Team 49 - Head Trauma Data Transmission and Logging

ECE 445 Spring 2017

Design Document

Philip Meyer, Conner Cecott, & Fahim Sheikh

TA: John Capozzo

Table of Contents

1 Introduction

1.1 Objective

Research suggests that there are as many as 3.8 million sports-related concussions in the United States every year and this number shows no signs of decreasing [1]. From high school athletics to the NFL, concussion rates are actually increasing annually. NFL athletes have remarked that if concussions don't happen on every play, then they occur on every other or every third play. Consequently, athletes end up playing through concussions and not getting immediate care.

The main reason that there does not exist a system to better monitor concussions is because there are numerous factors that play a role, including individual genetics. In addition, there simply is not enough data to enable research in this area. Our project hopes to address a few of these obstacles.

In conjunction with TEAM 55, the goal is to design a wearable device for athletes that has the capability of collecting heart rate and accelerometer data, wirelessly transmitting, performing particular algorithms, and storing the data for future analytics in the long term. Our team will be responsible for the latter half of the project. Specifically, our team will generate data on a Raspberry Pi, transmit the data wirelessly to our PCB, run our algorithm to gain insights, and wirelessly transmit the data to both an application and a computer that can send the data to the cloud (Amazon Web Services) for long term storage. The application will provide a user interface for either a coach or trainer. We hope that this device will provide insightful information that can be used in research for years to come.

1.2 Background

To date, a few systems do exist that try to address the media's concern around increasing concussions in athletics. For example, the Head Impact Telemetry System (HITS) was designed by Simbex in an effort to measure and record head impact exposure [1]. However, HITS, as well as other devices, only collect acceleration data. In addition, some of the devices try to predict concussions and alert coaches and/or trainers on the sidelines when a high-impact hit was made. Lastly, some devices are built into helmets, which might compromise the integrity of a helmet and would not be easily adopted by the NFL, for example. To summarize, current devices are limited by pure accelerometer data and provide no additional insights into the increasing amount of concussions.

Our design will be the first to simultaneously collect heart rate data and accelerometer data. However, it is important to note that we will not try to diagnose concussions, as this is not possible to do at the time of impact and will require detailed analysis by qualified medical researchers and professionals. We will simply analyze our data to infer impact locations and rotation of the rigid body to help enable research, as well as time lock this data to the player's heart rate. This data will then be available for coaches and/or trainers on the sideline before the start of the next play.

1.3 High Level Requirements

- Data resolution: In order to collect accurate accelerometer data, we will sample at 1000 Hz [11].
- Algorithm accuracy: In order to make use of the collected data, the algorithm should determine acceleration in six degrees of freedom for a rigid body with accuracy allowing 5+/- 5% error. Accuracy will be calculated based on simulated data.
- Scalability: The design must be capable of handling a varying number of accelerometers, with an upper limit set by the largest acceptable time delay before seeing data appear on the application. Beyond this, scaling requires an upgrade in processing power.
- Wearability: All hardware components, i.e. Bluetooth module and microcontroller, must fit within a volume of 124 x 59 x 7 mm. This is to ensure that our design can be worn by a player without significantly infringing on their normal movement and uniforms/protective wear.

