

The University of Akron

IdeaExchange@UAkron

Williams Honors College, Honors Research
Projects

The Dr. Gary B. and Pamela S. Williams Honors
College

Spring 2021

Design Project: Smart Headband

John Michel
jam544@uakron.edu

Jack Durkin
jdd120@zips.uakron.edu

Noah Lewis
nl50@zips.uakron.edu

Follow this and additional works at: https://ideaexchange.uakron.edu/honors_research_projects



Part of the [Electrical and Electronics Commons](#), [Hardware Systems Commons](#), [Numerical Analysis and Computation Commons](#), [Numerical Analysis and Scientific Computing Commons](#), [Other Computer Engineering Commons](#), [Other Computer Sciences Commons](#), [Probability Commons](#), [Systems and Communications Commons](#), and the [VLSI and Circuits, Embedded and Hardware Systems Commons](#). Please take a moment to share how this work helps you [through this survey](#). Your feedback will be important as we plan further development of our repository.

Recommended Citation

Michel, John; Durkin, Jack; and Lewis, Noah, "Design Project: Smart Headband" (2021). *Williams Honors College, Honors Research Projects*. 1396.

https://ideaexchange.uakron.edu/honors_research_projects/1396

This Dissertation/Thesis is brought to you for free and open access by The Dr. Gary B. and Pamela S. Williams Honors College at IdeaExchange@UAkron, the institutional repository of The University of Akron in Akron, Ohio, USA. It has been accepted for inclusion in Williams Honors College, Honors Research Projects by an authorized administrator of IdeaExchange@UAkron. For more information, please contact mjon@uakron.edu, uapress@uakron.edu.

Smart Headband

Senior Project Final Report

Design Team #10

John Durkin (EE)

Noah Lewis (CpE)

John Michel (CpE)

Faculty Advisor: Osama Alkhateeb

Date Submitted: April 23th, 2021

Table of Contents

Abstract	4
1. Problem Statement	4
1.1. Need	4
1.2. Objective	5
1.3. Background	5
1.4. Marketing Requirements	14
2. Engineering Analysis	14
2.1. Circuits	14
2.2. Electronics	17
2.3. Communications	19
2.4. Embedded Systems	21
2.5. Numerical Computations	22
3. Engineering Requirements	28
4. Engineering Standards	30
5. Accepted Technical Design	31
5.1. Hardware Design	31
5.1.1. Level 0 Block Diagram	31
5.1.2. Level 1 Block Diagram	32
5.1.3. Level 2 Block Diagrams	35
5.1.4. Printed Circuit Boards	41
5.1.5. Proto Board	44
5.1.6. Testing Apparatus	46
5.2. Software Design	48
5.2.1. Level 0 Block Diagram	48
5.2.2. Level 1 Software Behavior Models	50
5.2.3. Level 2 Software Behavior Models	57
5.2.4. UART Communication	70
5.2.5. Bluetooth Communication	71
5.2.6. SPI Peripheral Communication and Peripheral Operation	74
5.2.7. MicroSD Communication	75
5.2.8. Mobile Application and Matlab Script	79
5.3. Testing	83
6. Mechanical Sketch	85
7. Parts Lists	87
7.1. Parts List	87
7.2. Material Budget List	87
8. Project Schedule	93
9. Team Information	96
10. Conclusions and Recommendations	97
10.1. Future Implementation	98
10.1.1. Step Detection	98
10.1.2. SQLite Database Structure	99

10.1.3. Battery System	101
10.1.4. Impact Data File Read and Bluetooth Communication Integration	102
10.1.5. Mobile Application and Matlab Script Integration	102
10.2. Team Dynamics	103
11. References	104
12. Appendices	106
12.1. MPLAB X Configurations and Code	106
12.2. Matlab Code	126

List of Figures

Figure 5.1.1: Level 0 Block Diagram of the system's hardware.	31
Figure 5.1.2: Level 1 Block Diagram of the system's hardware.	32
Figure 5.1.3.1: Level 2 Block Diagram of the power distribution system.	35
Figure 5.1.3.1.1: Voltage Regulator Schematic.	36
Figure 5.1.3.1.2: Battery Charging Circuit.	37
Figure 5.1.3.2: Level 2 Block Diagram of the concussion sensors.	38
Figure 5.1.3.3: Level 2 Block Diagram of the microcontroller.	38
Figure 5.1.3.4: Schematic for the microcontroller.	40
Figure 5.1.4.1: Second and third PCB.	41
Figure 5.1.4.2: Board design for the microcontroller.	42
Figure 5.1.5: Protoboard implementation.	45
Figure 5.1.6: Testing apparatus.	46
Figure 5.2.1: Level 1 behavior model for step and impact data.	48
Figure 5.2.2: Level 1 Block Diagram of the system's software.	50
Figure 5.2.3.1: Level 2 behavior model for the data processing subprocess.	57
Figure 5.2.3.2: Level 2 behavior model for the data transmission subprocess.	61
Figure 5.2.3.3: Level 2 behavior model for the computation subprocess.	64
Figure 5.2.4: UART1 configuration settings for the PIC24FJ1024GB610.	70
Figure 5.2.5.1: Bluetooth module EEPROM configuration, as shown in the UI tool.	71-72
Figure 5.2.6.1: MCC SPI1 bus configuration.	74
Figure 5.2.7.1: MCC FatFs library configuration.	76
Figure 5.2.7.2: MCC SPI2 library configuration.	77
Figure 5.2.7.3: MCC SPI master library configuration.	77
Figure 5.2.7.4: MCC SD SPI library configuration.	78
Figure 5.2.8.1: Matlab script result using a botched dataset.	81
Figure 5.2.8.2: Matlab script result using a recorded dataset.	82
Figure 5.3.1: Angular acceleration test.	84
Figure 5.3.2: Linear acceleration test.	84
Figure 6.1: Headband Side View	86
Figure 6.2: Headband Top View	86
Figure 10.1.2.1: SQLite database schema for the mobile application.	99
Figure 10.1.2.2: Creation of the SQLite tables in the mobile application.	100

Figure 10.1.3: Coulomb Counting Circuit	101
Figure 12.1.X: MPLAB X project configurations and code.	106
Figure 12.2.1: Matlab script code.	126-129

List of Tables

Table 2.1.1: Component Power Usage	15
Table 2.2.1: Accelerometers	18
Table 2.2.2: Comparison of Gyroscopes	19
Table 2.4.1: Comparison Between Bluetooth Modules	22
Table 3.1: Engineering Requirements	28
Table 4.1: Engineering Standards	30
Table 5.1.1.X: Hardware Level 0 Functional Requirements	31
Table 5.1.2.X: Hardware Level 1 Functional Requirements	32
Table 5.1.3.X: Hardware Level 2 Functional Requirements	35
Table 5.1.4.1: Microcontroller Pin Configuration	43
Table 5.2.1.X: Software Level 0 Functional Requirements	49
Table 5.2.2.X: Software Level 1 Functional Requirements	51
Table 5.2.3.1.X: Software Level 2 - Microcontroller Data Processing	58
Table 5.2.3.2.X: Software Level 2 - Microcontroller Data Transmission	62
Table 5.2.3.3.X: Software Level 2 - Matlab Script Computation	65
Table 5.2.5.2: Bluetooth module status indication.	73
Table 7.1: Parts List	87
Table 7.2.X: Materials Budget List and Parts Orders	87
Table 8.X: Gantt Charts	93

Abstract

Concussion in sports is a prevalent medical issue. It can be difficult for medical professionals to diagnose concussions. With the fast pace nature of many sports, and the damaging effects of concussions, it is important that any concussion risks are assessed immediately. There is a growing trend of wearable technology that collects data such as steps and provides the wearer with in-depth information regarding their performance. The Smart Headband project created a wearable that can record impact data and provide the wearer with a detailed analysis on their risk of sustaining a concussion. The Smart Headband uses accelerometers and gyroscopes to record linear and angular acceleration. The data is evaluated in Matlab using various statistical techniques. The data is displayed to the user along with a concussion risk percentage. The result of the project is a prototype device that can record impacts and provide the user with in-depth concussion risk assessment. [JD, JM]

1. Problem Statement

1.1. Need:

For many athletes, it is essential to track their health and performance while working out. Many modern wearables or applications are uncomfortable to use and/or do not provide the required metrics in real time. According to a study by Endeavor Partners, approximately one third of people who own a wearable stopped using it within six months of purchase. This indicates that athletes require a new wearable solution for fitness tracking. High impact sports require a wearable that can both detect concussions and provide step rate and distance tracking. According to the Center for Disease Control and Prevention, it is estimated that there are roughly

300,000 concussions sustained a year in high school sports. There is no commercial solution that simultaneously handles concussion detection and workout metrics. [JD, NL, JM]

1.2. Objective:

The objective of this project is to design and prototype a headband device that provides a new option for performance wearables. This headband will provide step and concussion detection. For concussion detection, the headband will be able to sense strong impacts. The hardware will be removable and able to withstand impacts on the maximum magnitude of the detectable range. The headband will connect wirelessly to a mobile application that will provide the data tracked by the headband. This data will be viewable through a mobile application; additionally, after a session started by the user, the application will provide a summary of the data from the session. This summary will include approximate concussion risk and step count. The device will be able to alert the user of large forces requiring immediate medical examination in near real time. [JD, NL, JM]

1.3. Background:

This project requires extensive research into several relevant topics to meet the needs and objectives outlined above. The basic theory, current concepts, and limitations of existing technologies must be examined. The research required can be split into separate subcategories outlined in the block diagram: impact and concussion detection, step and distance tracking, power and physical components for the headband hardware, communication of relevant data, and the mobile application. Finally, current patents and designs must be compared to the concepts outlined in the proposed design for this project. [JM]

At the software level, the accelerometer data must be interpreted into different variables to assess concussion risk. Such variables would be some form of acceleration such as linear or angular acceleration. Using force as a metric would prevent normalization of the data for different head masses, so acceleration must be considered (Kleiven). Other measurements that consider different weightings of types of acceleration may be used to better estimate impacts that may cause a concussion. Weighted variable(s) would be determined through principal component analysis, a technique to reduce correlated variables to fewer uncorrelated variables (Jackson). To detect concussions, the impact data would need to exceed a given threshold specified for the data type chosen. Such a threshold would be determined through experimental data. The threshold chosen would be designed to detect a given percentage of concussions while limiting the rate of false positives. Impacts detected above the threshold would be classified as “at risk” impacts and would be tracked to notify the user of their concussion risk. [JM]

Current literature suggests using linear and angular acceleration as the base data for impact detection. Linear acceleration is the change in magnitude of velocity of the head; this is often a more effective predictor for skull fractures than for brain injury (Kleiven). Additional research by King et al. suggests that linear and angular acceleration are causes of strain on brain tissue, causing brain injury. Each linear and angular acceleration should be measured either at one point on the head, or at several locations on the head to increase accuracy. For example, one study used six single axis linear accelerometers “that recorded an acceleration time history of the head center of gravity” (Greenwald et al.). Current implementations of concussion detection further suggest resultant linear acceleration (resultant LA), resultant angular acceleration (resultant AA), the Head Injury Criterion (HIC), and the Gadd Severity Index (GSI) as appropriate measures to detect concussion risk (O’Connor et al.). According to O’Connor et al.,

“Resultant LA is the vector sum magnitude of the 3-dimensional LAs of the skull resulting from an impact.” Similarly, resultant AA is the vector sum magnitude of the 3-dimensional angular accelerations, measured in radians per second squared. The HIC is a measure described by the following equation:

$$HIC^* = (t_2 - t_1) \left[1/(t_2 - t_1) * \int_{t_1}^{t_2} a(t)dt \right]^{5/2} \quad (1)$$

Here, t_2-t_1 should be less than or equal to fifteen milliseconds for accuracy, and $a(t)$ is the linear acceleration of the head center of gravity measured in gs (gravitational accelerations) (Gao and Wampler 72). The GSI is a commonly used measure of injury in sports industries; it is described by the following equation:

$$GSI = \int_0^T a(t)^{5/2} dt \quad (2)$$

Here, T will be the same time interval used in the calculation of the HIC, and $a(t)$ is again the resultant linear acceleration during the time interval. Note that the GSI is highly correlated with the HIC. [JM]

A patent from Benzel et al. suggests organizing input data into several parameters that can be categorized: “At least one of a linear acceleration, an angular acceleration, an angular velocity and an orientation is measured at a first location on an athlete participating in the sporting event... A plurality of impact parameters are calculated from the determined acceleration at the location of interest. The calculated plurality of impact parameters are associated with an injury class of a plurality of injury classes.” A study by Greenwald et al. used principal component analysis to create an effective weighted sum of resultant LA, resultant AA, and HIC. [JM]

Concussion risk can be estimated by one or more of the variable predictors mentioned.

The predictor selection should also account for false positives. The false response rate of a detection variable is the rate at which non-concussive impacts are detected as concussive impacts. Research from Greenwald et al. utilized a Receiver Operating Characteristic (ROC), a “curve that defines the relationship between sensitivity and 1-specificity for each biomechanical measure” where sensitivity is the correct prediction level and 1-specificity is the false response rate. The area under the ROC curve measures “the probability that a randomly selected concussive impact will have a higher severity score, than a randomly selected non-injurious impact” (Greenwald et al.). If the area is 0.5, the predictor is no better than guessing. However, if the area under the curve is 1.0, the maximum, that would indicate a perfect predictor with no false positives. The statistical significance of the difference between the ROC curve area and the area obtained from guessing (0.5) can be measured via significance values (p-values). The study by Greenwald et al. found that all predictors tested (resultant LA, resultant AA, HIC, and a principal component analysis of the three prior variables denoted wPCS) were significant predictors of concussion for all impact data ($p < 0.001$ for all predictors). [JM]

For practical applications, selecting arbitrary threshold(s) based on experimental data can be used as a general predictor for concussion risk. For example, in a study of $n = 289,916$ impacts, “the threshold or impact severity associated with the 75% correct prediction level was 96g for linear acceleration, 7,235 rad/sec for rotational acceleration, and 160 for HIC” (Greenwald et al.). Hence, in this case, for linear acceleration, setting a threshold of 96g would detect 75% of concussions found by medical experts across the sample size. At this level, the study found that a threshold at 75% correct prediction level would cause anywhere between a 0.78% (wPCS) and 2.47% (resultant AA) false response rate for the predictors tested. The study also found that in general, higher correct prediction levels resulted in a higher false response rate.

Thus, in the final design of a concussion detector, the tradeoff between correct prediction level and false response rate must be considered. [JM]

There are several limitations to the design of a concussion detector. First, the accuracy of the predictors discussed decreases with higher magnitudes of impact. In the study by Greenwald et al., the predictors were accurate for all data and the top 5% of all data ($p < 0.001$ for all). However, the data was limited to the top 2% and 1%, only wPCS remained as a statistically significant predictor ($p < 0.001$ and $p = 0.004$, respectively). According to Greenwald et al., “the top 1%, 2%, and 5% of all impacts contained 11, 15, and 16 of the 17 diagnosed concussions, respectively.” Hence, the accuracy of the predictors for the top percentages of data is important to consider when choosing a predictor. Even though the predictor wPCS was a statistically significant predictor of concussion, the accuracy of the current concussion detection technology is not close to what is required for concussion diagnosis. The positive predictive value (PPV) of each predictor is the percentage of positive detections that were also diagnosed as concussions. The PPV of each predictor was extremely small: “The highest PPV for linear acceleration was 0.3% at the 50% correct prediction level. At this correct prediction level PPV for wPCS was 0.9%” (Greenwald et al.). This shows that current measurement techniques are still not nearly enough to achieve an accurate medical diagnosis of a concussion. Further limitations are examined in a review by O’Connor et al. One limitation to concussion detection technology that they examined was in the impact measurement techniques: “Currently, risk estimation for concussion has predominantly used acceleration variables to calculate risk; adding more biomechanical measures may improve the specificity of identifying concussive impacts” (O’Connor et al.). The researchers also suggest that current research may not be generalizable. Most studies on concussive impacts have “largely focused on high school and collegiate

athletes” (O’Connor et al.). Furthermore, female athletes were rarely considered in the studies examined. [JM]

The distance tracking and step count will be generated using the accelerometer data. When someone is in motion, the accelerometer is able to pick up the variable linear forces that are caused by the motion of someone's body in relation to their stride. The frequency of these peaks of force in the accelerometer’s data can be used to determine how fast the person was in motion. With additional information such as user height and leg length, an accelerometer can be used to accurately model how many steps a user has taken. Distance tracking using an accelerometer is more suited to a single mechanic of motion. In a study done on the accuracy of pedometers on the market, it was found that the Nike + Fuelband, a wrist band designed to monitor distance and steps for track and cross-country athletes, had an error rate of 36.2% when at walking pace. The Movemeter, a device designed for walking speed, had a much lower error rate of 0.9% when measuring at walking speed. (Storm et al.). Distance tracking is more accurate along linear paths of progression and can experience some error when there is a high rate of variability in someone's stride or direction. The nonlinear nature of many contact sports will lead to some inaccuracy when a single accelerometer is the only data point used, but it should suffice in providing an approximate value of distance traveled. The goal of this feature is to provide an estimate to the distance an athlete has covered at an elevated work rate. [JD]

Wearable design requires a method of data processing and often an associated mobile application to display the results. The wearable device collects the data via methods discussed earlier, and the data is then sent through a Bluetooth transmitter to the receiver in the linked mobile device. The resulting data is then saved to the mobile device’s storage, often a virtual cloud storage or the mobile device’s internal storage. From there, the application would receive

the data, and display the data dependent on the given variables and the time variant. Many recent articles also discuss the ability to share the collected information with medical professionals to assist in the analysis of the damage and set up for recovery, if needed. [NL]

There are many examples of wearables collecting data for health reasons in society today. In these examples, there are two types of data that is collected: data that is recorded when it goes over a certain threshold (as discussed earlier, the concussion sensors would be one example) and time dependent data. In June of 2018, a smart headband was used to track epileptic seizures, where the sensors would be tripped if the electroencephalogram, or EEG, signal were to go above a certain threshold. When the epileptic seizure detection tag (ESDT) is tripped, the system sends the signal raw data via a Bluetooth using a Bluetooth low-power chip to the mobile device. Then the application sends an alert to medical personnel, who dispatch an ambulance if needed. The EEG signal is also recorded every 30 to 60 seconds to monitor the seizure until it ends (Lin et al.). [NL]

Another case at the University of Washington Bothell, Washington used a device that wrapped around the user's hand and measured the user's emotional state. The existing hardware would record the user's emotions continuously (with a few second latency between the stimulus that caused the reaction and the response), and the data would be sent with the Bluno Beetle microcontroller, that converts the data and sends the analog signal to the android phone with Bluetooth with a connection interval of 10ms. The data is then automatically saved to the phone's internal storage, and a prototype application would display a scrollable graph to view the incoming data. From there, the article discussed that the device could record emotions while playing video games or sleeping (Lam and Szypula). [NL]

This project will consider currently existing patents and designs that relate to the

concepts proposed in this document. In the patent “Communication system for impact sensors” one variation includes a mouthguard with an attached accelerometer and gyros. The product is designed for football helmets and records acceleration data and sends a signal if an acceleration threshold is surpassed. The project intends to use a similar approach of measuring acceleration of the head as the means of concussion detection but with using different components and at a different data point. The patented design also makes use of wireless connections to provide data related to each player's impact data. The difference between the design of the patent and this project will be the mechanical systems measuring the acceleration, and the placement of the device as a headband to allow for a wider set of uses. [JD]

The patent “Sports headband to reduce or prevent head injury” by Victor Domingos outlines a headband concept that is designed to reduce concussion risk. Domingos proposes a multi-layer headband with a hard elastic outer layer, an intermediate layer with coolant, and an inner layer “composed of a material comprising cotton, rubber, nylon, and elastic.” This project will consider a headband of similar design, likely excluding one or more of the layers proposed. The concussion and step tracking hardware will be embedded in the headband and designed to be situated at a certain location on the head (either to the side, front, or back). Intellectual property concerns should not be an issue as this patent is currently expired. [JM]

This project will also consider the concussion detection variable proposed in “Head Impact Severity Measures for Evaluating Mild Traumatic Brain Injury Risk Exposure,” a study by Greenwald et al. This study uses principal component analysis to obtain an effective weighted sum, denoted ‘PCS’, of linear acceleration, angular acceleration, HIC and the Gadd Severity Index. Using experimental data, the researchers calculated a new weighting for PCS based on impact location (side, back, front, or top). This new weighted variable, denoted wPCS, “had the

lowest false response rate of all the measures evaluated” (Greenwald et al.). This project will use a similar measure. Since the original study assessed impact location with video footage, this project will use PCS or a similar weighted sum that does not account for location of impact. [JM]

The article written in part by Shih-Kai Lin, discussed a headband that would detect epileptic seizures when the EEG signal parameter went above the threshold. This is similar to the concussion sensors that would record the data when the threshold is passed. However, this project's design will have a better power source compared to Lin's, which used a coin battery that lasted only two hours. The data for recording the step rate over real time in certain intervals also compares to the paper partially written by Lawrence K Lam, which recorded emotions during specific activities. Lam's project is comparable to this project since this project will use the mobile device's internal storage for the raw data; furthermore, the headband project will be implementing a similar microcontroller to the Bluno Beetle microcontroller they used to transmit the data via Bluetooth. [NL]

1.4. Marketing Requirements:

1. The headband device must detect high acceleration impacts.
2. The headband device must track the number of steps taken across a user session.
3. The headband device must be durable enough for use during workouts or contact sports.
4. The headband device must be able to operate without recharge across an entire use session.
5. The headband hardware must be comfortable and removable from the headband itself.
6. The headband device must communicate with a mobile application that can access and analyze the recorded data.
7. The mobile application must be able to assess concussion risk based on the impacts recorded in the user session.
8. The mobile application must display step rate, concussion risk, and the potentially concussive impacts from the user session.

