Head Impact Telemetry Data Logging System

ECE 445 -- Final Paper -- Spring 2017

Contributors:
Matt Hebard | Matt Schafer | Evan Qi

TA:
John A. Capozzo

Abstract

This design project approached the issue of ignorance in effects of concussions during professional sports games by designing a wearable device that concurrently monitors the acceleration of the cranium and the heart rate of a subject. The data sets are merged and time stamped together before being stored to flash memory, which can later be accessed by a researcher. The data is stored in a datalog format, which allows for easy parsing for graphing or any other function needed post-game.

Table of Contents

1. Introduction

In the United States, the entertainment industry is vastly dominated by sports. Many of these sports are high impact sports, and the risk of injury must be considered. Football, lacrosse, and hockey, as well as many others, are among these high impact sports. According to the British Journal of Sports Medicine, there are an estimated two to three million cases of concussions in the States alone per year [1]. Most of these concussions are unreported and untreated. If the risk of concussion for each player were calculated in real time, then coaches and officials could be notified of any potential injuries as they happen, and medical attention could be given immediately. Many products introduced today analyze only the force applied to the head in these collisions, but do not focus on how the collision affect the different parts of the body. This data detection cannot give any insight to effects inside the body from head collisions.

The purpose of the Head Impact Telemetry System (HITS) is to monitor the acceleration of the cranium. Using an accelerometer and heart rate sensor, we will be able to simultaneously log and store data to be utilized for research and analysis at a later date. Upon a sharp impact, the accelerometer shows a spike in acceleration. The device has a preset threshold beyond which any acceleration higher than the threshold poses a risk for a concussion. At the same time, the HITS has a heart rate monitor as well to track the heart rate before and after impacts to provide insight on the effects of concussions on the heart. These two sensors are to be time locked before being stored to the SD card, allowing the two data sets to be relatable to each other when reviewed at a later time. These two data sets could be collected separately, for example using a *Garmin Heart Rate Monitor* as well as current head acceleration equipment already in commercial use, however, by collecting these data sets on a single device we can reduce size and cost, as well as eliminating additional technology needed to analyze these separate data sets together. We created a device that coordinates acceleration data with heart rate data on a single compact device to make analysis immediate and simple for the user to study this information. Once the player has received medical attention, the time locked data can be transferred to an interface for a researcher to study the variations in pulse. Ideally we would like to look at the data in real time, but for the purposes of our project, we are aiming to detect changes in head acceleration and how these impacts can affect the health of the subject.

- **High-level requirements list:**
    - Accuracy: The accelerometer must output acceleration data in x, y, and z directions with at minimum 85% accuracy. This 15% error will account for noise on the SPI lines as well as calibration errors within the accelerometer. The pulse sensor must detect 98% of pulses that occur, even while in motion. Heart rate monitors in current commercial use have been tested to be no more than 4 bpm error when compared to a medically graded ECG Q-test machine [10], and thus we require that our heart rate monitor also calculate heart rate to within ±4 bpm error.
    - Wearability: The entire device will be worn on the body and must be reasonably small and light as to not obstruct movement or throw off the player's center of gravity. The PCBs will not exceed 10cm x 9cm x 2.9 cm and weigh less than 400 grams.
    - Lifetime: The device must, at the minimum, carry enough battery life and storage space to last the duration of a three-hour professional sports game.

2. Design

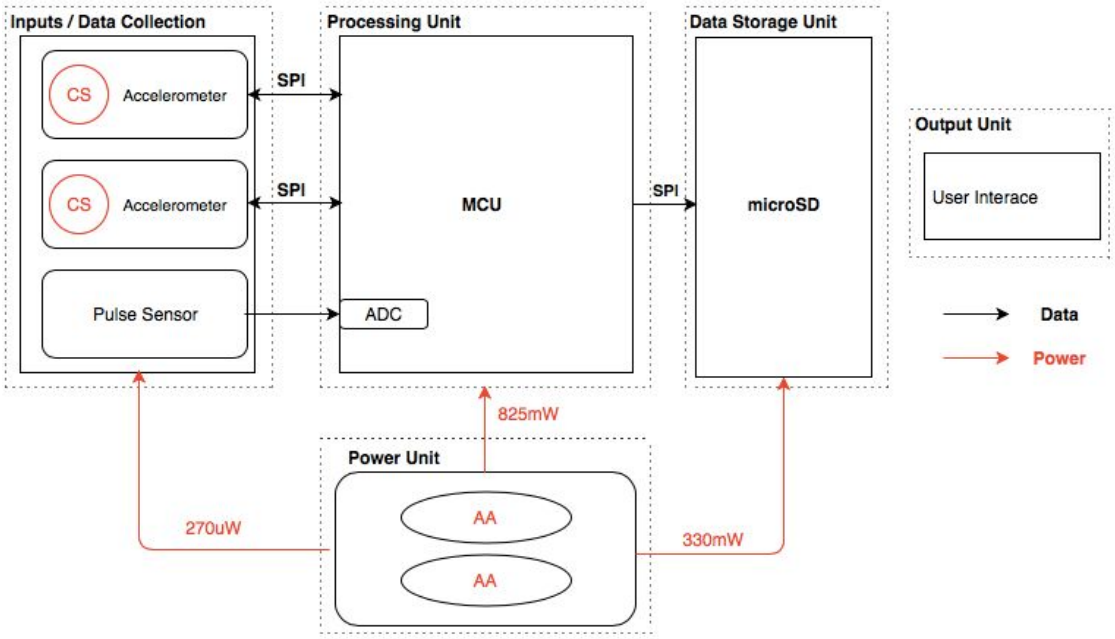


Figure 1: HIT Data Logging Block Diagram

Block Diagram Overview

The system above in Figure 1 can be broken down into five separate subsystems: inputs/data collection, processing, storage, output, and power unit. Moving from left to right: the input/data collection module consists of two sensors: an accelerometer (placed behind either ear) and a pulse sensor (that is placed on the individual's neck or ear). The power unit (bottom of figure 1) consists of 2AA batteries and a boost converter circuit to regulate voltage powering the 3 main subsystems. The processing unit consists of our ATmega328P microcontroller. The third unit in the block diagram is the storage unit. This is where all the data is stored, removable micro SD card mounted on the *printed circuit board* containing the MCU and processing unit. The last section of the block diagram is not entirely in our design, but we wanted to include it to indicate that we have an output monitor when we show our *proof-of-concept* to display all of the data we capture.

2.1 Sensors

The HIT system used a total of two sensors: an accelerometer and a heart rate sensor. The accelerometer gathers information on the linear acceleration of the head behind the ear. The heart rate sensor is used for gathering an athlete's pulse in real time. These sensor’s outputs are then fed as inputs into the PCB’s MCU where they are processed, filtered, time stamped, and stored onto a microSD card. The sensors are powered by the battery located on the printed circuit.

2.1.1 Accelerometer

The accelerometer that we utilized in our design is Analog Devices's ADXL375 3-Axis, ±200g Digital MEMS Accelerometer. Our initial design consisted of two accelerometer chips, one on either side of the head, however, when ordering the pcbs through the course, we only received a single accelerometer board and needed to adjust our designs to what we were given. One reason this particular accelerometer was chosen was due to its high range of sensing. Studies show that concussions can occur at cranial accelerations as low as 60*g*. Most other accelerometers do not reach such high values, and thus an accelerometer that can sense beyond this value is essential to this design. This device also contains an internal ADC and a built in digital filter, which proved to be valuable, saving us time as we do not need to filter the signal out ourselves. The device has a controllable output data rate and bandwidth of up to 1600 Hz. Sampling acceleration for concussions requires a bandwidth of at least 600 Hz [4]. For our desired design, power consumption is greatly reduced by limiting our ODR to 1600 Hz and bandwidth to 800 Hz. With a sample rate of 1600Hz, we chose to communicate over SPI at a clock speed of 4MHz in order to not lose necessary samples from the accelerometers. We chose to use SPI communication due to our need to log data from multiple slave devices in real time. The table below shows the calculations we derived of the minimum required time for SPI communication in order to capture a read at 4MHz.