2 Design

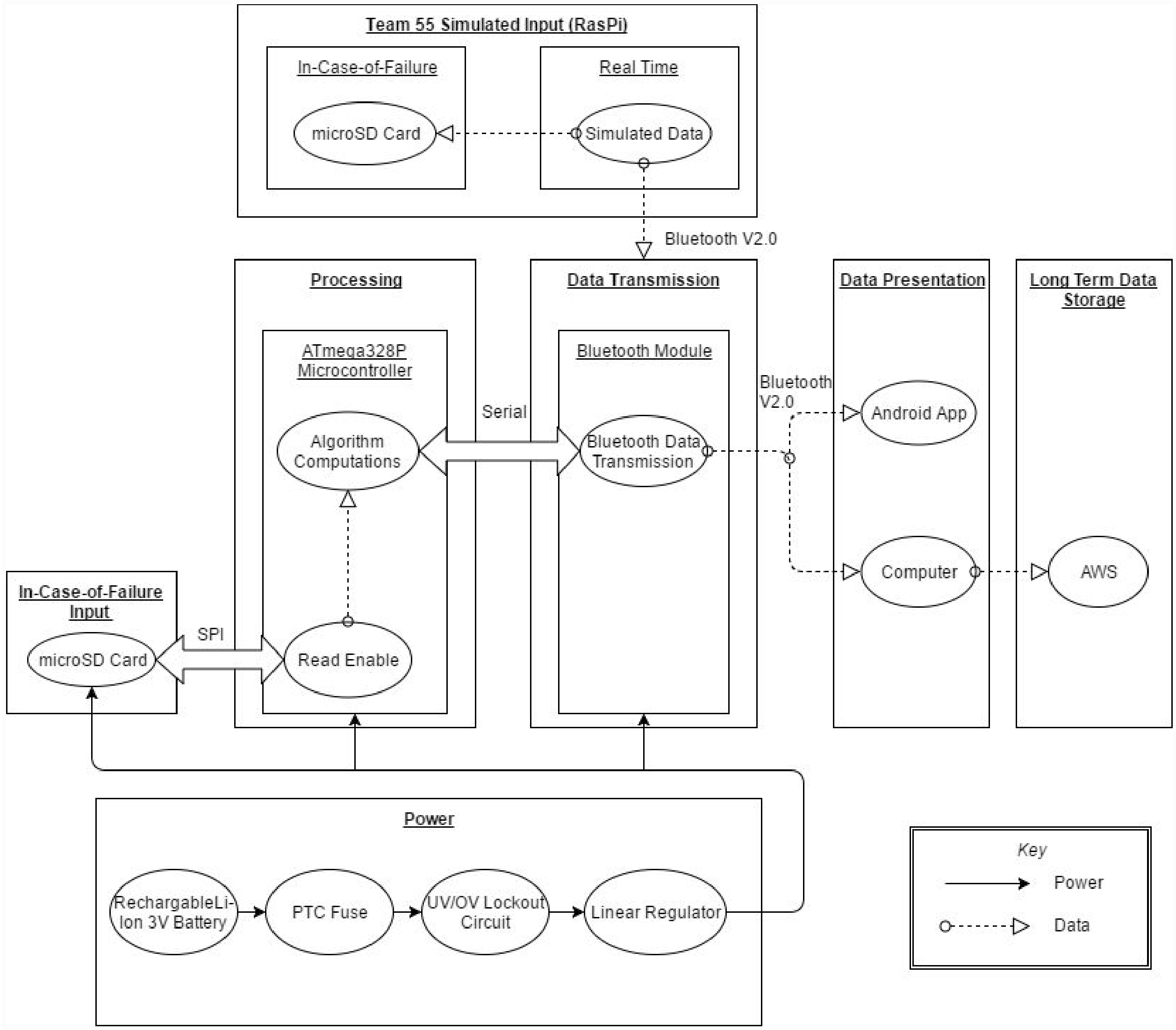

Figure 2-1 High Level Block Diagram

2.1 Block Diagram

The *Team 55 Simulated Input* block is the source of filtered data that will be wirelessly transmitted to our microcontroller from a Raspberry Pi. In case of transmission failure, the Raspberry Pi will also write the data to a microSD to ensure data is not lost. This block essentially resembles Team 55's portion of the overall design. If needed, the microSD card from the Raspberry Pi will be inserted into a connector that is soldered to the board that will communicate with the microcontroller through SPI transmission traces. This is the *In-Case-of-Failure* block. The *Data Transmission* block consists of a Bluetooth module that will receive the data wirelessly from the Raspberry Pi and send it to the microcontroller in the *Processing* block. The microcontroller will process the data by running the algorithm, and then transmit the data to a mobile application/computer (*Data Presentation* block) via the Bluetooth module. AWS will be used for *Long-Term Data Storage*. The *Power* block consists of a lithium ion rechargeable 3V battery with a PTC fuse for excess current protection, an UV/OV lockout protection IC for undervoltage and overvoltage protection, and a linear voltage regulator to ensure a consistent 3V power supply.

2.2 Functional Overview and Requirements

2.2.1 Team 55 Simulated Input (Raspberry Pi)

To emulate Team 55's wearable device that collects real-time acceleration and heart rate data, three tri-axle accelerometers and a Raspberry Pi will be used. In order to collect accurate accelerometer data, the Raspberry Pi will sample at a frequency of 1000 Hz [3]. Since each accelerometer sample will collect 16 bits per sensing axis, one impact sample will total 144 bits just for acceleration $(3\ accelerometers \times 3\frac{axis}{accelerometer} \times 16\frac{bits}{axis})$. 8 bits will be used for each simulated heart rate sample since they can represent a maximum heart rate of 256 BPM, which is outside the viable heart rate range. This totals 152 bits per sample. To calculate how much memory will be stored per game, a few parameters need to be set. Each impact event will include 60,000 samples. The average amount of plays in an NFL game is about 65, so if a skill position player (i.e. running back) saw about half of the plays *and* experienced an impact on each of those plays, the player would experience about 30 impacts per game. This amounts to approximately 34.2 MB of data per game, which does not impose any strict requirements on the size of the microSD card.

$$152\frac{bits}{sample} \times 60,000\frac{samples}{impact} \times 30\frac{impacts}{game} \times 1\frac{MB}{8e+6\ bits} = 34.2\ MB\ of\ data\ per\ game \qquad (1)$$

As data is collected, it will be wirelessly transmitted to the PCB's Bluetooth module. Additionally, as a precaution in case connectivity drops at some point for any reason, the Raspberry Pi will also store the data locally on a microSD card. This will ensure no data is lost and that the processing and storage can still be done at a later time, i.e. as soon as the microSD card is plugged in to our device.

2.2.2 In-Case-of-Failure Input

If a microSD card needs to be used to recover data that wasn't transmitted in real-time, it can be inserted into the connector on the PCB and used as the source of the data for the algorithm. On our PCB, the microSD card requires a voltage input of 2.7-3.6 Volts. The voltage being supplied from the lithium-ion battery is 3 volts. Therefore, the input voltage needs to be within 10% of 3 Volts to satisfy the microSD power requirement. Lastly, the average amount of data collected for one player over the course of a game will be about 273.6 MB. This does not impose any strict requirements on the size of microSD card we use.