2. Engineering Analysis

2.1. Circuits

2.1.1. Battery

The battery must power the headband for a duration of at least 4 hours and should be rechargeable. The device requires different voltage ranges and current draw for each device type. Table 2.2.1 has all the power constraints for the components used on the headband.

The power supplied to each of the components will be 3.3V. The controller will require the most power and has a wide current draw range of 18mA to 200mA. If the controller is running at maximum capacity, the battery will require 1140mAh. If the controller is running at minimum capacity the battery will require 412mAh. [JD]

Table 2.1.1: Component power usage					
Device	Current	Min Voltage	Max Voltage	Number of components	Power usage (3.3V)
PIC24FJ1024 GB610	18mA To 200mA	2V	3.6V	1	59.2mW
ITG-3701	3.3 mA	1.71V	3.6V	3	32.67mW
BM78	12 mA	3.3V	4.4V	1	39.6mW
ADXL-372	20 mA	1.6V	3.5V	3	198mW
LSM9DS1	4 mA	1.9V	3.6V	1	13.2mW
LTC1517-3.3	6uA	2V	4.4V	3	79uW
BQ26220	30uA	0.7V	7V	1	132uW
Total	285mA				940.5mW

Lithium ion battery provides a high-power density design and comes in smaller sizes. Due to the size constraints of the headband, a smaller lithium ion battery will be used. For the size of the headband, the voltage output will be 3.7 volts with a range of up to 1100mAh of charge. The battery will weigh 20g and have a dimension of 40.05mm by 41.5mm by 6.1 mm. To protect the battery there will be lithium ion batteries that have micro JST connectors and the batteries will be charged through a USB connection. [JD]

The components each have varying power consumption requirements. One thing that they all have in common is that they can operate at 3.3V. This is the optimal operating voltage for the accelerometers, gyroscopes, and IMU. To supply 3.3V to each component, a switching voltage regulator will be used. The LTC1517-3.3 will be used for the design. The device is rated to produce a 3.3V output. It can increase or decrease the voltage as required, has an extremely low

power usage rating. The device does experience some voltage drops caused by large current pulls. To mitigate these effects there will be 3 separate voltage regulators. One will supply power to the PIC24FJ1024GB610, and the other 2 will be split up between the accelerometers, gyroscopes, IMU, and Bluetooth connection. [JD]

The battery will require a charging circuit. This circuit will provide power to the battery through a USB cable. USB connections provide 5V DC power. The battery charging system will take use of this charge and apply it to the JST connection of the battery. Using a MCP7383 battery management controller, the 5Vs of powers supplied by the USB will be dropped down to 4.5V. This value will be put across the battery. A diode is placed in between the battery and the voltage regulators. The USB V+ pin is connected to the other side of the diode. This will cause the diode to reverse bias and stop any current flowing from the battery. Additionally, a physical switching component, PCM12SMTR, will be used to turn the device on and off. The switch will have an open circuit to stop any current flow from the battery, while allowing for charging. [JD]

It is important that the state of charge of the battery can be viewed by the user. The BQ26220 measures the resistance across the SRP and SRN. The device uses a clock cycle to take these measurements. The changes in voltage are monitored and the state of charge will be sent to monitor the charge of the battery, a coulomb counting device will be used. The state of charge will be communicated to the controller using an I2C digital signal sent from the HDQ pin. The battery management system schematics can be found in section 5.1.3.1. [JD]

2.2. Electronics

2.2.1. Accelerometer

Micro electro-mechanical accelerometers (MEMS) measure the vibration/acceleration they experience. They have a mass of piezoelectric material that when a force is applied, they produce a proportional electric charge. For the use of impact detection, it is important that the device is able to sample at a rate that will record enough data points to accurately reconstruct the impulse signal. In a Journal of Athletic Training study, it was found that 95% of all concussion impulses last between 5.5 and 13.7 milliseconds. Using the minimum impulse duration of 5.5 milliseconds and 20 samples per impulse, the sampling frequency of this input would be 3636.36 Hz. The bandwidth of the device needs to be half of the sampling frequency, which is 1818.18 Hz. To accurately sample this impulse the step rate has a maximum frequency of 3 strides/second, so an accelerometer with a lower frequency of sampling can cut out faster impulses that relate to impacts, while still measuring the stride of the wearer. The sampling frequency of this accelerometer would need to be at least 60 Hz, with a required device bandwidth of 30 Hz. [JD]

In a study by Greenwald et. al. on nearly 300,000 head impacts, only one percent of all recorded impacts exceeded 103.4g of resultant linear acceleration. Furthermore, of the impacts that led to a concussion diagnosis, approximately 90 percent had a resultant linear acceleration of less than 150g. The accelerometers on the headband that are used for impact detection will have a recording range of at least 150g of linear acceleration to have a range of measurements that contains a near entirety of concussion causing impacts. The accelerometer will transmit data only when the linear acceleration on any axis exceeds a given minimum threshold; this will drastically decrease the amount of data sent to the microcontroller. [JD, JM]

These accelerometers can have dimensions as small as 1mm x 1mm x 2mm and can require a power draw of around 0.054W at 3V and 18mA. This power draw is small enough that multiple accelerometers can be placed on the headband without creating an issue of power usage. Multiple accelerometers will be placed on the headband to increase the accuracy of the device. The MEMS accelerometer that will be used will measure three different forces in the X, Y and Z directions. With this data the device will be able to orient the direction and part of the head that sustained the force. MEMS accelerometers have multiple types of data outputs. For this headband it will be beneficial to have use of a digital signal output due to the evaluation and calculations of the data being done by a digital controller. I2C and SPI are used in a wide range of accelerometers and is a likely choice for the data output. The ADXL 372 Accelerometer was selected due to its high force range and appropriate bandwidth for the impulses being measured. [JD]

Table 2.2.1: Accelerometers [JD]				
Accelerometer	Price	Bandwidth	Force range	Output
ADXL372	\$6.33	3.2KHz	200g	SPI, I2C

2.2.2. Gyroscope

A gyroscope measures angular velocity by measuring the vibrations through a sheet of piezoelectric material. The angular velocity is an important measurement for brain trauma caused by torsion. An impact of 96g has the equivalent angular acceleration of 7,235 rad/sec². With an impulse that lasts 5.5 milliseconds, an impulse of 96g could reach an angular velocity of 6875.49 deg/sec². This means that the gyroscopes need to be able to measure a minimum of 7000 deg/sec² of angular velocity in order to properly measure angular acceleration. The gyroscope

should only send data to the microcontroller when the angular rate exceeds a given threshold; this threshold will be determined based on the assumption that the head will experience no angular velocity while at rest. The ITG-3701 was selected due to its fast sampling rate and large angular velocity range. The angular acceleration will be calculated using the rate of change of the angular velocity recorded by the gyroscope. [JD]

Table 2.2.2: Comparison of Gyroscopes [JD]				
Gyroscope	Price	Bandwidth	Typical Angular Velocity	Output
ITG-3701	\$9.43	16kHz	4000 deg/sec	I2C

2.3. Communications

2.3.1. Intra-headband Communication

Within the headband, multiple communication protocols will be required for the microcontroller to utilize the peripherals on the headband. The accelerometer and gyroscope will record the motion of the head and relay it to the controller over an I2C bus or through SPI. I2C requires an ICL and an SDA connection. SPI requires a four-wire connection for SCLK, MOSI, MISO, and CS. [NL, JM]

Reading and writing data to the microSD card from the microcontroller will require Serial Peripheral Interface (SPI) communication. To connect with the controller (besides providing power), the MISO (master in serial out) goes to the SDI (serial data in), MOSI (master out serial in) is connected to the SDO (serial data out), SCK (serial clock) pin goes to the SCK on the microcontroller and the CS (chip select) goes to the corresponding CS on the

microcontroller. These connections are paired with individual SD card commands, that are all 6 bytes long, and are paired with the byte-formatted data that is to be read and written. [NL]

Finally, Universal Asynchronous Receiver Transmitter (UART) communication will be used for asynchronous communication between the Bluetooth module and the headband microcontroller. This communication will require four wired connections: RX, TX, RTS, and CTS. If required, the connection requirements can be limited to just the RX and TX lines, but the RTS and CTS lines will make communication faster and simpler by enabling hardware handshaking. The UART connection between the microcontroller and Bluetooth module will be used in the transmission of impact and step data from the headband to the smartphone. [JM]

2.3.2. Headband to Smartphone Communication

The headband and smartphone will communicate wirelessly via Bluetooth Low Energy (BLE). BLE is defined by the IEEE 802.15.1 standard; compared to traditional Bluetooth, BLE has a significantly lower average current draw at the cost of effective range and transmission speed. The headband's Bluetooth module will search for a connection using BLE advertising. Once the smartphone application has connected to the headband Bluetooth module, the smartphone can begin issuing commands over the Bluetooth connection. All data transmitted over the Bluetooth connection will be forwarded to and from the headband microcontroller via the UART connection to the Bluetooth module. [JM]

2.4. Embedded Systems

2.4.1. Bluetooth Module

This project will employ a Bluetooth module for wireless communication between the microcontroller and the smartphone. As mentioned previously, the Bluetooth module's primary function is to establish a Bluetooth connection with the smartphone. Then, the Bluetooth module will forward the received UART data to the Bluetooth output; similarly, incoming data or commands from the smartphone to the microcontroller will be reformatted into UART packets and sent to the microcontroller. This "transparent UART," as Microchip calls it, will make it simple to transfer information between the phone and microcontroller. This project will utilize the Microchip BM78 Dual-Mode Module. The configuration of the BM78 Bluetooth Module will be handled by a Windows-based UI application provided by Microchip. The Bluetooth module will be configured for Bluetooth Low Energy. According to the BM78 data sheet, this will significantly lower the average current draw of active data transmission from about 15 mA to 7 mA for the worst case. Data throughput of the Bluetooth module is expected to be up to 6 Kbytes/sec at the standard 115200 UART baud rate. Below is a comparison of the BM78 to two other Bluetooth modules. The BM78 was selected for its higher data throughput; if transmitting all session data at the end of the session, a higher transfer rate will be required. [JM]

Table 2.4.1: Comparison Between Bluetooth Modules [JM]				
Bluetooth Module	Price	Peak Average Current Draw	Maximum Data Throughput	Operating Voltage
Microchip RN4870/71	\$5.82	2.23 mA while connected	Up to 10 Kbits/sec	1.9-3.6V
Microchip BM70/71	\$5.82	2.23 mA while connected	~8.6 Kbits/sec	1.9-3.6V
Microchip BM78	\$6.59	~7 mA while transmitting	Up to 56 Kbits/sec	3.3-4.4V

2.4.2. Microprocessor

The microprocessor that will be used on the microcontroller will be the PIC24FJ1024GB610, which is similar to the one used in embedded systems interfacing class. The PIC24 is capable of four different I2C inputs, which will be used for the peripherals, as well as UART compatibility for the Bluetooth module and SPI compatibility for use of a microSD card reader. There is also an embedded software debugger built in which will help with the development process, as well as a Real-Time built in clock for timestamping the incoming peripheral data from all the sensors. [NL]

2.5. Numerical Computations

Several studies on concussive impacts suggest that resultant linear acceleration, resultant angular acceleration, the Head Injury Criterion, and the Gadd Severity Index are appropriate measures of impact severity. An increase in each of these four measures is associated with an increased risk of concussion. Each impact detected by the headband will calculate all four

measures to assess concussion risk. Once calculated, a weighted sum, denoted PCS, will be determined based on the Principal Component Analysis performed by Greenwald et. al. [JM]

2.5.1. Linear Acceleration

The headband will utilize three triaxial accelerometers to determine the linear acceleration vectors received to the head center of mass. The triaxial accelerometers will be placed at the front, left, and right of the head; according to Greenwald et. al., impacts to the front and side of the head are more likely to cause injury than impacts to the back or top of the head. Furthermore, a review by O'Connor et. al. suggests that utilizing multiple accelerometers will limit error. Each triaxial accelerometer will transmit the magnitude of acceleration received for each of its axes when one of the axes' magnitudes exceeds the minimum recording threshold. The average acceleration for each axis will be calculated by averaging the corresponding axes for each accelerometer. Then, the magnitude of the acceleration $|a|$ will be calculated using equation 2:

$$|a| = \sqrt{(a_x^2 + a_y^2 + a_z^2)} \quad (3)$$

Here, x, y, and z are the magnitudes recorded at the three single-axis accelerometers in the triaxial accelerometer package. Ideally, this calculation will be performed on the microcontroller to limit the amount of data transmitted across the limited Bluetooth connection. However, since the computation may be expensive (given that there are three squares and one square root), the computation may be moved to the mobile application. In that case, the data for each axis would be sent to the microSD for storage, and later, the Bluetooth module for transmission.

On the mobile device, the data must be examined to determine which accelerations correspond to the same impact. Each continuous recording of impact data at the maximum recording frequency will be examined to find the peak resultant LA. That is, for a given impact,

data will be transmitted while the magnitude exceeds the recording threshold; during this time frame, the maximum acceleration will be used to describe the magnitude of linear acceleration for that impact. The matching of acceleration magnitudes to impacts cannot be done on the microcontroller because the discrete data set is needed for numerical integration when calculating the Head Injury Criterion and Gadd Severity Index. [JM]

2.5.2. Angular Acceleration

To determine angular acceleration, the headband hardware will need either an angular rate sensor or a gyroscope. For this project, a triaxial gyroscope will be placed on the left, front, and right of the head. The triaxial gyroscope will measure angular velocity in radians per second; thus, numerical computation is needed to determine angular acceleration in radians per second squared. The microcontroller will receive the data from each triaxial gyroscope as three values: the x, y, and z values for angular velocity in degrees per second, also known as pitch, yaw, and roll. For each axis, the average angular velocity recorded by each gyroscope will be calculated. The averages will then be converted to radians per second, then numerically differentiated. The order of numerical differentiation will be determined by the capabilities of the headband processor. Once each axis' data is numerically differentiated, the resultant magnitude of angular acceleration $|\alpha|$ will be calculated according to equation 3:

$$|\alpha| = \sqrt{(\alpha_x^2 + \alpha_y^2 + \alpha_z^2)} \quad (4)$$

This data will be stored on the microSD and later sent to the mobile application. On the mobile application, similar to linear acceleration, the data will be searched to determine the maximum magnitude of angular acceleration observed for each recorded impact. [JM]

2.5.3. Head Injury Criterion

The Head Injury Criterion, or HIC, is an assessment of impact severity often used by the automobile industry. The value of the HIC will be calculated using equation 1, reprinted below:

$$HIC^* = (t_2 - t_1) \left[1/(t_2 - t_1) * \int_{t_1}^{t_2} a(t) dt \right]^{5/2} \quad (1)$$

Here, $a(t)$ is the linear acceleration of the head and $t_2 - t_1$ is the time step of the accelerometers, which will be denoted Δt . This integration will be estimated using numerical integration with $\Delta t = 15$ ms, a standard time step for the HIC. With a sample rate of 1000Hz, numerical integration can be performed using all fifteen steps within the 15 ms window to limit error. [JM]

2.5.4. Gadd Severity Index

The Gadd Severity Index, or GSI, is another measure of impact severity that is common in sports industries. This measure is similar to the HIC; it will be calculated as shown in equation 4:

$$GSI = \int_0^T a(t)^{5/2} dt \quad (2)$$

In this formula, $a(t)$ is the resultant linear acceleration and T is the impact duration. To determine the GSI value, numerical integration will be performed using an estimation of impact duration determined by the time frame for which the impact magnitude exceeded the minimum recording threshold. As done for the HIC, the numerical integration will be performed across all possible time steps to increase accuracy. [JM]

2.5.5. Principal Component Score

This project originally intended to use Principal Component Analysis to determine an objective, uncorrelated sum of the previous four variables. However, without access to the original impact data set, this was not possible. Hence, this project will employ the equation

obtained by the Principal Component Analysis performed by Greenwald et. al. on a set of nearly 300,000 impacts. This equation solves for the Principal Component Score, or PCS. The equation for this effective weighted sum is given below:

$$PCS = 10 \cdot ((0.4718 \cdot sGSI + 0.4742 \cdot sHIC + 0.4336 \cdot sLIN + 0.2164 \cdot sROT) + 2) \quad (5)$$

Here, sX is equal to $(X - \text{mean}(X))/\text{SD}(X)$ where $\text{mean}(X)$ is the average value for X in the data set and $\text{SD}(X)$ is the standard deviation of X (Greenwald et. al). This means that the mean and standard deviation of the data set from the study by Greenwald et. al. are required. The numerical computations will be performed using each of these values as given constants obtained from the study. Once PCS is obtained, the approximate concussion risk will be displayed to the user based on the correct prediction level and false response rate associated with the given PCS value. The correct prediction level and false response rate will be based on the data in the study by Greenwald et. al. [JM]

2.5.6. Step Detection

There will also have to be calculations based on the linear acceleration for upright movement detection for step counting. To achieve this, the Signal Magnitude Area, or SMA, is calculated using the following equation:

$$SMA = \frac{1}{t} (\int_0^t |x(t)| dt + \int_0^t |y(t)| dt + \int_0^t |z(t)| dt) \quad (6)$$

This equation categorizes the data into two sections: walking and jogging. Walking is in the range between 0.135g's and 0.8g's, while jogging is above 0.8g's. [NL]

The calculations can be improved further by using adaptive timing threshold to eliminate false positives. The following equation measures the timing threshold:

$$t_l = f_s * \frac{0.1}{\text{mean}(SMA)} \quad (7)$$

The adaptive timing threshold calculation uses the sampling frequency of the accelerometer that is used in the calculation, as well as the mean of the SMA calculated in equation (6) above, to determine the timestamp between each heel-strike to the ground. If the timestamp is not big enough, or is too small for a regular step, then the system determines the accuracy of the step that was detected. The adaptive timing threshold can be used to decrease the error range of the calculations. [NL]

3. Engineering Requirements

Table 3.1: Engineering Requirements [JD, JM, NL]		
Marketing Requirements	Engineering Requirements	Justification
3, 4	The headband system must be able to operate without a recharge for 4 hours.	A user's workout session is expected to take no more than 4 hours at a time.
1, 7	The headband system must be able to detect impacts with linear accelerations up to 150g.	Studies on concussive impacts suggest that most impacts will be below this threshold.
1, 7	The headband system must be able to detect impacts with angular accelerations up to 12000 rad/s ² .	Studies suggest that significantly fewer than five percent of concussive impacts will cause angular acceleration over 12000 rad/s ² .
1, 7	The headband system's impact detection should experience a maximum of 20% error.	This will reduce the error in the approximation of concussion risk.
1, 6, 7	The headband system must store resultant linear and angular acceleration with a timestamp.	The data must be stored locally before it is sent to the mobile application for processing.
2, 6	The headband system must determine when a step is taken and increment a step counter in local storage.	The number of steps taken must be stored locally before it is sent to the mobile application.
4, 5	The headband hardware must be removable from the headband itself.	The user will need to recharge the hardware and wash the headband.
1, 3	The headband hardware must be able to withstand impacts with linear acceleration of 200g.	The device needs to withstand and operate during strong impacts in order to provide proper data.

6	The headband system must communicate wirelessly with a mobile device.	Wireless communication will be convenient and easy to use.
4, 6	The headband system communication protocol should not require connection to the mobile device until the session is complete.	This will retain battery power on the headband and will allow the headband user to be at a distance from their smartphone.
7	The mobile application will calculate a weighted sum of linear acceleration, angular acceleration, and the Head Injury Criterion to assess concussion risk.	A weighted sum will be a more effective predictor than any of the measures individually.
2, 6, 7, 8	The mobile application must display session data using statistical analysis and graphs.	The user likely does not need the raw impact and step data, but rather a summary of the session's impacts and steps.

Marketing Requirements

1. The headband device must detect high acceleration impacts.
2. The headband device must track the number of steps taken across a user session.
3. The headband device must be durable enough for use during workouts or contact sports.
4. The headband device must be able to operate without recharge across an entire use session.
5. The headband hardware must be comfortable and removable from the headband itself.
6. The headband device must communicate with a mobile application that can access and analyze the recorded data.
7. The mobile application must be able to assess concussion risk based on the impacts recorded in the user session.
8. The mobile application must display step rate, concussion risk, and the potentially concussive impacts from the user session.

4. Engineering Standards

Table 4.1: Engineering Standards [JD, JM, NL]		
Section	Standard	Use
Communication	UART	Used for data transmission from the microcontroller to the Bluetooth module; serves as the link layer protocol for all Bluetooth communication.
	SPI	Connects the microSD, accelerometers, and gyroscopes to the microcontroller.
	Bluetooth (IEEE 802.15.1)	Forms the wireless connection between the headband Bluetooth module and the smartphone.
Programming Languages	C	Language to process the data on the microcontroller.
	Java	Used to construct the mobile app to display the data.
	IS1678S Command Set	Command set for the Bluetooth chip.
Connector Standards	SPI Bus	Connects the accelerometers and gyroscope to the microcontroller.

5. Accepted Technical Design

5.1. Hardware Design

5.1.1. Level 0 Block Diagram

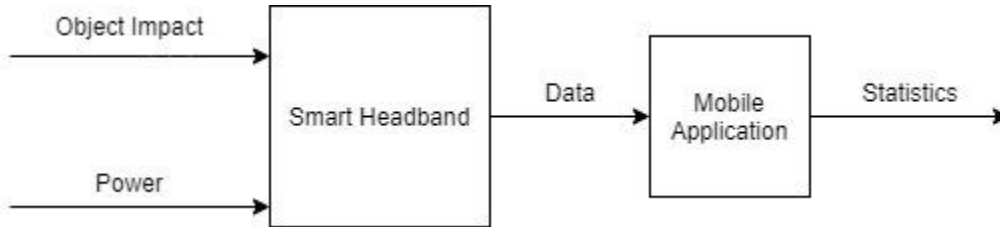


Figure 5.1.1: Level 0 Block Diagram of the system’s hardware. [NL]

Table 5.1.1.1: Level 0 Functional Requirements for the Smart Headband. [JD, NL, JM]	
Module	Smart Headband
Designer	Jack Durkin, Noah Lewis, and John Michel
Inputs	Object Impact: obtained from user movement. Power: Lithium-ion button cell.
Outputs	Data: concussion risk data.
Description	Detect user’s concussion risk based on accelerometer readings from the headband device.

