Gyro.py
    @brief          Driver class that sets up and receives data from an Gyroscope
""


from pyb import I2C
import struct
import time
import os

class Gyro:
    ''' @brief      An gyro driver class
        @details    Objects of this class can be used to recieve gyro data.
    '''
    ## Constructor
    def __init__(self):
        ''' @brief      Initializes and returns an gyro object
            @details    Initializes the gyro in master mode, allowing the
                        retrieval of data.
        '''
        self.i2c = I2C(1, I2C.MASTER)

        #FIFO Enable
        self.i2c.mem_write(0b1110001,0b110100,0x12)

        #SAMPLE RATE DIVIDER (8KHZ)
        self.i2c.mem_write(0b00000111,0b110100, 0x15)

        #Parameter Configuration
        self.i2c.mem_write(0b00011000,0b110100, 0x15)

    def Who_am_I(self, Identity):
        self.i2c.mem_write(Identity, 0b110100, 0x0)

    def USER_CONTROL(self, Control):
        self.i2c.mem_write(Control, 0b110100, 0x3D)

    def sleep(self, Power_Management):
        self.i2c.mem_write(Power_Management, 0b110100, 0x3E)

    def gyro_offsets(self, offsets):
        self.i2c.mem_write(offsets, 0b110100, 0xC)

    def Read_FIFO_COUNT(self):
        return self.i2c.mem_read(2, 0b110100, 0x3A)

    def READ_FIFO_DATA(self):
```

```
    # confused on what a burst read refers to
    return

## Method to read angular velocity from the Gyro
def omega(self):
    ''' @brief Method to read angular velocity from the Gyro to use as
        state measurements
    '''
    self.omega_signed_ints = struct.unpack('<hhh',
                                           self.i2c.mem_read(6, 0b110100,
                                                             0x29))
    self.omega_vals = tuple(self.omega_int/16 for self.omega_int in
                            self.omega_signed_ints)
    return self.omega_vals
```

C - Data Processing MATLAB Code

```
%% Accelerometer_Data_Testing.m
%==========================================================================
% ABOUT:
% This code interprets the binary data saved from the ADXL375BCCZ
% accelerometer using the "Convert_ADXL375_Data" function and then plots
% the results.
%==========================================================================
% WRITTEN BY: Steven Waal
% DATE: 02.29.2021
% UPDATED BY: DYLAN RUIZ & THEO PHILLIBER
% DATE: 06.02.2022
%==========================================================================
% NOTES:
%   02.29.2021 - File created (SRW).
%   06.02.2022 - Post Processed Data for Transmissibility Metric
%===========================================================================

% Clear statements
clear all;
close all;
clc;

%%=========================================================================
% First Test Run
%%=========================================================================
% Set the filepath here!
% filePath = "/Volumes/DATA/log/data1.bin"; % Filepath to where the binary
% data file is stored
filePath = ('data155.bin');
frequency = 1600; % [Hz] Frequency of the accelerometers

% Run "Convert_ADXL375_Data" function
out = Convert_ADXL375_Data(filePath, frequency);

% Rename outputs for clarity
time_1         = out(1:92500,1);
x1           = out(1:92500,2);
y1           = out(1:92500,3);
z1           = out(1:92500,4);
x2           = out(1:92500,5);
y2           = out(1:92500,6);
z2           = out(1:92500,7);
sum1 = 0;
sum2 = 0;
factor =2000; % Factor used to create RMS timesteps for post processing.
% The larger the factor, the bigger timestep between RMS calcs.
% length(x1) > factor > 1

% Calculates the total magnitude of accelerations per accelerometer
for i = 1:length(x1)
    total_mag_1(i) = (x1(i)^2+y1(i)^2+z1(i)^2)^(1/2);
    total_mag_2(i) = (x2(i)^2+y2(i)^2+z2(i)^2)^(1/2);
end
```

```
% calculates the total RMS of each accelerometer for the entire run
for n = 1:length(total_mag_1)
    sum1 = sum1+(total_mag_1(n))^2;
    sum2 = sum2 + (total_mag_2(n))^2;
end
Accelrms_1 = ((1/length(total_mag_1))*sum1)^(1/2);
Accelrms_2 = ((1/length(total_mag_2))*sum2)^(1/2);

% Transmissibility of the entire run
Transmiss = Accelrms_2/Accelrms_1;

% Calculates the rms of each timestep based on the factor.
for n = 1:((length(total_mag_1)/factor)-1)

    rms_array_1 = total_mag_1((n-1)*factor+1 : (n)*factor);
    rms_array_2 = total_mag_2((n-1)*factor+1 : (n)*factor);
    time_array = time_1((n-1)*factor+1 : (n)*factor);

    for i = 1:length(rms_array_1)
        sum1 = sum1+(rms_array_1(i))^2;
        sum2 = sum2+(rms_array_2(i))^2;

    end

    Accel_rms_1(n) = ((1/length(rms_array_1))*sum1)^(1/2);
    Accel_rms_2(n) = ((1/length(rms_array_2))*sum2)^(1/2);
    time_list(n,1) = mean(time_array);
    sum1 = 0;
    sum2 = 0;
end
trans = (Accel_rms_2./Accel_rms_1)';


% Plot raw data accelerometer 1
figure
plot(time_1, x1, time_1, y1, time_1, z1);
legend('X', 'Y', 'Z');
xlabel('Time, sec');
ylabel('Acceleration, g');
title('Accelerometer 1');

% Plot raw data accelerometer 2
figure
plot(time_1, x2, time_1, y2, time_1, z2);
legend('X', 'Y', 'Z');
xlabel('Time, sec');
ylabel('Acceleration, g');
title('Accelerometer 2');
%axis([0,10.5,-5,3])

% Plot magnitude accelerometer 1
figure
plot(time_1, total_mag_1);
xlabel('Time, sec');
```

```
ylabel('Acceleration Magnitude, g');
title('Accelerometer 1 Magnitude');
%axis([0,10.5,0,7])

% Plot magnitude accelerometer 1
figure
plot(time_1, total_mag_2);
xlabel('Time, sec');
ylabel('Acceleration Magnitude, g');
title('Accelerometer 2 Magnitude');
%axis([0,10.5,0,7])

% Plot RMS accelerometer 1
figure
plot(time_list, Accel_rms_1);
xlabel('Time, sec');
ylabel('Acceleration RMS, g');
title('Accelerometer 1 RMS');
%axis([0,10.5,0,5])

% Plot RMS accelerometer 2
figure
plot(time_list, Accel_rms_2);
xlabel('Time, sec');
ylabel('Acceleration RMS, g');
title('Accelerometer 2 RMS');
%axis([0,10.5,0,5])

% Plot Transmissibility
figure
plot(time_list, trans);
xlabel('Time, sec');
ylabel('Transmissibility');
title('Transmissibility [Handlebars/Axle]');

%%===========================================================================
% Second Test Run
%%===========================================================================

% Set the filepath here!
%filePath = "/Volumes/DATA/log/data1.bin"; % Filepath to where the binary
% data file is stored
filePath = ['data156.bin'];
frequency = 1600; % [Hz] Frequency of the accelerometers

% Run "Convert_ADXL375_Data" function
out = Convert_ADXL375_Data(filePath, frequency);

% Rename outputs for clarity
time_2      = out(:,1);
x1          = out(:,2);
y1          = out(:,3);
z1          = out(:,4);
x2          = out(:,5);
```

```
y2          = out(:,6);
z2          = out(:,7);
sum1 = 0;
sum2 = 0;
total_mag_1 = 0;
total_mag_2 = 0;
rms_array_1 = 0;
rms_array_2 = 0;

% Calculates the total magnitude of accelerations per accelerometer
for i = 1:length(x1)
    total_mag_1(i) = (x1(i)^2+y1(i)^2+z1(i)^2)^(1/2);
    total_mag_2(i) = (x2(i)^2+y2(i)^2+z2(i)^2)^(1/2);
end

% calculates the total RMS of each accelerometer for the entire run
for n = 1:length(total_mag_1)
    sum1 = sum1+(total_mag_1(n))^2;
    sum2 = sum2 + (total_mag_2(n))^2;
end
Accelrms_11 = ((1/length(total_mag_1))*sum1)^(1/2);
Accelrms_12 = ((1/length(total_mag_2))*sum2)^(1/2);

% Transmissibility of the entire run
Transmiss_2 = Accelrms_12/Accelrms_11;

% Calculates the rms of each timestep based on the factor.
for n = 1:((length(total_mag_1)/factor)-1)

    rms_array_1 = total_mag_1((n-1)*factor+1 : (n)*factor);
    rms_array_2 = total_mag_2((n-1)*factor+1 : (n)*factor);
    time_array = time_2((n-1)*factor+1 : (n)*factor);
    for i = 1:length(rms_array_1)
        sum1 = sum1+(rms_array_1(i))^2;
        sum2 = sum2+(rms_array_2(i))^2;

    end

    Accel_rms_11(n) = ((1/length(rms_array_1))*sum1)^(1/2);
    Accel_rms_22(n) = ((1/length(rms_array_2))*sum2)^(1/2);
    time_list_11(n,1) = mean(time_array);
    sum1 = 0;
    sum2 = 0;
end

trans_1 = (Accel_rms_22./Accel_rms_11)';


% Plot raw data accelerometer 1
figure
plot(time_2, x1, time_2, y1, time_2, z1);
legend('X', 'Y', 'Z');
xlabel('Time, sec');
ylabel('Acceleration, g');
```

```
title('Accelerometer 1');