Requirements	**Verification**
1. Continuously store data on microSD, as well as transmit wirelessly (if possible).	1. Connect device to Raspberry Pi via Bluetooth and insert microSD into Raspberry Pi. Collect accelerometer data, which should be received wirelessly, in real-time. Remove microSD card from the Raspberry Pi and plug into computer. Check that the data was written to the microSD.
2. Wirelessly transmit/write data when the accelerometers experience *at least* 50g's (+/- 3%).	2. Connect the Raspberry Pi and computer to the device via Bluetooth. Apply 50g's to the

	accelerometers. Verify on the computer that data was collected.
3. Wirelessly transmit/write data for 1 minute.	3. Apply 50g's to the accelerometers. Verify on the computer that 60,000 samples of data were collected.
4. Scalability: Transmit *n* x 48 bits per accelerometer plus 8 bits for heart rate for one impact event sample, where *n* is the number of accelerometers on the device.	4. Depending on *n,* compute the amount of data for each sample and ensure the Bluetooth module and microprocessor can transmit and compute in real time, respectively.

Table 2.2.2-1

2.2.3 Power

The power block consists of a lithium ion battery with a PTC fuse, undervoltage/overvoltage protection, and 3 volt linear regulator. The battery will be a 3 volt battery that uses the undervoltage/overvoltage and linear regulator to ensure an accurate 3 volts is supplied. The microSD card accepts 2.7-3.6 Volts, the microcontroller accepts 1.8-5.5 Volts, the Bluetooth module accepts 2.7-4.2 volts, the linear regulator accepts 2.7-5.5 Volts, and the UV/OV protection IC accepts 2.5-34 Volts. Therefore, the limiting factor on the input voltage is the microSD card, Bluetooth module, and linear regulator which requires the voltage to be within 10% of 3 Volts. Additionally, the microSD requires up to 200 mA, the UV/OV protection IC requires up to 125 μA, the Bluetooth module requires up to 35mA, the linear regulator requires up to 385 μA and the microcontroller IC requires up to 100 mA, so the maximum total current required for the PCB will be approximately 335.51 mA. However, the battery will be capable of supplying more current than this and the PTC fuse will be set to fault at 500 mA in case there are additional resistive losses that would draw unforeseen current.

For power protection purposes, the undervoltage/overvoltage lockout protection IC will be used. In terms of an open or short applied to the circuit, the PTC fuse will protect the circuit in the case of a short. In the event of an open, the UV/OV protection would not allow the circuit to operate and no damage would be done. The linear regulator will be used to keep the voltage level constant rather than fluctuating within the range allowed by the UV/OV IC. Additionally, the linear regulator has built in undervoltage protection. However, since it does not have overvoltage protection, the UV/OV protection is still needed in case someone inserts a higher voltage battery accidentally.

Furthermore, for real world use cases the unit should be able to function throughout the course of a whole game of any sport. Assuming that the upper limit on this requirement is three hours, the battery should be able to power the device at its average load current for three hours. The average load current, assuming real time data is being used (microSD in sleep mode), will be 350 μA for the microSD, 9.46 mA for the microcontroller, 35 mA for the Bluetooth module, 125 μA for the UV/OV IC, and 385 μA for the linear regulator, which comes out to be 45.32 mA total. In terms of scalability, with more accelerometers more power would be required, however for the purpose of our half of the project we do not supply power to the accelerometers. But, if enough accelerometers are used then a faster processor may be needed which in turn would require more power.

Requirements	**Verification**
1. Supply voltage within 10% of 3 Volts.	1. Apply a voltage meter between ground and the output pin of the linear regulator to verify that the voltage is within 10% of 3 Volts.
2. Supply 500 mA at 3 Volts.	2. Use a variable resistance load set to draw 500 mA at 3 Volts to verify the battery can supply that current.

	Use current meter to see current being drawn.
3. Supply average current, 45 mA, for three hours.	3. Use a variable resistance load set to draw 45 mA and allow to run for 3 hours. Verify battery is still capable of supplying 45 mA at 3 Volts after 3 hours by using a current meter and voltmeter.

Table 2.2.3-1

2.2.4 Processing

The *Processing block* consists of an ATmega328 microcontroller IC. The algorithm will be stored in its flash memory (32 KB available) and will receive the data to be processed from either the Bluetooth module or the microSD card. In its current state, the algorithm takes up approximately 8 KB of data. In real time, the algorithm will calculate acceleration in six degrees of freedom, which includes three axis each for both linear and rotational acceleration from the raw accelerometer data [3]. This processed data, along with the raw heart rate data, will be sent to the computer and Android app via the Bluetooth module at 2.4 GHz.