Table 5.1.1.2: Level 0 Functional Requirements for the Mobile Application. [JD, NL, JM]	
Module	Mobile Application
Designer	John Michel, Noah Lewis
Inputs	Raw impact data.
Outputs	Concussion detection: display data when threshold was passed.
Description	Display data for the user to view and be able to share the data with their healthcare provider.

5.1.2. Level 1 Block Diagram

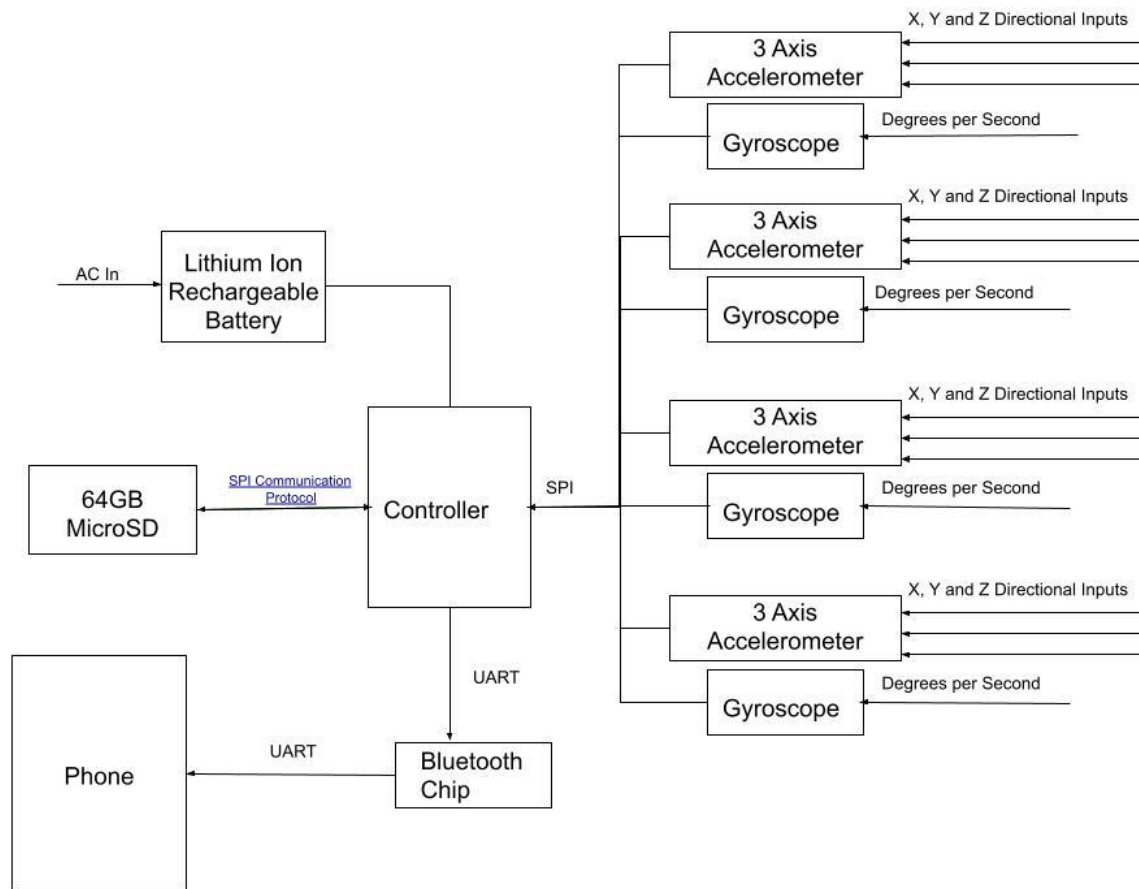


Figure 5.1.2: Level 1 Block Diagram of the system's hardware. [NL]

Table 5.1.2.1: Level 1 Functional Requirements for the Phone. [JM]	
Module	Phone
Designer	John Michel, Noah Lewis
Inputs	Data for impacts and steps, transmitted via Bluetooth and Microchip's transparent UART.
Outputs	None.
Description	The smartphone receives impact data via Bluetooth and processes the results.

Table 5.1.2.2: Level 1 Functional Requirements for the Bluetooth Chip. [JM]	
Module	Mobile Application
Designer	John Michel
Inputs	UART communication and data from the microcontroller.
Outputs	Bluetooth transmission of impact and step data.
Description	Uses Microchip's Transparent UART to retransmit UART data over the Bluetooth connection, and vice versa.

Table 5.1.2.3: Level 1 Functional Requirements for the Microcontroller. [NL, JM]	
Module	Microcontroller
Designer	Noah Lewis, John Michel
Inputs	Accelerometer and Gyroscope data.
Outputs	Concussion detection: Calculated probability of concussions and the time they happened.
Description	Using the data from the accelerometer and gyroscope to calculate the probability of a concussion and the step counts to send to the mobile application on the phone.

Table 5.1.2.4: Level 1 Functional Requirements for the MicroSD. [NL]	
Module	MicroSD
Designer	Noah Lewis
Inputs	Raw and processed data from microcontroller.
Outputs	Processed data from microcontroller
Description	The microSD card is used to store all raw data that is not in a buffer and/or waiting to be processed by the microcontroller. When the phone is in range, the data is taken from the card and sent to the mobile app via Bluetooth.

Table 5.1.2.5: Level 1 Functional Requirements for the Accelerometer. [JD]	
Module	Accelerometer
Designer	Jack Durkin
Inputs	Linear acceleration applied to the device in the X, Y, and Z directions.
Outputs	SPI digital signal of the linear acceleration applied to the device.
Description	Takes the continuous linear acceleration that is applied to the device, and by measuring electrical changes to a piezoelectric material a force is measured on each axis. The measured force is converted into a digital signal and sent to the controller using SPI.

Table 5.1.2.6: Level 1 Functional Requirements for the Gyroscope. [JD]	
Module	Gyroscope
Designer	Jack Durkin
Inputs	Angular velocity applied to the device in yaw, roll, and pitch directions.
Outputs	SPI digital signal of the angular velocity applied to the device
Description	Takes continuous angular velocity and that is applied to the device, and by measuring electrical changes to a piezoelectric material a velocity is measured for each axis of rotation. The measured velocity is converted into a digital signal and sent to the controller using SPI.

Table 5.1.2.7: Level 1 Functional Requirements for the Lithium Ion Battery. [JD]	
Module	Lithium Ion Battery
Designer	Jack Durkin
Inputs	DC voltage to USB connection
Outputs	DC voltage of 3.7V
Description	The lithium ion battery takes in charge from a USB connection and stores this charge. It then uses this stored energy to power the device's hardware components.

5.1.3. Level 2 Block Diagrams

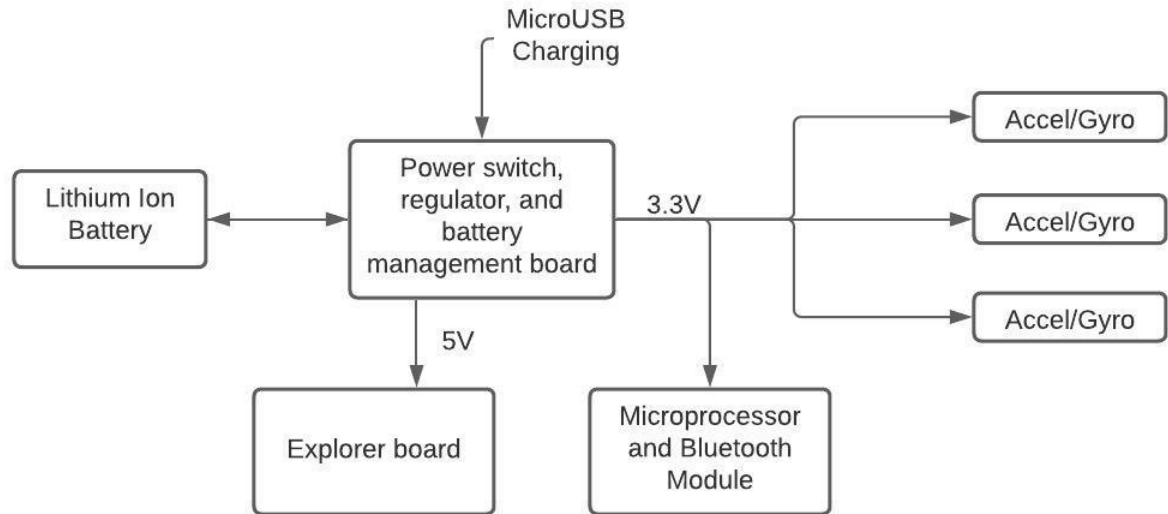


Figure 5.1.3.1: Level 2 Block Diagram of the power distribution system. [NL,JD]

Table 5.1.3.1: Level 2 Functional Requirements for the Battery Management System. [JD]	
Module	Lithium Ion Battery
Designer	Jack Durkin
Inputs	A DC current rated at no more the 500mA per battery
Outputs	Voltages ranging between 4.2V-1.7V
Description	The system contains a USB input for a DC current to be applied. The current should not exceed 500mA to protect the battery from damage. The system needs to supply power for an IMU, a microcontroller, and 3 pairs of accelerometers and gyroscopes. The battery also powers an explorer board. Each of these has different power requirements that will be explained in their own power distribution functional requirements. Two of the batteries are placed in series to increase the output voltage to a range of 8.4-3.4V. This voltage supplies a 3.3V regulator and a 5V regulator. The 3.3V regulator supplies the physical components of the headband, and the 5Vs powers the explorer board.

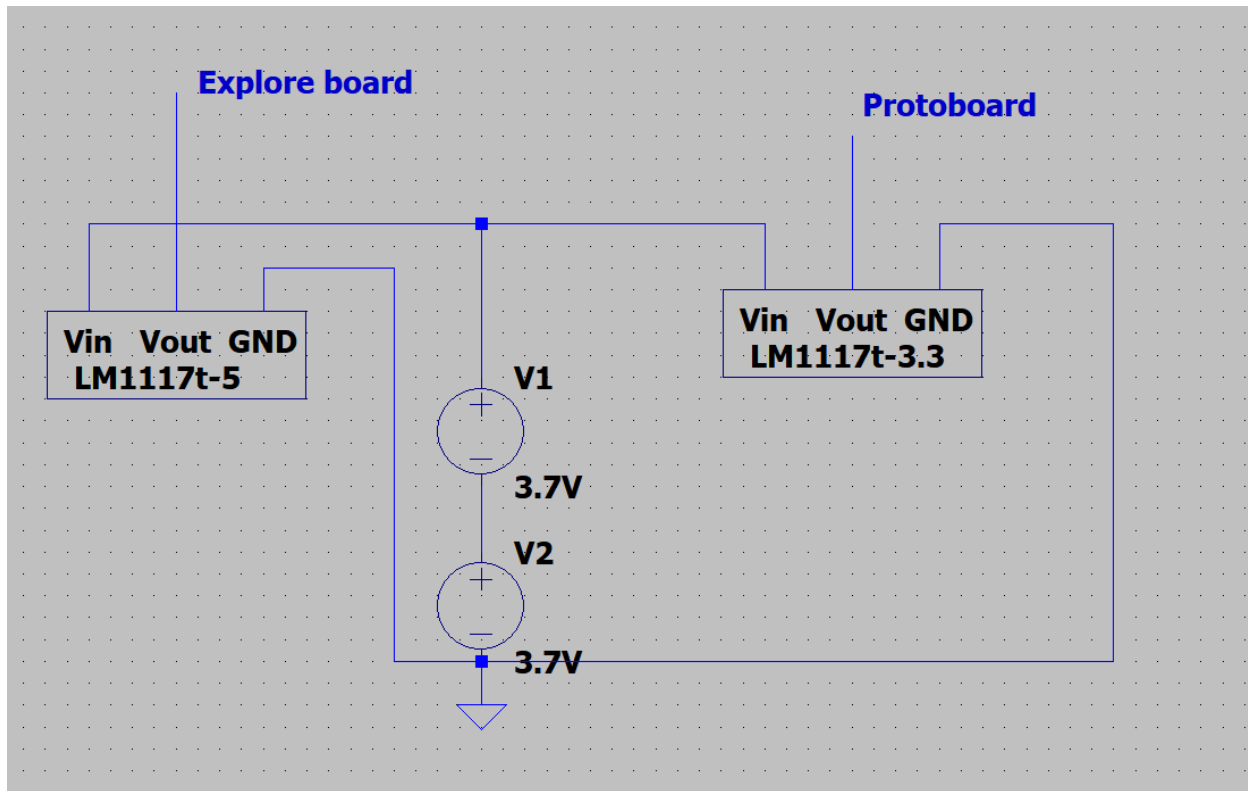


Figure 5.1.3.1.1: Voltage Regulator Schematic [JD]

When placed on the PCB the overall current draw from the three accelerometers, gyroscopes, PIC24, microSD, and Bluetooth device, had a current draw of 800mA. The voltage regulator originally selected had a max current of 40mA and could not provide enough power for the entire board, even with three separate regulators. To solve this issue a LM1117t-3.3 3.3V regulator was selected due to the fact it could supply over 1A of current. The LM117t-3.3 required an input voltage of 7 volts to operate. By taking a second battery and placing it in series the voltage supplied by the batteries had enough voltage, stored power to operate for the specified amount of time. A LM1117t-5 was selected to power the Explorer board that was attached to the box. [JD]

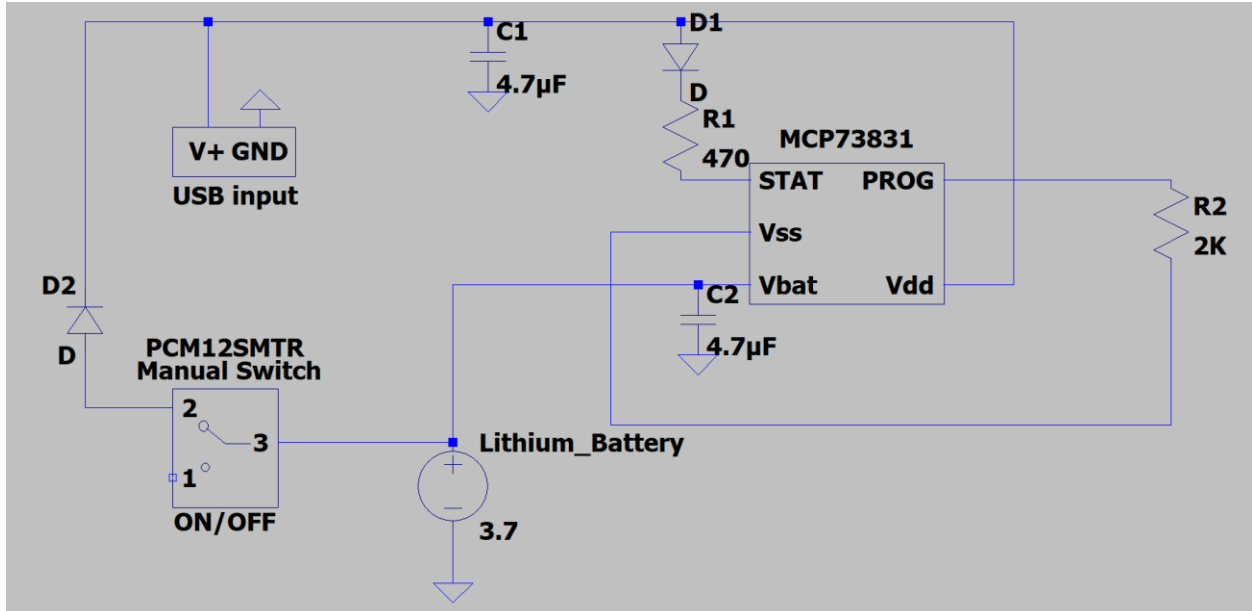


Figure 5.1.3.1.2: Battery Charging Circuit [JD]

The battery charging circuit was designed to charge a 3.3V battery. Because the design called for the two batteries to be put into series, this circuit would not be able to charge both simultaneously. To solve this issue, both the batteries were made to be removable. Each could be plugged into this charging circuit and charged separately. [JD]

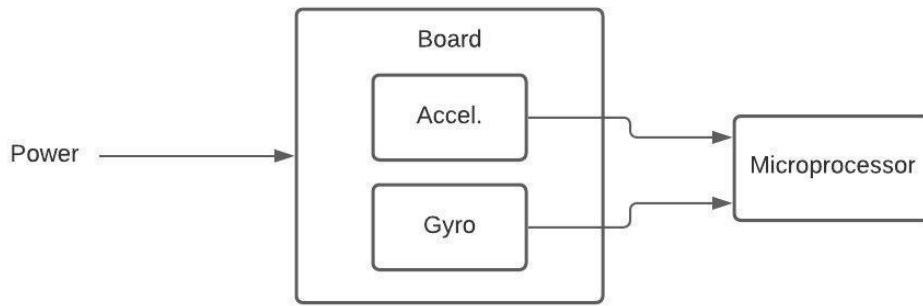


Figure 5.1.3.2: Level 2 Block Diagram of the concussion sensors. [NL]

Table 5.1.3.2: Level 2 Functional Requirements for the Accelerometer/Gyroscope Board. [JD]

Module	Lithium Ion Battery
Designer	Jack Durkin
Inputs	A DC input with a voltage inside the range of 1.71V-3.5V.
Outputs	Separate signals of the accelerometer and Gyroscope data in SPI format
Description	The accelerometer has an operational voltage range of 1.6V-3.5V, with peak performance at 2.5V and a current draw ranging in the pico-amps. The gyroscope has an operational voltage range of 1.71V-3.6V and a current supply near 3.3mA. The voltage is regulated at the battery control board. The outputs are in SPI format and go directly to the microprocessor.

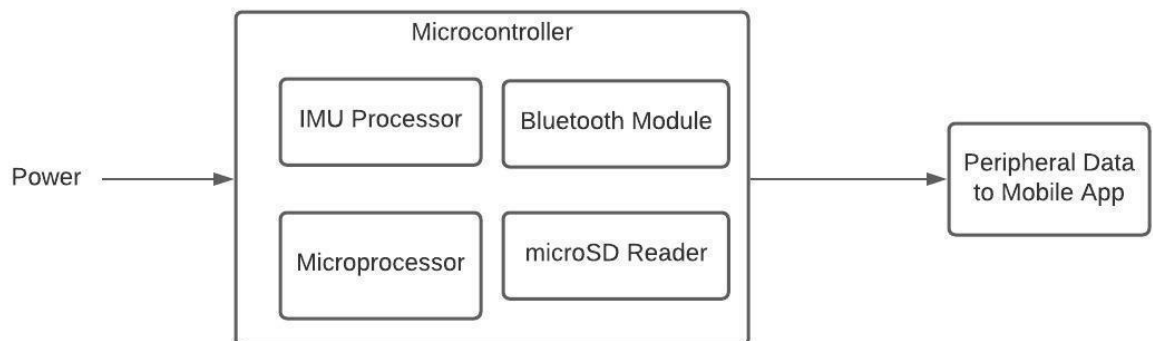


Figure 5.1.3.3: Level 2 Block Diagram of the microcontroller. [NL]

Table 5.1.3.3: Level 2 Functional Requirements for the PIC24 Microprocessor. [JD]	
Module	PIC24FJ1024GB610
Designer	Jack Durkin, Noah Lewis
Inputs	3.3V input voltage
Outputs	Peripheral Data
Description	The microprocessor is powered by a 3.3V input voltage. The microprocessor collects I2C signals from the IMU, accelerometers, and gyroscopes, as well as state of charge, and sends this data to the mobile device through the Bluetooth module.

Table 5.1.3.4: Level 2 Functional Requirements for the Bluetooth Module. [JD]	
Module	BM78SPPS5MC2-0002AA
Designer	Noah Lewis, John Michel
Inputs	Supply Voltage
Outputs	Peripheral Data
Description	The Bluetooth module receives peripheral data from the microprocessor in the form of UART. it transmits this data to the mobile device.

Table 5.1.3.5: Level 2 Functional Requirements for the microSD Reader. [JD]	
Module	5031821852
Designer	Jack Durkin, Noah Lewis
Inputs	Supply Voltage, Peripheral Data
Outputs	Peripheral Data
Description	The microSD reader stores peripheral data from the IMU, accelerometer, and gyroscope, as well as state of charge. It communicates with a microprocessor via SPI. The data will be stored in buffers.

Table 5.1.3.6: Level 2 Functional Requirements for the IMU Processor Carrier Board. [NL]	
Module	Lithium Ion Battery
Designer	Jack Durkin, Noah Lewis
Inputs	1.95 V to 3.6 V supply voltage
Outputs	Accelerometer and Gyroscope data from the IMU Processor
Description	The carrier board is to be designed to power the IMU Processor along with the capability of outputting the 3-Axis Gyroscope data of ± 250 dps, ± 500 dps, ± 1000 dps, and ± 2000 dps and/or the 3-Axis Accelerometer data of $\pm 2g$, $\pm 4g$, $\pm 8g$, and $\pm 16g$ to the microprocessor.

