Self-Contained Analytical Skating Form Tracker

By

Charles Hu

Jonathan Wang

Qian Ma

Final Report for ECE 445, Senior Design, Spring 2017

TA: Yuchen He

3 May 2017

Project No. 38

Abstract

The Self-Contained Analytical Skating Form Tracker is designed to collect movement data of the user and provide feedback on his or her skating form. Our system involves both hardware and software design. The hardware component consists of two wearable devices for data collection and the software component is a specialized data analysis tool with a graphic user interface. The two devices are strapped onto each of the user's skate boot, where all the information will be integrated and stored into a microSD card. The software will then process the information and provide feedback through the GUI. This report will document the development process of our project, including design choices, calculations, requirements, verification and and future development plans.

Contents

1 Introduction

1.1 Motivation

As an essential part of figure skating and hockey, ice skating is not only the foundation for the professional players, but also a widely popular recreational sport for the public all over the world. The increased speed and reduced friction on ice often brings about rising levels of safety issues. Therefore, ice skating calls for extreme precision and intricate body position. However, most skaters did not receive formal training on their skating forms. Even for people with professional instructors, the traditional ways of coaching they receive relies heavily on visual inspection and depends on the instructor's experience. Thus, many skaters, including some professionals, are unaware of their bad habits in skating form, which might lead to more risks of injury and will overall waste energy.

Our goal is to design an affordable, easy-to-wear device that can help users increase efficiency in skating while reducing the chances of injury. This device can track the skating forms and provide feedback to the user. It will assist the process of evaluating the skater performance and identify some subtle, unwanted habits that are hard to notice through visual inspection. We aim to make this design small and straightforward to use, while being affordable so the general public can utilize its benefits and improve upon their skating forms.

1.2 Objectives

High-level requirements

- The device must be reusable, water resistant, and must be less than 1 pound
- The device must be able to capture the translational movement and rotational orientation of the users skate. The movement of the skate must be accurate within $\pm 0.1g$ and the orientation of the skate must be accurate within ± 7 degrees
- The device must provide graphical and numerical feedback on the users skating form, based on the collected data.

2 Design

The Skating Form Tracker consists of two wearable devices, one for each skate. The device for the right side is the master while the left side is the slave. Besides small user interface differences, the hardware for the master and the slave are almost identical. Each wearable device consists of a power unit, a control unit, and a sensing unit. Each unit is broken down into separate modules. These individual modules were tested and verified based on determined requirements.

The system also has a software component. Its purpose is to analyze the collected data and display the analysed outputs for the user to view.

2.1 Design Procedure

We derived a set of parameters that qualifies as proper skating form. These parameters are obtained through Laura Stamm's book, "Power Skating"[1] and through personal experience. There are many nuances that the skater needs to master in order to skate effectively, but for practicality we decided to only track a small portion of them. This device will be used only to track forward skating in a straight line. There are four parameters used to determine good form. The first parameter is that the skate of the push-off leg is at a 45 degree angle with the ice. The second parameter is that for each stride, the push-off leg must be fully extended for maximum propulsion and stride efficiency. The third parameter is that during the push-off stage of the leg, the majority of the force should be at the heel of the corresponding skate. When the leg becomes fully extended, the weight is transferred from the heel to toe area. Lastly, symmetry between each leg is analysed. This includes making sure that both skates are at the same angle with the ice during push-off, both legs reach the same distance during push-off, both legs transfer force from heel to toe at the same speed, and for each stride both legs open up at the hip in the same angle.

The Skating Form Tracker is meant to log the movement of the user's stride while skating and then analyze the data to see if the user fulfills the parameters described above. There have been other devices that were developed in the past to achieve similar purposes, however they were either not easily usable or easily accessible. For instance, at the University of Calgary, a team of researchers developed a system geared towards tracking forward skating[2]. However, their system had many components and would be too complex to use for the average consumer. At Bringham Young University customized a figure skate to track applied force during jumping and landing [3]. Their design would require the users to buy a set of customized skates, which would have been very expensive. We wanted to make our device as user friendly as possible and so we limited the number of sensors included in our device. The sensors that we do include should be effective enough in quantifying the user's skating such that an algorithm would be able to distinguish proper skating from improper skating. With these considerations in mind, we decided to use the combination of an accelerometer, a gyroscope and two force sensors. The accelerometer and gyroscope will be used to quantify the strides through tracking the skate's acceleration and its angle, thus being used to validate skating form parameter one, two, and four. The force sensor is used to validate parameter three and four. Based on our reference[2], we would need a low pass filter to deal with noise due to the vibration between the skate blade and the ice. Thus each force sensor signal is sent through a low pass filter before leaving the sensor unit.

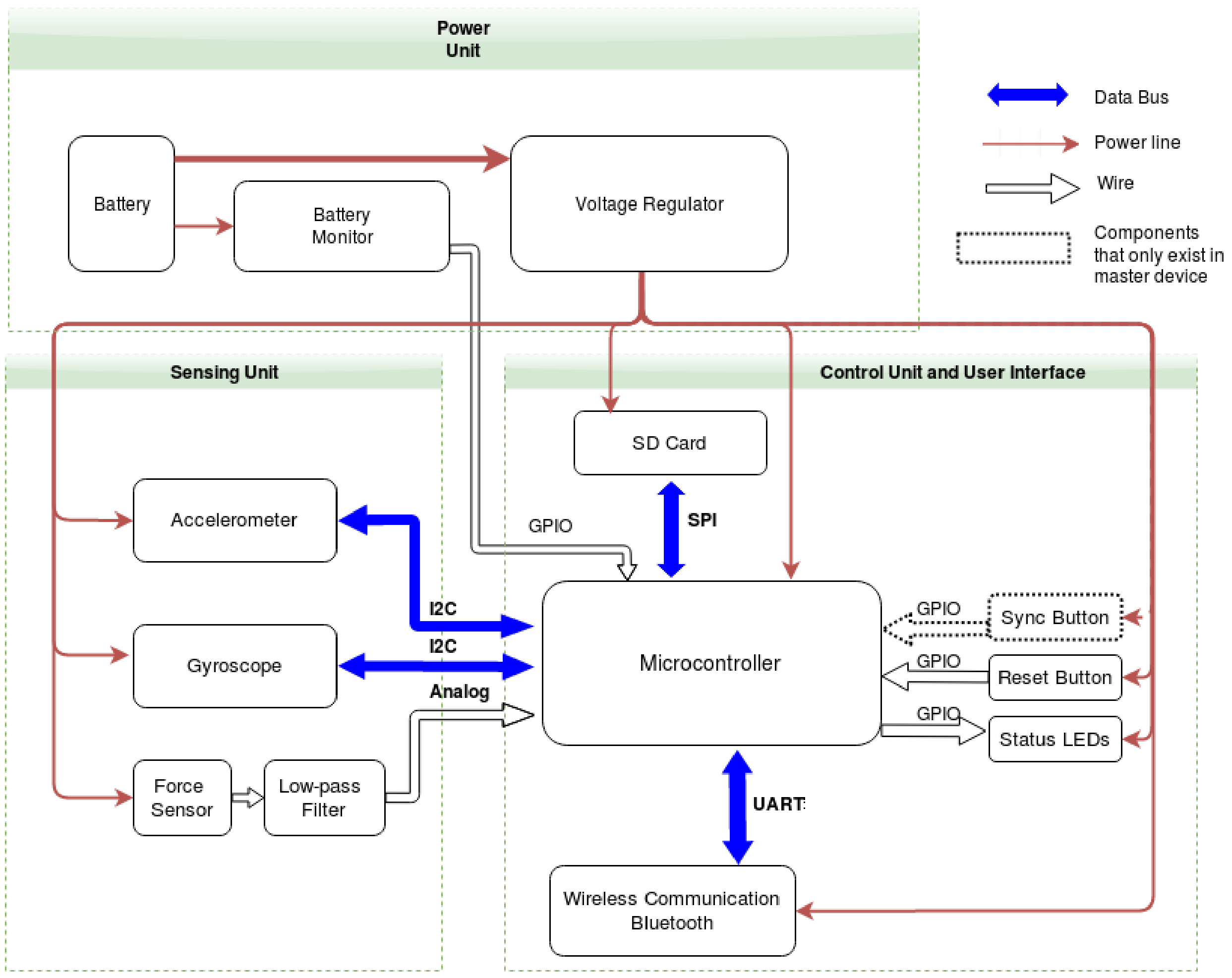


Figure 1: Hardware System Block Diagram

The Skating Form Tracker is a wearable device so the power unit has to run on a battery. The power unit also includes a voltage regulator to ensure that the rest of our device receives consistent voltage from the battery. Lastly, a low-voltage monitor is included in the power unit to keep track of the battery's capacity. Its serves to signal the microcontroller to shut down if the battery voltage falls below a certain level.

The control unit is essentially the hub for the sensor unit and directly deals with signals from the user interface. It needs to be able to compute signals, hold states, store data, and communicate with the control unit on the device of the other skate. To meet the above requirements we use a microcontroller to oversee all of these functions. A bluetooth module is used as the communication medium between the microcontrollers on each skate. That is the simplest method for short range communication. Data is stored in SD cards during a skating session. Each device has its own SD card reader to store data and to use as a debugging method for each device. Lastly, the control unit contains two tactile buttons and status LEDs. The buttons allow the user to control the device, while the LEDs give feedback to the user on the state of the device.

Once the data is collected from a skating session, it is loaded onto a computer to be analyzed for proper form. After the analytical algorithm is run, the results need to be displayed through a graphical user interface (GUI). For prototyping purposes, we decided to use MATLAB because of its computational robustness and its simple GUI functionality called GUIDE.

2.2 Design Details

Since the two tracking devices are nearly identical in hardware, we will focus our discussions below on the master device, and point out minute differences of the slave device only when relevant. The overall PCB schematic of the master device can be found in appendix D.

2.2.1 Power Unit

The power unit consists of a 9V Alkaline battery, a voltage regulator, and a low-voltage monitor.

Voltage Regulator
The voltage regulator regulates the 9V battery input voltage to the 3.3V operating voltage for the individual modules. As shown in figure 2, capacitor C1 is connected between the input pin and ground to improve the power supply rejection ratio (the ability to maintain output voltage as input voltage is varied). Capacitor C5 is connected between output and ground for noise reduction.

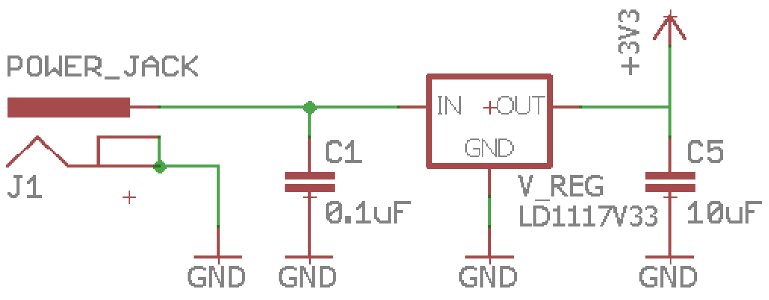


Figure 2: Power Supply Circuit Schematic

Low-voltage Monitor
Since the alkaline battery voltage quickly drops below its nominal value after constant use, it is necessary to have a low-voltage monitoring system to ensure that the voltage regulator is operating at proper range. Figure 3 shows a voltage divider circuit that will be probed by analog pin 25 of the MCU to periodically read the voltage level, and therefore the current capacity, of the 9V battery. Since keeping this circuit on at all times drains a significant amount of power, an STN3NF06L NMOS Transistor is attached to act as a switch. High signals are sent from pin 6 of the MCU to the gate terminal of the NMOS so that the circuit is only active during probing.