To ensure that the Atmega328 can handle the necessary computations in real-time, we have simulated our algorithm in Python. The Atmega328 achieves throughout at about 1 MIPS per 1 MHz. It also operates at 20 MHz. This allows for $20 \times 10^6 \frac{instructions}{second}$:

$$1 \frac{instruction}{cycle} \times 20 \times 10^6 \frac{cycles}{second} = 20 \times 10^6 \frac{instructions}{second} \quad (2)$$

Our simulated gradient descent algorithm is primarily broken up into two main components (*while* loops). The first component checks if we have have achieved the minimum of our loss function. If the minimum has not been achieved, we move in a direction opposite of the gradient. On average, it takes approximately 20 iterations (steps) to achieve the minimum. Each iteration is composed to a multiple calculations such as 2-norms, multiplications, subtractions, assignments, and comparisons. These calculations are negligible compared to the 80 function evaluations and 180 gradient evaluations that take place. Below in the *Algorithm* description, the loss function is defined. To evaluate the function once, it takes approximately 234 operations, due mostly to the cross and dot products. To evaluate the gradient of the function, it takes approximately 297 operations for the same reason. In total, about 67,320 operations (additions, multiplications, subtractions) are evaluated per data point from an accelerometer. Since each impact event total 60,000 samples, it will take approximately 3.4 seconds to process the data using our algorithm.

$$67,320\frac{instructions}{data\ point} \times 60,000\ data\ points \div (20 \times 10^6 \frac{instrucitons}{sec}) = 202\ sec = 3.4\ mins \qquad (3)$$

An additional requirement of this block is being able to receive the collected data that we wish to process. To aid in this endeavor, we will be designing our own network protocol. This is done partially so that we can have more control over how the data is transferred as well as to ensure that anyone who uses our protocol will immediately be able to utilize our system.

Requirements	**Verification**
1. Data processing to be completed within 3.4 minutes	1. Load 60,000 data points onto the computer. Run and time the algorithm and ensure it executes in under 3.5 minutes.

2. Bluetooth module successfully transmits data to Android app/computer	2. Ensure data is successfully received by Android app/computer.

Table 2.2.4-1

2.2.5 Algorithm

In the field, the wearable device will have three tri-axle accelerometers and one heart-rate sensor. When our microcontroller receives this time-locked data, the processing engine will either (1) perform an algorithm to gather insights on the accelerometer data or (2) simply pass the heart-rate data into transmission. Our algorithm has one main objective: to infer the linear acceleration of the head's center of gravity, as well as to infer the rotational acceleration of the head's center of gravity. This results in six degrees of freedom that describe the motion of a rigid body upon impact [5]. The acceleration at any point, *i,* on the head, $\overline{a_i}$, undergoing linear and rotational acceleration can be described by:

$$\left\|\overline{a_i}\right\| = \overline{r_{a_i}} \bullet \overline{H} + \overline{r_{a_i}} \bullet (\overline{\alpha} \times \overline{r_i}), \qquad (2)$$

where $\overline{r_{a_i}}$ is the sensing access of the accelerometer, $\overline{H}$ is the linear acceleration of the head's center of gravity, $\overline{\alpha}$ is the rotational acceleration of the head's center of gravity, and $\overline{r_i}$ is the position vector of the accelerometer. All parameters are known to us except $\overline{H}$ and $\overline{\alpha}$. The following convex optimization problem will be solved either numerically using gradient descent, or analytically since the matrix is small enough to compute an inverse/pseudo-inverse:

$$min_{\overline{H},\overline{\alpha} \in \Re^3} \sum_{i=1}^{n} \sum_{j=1}^{m} \left[\left\|\overline{a_{i_j}}\right\| - \left(\overline{r_{a_{i_j}}} \bullet \overline{H} + \overline{r_{a_{i_j}}} \bullet (\overline{\alpha} \times \overline{r_{i_j}}) \right) \right]^2, \qquad (3)$$

where n is the number of sensors and m is the x, y, or z axis.

Requirements	Verification
1. Calculate linear and rotational acceleration of the rigid body to within 5% +/- 5%	1. Generate test data in Unity and gather accelerometer data from Simulink block. On the computer, run the algorithm that is stored on the microcontroller and compare the calculated accelerations against generated test data from Unity.
2. Scalability: for *n* accelerometers, the algorithm more accurately calculates linear and rotational acceleration than our implementation, which uses 3 accelerometers	2. Generate test data in Unity and gather accelerometer data from Simulink block. On the computer, run the algorithm that is stored on the microcontroller and compare the calculated accelerations against generated test data from Unity.

Table 2.2.5-1

2.2.6 Data Presentation

If the user wants to view the processed data in real time, a mobile application will be responsible for presenting data sent wirelessly from the hardware. Software will be used to generate useful plots, such as waveforms and histograms, that will depict magnitudes of impacts and direction of head rotation. There will also be an added feature that will allow a coach or trainer to tie a player's information to their respective data. This data will then be logged to AWS after being sent to a computer. Since our processor only has 32 KB of memory, we are estimating that we will send packets of 20KB via Bluetooth, which will take approximately 0.15 seconds (includes transmission to and from device). To processes this data using a gradient descent algorithm at 16 MHz, it will take approximately 3.16 seconds. If we give the application a half a second to generate useful plots, data will be displayed on the application's screen within 4 seconds.