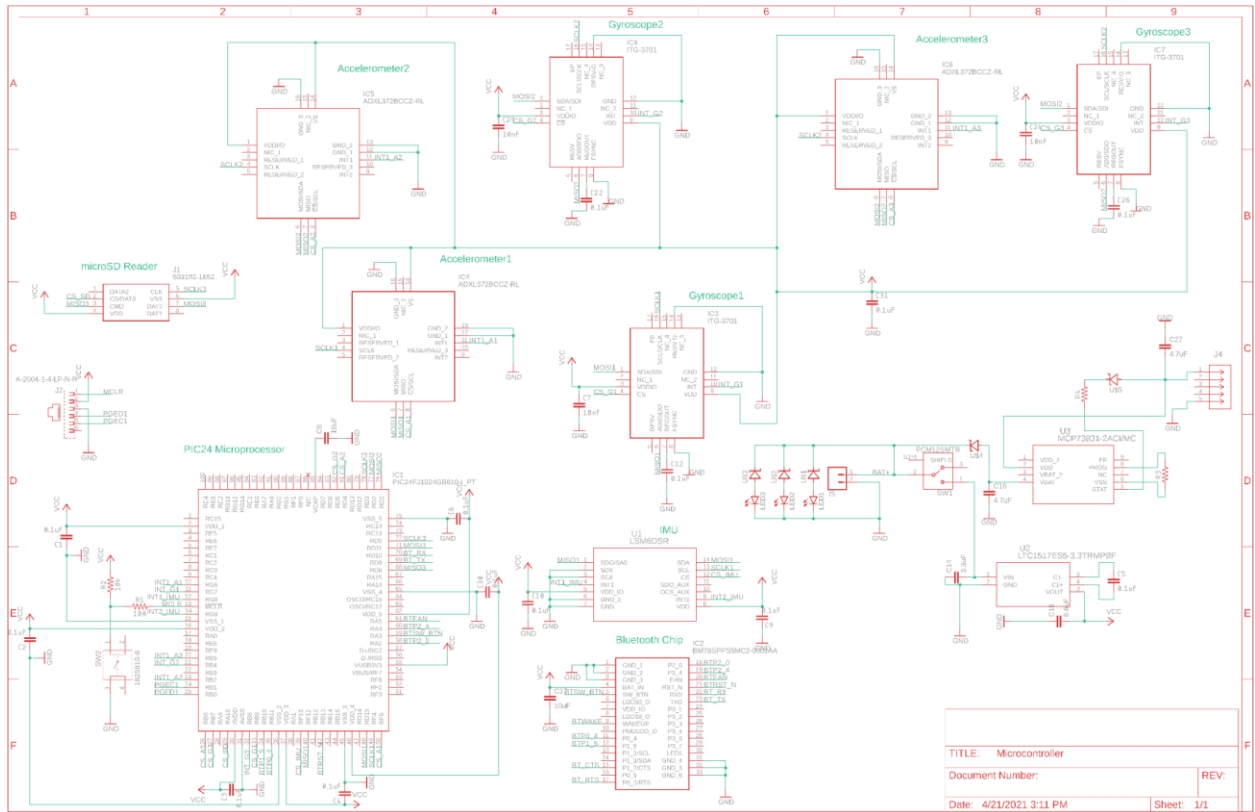


Figure 5.1.3.4: Schematic for the microcontroller. [NL]

5.1.4. Printed Circuit Boards

Using the schematic from above, we created a printed circuit board (PCB) to house all of our components, as well as wire them all together properly. There were 3 fully built iterations of the PCB designed during the spring session of senior design. The first design incorporated all of the peripherals and the microprocessor, but lacked the resistors and capacitors required. The second iteration added all the resistors and capacitors, along with a new charging circuit. However, the third design was created because it was discovered that some chip selects were not connected, and the microSD required its own individual SPI Bus. Below is a picture of the second and third PCBs: [NL]



Figure 5.1.4.1 Second (top) and third PCB (bottom) [NL]

The third design of the PCB was close to working; however, the board had a couple problems along the way. The first was discovered that the voltage regulator could not actually handle all the components on the board, so a smaller protoboard was designed with another

regulator to get the voltage to the correct amount. After being able to power the board, the PIC24 microcontroller was having issues with being programmed. Further investigation showed the VSS pins were connected to power instead of ground, which caused the microcontroller to malfunction. A corrected Eagle board design is down below, which fixed the main issues with the board:

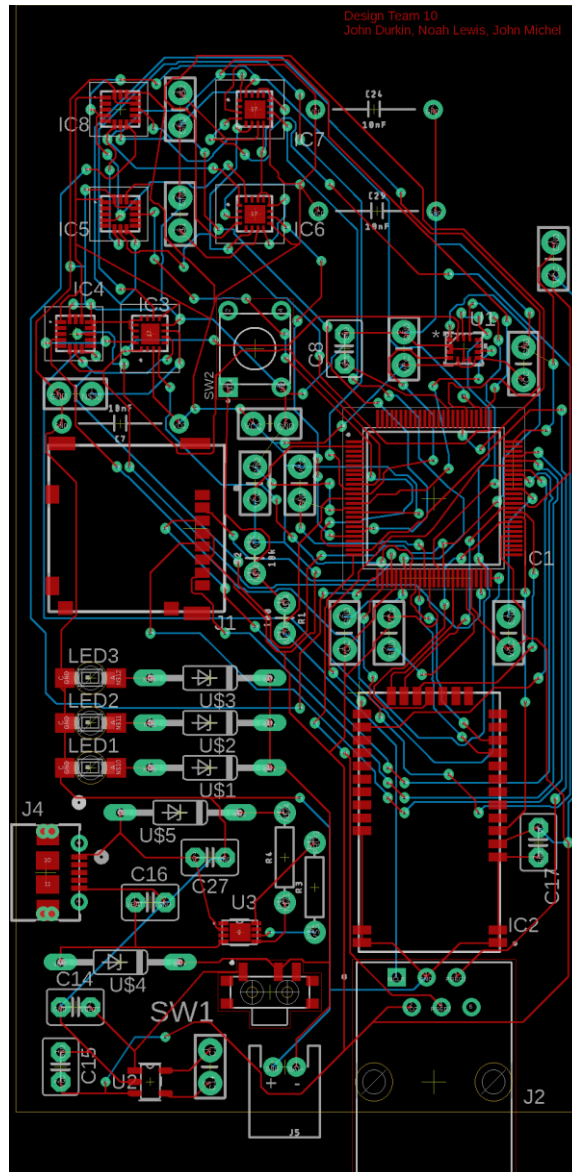


Figure 5.1.4.2: Board design for the microcontroller. [NL]

Table 5.1.4.1: Microcontroller Pin Configuration			
Group	Pin #	Pin Purpose	Notes
SPI Bus 1	40	MISO	
SPI Bus 1	48	MOSI	
SPI Bus 1	49	SCLK	
SPI Bus 1	33	Gyro1 Chip Select	
SPI Bus 1	39	IMU Chip Select	
SPI Bus 1	50	Accel1 Chip Select	
Bus 1 Interrupts	10	Accel1 Interrupt	
Bus 1 Interrupts	11	Gyro1 Interrupt	
Bus 1 Interrupts	12	IMU Interrupt 1	
Bus 1 Interrupts	14	IMU Interrupt 2	
SPI Bus 2	76	MISO	
SPI Bus 2	77	MOSI	
SPI Bus 2	78	SCLK	
SPI Bus 2	81	Accel2 Chip Select	
SPI Bus 2	82	Gyro2 Chip Select	
SPI Bus 2	26	Accel3 Chip Select	
SPI Bus 2	27	Gyro3 Chip Select	
Bus 2 Interrupts	21	Accel2 Interrupt	
Bus 2 Interrupts	23	Gyro2 Interrupt	
Bus 2 Interrupts	20	Accel3 Interrupt	
Bus 2 Interrupts	32	Gyro3 Interrupt	
SPI Bus 3	68	MISO	microSD
SPI Bus 3	71	MOSI	microSD
SPI Bus 3	72	SCLK	microSD
SPI Bus 3	29	CS for microSD	microSD
Bluetooth	34	P1_5	
Bluetooth	35	P0_4	
Bluetooth	42	RST_N	
Bluetooth	58	P2_0	

Bluetooth	59	SW_BTN	
Bluetooth	60	P2_4	
Bluetooth	61	EAN	
Bluetooth	69	Bluetooth RX	
Bluetooth	70	Bluetooth TX	
MISC	13	MCLR	
MISC	24	PGEC	
MISC	25	PGED	

5.1.5. Proto Boards

To implement the design two separate proto boards were created. The hardware board would house the Bluetooth module, accelerometer, gyroscope, and SD card reader. The battery protoboard had the two separate voltage regulators that are supplied by two batteries in series. The first regulator supplies the first mentioned protoboard with 3.3V. The voltage and ground from this regulator are attached to each component's corresponding power and ground pins. The second voltage regulator supplies the explore board with 5V. The output of this regulator is soldered to a micro-USB connection. The two boards are attached to a black box, and secured using screws, bolts, and risers. The batteries are removable and connected to the box with Velcro. The black box has a 1-inch by 3-inch square cut out of the top so that the lid could be secured without damaging the Bluetooth module. The explore board is secured to the lid of the board using screws and bolts. The lid is secured using screws. The communication pins from the hardware components on the hardware proto board components are soldered to connected pins in their corresponding explore board positions. This allows for easy removal from the explore board. [JD]

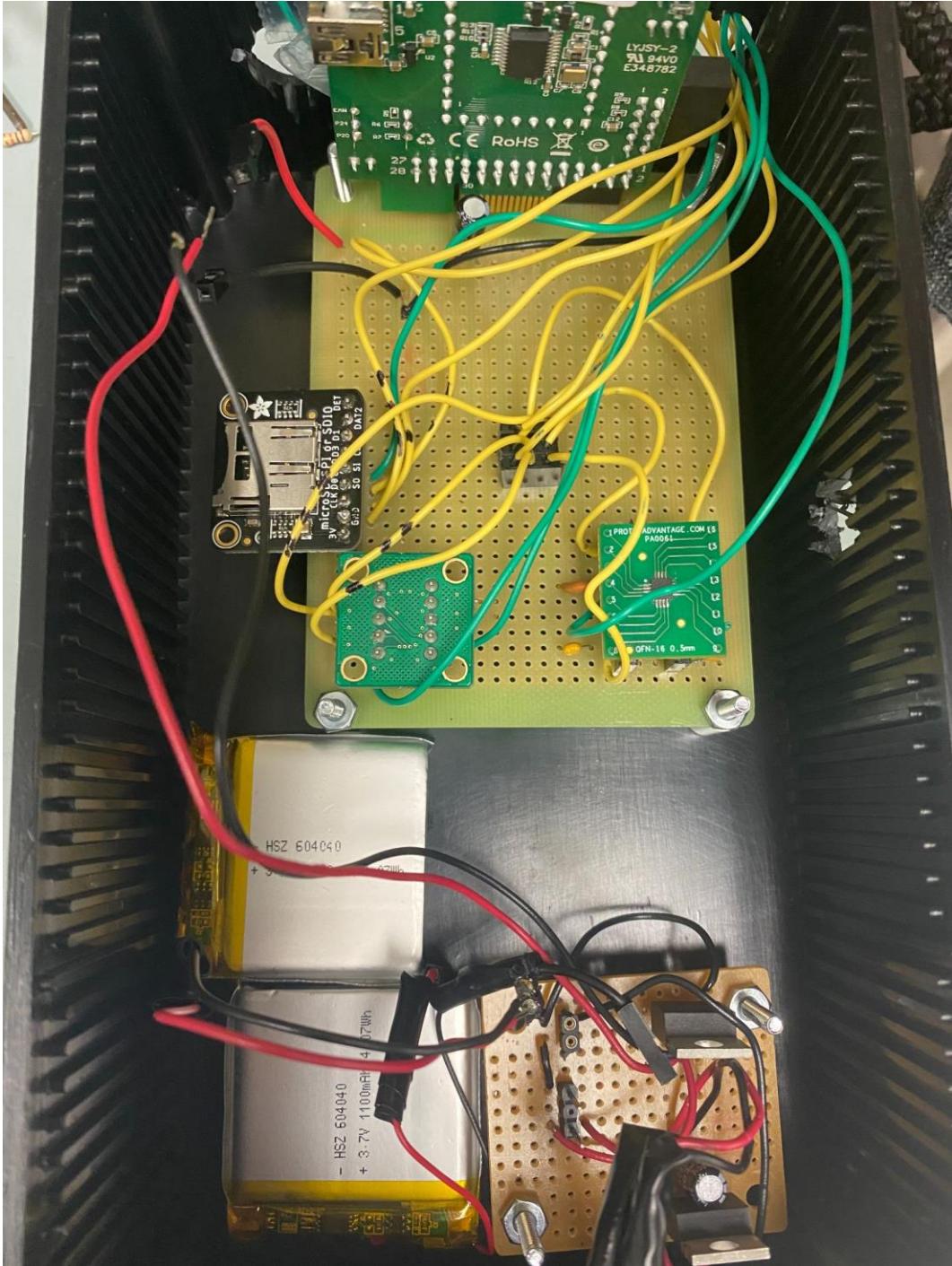


Figure 5.1.5: Protoboard implementation [JD]

5.1.6. Testing Apparatus

The device needs to be able to measure both linear and angular acceleration with an error rate of 20% or lower. For accuracy purposes it is best that these measurements are split up into two separate tests. To test the linear acceleration a pulley system was created using a 2x4 wood beam cut into sections of 3-foot, 2-foot, and two 1-foot pieces. The pieces are attached using screws into the structure seen in the image below.



Figure 5.1.6: Testing Apparatus [JD]

A $\frac{1}{2}$ inch diameter hole is drilled at the top of the board, and a rod is secured through this hole. A pulley is secured at the end of the rod. The headband is tied to a string that is fed through

the pulley and tied to a weight. Using the equations of gravitation acceleration an equation can be derived to measure the force applied from the pulling of the string. The mass of the headband is measured and recorded. To apply a consistency acceleration to the headband a weight is tied to the other side of the pulley. This weight is dropped, forcing the headband to accelerate upwards. With the mass of the weight and headband, the equation below can be used to calculate the linear acceleration of the headband. [JD]

$$A=(M_2-M_1)/g$$

To test the angular acceleration the same rig created to measure linear acceleration can be used. By tying the string around the rod with the headband attached at the end a pendulum is created. By measuring the length of the pendulum and the mass of the headband, the period of the oscillations and the angular acceleration can be calculated using the base equations of a pendulum.[JD]

$$T=2\pi/\omega$$
$$\omega^2=M/R$$

5.2. Software Design

5.2.1. Level 0 Block Diagram

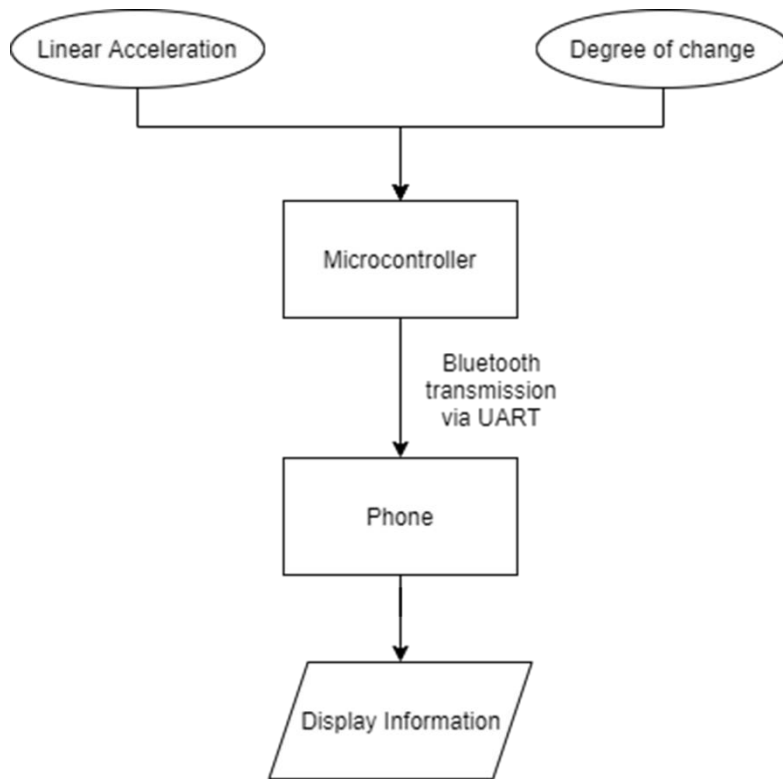


Figure 5.2.1: Level 0 Block Diagram of the system's software. [NL]

Table 5.2.1.1: Level 0 Functional Requirements for the Microcontroller. [JM]	
Module	Microcontroller
Designer	Noah Lewis, John Michel
Inputs	Linear acceleration and degree of change (angular velocity).
Outputs	Impact data from a workout session.
Description	Data is transmitted wirelessly over Bluetooth. The data is the original angular rate and linear acceleration data from the gyroscope and accelerometer, respectively.

Table 5.2.1.2: Level 0 Functional Requirements for the Phone. [JM]	
Module	Phone
Designer	John Michel, Noah Lewis
Inputs	Impact data from the headband.
Outputs	Impact data from the headband, as well as statistics based on that data (e.g. mean, peak impact, concussion risk).
Description	The smartphone processes and displays the impact data received from the headband device. Data to be processed includes linear acceleration, angular acceleration, the HIC, the GSI, and the PCS for each impact.

5.2.2. Level 1 Behavior Model

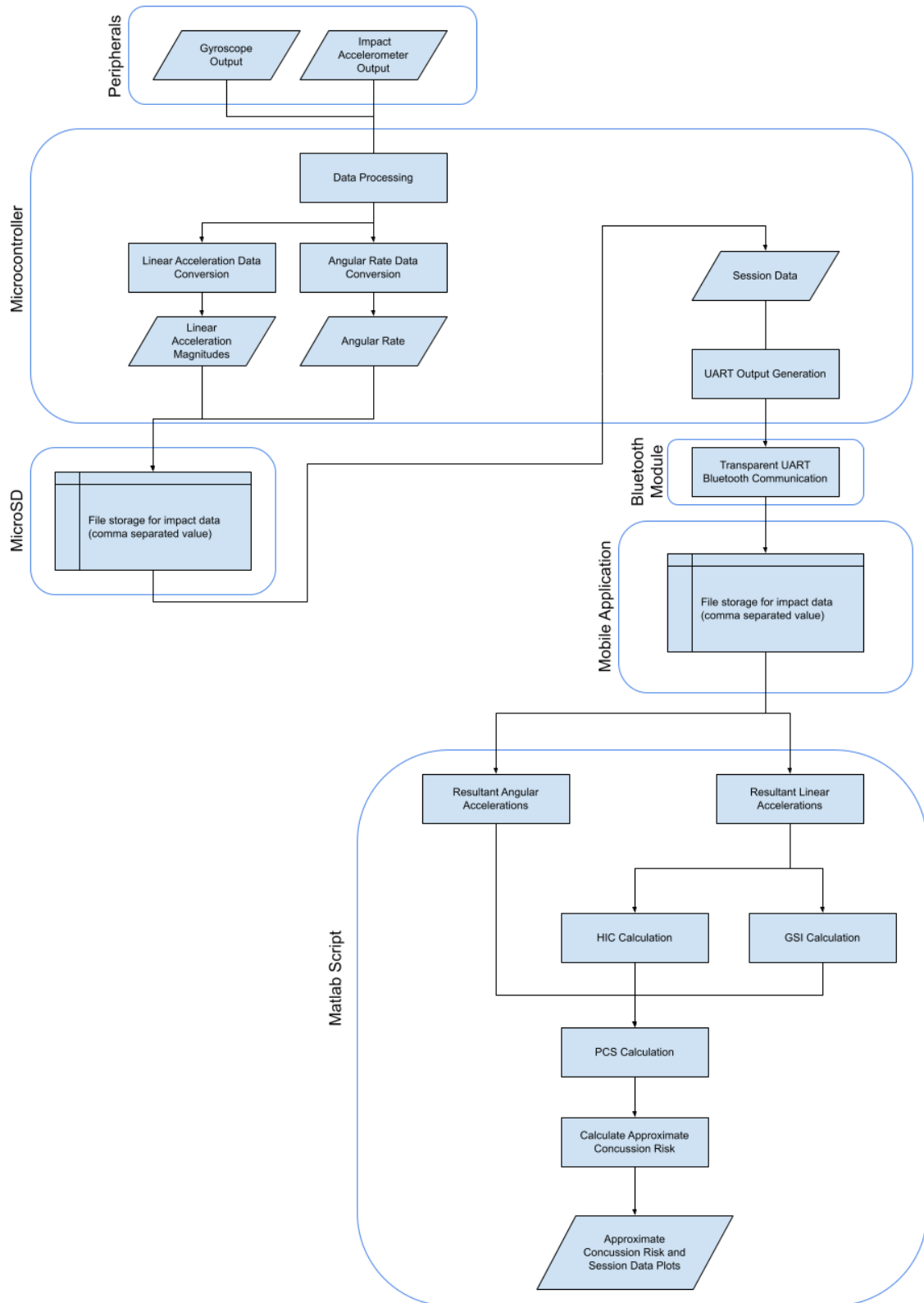


Figure 5.2.2 (previous page): Level 1 behavior model for step and impact data. The processing and storage blocks are separated by the subsystem at which the process or storage occurs. [JM]

Table 5.2.2.1: Level 1 Functional Requirements for Data Processing. [JM]	
Module	Data Processing
Designer	Noah Lewis and John Michel
Inputs	SPI digital signals of linear acceleration and angular velocity from the accelerometer and gyroscope.
Outputs	Unsigned characters giving the single-axis magnitudes of linear acceleration and angular velocity for the accelerometer and gyroscope, respectively.
Description	The data processing interface should transform digital signals into readable unsigned character values. The microcontroller should be able to potentially read data recorded on both the gyroscope and accelerometer simultaneously. This process should only occur when data exceeds each peripheral's minimum recording threshold. See section 5.2.6 for SPI interfacing code.