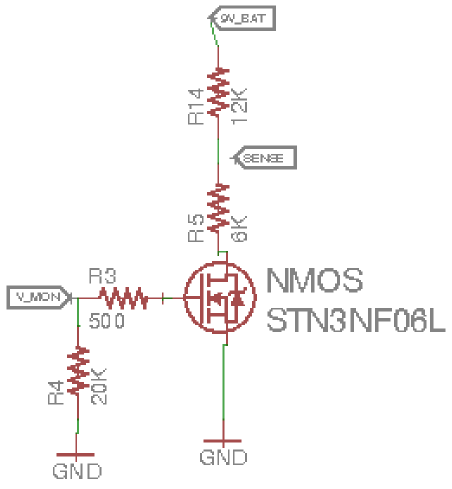


Figure 3: Low-voltage Monitor Schematic

2.2.2 Control Unit

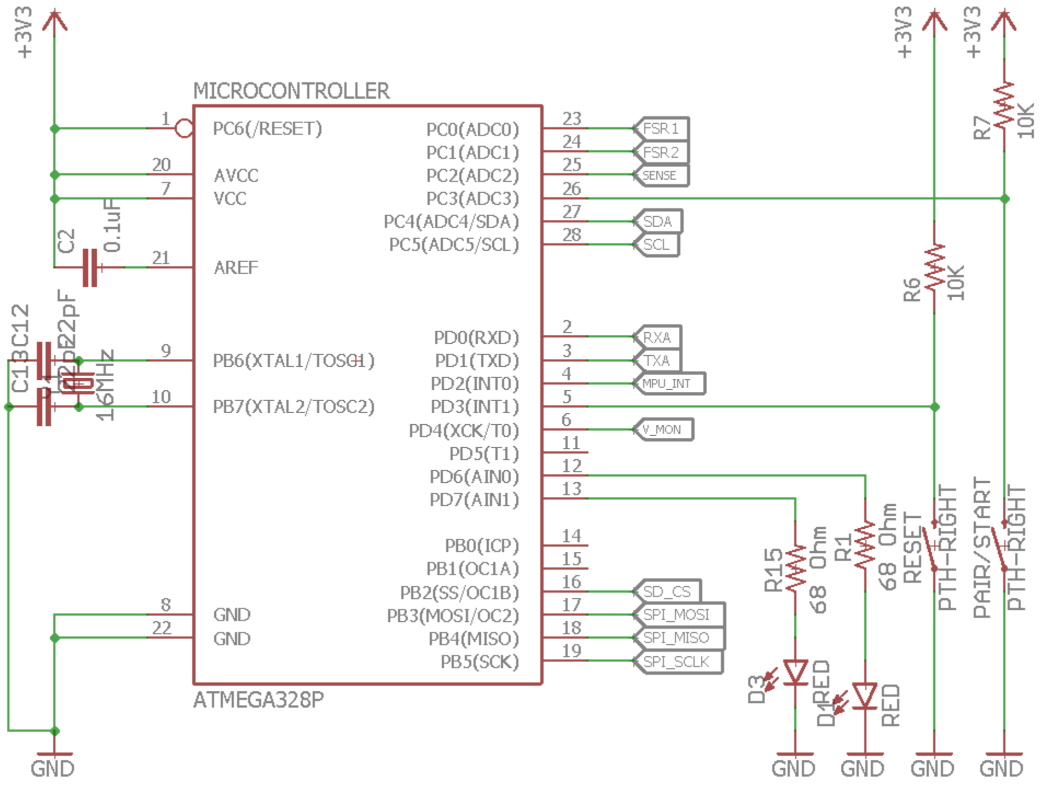


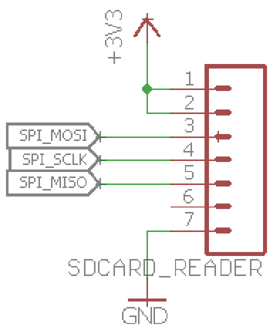
Figure 4: Microcontroller With Connections

Microcontroller Unit (MCU)

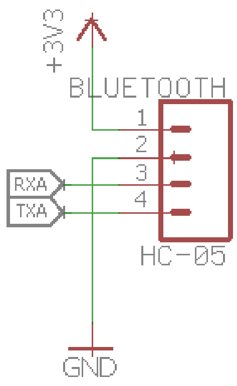
This unit sits at the center of our data logging device, and interfaces between the sensor unit, the microSD card, the bluetooth controller, and the buttons. We decided to use an ATmega328p AVR MCU for both the master and the slave, as it has enough pins as well as a well-tested database of drivers to drive these devices. Figure 4 shows the circuit connections external to the MCU

To facilitate data flow from the sensors to the microSD card at a desired refresh rate of 30Hz, the MCU must be able to poll the IMU through the I2C interface at 100 kbit/s, and be able to write to the microSD card at the maximum speed allowed by the card itself via SPI. The SPI bus speed is not a limit here since it is, as a synchronous signal, much faster than the microSD card can accept. The microSD card is formatted in the FAT file format, which has a unit write block of 512 bytes. Since the actual write time is slow, we mitigate this by using a 512-byte write buffer in the RAM of the MCU to ensure maximum write efficiency. To give an idea of the amount of data being logged, we poll 2 bytes per axis of the IMU, for 6 axes in total, as well a byte-sized value from each FSR. This amounts to a total of 14 bytes per 1/30 of a second. This data, converted to user readable strings, is approximately 70 bytes.

We require the slave device to transmit all its data to the master device at the end of data collection. Since there's 70 bytes $*$ 30 Hz $\approx$ 2100 bytes of data per second, the MCU must be able to transfer data through the bluetooth controller at a high baud rate. We chose the baud rate of 115200 as it allows a 10-minute session of data ($\approx$ 1260 kilobytes) to be transmitted in roughly 10 seconds. In order to achieve this desired baud rate, we externally connected the AVR to a 16MHz clock crystal, so as to reduce the error rate to 2.1% while maintaining a high throughput for the data transfer[4]. The high clock speed also ensures the MCU's ability to poll data at the desired overall rate of 30Hz from all the sensors combined.



(a) microSD Pinout Port



(b) Bluetooth Pinout Port

Figure 5: Pinout Ports

microSD Card Reader

A port is reserved on the PCB for the external microSD card reader. This is shown in fig. 5a The ports are connected to the 3.3V supply voltage, the ground, as well as the SCL and SDA pins needed for the SPI protocol. Since the SD card standard uses SPI for communications, this reader is merely a breakout board that connects the SPI signals, as well as VDD and GND, directly to the corresponding pins on the SD card. The driver used to interface with the SD card is an adapted version of Roland-Riegel's library [5], with the

UART library switched to Peter-Fleury's library [6] for compatibility with the rest of the system.

Bluetooth Controller

To facilitate wireless communications between the master-slave pair, we require a bare-bones, cost-efficient bluetooth device capable of transmitting at 115200 baud. For that reason we chose the HC-05 device, as it can function as both a bluetooth master and slave. Once the two devices are paired, they form a bridge that connects the UARTs of the master and slave tracking devices (note: not to be confused with the master and slave bluetooth controllers) so that they can communicate as if they are physically connected. The pinout connections reserved for this external chip is shown in fig. 5b.

LEDs and Tactile Buttons

There are two tactile push-buttons (one on the slave) and two LEDs to visually cue the user device's running state, and allow the user to operate it at ease. Red LEDs have been chosen because they use less power to operate, and are the least distracting to both the user and the skaters around him/her. Angled push-buttons are added so that the PCB can be easily secured in the housing. One button is used as a reset that puts the device back to its initial state, and the other is used to switch states/operating modes of the device. Since the slave tracking device is entirely controlled by the wireless input it receives from the master, it does not require a second button for state switching. Figure 4 shows the ports of the MCU these user IO components are connected to.

Hardware Finite-State Machine

Appendix C shows the detailed diagram and descriptions of the FSM for both the master and slave devices. The basic operation flow of the two devices is that upon powering on both devices, the master will be in an idle state, while the slave will be waiting for a pair. The user can trigger the master's pairing sequence with the main button, and upon successful pairing, the two devices will both in an idle state waiting for the data collection signal. The master's main button now functions as the state-transition button for both devices, changing between data collection, data transmission, and idle states.

The actual implementation if the FSM included a case statement within the main loop that, depending on the value of the case variable, will only execute the corresponding portion. We use the pin-change interrupt capabilities on the MCU to facilitate state changing. Upon pressing the button, the MCU will detect a voltage change on the designated button-pin. This executes a special interrupt ISR function, which also contains a case statement. This case statement checks the current state and other relevant variables, and assigns the next state accordingly. When the ISR function exits, the machine returns to its previous execution state, with its next state value changed.

2.2.3 Sensor Unit

Inertial Measurement Unit (IMU)

We decided to use an inertial measurement unit because it contains both an accelerometer and a gyroscope. The MPU6050 is chosen specifically due to its range flexibility and popularity with hobbyists. The accelerometer on the MPU6050 has multiple range configurations, from ±2g to ±16g. The gyroscope has the same flexibility, with ranges from ±250 degrees/sec to ±1000 degrees/sec. The selected range affects the

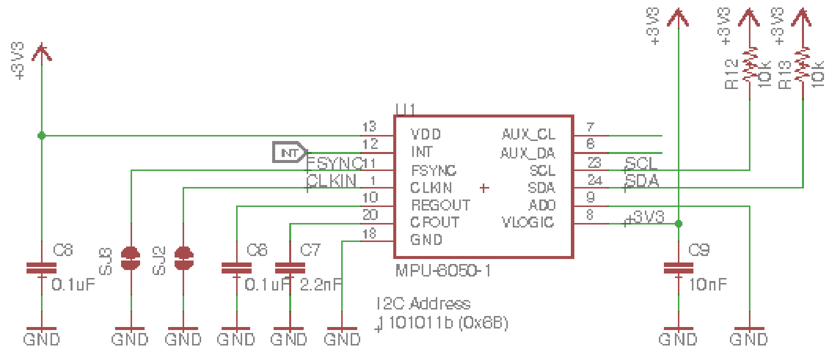


Figure 6: IMU Connection Schematic

precision of the data collected because numerically the same number of bits must evenly map values through the whole range. Thus this range flexibility allows us to fine tune the data collected to increase accuracy during development. Since the MPU6050 is widely used, there is an abundance of libraries that interfaces between the MPU6050 and our microcontroller. This would cut down on development time. The ranges that we used for the is ±4g for the accelerometer and ±250 degrees/sec for the gyroscope.

This IMU communicates via the I2C protocol, which our microcontroller can easily support through its serial ports. This allows for simple additions for future development. Also the MPU6050 meets our speed requirement of obtaining a set of six axes at 30 Hz. The accelerometer and gyroscope each has a programmable sampling rate ranging up to 8000 Hz for the gyroscope and 1000 Hz for the accelerometer. The driver used to drive the IMU is adapted from Davide Gironi's IMU library [7].

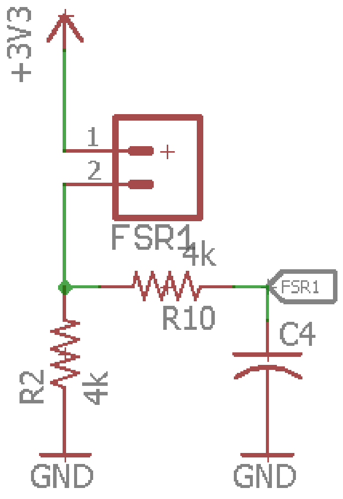


Figure 7: FSR and Low-pass Filter Schematic

Force Sensitive Resistor

The force sensitive resistors (FSR) serve as the force sensors in our design. The amplitude of the pressure

will be reflected by the resistance of the FSR. There will be two FSRs installed beneath the insole of each skate. FSR1 will be placed between the third and fourth toe mounds and FSR2 will be placed on the heel area. The sensors will each be connected in series with a 4 kOhm reference resistor. The output terminals are connected to analog pin 23 and 24 of the MCU. In reality, because the Analog pins of the microcontroller have a load impedance of about 10 kOhms, which is much larger than the typical resistance the FSRs will display, the finite load impedance is unlikely to cause significant error in measurement.
Another factor we took into consideration was the natural vibration during skating due to the uneven ice surface. The vibration normally causes high frequency noises of the analog data. which requires noise reduction steps. As a result, we decided to design a low-pass filter with a cut-off frequency at 20Hz. This cut-off frequency was chosen based on empirical data and past research.