Requirements	Verification
1. Ability to present data in a useful fashion	1. View magnitude of acceleration experienced during impact event.
2. Start displaying data on the sideline application in 4 seconds.	2. Time the following process: apply force to the accelerometers, send data to the microcontroller, send data to the application, and display plots.

Table 2.2.6-1

2.2.7 Long-Term Data Storage

Long-term data storage will most likely be in an AWS S3 bucket used for safe, low cost storage (free 20,000 get requests of up to 15GB per month). Put requests will be made from a computer (2,000 free put requests per month).

2.3 Supporting Material

2.3.1 Design Schematic

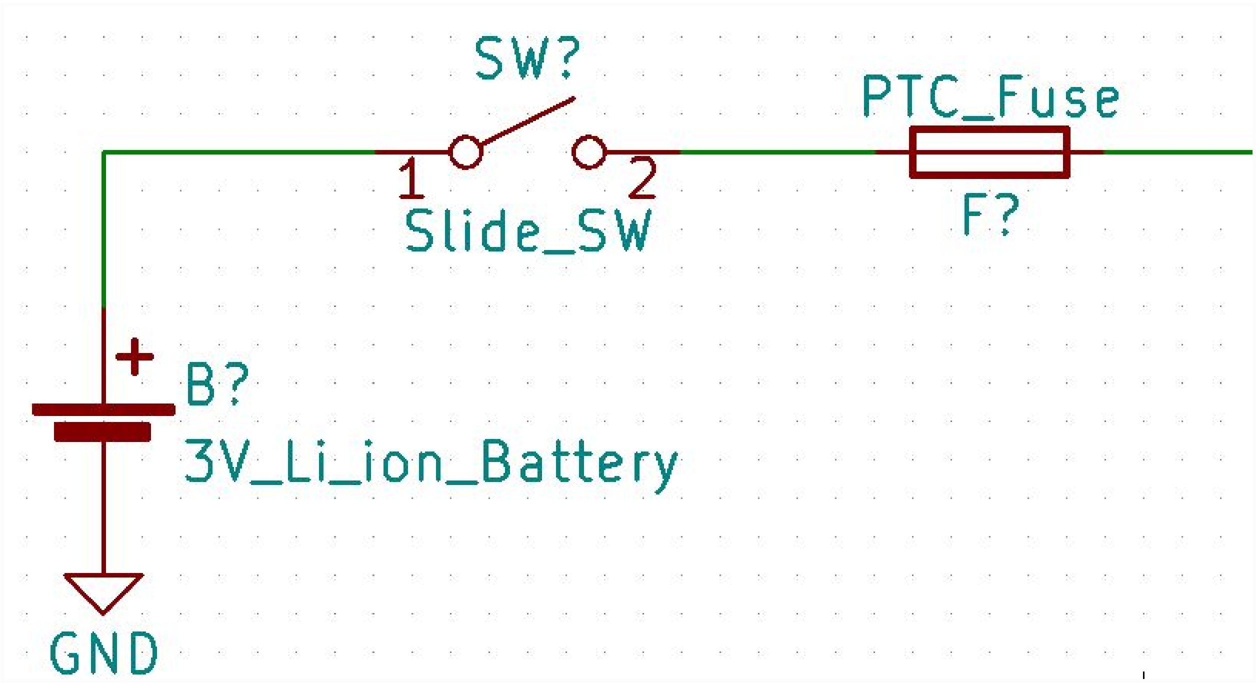

Figure 2.3.1-1 Power Input Circuit Schematic

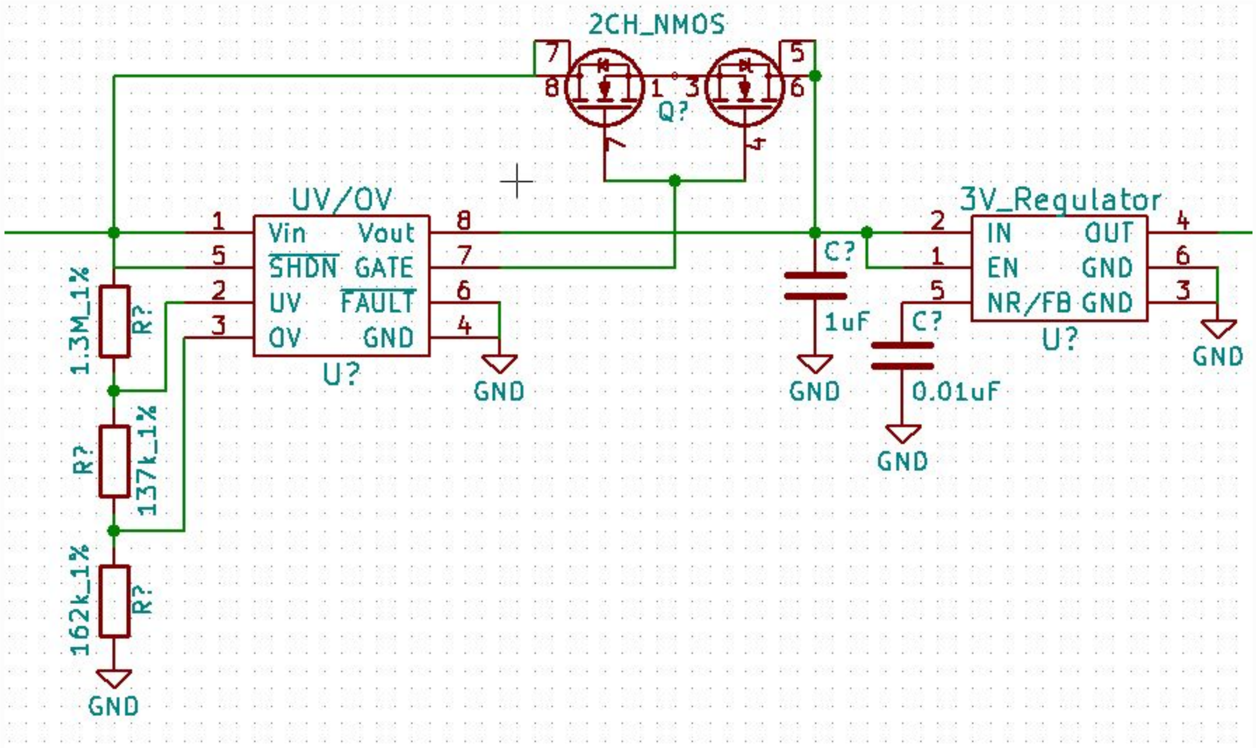

Figure 2.3.1-2 UV/OV Protection and Linear Regulator