% Plot raw data accelerometer 2
figure
plot(time_2, x2, time_2, y2, time_2, z2);
legend('X', 'Y', 'Z');
xlabel('Time, sec');
ylabel('Acceleration, g');
title('Accelerometer 2');


% Plot Magnitude accelerometer 1
figure
plot(time_2, total_mag_1);
xlabel('Time, sec');
ylabel('Acceleration Magnitude, g');
title('Accelerometer 1 Magnitude');

% Plot Magnitude accelerometer 2
figure
plot(time_2, total_mag_2);
xlabel('Time, sec');
ylabel('Acceleration Magnitude, g');
title('Accelerometer 2 Magnitude');

% Plot RMS accelerometer 1
figure
plot(time_list_11, Accel_rms_11);
xlabel('Time, sec');
ylabel('Acceleration RMS, g');
title('Accelerometer 1 RMS');

% Plot RMS accelerometer 2
figure
plot(time_list_11, Accel_rms_22);
xlabel('Time, sec');
ylabel('Acceleration RMS, g');
title('Accelerometer 2 RMS');



%%=============================================================================
% Third Test Run
%%=============================================================================

filePath = ['data157.bin'];
frequency = 1600; % [Hz] Frequency of the accelerometers

% Run "Convert_ADXL375_Data" function
out = Convert_ADXL375_Data(filePath, frequency);

% Rename outputs for clarity
time_3        = out(:,1);
x1          = out(:,2);
```

```
y1          = out(:,3);
z1          = out(:,4);
x2          = out(:,5);
y2          = out(:,6);
z2          = out(:,7);
sum1 = 0;
sum2 = 0;

% Calculates the total magnitude of accelerations per accelerometer
for i = 1:length(x1)
    total_mag_1(i) = (x1(i)^2+y1(i)^2+z1(i)^2)^(1/2);
    total_mag_2(i) = (x2(i)^2+y2(i)^2+z2(i)^2)^(1/2);
end


% Calculates the total RMS of each accelerometer for the entire run
for n = 1:length(total_mag_1)
    sum1 = sum1+(total_mag_1(n))^2;
    sum2 = sum2 + (total_mag_2(n))^2;
end
Accelrms_1 = ((1/length(total_mag_1))*sum1)^(1/2);
Accelrms_2 = ((1/length(total_mag_2))*sum2)^(1/2);

% Transmissibility of the entire run
Transmiss_3 = Accelrms_2/Accelrms_1;

% Calculates the RMS of each timestep based on the factor.
for n = 1:((length(total_mag_1)/factor)-1)

    rms_array_1 = total_mag_1((n-1)*factor+1 : (n)*factor);
    rms_array_2 = total_mag_2((n-1)*factor+1 : (n)*factor);
    time_array = time_3((n-1)*factor+1 : (n)*factor);

    for i = 1:length(rms_array_1)
        sum1 = sum1+(rms_array_1(i))^2;
        sum2 = sum2+(rms_array_2(i))^2;

    end

    Accel_rms_31(n) = ((1/length(rms_array_1))*sum1)^(1/2);
    Accel_rms_32(n) = ((1/length(rms_array_2))*sum2)^(1/2);
    time_list_3(n,1) = mean(time_array);
    sum1 = 0;
    sum2 = 0;
end
trans_3 = (Accel_rms_32./Accel_rms_31)';
%rms_array_1(1:length(total_mag_1),1) = Accel_rms_1;
%rms_array_2(1:length(total_mag_2),1) = Accel_rms_2;

% Plot raw data accelerometer 1
figure
plot(time_3, x1, time_3, y1, time_3, z1);
legend('X', 'Y', 'Z');
xlabel('Time, sec');
```

```
ylabel('Acceleration, g');
title('Accelerometer 1');

% Plot raw data accelerometer 2
figure
plot(time_3, x2, time_3, y2, time_3, z2);
legend('X', 'Y', 'Z');
xlabel('Time, sec');
ylabel('Acceleration, g');
title('Accelerometer 2');


% Plot Magnitude accelerometer 1
figure
plot(time_3, total_mag_1);
xlabel('Time, sec');
ylabel('Acceleration Magnitude, g');
title('Accelerometer 1 Magnitude');

% Plot Magnitude accelerometer 2
figure
plot(time_3, total_mag_2);
xlabel('Time, sec');
ylabel('Acceleration Magnitude, g');
title('Accelerometer 2 Magnitude');

% Plot RMS accelerometer 1
figure
plot(time_list_3, Accel_rms_31);
xlabel('Time, sec');
ylabel('Acceleration RMS, g');
title('Accelerometer 1 RMS');

% Plot RMS accelerometer 2
figure
plot(time_list_3, Accel_rms_32);
xlabel('Time, sec');
ylabel('Acceleration RMS, g');
title('Accelerometer 2 RMS');

% Plot Transmissibility Run 3
figure
plot(time_list_3, trans_3);
xlabel('Time, sec');
ylabel('Transmissibility');
title('Transmissibility [Handlebars/Axle]');

% Plot the Transmissibilities of the three runs
figure
plot(time_list, trans,'g',time_list_11,trans_1,'r',time_list_3,trans_3,'b');
xlabel('Time, sec');
ylabel('Transmissibility');
title('Transmissibility Rebound Variable [Handlebars/Axle]');
legend('8LSR/8HSR','4LSR/4HSR','2LSR/2HSR')
```

D - Converting Data MATLAB Code

```
%% Convert_ADXL375_Data.m
%==========================================================================
% ABOUT:
% This function interprets the binary data saved from the ADXL375BCCZ
% accelerometer.
%==========================================================================
% ARGUMENTS:
%   filePath        = The filepath to the binary file.
%
%   frequency       = The frequency at which the data was collected.
%==========================================================================
% OUTPUTS:
%   out             = [time, x_acceleration_1, y_acceleration_1,
%                      z_acceleration_1, x_acceleration_2, y_acceleration_2,
%                      z_acceleration_2];
%                      The interpreted data from both accelerometers (1 and
%                      2) for the x, y, and z axes.
%==========================================================================
% WRITTEN BY: Steven Waal
% DATE: 05.06.2019
% UPDATED BY: DYLAN RUIZ
% DATE: 06.02.2022
%==========================================================================
% NOTES:
%   05.06.2019 - File created (SRW). Referenced EOM_2DOF (older version).
%                (SRW)
%   09.12.2020 - Cleaned up code and added more comments. (SRW)
%   06.02.2022 - Added functionality for third accelerometer, gyro and hall
%   effect sensor.
%==========================================================================

function out = Convert_ADXL375_Data(filePath, frequency)

% CONSTANTS
SCALE_FACTOR = 0.0488;          % [g/LSB] (see ADXL375 data sheet)
SCALE_FACTOR_GYRO = 1/16.4;     % [(deg/s)/LSB] (see MPU3050 data sheet)
                                % (FS_SEL = 3)
% Open binary data file from MTB DAQ
file = fopen(filePath);

% Read binary data from file and store in variable 'data'
data = fread(file);

% Close binary data file
fclose(file);

data = reshape(data, 29, []);

% Decode data

% Add LSBs and MSBs together to reconstruct X, Y, and Z data

% Accel 1
x1 = data(2,:) + 256*data(3,:);
```

```
y1 = data(4,:) + 256*data(5,:);
z1 = data(6,:) + 256*data(7,:);

% Accel 2
x2 = data(9,:) + 256*data(10,:);
y2 = data(11,:) + 256*data(12,:);
z2 = data(13,:) + 256*data(14,:);

% Accel 3
x3 = data(16,:) + 256*data(17,:);
y3 = data(18,:) + 256*data(19,:);
z3 = data(20,:) + 256*data(21,:);

% Gyro

x4 = data(23,:) + 256*data(24,:);
y4 = data(25,:) + 256*data(26,:);
z4 = data(27,:) + 256*data(28,:);

% Hall Effect
HE = data(29,:);

% Take two's complement to get sign of data
for i=1:length(x1)
    if x1(i)>32767
        x1(i)=x1(i)-65536;
    end
    if y1(i)>32767
        y1(i)=y1(i)-65536;
    end
    if z1(i)>32767
        z1(i)=z1(i)-65536;
    end
    if x2(i)>32767
        x2(i)=x2(i)-65536;
    end
    if y2(i)>32767
        y2(i)=y2(i)-65536;
    end
    if z2(i)>32767
        z2(i)=z2(i)-65536;
    end
    if x3(i)>32767
        x3(i)=x3(i)-65536;
    end
    if y3(i)>32767
        y3(i)=y3(i)-65536;
    end
    if z3(i)>32767
        z3(i)=z3(i)-65536;
    end
    if x4(i)>32767
        x4(i)=x4(i)-65536;
    end
    if y4(i)>32767
        y4(i)=y4(i)-65536;
    end
    if z4(i)>32767
        z4(i)=z4(i)-65536;
    end
end

x1 = x1.*SCALE_FACTOR;
y1 = y1.*SCALE_FACTOR;
z1 = z1.*SCALE_FACTOR;
x2 = x2.*SCALE_FACTOR;
y2 = y2.*SCALE_FACTOR;
z2 = z2.*SCALE_FACTOR;
x3 = x3.*SCALE_FACTOR;
y3 = y3.*SCALE_FACTOR;
z3 = z3.*SCALE_FACTOR;
x4 = x4.*SCALE_FACTOR_GYRO;
y4 = y4.*SCALE_FACTOR_GYRO;
z4 = z4.*SCALE_FACTOR_GYRO;

time = [0: 1/frequency: (length(x1)-1)/frequency];