2.2.4 GUI Design

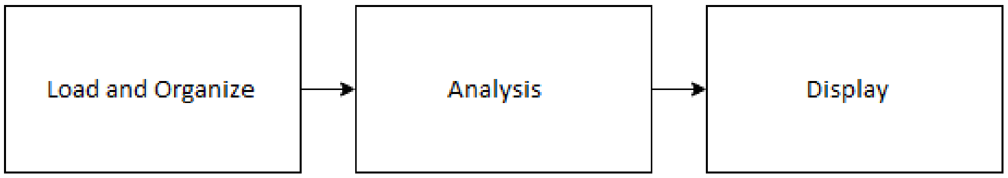


Figure 8: High Level Software Workflow

Once data is collected by wearable devices, the user would take the SD card from the master device and insert it into a computer. The data on the SD card is organized in the form of csv, which can be loaded into MATLAB directly.

Then through the GUI, there are two choices of analysis that can be done on the loaded data. The first analysis option is to integrate the gyroscope data to obtain the rotational displacement for each axis. Integration is done using the Kutta-Runge integrator, for which the equation is as follows [8]:

$$y(i) = y(i-1) + 1/6 * (x(i-3) + 2x(i-2) + 2x(i-1) + x(i)) \tag{1}$$

$y(i)$ is the integrated output at iteration i. $x(i)$ is the input of the gyroscope at iteration i.

The second analysis option runs the raw accelerometer, gyroscope, and force sensor data through an ideal digital low pass filter. Next, the gyroscope data is integrated to obtain the rotational displacement for each axis. The accelerometer data is then used to calculate the angle of rotation with the following functions [9]:

$$x_{rot} = \text{atan}\left(\frac{y_{accel}^2}{x_{accel}^2 + z_{accel}^2}\right) \tag{2}$$

$$y_{rot} = -\text{atan}\left(\frac{x_{accel}^2}{y_{accel}^2 + z_{accel}^2}\right) \tag{3}$$

x_{rot} and y_{rot} are the angles of rotation around the x axis and y axis respectively. x_{accel}, y_{accel}, z_{accel} are the data from the accelerometer along the x axis, y axis, and z axis respectively. These calculated rotations are used in the complimentary filter. A complimentary filter is a type of fusion filter used to compensate for the drift. The raw gyroscope data is inherently noisy, so integrating such noisy data would cause the calculated angle to experience drift. The complimentary filter essentially combines the accelerometer with the gyroscope data to counteract drift. The following is the representation of a complimentary filter [10]:

$$\angle out = (1 - \alpha) * (\angle gyro) + (\alpha) * (\angle accel) \tag{4}$$

$\angle gyro$ is the calculated angle by integrating the gyroscope data. $\angle accel$ is the calculated angle by using Equation (2) and Equation (3). $\angle out$ is the output angle from the complimentary filter.

The results of the analyses are displayed on GUI in the form of graphs. The user has the option to view raw data, the outputs of the first analysis, and the outputs of the second analysis. Data from the accelerometer, gyroscope, and force sensors are viewable in each case. screen shots of the GUI can be found in Appendix H.

3 Design Verification

3.1 Power Unit

3.1.1 Battery and Voltage Regulator

To verify the power supply, we tested the circuit in Figure 2 at different load currents using a variable load circuit[11] that are specified in the requirement table. After an hour of current draw, the battery voltage displayed a curve as shown in Figure 9(a). In the meantime, different DC loads were applied and the output of the voltage regulator is shown in Figure 9(b). The two figures indicate that despite the fluctuation of battery voltage, the voltage regulator output stayed within the defined threshold of $3.3V \pm 5\%$.

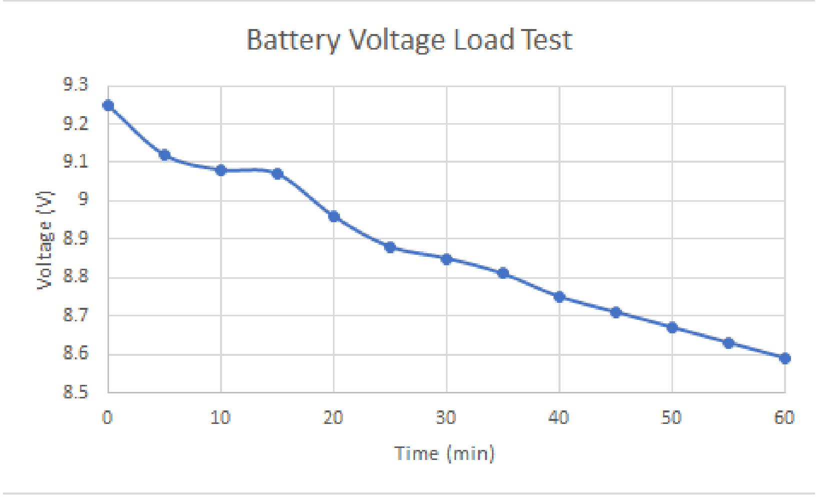


Figure 9: Battery Load Test Output

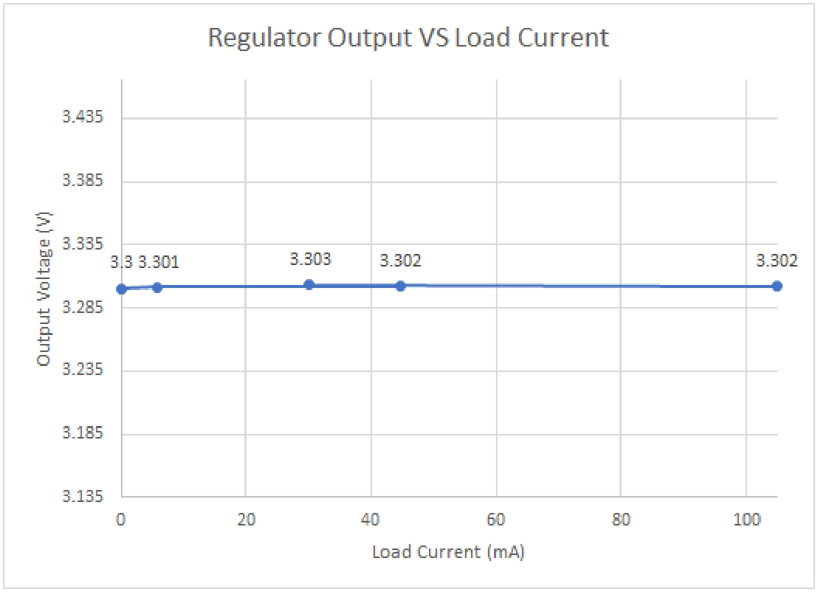


Figure 10: Voltage Regulator Output

3.1.2 Low-voltage Monitor

The voltage regulator has a dropout voltage of 1V, which implies that the battery needs to stay above 4.3V for the voltage regulator to maintain a reliable 3.3V output. We therefore designed a low-voltage monitor that keeps track of battery voltage and halts system operations if the value gets too low. Through our measurements, the drain and source terminal of the NMOS in Figure 3 has zero voltage drop, so the circuit serves as a simple voltage divider. Table 1 shows how the MCU accurately reports the set of battery voltages we applied to the circuit. We first calculate the expected voltage by using voltage divider rule, then multiply the raw calculation result with an empirical correction factor k = 1.116. The product of column two and three entries gives us the predicted voltage, which matches the inputs at very high accuracy.

Table 1: Low-voltage Monitor Verification Results

Applied Voltage	Calculation Result	Correction Factor k	Predicted Voltage	Percentage Error (%)
9.0	8.0421	1.1160	9.0009	0.0095
8.5	7.5957	1.1160	8.5013	0.0150
8.0	7.1493	1.1160	8.0017	0.0213
7.5	6.7030	1.1160	7.5021	0.0284
7.0	6.2567	1.1160	7.0025	0.0365
6.5	5.80787	1.1160	6.5001	0.0022
6.0	5.3614	1.1160	6.0006	0.0095

3.2 Control Unit

3.2.1 UART Baud Rate

For guaranteed fast data transfer, both the MCU and the bluetooth controller must be able to communicate at the 115200 baud. In our appendix B, we described the verification the UART capabilities of both by manually inspecting their output with an oscilloscope, as well as reading the data they send out with an FTDI chip (a specialized microchip that acts as a bridge between UART and the USB interface).

3.2.2 Data Polling Rate

To guarantee overall data polling at the specified rate of 30 Hz, we implemented a timer in the main data logging loop of the MCU FSM code. The hardware timer is implemented by using the built in overflow counter of the MCU. The counter is clocked to 500/1024 times the system clock, which ensures a steady 31.25 Hz rate. Due to hardware limitations, this was the closest we could get to 30 Hz.

3.2.3 microSD Read/Write

We had no specific requirements for the microSD read/write speed, except that it had to be able to record the data collected at the aforementioned 31.25 Hz without any drops. The verification for this was performed in parallel with the data polling procedures above. We let the timed loop described above run for a set period of time, and wrote the data into the microSD card. We then counted the number of data points collected, and verified that it matched the desired number of points determined by the set polling rate.

3.3 Sensor Unit

3.3.1 Accelerometer and Gyroscope

Following the verification procedure described in Appendix B, we were able to obtain results that met our desired requirements. The results are graphed and are shown below.

For each axis a simple digital low pass filtered is applied and the result is plotted along with the raw acceleration data. Looking at Figure 17, we can see that when the IMU is suspended in the air, as indicated by the red circle marked ”A”, the IMU outputs 1g of force. This is as expected because gravity is the only force acting on the IMU, all of which is along the x axis. Then, when the IMU is let go, it is essentially in free fall. When the IMU is in free fall, it experiences no forces so the output should be 0g. This can be shown in the yellow circle marked ”B”. The following spike in the data is due to the landing impact; the impact causes the crystal in the IMU to react with a huge spike in the output. For each axis test, four drops were done and each is evident in the graphs.

Based on the requirements for the gyroscope, only the x axis and y axis needed to be verified. Figure 20, Figure 21, and Figure 22 show the results for the x, y, and z axis tests respectively. As Figure 20 and Figure 21 indicate, the IMU is able to track the angular displacement from resting position to within the error margin specified. The green line is the lower bound of the allowable error. The IMU also accurately tracks the rotation after rotating it back the its initial resting position. The result for the z axis shows that for the first rotation, the IMU is able to accurately track the displacement, however upon returning to the resting position the graph shows that the z axis experiences drift.

3.3.2 Force Sensitive Resistor

We tested three types of actuation methods during the verification process. Method three as labeled in Figure 11 proved to be the most efficient and consistent throughout the test procedures. Therefore, all the FSR test results displayed in this paper were obtained using actuation method 3. As shown in Figure 12, most data points follow a power curve with a few outliers outside of the required range of $\pm 10\%$ of the theoretical value. We think this is acceptable because the errors most likely originate from inaccuracies in manual actuation. In addition, the purpose of the FSRs are to outline the trend of force shifting between heel and toe rather than to make precise measurements. Therefore, for the purpose of this project, the few data points that did not meet the requirement can be tolerated.

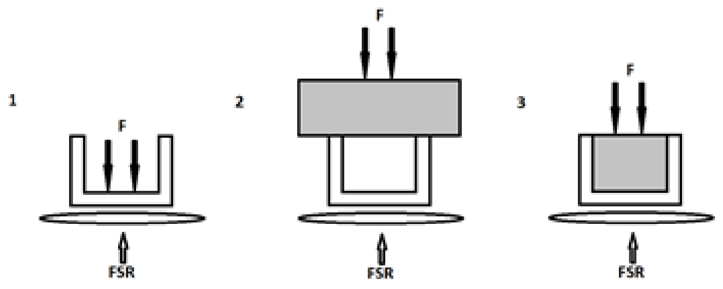


Figure 11: Actuation Methods

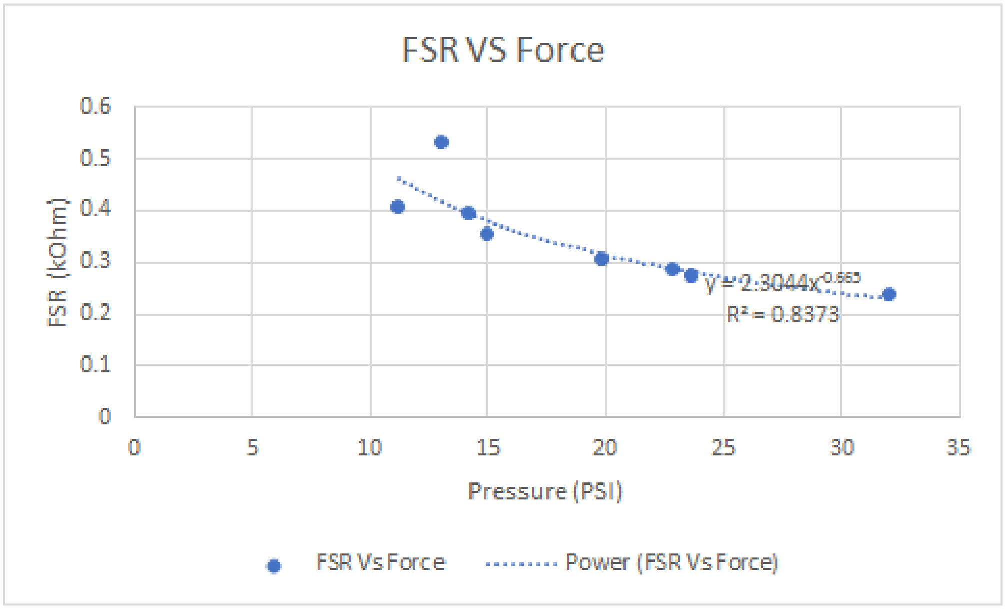


Figure 12: FSR Verification Results

3.3.3 Low Pass Filter

There are four low pass filters in our device system, two for the master device and two for the slave device. Each LPF filters the analog signal that is outputted from the FSR. In order to distinguish between the two filters on a device, we will refer to them based the capacitor name that is printed on the pcb board next to where the filter is laid out, c4 and c5.