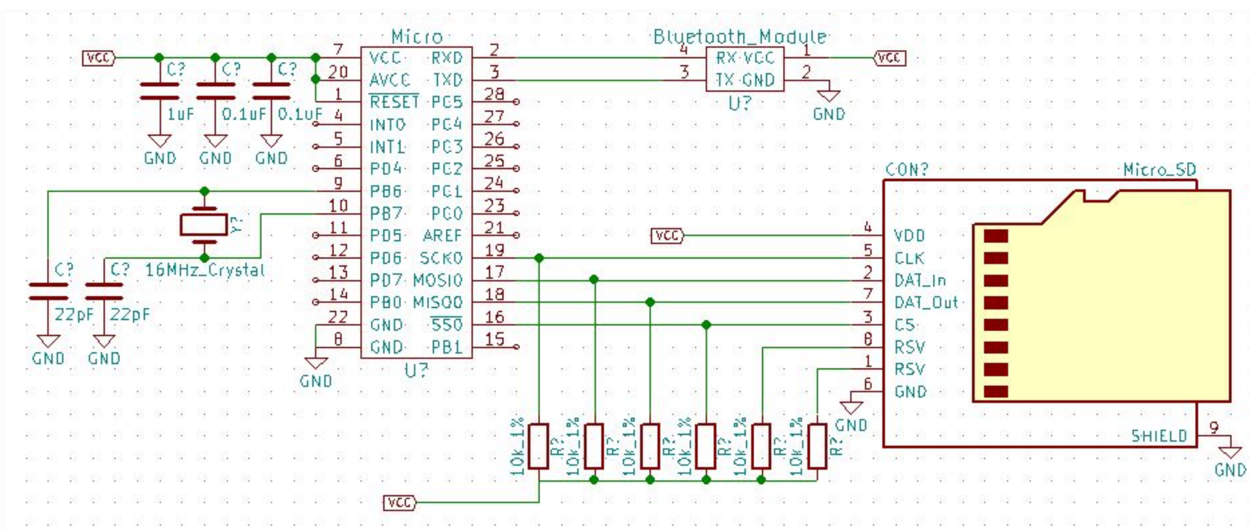

Figure 2.3.1-3 Microcontroller, Bluetooth Module, and microSD

These circuit schematics are for the design described above. The design was determined using the recommended layout as in the datasheets [6], [7], [8], [9].

2.3.2 Test Data Generation Block

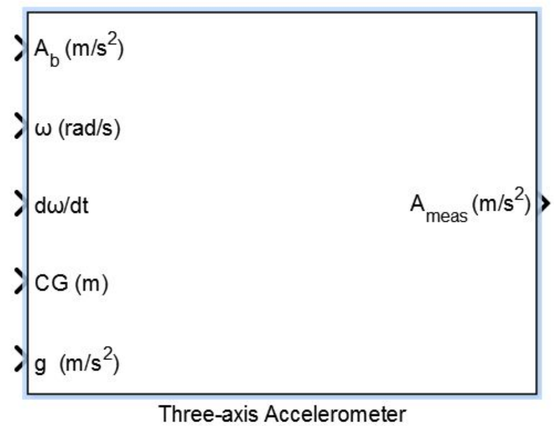

Figure 2.3.2-4 The simulink block diagram of an accelerometer.

In Unity, true linear and rotational acceleration of a rigid body is generated by applying a force. Those generated values are sent to the Simulink block seen above to generate accelerometer data which is used as test input to our algorithm. The outputs of the algorithm, i.e. estimated values of linear and rotational acceleration, are then compared with the true values generated in Unity. This process provides us a way of testing and validating the algorithm.