% Output the acceleration values, in g's, rotational velocity in deg/s
out = [time', x1', y1', z1', x2', y2', z2',x3',y3',z3',x4',y4',z4',HE'];

end
```

E - User Manual

The following user manual provides instructions to operate the MTB DAQ as well as important safety information. Read this section this prior to operation and see the troubleshooting section if problems arise.

Operation of DAQ System

Flashing the Main DAQ (directly from Steven Waal's Thesis)

The main board was designed based on the PYBv1.1 schematic. As a result, the firmware and flashing instructions are the same as those for the PyBoard. The main board utilizes the device firmware update (DFU) protocol that comes embedded with each STM32 microcontroller. DFU mode allows for a simple way to update the firmware of an STM32 without requiring specialized hardware. It was mainly designed for updating the firmware remotely on devices that have already been released. To flash firmware to the board, use the following steps:

1. Make sure that a DFU utility program is installed on the computer that will be used to flash the firmware. "dfu-util" is a free DFU utility program than runs in terminal. Install this program via the package manager.
2. With the power to the board turned off, move the jumper on port JP1 from "JMP STORE" position to the "DFU" position. This will tie the DFU pin of the microcontroller to 3.3V. Figure 4.11 depicts these positions. When the board is powered on, the microcontroller will enter DFU mode upon boot.
3. Connect to the board to a computer via USB.
4. Use the dfu-util commands to flash the firmware to the board. For more details on using this software to flash the board, refer to.
5. Once the firmware has been loaded on, power off the board and return the jumper on JP1 to the "JMP STORE" position. The main board runs off the standard released PYBv1.1 DFU firmware files available on the Micropython website. At the time of this writing, the most current version that worked with the main board was pybv11-20191220-v1.12.dfu. [Reference: Steven Waal's Thesis Defense]

Once the proper firmware has been loaded onto the board, the main board will appear as a standard USB device when connected to the computer. At this point, the Micropython files outlined in Section 4.5 can be loaded on to the board using a standard method for transferring files. Note that when first loading on the files, make sure that there is no Micro SD card loaded in, as this will appear instead of the USB device representing the microcontroller internal flash memory. It is important to load the files onto the internal flash memory and not the Micro SD card. Once this has been done, the MTB DAQ is ready for operation.

Formatting the SD Card

The Micro SD card needs to be formatted according to the SD card association. In order to remove old data, it is important to completely erase both the "log" and "count" folders and all of the contents in them. These folders will be remade if they don't already exist, and the proper files will be generated upon the next power up of the main unit. If only the "log" file is deleted, the system will continue counting based off of the "count.txt" file. The data will still be saved, and the proper number will be displayed on the display of the system, but the numbering will not start over as desired. Reformatting the Micro SD card is a good way to ensure that it is completely erased and ready for a new testing session. [Reference: Steven Waal's Thesis Defense]

Mounting the System

Attaching the MTB DAQ System is fast and easy to do with any bike you might be using.

The OneUp straps slide through the slots in both auxiliary sensor housings, and then they are placed with the angled surface against the bike so that the strap wraps around the chain stay, fork housing, or handlebars. It is important to place the cadence sensor on the fork housing, adjacent to the magnet location on the spokes of the front wheel, so that the wheel speed may be measured if desired. This sensor should be oriented with the cable port facing the handlebars so that the magnet is in the range of the hall effect sensor.

The central unit mounts to the middle of the main frame, with bolts going through the designated holes in the housing through the water bottle boss to secure the unit in place. Make sure the faceplate with the display screen and indicator lights is face up and visible.

Finally, the ethernet cables plug in to each auxiliary sensor unit, connecting them to the central unit. Any excess cable should be secured to the bike frame so that it does not interfere with the rider's motion. This will prevent injury to the rider and damage to the DAQ system.

Collecting the Data (directly from Steven Waal's Thesis)

Before powering on the MTB DAQ, make sure both ADXL375 accelerometers are plugged in to the main unit. The main unit configures the accelerometers upon startup and is not able to re-configure after the power has been turned on. If the accelerometers are not plugged in before the power is turned on, turn off the main unit, plug them in, and turn it back on.

The MTB DAQ is powered on and off via the power switch. Power status is indicated by the power LED (green). Upon power up, the main unit will check for the presence of a Micro SD card. If no Micro SD card is inserted, "SD" will flash continuously on the display. Once a Micro SD card is inserted, "SD" will flash three more times until the main unit detects the card. Once the card has successfully been detected, the display will show the number of the current data file stored on the Micro SD card and the record LED (red) will turn off. If the Micro SD card has just been formatted, the display will show “0” indicating that no data has been logged. The MTB DAQ is now in standby mode and ready to record data.

Data recording is started by pressing down on the record button. After the first press, the record LED (red) will turn on and the display will increment the displayed number indicating that the main unit is recording data. The number that is displayed during and after recording indicates the number of the data file associated with that recording session. Pressing the record button a second time will make the record LED (red) turn off indicating that the main unit is done recording. The display will continue to show the number from that most recent recording session. The main unit is immediately available for another recording session. When testing is finished, it is best to turn the main unit off before removing the Micro SD card. [Reference: Steven Waal’s Thesis Defense]

Interpreting the Data

After collecting the data on the Micro SD card, the files on the card should be in the format ‘.bin’. Download the MATLAB scripts named “Convert_ADXL375_Gyro_Data.m” and “Accelerometer_Data_Testing.m” (Testing script). The first file is used to convert the accelerometer, gyroscope, and hall effect data from binary to its intended values. The first file is a function file, so it will be referenced when the testing script is run. To run the testing script, you must first insert the file name that you wish to process in the line below,

```
filePath = ('data155.bin');
```

For the transmissibility metric, our team added a post-processing segment to the testing script. The post processing of the data includes turning the raw acceleration data into a total magnitude of acceleration, not dependent on direction. Using the magnitudes, we find the total RMS of the entire run and calculate the RMS at certain timesteps to make the Transmissibility plot easier to understand. The timestep is changed based on a factor, which the user can manipulate to change the timestep. The timestep is calculated to be (factor/1600) seconds. You can change the factor in the line below,