Based on the Appendix B for the LPF, the acceptable error for the half power cutoff frequency is 10% from 20Hz. Thus the acceptable range for the cutoff frequency of the filters is between 18Hz and 20Hz. The method we used to verify the filters is to test each filter at 20 Hz and then calculate the gain from the input and the output. If the gain is less than $1/\sqrt{2}$ then we would test the filter at 18 Hz input to see if the gain is larger than $1/\sqrt{2}$. If this is the case then the real half power gain would fall between 20 Hz and 18 Hz, which would fall in between the allowable range. If at 20 Hz the gain is larger, then we would test at 22 Hz and check if the gain is less than $1/\sqrt{2}$. For all of the filters, the gain at 20 Hz was on the lesser side. The oscilloscope captures for this process can be seen in Appendix F.

4 Cost

4.1 Parts

Table 2: Parts Costs

Part	Manufacturer	Retail Cost ($)	Quantity	Actual Cost ($)
ATMega328P	Atmel	4.49	3	13.47
MPU6050	DFRobot	9.90	2	19.80
Force Sensitive Resistor	SparkFun	6.95	4	27.80
3.3V Voltage Regulator	STMicroelectronics	1.95	2	3.90
HC-05 Bluetooth Module	UXCell	7.87	2	15.74
microSD Card Reader	SparkFun	4.95	2	9.90
9V Alkaline Battery	Energizer	1.95	2	3.90
Total				**94.51**

4.2 Labor

Table 3: Labor Costs

Member	Hourly Rate ($)	Hours	Total Cost ($)
Charles Hu	30	200	6000
Jonathan Wang	30	200	6000
Qian Ma	30	200	6000
Total			**18000**

4.3 Grand Total

Grand Total = Parts + Labor Costs $*$ 2.5 = $45094.51

5 Conclusion

5.1 Accomplishments

The primary objective for this project is to build a portable device that can track the user's movement data and provide a certain level of feedback on the skating form. At this stage, we managed to build a self-contained device that is not only functional in data logging, but also lightweight and straightforward to use while being affordable.

In terms of hardware design, we successfully verified all our individual modules, as well as integrated the majority of our components. In terms of software design, We were able to create a fully functional GUI that portrayed the data collected, and provided visualisation for both raw and processed data. Although we encountered issues interpreting the data, we established a good foundation for future development.

5.2 Uncertainties

Unfortunately, although each Bluetooth module was tested and verified on the PC, we faced difficulties in fully integrating the Bluetooth communications. Our slave device was able to correctly transmit data and signals through Bluetooth, but our master device often picks up garbage data that interferes with both data collection and state transitions.

Another aspect of the project that did not fully meet expectations was signal processing. Extracting useful feedback based on the information collected proved to be a more daunting task than we anticipated, as the data from the IMU were often noisy, and given the time constraints, we were unable to devise of a satisfactory method to denoise it.

5.3 Safety and Ethical considerations

The primary safety concerns of our project include battery safety and mounting safety. The battery safety stems from potential leakage of the hazardous chemicals within the alkaline battery. This risk will only manifest itself when the battery is broken due to a great amount of impact. To prevent such an event from happening, we placed the battery inside a sturdy box and secured it to the bottom of the box. If a leakage happened, the caustic chemicals will be contained within the box and should be easy to clean up. The mounting problem happens when the mount fails and the device falls off the skate. To account for this issue, we fixed the device using laces and strapped them around the skate. Even if the device come loose, the device will still stay attached to the skate boot and will not trip the user.

In terms of ethical considerations, this project is purposed to help people maximize their efficiency while skating. We believe that the implementation of the device is strictly following the IEEE Code of Ethics, #9: " to avoid injuring others, their property, reputation, or employment by false or malicious action;"[12]. Throughout our design process, we took responsibility to address the potential safety issues and made every effort to minimize them. We believe that this is our commitment to the IEEE Code of Ethics, #1:" to accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment;"[12].

5.4 Future work

There remains many aspects on our projects that can be improved upon. The first step for our device would be to incorporate a magnetometer into our hardware design. The magnetometer would be used as a reference signal to stabilize the rotational reading around the z axis and fixes its drift issue. Next, we would further analyze the data received from our sensors to implement more filters. This would largely enhance the quality of the data by reducing noise that is currently present. Then we would be able use the extracted data to develop an algorithm that would perform analysis and give feedback about the user's skating form. Lastly, we would completely integrate the communication hierarchy between the master and the slave. This would make using the device easier as a whole.

References

[1] L. Stamm, *Laura Stamm's power skating*, 1st ed. Leisure Press, 1989.

[2] E. Buckeridge, M. C. LeVangie, B. Stetter, S. R. Nigg, and B. M. Nigg, "An On-Ice Measurement Approach to Analyse the Biomechanics of Ice Hockey Skating," *PLOS ONE*, vol. 10, no. 5, p. e0127324, 2015.

[3] J. Pearson, "Researchers at Brigham Young developed a smart skate," 2017. [Online]. Available: https://motherboard.vice.com/en_us/article/this-smart-skate-measures-how-hard-figure-skaters-land

[4] *ATMEL 8-BIT MICROCONTROLLER WITH 4/8/16/32KBYTES IN-SYSTEM PROGRAMMABLE FLASH DATASHEET*. Atmel, 2017. [Online]. Available: http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf

[5] R. Riegel, "MMC/SD/SDHC card library," 2017. [Online]. Available: http://www.roland-riegel.de/sd-reader/

[6] P. Fleury, "Interrupt UART library with receive/transmit circular buffers," 2017. [Online]. Available: http://homepage.hispeed.ch/peterfleury/avr-software.html

[7] D. Gironi, "Avr atmega mpu6050 gyroscope and accelerometer lib + processing," 2017. [Online]. Available: http://davidegironi.blogspot.com/2013/02/avr-atmega-mpu6050-gyroscope-and.html

[8] "mav-blog : gyroscope to roll, pitch and yaw," 2017. [Online]. Available: http://tom.pycke.be/mav/70/gyroscope-to-roll-pitch-and-yaw

[9] "gyroscopes and accelerometers on a chip," 2017. [Online]. Available: http://www.geekmomprojects.com/gyroscopes-and-accelerometers-on-a-chip/

[10] H. Vathsangam, "Complementary filter - my imu estimation experience," 2017. [Online]. Available: https://sites.google.com/site/myimuestimationexperience/filters/complementary-filter

[11] "Variable dummy load for power supply testing," 2017. [Online]. Available: http://www.deeptronic.com/electronic-circuit-design/variable-dummy-load-for-power-supply-testing/

[12] "IEEE Code of Ethics 2017," 2017. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html

Appendix A List of Acronyms

- **AVR**: Family of microcontrollers developed by Atmel, here used to refer to the ATmega328p
- **FSR**: Force Sensitive Resistor
- **FTDI**: Future Technology Devices International, a company that specializes in UART communications. The chip it makes is commonly referred to as the FTDI chip
- **GUI**: Graphical User Interface
- **I2C**: Inter-Intergrated Circuit bus
- **IMU**: Inertial Measurement Unit
- **LPF**: Low Pass Filter
- **MCU**: Microcontroller Unit
- **SPI**: Serial Peripheral Interface bus
- **UART**: Universal Asynchronous Receiver/Transmitter bus

Appendix B Requirement and Verification Table

Table 4: System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
9V Alkaline Battery		
1. Must supply voltage that is above the dropout range of the voltage regulator at a current draw of up to 100mA.	1. Design a variable load test circuit with the multimeter monitoring the voltage across the supply. 2. Tune the load resistance to adjust the load current to 100mA, measure the supply voltage drop and check if it is above the 4.3V margin(regulator voltage plus dropout voltage)	**Y**
2. The battery needs to support normal operation of the device for at least 1 hour.	1. Connect a load to the battery that draws consistent current. 2. Check if the voltage of the battery is above 5V after 1 hour.	**Y**
Voltage Regulator		
1. Output voltage must be regulated to +3.3V 5% at a range of current draw (up to 100mA) as required by the modules.	1. Connect the multimeter across the voltage regulator. 2. Check if the voltage drop between the input and output of the voltage regulator matches specification. 3. Use multimeter to track if the output voltage is stable. 4. Adjust the load current to 15mA and 30mA, respectively, repeat the above process.	**Y**
Status LED		

Continued on next page

Table 4 – continued from previous page

Requirement	Verification	Verification status (Y or N)
1. At forward current of 10mA, the LED should emit red light and visible at direct viewing angle.	1. Connect the LED in series with a 330 Ohm resistor. 2. Use the multimeter to measure current through the LED. 3. At 10mA, make sure the LED light is visible from direct viewing angle.	**Y**
Sync Button		
1. Buttons should be debounced and should indicate correct signal transition upon pressed.	1. Connect the button in series with a 330 Ohm resistor. 2. Supply 3.3 V. 3. Use the oscilloscope to monitor the voltage transition across the resistor, make sure the signal is debounced and correspond to correct digital value (digital High when pressed and Low when released).	**Y**
AVR Microcontroller		
1. Module is capable of running UART at a baud rate of 115200	1. Program the AVR to send 0x55 repetitively through the UART Tx pin. 2. Probe the Tx pin with the oscilloscope. 3. Verify the presence of a square wave with each pulse lasting 8.68s.	**Y**
2. Device is capable of writing text files to the microSD card that can be read from a PC.	1. Write a simple test program onto the AVR that uses the roland-riegel MMC/SD/SDHC card library to write 16-bit signed integers onto the microSD card. 2. Verify on the PC that the number are readable and consistent with what was written.	**Y**

Continued on next page

Table 4 – continued from previous page

Requirement	Verification	Verification status (Y or N)
3. Device is capable of monitoring the voltage of the battery when the MOSFET is ON. The reported voltage should be within 5%.	1. Get readings from the analog pin 25 that reports the voltage of the battery by using the formula0$V_{battery} = 3 * V_{adc}$. 2. Check if the voltage matches voltmeter readings.	**Y**
4. Device is capable of receiving data from the IMU via I2c at a minimum bitrate of 5580 bps (calculated requirement for maintaining the target 180 Hz sampling rate).	1. Orient the IMU such that the gyroscope has a constant non-zero reading. 2. Pull data from the IMU at 180 Hz for a second 3. Verify that the quantity and validity of data received meets expectations.	**Y**
Bluetooth Module		
1. Module is capable of receiving and transmitting over UART at a baud rate of 115200.	1. Program an Arduino to start serial communication at the baud rate of 115200. 2. Use the Arduino to program the Bluetooth module to operate at baud rate 115200. 3. Connect the Bluetooth module to an Arduino (Tx-¿Rx, Rx-¿Tx). 4. Short the Tx and Rx pins. 5. Pair computer with Bluetooth module. 6. Use a terminal program (eg. puTTY), connect to the correct serial line at baud rate 115200. 7. Verify that whats typed into the terminal is echoed back. If so, the loopback test is successful.	**Y**