Table 1: Timing Chart for SPI communication between MCU and Accelerometer

Cycle Parameter	Time (ns)
t_{delay}	5
t_{setup}	5
t_{hold}	5
t_{SDO}	40
Select Address	8bits = 2000
Send Data back to MCU	48bits = 12000
t_{DIS}	10
SPIxTXB → SPIxTXSR	3000
SPIxRXSR → SPIxRXB	3000
SPIxRXB → SPIxBUF	3000
Total	23065ns

At a sampled data rate of 800Hz from the accelerometer, the next sample will be collected in roughly .625ms. With this in mind, reading each sample in .04613ms with our SPI communication of 4MHz we found provides enough time for each accelerometer sample to be collected without skipping reads.

When configuring the accelerometer we also wanted to take into account low power consumption. The 800Hz bandwidth gives us plenty of range to detect all necessary data for logging a concussive blow and also effectively reduces accelerometer current draw by 20%, from 150 uA to 118 uA. By decreasing the ODR of our accelerometer,

we also reduced a lot of the noise in our accelerometer data being sent to the MCU, allowing for less filtering required by MCU, further lowering power consumption. In addition to this, we also implemented a low power state on the accelerometer, which forces the accelerometer to reduce sample rate and power consumption when there is no activity for a certain amount of time. In this low power mode, the accelerometer can still detect activity at a lower sample rate, and when it does, it will return to sampling acceleration at a high rate. Using this low power state we were able to reduce the accelerometer's power consumption by an additional 60%.

Figure 2: Accelerometer Schematic

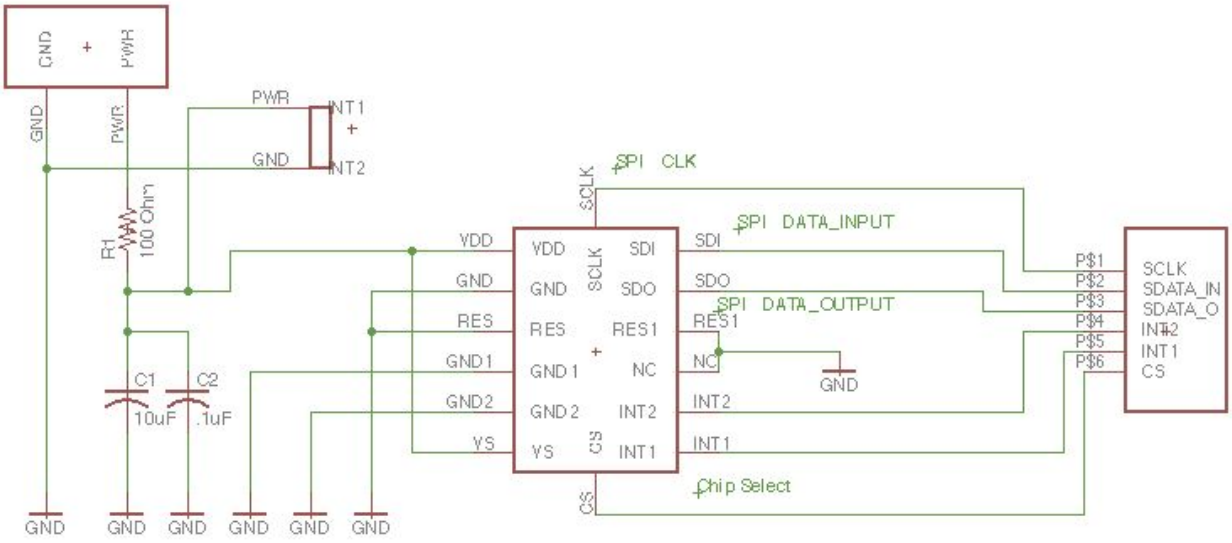
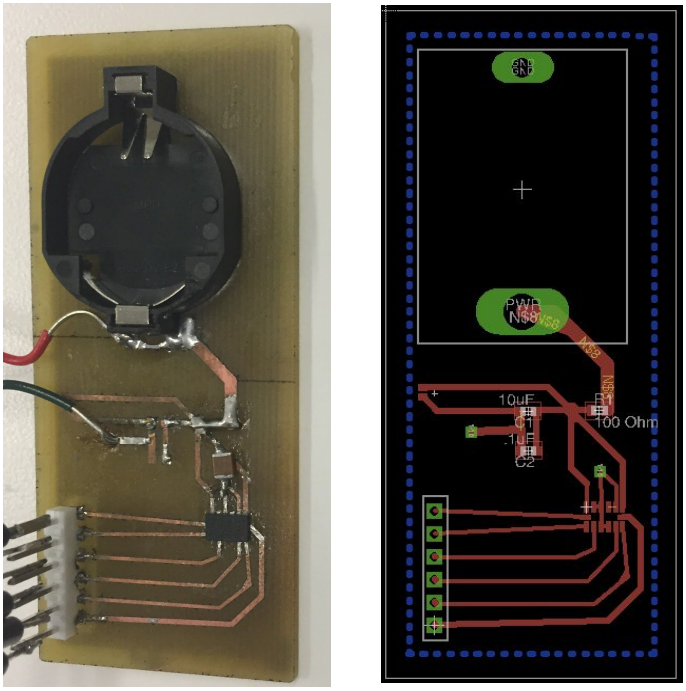


Figure 3: Accelerometer PCB



When designing our Accelerometer board, we chose to add in two parallel capacitors in series with a 100Ohm resistor in an effort to decouple the Supply voltage and Digital voltage pins from the coin cell supplying a 3.0V input with significant noise. We noticed when we hooked up our design the chip was still reading an unreasonable amount of noise when communicating with it via SPI. Matt determined that our capacitors may have been placed too far away from the input pins, and were not filtering the supply noise as much as we had planned. Matt then resoldered the capacitors and determined a significant noise decrease and improvement of signal clarity.

2.1.2 Pulse Sensor

The Pulse Sensor we utilized was the Amped Pulse Sensor, SEN-11574, provided by Adafruit. Our reasoning behind choosing this device is its low cost and low power consumption. The pulse sensor relies on an LED and a surface mount photodetector to detect the reflected light off the surface of the skin to produce a photoplethysmographic signal. Due to the physiological workings of the body, the signal shows a large spike during the heart beat, afterwards dropping to below the original value. After the pulse, the signal oscillates, quickly diminishing to the resting value. The waveform generated from a reading the pulse from the earlobe can be seen in figure 4. The output is sent through an op-amp where it then outputs an analog signal to our MCU's 10-bit ADC module.

Based on research studies, it is necessary to sample heart rate sensors at a minimum of 500Hz to collect viable data [3]. For our design we chose a higher sample rate of 800 Hz in order to match the 1600 Hz data collection from the accelerometer to allow for easier integration. The samples are vulnerable to noise due to the higher sampling frequency, and thus the sampled data must be filtered before being processed. At first, we tried to filter 16 samples at a time (in order to match the 32 acceleration samples), but later realized that filtering through convolution was too slow. Circular convolution between two size 16 data sets required 42,000 instructions, which took 2.5 ms on our microcontroller. Considering that we need to sample the pulse sensor every 1.25 ms, this idea was scrapped due to its speed restrictions. Thus, we ended up using a real-time 16-tap FIR filter generated through the Parks-McClellan algorithm through MATLAB with the following Z transform:

$$H(Z) = -0.0019 + 0.0614Z^{-1} - 0.0292Z^{-2} - 0.0180Z^{-3} + 0.0802Z^{-4} - 0.0943Z^{-5} - 0.0206Z^{-6} + 0.5606Z^{-7} + 0.5606Z^{-8} - 0.0206Z^{-9} - 0.0943Z^{-10} + 0.0802Z^{-11} - 0.0180Z^{-12} - 0.0292Z^{-13} + 0.0614Z^{-14} - 0.0019Z^{-15}$$

[Eq. 1]

The frequency response of the digital filter can be seen in figure 5, where the signal is clearly attenuated beyond 0.75π, or equivalently 600 Hz.