```
factor = 2000; % Factor used to create RMS timesteps for post processing.
```

The script will ultimately produce three transmissibility values for the entire run, one for each run within the script and plots similar to Figure 13.

Troubleshooting and Known Issues

Our team’s project was a continuation of Steven Waal’s Master’s Thesis, in which he designed a DAQ system consisting of only two external accelerometers. Our team worked off his design and encountered many problems with his data acquisition system. The following are some of the problems/bugs we encountered.

Battery Issues

Throughout our time working with Waal’s version of the DAQ, we encountered problems with the battery design. The DAQ at times would not display the correct message upon bootup or display no message at all. This could be due to the batteries being dead, even if it happens within a day or two of no use. Our team believes that the DAQ design is leaking power through one of the connections, even when it is switched off. If this problem occurs, unplug the batteries from the ports until you must use the DAQ again. This will keep the batteries charged and ready.

Accelerometer Issues

When operating the DAQ, at least one external accelerometer must be plugged in. The DAQ will not function correctly if the accelerometer is not plugged in via an ethernet cable. The DAQ will show it is recording, but the button will not be able to end the recording session.

Display Bugs

Sometimes, when starting a new recording session, the number on the display won't change. This is a common bug that happens, but it does not affect the file creation of the recording session. If another recording session is created, the display will skip the number of the previous recording and display the correct file number.

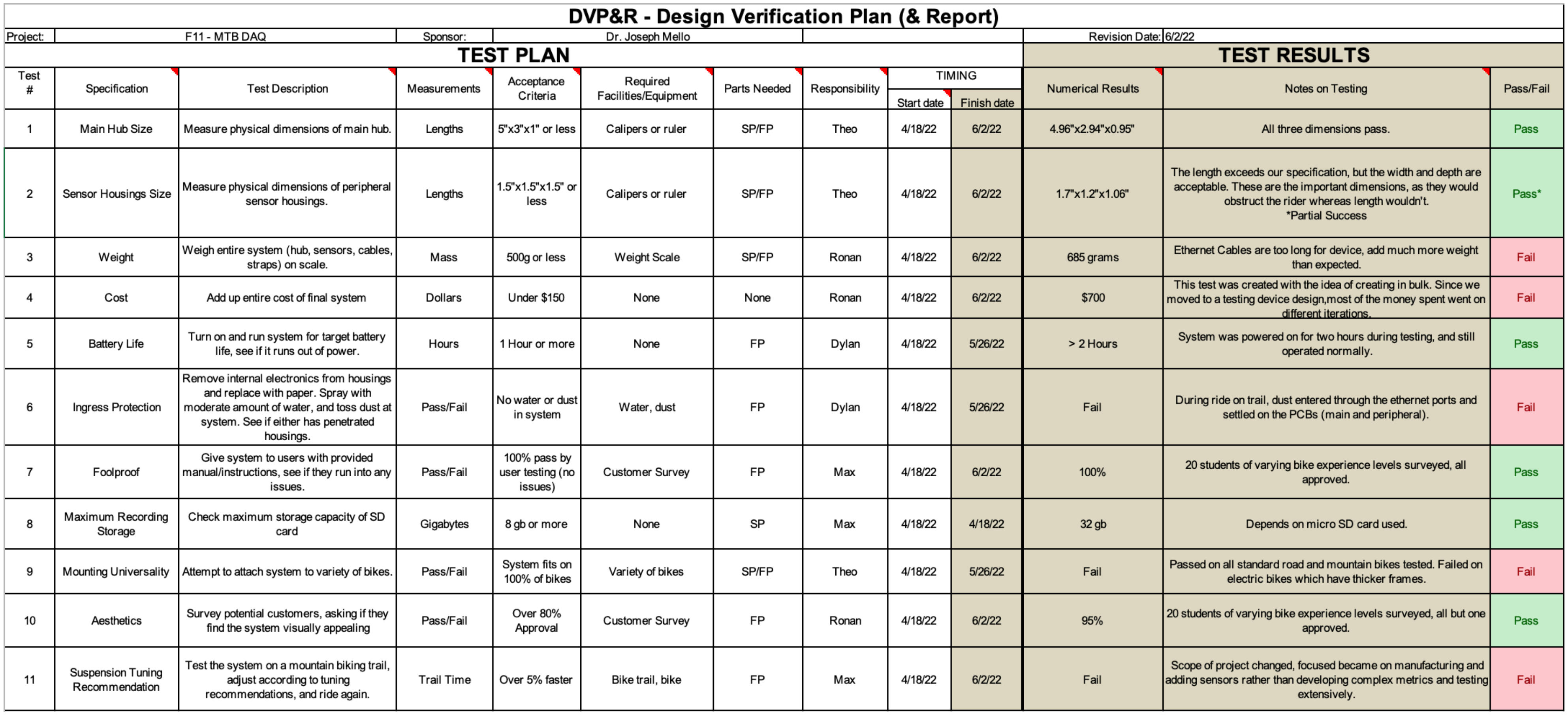
Reset Button

On the PCB of the main DAQ, there is a small black button labeled with the word 'reset'. Pressing this button does not restart the system, but instead clears all of the program files off of the MCU. If this button is pressed, the device must be plugged into a computer with USB, and the files copied back onto the device labeled 'PybFlash'.

Soldered Connections

If issues not listed above are occurring, they may be the result of broken solder joints. We have dealt with the display not working properly and identified the issue to be a broken solder joint in the record button. This button is held in place by solder, and the force of pressing the button can break the connection. Through visual inspection and voltmeter readings, the PCBs can be analyzed to find these iss

F - Design Verification Plan & Report



DVP&R - Design Verification Plan (& Report)

Project: F11 - MTB DAQ | Sponsor: Dr. Joseph Mello | Revision Date: 6/2/22

TEST PLAN										TEST RESULTS		
								TIMING				
Test #	Specification	Test Description	Measurements	Acceptance Criteria	Required Facilities/Equipment	Parts Needed	Responsibility	Start date	Finish date	Numerical Results	Notes on Testing	Pass/Fail
1	Main Hub Size	Measure physical dimensions of main hub.	Lengths	5"x3"x1" or less	Calipers or ruler	SP/FP	Theo	4/18/22	6/2/22	4.96"x2.94"x0.95"	All three dimensions pass.	Pass
2	Sensor Housings Size	Measure physical dimensions of peripheral sensor housings.	Lengths	1.5"x1.5"x1.5" or less	Calipers or ruler	SP/FP	Theo	4/18/22	6/2/22	1.7"x1.2"x1.06"	The length exceeds our specification, but the width and depth are acceptable. These are the important dimensions, as they would obstruct the rider whereas length wouldn't. *Partial Success	Pass*
3	Weight	Weigh entire system (hub, sensors, cables, straps) on scale.	Mass	500g or less	Weight Scale	SP/FP	Ronan	4/18/22	6/2/22	685 grams	Ethernet Cables are too long for device, add much more weight than expected.	Fail
4	Cost	Add up entire cost of final system	Dollars	Under $150	None	None	Ronan	4/18/22	6/2/22	$700	This test was created with the idea of creating in bulk. Since we moved to a testing device design,most of the money spent went on different iterations.	Fail
5	Battery Life	Turn on and run system for target battery life, see if it runs out of power.	Hours	1 Hour or more	None	FP	Dylan	4/18/22	5/26/22	> 2 Hours	System was powered on for two hours during testing, and still operated normally.	Pass
6	Ingress Protection	Remove internal electronics from housings and replace with paper. Spray with moderate amount of water, and toss dust at system. See if either has penetrated housings.	Pass/Fail	No water or dust in system	Water, dust	FP	Dylan	4/18/22	5/26/22	Fail	During ride on trail, dust entered through the ethernet ports and settled on the PCBs (main and peripheral).	Fail
7	Foolproof	Give system to users with provided manual/instructions, see if they run into any issues.	Pass/Fail	100% pass by user testing (no issues)	Customer Survey	FP	Max	4/18/22	6/2/22	100%	20 students of varying bike experience levels surveyed, all approved.	Pass
8	Maximum Recording Storage	Check maximum storage capacity of SD card	Gigabytes	8 gb or more	None	SP	Max	4/18/22	4/18/22	32 gb	Depends on micro SD card used.	Pass
9	Mounting Universality	Attempt to attach system to variety of bikes.	Pass/Fail	System fits on 100% of bikes	Variety of bikes	SP/FP	Theo	4/18/22	5/26/22	Fail	Passed on all standard road and mountain bikes tested. Failed on electric bikes which have thicker frames.	Fail
10	Aesthetics	Survey potential customers, asking if they find the system visually appealing	Pass/Fail	Over 80% Approval	Customer Survey	FP	Ronan	4/18/22	6/2/22	95%	20 students of varying bike experience levels surveyed, all but one approved.	Pass
11	Suspension Tuning Recommendation	Test the system on a mountain biking trail, adjust according to tuning recommendations, and ride again.	Trail Time	Over 5% faster	Bike trail, bike	FP	Max	4/18/22	6/2/22	Fail	Scope of project changed, focused became on manufacturing and adding sensors rather than developing complex metrics and testing extensively.	Fail

Design Verification Plan & Report (DVP&R)

G - Risk Assessment

Senior Design Project F11 Risk Assesment 2/16/2022

designsafe Report

Application:	Senior Design Project F11 Risk Assesment	Analyst Name(s):	Dylan Ruiz, Max Ringrose, Ronan Schaffer, Theo Philliber
Description:	This assesment shows the risks and hazards when using our product. Does not include the regular hazards when riding a mountain bike because our product is not related to the riding portion.	Company:	Cal Poly San Luis Obispo
Product Identifier:		Facility Location:	San Luis Obispo, CA
Assessment Type:	Detailed		
Limits:			
Sources:	personnel experiences, ANSI B11 standards, assembly drawings W-Z		
Risk Scoring System:	ANSI B11.0 (TR3) Two Factor		

Guide sentence: When doing [task], the [user] could be injured by the [hazard] due to the [failure mode].

Item Id	User / Task	Hazard / Failure Mode	Initial Assessment Severity Probability	Risk Level	Risk Reduction Methods /Control System	Final Assessment Severity Probability	Risk Level	Status / Responsible /Comments /Reference