Continued on next page

Table 4 – continued from previous page

Requirement	Verification	Verification status (Y or N)
2. Two of such modules are capable of performing at the speed mentioned above when placed 2 5% meters apart.	1. Repeat steps 1-6 from the verification above. 2. Pre-populate a string field of length 115200kb. 3. Move the HC-05 2 meters away from the testing PC. 4. Send the pre-populated string and check for accurate echo response.	**Y**
Accelerometer		
1. Acceleration output accurate up to 0.1 g.	1. Connect the accelerometer to microcontroller with test code on it. The test code will collect the acceleration data from the accelerometer. 2. Turn on the test circuitry and use one hand to hold the accelerometer about 6 inches above the other hand. Based on the accelerometer the x axis should be facing upward. 3. Drop the accelerometer from one hand to the other hand. Try to drop is so the accelerometer drops straight down and does not rotate during the fall. 4. Repeat step 2 and step 3 with the y axis and the z axis. 5. Port the collected data and graph it. Recommend using MATLAB but other scripting languages and environments could work too. 6. Plot the data and verify that when the accelerometer is held, the reading for the tested axis should be 1 g. Then the data should drop to 0 g when the accelerometer is falling. Both readings should be within the specified error margin.	**Y**

Continued on next page

Table 4 – continued from previous page

Requirement	Verification	Verification status (Y or N)
2. Samples at a rate of at least 180 Hz in order to give enough resolution to represent movement.	1. Connect the accelerometer to the microprocessor and load a test code that will collect and log the data from the accelerometer 2. Obtain a stopwatch to record the time data is collected. Collect data for 10 seconds. 3. Analyze the data collected to ensure that there are enough entries to meet the requirement of 180 Hz (for 10 seconds example there should be at least 1800 entries).	**Y**
Gyroscope		
1. Accurate up to 7 degrees in order to assure accuracy in data.	1. Connect the gyroscope to a microcontroller that has test code to collect data from the gyroscope. 2. Turn on the test circuitry with the gyroscope lying flat. 3. Rotate the x axis (roll) 90 degrees and then rotate back to initial position. 4. Repeat step 3 for the y axis (pitch). 5. Port the collected data and graph it. Recommend using MATLAB but other scripting languages and environments could work too. 6. Plot the data and verify that the axis being tested reaches 90 degrees or is within the error margin. The data should also reach 0 degrees when rotated back to the initial position. If the data collected is the raw angular velocity then, integrate it before plotting.	**Y**

Continued on next page

Table 4 – continued from previous page

Requirement	Verification	Verification status (Y or N)
2. Samples at a rate of at least 180 Hz in order to give enough resolution to represent movement.	1. Connect the gyroscope to the microprocessor and load a test code that will collect and log the data from the gyroscope 2. Obtain a stopwatch to record the time data is collected. Collect data for 20 seconds. 3. Analyze the data collected to ensure that there are enough entries to meet the requirement of 180 Hz (for 20 seconds example there should be at least 3600 entries).	**Y**
Force Sensitive Resistor and Low-pass Filter		
1. Pressure measurement range needs to be from 1psi to at least 6.409psi (Equivalent: force measurement range needs to be between 14-90N) 2. The measurement needs to be within 10% of the theoretical value at the two boundary values. *see derivation of the values used here in the calculation and simulation section	1. Place the FSR on a flat surface and connect the FSR in series with a 20 kOhm reference resistor R_{ref}. 2. Supply 3.3V voltage across the two resistors. 3. Place a 1.4kg weight on top of the FSR; 4. After the readings stabilize, measure the voltage drop across the reference resistor, V_{ref}. 5. Find V_{FSR} by using the relation $V_{FSR} = 3.3V - V_{ref}$ and then find the current across the components $I = V_{ref}/R_{ref}$. 6. Use the voltage and current found to calculate FSR resistance, compare with theoretical value and check if it is within 10% error margin. 7. If the value does not match the standards, change the weight until the value meets the requirement, record the adjusted value and check if it is among a reasonable range. 8. If the value matches the standards, move on to replace the 1.4kg weight with a 9.2kg weight and repeat the above steps.	**Y**

Continued on next page

Table 4 – continued from previous page

Requirement	Verification	Verification status (Y or N)
2. Measurement result needs to be accurate within 10% of the theoretical value.	1. Select 5-10 weight values between the two threshold values. 2. Repeat the same steps as specified in part 1 of FSR verification and record the corresponding measurements. 3. Use the values obtained to plot the Pressure vs Force curve, fit a trend line. 4. Check if the trendline matches theoretical curve as shown in Figure 4 and the points have less than 10% error.	**Y**
3. Must draw current less than 1uA at resting state.	1. Supply 3.3V voltage across the resistor. 2. Measure the current drawn from the supply using the multimeter.	**Y**
4. The low-pass filter must have -3dB frequency of 20Hz 10%.	1. Use the function generator to generate a sinusoidal input with amplitude of 50mV at frequency of 5Hz and slowly increase the frequency to 30Hz while using the oscilloscope to monitor the amplitude; 2. When the amplitude dropped to 35.34mV, verify that it is within 10% of 20Hz.	**Y**

Appendix C Hardware Finite-State Machine

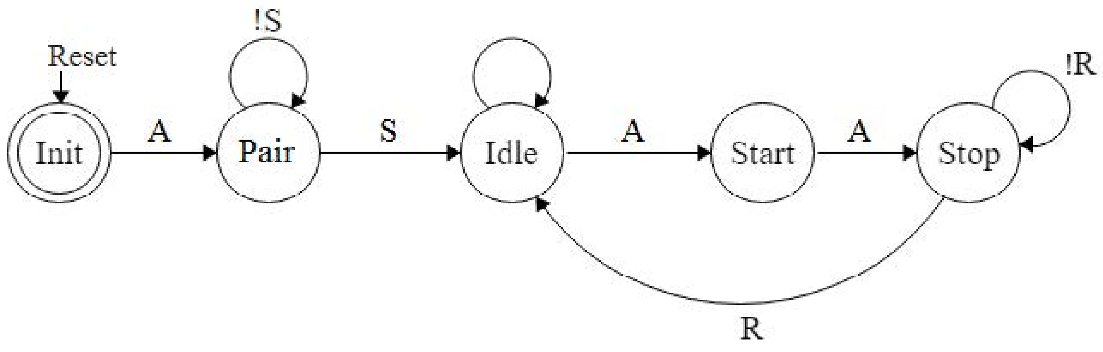


Figure 13: Master Device FSM Diagram

Input	Explanation
Reset	Triggered by pressing the Reset Button
A	Triggered by pressing the Sync Button
S	Bluetooth pairing process success
R	Bluetooth data transmission success

Table 5: Master FSM Input Explanation

State	Description
Init	Initial setup state when reset is pressed
Pair	Pairing state, master sends out pairing signal P to slave
Idle	Wait state in between Bluetooth pairing and data collection process, or after slave data transmission at Stop state is finished; Status LED2 on
Start	Data collection in progress; Status LED2 on
Stop	Initiate slave data transmission; Status LED2 off

Table 6: Master FSM State Description

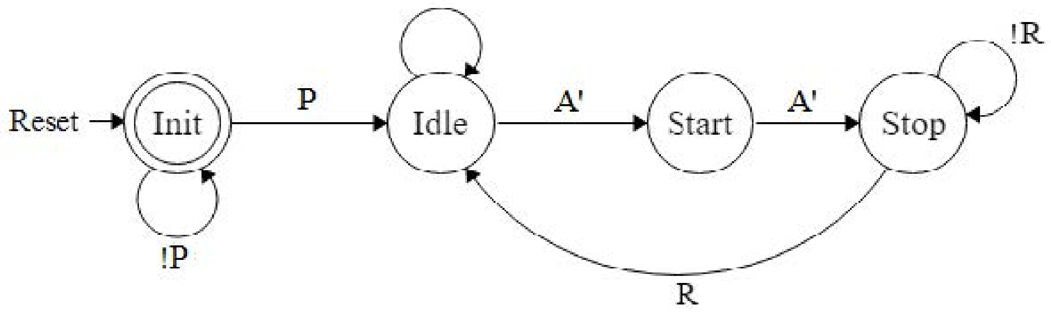


Figure 14: Slave Device FSM Diagram

Input	Explanation
Reset	Triggered by pressing the Reset Button
P	Pairing signal from master module
A'	Signal A received through Bluetooth
R	Bluetooth data transmission success

Table 7: Slave FSM Input Explanation

State	Description
Init	Initial setup and pairing state
Idle	Wait state in between Bluetooth pairing and data collection process, or after slave data transmission at Stop state is finished; Status LED2 on
Start	Data collection in progress; Status LED2 on
Stop	Initiate slave data transmission; Status LED2 off