Since the waveform consists of peaks in voltage that signifies a pulse, a threshold voltage had to be set for a pulse to be detected. To set this limit, the program initialized by reading the samples for two seconds, and finding the average voltage. Within these two seconds, the maximum voltage is also found, which is the voltage of an arbitrary pulse in that time span. The threshold voltage is then set to be 2/3 the difference between the average and maximum voltages. The threshold voltage ratio could be changed to whatever value between 0 and 1, but setting it too low would cause more false alarms, while setting it too high would cause missed detections. Multiple ratios were tested, from as low as 1/2 to as high as 9/10, of the difference between the average and maximum, and the results seemed to be optimal at around 2/3. After initialization, the program proceeds to loop through a set of commands until the device is turned off, constantly calculating the BPM using the periods between pulses. The full flow chart can be seen in figure 6.

Figure 4: Pulse Waveform (X-axis: 500ms/div, Y-axis: 2V/div)

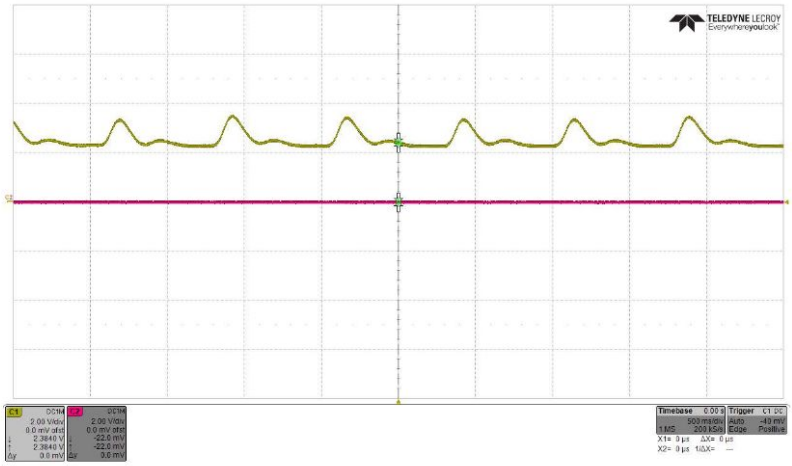


Figure 5: MATLAB Graph of Digital Filtering

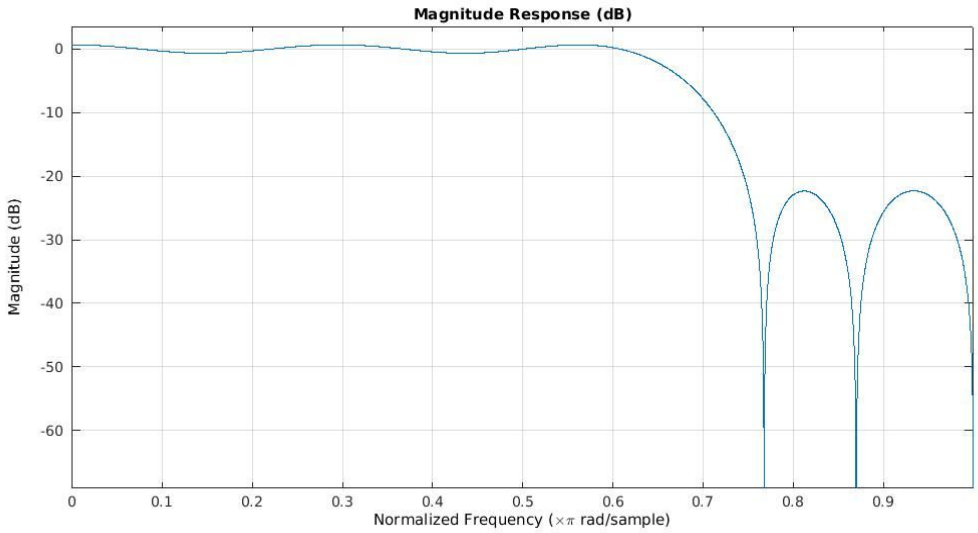
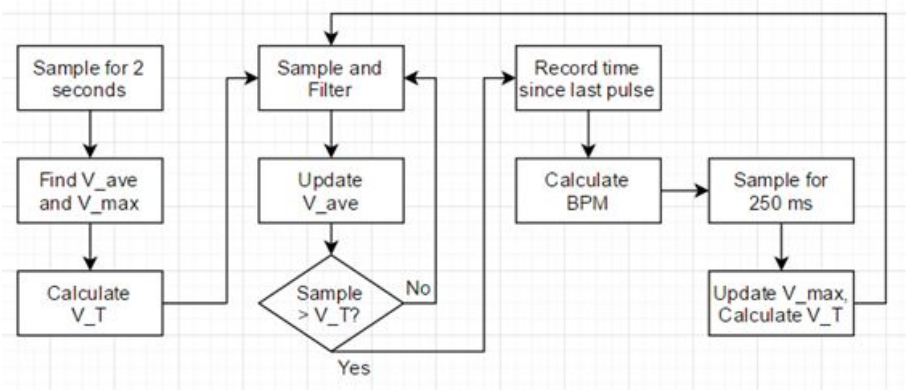


Figure 6: BPM Calculation Flow Chart



2.2 Processing

2.2.1 Microcontroller

The microcontroller unit (MCU) serves as the processing unit, temporary storage, and communications master needed in this design. The MCU we chose was the ATmega328P, which was chosen due to it's simplicity as well as its ability to achieve our desired goals. The only downfall of our MCU choice was that it did not have enough on-chip non-volatile memory to be able to temporarily store the large amounts of data we needed for processing. The accelerometer and microSD card are hooked up to the MCU through SPI protocol. By choosing multiple chip selects, we can use the same SPI data bus to communicate with multiple devices, which is an important feature if we choose to add more accelerometers to our design. When the threshold is detected by the accelerometer, it will send a signal to the microcontroller, which will read the data from the accelerometer's 32-level FIFO buffers, which each contains 32 of its most recent samples. This corresponds to 20ms of data, which exceeds the 8ms pre-collision time suggested by literature to acquire sufficient acceleration data [4]. At the same time, the MCU will pull heart rate samples from the previous 20ms from the cache and time lock it accordingly to the acceleration, and store the results to the micro SD card.

For the next 300 seconds, the MCU will continue to pull data from the accelerometer and heart rate sensor and store to flash memory after processing in real time. At a rate of 1600Hz, or a period of 20ms between each read, the microcontroller must read 32 samples of 3 dimensional points, or 96 16-bit numbers. Then, we need to process the heart rate data. The HR sensor will be sampled at a rate of 800Hz (after the 240 ms cooldown time), or a period of 1.25ms, which corresponds to 16 samples and read cycles per accelerometer read. These 16 samples will be filtered through fast convolution, so 256 cycles are needed for processing. Then, after these data values are calculated, they must be written to flash memory. Writing 32 samples of 88 8-bit characters each requires 1408 cycles. Adding all this together, we need 1776 operations per read, or a grand total of 88800Hz. Since we do not want to saturate the timing but rather aim for 70% usage, 130kHz is the bare minimum for our processor. However, since the accelerometer require at least 2MHz for an output data rate of 1600Hz, we set up our ATmega to communicate at 4MHz to allow for tolerance.