Table 5.2.2.2: Level 1 Functional Requirements for Linear Acceleration Data Conversion. [JM]	
Module	Linear Acceleration Data Conversion
Designer	John Michel
Inputs	Unsigned one-byte characters giving the single-axis magnitudes of linear acceleration for the accelerometer.
Outputs	Strings containing signed integers describing the linear acceleration magnitudes from the impact detection accelerometer as sent by the accelerometer.
Description	Data received from the accelerometer requires basic processing to convert the 8-bit unsigned character signals from the SPI bus into the 12-bit signed integer format used by both the accelerometer and gyroscope. This 12-bit signed integer is converted to a string via sprintf for output. The strings are output consecutively in comma separated value format, with a newline separating each impact.

Table 5.2.2.3: Level 1 Functional Requirements for Angular Rate Data Conversion. [JM]	
Module	Angular Rate Data Conversion
Designer	John Michel
Inputs	Unsigned one-byte characters giving the single-axis angular velocities for the gyroscope.
Outputs	Strings containing signed integers describing the angular velocities recorded by the gyroscope as sent by the gyroscope.
Description	Data received from the accelerometer requires basic processing to convert the 8-bit unsigned character signals from the SPI bus into the 12-bit signed integer format used by both the accelerometer and gyroscope. This 12-bit signed integer is converted to a string via sprintf for output. The strings are output consecutively in comma separated value format, with a newline separating each impact.

Table 5.2.2.4: Level 1 Functional Requirements for UART Output Generation. [JM]	
Module	UART Output Generation
Designer	John Michel
Inputs	Session data containing linear acceleration magnitudes and angular rate magnitudes stored in comma separated value format.
Outputs	UART transmission containing a digital UART signal describing the input data.
Description	The UART output is generated using the printf to UART function provided by the MPLAB Code Configurator. The implementation of the MPLAB Code Configurator for the PIC24FJ1024GB610 is detailed in section 5.2.4.

Table 5.2.2.5: Level 1 Functional Requirements for Transparent UART Bluetooth Communication. [JM]	
Module	Transparent UART Bluetooth Communication
Designer	John Michel
Inputs	UART signal containing session data sent by the microcontroller.
Outputs	Bluetooth transmission of the UART data sent by the microcontroller.
Description	The BM78 Bluetooth module is configured in Auto Pattern mode so that all data sent to it across its UART bus is automatically forwarded across an active Bluetooth connection. No coding is required, except to write the module's configuration to its EEPROM using the microcontroller. The configuration used for the BM78 evaluation board is shown in section 5.2.5.

Table 5.2.2.6: Level 1 Functional Requirements for Resultant LA Calculation. [JM]	
Module	Resultant LA Calculation
Designer	John Michel
Inputs	Signed integers giving the unconverted linear acceleration output from the accelerometer.
Outputs	Floats or doubles giving the three-dimensional magnitude of linear acceleration, also known as resultant linear acceleration (resultant LA).
Description	<p>First, the data is converted from the accelerometer output format of 0.1g per least significant bit (LSB) to gs, where g is the acceleration of gravity on Earth's surface (9.8 m/s²). Then the resultant LA is found. The resultant LA calculation is performed using the traditional 3D magnitude formula, given below:</p> $ a = \sqrt{a_x^2 + a_y^2 + a_z^2} \quad (3)$ <p>This computation may be expensive, so testing was done to examine the plausibility of performing this calculation on the microcontroller. Since the calculation is too expensive for the microcontroller to perform in real-time, this computation was moved to the mobile device.</p>

Table 5.2.2.7: Level 1 Functional Requirements for Resultant AA Calculation. [JM]	
Module	Resultant AA Calculation
Designer	John Michel
Inputs	Signed integers giving the unconverted angular rate output from the gyroscope.
Outputs	Floats or doubles giving the three-dimensional magnitude of angular acceleration, also known as resultant AA.
Description	<p>First, the angular rate data is converted from the gyroscope output format of 8.2 LSBs per degree per second. The angular rates are then numerically differentiated to obtain angular accelerations. Then the resultant AA is found. The calculation of resultant AA is performed using the traditional 3D magnitude formula, given below:</p> $ \alpha = \sqrt{\alpha_x^2 + \alpha_y^2 + \alpha_z^2} \quad (4)$ <p>Here, α_x, α_y, and α_z are the pitch, yaw, and roll observed by the gyroscope. This computation may be expensive, so testing was done to examine the plausibility of performing this calculation on the microcontroller. Since the calculation is too expensive for the microcontroller to perform in real-time, this computation was moved to the mobile device.</p>

Table 5.2.2.8: Level 1 Functional Requirements for the HIC Calculation. [JM]	
Module	Head Injury Criterion (HIC) Calculation
Designer	John Michel
Inputs	Resultant linear acceleration data for a given impact.
Outputs	A Head Injury Criterion score for the given impact.
Description	<p>The calculation is performed using the following formula:</p> $HIC^* = (t_2 - t_1) \left[1/(t_2 - t_1) * \int_{t_1}^{t_2} a(t) dt \right]^{5/2} \quad (1)$ <p>Here, $t_2 - t_1$ is fifteen milliseconds; this time frame is the time frame containing the impact in question. The linear acceleration $a(t)$ is integrated numerically using the trapezoid rule or another numerical integration technique for discrete data sets.</p>

Table 5.2.2.9: Level 1 Functional Requirements for the GSI Calculation. [JM]	
Module	Gadd Severity Index (GSI) Calculation
Designer	John Michel
Inputs	Resultant linear acceleration data for a given impact.
Outputs	A Gadd Severity Index score for the given impact.
Description	<p>The calculation is performed using the following formula:</p> $GSI = \int_0^T a(t)^{5/2} dt \quad (2)$ <p>Here, T is fifteen milliseconds; this time frame is the time frame containing the impact in question. The integration is performed numerically using the trapezoid rule or another numerical integration technique for discrete data sets.</p>

Table 5.2.2.10: Level 1 Functional Requirements for the PCS Calculation. [JM]	
Module	Principal Component Sum (PCS) Calculation
Designer	John Michel
Inputs	Resultant linear acceleration, resultant angular acceleration, the HIC, and the GSI for a given impact.
Outputs	A Principal Component Score (PCS) for the given impact.
Description	<p>The calculation is performed using the following formula:</p> $PCS = 10 \cdot ((0.4718 \cdot sGSI + 0.4742 \cdot sHIC + 0.4336 \cdot sLIN + 0.2164 \cdot sROT) + 2) \quad (5)$ <p>Here, sX is equal to $(X - \text{mean}(X))/SD(X)$ where mean(X) is the average value for X in the data set and SD(X) is the standard deviation of X (Greenwald et. al). Thus, this equation must utilize the mean and standard deviation of the original data set from the Greenwald et. al. study.</p>

Table 5.2.2.11: Level 1 Functional Requirements for Calculation of Concussion Risk. [JM]	
Module	Calculation of Concussion Risk
Designer	John Michel
Inputs	PCS values for each recorded impact.
Outputs	Approximate concussion risk.
Description	For each PCS value, the Positive Predictive Value (PPV) is found based on the Greenwald et. al. study of nearly 300,000 head impacts. The PPV describes the likelihood that a given PCS value is associated with a concussion. The multiplication of (1-PPV) for all PCS values yields an approximate likelihood that none of the impacts caused a concussion, and thus one minus that result is the likelihood that at least one concussion occurred.

5.2.3. Level 2 Software Behavior Models

5.2.3.1. Microcontroller Data Processing

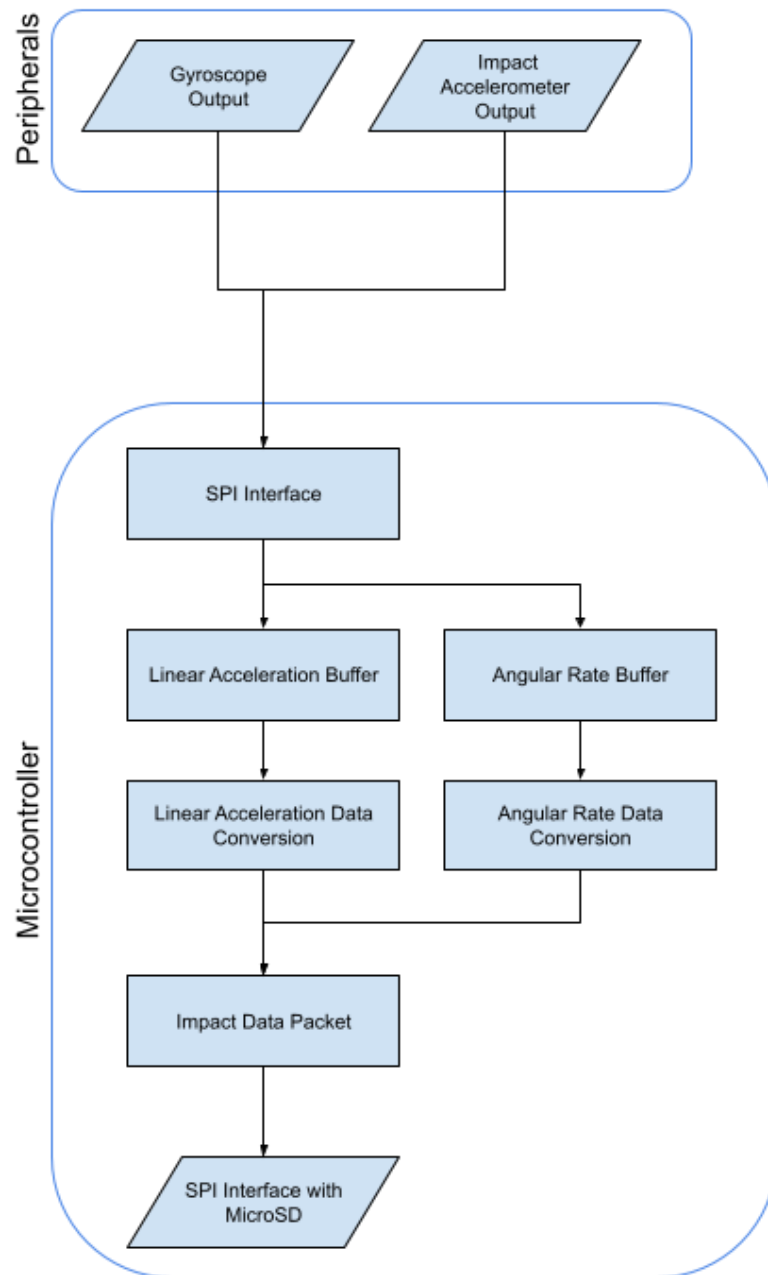


Figure 5.2.3.1: Level 2 behavior model for the data processing subprocess of the microcontroller. This subprocess interprets the input data from the accelerometer and gyroscope and sends the results to the microSD. [JD, JM]

Table 5.2.3.1.1: Level 2 Functional Requirements for SPI Interface. [JM]	
Module	SPI Interface
Designer	Noah Lewis and John Michel
Inputs	SPI signals from the accelerometer and gyroscope.
Outputs	Unsigned characters giving the single-axis magnitudes of linear acceleration and angular velocity for the accelerometer and gyroscope, respectively.
Description	The data processing interface should transform digital signals into readable unsigned character values. The microcontroller should be able to potentially read data recorded on both the gyroscope and accelerometer simultaneously. This process should only occur when data exceeds each peripheral's minimum recording threshold. See section 5.2.6 for SPI interfacing code.

Table 5.2.3.1.2: Level 2 Functional Requirements for Linear Acceleration Buffer. [JM]	
Module	Linear Acceleration Buffer
Designer	John Michel
Inputs	Unsigned characters giving the single-axis magnitudes of linear acceleration for the accelerometer.
Outputs	Unsigned characters giving the single-axis magnitudes of linear acceleration for the accelerometer for a given impact.
Description	The linear acceleration buffer holds data while data from the accelerometer is obtained. When an impact is detected by the accelerometer, the entirety of the FIFO buffer on the accelerometer can be read. The FIFO buffer can hold up to 169 3-axis data points. Thus, the buffer holds $169 \times 3 = 507$ data points in total (2 bytes per data point).

Table 5.2.3.1.3: Level 2 Functional Requirements for Angular Rate Buffer. [JM]	
Module	Angular Rate Buffer
Designer	John Michel
Inputs	Unsigned characters giving the single-axis angular rates for the accelerometer.
Outputs	Unsigned characters giving the single-axis angular rates for the accelerometer for a given impact.
Description	The angular rate buffer holds data while data from the gyroscope is obtained. When an impact is detected by the accelerometer, the entirety of the FIFO buffer on the gyroscope can be read. The FIFO buffer can hold up to 84 3-axis data points. Thus, the buffer holds $84 \times 3 = 252$ data points in total (2 bytes per data point).

Table 5.2.3.1.4: Level 2 Functional Requirements for Linear Acceleration Data Conversion. [JM]	
Module	Linear Acceleration Data Conversion
Designer	John Michel
Inputs	Unsigned one-byte characters giving the single-axis magnitudes of linear acceleration for a given impact.
Outputs	Strings containing signed integers describing the linear acceleration magnitudes from the impact detection accelerometer as sent by the accelerometer.
Description	Data received from the accelerometer requires basic processing to convert the 8-bit unsigned character signals from the SPI bus into the 12-bit signed integer format used by both the accelerometer and gyroscope. This 12-bit signed integer is converted to a string via printf for output. The strings are output consecutively in comma separated value format, with a newline separating each impact.

Table 5.2.3.1.5: Level 2 Functional Requirements for Angular Rate Data Conversion. [JM]	
Module	Angular Rate Data Conversion
Designer	John Michel
Inputs	Unsigned one-byte characters giving the single-axis angular velocities for a given impact.
Outputs	Strings containing signed integers describing the angular velocities recorded by the gyroscope as sent by the gyroscope.
Description	Data received from the accelerometer requires basic processing to convert the 8-bit unsigned character signals from the SPI bus into the 12-bit signed integer format used by both the accelerometer and gyroscope. This 12-bit signed integer is converted to a string via sprintf for output. The strings are output consecutively in comma separated value format, with a newline separating each impact.

Table 5.2.3.1.6: Level 2 Functional Requirements for the Impact Data Packet. [JM]	
Module	Impact Data Packet
Designer	John Michel
Inputs	Strings containing signed integers describing the linear acceleration magnitudes and angular velocities for a given impact.
Outputs	A data packet containing relevant data for a given impact, including the inputs listed above.
Description	The data packet is a comma separated value format containing the 169 3-axis linear acceleration data points followed by the 84 3-axis angular rate data points. A newline is printed following the impact data to separate impacts in the data file. This data packet is sent to the SPI interface and sent to the microSD for storage. The data file is later be accessed by the microcontroller to be transmitted to the mobile application via Bluetooth.

5.2.3.2. Microcontroller Data Transmission

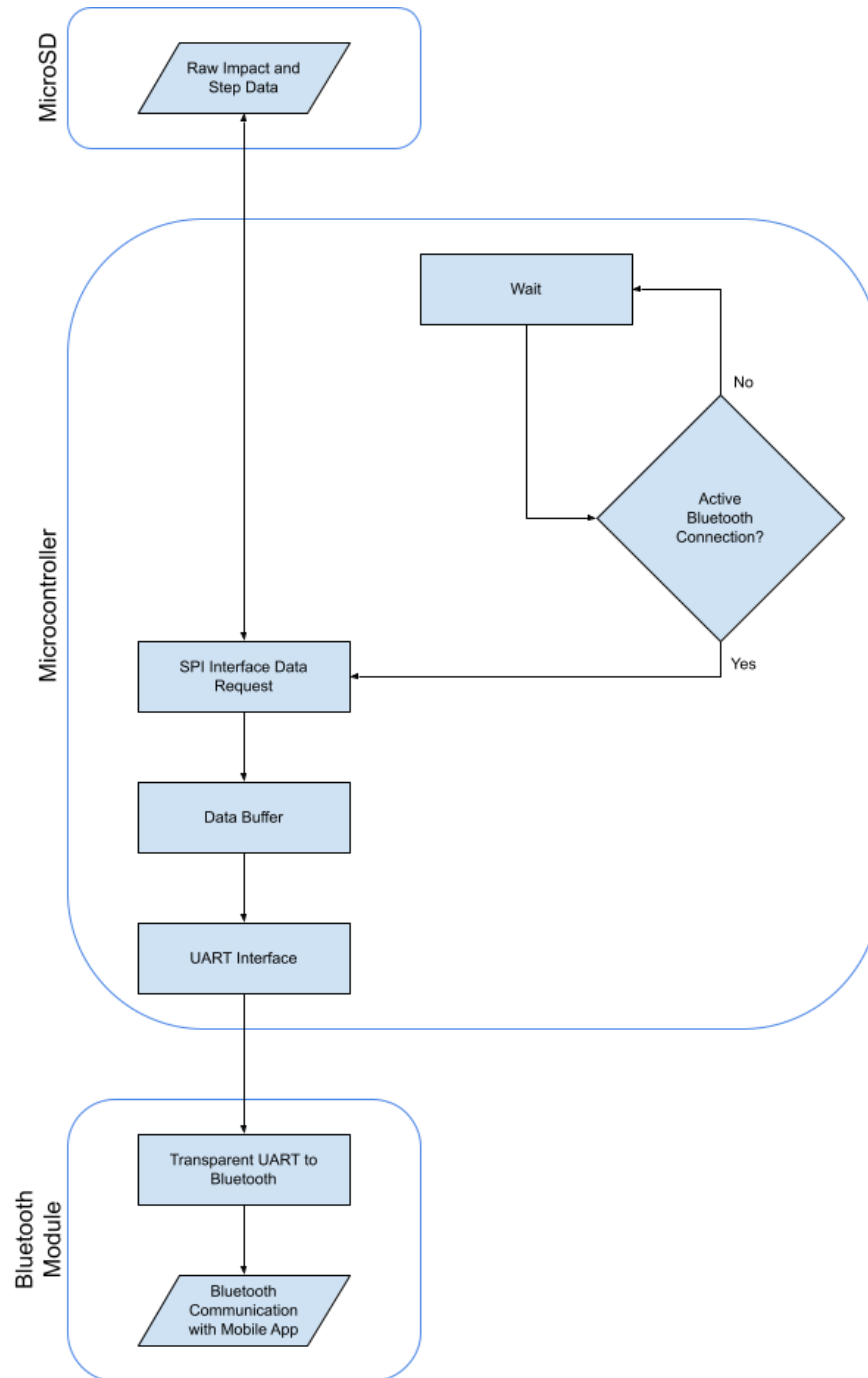


Figure 5.2.3.2: Level 2 behavior model for the data transmission subprocess of the microcontroller. This subprocess reads step and impact data from the microSD and sends it to the

Bluetooth module. [JM]

Table 5.2.3.2.1: Level 2 Functional Requirements for the Wait. [JM]	
Module	Wait
Designer	John Michel
Inputs	No active Bluetooth connection.
Outputs	None.
Description	The wait block is the default block for this subprocess. The wait block is only be exited when an active Bluetooth connection is detected. The Bluetooth connection is monitored via two status pins connected to the Bluetooth module.

Table 5.2.3.2.2: Level 2 Functional Requirements for the SPI Interface Data Request. [JM]	
Module	SPI Interface Data Request
Designer	Noah Lewis and John Michel
Inputs	Active Bluetooth connection and communication with the microSD.
Outputs	Data to be stored in preparation for transmission across Bluetooth.
Description	This block indicates the SPI request for data necessary for obtaining the data stored on the microSD. After the microSD is mounted and the data file is opened, the data is read into a data buffer until the end of file is reached.

Table 5.2.3.2.3: Level 2 Functional Requirements for the Data Buffer. [JM]	
Module	Data Buffer
Designer	John Michel
Inputs	Data obtained from the microSD.
Outputs	Data ready to be sent to the Bluetooth module.
Description	This block indicates a data buffer that holds the data sent over SPI by the microSD. This data is stored locally on the microprocessor and then sent to the Bluetooth module via UART. This buffer can be iteratively used via a maximum buffer size to limit data storage requirements for the microcontroller.

Table 5.2.3.2.4: Level 2 Functional Requirements for the UART Interface. [JM]	
Module	UART Interface
Designer	John Michel
Inputs	Data ready to be sent to the Bluetooth module.
Outputs	A digital UART signal on the TX line of the microprocessor.
Description	Using the printf to UART functionality, the session data is sent on the TX line of microprocessor to the Bluetooth module's RX line. For more information, see section 5.2.5.

Table 5.2.3.2.5: Level 2 Functional Requirements for the Transparent UART to Bluetooth. [JM]	
Module	Transparent UART to Bluetooth
Designer	John Michel
Inputs	UART data stream from the microprocessor.
Outputs	A Bluetooth signal containing the UART data.
Description	The Transparent UART Auto Pattern mode automatically forwards data received on the UART line across the Bluetooth connection. For more information, see section 5.2.5.