Table 8: Slave FSM State Description

Appendix D Circuit Schematic and Layout of the Master PCB

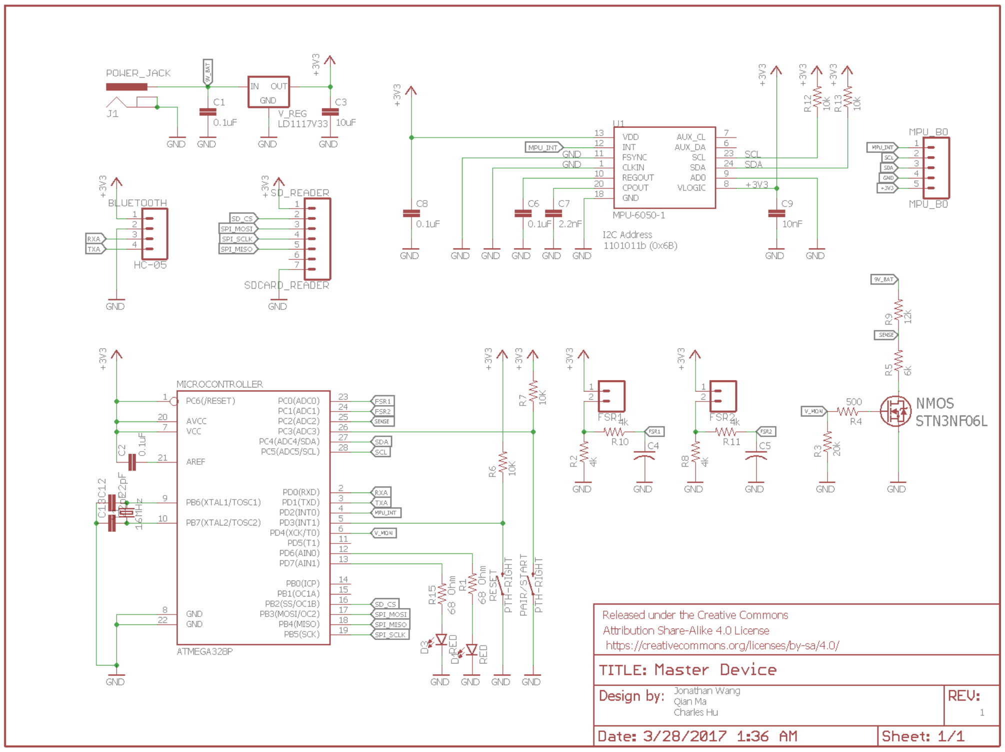


Figure 15: Overall Circuit Schematic

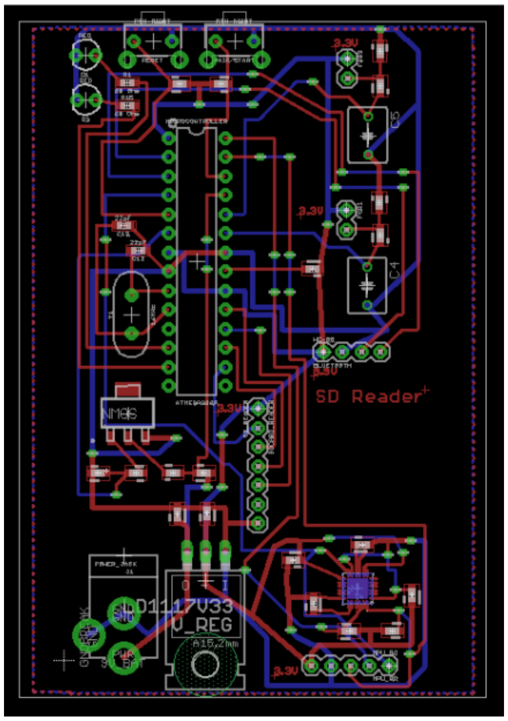


Figure 16: Overall PCB Layout

Appendix E   Accelerometer Verification Results

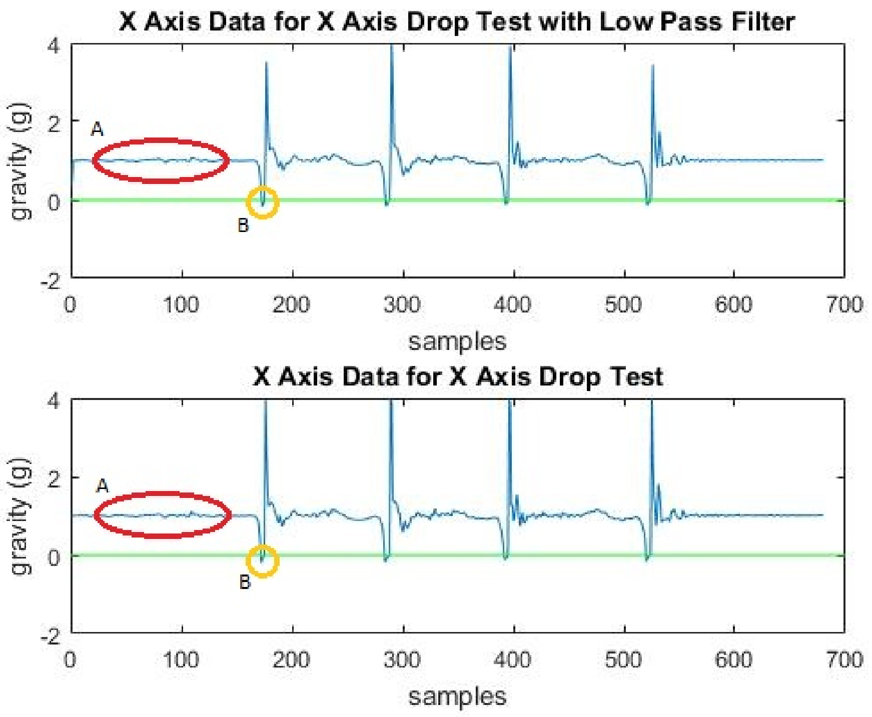


Figure 17: X Axis Accelerometer Result from Verification Procedure

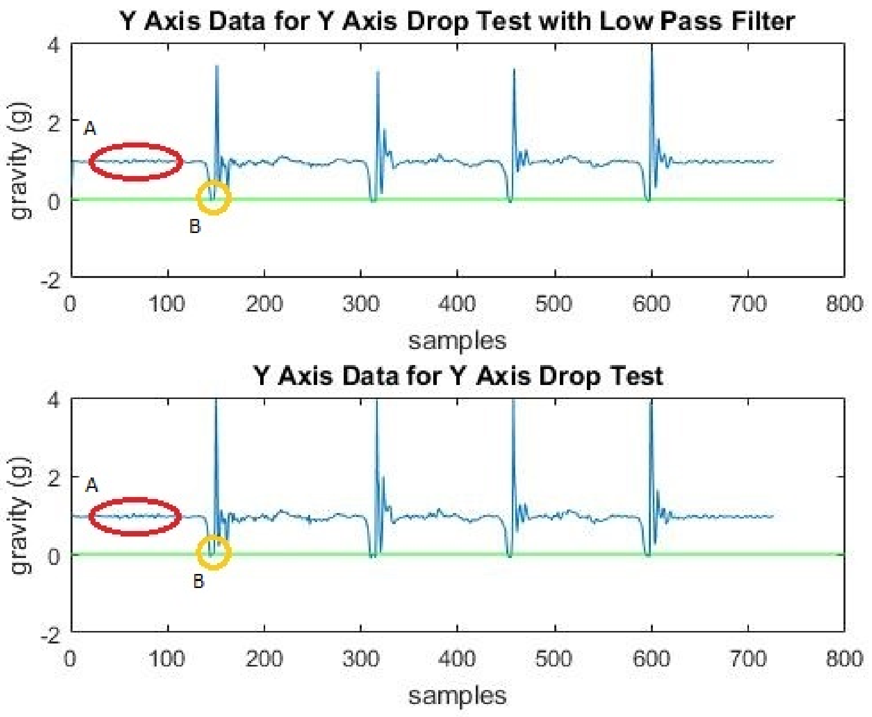


Figure 18: Y Axis Accelerometer Result from Verification Procedure

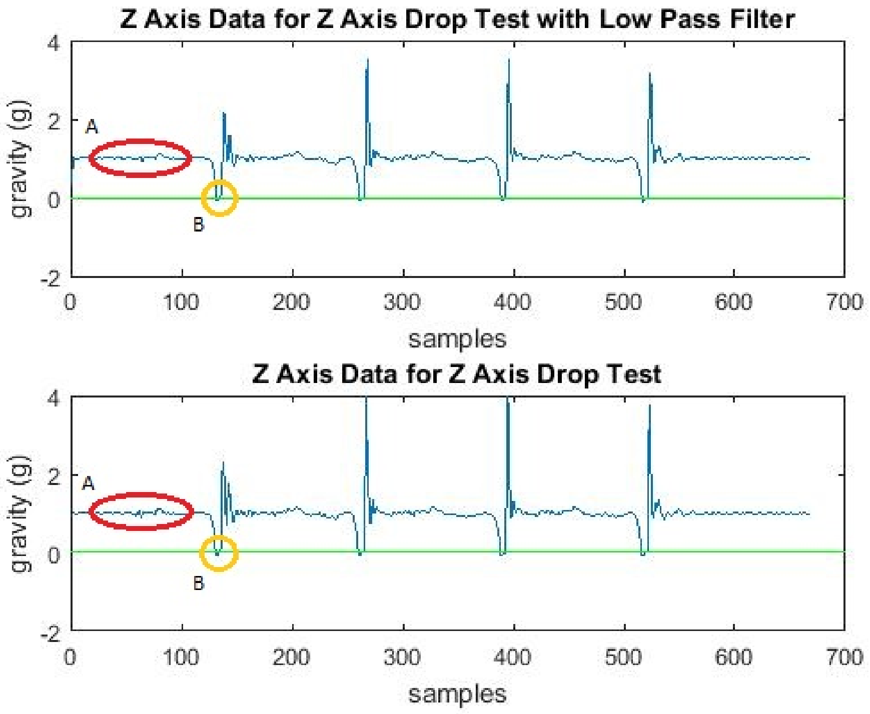


Figure 19: Z Axis Accelerometer Result from Verification Procedure

Appendix F Gyroscope Verification Results

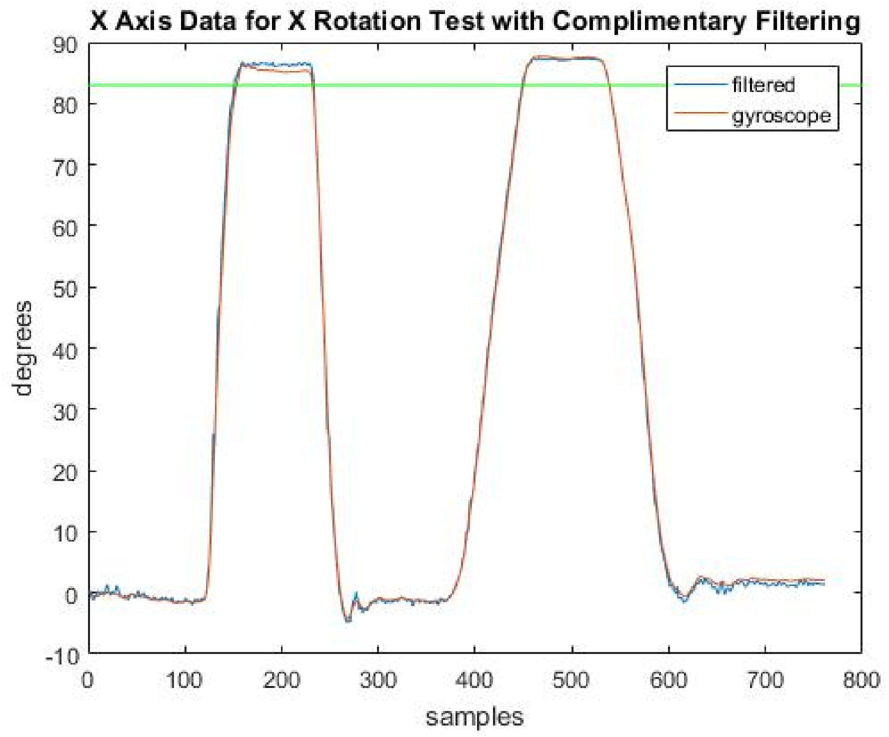