1-1-1	Mountain Bike Rider normal operation	slips / trips / falls : User Interference with Wires / Housings User does not install product correctly, wires and housing are in the path of the user when riding a mountain bike.	Moderate Likely	Medium	Include instruction manual giving step by step instructions on how to install device, other design change, fixed enclosures / barriers	Moderate Unlikely	Low	TBD Theo
1-1-2	Mountain Bike Rider normal operation	fire and explosions : Battery Explosion Batteries explode when impacted with high force	Serious Unlikely	Medium	Create the housing with a sturdy material (we will just be using PLA) and incease thickness of housing, warning label(s)	Serious Remote	Low	Complete [2/16/2022] Ronan

Mounting Universality Test

By Team F11 – MTB DAQ
Theo Philliber
Dylan Ruiz
Ronan Shaffer
Max Ringrose

Created: March 7th, 2022
Revision: 2

Purpose

To ensure our design for the mounting apparatus will effectively apply to any bike frame, without the effect of distorting data.

Test Equipment Required

- Helmets for rider and attendees
- Rider for each bike
- Stopwatch
- Tape
- Calipers

Hazards

Most hazards will be bike-riding related. All attendees will be positioned uphill from the rider, except the person taking the time. The person taking the time will stand at a comfortable distance from the bike path.

Safety Issues	Responses
Rider crashes bike while riding.	While this is a possibility, the rider will be experienced and will be wearing the proper safety gear.
Rider crashes into others.	The people conducting the tests will not stand in the path of the rider, and the one taking the time at the bottom of the downhill segment will be standing at a minimum of 20 feet from the path.
Wires interfere with the rider.	The wires will be tied to the frame of the bike, allowing enough clearance between the rider's natural pedal path and the wires.

Procedure

1) Gather ten different bike models and riders (Friends and people from bike club).

2) Go to a short, dedicated segment on a bike trail.

3) Set all bikes' suspension settings to similar settings (max rebound and compression dampening)

4) Mount the DAQ to the water bottle boss and the accelerometers to both the rear axle and front axle on one of the bikes.

5) Mark the position of the Accelerometers using tape.

6) Coordinate start of segment riding with stopwatch person. Press the Record button on the DAQ.

7) Driver rides segment to finish at ~ 5 sec.

8) Press the Record button on the DAQ to stop recording. Stopwatch person inspects the accelerometer's position from the tape using calipers. Records the movement of the accelerometer from initial position. Repeat steps 4-7 to sample each bike.

9) Analyze the data, accelerations should be within the same magnitude of each other.

Results

Pass:

- Accelerometer displacement must be less than .2in
- Data between bike models should be reasonably similar

Fail:

- Accelerometer displacement is greater than .2in
- Data between bike models is distorted (RMS values of acceleration should be within +/- 1g of control data)

Test Dates:

Test Results:

Table 1. Accelerometers Offset

Bike Model (Insert Bike models here)	Front Axle Accelerometer Displacement (in)	Rear Axle Accelerometer Displacement (in)

Customer Survey Procedures

By Team F11 – MTB DAQ

Theo Philliber

Dylan Ruiz

Ronan Shaffer

Max Ringrose

Created: April 5th, 2022
Revision: 1

Mechanical Engineering
CAL POLY

Purpose

The purpose of this test is to validate that the DAQ meets the subjective criteria of 'aesthetics' and 'foolproof-ness'. These will both be performed by surveying people to represent customers.

Scope

The scope of this protocol is to validate the aesthetic appeal and ease-of-use of the DAQ unit. Since both specifications will be performed by directly surveying customers, they will be performed at the same time and are laid out together in this document. The ease-of-use specification will be tested by providing users with a basic manual and the DAQ and letting them operate it. The aesthetics will be a numerical scoring by the users after they have tested it out.

Equipment

The following list requires all equipment necessary to complete this test protocol.

- DAQ Unit (Full Prototype)
- People to survey

Hazards

There are no potential hazards to this procedure.

PPE Requirements

- None

Facility

This test will take place anywhere people agree to meet on an individual basis.

Procedure

1. EASE OF USE:
   a. Meet with a person who agreed to be interviewed (hereafter referred to as the user)
   b. Provide user with operating manual and DAQ.
   c. Allow user to attempt to use DAQ system, providing minimal feedback or guidance.
   d. Afterwards, ask user of any issues they had operating the device, and ask to rank the ease-of-use on a 1-10 scale.
2. AESTHETICS
   a. After user has gotten familiar with operating the system, ask them to rank the aesthetic appeal of the device on a 1-10 scale.

Results

- Ease-of-Use Pass Criteria: No major user issues, average rating of 8 or higher.
- Aesthetics Pass Criteria: Average rating of 8 or higher.
- Number of Samples: 3

Test Date:

Test Results:

User #	Ease of Use	Aesthetics	Comments
1	10	10	-
2	10	9	Dislikes cables
3	10	9	Cables too long
4	10	10	-
5	10	9	Profile too big
6	10	10	Likes main DAQ enclosure
7	10	10	Likes main DAQ enclosure
8	10	10	-
9	10	10	Likes main DAQ enclosure
10	10	9	Cables too long
11	10	8	Cables too long
12	10	10	-
13	10	10	-
14	10	10	-
15	10	9	Dislikes sensor housing prints, too 'blocky'
16	10	10	-
17	10	9	Cables too long
18	10	9	Cables too long
19	10	10	Likes main DAQ enclosure
20	10	9	Cables too long
AVG:	**10**	**9.5**	Nobody had issues with operating or connecting the device. Everybody liked the aesthetics, with the only common complaint being the length/size of the cables.

Ingress Protection Validation Test

By Team F11 – MTB DAQ

Theo Philliber

Dylan Ruiz

Ronan Shaffer

Max Ringrose

Created: March 7th, 2022
Revised: April 21st, 2022
Revision: 2

Purpose

To evaluate the extent to which the housings protect the sensitive electronic components contained inside.

Scope

This test is for the waterproof/dustproof feature of the housings

Equipment

The following list requires all equipment necessary to complete this test protocol.

- 3D printed housings
- Mock ethernet cable
- Paper or tissue
- Tape
- Water
- Dirt/dust

Hazards

N/A

PPE Requirements

None

Facility

Teammate's backyard

Procedure

1) Remove electronics from housings

2) Insert tissue paper in housing

3) Plug mock ethernet cable into port

4) Splash water by hand or spray water with bottle or hose at housing, especially at port opening

5) Remove tissue paper and inspect for water marks

6) Replace with fresh tissue paper, surrounded with tape, sticky side facing outwards

7) Plug mock ethernet cable into port

8) Throw dirt at housing, especially at port opening

9) Remove tape and inspect for dirt/debris accumulation

10) Write down notes from visual inspection with a 0-10 rating for ingress protection

0 –	Water: Soaking wet tissue paper, lots of water inside
	Dirt/Debris: Tape is covered with dirt/dust
10 –	Water: Bone dry tissue paper, no trace of moisture
	Dirt/Debris: Perfectly clean tape, no trace of dirt or debris