5.2.3.3. Matlab Script Computations

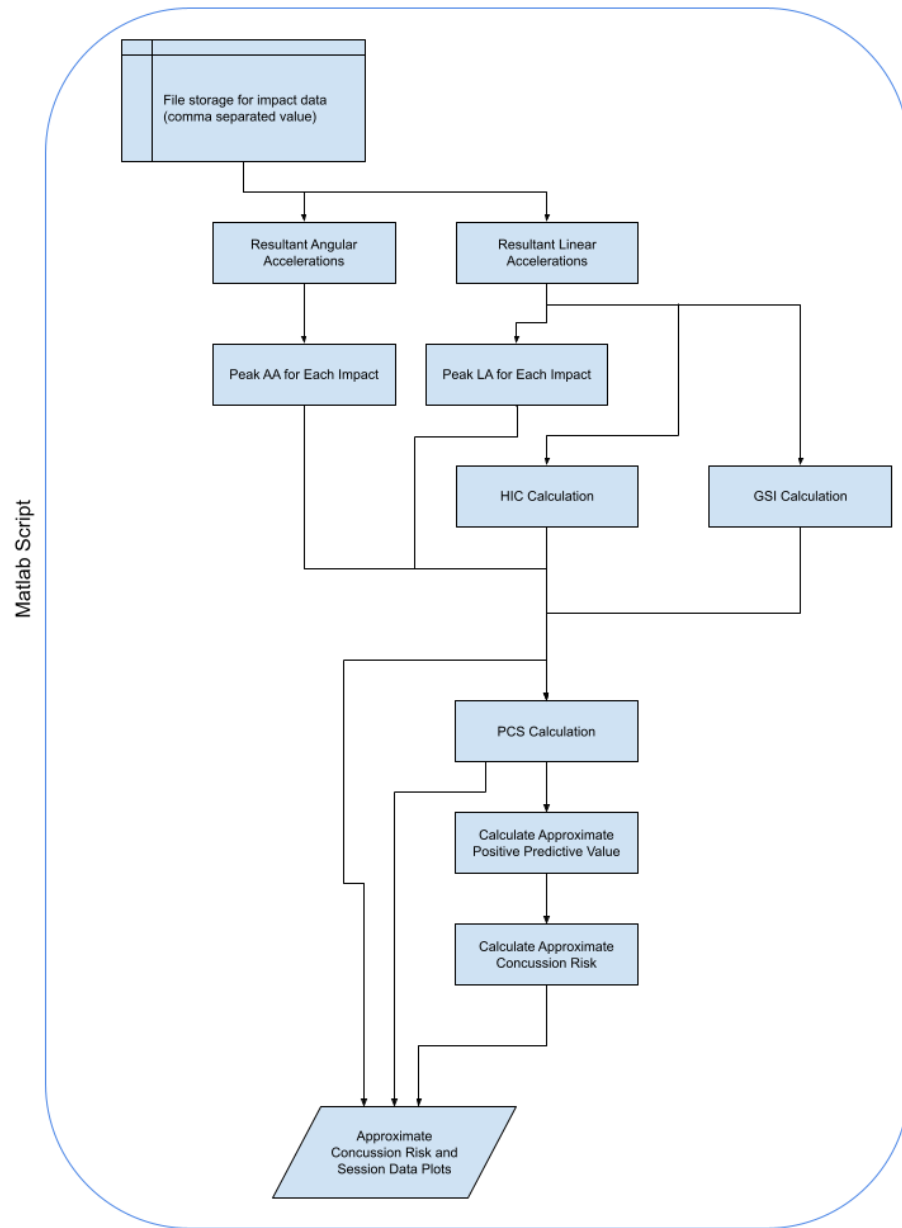


Figure 5.2.3.3: Level 2 behavior model for the Matlab script computation subprocess.

This subprocess interprets the impact data sent by the headband device and calculates several parameters to assess concussion risk. This script was intended to be exported and used in the mobile application; future work should implement this script within a mobile application. [JM]

Table 5.2.3.3.1: Level 2 Functional Requirements for Resultant Angular Accelerations. [JM]	
Module	Resultant Angular Accelerations
Designer	John Michel
Inputs	Signed integers giving the unconverted angular rate output from the gyroscope.
Outputs	Floats or doubles giving the three-dimensional magnitude of angular acceleration, also known as resultant AA.
Description	<p>First, the angular rate data is converted from the gyroscope output format of 8.2 LSBs per degree per second. The angular rates are then numerically differentiated to obtain angular accelerations. Then the resultant AA is found. The calculation of resultant AA is performed using the traditional 3D magnitude formula, given below:</p> $ \alpha = \sqrt{\alpha_x^2 + \alpha_y^2 + \alpha_z^2} \quad (4)$ <p>Here, α_x, α_y, and α_z are the pitch, yaw, and roll observed by the gyroscope. This computation may be expensive, so testing was done to examine the plausibility of performing this calculation on the microcontroller. Since the calculation is too expensive for the microcontroller to perform in real-time, this computation was moved to the mobile device.</p>

Table 5.2.3.3.2: Level 2 Functional Requirements for Resultant Linear Accelerations. [JM]	
Module	Resultant Linear Accelerations
Designer	John Michel
Inputs	Signed integers giving the unconverted linear acceleration output from the accelerometer.
Outputs	Floats or doubles giving the three-dimensional magnitude of linear acceleration, also known as resultant linear acceleration (resultant LA).
Description	<p>First, the data is converted from the accelerometer output format of 0.1g per least significant bit (LSB) to gs, where g is the acceleration of gravity on Earth's surface (9.8 m/s²). Then the resultant LA is found. The resultant LA calculation is performed using the traditional 3D magnitude formula, given below:</p> $ a = \sqrt{a_x^2 + a_y^2 + a_z^2} \quad (3)$ <p>This computation may be expensive, so testing was done to examine the plausibility of performing this calculation on the microcontroller. Since the calculation is too expensive for the microcontroller to perform in real-time, this computation was moved to the mobile device.</p>

Table 5.2.3.3.3: Level 2 Functional Requirements for Peak AA for Each Impact. [JM]	
Module	Peak AA for Each Impact
Designer	John Michel
Inputs	Floats or doubles giving the three-dimensional magnitude of angular acceleration, also known as resultant AA.
Outputs	Maximum recorded resultant AA for each impact.
Description	The maximum magnitude of angular acceleration is used in the calculation of the PCS value for each impact.

Table 5.2.3.3.4: Level 2 Functional Requirements for Peak LA for Each Impact. [JM]	
Module	Peak LA for Each Impact
Designer	John Michel
Inputs	Floats or doubles giving the three-dimensional magnitude of linear acceleration, also known as resultant linear acceleration (resultant LA).
Outputs	Maximum recorded resultant LA for each impact.
Description	The maximum magnitude of linear acceleration is used in the calculation of the PCS value for each impact.

Table 5.2.3.3.5: Level 2 Functional Requirements for the HIC Calculation. [JM]	
Module	Head Injury Criterion (HIC) Calculation
Designer	John Michel
Inputs	Resultant linear acceleration data for a given impact.
Outputs	A Head Injury Criterion score for the given impact.
Description	<p>The calculation is performed using the following formula:</p> $HIC^* = (t_2 - t_1) \left[1/(t_2 - t_1) * \int_{t_1}^{t_2} a(t)dt \right]^{5/2} \quad (1)$ <p>Here, $t_2 - t_1$ is fifteen milliseconds; this time frame is the time frame containing the impact in question. The linear acceleration $a(t)$ is integrated numerically using the trapezoid rule or another numerical integration technique for discrete data sets.</p>

Table 5.2.3.3.6: Level 2 Functional Requirements for the GSI Calculation. [JM]	
Module	Gadd Severity Index (GSI) Calculation
Designer	John Michel
Inputs	Resultant linear acceleration data for a given impact.
Outputs	A Gadd Severity Index score for the given impact.
Description	<p>The calculation is performed using the following formula:</p> $GSI = \int_0^T a(t)^{5/2} dt \quad (2)$ <p>Here, T is fifteen milliseconds; this time frame is the time frame containing the impact in question. The integration is performed numerically using the trapezoid rule or another numerical integration technique for discrete data sets.</p>

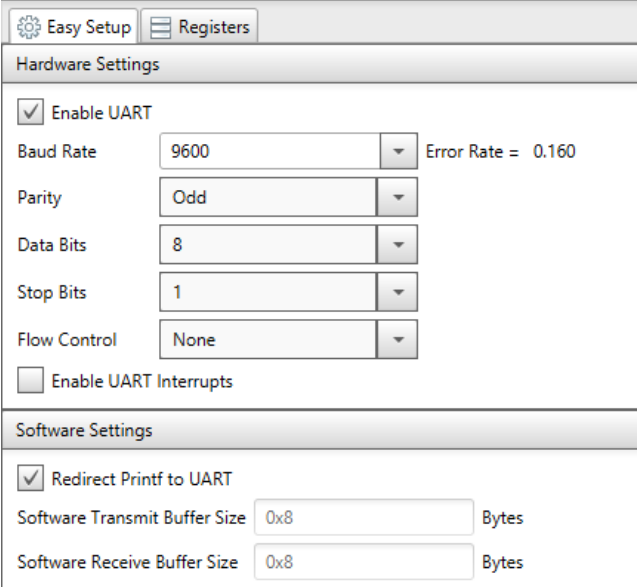
Table 5.2.3.3.7: Level 2 Functional Requirements for the PCS Calculation. [JM]	
Module	Principal Component Sum (PCS) Calculation
Designer	John Michel
Inputs	Resultant linear acceleration, resultant angular acceleration, the HIC, and the GSI for a given impact.
Outputs	A Principal Component Score (PCS) for the given impact.
Description	<p>The calculation is performed using the following formula:</p> $PCS = 10 \cdot ((0.4718 \cdot sGSI + 0.4742 \cdot sHIC + 0.4336 \cdot sLIN + 0.2164 \cdot sROT) + 2) \quad (5)$ <p>Here, sX is equal to $(X - \text{mean}(X))/\text{SD}(X)$ where mean(X) is the average value for X in the data set and SD(X) is the standard deviation of X (Greenwald et. al). Thus, this equation must utilize the mean and standard deviation of the original data set from the Greenwald et. al. study.</p>

Table 5.2.3.3.8: Level 2 Functional Requirements for Calculate Approximate Positive Predictive Value. [NL, JM]	
Module	Calculate Approximate Positive Predictive Value
Designer	John Michel
Inputs	PCS values for each recorded impact.
Outputs	Approximate PPV for each recorded impact.
Description	For each PCS value, the Positive Predictive Value (PPV) is found based on the Greenwald et. al. study of nearly 300,000 head impacts. The PPV describes the likelihood that a given PCS value is associated with a concussion.

Table 5.2.3.3.9: Level 2 Functional Requirements for Calculate Approximate Concussion Risk. [NL, JM]	
Module	Calculate Approximate Positive Predictive Value
Designer	John Michel
Inputs	Approximate PPV for each recorded impact.
Outputs	Approximate concussion risk.
Description	The value (1-PPV) is multiplied for all impacts across the session. This should provide an approximate value of the likelihood that the user did not have a concussive impact during the session. Thus, one minus that value yields the likelihood of at least one concussion occurring.

5.2.4. UART Communication

Communication was established between the PIC24FJ1204GB610 microcontroller and the BM78 Bluetooth module via UART. This communication required TX and RX lines, connected on pins 44 and 43 of the microcontroller, respectively. Manual coding of this communication was avoided with the use of the MPLAB Code Configurator, which is a UI plugin for MPLAB X that configures a target microcontroller. This includes setup of pin assignments, communication modules, and more. This project utilizes the UART1 configuration for communication between the Bluetooth module and the PIC24 microcontroller. The UART1 configuration is shown below. Note that the baud rate may be adjusted for higher speed communication if possible. Also note the printf to UART functionality; this drastically simplifies UART communication for this project. The only configuration and coding required beyond the printf() function is the definition of the RX and TX pins, as seen in the Code Configurator package view (Figure 12.1.1 in the appendices). [JM]



UART1

Easy Setup Registers

Hardware Settings

Enable UART

Baud Rate: 9600 Error Rate = 0.160

Parity: Odd

Data Bits: 8

Stop Bits: 1

Flow Control: None

Enable UART Interrupts

Software Settings

Redirect Printf to UART

Software Transmit Buffer Size: 0x8 Bytes

Software Receive Buffer Size: 0x8 Bytes

Figure 5.2.4: UART1 configuration settings for the PIC24FJ1024GB610. [JM]

5.2.5. Bluetooth Communication

For the Bluetooth module's software, a configuration file can be written to the module to define its operation settings. The Bluetooth module can be configured in Auto Pattern mode so that any UART communication is automatically forwarded to the Bluetooth connection while the BM78 operates in application mode. The settings are updated using a Windows UI tool, then saved as a .txt file. This file can be written to the BM78 EEPROM ROM using either a USB or UART connection. The most recently used relevant configuration settings are shown below.

[JM]

The screenshot shows a configuration window with the following settings:

Section	Parameter	Value	Unit/Range	Help	
Device Information	BT Address	<input type="checkbox"/> Enable	0x []	[6 Bytes]	Help
	Class of Device	0x	040424	[3 Bytes]	
	Device Name		BM78	[16 characters]	
	PIN Code		0000	[4 ~ 16 Number]	
	BT3.0 UUID	0x	0000	[2 Bytes]	
SPP Server Setting	RFCOMM Server Channel		0x00 : default	Help	
Uart Setting	HCI Baud Rate Index		0x09 : 9600	Help	
	Parity Check		Disable		
	Stop Bits		1 bit		
	Parity Mode		ODD		
	H/W Flow Control (CTS)		Disable		

Beacon Setting

System Setup | System Setup2 | LE Mode Setup | LE DIS Setup | LED Setup | Description Setup | MFi Setup

--- Max BR/EDR Data Segment 0x 0000 [2bytes]
 (0x0000:Disable,0x0018~0xFFFF , unit: byte)

--- UART RX_IND Disable

--- Max BLE Data Segment Size On Air Segment UART RX da

--- Check UART RX Data Interval 0x 00
 (unit: 0.625ms) total : 0.000 ms

Operation Mode Setting

--- BT Operation Mode Dual Mode Help

--- Operation Pattern Auto Pattern

--- Configure Mode Timeout 0x 06
 (0:Disable Configure Mode, unit: 640ms) total : 3840 ms

Sniff Mode Setting

--- Sniff Interval 0x 0320 [2bytes] Help
 (unit: 0.625ms) total : 500.000 ms

--- Enter Sniff Waiting Time 0x 7D
 (0x01~0xFF,unit: 80ms) total : 10000 ms

Security Setting

--- Pairing Method Just Work Help

--- Bluetooth 3.0 Pairing Mode Security Mode 4 (SSP)

--- Bluetooth 4.0 BLE Security Disable

--- BLE User Confirm Option Disable LE user confi

--- Trust Device Connection Disable

Figures 5.2.5.1: Bluetooth module EEPROM configuration, as shown in the UI tool. [JM]

The Bluetooth module itself requires startup code to define the correct operating mode and status pins. The Bluetooth mode must be set to operate in ROM application mode to correctly load the configuration file detailed above. Furthermore, the status pins P0_4 and P1_5 must be set as inputs so that the microcontroller can read the statuses as needed. The Bluetooth module startup code used by the microcontroller can be found in lines 199 through 228 of main.c in the appendices. [JM]

The Bluetooth module can also provide status updates during runtime about the state of Bluetooth connectivity via two status pins, label P0_4 and P1_5. The statuses are given by the table below, which is in the BM78 Bluetooth module data sheet. For usage of the status pins' values within the code, see main.c in the appendices. [JM]

TABLE 2-4: STATUS INDICATION

P1_5/STATUS_IND_1	P0_4/STATUS_IND_2	Indication
H	H	Power-on (default setting) and Deep Sleep state. HH status should be stable for at least 500 ms.
H	L	Access state.
L	H	Link state (UART data transmitting).
L	L	Link state (no UART data transmitted).

Legend: L = Low H = High

Table 5.2.5.2: Bluetooth module status indication. [JM]

5.2.6. SPI Peripheral Communication and Peripheral Operation

SPI communication is used for communication with the accelerometer and gyroscope. SPI uses Serial Data Out (SDO/MOSI), Serial Data In (SDI/MISO), and the Serial Clock (SCLK) for each SPI Bus, along with a specific Chip Select (CS) for each peripheral. The microcontroller is connected to the accelerometer and gyroscope on the same SPI bus, with a chip select for each peripheral. Microchip Code Configurator's SPI library sets the SPI1 bus configuration and adds macro functions for SPI communication. The microcontroller calls the SPI1_Exchange8bitBuffer function to initiate an SPI communication with a peripheral. The first byte contains the address to read or write on the peripheral, followed by a stream of data. The function specifies the number of bytes to transfer; for a multibyte read, if n bytes are to be read, the byte transfer total should be n+1. The SPI1 MCC configuration is shown below. For pin layouts, see the package view (Figure 12.1.1) in the appendices. [JM]

SPI1

Easy Setup Registers

Hardware Settings

Mode Selection Master

Enable SPI

Communication Width 8 bit

SPI Clock

Baudrate Generator Value: 0x0 ≤ 0x8 ≤ 0xFF

SPI Clock Frequency 222.22222 kHz

SPI Mode

Clock Polarity Idle:Low, Active:High

Clock Edge Active to Idle

SPI Mode 0

Input Data Sampled at Middle End

Figure 5.2.6.1: MCC SPI1 bus configuration.

The SPI communication is first used in the initialization of the accelerometer and gyroscope. Both modules require initial setup to determine the internal configuration of the modules. In particular, the accelerometer is set up to perform in impact detection mode, which prevents the FIFO queue from filling until the accelerometer detects a magnitude of 10g or greater. The accelerometer also operates its FIFO queue in oldest saved mode to prevent unprocessed data from being overwritten. [JM]

Once the impact detection trigger activates, the FIFO queue fills at the sample rate defined in the configuration. The external interrupt of the accelerometer is set to trigger when the FIFO queue is full. This interrupt is tied to an external interrupt on the microcontroller which is set up via the External Interrupt module of MCC. When the microcontroller processes this interrupt, an interrupt service routine is triggered which reads the FIFO queue from the accelerometer via SPI communication, as detailed above. The gyroscope's FIFO queue, which operates in standard FIFO mode (i.e. old data is overwritten), is also read when the interrupt is triggered. The data from each peripheral is stored locally on the microcontroller and then output to the data file on the microSD. For full code implementation of this interrupt service routine, see `ext_int.c` in the appendices. [JM]

5.2.7. microSD Communication

The microSD also utilizes SPI communication, but because its SPI communication is complex, an additional software library called FatFs was utilized to communicate with the microSD. The microSD uses the FAT file system; hence, the name FatFs stands for FAT File System. The FatFs library sets up SPI communication as well as the SPI commands required to communicate with and initialize the microSD for data transfer. The FatFs library is compatible

with Microchip Code Configurator and requires several configurations, shown below. The FatFs library has multiple dependencies, including an SPI library (SPI2 used here), an SD card SPI library, and an SPI master configuration. [JM]

FatFs

Easy Setup

Information Configuration


Generate example/demo files

▼ Physical Driver Selection

Select the physical drivers used by the file system and the drive name/label associated with that physical driver. An example template driver is provided as a starting point for physical devices/drivers not supported inside of MCC.

Select a physical layer driver:

Example_Template_Driver + Insert Driver

Number	Label	Driver	Remove
0	DRVA	SD Card (SPI)	

▼ Feature Selection and Optimizations

Select which functions/features will be supported in the library. Enabling features will increase the code footprint required.

Function optimization: 0: Basic functions are fully enabled

String function optimization: 0: Disable string functions.

Enable functions that write to the drive (f_write, f_mkdir, f_rename, etc.)

Enable relative path support (f_chdir and f_chdrive)

Enable get working directory support (f_getcwd)

Enable find/search functions (f_find)

Enable formatting functions (f_mkfs)

Figure 5.2.7.1: MCC FatFs library configuration. Note that an SD card (SPI) driver is added to support the physical microSD. Also note that the function optimization has been selected to enable all basic functions. [JM]

SPI2

Easy Setup

Hardware Settings

Default SPI Clock Frequency kHz

Actual Clock Frequency 125.0 kHz

Figure 5.2.7.2: MCC SPI2 library configuration. This configuration primarily sets up the default clock speed and the necessary pins for SPI communication. [JM]

SPIMASTER

Easy Setup

Hardware Settings

Name	SPI Mode	SPI Data Input...	Speed (kHz)	Actual Speed (...)	SPI
SDSLOW2	MODE3	MIDDLE	250	250	SPI2
SDFAST2	MODE3	MIDDLE	250	250	SPI2
SDSLOW	MODE3	MIDDLE	400	400	SPI2
SDFAST	MODE3	MIDDLE	10000	2,000	SPI2

Figure 5.2.7.3: MCC SPI master library configuration. This configures the SPI clock speeds used by the SD SPI library. [JM]

SD Card (SPI)

Easy Setup

SD-Card Settings

Pin Enable

Enable Chip Select (CS)

Enable Card Detect (CD)

Enable Write Protect (WP)

Polarity

Active Low

Active High

Active High

Figure 5.2.7.4: MCC SD SPI library configuration. This configures any additional features used by the SPI communication. Card detect and write protect were deemed unnecessary for the purpose of this project. [JM]

This project employs the FatFs library when writing impact data to the data file and when reading the data file from the microSD for output to the Bluetooth connection. The implementation of the data file write can be found in `ext_int.c` in the appendices. The implementation of the data file read was not entirely completed; the data read was not completed correctly in `main.c`, but its most advanced stage is shown in `main.c` lines 251 to 275. For more details about the implementation of the data file read and subsequent transmission over Bluetooth, see section 10.1.4. [JM]

5.2.8. Mobile Application and Matlab Script

The mobile application is based on the Microchip Bluetooth Data application, which is provided by Microchip and has source code publicly available for download at the link found at "Bluetooth Low Energy" in the references. This source code was downloaded and compiled so that a working Bluetooth application could be used as the basis for mobile application development for this project. The mobile application was only modified briefly to allow it to automatically save any Bluetooth communications received to a text file. This text file can then be exported to the Matlab script to process any impact data received. The code for the mobile application is complex and the changes made were minimal, so the code is not shown in this report. For the original source code, see "Bluetooth Low Energy" in the references. [NL, JM]

The initial design for this project intended to implement the code within the Matlab script in the mobile application. It was determined that a Matlab script should be exportable using Matlab's Compiler SDK, which compiles Matlab code for other languages such as C++ and Java. In theory, the content in the Matlab script should be implemented within the mobile application, but this has been left for future work; see section 10.1.5 for more information. For the purposes of this design report, the Matlab script and mobile application are often used interchangeably since the Matlab script is designed to be exported into the mobile application. [JM]