Figure 20: X Axis Gyroscope Result from Verification Procedure

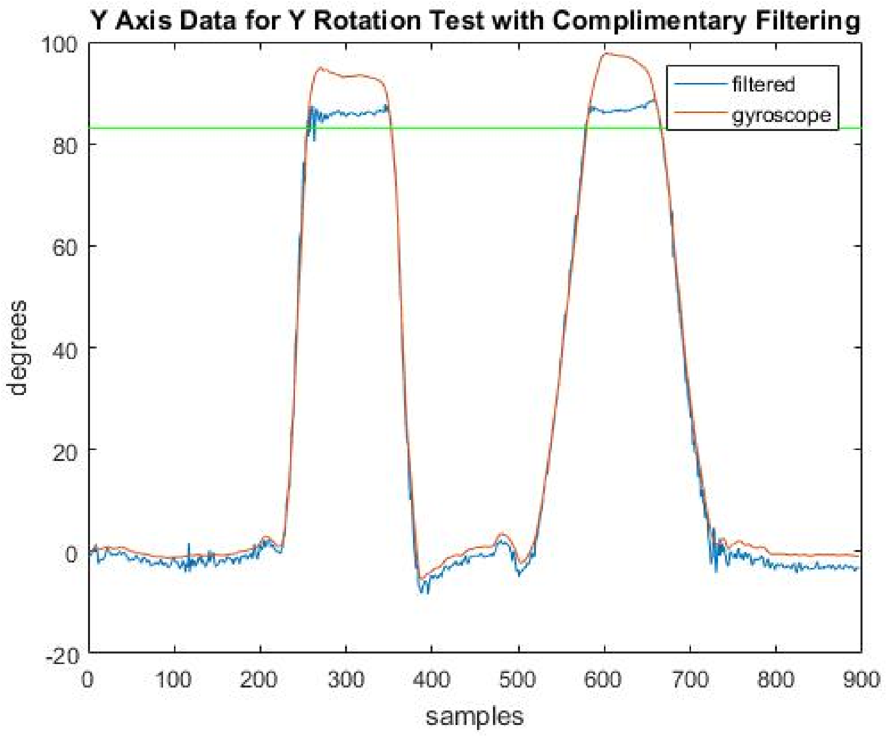


Figure 21: Y Axis Gyroscope Result from Verification Procedure

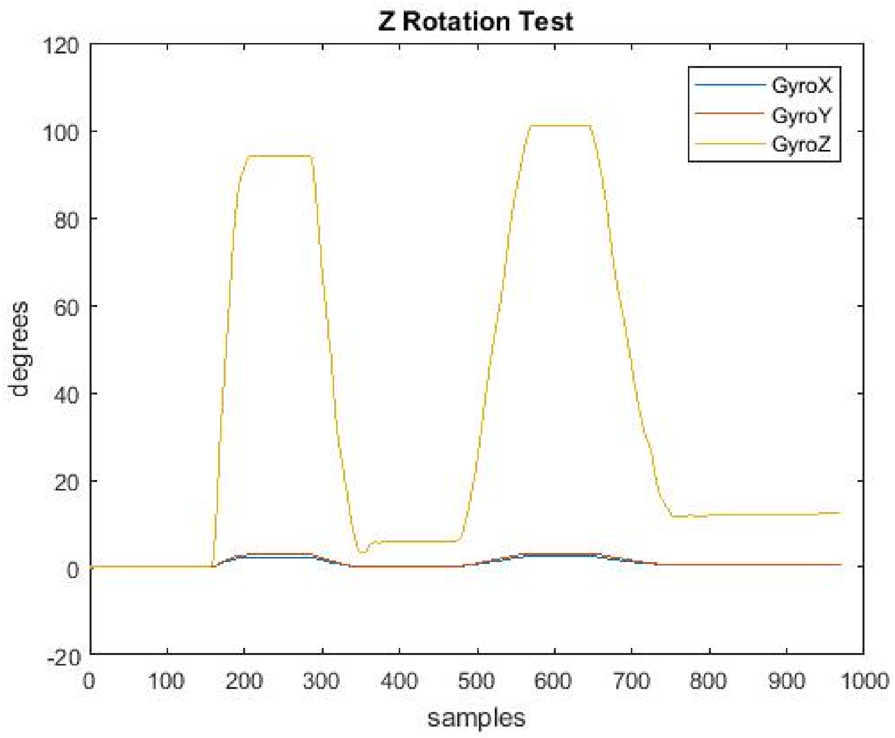


Figure 22: Z Axis Gyroscope Result from Verification Procedure

Appendix G Low Pass Filter Verification Results

G.1 Master Device LPFs

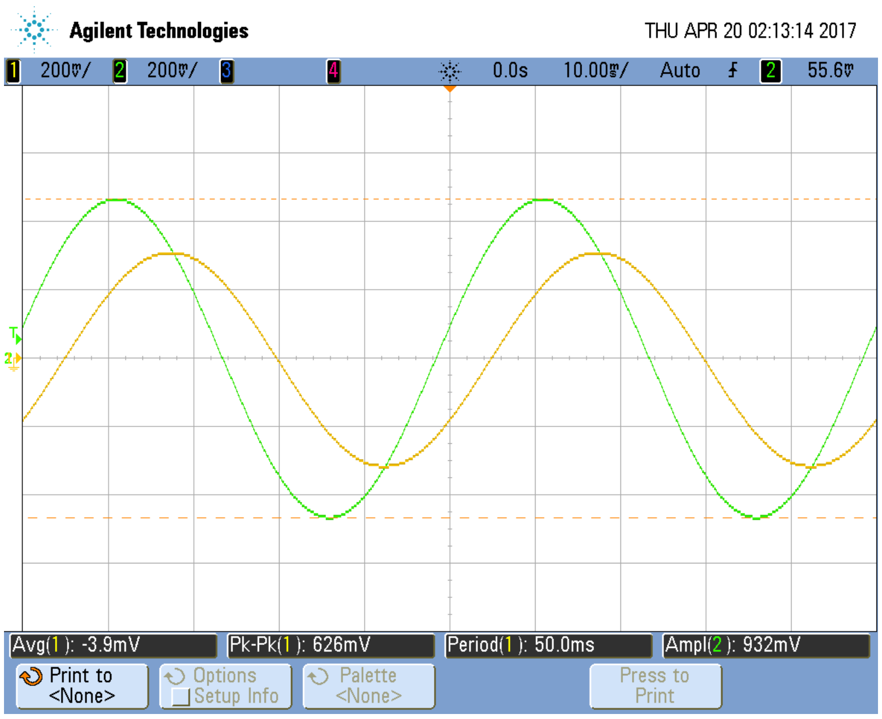


Figure 23: Filter for C4 of master PCB at 20 Hz (gain = .671)

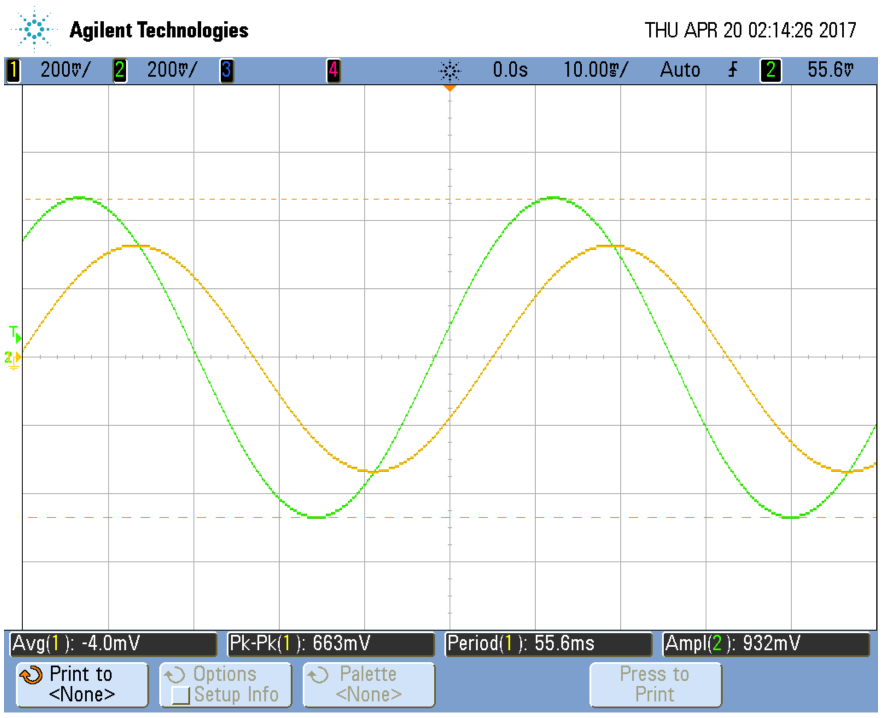


Figure 24: Filter for C4 of master PCB at 18 Hz (gain = .712)

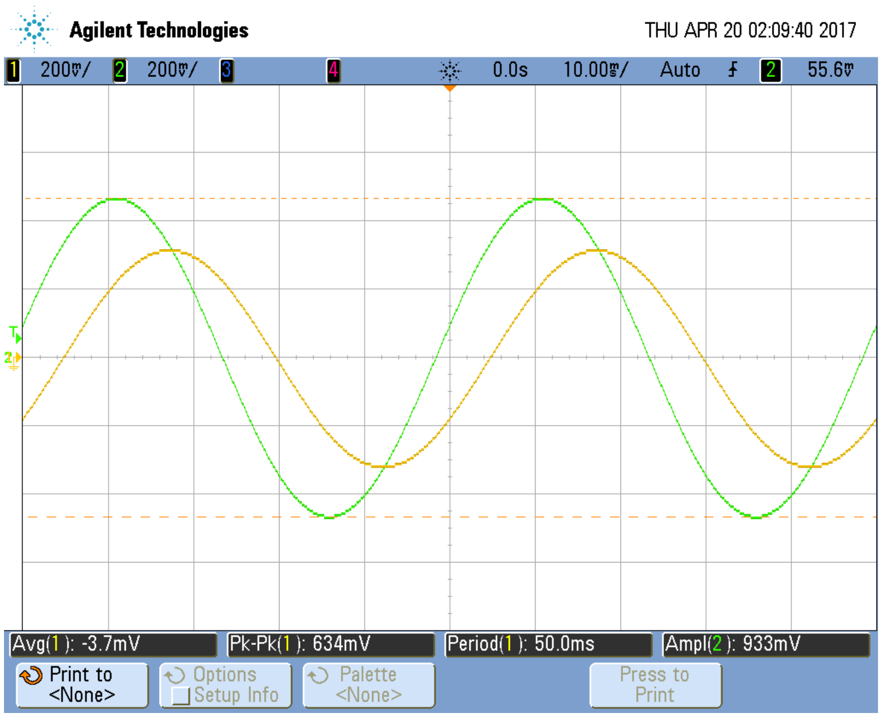


Figure 25: Filter for C5 of master PCB at 20 Hz (gain = .680)

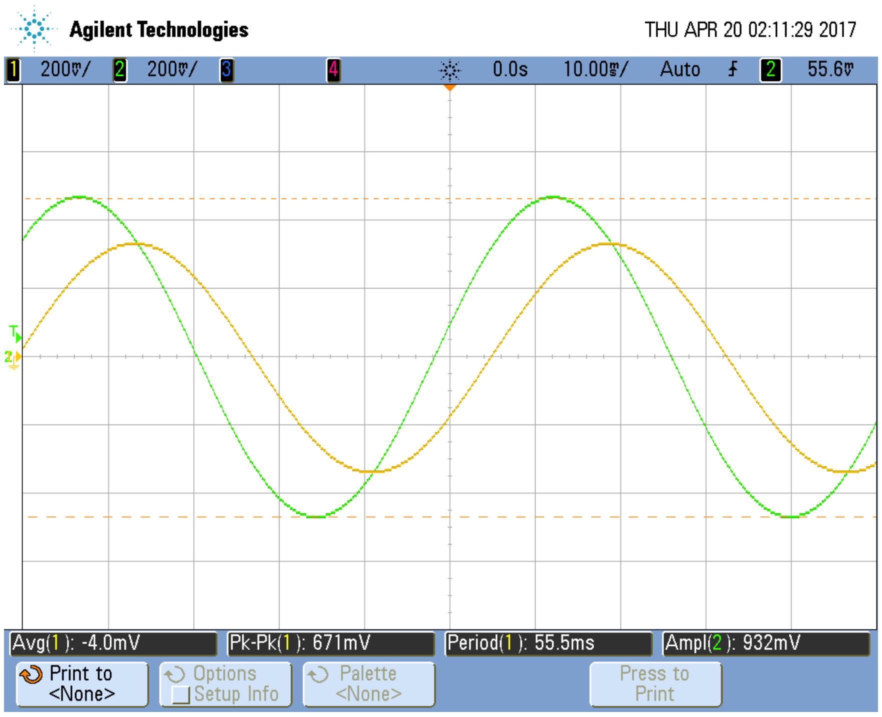
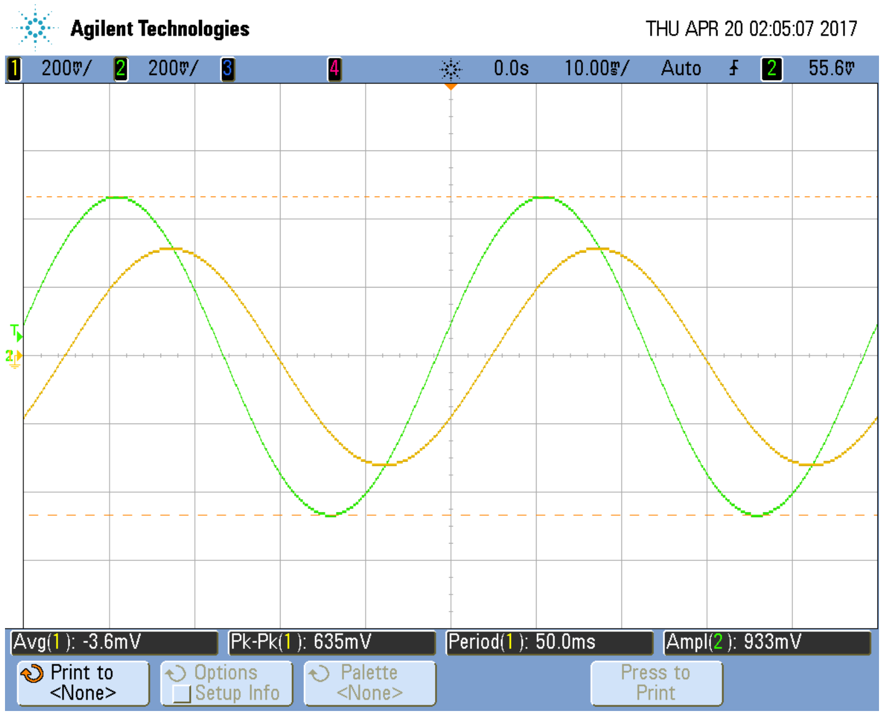