Results

Pass Criteria: 9/10 for both water and dirt

Fail Criteria: 8/10 or lower

Conduct each test 3 times for both main and auxiliary housings to ensure accuracy/repeatability of result

Test Date(s):

Test Results:

Performed By:

Battery Life Validation Test

By Team F11 – MTB DAQ

Theo Philliber

Dylan Ruiz

Ronan Shaffer

Max Ringrose

Created: April 5th, 2022
Revision: 2

Purpose

The purpose of this test protocol is to validate the battery life of the DAQ system during operation. Specifically, to verify that the duration of operation on a single full charge meets specification.

Scope

The scope of this protocol is to validate the battery life of the DAQ system during operation. The life of the battery will be estimated by measuring the current and voltage used by the system and calculating the power from this. This will be compared with the battery capacity to calculate how long the DAQ will run on a single charge.

Equipment

The following list requires all equipment necessary to complete this test protocol.

- DAQ Unit
- Multimeter
- Jumper wires

Hazards

The hazards of this test are entirely based on the batteries. Electric shock can occur, but will not likely be harmful at the low current and voltages supplied. The other risk could be from puncturing the pouch-style batteries, which can then become explosive. Care should be taken with the batteries to not puncture them.

Hazard	Danger Level	Likelihood	Preventative Action
Battery Puncture	Dangerous	Low	Following the procedures should have little to no possibility of puncturing the batteries. If the batteries aren't easily going back in when closing the case, take them back out and reseat the wires so they fit better.
Electric Shock	Low	Medium	Power system off before making changes to wiring or measuring the current and voltage.

PPE Requirements

- Safety Glasses

Facility

This test will take place in the ME 305/405 Laboratory, or anywhere with a multimeter.

Procedure

1. VOLTAGE MEASUREMENT:
   a. Set the multimeter to voltage mode.
   b. With the DAQ off, connect the multimeter probes to the positive and negative terminals of the battery.
   c. Power the DAQ and mark the voltage.
   d. Try other operations on the DAQ (start/stop recording, etc.) and observe (if any) changes in power consumption from the DAQ. Power on and off DAQ twice and observe changes.
2. CURRENT MEASUREMENT:
   a. Power the DAQ off and set the multimeter to current mode.
   b. Disconnect the battery from the PCB board via the quick-connect jack.
   c. Plug a jumper wire from the ground terminal of the quick-connect to ground terminal on PCB.
   d. Connect multimeter wire to positive terminal of battery and to PCB.
   e. Power the DAQ and read the current.
   f. Power on and off DAQ twice more and observe any changes in current.

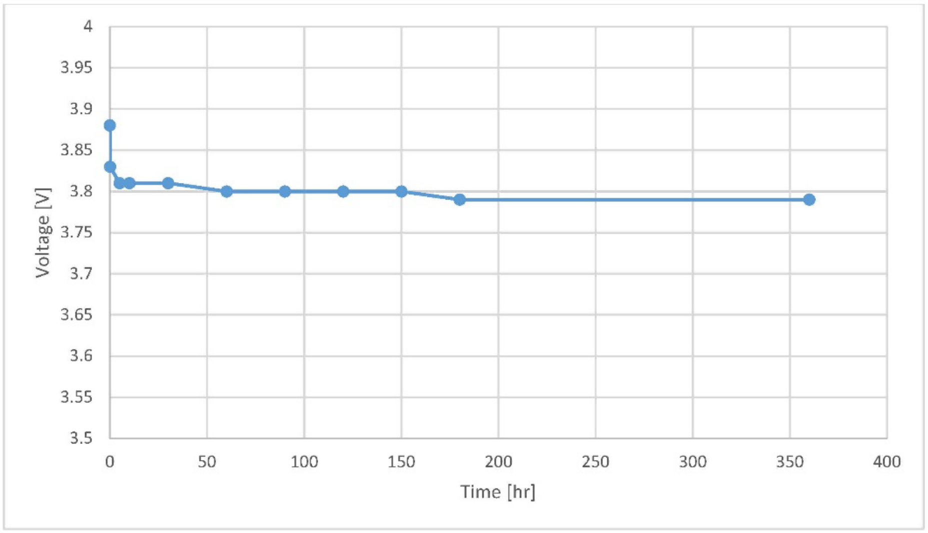
Results

- Pass Criteria: Voltage stays above 3.5V for at least one hour of continuous operation.

Test Date: 5/9/22

Test Results: PASS

Time (mins)	Voltage (V)
0	3.88
0.1	3.83
5	3.81
10	3.81
30	3.81
60	3.80
90	3.80
120	3.80
150	3.80
180	3.79
360	3.79



Based on the data collected in this test, the battery will last for far longer than one hour. It loses some voltage immediately, but then stays mostly consistent for our 6 hour test range, far exceeding our specification of over 3.5V for 1 hour.

Hall Effect Validation Test

By Team F11 – MTB DAQ

Theo Philliber

Dylan Ruiz

Ronan Shaffer

Max Ringrose

Created: March 7th, 2022
Revision: 2

Mechanical Engineering
CAL POLY

Purpose

The purpose of this test protocol is to validate the speed collected by the Allegro MicroSystems APS12205LUAA hall effect sensor is accurate and precise for all conditions of riding.

Scope

The scope of this protocol is to validate the hall effect on a benchtop truing stand using a calibrated rotational speed sensor. The results of the speed collected by the hall effect will be compared with the calibrated rotational speed sensor. The goal of this test is to develop code that will provide speed accuracy of ± 5% using the hall effect sensor as the independent variable and the rotational speed sensor as the control. The dependent the code constant of spoke magnet radius *r* will be modified until trials achieve the accuracy criteria.

Equipment

The following list requires all equipment necessary to complete this test protocol.

- 29" MTB Wheel
- 15x110 MTB through-axle
- Wheel Truing Stand
- 2 C-clamps
- Wahoo rpm cycling speed sensor
- MTB DAQ Front sensor and main DAQ
- Spoke Magnet
- Drill Adapter
- Cordless drill

Hazards

The hazards of this test include the wheel rotating at high speeds so stay clear of the spokes of the wheel. Ensure that the axle is secured in the truing stand and the truing stand is secured to the table. Make sure the drill is held firmly and to use lower speeds. If something is to be tangled in the wheel, immediately stop the drill.

PPE Requirements

- Safety Glasses

Facility

This test will take place in the ME 305/405 Laboratory.

Procedure

1) Using the 2 C-clamps, attach the truing stand to a desktop.
2) Attach the 29" MTB wheel to the Truing stand using the through-axle and end nuts.
3) Attach the drill adapter to the axle and secure the drill to the adapter.
4) Begin rotational trials. Record speeds every 10 seconds for 2 minutes.

Note: Stay clear of the spinning wheel

Results

- Pass Criteria: ± 5% percent error of rotational speed sensor
- Number of Samples: 5 tests, 2 minutes long

Test Date:

Test Results:

Hall Effect	▨
Rotational Speed Sensor	□

	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
Speed #1					
Speed #2					
Speed #3					
Speed #4					
Speed #5					
Speed #6					
Speed #7					
Speed #8					
Speed #9					
Speed #10					
Speed #11					
Speed #12					

I – iBOM (Indented Bill of Materials)

Indented Bill of Material (iBOM)

Assy Level	Part Number	Descriptive Part Name	Qty	Part Cost	Total Cost	Source	URL	More Info
		Lvl0 Lvl1 Lvl2 Lvl3 Lvl4						
0	1.0.0.0.0	Final Assy				------		
1	1.1.0.0.0	|------- Main DAQ				------		
2	1.1.1.0.0	| |--- Housing Assy				------		
3	1.1.1.1.0	| | |--- Housing	1			custom		3D Printed in PLA
3	1.1.1.2.0	| | |--- Bolts	2	$ 0.31	$ 0.62	McMaster	https://www.mcmas	M5X0.8 35MM LONG HEX CAP SCREW
3	1.1.1.3.0	| | |--- Rubber Pad	1	$ 8.99	$ 8.99	Amazon	https://www.amazo	3M Self-Stick Rubber Anti-Skid Pad
3	1.1.1.4.0	| | |--- OneUp Strap	1	$ 16.50	$ 16.50	Amazon	https://www.amazo	OneUp Components EDC Gear Straps Grey, Pair
2	1.1.2.0.0	| |--- Power Switch	1	$ 1.96	$ 1.96	Digikey	https://www.digikey	Rocker Switch SPST 10A (AC) 125 V Panel Mount, Snap-In
2	1.1.3.0.0	| |--- Quick Connect Cable Crimp	4	$ 1.61	$ 6.44	Digikey	https://www.digikey	CONN QC RCPT 14-18AWG 0.187
2	1.1.4.0.0	| |--- Receptacle Sleeve	4	$ 0.18	$ 0.72	Digikey	https://www.digikey	CONN RCPT SLEEVE 0.187 1POS CLR
2	1.1.5.0.0	| |--- Ribbon Cable	1		$ -	Digikey	https://www.digikey	CABLE ASSEM .05" 10POS F-F 2"
3	1.1.6.0.0	| |--- Ethernet Cables	2	$ 5.05	$ 10.10	McMaster	https://www.mcmas	3 ft Ethernet Cord RJ45
2	1.1.7.0.0	| |--- Thread Locker	0.01	$ 15.35	$ 0.15	McMaster	https://www.mcmas	ADJUSTABLE THREAD LOCKER; 0.34 OZ. CAN