2.4 Tolerance Analysis

The overall usefulness of this project is determined by the accuracy of the algorithm and the ability to be used in a real world environment. The ability to be used in a real world environment implies that the unit must be small enough to not inhibit game play and have enough power to last a full three hour game. If the requirements above about size and battery life are met, then the unit would have the ability to be used in a real world situation.

In terms of accuracy, although the algorithm was carefully chosen due to its promise of reliable and accurate calculation of linear and rotational acceleration, we have yet to put it in practice. Since the main purpose of our design is to log accurate data, we have to ensure that the algorithm produces accurate data. Because in the end, even if everything else works perfectly, the overall objective won't be accomplished if the data we generate is useless.

To actually check the validity of our data, we will calculate error based on test data generated from a simulation. To generate the simulation, we apply a force and torque to a rigid body of a certain mass m and moment of inertia J. Using equations from statics such as $F = m*a$ and $\tau = J*\alpha$ we can calculate the linear and rotational acceleration and treat those values as the true quantities that the algorithm is trying to estimate. The simulation also creates a list of accelerometer data that we can export to a program like Matlab for analytics. Applying the exported data as input to our algorithm allows us to create estimates for linear and rotational acceleration that can be compared with the "true" values mentioned earlier. Since the goal is to minimize error, we would ideally like to have 0% error between the estimated values and the true values. However, since we do not live in an ideal world we allow a tolerance of up to 5% +/- 5% error, averaged out over a series of 100 comparisons. This tolerance range was derived from a paper that is the source of our algorithm. In the paper, their tolerance range was much larger, 1%+/-18% for linear acceleration and 3%+/-24% for rotational [3]. The stated reason for such a large range is due to nonideal placement of accelerometers and the shifting of helmets, neither of which is a major concern for us in these tests. As such we decided on a stricter range for our design. If we remain within this tolerance range the algorithm will be considered an accurate predictor and the system as a whole will be a success, assuming, of course, that all other components work. Falling outside of this tolerance, however, means that we are not predicting values with enough accuracy to reach our intended target. However it is important to note that even falling to 10% outside our desired range would still give our system a better error percentage than the cited 6DOF paper, as well as the original HITS device [3].

Another factor closely related to the overall success of this design is the wearability of the device. Since the data we want to process and store comes from an athlete playing their sport, we need to ensure that our device is small enough to not interfere with the athletes performance on the field. Because if the device is seen as too encumbering then no one will want to use it and thus we will have no data to process, store, and share. This obviously means that, even if the rest of the device works as expected, our overall goal won't be met since we would not be able to gather data from actual people. And since it is that "live" data which makes our device fundamentally useful and desirable, we have to ensure that this device fits within the specified volume. In this way, the accuracy of the algorithm and the wearability of the device are linked since increasing the accuracy may involve getting a larger microprocessor, but a larger microprocessor may violate the space requirement of wearability.

3 Cost and Schedule

3.1 Cost Analysis

The labor cost of development of this project is predicted to be proportional to $30/hour for each member of the team over a period of 16 weeks. Thus the total cost of labor for the whole team is estimated to be:

3 * $30/hour * 20 hours/week * 14 weeks = $ 25,200 (3)

As for the cost for parts, please refer to the table below.

Part	Vendor	Quantity	Cost (Individual)
3 V Li-Ion Battery	DigiKey	1	$0.90
Microcontroller (ATmega32)	Microchip	1	$1.83
PCB’s	PCBWay	1	$10
Various passive components (RLC) and ICs	ECE Supply Store	N/A	$0.00
Sandisk 8GB microSD	Amazon	1	$5.49
microSD Slot (472192001)	Avnet	1	$0.98
PTC Fuse (MF-FSMF035X)	Mouser	1	$0.42
Switch	Digikey	1	$2.52

(CL-SA-12C-02)			
Raspberry Pi 3 Model B	Amazon	1	$35
UV/OV (LTC4365CTS8#TRMPBF)	Arrow	1	$2.13
Accelerometer (ADXL375BCCZ)	Analog Devices	3	$7.92
HC-06 Serial Bluetooth Module	Amazon	1	$9.99
Total	N/A	13	$77.18

We only plan on building one fully functional device by the end of the semester so the total development cost is predicted to be $25,277.18. This price may increase if replacement parts are required and so this price should be taken as a lower bound.

3.2 Schedule

Week	Phil	Conner	Fahim
2/27/17	• Design Review • Research bluetooth protocol	• Design Review • Order hardware • Begin PCB Layout	• Design Review • Generate test data • Research bluetooth protocol
3/6/17	• Soldering Assignment • Simulate algorithm	• Soldering Assignment • Complete PCB layout • Order PCB	• Soldering Assignment • Start building Android App
3/13/17	• Collect	• Start soldering	• Design

	accelerometer data on Ras Pi ● Set up AWS environment	PCB components	network protocol ● Debug Android App
3/20/17	● Begin programming microcontroller	● Finish Soldering PCB	● Implement network protocol
3/27/17	● Individual Progress Reports ● Test microcontroller functionality with microSD data	● Individual Progress Reports ● Verify power requirements throughout circuit	● Individual Progress Reports ● Continue network protocol
4/3/17	● Test connectivity between PCB and computer/AWS	● Test connectivity between PCB and Android app	● Test connectivity between Ras Pi and PCB
4/10/17	● Test overall functionality ● Prepare demonstration	● Test overall functionality ● Prepare demonstration	● Test overall functionality ● Prepare presentation
4/17/17	● Mock Demo ● Prepare final demonstration	● Mock Demo ● Prepare final demonstration	● Mock Demo ● Prepare final presentation
4/24/17	● Final Demonstration ● Complete final paper	● Final Demonstration ● Complete final paper	● Final Demonstration ● Complete final paper
5/1/17	● Final Paper Due	● Final Paper Due	● Final Paper Due