Figure 7: Microcontroller/MicroSD Schematic

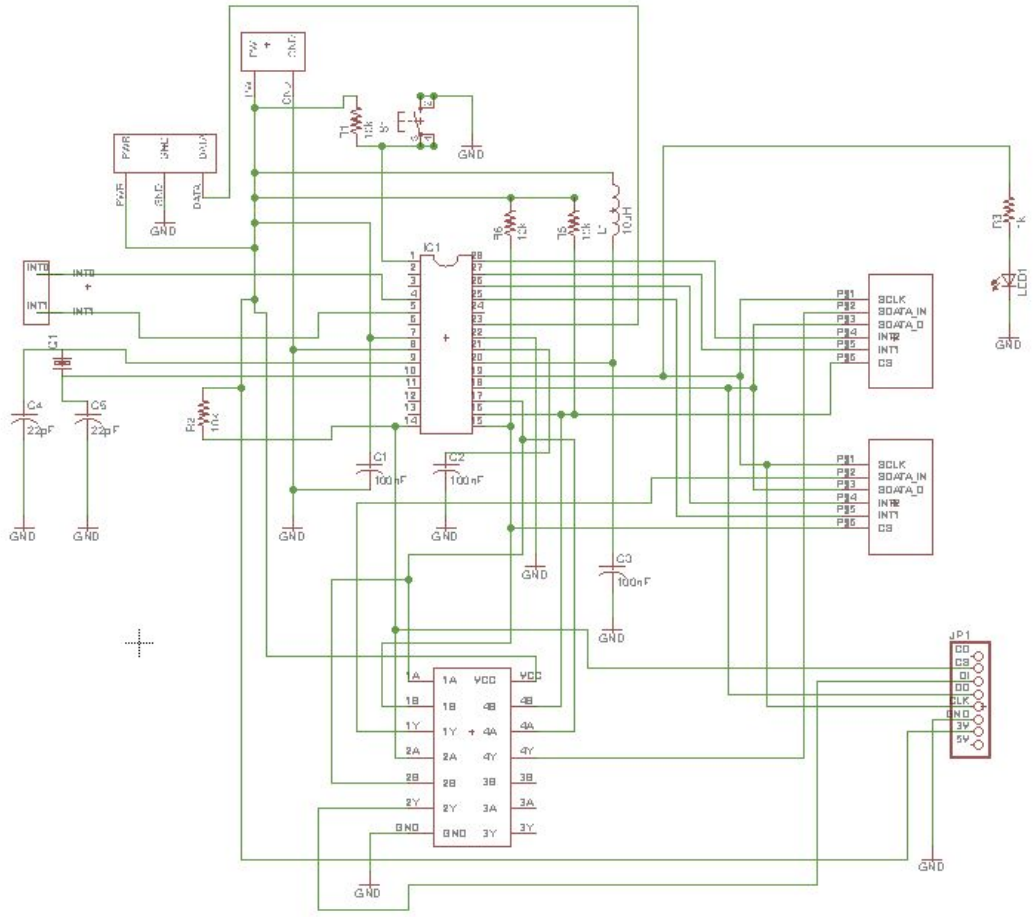
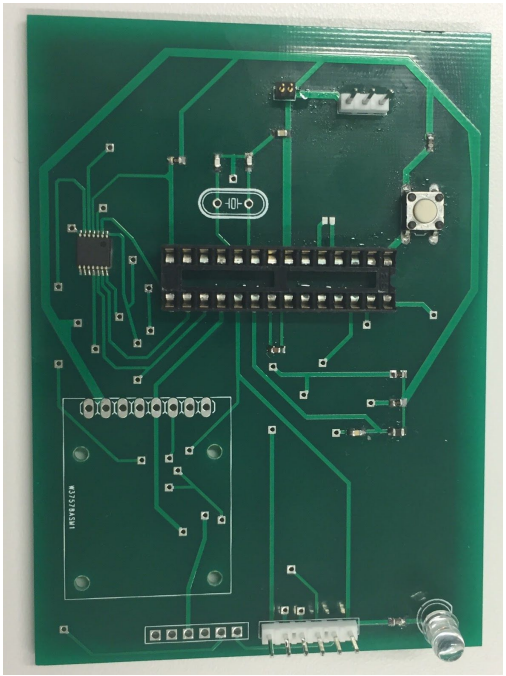


Figure 8: Microcontroller PCB



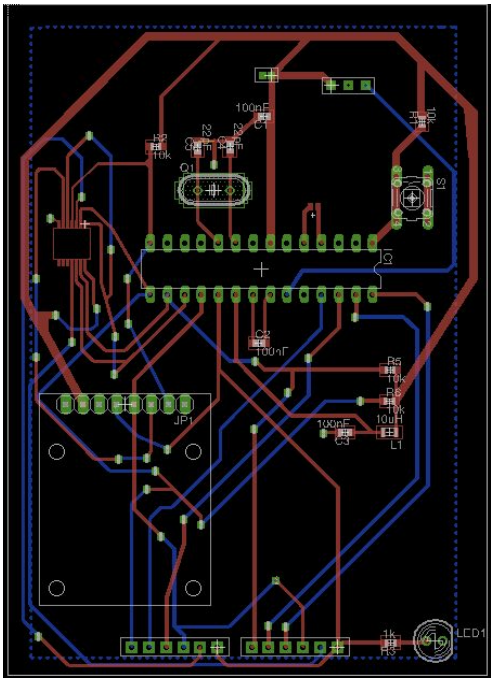
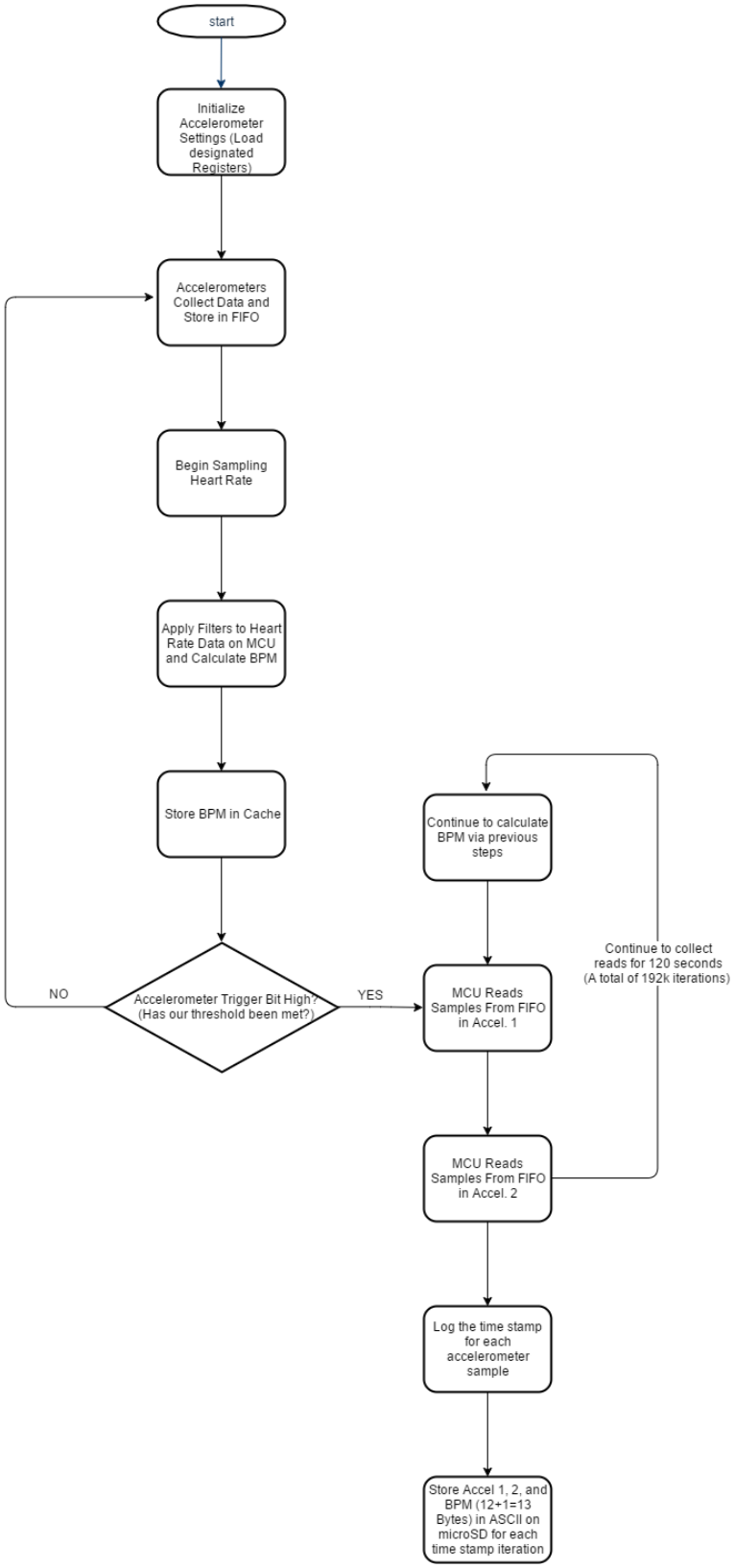