2	1.1.8.0.0	| |--- PCB Assy				------		
3	1.1.8.1.0	| | |--- PCB	1	$ 54.00	$ 54.00	JLC PCB		Custom printed circuit board for main DAQ
3	1.1.8.2.0	| | |--- Gyroscope (MPU-3050)	1	$ 8.26	$ 8.26	Digikey	https://www.digikey	GYROSCOPE I^{2}C 24-QFN (4x4)
3	1.1.8.3.0	| | |--- Batteries Assy			$ -	------		
4	1.1.8.3.1	| | | |--- 370 mAh Battery	4	$ 9.49	$ 37.96	Digikey	https://www.digikey	3.7 V Lithium Polymer Battery Rechargeable (Secondary) 370mAh
4	1.1.8.3.2	| | | |--- 1.2 Ah Battery	1	$ 9.95	$ 9.95	Digikey	https://www.digikey	3.7 V Lithium-Ion Battery Rechargeable (Secondary) 1.2Ah
3	1.1.8.4.0	| | |--- Accelerometer (ADXL375)	1	$ 11.24	$ 11.24	Arrow	https://www.arrow.	Accelerometer Triple ±200g 2.5V/3.3V 14-Pin LGA T/R
3	1.1.8.5.0	| | |--- Accel Breakout Board	1	$ 7.79	$ 7.79	DigiKey	https://www.digikey	LGA-14 TO DIP-14 SMT ADAPTER
3	1.1.8.6.0	| | |--- Ferrite Bead	1	$ 0.26	$ 0.26	DigiKey	https://www.digikey	30 Ohms @ 100 MHz 1 Power Line Ferrite Bead 0603, 6A 8mOhm
3	1.1.8.7.0	| | |--- 0.1 µF Capacitor	6	$ 0.15	$ 0.90	DigiKey	https://www.digikey	0.1 µF ±10% 50V Ceramic Capacitor X7R 1206
3	1.1.8.8.0	| | |--- 1 µF Capacitor	2	$ 0.13	$ 0.26	Digikey	https://www.digikey.com/product-detail/en/kemet/C0805C105K4RACTU/399-1284-1-ND/416060	
3	1.1.8.9.0	| | |--- 2.2 µF Capacitor	2	$ 0.25	$ 0.50	Digikey	https://www.digikey.com/product-detail/en/kemet/C0805C225K8RACAUTO/399-6951-1-ND/3314459	
3	1.1.8.10.0	| | |--- 4.7 µF Capacitor	3	$ 0.13	$ 0.39	Digikey	https://www.digikey.com/product-detail/en/kemet/C0805C475K9PACTU/399-3134-1-ND/551639	
3	1.1.8.11.0	| | |--- 10 µF Capacitor	1	$ 0.18	$ 0.18	DigiKey	https://www.digikey	10 µF ±10% 25V Ceramic Capacitor X5R 0805
3	1.1.8.12.0	| | |--- 2 pF Capacitor	2	$ 1.29	$ 2.58	Digikey	https://www.digikey.com/product-detail/en/kemet/CBR08C209BAGAC/399-8701-1-ND/3480301	
3	1.1.8.13.0	| | |--- 24 pF Capacitor	2	$ 1.49	$ 2.98	Digikey	https://www.digikey.com/product-detail/en/kemet/CBR08C240JAGAC/399-8758-1-ND/3481277	
3	1.1.8.14.0	| | |--- 2200 pF Capacitor	1	$ 0.02	$ 0.02	DigiKey	https://www.digikey	2200 pF ±10% 50V Ceramic Capacitor X7R 0402
3	1.1.8.15.0	| | |--- Schottky Diode	1	$ 0.35	$ 0.35	Digikey	https://www.digikey	DIODE SCHOTTKY 20V 1A SOD123W
3	1.1.8.16.0	| | |--- Mini-B USB Jack	1	$ 1.08	$ 1.08	Digikey	https://www.digikey	SMT USB Jack for USB Mini-B cable
3	1.1.8.17.0	| | |--- Quick Connect 0.187"	1	$ 0.23	$ 0.23	Digikey	https://www.digikey.com/product-detail/en/molex/0197084011/WM6456-ND/459342	
3	1.1.8.18.0	| | |--- RJ45_SURFACE_MOUNT	2	$ 1.60	$ 3.20	Digikey	https://www.digikey	SMT RJ45 Jack for ethernet cable
3	1.1.8.19.0	| | |--- Blue LED	1	$ 0.18	$ 0.18	Digikey	https://www.digikey	LED BLUE CLEAR 0805 SMD
3	1.1.8.20.0	| | |--- Green LED	1	$ 0.18	$ 0.18	Digikey	https://www.digikey	LED GREEN CLEAR 0805 SMD
3	1.1.8.21.0	| | |--- Red LED	1	$ 0.18	$ 0.18	Digikey	https://www.digikey	LED RED CLEAR 0805 SMD
3	1.1.8.22.0	| | |--- Yellow LED	1	$ 0.18	$ 0.18	Digikey	https://www.digikey	LED YELLOW CLEAR 0805 SMD
3	1.1.8.23.0	| | |--- LIPO Charger	1	$ 0.56	$ 0.56	Digikey	https://www.digikey	IC CONTROLLR LI-ION 4.2V SOT23-5
3	1.1.8.24.0	| | |--- 100 KΩ Resistor	5	$ 0.10	$ 0.50	Digikey	https://www.digikey	RES SMD 100K OHM 1% 1/3W 0805
3	1.1.8.25.0	| | |--- 10 KΩ Resistor	1	$ 0.50	$ 0.50	Digikey	https://www.digikey	RES SMD 10K OHM 0.1% 1/4W 0805
3	1.1.8.26.0	| | |--- 4.7 KΩ Resistor	5	$ 0.10	$ 0.50	Digikey	https://www.digikey	CRGCQ 0805 4K7 1%
3	1.1.8.27.0	| | |--- 560 Ω Resistor	6	$ 0.10	$ 0.60	Digikey	https://www.digikey	RES SMD 560 OHM 1% 1/8W 0805
3	1.1.8.28.0	| | |--- 22 Ω Resistor	2	$ 0.10	$ 0.20	Digikey	https://www.digikey	RES SMD 22 OHM 1% 1/8W 0805
3	1.1.8.29.0	| | |--- Reset Switch	1	$ 0.41	$ 0.41	Digikey	https://www.digikey	SWITCH TACTILE SPST-NO 0.05A 12V
3	1.1.8.30.0	| | |--- SD Slot	1	$ 2.45	$ 2.45	Digikey	https://www.digikey	CONN MICRO SD CARD PUSH-PUSH R/A
3	1.1.8.31.0	| | |--- Microcontroller	1	$ 11.61	$ 11.61	Digikey	https://www.digikey	STM32F405RGT6 - IC MCU 32BIT 1MB FLASH 64LQFP
3	1.1.8.32.0	| | |--- Voltage Regulator	1	$ 0.42	$ 0.42	Digikey	https://www.digikey	IC REG LINEAR 3.3V 1A SOT223
3	1.1.8.33.0	| | |--- 10-Pin Connector Header	1	$ 2.78	$ 2.78	Digikey	https://www.digikey	CONN HEADER SMD 10POS 1.27MM
3	1.1.8.34.0	| | |--- 12 MHz Clock	1	$ 0.50	$ 0.50	Digikey	https://www.digikey	CRYSTAL 12.0000MHZ 18PF SMD
3	1.1.8.35.0	| | |--- 32.768 KHz Clock	1	$ 0.77	$ 0.77	Digikey	https://www.digikey	CRYSTAL 32.7680KHZ 6PF SMD
3	1.1.8.36.0	| | |--- Jumper	2	$ 0.31	$ 0.62	Digikey	https://www.digikey	JUMPER W/TEST PNT 1X2PINS 2.54MM
3	1.1.8.37.0	| | |--- Battery Connector	2	$ 1.86	$ 3.72	Digikey	https://www.digikey	CONN HEADER SMD R/A 2POS 2MM
3	1.1.8.38.0	| | |--- 3 Pin Header	4	$ 0.13	$ 0.52	Digikey	https://www.digikey	CONN HEADER VERT 3POS 2.54MM
3	1.1.8.39.0	| | |--- 2K Ω Resistor	10	$ 0.47	$ 4.74	Digikey	https://www.digikey	RES SMD 2K OHM 0.1% 1/4W 0805
3	1.1.8.40.0	| | |--- 44.2 Ω Resistor	6	$ 0.64	$ 3.84	Digikey	https://www.digikey	RES SMD 44.2 OHM 0.1% 1/4W 0805

3	1.1.8.41.0	| | |--- 80.6 Ω Resistor	4	$	0.64	$	2.56	Digikey	https://www.digikey	RES SMD 80.6 OHM 0.1% 1/4W 0805
2	1.1.9.0.0	| |--- UI Board Assy								
3	1.1.9.1.0	| |--- UI BOARD PCB	1	$	34.55	$	34.55	JLC PCB		UI BOARD PRINTED CIRCUIT BOARD
3	1.1.9.2.0	| |--- LED HOLDER	1					custom		3D Printed LED holder
3	1.1.9.3.0	| |--- 7 SEGMENT DISPLAY	1	$	9.95	$	9.95	Adafruit	https://www.adafrui	Quad Alphanumeric Display - Red 0.54" Digits w/ I2C Backpack
3	1.1.9.4.0	| |--- CHG	1	$	2.49	$	2.49	Digikey	https://www.digikey	PNL MNT W/ WIRE 590NM 25MCD YLW
3	1.1.9.5.0	| |--- PWR	1	$	2.49	$	2.49	Digikey	https://www.digikey	PNL MNT W/ WIRE 568NM 25MCD GRN
3	1.1.9.6.0	| |--- R1, R3	2	$	0.66	$	1.32	Digikey	https://www.digikey	RES SMD 44.2 OHM 0.1% 1/4W 0805
3	1.1.9.7.0	| |--- R2	1	$	0.62	$	0.62	Digikey	https://www.digikey	RES SMD 80.6 OHM 0.1% 1/4W 0805
3	1.1.9.8.0	| |--- R4	1	$	0.10	$	0.10	Digikey	https://www.digikey	CRGCQ 0805 4K7 1%
3	1.1.9.9.0	| |--- REC	1	$	2.49	$	2.49	Digikey	https://www.digikey	PNL MNT W/ WIRE 625NM 25MCD RED
3	1.1.9.10.0	| |--- STANDOFFS	1	$	11.83	$	11.83	Mcmaster	https://www.mcmas	Splined Press-Fit Threaded Standoffs with Open End; Miniature, 2.4mm OD, 2mm Long
3	1.1.9.11.0	| |--- U$4	1	$	8.52	$	8.52	Digikey	https://www.digikey	SWITCH PUSHBUTTON SPST-NO 2A 48V
3	1.1.9.12.0	| |--- X1	1	$	2.78	$	2.78	Digikey	https://www.digikey	CONN HEADER SMD 10POS 1.27MM
3	1.1.9.13.0	| |--- #3 Washer	1	$	2.82	$	2.82	McMaster	https://www.mcmas	316 Stainless Steel Washer; for Number 3 Screw Size, 0.109" ID, 0.25" OD
3	1.1.9.14.0	| |--- 1/4" Spacer	4	$	0.27	$	1.08	McMaster	https://www.mcmas	Aluminum Unthreaded Spacer; 1/4" OD, 3/16" Long, for Number 2 Screw Size
3	1.1.9.15.0	| |--- M1 Washer	1	$	3.84	$	3.84	McMaster	https://www.mcmas	18-8 Stainless Steel Washer for M1 Screw Size, 1.1 mm ID, 3.2 mm OD
3	1.1.9.16.0	| |--- M1x0.25mm Slotted Screw	1	$	8.00	$	8.00	McMaster	https://www.mcmas	18-8 Stainless Steel Narrow Cheese Head Slotted Screws; M1 x 0.25mm Thread, 10mm Long
1	1.2.0.0.0	|------- Front Sensor						------		
2	1.2.1.0.0	| |--- Housing Assy						------		
3	1.2.1.1.0	| | |--- Housing	1					custom		3D Printed in PLA