4 Ethics and Safety

4.1 Ethics

Since our device is primarily a data logger/processor we do not foresee any major ethical concerns. As far as our data processing is concerned, we are not going to make any inferences or claims based on our data. Rather we are simply collecting and transmitting useful data that can be provided to other individuals or organizations to utilize in their research. Any claims or conclusions that are made based off of our data is not our ethical responsibility. We will go so far as to assure that the shared data has not been altered or tampered with in any way to support any particular study or conclusion. Additionally, since the data we are collecting is health information of athletes we can fall under the protection of HIPPA [2]. Under HIPPA (Health Insurance Portability and Accountability Act of 1996) the healthcare information of an individual, such as heart rate, is protected by federal law. An individual can choose to share their healthcare information with apps and companies such as ours and this ensures the protection of our data as well as the privacy of the athletes.

4.2 Safety

The only real piece of our design that could potentially harm someone is the lithium ion battery. A relatively small explosion or a small fire can be caused by sudden capacity loss from thermal runaway which could be caused by a short across the battery. Depending on how close an individual is to our device if such a failure was to occur, there could be some minor injuries. To mitigate the risk of such an event and ensure the safety of our users we will include a PTC fuse to disconnect the hardware from the power source in the case of a short. This will also have the added benefit of protecting the rest of the circuit from damage. Also, since we are providing a detailed description of our inputs and outputs, all potential risks associated with our device are clearly defined. This minimizes any major risk of injury due to ignorance.

5 References

[1] Beckwith, J.G., Greenwald, R.M. & Chu, J.J. Ann Biomed Eng (2012) 40: 237. doi:10.1007/s10439-011-0422-2

[2] "Your rights under HIPAA," in *US Department of Health And Human Services*, HHS.gov, 2017. [Online]. Available: https://www.hhs.gov/hipaa/for-individuals/guidance-materials-for-consumers/. Accessed: Feb. 8, 2017.

[3] S. Rowson, G. Brolinson, M. Goforth, D. Dietter, and S. Duma, "Linear and angular head acceleration measurements in collegiate football," *Journal of Biomechanical Engineering*, vol. 131, no. 6, p. 61016, Jun. 2009. [Online]. Available: https://biomechanical.asmedigitalcollection.asme.org/article.aspx?articleid=1422176. Accessed: Feb. 8, 2017.

[4] Burger, Thomas. "How Fast Is Realtime? Human Perception and Technology." *PubNub*. PubNub, 09 Feb. 2015. Web. 09 Feb. 2017.

[5] Chu, J. J., J. G. Beckwith, and R. M. Greenwald. "A Novel Algorithm to Measure Linear and Rotational Acceleration Using Single Axis Accelerometers." *Journal of Biomechanics*39.1 (2006): 60. Web.

[6] "TPS795 Ultralow-Noise, High-PSRR, Fast, RF, 500-mA Low-Dropout Linear Regulators." *Texas Instruments.* Ti.com, 2015. [Online]. Available: http://www.ti.com/lit/ds/symlink/tps795.pdf. Accessed: Feb. 24, 2017.

[7] "ATmega328/P." *Atmel.* Microchip.com, 2016. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Summary.pdf. Accessed: Feb. 24, 2017.

[8] " Overvoltage, Undervoltage and Reverse Supply Protection Controller." *Linear Technology.* Linear.com, 2013. [Online]. Available: http://cds.linear.com/docs/en/datasheet/4365fa.pdf. Accessed: Feb. 24, 2017.

[9] "HC-06 Module Data Sheet." *Guangzhou HC Information Technology Co., Ltd*. Olimex.com, 2011. [Online]. Available: https://www.olimex.com/Products/Components/RF/BLUETOOTH-SERIAL-HC-06/resources/hc06.pdf. Accessed: Feb. 24, 2017.

[10] Williams, Mark. "Guide to Interfacing a Gyro and Accelerometer with a Raspberry Pi." *Ozzmaker.com*. N.p., 02 Sept. 2016. Web. 24 Feb. 2017.

[11] Wu, Lyndia C., Kaveh Laksari, Calvin Kuo, Svein Kleiven, Cameron R. Bass, and David B. Camarillo. "Bandwidth and Sample Rate Requirements for Wearable Head Impact Sensors." *Journal of Biomechanics* 19.13 (2016): 2918-924. Web. <http://www.sciencedirect.com/science/article/pii/S0021929016307448>.