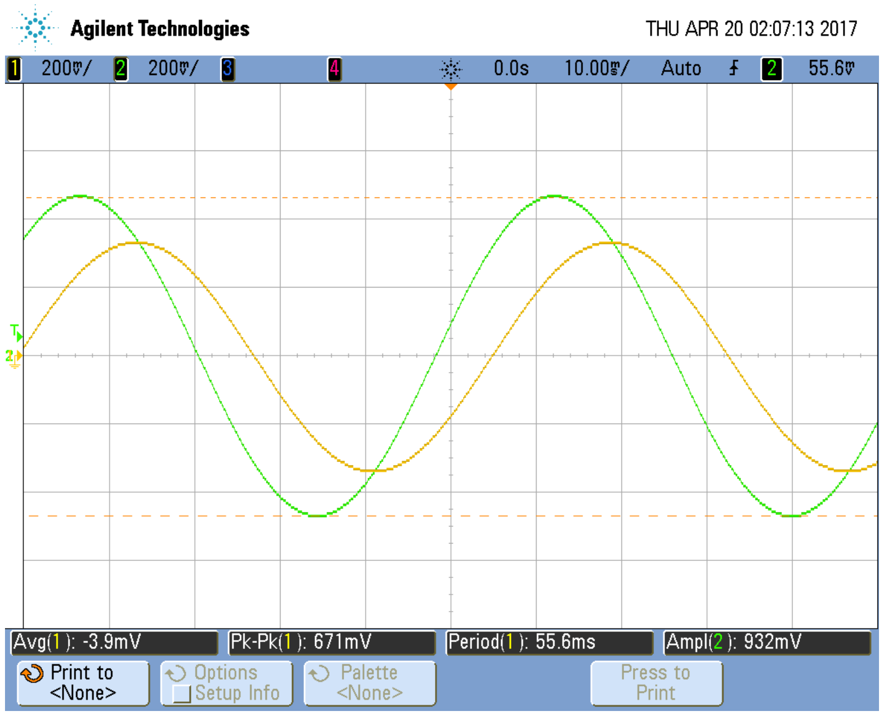
Figure 26: Filter for C5 of master PCB at 18 Hz (gain = .720)

G.2 Slave Device LPFs



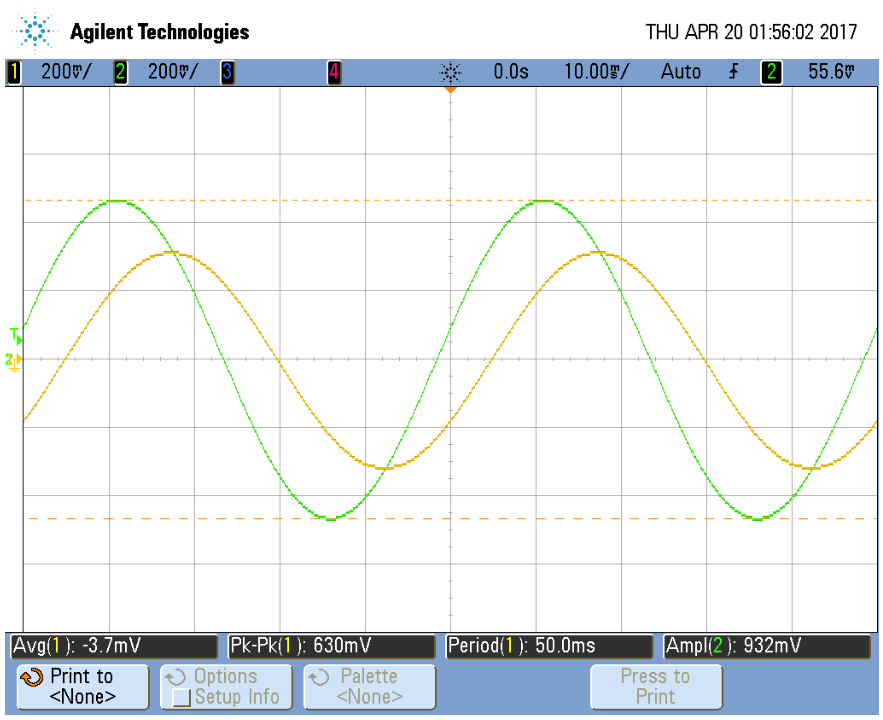
Avg(1): -3.6mV Pk-Pk(1): 635mV Period(1): 50.0ms Ampl(2): 933mV

Figure 27: Filter for C4 of slave PCB at 20 Hz (gain = .681)



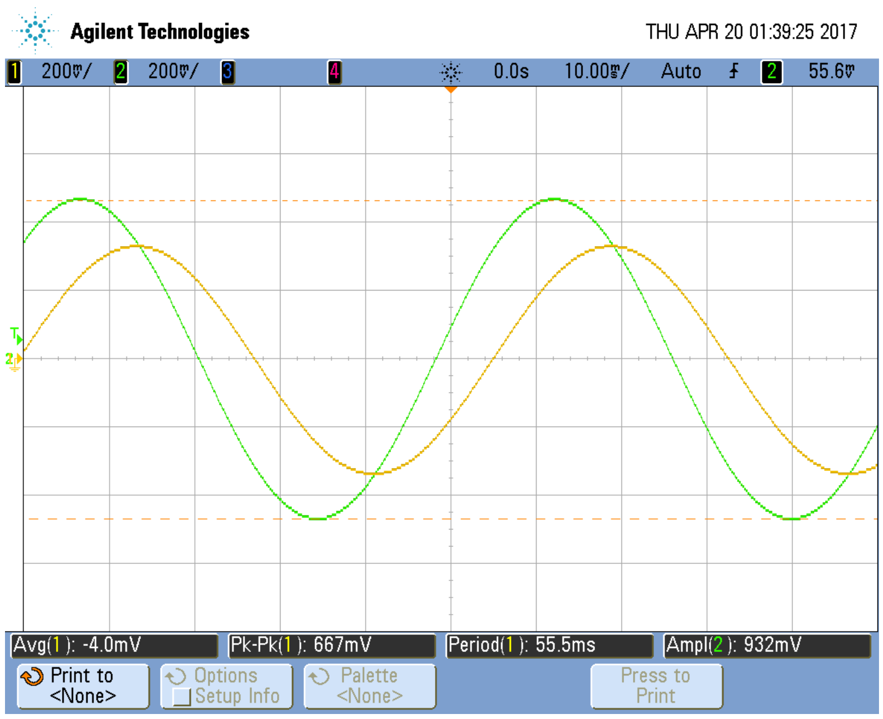
Avg(1): -3.9mV Pk-Pk(1): 671mV Period(1): 55.6ms Ampl(2): 932mV

Figure 28: Filter for C4 of slave PCB at 18 Hz (gain = .721)



Avg(1): -3.7mV Pk-Pk(1): 630mV Period(1): 50.0ms Ampl(2): 932mV

Figure 29: Filter for C5 of slave PCB at 20 Hz (gain = .681)



Avg(1): -4.0mV Pk-Pk(1): 667mV Period(1): 55.5ms Ampl(2): 932mV

Figure 30: Filter for C5 of slave PCB at 18 Hz (gain = .717)

Appendix H Example GUI Screenshots

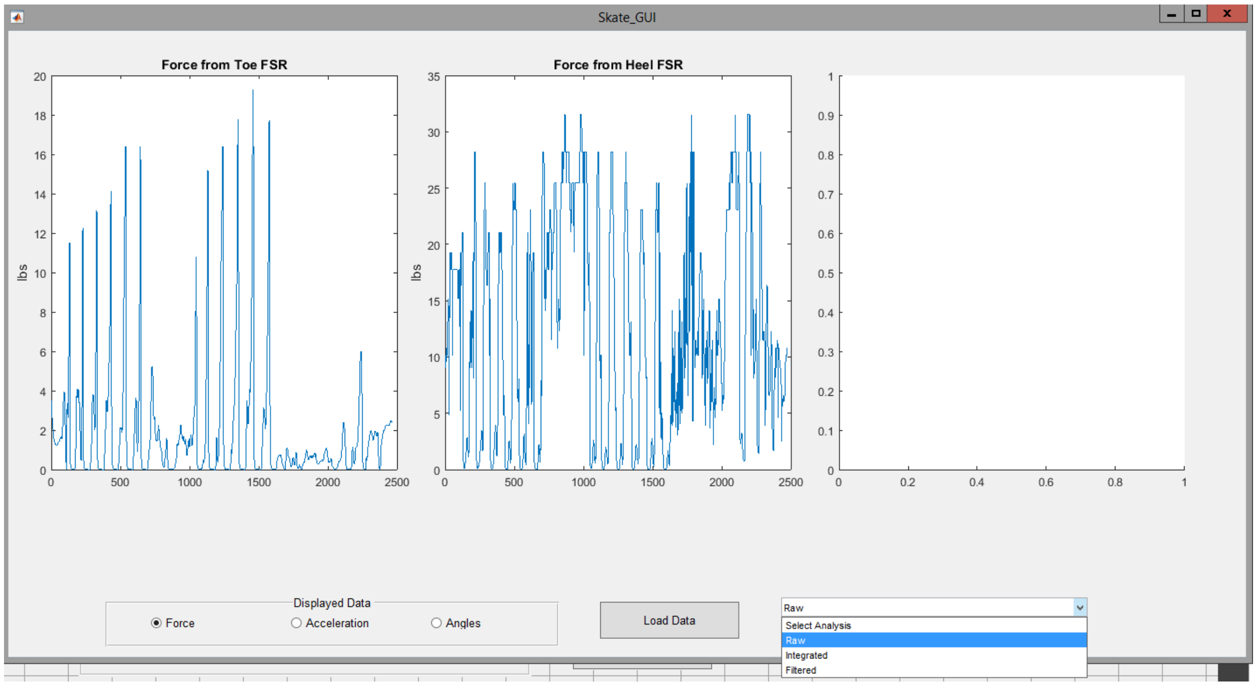


Figure 31: GUI View of Raw Force Sensor Data

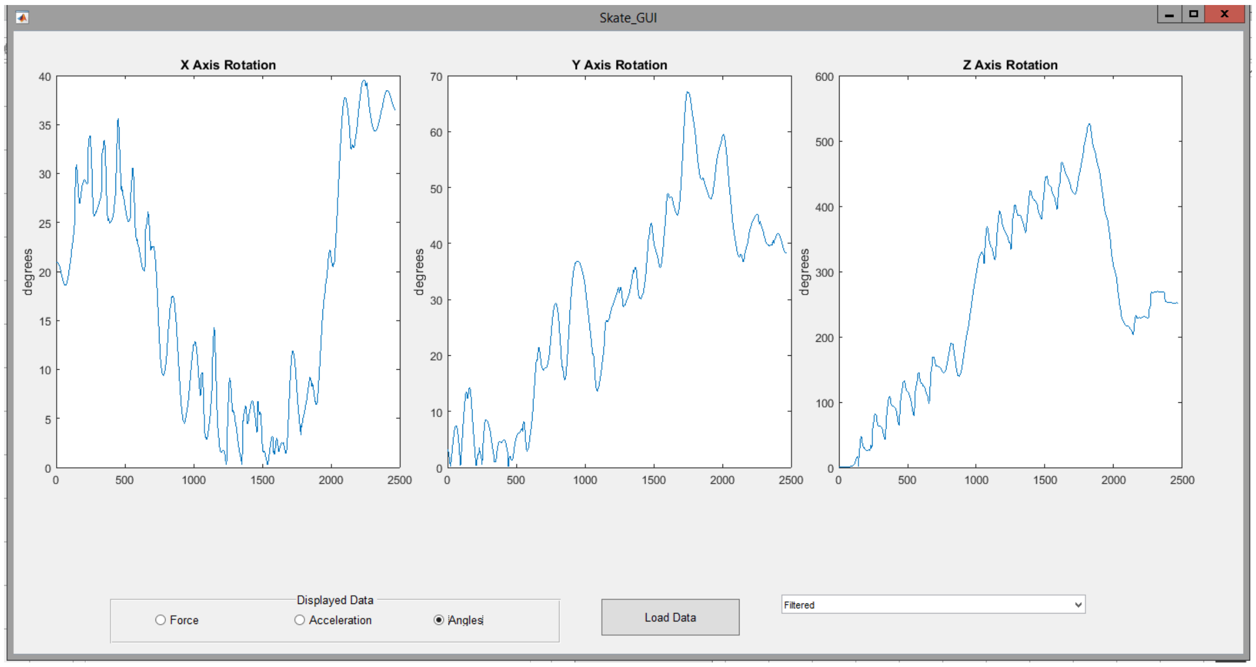


Figure 32: GUI View of Filtered Rotational Angle Data