3	1.2.1.2.0	| | |--- Strap	1	$	16.50	$	16.50	Oneup	https://www.oneupc	14" stretch polyurethane strap with tail clip.
3	1.2.1.3.0	| | |--- M3X0.5 Brass Inserts	4	$	0.50	$	2.00	McMaster	https://www.mcmas	Brass Heat-Set Inserts for Plastic; M3 x 0.50 mm Thread Size, 5.700 mm Installed Length
3	1.2.1.4.0	| | |--- M3X0.5X6 Screw	4	$	0.42	$	1.66	McMaster	https://www.mcmas	Stainless Steel Pan Head Screws; with External-Tooth Washer, M3 x 0.5 mm Thread, 6 mm Long
3	1.2.1.5.0	| | |--- 1/4" Washer	4	$	0.10	$	0.40	McMaster	https://www.mcmas	316 Stainless Steel Washer; for 1/4" Screw Size, 0.281" ID, 0.625" OD
2	1.2.2.0.0	| |--- Front PCB Assy								
3	1.2.2.1.0	| |--- Accelerometer (ADXL375)	1	$	11.24	$	11.24	Arrow	https://www.arrow.c	Accelerometer Triple ±200g 2.5V/3.3V 14-Pin LGA T/R
3	1.2.2.2.0	| |--- Hall Effect Sensor	1	$	0.98	$	0.98	Digikey	https://www.digikey	Digital Switch Latch Open Drain Hall Effect 3-SIP
3	1.2.2.3.0	| |--- Spoke magnet	1	$	5.00	$	5.00	REI	https://www.rei.con	MSW Universal Speed Sensor Spoke Magnet
3	1.2.2.4.0	| |--- PCB	1	$	34.00	$	34.00	JLC PCB		Custom printed circuit board for rear sensor
3	1.2.2.5.0	| |--- OR Gate	1	$	0.29	$	0.29	Digikey	https://www.digikey	IC GATE OR 1CH 2-INP SC70-5
3	1.2.2.6.0	| |--- RJ45 Jack	1	$	1.60	$	1.60	Digikey	https://www.digikey	RJ45 Jack for ethernet cable
3	1.2.2.7.0	| |--- Ferrite Bead	1	$	0.10	$	0.10	Digikey	https://www.digikey	FERRITE BEAD 30 OHM 0805 1LN
3	1.2.2.8.0	| |--- 0.1 µF Capacitor	2	$	0.15	$	0.30	Digikey	https://www.digikey	CAP CER 0.1UF 25V X7R 0805
3	1.2.2.9.0	| |--- 10 µF Capacitor	1	$	0.16	$	0.16	Digikey	https://www.digikey	CAP CER 10UF 10V X5R 0805
1	1.3.0.0.0	|------- Rear Sensor						------		
2	1.3.1.0.0	|--- Rear Housing Assy						------		
3	1.3.1.1.0	| |--- Housing	1					custom		3D Printed in PLA
3	1.3.1.2.0	| |--- Strap	1	$	16.50	$	16.50	Oneup	https://www.oneupc	14" stretch polyurethane strap with tail clip.
3	1.3.1.3.0	| |--- M3X0.5 Brass Inserts	4	$	0.50	$	2.00	McMaster	https://www.mcmas	Brass Heat-Set Inserts for Plastic; M3 x 0.50 mm Thread Size, 5.700 mm Installed Length
3	1.3.1.4.0	| |--- M3X0.5X6 Screw	4	$	0.42	$	1.66	McMaster	https://www.mcmas	Stainless Steel Pan Head Screws; with External-Tooth Washer, M3 x 0.5 mm Thread, 6 mm Long
3	1.3.1.5.0	| |--- 1/4" Washer	4	$	0.10	$	0.40	McMaster	https://www.mcmas	316 Stainless Steel Washer; for 1/4" Screw Size, 0.281" ID, 0.625" OD
2	1.3.2.0.0	|--- Rear PCB Assy								
3	1.3.2.1.0	|--- Accelerometer (ADXL375)	1	$	11.24	$	11.24	Arrow	https://www.arrow.c	Accelerometer Triple ±200g 2.5V/3.3V 14-Pin LGA T/R
3	1.3.2.2.0	|--- PCB	1	$	34.00	$	34.00	JLC PCB		Custom printed circuit board for front sensor
3	1.3.2.3.0	|--- OR Gate	1	$	0.29	$	0.29	Digikey	https://www.digikey	IC GATE OR 1CH 2-INP SC70-5
3	1.3.2.4.0	|--- RJ45 Jack	1	$	1.60	$	1.60	Digikey	https://www.digikey	RJ45 Jack for ethernet cable
3	1.3.2.5.0	|--- Ferrite Bead	1	$	0.10	$	0.10	Digikey	https://www.digikey	FERRITE BEAD 30 OHM 0805 1LN
3	1.3.2.6.0	|--- 0.1 µF Capacitor	2	$	0.15	$	0.30	Digikey	https://www.digikey	CAP CER 0.1UF 25V X7R 0805
3	1.3.2.7.0	|--- 10 µF Capacitor	1	$	0.16	$	0.16	Digikey	https://www.digikey	CAP CER 10UF 10V X5R 0805
1	2.0.0.0.0	Solder Stencil	1	$	28.00	$	28.00	JLC PCB	(CUSTOM)	
	Total Parts		**177.01**	**$**	**438.52**	**$**	**489.50**			

J – Total Expenditures

Team F11 MTB DAQ Senior Project Budget

Item Description	Quantity	Vendor	Vendor Part No.	F11 Part No.	Cost	S&H+ Tax	Purchase Method	Account	Date Purchased	Location
Al. Enclosure	2	Digi-Key	HM890-ND	1.0.0.0.0	$ 41.32	$ 8.61	Reimbursement	RMS	4/21/2022	Bonderson
Jumper	2	Digi-Key	732-2678-ND	1.1.8.36.0	$ 0.62					
Switch Rocker	2	Digi-Key	EG1526-ND	1.1.2.0.0	$ 3.92					
Wire Sleeve	8	Digi-Key	2-170823-8-SI-ND	1.1.4.0.0	$ 1.68					
Conn. Receptor	8	Digi-Key	A110952CT-ND	1.1.3.0.0	$ 1.68					
Rbn Cable	2	Digi-Key	SAM8899-ND	1.1.5.0.0	$ 32.30					
Conn. Tab	4	Digi-Key	WM14275CT-ND	1.1.8.17.0	$ 0.52					
Conn. Mod Jack	10	Digi-Key	RJCSE538001CT-ND	1.1.8.18.0	$ 16.31					
Conn. Head 2pos	4	Digi-Key	3-292173-2-SI-ND	1.1.8.37.0	$ 3.72					
1 uF Capacitor	12	Digi-Key	399-C0805C105K4RAC7800CT-ND	1.1.8.8.0	$ 1.25					
4.7 uF Capacitor	10	Digi-Key	311-3422-1-ND	1.1.8.10.0	$ 1.18					
2 pF Capacitor	10	Digi-Key	399-8701-1-ND	1.1.8.12.0	$ 10.21					
24 pF Capacitor	10	Digi-Key	399-8758-1-ND	1.1.8.13.0	$ 11.78					
2.2 uF Capacitor	10	Digi-Key	587-3421-1-ND	1.1.8.9.0	$ 1.49					
10 uF Capacitor	16	Digi-Key	399-11939-1-ND	1.1.8.11.0	$ 2.91					
Schottkey Diode	4	Digi-Key	1727-5191-1-ND	1.1.8.15.0	$ 1.76					
USB Mini B RCPT	3	Digi-Key	WM5461CT-ND	1.1.8.16.0	$ 4.26					
Conn. Head 3pos	4	Digi-Key	732-5316-ND	1.1.8.38.0	$ 0.52					
Ferrite Bead 120	8	Digi-Key	732-1613-1-ND	1.1.8.6.0	$ 1.76					
Blue LED	4	Digi-Key	732-4982-1-ND	1.1.8.19.0	$ 0.72					
Green LED	4	Digi-Key	732-4986-1-ND	1.1.8.20.0	$ 0.72					
Red LED	4	Digi-Key	732-4985-1-ND	1.1.8.21.0	$ 0.72					
Yellow LED	4	Digi-Key	1516-1088-1-ND	1.1.8.22.0	$ 1.28					
320 mAh Battery	10	Digi-Key	1908-LP402535JU+PCM+2WIRES50MM-ND	1.1.8.3.1	$ 86.63					
100k Resistor	14	Digi-Key	A126415CT-ND	1.1.8.24.0	$ 1.72					
2k Resistor	10	Digi-Key	A110572CT-ND	1.1.8.39.0	$ 4.74					
4.7k Resistor	14	Digi-Key	A129757CT-ND	1.1.8.26.0	$ 0.81					
560 Resistor	16	Digi-Key	A126376CT-ND	1.1.8.27.0	$ 1.18					
22 Resistor	10	Digi-Key	A126352CT-ND	1.1.8.28.0	$ 0.56					
Tactile Switch	2	Digi-Key	EG4375CT-ND	1.1.8.29.0	$ 0.95					
MicroSD Conn	2	Digi-Key	WM24066CT-ND	1.1.8.30.0	$ 7.32					
12 MHz Crystal	2	Digi-Key	XC1289CT-ND	1.1.8.34.0	$ 0.90					
Yellow Indic.	2	Digi-Key	492-1576-ND	1.1.9.4.0	$ 5.88					
Green Indic.	2	Digi-Key	492-1571-ND	1.1.9.5.0	$ 5.88					
44.2 Resistor	6	Digi-Key	A110413CT-ND	1.1.8.40.0	$ 3.84					
80.6 Resistor	4	Digi-Key	A110438CT-ND	1.1.8.41.0	$ 2.56					
Switch Button	2	Digi-Key	EG5385-ND	1.1.9.11.0	$ 19.78					
0.1 uF Capacitor	18	Digi-Key	399-C0805C104K3RAC7800CT-ND	1.2.2.8.0	$ 1.64					
10 uF Capacitor	18	Digi-Key	1276-6456-1-ND	1.2.2.9.0	$ 1.58					
Accelerometer	5	Digi-Key	505-ADXL375BCCZ-ND	1.3.2.1.0	$ 59.45					
Gyroscope	1	Digi-Key	1428-1001-1-ND	1.1.8.2.0	$ 9.49					
IC Gate	5	Digi-Key	296-SN74AHC1G32DCK3CT-ND	1.2.2.5.0	$ 2.20					
IC Volt Reg	2	Digi-Key	488-NCP5661MN33T2GCT-ND	1.1.8.32.0	$ 3.18					
10pos Conn.	4	Digi-Key	1175-2649-1-ND	1.1.8.33.0	$ 3.48	$ 35.47	Reimbursement	RMS	4/19/2022	Bonderson
32.768 kHz Crystal	2	Digi-Key	2388-SWS8346D16-32.768K-ND	1.1.8.35.0	$ 3.96	$ 9.34	Reimbursement	RMS	4/19/2022	Bonderson
Gyroscope	1	Digi-Key	1428-1001-1-ND	1.1.8.2.0	$ 8.26					
Hall Effect Sensor	1	Digi-Key	620-1964-ND	1.2.2.2.0	$ 0.98					
320 mAh Battery	4	Digi-Key	1908-LP402535JU+PCM+2WIRES50MM-ND	1.1.8.3.1	$ 41.76	$ 12.60	Reimbursement	RMS	2/7/2022	Bonderson
Rubber Pad	1	Amazon	-	1.1.1.3.0	$ 8.89					
OneUP Strap	1	Amazon	-	1.1.1.4.0	$ 19.99	$ 2.53	Reimbursement	JMR	11/4/2021	Bonderson
Solder Stencil	4	JLC	73145000	2.0.0.0.0	$ 28.00					
PCB	25	JLC	Y9-12	1.1.8.2.0	$ 21.60	$ 86.39	Reimbursement	JMR	4/20/2022	Bonderson
Breakout Board	1	Amazon	-	1.1.8.5.0	$ 12.99	$ 1.14	Reimbursement	JMR	2/13/2022	Bonderson
MCU	2	Charlie Refvem	-	1.1.8.31.0	$ 30.00	$ -	Reimbursement	TJP	4/5/2022	Bonderson
Total					$ 542.83	$ 156.08				

Grand Total	**$ 698.91**