The Matlab script processes the impact data recorded by the microcontroller onto the microSD. The impact data file can be extracted from the microSD via a microSD to USB converter and then placed in the same folder as the Matlab script so it can be called within the code. In theory, the impact data file could be sent over Bluetooth and then called by the Matlab script within the mobile application, but this has been left for future work. The Matlab script processes the impact data, separating each impact as well as the linear accelerations and angular

rates within each impact. The script calculates resultant linear and angular acceleration as well as the HIC and GSI for each impact. The peak linear and angular accelerations and the HIC and GSI are used to determine a PCS score for each impact. These scores are analyzed using results from the study of nearly 300,000 impacts by Greenwald et. al. and an approximate positive predictive value (PPV) is determined for each impact. This value describes the likelihood that an impact at the recorded PCS value or higher would result in a concussion. The multiplication of (1-PPV) for each impact yields the approximate likelihood that no concussion occurred across all recorded impacts. Finally, one minus this result gives the likelihood that at least one concussion occurred within the impact data. This value is an approximate concussion risk value. Note that this value is approximate and is designed to give the user an approximation of how likely it is that they sustained a concussion during a session; it should not be treated as an accurate medical likelihood of concussion. [JM]

The Matlab script also provides histograms of the recorded data. For the purposes of this project, all recorded linear and angular accelerations are shown in histograms since the sample size of impacts was often small for demonstrations of the headband impact recording system. Additionally, the HIC and GSI are shown for each impact. [JM]

To demonstrate the concussion risk calculation, a botched sample set of data was created. The sample set contained three impacts. The first impact contains values that are just over the accelerometer's recording threshold, but do not reach values that should indicate any risk of concussion. The second impact contains some values that are relatively large and should indicate some concussion risk. The third impact is a copy of the second. The results of this simulation, including the positive predictive values for each impact, are shown below. For the full code of the Matlab script, see the appendices in section 12.2. [JM]

```

PPVs =
    0
    0.0071
    0.0071

percentage_risk =
    1.4229

```

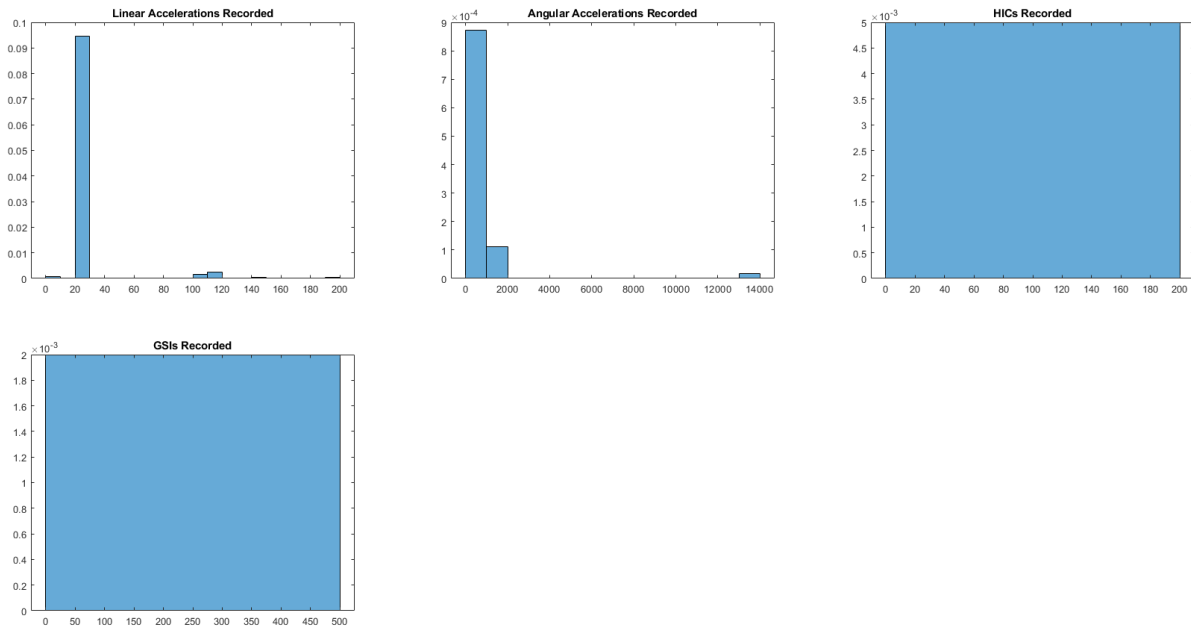


Figure 5.2.8.1: Matlab script result using a botched dataset intended to demonstrate the concussion risk calculation. Here, three impact data sets were manually created to demonstrate the concussion risk calculation in the case of high acceleration impacts. Note that the concussion risk is nonzero because the impacts contained linear accelerations and angular accelerations with high magnitudes, as indicated in the figures. Also note that the positive predictive values for the second and third impact are identical, as expected. [JM]

The results of the Matlab script when run upon three impacts recorded by the headband prototype in the demonstration are shown below. Note that since the impacts were small in magnitude (less than 20g peak linear acceleration and less than 1500 rad/s² peak angular acceleration), the percentage risk of concussion was zero, indicating a concussion risk not significantly over zero percent. For the full code of the Matlab script, see the appendices in section 12.2. [JM]

```
percentage_risk =
0
```

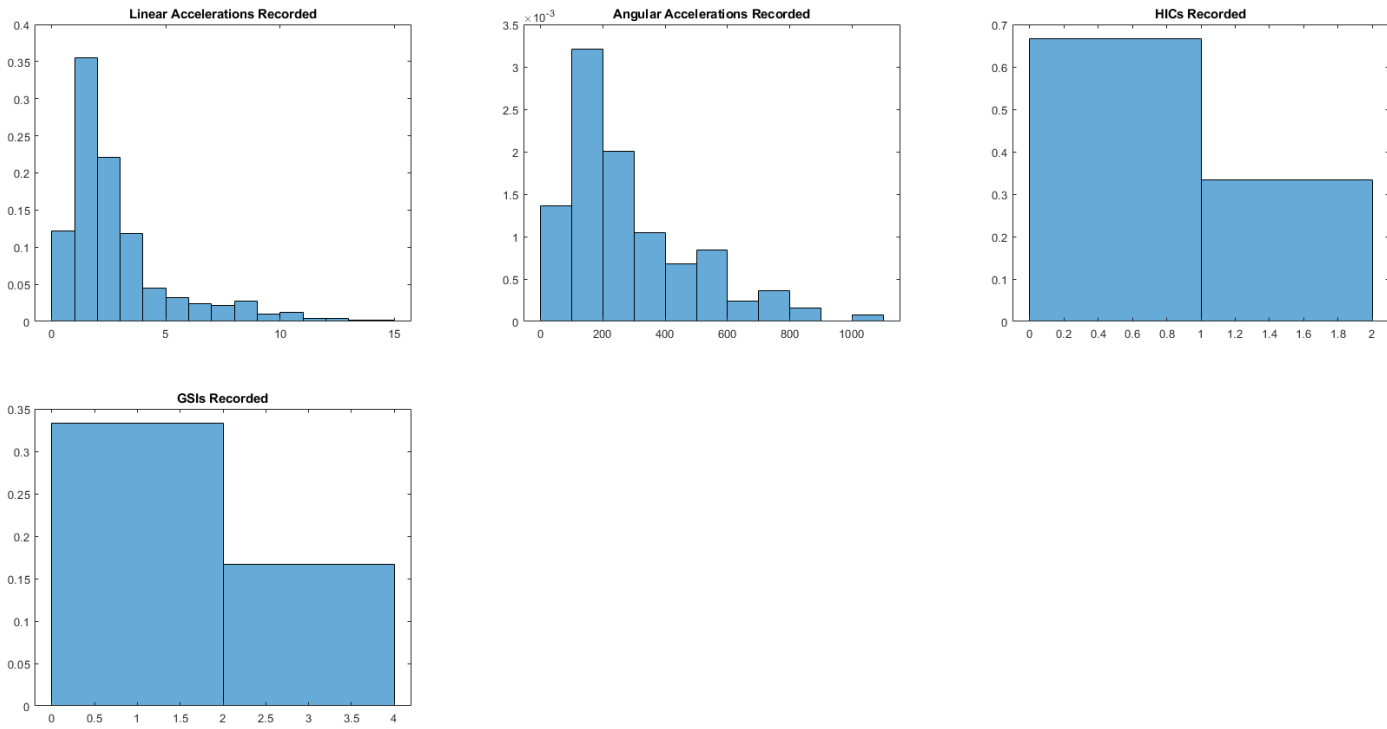


Figure 5.2.8.2: Results of the Matlab script when run upon three sample impacts recorded during a demonstration.

5.3. Testing

The accuracy of the measurement devices was tested via the methods detailed in section 5.1.6. The data obtained from that test was analyzed by the Matlab script as detailed in section 5.2.8. The test data is not intended to indicate a potentially concussive impact, but rather an impact that shows an approximate 5g linear acceleration after the weight is dropped. The initial acceleration caused by tension of the rope is enough to trigger the impact detection threshold. Further data values should exhibit approximately a 5g linear acceleration as the block drops due to gravity. Figure 5.3.2 indicates that the section of accelerations on or about 5g was significantly higher than expected by the right-skewed data. This indicates that the accelerometer is measuring the linear acceleration correctly and that data processing is correctly converting that data into gs. Future testing should examine higher target linear acceleration values and should verify that the data is being recorded in the appropriate order (i.e. highest impact first, followed by a section of data approximately equal to the target acceleration, followed by reduced values upon deceleration). [JM]

For angular acceleration, the test was designed using the sinusoidal nature of a pendulum. The resulting data shows that the angular acceleration that was recorded matches this sinusoidal pattern. The peak acceleration occurs at the lowest point in its swing pattern and is over the quickest. The pendulum then swings upwards, losing momentum and recording slower acceleration. This shows that the gyroscopes were recording data correctly. Future tests should examine high angular acceleration values to show that the gyroscope is able to record larger impacts. [JD]

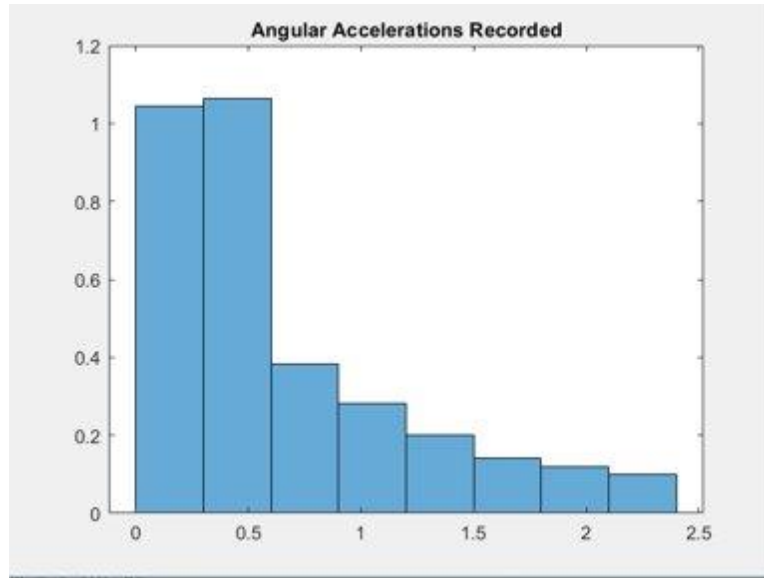


Figure 5.3.1: Angular Acceleration test [JD, JM]

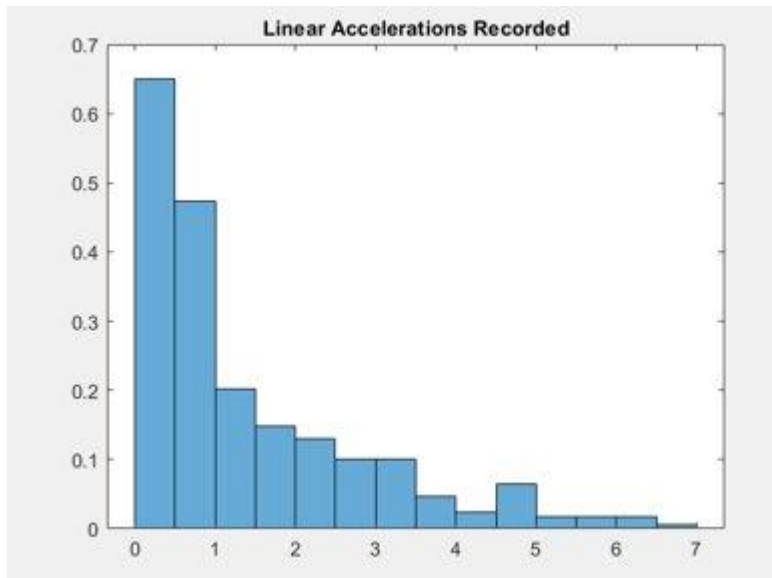


Figure 5.3.2: Linear Acceleration [JD, JM]

6. Mechanical Sketch

The headband was designed using SolidWorks, with the goal of it being 3D printed to hold all of the components, while also being comfortable for the user. The left side would contain the microcontroller, with all of the main peripherals and microprocessor, while the right side would house the battery components, with power wires running through holes and on the side of the headband to connect power to the microcontroller. The front of the headband also had a second extrusion to provide protection for the power wires from anything that could potentially damage them.

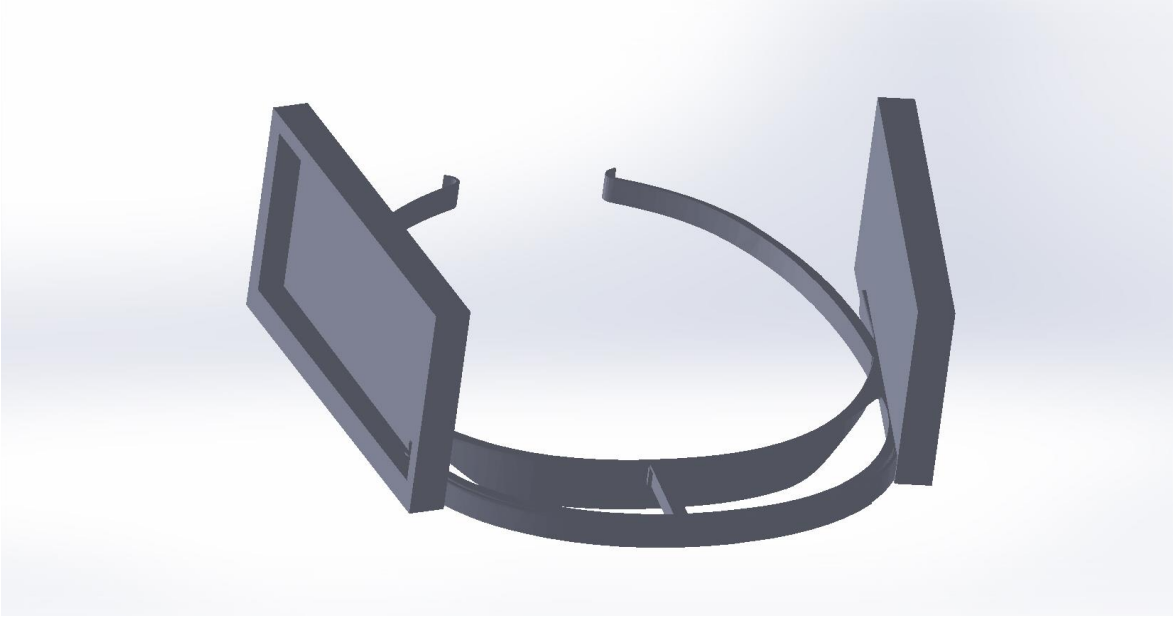


Figure 6.1: Headband Side View [NL]

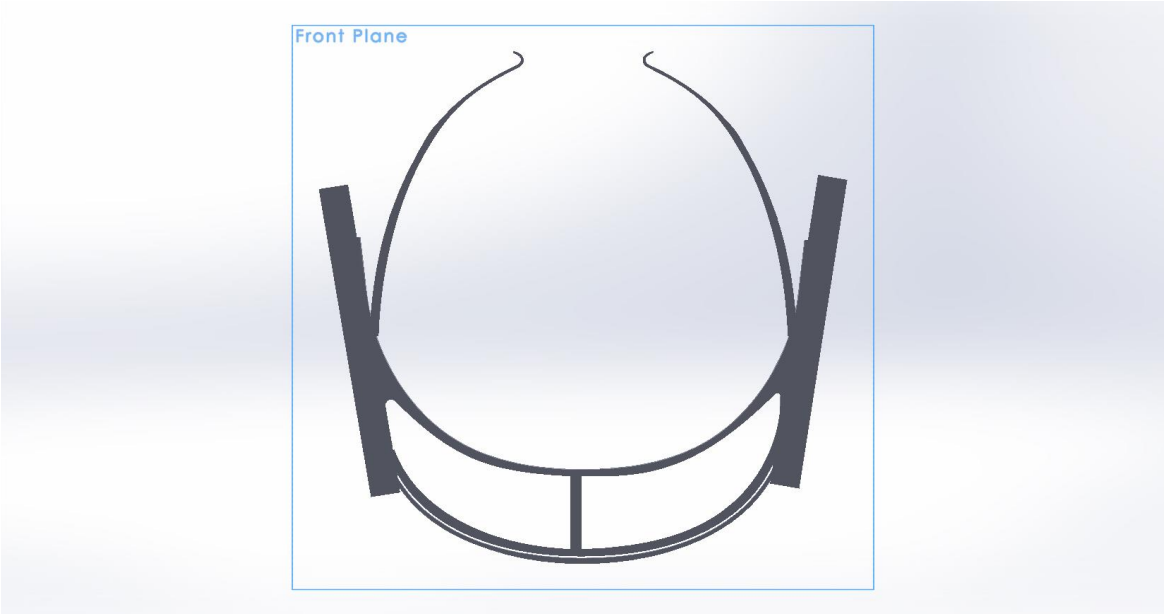


Figure 6.2: Headband Top View [NL]

7. Parts Lists

7.1. Parts List

Qty.	Refdes	Part Num.	Description
1		LSM9DS1	IMU ACCEL/GYRO/MAG I2C/SPI 24LGA with Breakout board
1		BM78SPPS5MC2-0002AA	BM78 PICtail Plus
1		ADXL372	MEMS Accelerometer with board
1		PCM12SMTR	manual on/off switch
1		MCP73831	Battery management system
1		BQ26220	coulomb counter
3		ITG-3701	Gyroscope
1		ADXL372	MEMS Accelerometer
1		BM78SPPS5MC2-0002AA	BM78 Bluetooth Dual-Mode Module

Table 7.1: Parts list [JD, NL]

7.2. Materials Budget List

Qty.	Part Num.	Description	Cost	Total Cost
1	LSM9DS1	IMU ACCEL/GYRO/MAG I2C/SPI 24LGA with Breakout board	\$15.95	\$15.95
1	BM78SPPS5MC2-0002AA	BM78 PICtail Plus	\$89.99	\$89.99
1	ADXL372	MEMS Accelerometer with board	55.00	55.00
5	PCM12SMTR	manual on/off switch	0.92	4.60
5	MCP73831	Battery management system	0.56	2.80
5	BQ26220	coulomb counter	16.01	80.05
5	ITG-3701	Gyroscope	9.10	45.50
5	ADXL372	MEMS Accelerometer	6.33	31.65
4	LSM9DS1	IMU ACCEL/GYRO/MAG	6.14	24.56
5	BM78SPPS5MC2-0002AA	BM78 Bluetooth Dual-Mode Module	9.59	47.95
		Total		\$398.05

Table 7.2.1: Original budget list [JD, NL]

The budget seen above is for the original design of the headband. The idea of the purchases is that the first round of part orders would be to test each subsystem. Enough parts were ordered to construct at least 2 of each subsystem. This was so that in case a part was damaged or lost, there would not be lost time waiting for another shipment. After the initial round of subsystems were

tested and integrated the PCB boards would be ordered. With the idea that PCB design often takes more than one iteration, 3 of each part going on the PCB would be ordered. These costs came in at 398.05\$. They do not include the cost for PCB boards, capacitors, resistors, switches, and other small components that would be added to the designed system. With most smaller components costing less than a dollar a part, and only requiring an estimated 3 iterations of PCB, 200 dollars of additional budget was more than enough to get the project done.

Final Budget Outcome:

Expense	Cost
Parts Request 1	\$ 221.65
Parts Request 2	\$ 28.44
Parts Request 3	\$ 9.04
Parts Request 4	\$ 145.84
Parts Request 5	\$ 24.15
Parts Request 6	\$ 102.00
Parts Request 7	\$ 142.86
PCB 1	\$ 20.30
PCB 2	\$ 18.30
Total:	\$ 712.58

Table 7.2.2 Budget breakdown per each order. [NL]

The total expenditures were slightly over the allotted budget of 600 dollars. The BM78 PICtail Plus was bought with the intention of being used by the department. If that expenditure is subtracted from the total, the total would be \$622.59.

Qty.	Part Num.	Description	Cost	Total Cost
1	LSM9DS1	IMU ACCEL/GYRO/MAG I2C/SPI 24LGA with Breakout board	\$15.95	\$15.95
1	BM78SPPS5MC2-0002AA	BM78 PICtail Plus	\$89.99	\$89.99
1	ADXL372	MEMS Accelerometer with board	55.00	55.00
1	PCM12SMTR	manual on/off switch	0.92	0.92
1	MCP73831	Battery management system	0.56	0.56
1	BQ26220	coulomb counter	16.01	16.01
3	ITG-3701	Gyroscope	9.10	27.30
1	ADXL372	MEMS Accelerometer	6.33	6.33
1	BM78SPPS5MC2-0002AA	BM78 Bluetooth Dual-Mode Module	9.59	9.59

Table 7.2.2: Parts request order 1 [JD]

The first order form is the initial testing phase for each individual section of the design. The intent of this order is to make sure that each component can perform its specific task, and to gain experience working with each individual module.