Figure 9: System Flow Chart



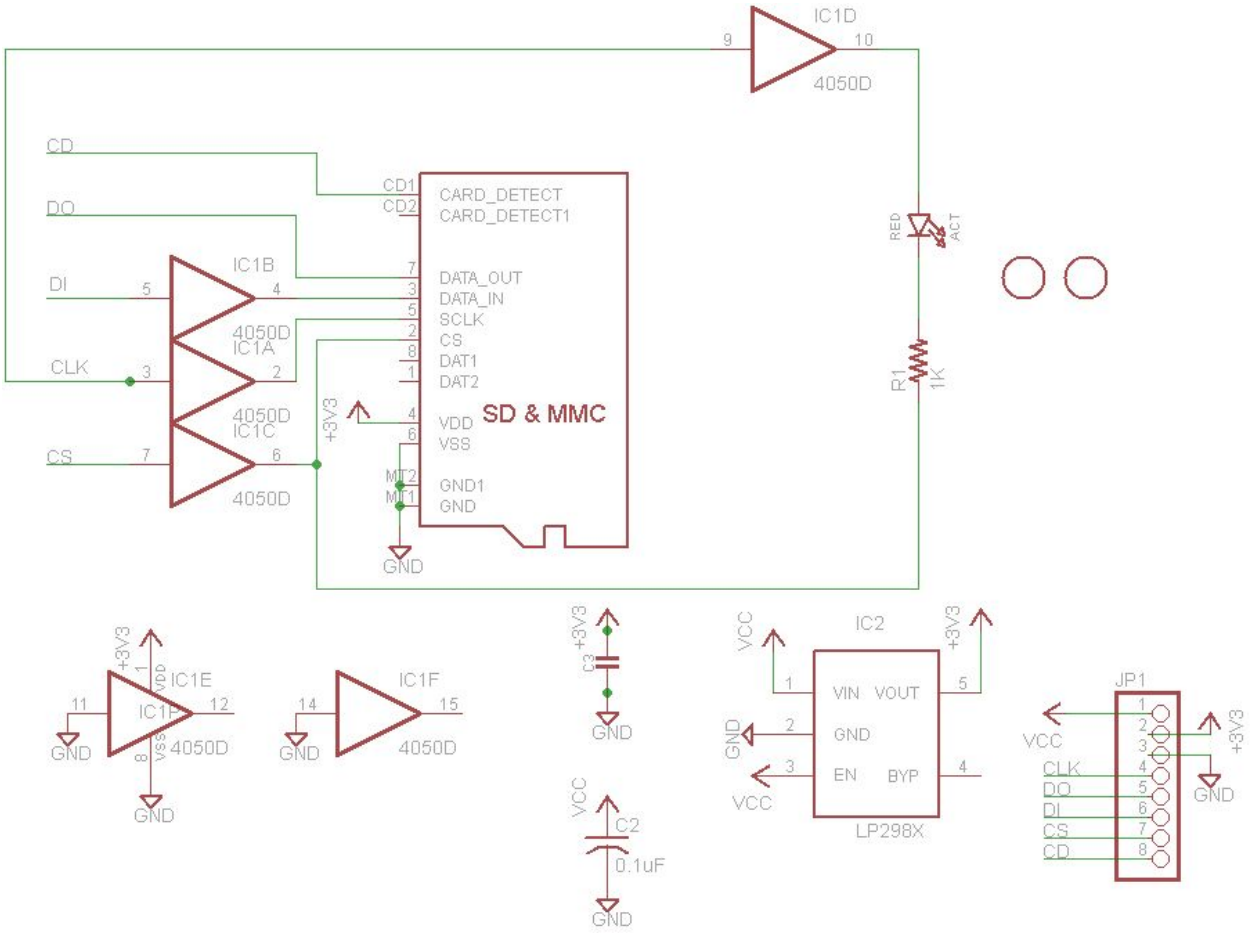
2.3 Data Storage

Given that our device is a datalogging system, storage is a necessary component. The purpose of our system is to store all the information from the sensors to flash memory, so that it may be reviewed at a later time. Adafruit's microSD breakout board was used to interface with a 4GB microSD with FAT16 format. This SD card was incorporated on the same SPI bus as the accelerometers, with a unique chip select wire.

2.3.1 Micro SD card holder

The card is securely held in place and interfaced with using Adafruit's breakout board. This part communicates with the card through SPI protocol. One benefit that the breakout board provided was an onboard chip that automatically converts 5 V intakes to 3.3 V. This is especially important due to the SD card hardware's strict 3.3 V requirements.

Figure 10: MicroSD card holder schematic [8]



2.3.2 Micro SD card

The storage device we are using was a 4GB microSD card. We calculated that 1GB is more than enough storage, but larger and more costly cards are usable. The data was stored in a .txt file, with the sensors' data recorded in rows alongside the date and time in the following format: "hh:mm:ss.sss - A +xxx.xx +yyy.yy +zzz.zz H hhh.hh". The acceleration A stored the dimensional accelerations in the x, y, and z axes sensed by the accelerometer in the units of g-force. The heart rate H was stored alongside the accelerations in units of beats per minute. Since the micro SD was written to only after the threshold acceleration is met, the data was stored in portions of 300 seconds. The microSD was written to in real time, and so a row of 50 ASCII characters was written at a rate of 1600Hz, or a data rate of 80kB/s. The 4GB micro SD card used in our design held enough for 50,000 seconds, or just under 14 hours of data. Since each impact only wrote 300 seconds worth of data, this card can hold information of up to 165 impacts before needing to be replaced or cleared.

2.4 Power

The goal of this device is to secure enough power to endure the entire cycle of an athletic sports game, so a minimum of 3 hours. The accelerometer carries its own CR-2032 coin cells to reduce the number of wires running up and down the body. The power board we designed and completed was able to power the MCU, microSD, and heart sensor. We implemented a power system with the starting supply of two double AA Alkaline (Energizer) batteries in series, with a nominal voltage per battery of 1.5 volts. The MCU, micro SD, and pulse sensor all needed a minimum voltage of at least 3 volts and preferably around 3.3 volts. In the figure below you can see the system we implemented (without the three loads that would be connected in parallel to the 3.3 volt output). Our completed design used a LTC3402 boost converter that took in an input ranging from 1.8v-3v and held a constant steady output of 3.3v. The chip enabled high load current at the output end, which was essential for our MCU and microSD because these two loads drew a significant amount of current (approximately around 100mA-200mA each).

Figure 11: Power Schematic For Processing-Storage-Pulse Sensor Loads [11]

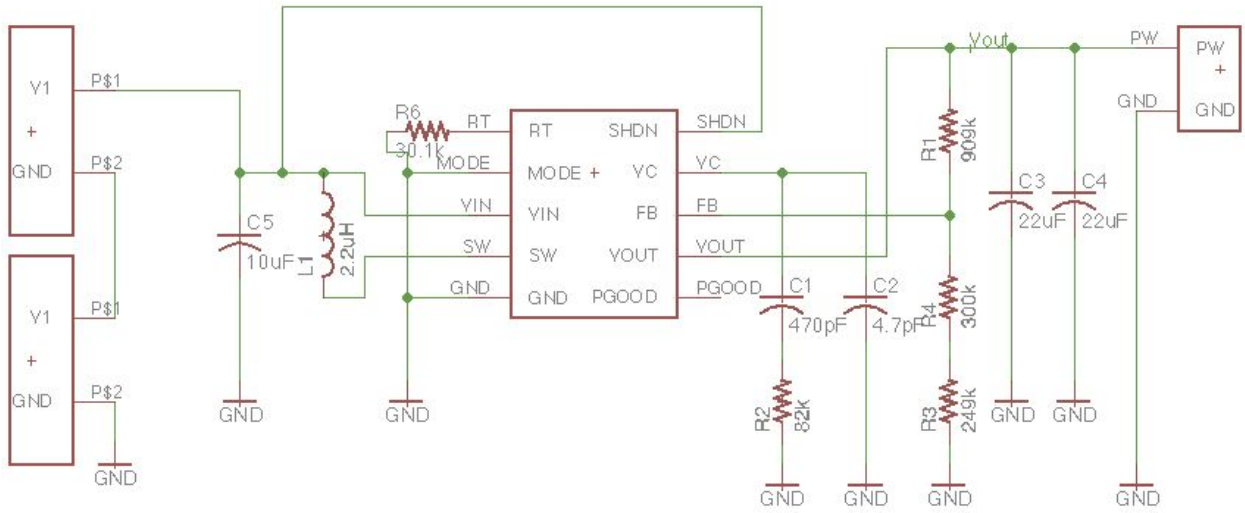
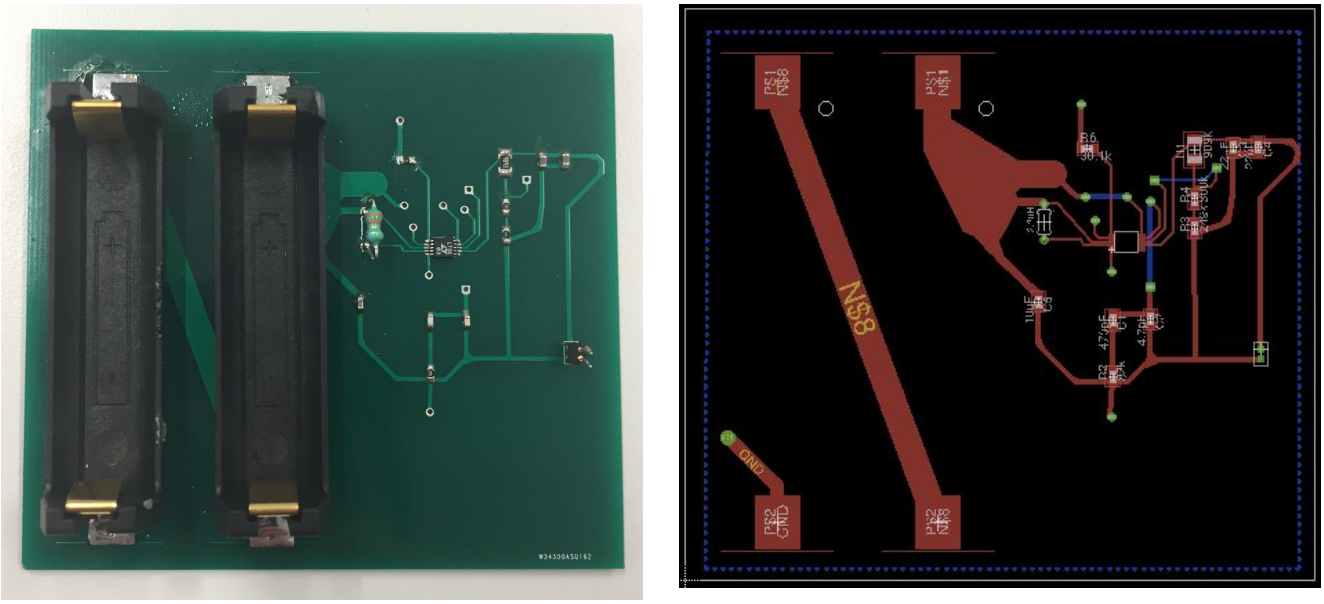


Figure 12: Power PCB



2.4.1 Sensors

The accelerometer we used was powered with its own CR-2032 coin cell. This coin cell carried a charge of 30mAh at 3V. Since the accelerometer ran at 1600Hz, it required roughly 90uA of current at 3V. We chose the ADXL375 accelerometer for several reasons but the benefit of this chip is that when it is not recording data it falls into *sleep mode*. In sleep mode the ADXL375 uses minimal power consumption, which helps conserve the power of our overall design. Using this low power state we were able to reduce the accelerometers power consumption by an additional 60%. Below is a table to show the actual power drawn by the accelerometer when in both high power and low power modes. The actual current draw deviates from the theoretical values by roughly 30%.

Table 2: Accelerometer Low Power Mode Power Savings

	High Power (1600Hz Bandwidth)	**Low Power (8Hz Bandwidth)**
Voltage (V)	3.0 V	3.0 V
Current (mAhrs)	.118 mA	.048 mA
Power (mW-hrs)	.354mW-hrs	.144mW-hrs

The heart rate sensor pulls power from bank of 2 AA batteries that is mounted with the PCB. The heart rate monitor intakes 4mA at 5V, but has an option for a 3V intake. After fully testing the capabilities of the pulse sensor at 3V, we found that there were no drawbacks to using 3V versus 5V. Thus, we powered the pulse sensor directly from the 3.3V regulator, and expect a maximum power consumption of around 13mW.

2.4.2 Processing Unit

It is difficult to predict the power consumption of the microcontroller, but the datasheet claims an absolute maximum current of 250mA. Running at 3.3V, this gives an absolute maximum power consumption of 825mW. Given that we do not run the microcontroller at its maximum speeds, since we do not need its full 8MHz potential in a two accelerometer case, the actual power consumption should be significantly less. The microcontroller pulls power from the battery bank through the voltage regulator.

2.4.3 Storage

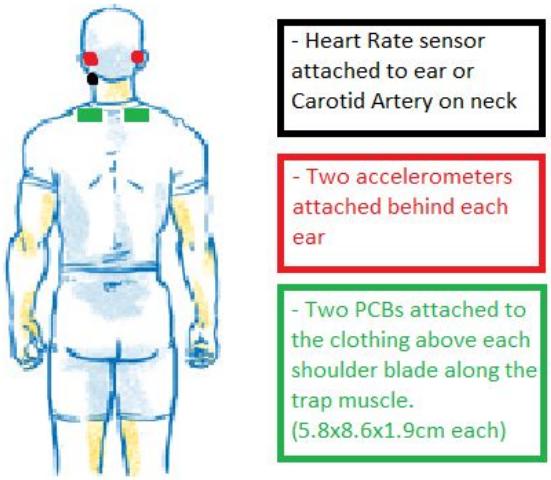
The SD breakout board that we choose to implement has two modes to choose from: default mode, which runs at 25MHz, and high speed mode, which runs at 50MHz. The default mode was used, since the extra speed is not needed. A current draw of approximately 100mA is expected at a voltage of 3.3V, giving a maximum dissipation of 330mW for the storage unit.

2.5 Wearability

The first three blocks in the diagram above consist of our overall wearable design module. The first module (inputs / data collection) consists of two sensors: one accelerometer and one pulse sensor. The wearable design - specifically the three sensors - need to be small enough to fit comfortably yet strongly attached to the individual's skull. Each accelerometer is mounted on its own miniature printed circuit board and powered by a coin cell. The accelerometer is glued or taped on either side of the head, behind the ears, to catch linear acceleration. Our pulse sensor is pinned to the ear with a clip. The HIT Data Logging System would ideally be able to scale to any athlete on this planet, but for the scope of this class we are going to focus on a few essential qualities of the device.

The second thing we are going to focus on is making this wearable design as light as possible. The main printed circuit board is going to contain the microcontroller unit for processing and storage. The power unit is carried on a different platform. Design should allow for player to achieve full range of motion of the head, neck, and upper body. According to research, the optimal location to wear this device would be on the back, on the upper shoulder blades. The units on the shoulder blades should not exceed 5.8x8.6x1.9 cm. To prevent throwing off the player's balance, the entire system should weigh less than 300 grams, or approximately the weight of an iPhone 6 on either shoulder.

Figure 13: Wearable Design Layout



2.6 Tolerance Analysis

The purpose of our device is to timelock accurate sets of acceleration and heart rate data. Since our highest sampling rate is from the accelerometer at 1600Hz, or a period of 625 μs, in order for the data to be considered to be within the same timestamp, an absolute maximum in error would be ∓312 μs. Our device aims to time lock the data within an error range of ∓100 μs, or within a period of 200 μs. Since each read of the accelerometer reads the buffer of 32 samples of 6 bytes each, an 8MHz processor should theoretically read the entire buffer in 12 μs. This means that the accelerometer have no issue in being time locked within the desired error range.

The heart rate sensor proves to be easier to implement within a tolerable error. The trigger mode on the accelerometer sends an interrupt signal once the threshold acceleration is met, which is used to signal the microcontroller. At this moment in time, the current calculated BPM is logged along with the accelerations. Since the BPM is continuously updated in real time with every heartbeat, or approximately every 500 ms, the time lock can take the current BPM at any moment and consider it to be accurate.

3. Requirements and Verification

Accelerometer:

Requirements:	Verification:
(1) Detection: MCU detects and outputs acceleration in x,y, and z directions from accelerometer. (2) Triggers are active/inactive when the accelerometer detects force in x, y, or z directions. (3) Accelerometer enters a low power state upon low active, no movement. (4) Acceleration data is timestamped with time of detection in relation to time of start-up with a deviation no more than 20ms.	(1) On an impact of the accelerometer, output will show data deviation of at minimum 10gs. (2) When any small force of at minimum 10gs is exerted on accelerometer, activity should be detected. (3) When there is no activity for 15s, current draw detected on multimeter decreases by at least 50%. (4) Activity times will be tracked via stopwatch to determine the time elapsed is within 20ms of the stamped value.

Pulse Sensor:

Requirements:	Verification:
(1) Sample HR sensor output at a rate of 800 ± 50 Hz. (2) Heart Rate Sensor must detect 98% of pulses and update BPM accordingly. (3) The pulse sensor remains 98% accurate while the subject is moving.	(1) Use an arduino with timestamped outputs to demonstrate 240,000 ± 15,000 samples in 300 seconds (Amount of time required for our design). (2) We will physically take our pulse through finger-to-wrist method and ensure 98% of pulses felt correspond to a BPM update. (3) The subject will move the head in all 3 rotational degrees of freedom with varying

	speeds, while the finger-to-wrist verification is performed.

Microcontroller:

Requirements:	Verification:
(1) The MCU must be able to store all information from the accelerometers and process heart rate data quickly enough in between each accelerometer read with a 100% sample throughput. (2) Each row of the final stored data must be time locked within a 100 μs window.	(1) We will manually give a triggering concussive blow to a ball with the device attached and ensure that 148,000 ± 30,000 rows are recorded in the storage at the end of 300 seconds. (2) Each row of data must have a corresponding timestamp with a spacing of 625 ± 100 μs.

SD Card Breakout:

Requirements:	Verification:
(1) Using the breakout board gives a 100% accurate writes.	(1) We will write a random data set of size 200 kB to the SD card and use the built in Arduino time functions to ensure that write time is less than 2 seconds. (2) After the above verification, we will read from the SD and show that the inputted and stored data are identical.

SD Card:

Requirements:	Verification:
(1) The SD card must be capable of writing at speeds above 100 kB/s.	(1) We will write a data set of size 16 B to the SD card and use the built in Arduino time functions to ensure that write time is less than 160 microseconds.

Wearability:

Requirements:	Verification:
(1) The x-, y-, and z-dimensions of any unit carried on the shoulder blades should not exceed 10.5 cm, 9 cm, and 2 cm respectively. (2) The entire device must have a mass of less than 500 grams. The PCBs that lie on the athletes trapezius should be at least 95%	(1) The units will be measured with a ruler to determine the dimensions. The device will also be weighed using a scale (2) Borrow weight scale from a lab to verify the weight of each PCB and then measure entire device

similar in weight/size to maintain athlete's balance.	

Power:

Requirements: (1) The Power PCB should output a steady 3.3v output when the input voltage is varied from 1.8v-3v (2) It should be within the size range of 10cmx9cmx2cm	Verification: (1) Connect the Power PCB to a power supply and vary the voltage from 1.8v-3v and show that the output is a steady 3.3v (2) Use a ruler to measure the size of printed circuit board

4. Cost Analysis

Part	Cost
MCU (ATmega328P)	$4
Micro SD	$5
Accelerometer (4 x $8)	$32
Pulse Sensor	$0 (Acquired in ECE 445 Lab)
Lithium Double AA Batteries (pack of 4)	$15
Coin Cell Battery (x2 x $1.50)	$3
Solder Paste	$10
2.2 uH Inductor (Amazon 2-day shipping)	$9
Capacitors	$4
Resistors	$3
TOTAL COST:	**$85.00**

Name	Hourly Rate	Hours	Cost
Matthew Schafer	30 USD/Hr	250 Hrs	7,500 USD
Matthew Hebard	30 USD/Hr	250 Hrs	7,500 USD
Evan Qi	30 USD/Hr	250 Hrs	7,500 USD
Total	**90 USD/Hr**	**250 Hrs**	**22,500 USD**

5. Risk Analysis

The module with the highest risk within our design would be the sensors module. Whereas the other modules would likely have terminal consequences in case of failure, buggy sensors would be much harder to detect and troubleshoot. Despite the verification of the integrity of the sensors before implementing them into the system, continuous impact may cause deterioration. A failure in the sensors would cause the entire system to be useless, as the data it collects may not only be incorrect, but may even sabotage all prior research in the subject. To minimize this risk, we must decrease the probability of failure in the sensor module as much as possible. The sensors have to be secured as tightly as possible in place to prevent wiggling loose in extreme physical impacts.

6. Ethics and Safety

This portion of this design project does not conflict with any of the tenets listed in the IEEE and ACM Code of Ethics [2] [3]. The information logged by the device, once in the hands of the user, may be used in conflict with IEEE Code of Ethics' agreement 3: to be honest and realistic in stating claims or estimates based on available data. As a logging device, our project provides a means to obtain this data, but holds no responsibility in how the user may use this information.?

7. Conclusions

7.1 Results

When it was finally time to test out our design on demo day, we were not able to collect data simultaneously on a single MCU, and needed to collect the data from each component modularly to prove that our design features were working. Our acceleration data was read by the MCU properly and output onto a monitor where it could be read in real time to prove that our design was reading data accurately and quickly. Below is a snapshot of some of the output received on the monitor, printing x,y,z acceleration in gs along with the timestamp at which the acceleration occurred in relation to startup of the device.

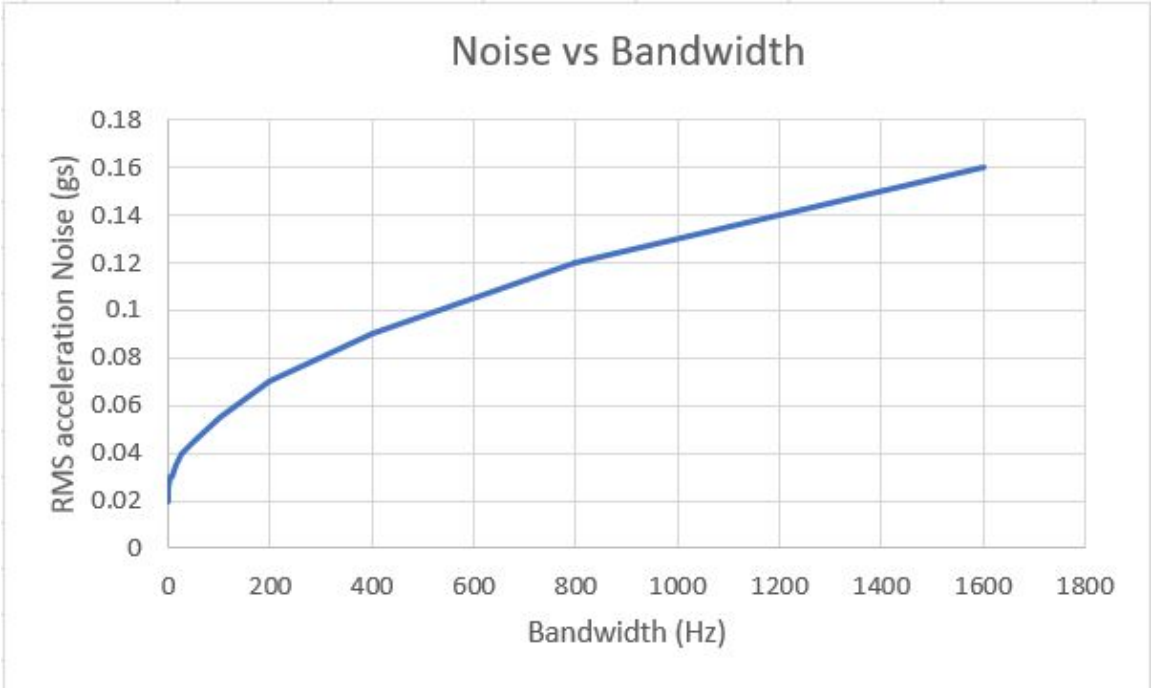
Figure 14: Accelerometer Data Output

```
0.10, 0.00, 0.98 time: 518ms
0.10, 0.00, 1.27 time: 523ms
0.10, 0.00, 0.78 time: 529ms
0.00, 0.10, 1.57 time: 534ms
-7.74, 0.88, -3.14 time: 540ms
*** Impact ***
7.64, -5.88, -2.45 time: 545ms
1.37, -0.69, 5.39 time: 551ms
-0.29, -0.29, 0.39 time: 558ms
-0.78, 0.29, 1.47 time: 563ms
-0.29, 1.57, 0.69 time: 569ms
0.20, 0.00, 0.59 time: 575ms
-0.29, 0.00, 1.67 time: 580ms
0.20, 0.00, 0.98 time: 586ms
-0.39, -0.20, 1.18 time: 591ms
-0.29, 0.49, 1.47 time: 598ms
-0.88, 0.39, 1.76 time: 604ms
```

As you can see in the figure above, at 518ms after program startup both x- and y-axis are outputting stable values around zero with some slight noise detected, while the z-axis data is stable roughly around 1.0 with some slight noise as well. This is exactly what we want when the device is not moving, so we can clearly show that the z-axis is parallel to the force of gravity, which equals 1g. As time passes we see the acceleration of all 3 axis spike when I hit the accelerometer, and an “Impact” message is received. The impact message indicates that an acceleration has occurred beyond our threshold, which we set to be 50gs. The reason these high gs are not recorded (only 7, 5, and 3gs) is because the serial monitor cannot output data as fast as the MCU can store or the accelerometer can read, so it only outputs every 5ms, and by that time the acceleration magnitude has already passed.

One other trend we notice in this data is that there is a large amount of deviation from the theoretical output in this data prior to the impact. The x,y,z coordinates should read (0.0,0.0,1.0) but we see some slight noise even though the device was still at that time. We looked to determine how much noise was actually in the data we obtained and plotted the following graph by sweeping our data collection through different bandwidths.

Figure 15: RMS Noise Density vs Bandwidth Graph



We varied the collection bandwidth from .05 Hz to 1600Hz and calculated the RMS noise values in the figure above. This shows a very nice curve that corresponds well with the ADXL375 datasheet. On the datasheet it specifies that the noise should be roughly proportional to 5 mg/ $\sqrt{Hz}$. In our case, at a bandwidth of 800Hz we should ideally be getting a noise level of 141mg as shown in equation 2 below. In our trials, we see that our noise level actually falls just below that to a level of 120mg. This is a reassurance that our design is accurate.

$$\frac{5mg}{\sqrt{Hz}} * \sqrt{800\ Hz} = 141.42\ \text{mg} \qquad \textbf{[Eq. 2]}$$

From this noise we can move into calculating our signal to noise ratio (SNR), which is the typical scale to evaluate the noise density and signal quality in a signal sensing system. Using the equation below we obtained an SNR of approximately 18dB, which is a very good value for our system.

$$\text{SNR} = 20log(\frac{Signal}{Noise}) = 20log(\frac{1.0}{0.12}) = 18.4\ \text{dB} \qquad \textbf{[Eq. 3]}$$

7.2 Challenges

One of the major challenges we faced was the integration of our power board with the AA alkaline energizer batteries. When we tested the power board with a power supply and changed the input voltage from 1.8v-3v we were getting a correct output of 3.3v. Once the two battery holders were soldered with their corresponding AA batteries we noticed that the batteries in series were outputting a voltage of around 3.2v-3.3v, which clearly exceeds the nominal voltage noted on the data sheets. This system was designed to withstand a maximum voltage of 3v and once this was exceeded the voltage feedback into the LTC3402 was .5v higher than its maximum rating on the LTC3402

data sheet. After several periods of testing this eventually led to the chip being burned out. What we learned is that you can't always trust the data sheet of the product you are receiving. You need to test each part individually to make sure it is working properly.

One key challenge that we faced was the integration of both the SD card and the accelerometer to the MCU. Due to some issues with the library functions in the SD header files, we encountered a bug in which the MISO bus was continuously flooded by the SD card, even while the chip select for the SD card was inactive. This bug was found by probing the MISO wire with an oscilloscope, which showed that the wire was in constant active high. Research on forums revealed that many other users reported this bug, with no official fixes. We had based our entire design around a single SPI bus at this point in our project, so we did not have the time to order a new MCU with two SPI ports. If we had known this issue ahead of time, we could have easily chosen a more appropriate MCU, capable of handling the SD card and the accelerometer simultaneously.

7.3 Future Work

Although this was not explicitly one of our requirements in this design project, we had hopes to make this device scalable, in the sense that any arbitrary number of accelerometers could be hooked up to the MCU for more accurate and detailed data. Since only one accelerometer was used to log acceleration, rotational accelerations are impossible to calculate. A minimum of three accelerometers are required to fully calculate rotational acceleration on all three degrees of freedom - pitch, yaw, and roll - with more accelerometers giving more precise data. We planned to complete this project with two accelerometers, but due to errors in ordering parts, we were forced to make do with only one.

References

[1] Harmon, K. *et al*. "American Medical Society for Sports Medicine position statement: concussion in sport". *British Journal of Sports Medicine.* 47 (1): 15–26. doi:10.1136/bjsports-2012-091941

[2] Ramasamy, V. *et al*. "The Basics of Designing Wearable Electronics with Microcontrollers."

[3] Rijnbeek, P. *et al.* "Minimum Bandwidth Requirements for Recording of Pediatric Electrocardiograms." *Circulation* (2001). 3087-3090. doi:10.1161/hc5001.101063

[4] Wu, L. *et al*. "Bandwidth and sample rate requirements for wearable head impact sensors." *Journal of Biomechanics* **49 (**2016). 2918-2924. doi:10.1016/j.jbiomech.2016.07.004

[5] Rowson, S. *et al*. "Head Acceleration Measurements during Head Impact in Pediatric Populations."

[6] Gemperle, F. *et al.* "Design for Wearability."

[7] Life Span Fitness. Target Heart Rate Calculator: https://www.lifespanfitness.com/fitness/resources/target-heart-rate-calculator

[8] Lee, H. *et al*. "The Periodic Moving Average Filter for Removing Motion Artifacts from PPG Signals." *International Journal of Control, Automation, and Systems* **5** (2007). 701-706.

[9] "Target Heart Rates." Target Heart Rates. N.p., 12 Oct. 2016. Web. 28 Feb. 2017.

[10] Prospero, Mike. "Who Has The Most Accurate Heart Rate Monitor?" Tom's Guide. Tom's Guide, 01 June 2016. Web. 09 Mar. 2017.

[11] Corporation, Linear Technology. "LTC3402." LTC3402 - 2A, 3MHz Micropower Synchronous Boost Converter (n.d.): n. pag. Http://cds.linear.com/docs/en/datasheet/3402fb.pdf. Linear Technology. Web.