Qty.	Part Num.	Description	Cost	Total Cost
2	ITG-3701	Gyroscope	\$9.43	\$18.86
2	PA0061	QFN 16 to DIP 16 Adaptor	4.79	9.58

Table 7.2.3: Parts request order 2 [JD]

Qty.	Part Num.	Description	Vendor Part Num.	Cost	Total Cost
2	TPS3808G18DBVT	Low-Quiescent current supervisor	TPS3808	\$1.22	\$3.66
2	PA0085	SOT23	PA0085	2.69	5.38

Table 7.2.4: Parts request order 3 [JD]

The second and third order forms were placed quickly after the first order form. The order includes missing hardware and connection pieces used to test the devices on breadboards.

Qty.	Part Num.	Description	Cost	Total Cost
1	BM78SPPS5MC2-0002AA	BM78 Bluetooth Module	\$9.59	\$9.59
1	PCM12SMTR	ON/OFF switch	0.92	0.92
3	LTC1517ES5-3.3	voltage regulator	5.29	15.87
1	BQ26220PWR	Coloumb counter	8.11	8.11
5	TD4.7M16	4.7uF capacitor	0.69	3.45
4	FK20X5R1H335KN000	3.3uF capacitor	0.63	2.52
6	FA28X8R1E104KNU00	0.1 uF capacitor	0.36	2.16
10	294-470-RC	470 Ohm Resistor	0.15	1.50
10	294-10K-RC	10K Ohm Resistor	0.15	1.50
6	SB560TA	Diode	0.48	2.88
1	BOB-12700	USB power conection	4.50	4.50
1	ASR0008	Lithium Ion Polymer Battery	9.97	9.97
1	MCP73831T-2ACI/TI	Charge managment system	0.56	0.56
1	5031821852	MicroSD Reader	3.31	3.31
3	ADXL372	Accelerometer 200G	13.29	39.87
3	ITG-3701	Gyroscope 3 Axis	9.43	28.29
1	LSM9DS1	IMU ACCEL/GYRO/MAG	6.14	6.14
1	PIC24FJ1024GB610	PIC24 Microprocessor	4.7	4.70

Table 7.2.5: Parts request order 4 [JD]

This order was placed with the intent of using these pieces in a protoboard, and if the protoboard was successful, to be placed on a PCB. capacitors, resistors, diodes, and cord connections were ordered so that once each individual component was working, the process of connecting the subsystems could begin.

Qty.	Part Num.	Description	Cost	Total Cost
5	SB560TA	Diode	0.48	2.40
1	BQ26220PW	Coulomb counter	16.98	16.98
1	LSM6DSRTR	IMU	4.77	4.77
				24.15

Table 7.2.6: Parts request order 5 [JD]

A new IMU is ordered due to limited supplies from the IMU originally selected. Also, a new coulomb counter is ordered, as well as diodes that will be used to integrate the battery charging circuit and the other subsystems.

Qty.	Part Num.	Description	Cost	Total Cost
1	BM78SPPS5MC2-0002AA	BM78 Bluetooth Module	\$9.59	\$9.59
1	PCM12SMTR	ON/OFF switch	0.92	0.92
3	LTC1517ES5-3.3	voltage regulator	5.29	15.87
1	BQ26220PWR	Coloumb counter	8.11	8.11
10	294-470-RC	470 Ohm Resistor	0.15	1.50
10	294-10K-RC	10K Ohm Resistor	0.15	1.50
6	SB560TA	Diode	0.48	2.88
1	ASR0008	Lithium Ion Polymer Battery	9.97	9.97
1	5031821852	MicroSD Reader	3.31	3.31
1	ADXL372	Accelerometer 200G	13.29	13.29
1	ITG-3701	Gyroscope 3 Axis	9.43	9.43
1	LSM6DSRTR	IMU	4.77	4.77
1	PIC24FJ1024GB610	PIC24 Microprocessor	4.7	4.70
4	MCP73831T-2ACI/TI	Charge managment system	0.59	2.36
1		Micro USB adaptor	4.33	4.33
2	445-173321-1-ND	6.8uF capacitor	1.45	2.90
3	1825910-6	Reset Switch	0.10	0.30
4	1050170001	usb micro B port	0.83	3.32
1		microSD Reader breakout board	2.95	2.95

Table 7.2.7: Parts request order 6 [JD]

The order includes all of the components needed for placement on the PCB. Some are excluded, including the gyroscopes and accelerometers, because extras had not been used in previous test cycles.

8. Project Schedule

Fall Semester:

ID	Task Name	Duration	Start	Finish	Resource Names
1	SDP I 2020				
2	Project Design	91 days	Wed 8/26/20	Wed 11/25/20	
3	Midterm Report	40 days	Wed 8/26/20	Mon 10/5/20	
4	Cover page	40 days	Wed 8/26/20	Mon 10/5/20	
5	T of C, L of T, L of F	40 days	Wed 8/26/20	Mon 10/5/20	
6	Problem Statement	40 days	Wed 8/26/20	Mon 10/5/20	
7	Need	40 days	Wed 8/26/20	Sun 10/4/20	Jack Durkin, Noah Lewis, John Michel
8	Objective	40 days	Wed 8/26/20	Sun 10/4/20	Jack Durkin, Noah Lewis, John Michel
9	Background	40 days	Wed 8/26/20	Sun 10/4/20	Jack Durkin, Noah Lewis, John Michel
10	Marketing Requirements	40 days	Wed 8/26/20	Mon 10/5/20	Jack Durkin, Noah Lewis, John Michel
11	Engineering Requirements Specification	40 days	Wed 8/26/20	Mon 10/5/20	Jack Durkin, Noah Lewis, John Michel
12	Engineering Analysis	40 days	Wed 8/26/20	Mon 10/5/20	
13	Circuits	40 days	Wed 8/26/20	Mon 10/5/20	
14	Battery	40 days	Wed 8/26/20	Mon 10/5/20	Jack Durkin
15	Electronics	40 days	Wed 8/26/20	Mon 10/5/20	
16	Accelerometer	40 days	Wed 8/26/20	Mon 10/5/20	Jack Durkin
17	Gyroscope	40 days	Wed 8/26/20	Mon 10/5/20	Jack Durkin
18	Communications	40 days	Wed 8/26/20	Mon 10/5/20	
19	Intra-headband Communication	40 days	Wed 8/26/20	Mon 10/5/20	Noah Lewis, John Michel
20	Headband to Smartphone Communication	40 days	Wed 8/26/20	Mon 10/5/20	John Michel
21	Embedded Systems	40 days	Wed 8/26/20	Mon 10/5/20	
22	Bluetooth Module	40 days	Wed 8/26/20	Mon 10/5/20	John Michel
23	Microcontroller	40 days	Wed 8/26/20	Mon 10/5/20	Noah Lewis
24	Numerical Computations	40 days	Wed 8/26/20	Mon 10/5/20	
25	Linear Acceleration	40 days	Wed 8/26/20	Mon 10/5/20	John Michel
26	Angular Acceleration	40 days	Wed 8/26/20	Mon 10/5/20	John Michel
27	Head Injury Criterion	40 days	Wed 8/26/20	Mon 10/5/20	John Michel
28	Gadd Severity Index	40 days	Wed 8/26/20	Mon 10/5/20	John Michel
29	Principal Component Score	40 days	Wed 8/26/20	Mon 10/5/20	John Michel
30	Step Detection	40 days	Wed 8/26/20	Mon 10/5/20	Noah Lewis
31	Accepted Technical Design	40 days	Wed 8/26/20	Mon 10/5/20	
32	Hardware Design: Phase 1	40 days	Wed 8/26/20	Mon 10/5/20	
33	Hardware Level 0 Block Diagram	40 days	Wed 8/26/20	Mon 10/5/20	Jack Durkin, Noah Lewis, John Michel
34	Hardware Level 1 Block Diagram	40 days	Wed 8/26/20	Mon 10/5/20	Jack Durkin, Noah Lewis, John Michel
35	Software Design: Phase 1	40 days	Wed 8/26/20	Mon 10/5/20	
36	Software Level 0 Block Diagram	40 days	Wed 8/26/20	Mon 10/5/20	Noah Lewis, John Michel
37	Team information	40 days	Wed 8/26/20	Mon 10/5/20	
38	Project Schedule	40 days	Wed 8/26/20	Mon 10/5/20	
39	Midterm Design Gantt Chart	40 days	Wed 8/26/20	Mon 10/5/20	John Michel
40	References	40 days	Wed 8/26/20	Mon 10/5/20	
41	Hardware Design: Phase 2	20 days	Mon 10/5/20	Wed 11/18/20	Jack Durkin, Noah Lewis, John Michel
42	Software Design: Phase 2	20 days	Mon 10/5/20	Wed 11/18/20	Jack Durkin, Noah Lewis, John Michel
43	Subsystems Demonstration	20 days	Mon 10/5/20	Wed 11/18/20	Jack Durkin, Noah Lewis, John Michel
44	Final Report	20 days	Mon 10/5/20	Wed 11/25/20	Jack Durkin, Noah Lewis, John Michel

Table 8.1: Fall Semester Original Gantt Chart [NL]

Spring Semester Version 1:

		Task Name	Duration	Start	Finish	Resource Names
1		SDP2 Implementation 2020	103 days	Mon 1/11/21	Fri 4/23/21	
2		Implement Project Design	89 days	Mon 1/11/21	Fri 4/9/21	
3		Hardware Implementation	47 days	Mon 1/11/21	Sat 2/27/21	
4		Breadboard Components	21 days	Mon 1/11/21	Sun 1/31/21	
5		Voltage Regulator Circuit	21 days	Mon 1/11/21	Sun 1/31/21	Jack Durkin
6		Battery Charging Circuit	21 days	Mon 1/11/21	Sun 1/31/21	Jack Durkin
7		Coulomb Counting Circuit	21 days	Mon 1/11/21	Sun 1/31/21	Jack Durkin
8		Layout and Generate PCB(s)	21 days	Mon 1/11/21	Sun 1/31/21	
9		Microcontroller Board	21 days	Mon 1/11/21	Sun 1/31/21	Noah Lewis,John Michel
10		Assemble Hardware	14 days	Mon 1/25/21	Sun 2/7/21	
11		Microcontroller Board	14 days	Mon 1/25/21	Sun 2/7/21	Noah Lewis,John Michel,Jack D
12		Accel/Gyro Board	14 days	Mon 1/25/21	Sun 2/7/21	Noah Lewis
13		<subsystem n>	14 days	Mon 1/25/21	Sun 2/7/21	Student 4
14		Revise Hardware	7 days	Mon 2/8/21	Sun 2/14/21	13
15		Microcontroller Board	7 days	Mon 2/8/21	Sun 2/14/21	13 Noah Lewis,John Michel,Jack D
16		Accel/Gyro Board	7 days	Mon 2/8/21	Sun 2/14/21	13 Noah Lewis
17		<subsystem n>	7 days	Mon 2/8/21	Sun 2/14/21	13 Student 4
18		Revise Hardware	7 days	Mon 2/15/21	Sun 2/21/21	17
19		Microcontroller Board	7 days	Mon 2/15/21	Sun 2/21/21	17 Noah Lewis,John Michel,Jack D
20		Accel/Gyro Board	7 days	Mon 2/15/21	Sun 2/21/21	17 Noah Lewis
21		<subsystem n>	7 days	Mon 2/15/21	Sun 2/21/21	17 Student 4
22		MIDTERM: Demonstrate Hardware Subsystems	5 days	Mon 2/22/21	Fri 2/26/21	21
23		SDC & FA Hardware Approval	0 days	Sat 2/27/21	Sat 2/27/21	22
24		Software Implementation	47 days	Mon 1/11/21	Sat 2/27/21	
25		Develop Software	28 days	Mon 1/11/21	Sun 2/7/21	
26		Mobile App	28 days	Mon 1/11/21	Sun 2/7/21	John Michel,Noah Lewis
27		Receive sensor data and store in csv files	28 days	Mon 1/11/21	Sun 2/7/21	Noah Lewis,John Michel
28		Send csv files to microSD	28 days	Mon 1/11/21	Sun 2/7/21	Noah Lewis
29		Retrieve csv files from microSD	28 days	Mon 1/11/21	Sun 2/7/21	John Michel
30		Send csv files through bluetooth to mobile app	28 days	Mon 1/11/21	Sun 2/7/21	John Michel
31		Test Software	28 days	Mon 1/11/21	Sun 2/7/21	
32		Mobile App	28 days	Mon 1/11/21	Sun 2/7/21	John Michel,Noah Lewis
33		Receive sensor data and store in csv files	28 days	Mon 1/11/21	Sun 2/7/21	Noah Lewis,John Michel
34		Send csv files to microSD	28 days	Mon 1/11/21	Sun 2/7/21	Noah Lewis
35		Retrieve csv files from microSD	28 days	Mon 1/11/21	Sun 2/7/21	John Michel
36		Send csv files through bluetooth to mobile app	28 days	Mon 1/11/21	Sun 2/7/21	John Michel
37		Revise Software	14 days	Mon 2/8/21	Sun 2/21/21	25
38		Mobile App	14 days	Mon 2/8/21	Sun 2/21/21	25 John Michel,Noah Lewis
39		Receive sensor data and store in csv files	14 days	Mon 2/8/21	Sun 2/21/21	25 Noah Lewis,John Michel
40		Send csv files to microSD	14 days	Mon 2/8/21	Sun 2/21/21	25 Noah Lewis
41		Retrieve csv files from microSD	14 days	Mon 2/8/21	Sun 2/21/21	25 John Michel
42		Send csv files through bluetooth to mobile app	14 days	Mon 2/8/21	Sun 2/21/21	25 John Michel
43		MIDTERM: Demonstrate Software Subsystems	5 days	Mon 2/22/21	Fri 2/26/21	42,
44		SDC & FA Software Approval	0 days	Sat 2/27/21	Sat 2/27/21	43

45		▾ System Integration	42 days	Sat 2/27/21	Fri 4/9/21	
46		▾ Assemble Complete System Integration	14 days	Sat 2/27/21	Fri 3/12/21	43
47	🚫	Microcontroller	14 days	Sat 2/27/21	Fri 3/12/21	43 Jack Durkin,John Michel,Noah L
48	🚫	3D Printed Head mount	14 days	Sat 2/27/21	Fri 3/12/21	43 Noah Lewis
49	🚫	Accel/Gyro Boards	14 days	Sat 2/27/21	Fri 3/12/21	43 John Michel,Noah Lewis
50		▾ Test Complete System Integration	7 days	Sat 3/13/21	Fri 3/19/21	46
51	🚫	Microcontroller	7 days	Sat 3/13/21	Fri 3/19/21	46 Jack Durkin,John Michel,Noah L
52	🚫	3D Printed Head mount	7 days	Sat 3/13/21	Fri 3/19/21	46 Noah Lewis
53	🚫	Accel/Gyro Boards	7 days	Sat 3/13/21	Fri 3/19/21	46 John Michel,Noah Lewis
54		▾ Revise Complete System Integration	16 days	Sat 3/20/21	Sun 4/4/21	50
55	🚫	Microcontroller	16 days	Sat 3/20/21	Sun 4/4/21	50 Jack Durkin,John Michel,Noah L
56	🚫	3D Printed Head mount	16 days	Sat 3/20/21	Sun 4/4/21	50 Noah Lewis
57	🚫	Accel/Gyro Boards	16 days	Sat 3/20/21	Sun 4/4/21	50 John Michel,Noah Lewis
58		Demonstration of Complete System	5 days	Mon 4/5/21	Fri 4/9/21	54
59	🚀	▾ Develop Final Report	103 days	Mon 1/11/21	Fri 4/23/21	
60		Write Final Report	103 days	Mon 1/11/21	Fri 4/23/21	
61		Submit Final Report	0 days	Fri 4/23/21	Fri 4/23/21	60
62	🌿	Spring Recess	7 days	Mon 4/12/21	Sun 4/18/21	
63	🚀	Project Demonstration and Presentation				

Table 8.2: Spring Semester Original Gantt Chart [NL]

Spring Semester Version 2:

	Task Mode	Task Name	Duration	Start	Finish	Resource Names
1		▾ SDP2 Implementation 2020	103 days?	Mon 1/11/21	Fri 4/23/21	
2		▾ Implement Project Design	89 days?	Mon 1/11/21	Fri 4/9/21	
3		▾ Hardware Implementation	77 days?	Mon 1/11/21	Sun 3/28/21	
4	🚀	▾ Breadboard Components	21 days	Mon 1/11/21	Sun 1/31/21	
5	🚫	Voltage Regulator Circuit	21 days	Mon 1/11/21	Sun 1/31/21	Jack Durkin
6	🚫	Battery Charging Circuit	21 days	Mon 1/11/21	Sun 1/31/21	Jack Durkin
7	🚀	▾ Layout and Generate PCB(s)	21 days	Mon 1/11/21	Sun 1/31/21	
8	🚫	Microcontroller Board Version 1	21 days	Mon 1/11/21	Sun 1/31/21	Noah Lewis
9	🚀	▾ Assemble Hardware	14 days?	Mon 1/25/21	Sun 2/7/21	
10	🚫	Microcontroller Board Version 1	14 days	Mon 1/25/21	Sun 2/7/21	Noah Lewis,Jack Durkin
11	🚫	Protoboard (Accel/Gyro/MicroSD/IMU/Bluetooth)	14 days	Mon 1/25/21	Sun 2/7/21	Noah Lewis
12	🚀	▾ Test Hardware	35 days	Mon 2/8/21	Sun 3/14/21	
13	🚫	Microcontroller Board Version 1	7 days	Mon 2/8/21	Sun 2/14/21	Noah Lewis,Jack Durkin
14	🚫	Protoboard (Accel/Gyro/MicroSD/IMU/Bluetooth)	7 days	Mon 2/8/21	Sun 2/14/21	Noah Lewis
15	🚀	▾ Revise Hardware	42 days	Mon 2/15/21	Sun 3/28/21	
16	🚫	Microcontroller Board Version 1	7 days	Mon 2/15/21	Sun 2/21/21	Noah Lewis
17	🚫	Protoboard (Accel/Gyro/MicroSD/IMU/Bluetooth)	7 days	Mon 2/15/21	Sun 2/21/21	Noah Lewis
18	🚫	MIDTERM: Demonstrate Hardware Subsystems	5 days	Mon 2/22/21	Fri 2/26/21	
19	🚀	SDC & FA Hardware Approval	0 days	Sat 2/27/21	Sat 2/27/21	18

20		Software Implementation	47 days	Mon 1/11/21	Sat 2/27/21	
21		Develop Software	28 days	Mon 1/11/21	Sun 2/7/21	
22	🚫	Receive sensor data and store in csv files	28 days	Mon 1/11/21	Sun 2/7/21	Noah Lewis,John Michel
23	🚫	Send data to txt files on microSD	28 days	Mon 1/11/21	Sun 2/7/21	Noah Lewis
24	🚫	Send txt files through bluetooth to mobile app	28 days	Mon 1/11/21	Sun 2/7/21	John Michel
25		Test Software	28 days	Mon 1/11/21	Sun 2/7/21	
26	🚫	Receive sensor data and store in csv files	28 days	Mon 1/11/21	Sun 2/7/21	Noah Lewis,John Michel
27	🚫	Send data to txt files on microSD	28 days	Mon 1/11/21	Sun 2/7/21	Noah Lewis,John Michel
28	🚫	Send txt files through bluetooth to mobile app	28 days	Mon 1/11/21	Sun 2/7/21	John Michel
29		Revise Software	14 days	Mon 2/8/21	Sun 2/21/21	21
30	🚫	Receive sensor data and store in csv files	14 days	Mon 2/8/21	Sun 2/21/21	21 Noah Lewis,John Michel
31	🚫	Send data to txt files on microSD	14 days	Mon 2/8/21	Sun 2/21/21	21 Noah Lewis,John Michel
32	🚫	Send txt files through bluetooth to mobile app	14 days	Mon 2/8/21	Sun 2/21/21	21 John Michel
33		MIDTERM: Demonstrate Software Subsystems	5 days	Mon 2/22/21	Fri 2/26/21	32
34	🚀	SDC & FA Software Approval	0 days	Sat 2/27/21	Sat 2/27/21	33
35		System Integration	42 days	Sat 2/27/21	Fri 4/9/21	
36		Assemble Complete System Integration	14 days	Sat 2/27/21	Fri 3/12/21	33
37	🚫	Microcontroller Version 2	14 days	Sat 2/27/21	Fri 3/12/21	33 Noah Lewis
38	🚫	Accel/Gyro Sensors	14 days	Sat 2/27/21	Fri 3/12/21	33 John Michel,Noah Lewis
39		Test Complete System Integration	7 days	Sat 3/13/21	Fri 3/19/21	36
40	🚫	Microcontroller Version 3	7 days	Sat 3/13/21	Fri 3/19/21	36 Jack Durkin,John Michel,Noah L
41	🚫	Accel/Gyro Sensors	7 days	Sat 3/13/21	Fri 3/19/21	36 John Michel,Noah Lewis
42		Revise Complete System Integration	16 days	Sat 3/20/21	Sun 4/4/21	39
43	🚫	Final Protoboard	16 days	Sat 3/20/21	Sun 4/4/21	39 Jack Durkin,John Michel,Noah L
44	🚫	3D Printed Head mount	16 days	Sat 3/20/21	Sun 4/4/21	39 Noah Lewis
45	🚫	Accel/Gyro Boards	16 days	Sat 3/20/21	Sun 4/4/21	39 John Michel,Noah Lewis
46		Demonstration of Complete System	5 days	Mon 4/5/21	Fri 4/9/21	42
47	🚀	Develop Final Report	103 days	Mon 1/11/21	Fri 4/23/21	
48	🚫	Write Final Report	103 days	Mon 1/11/21	Fri 4/23/21	Jack Durkin,John Michel,Noah L
49		Submit Final Report	0 days	Fri 4/23/21	Fri 4/23/21	48
50	🚀	Spring Recess	7 days	Mon 4/12/21	Sun 4/18/21	
51	🚀	Project Demonstration and Presentation				

Table 8.3: Spring Semester Final Gantt Chart [NL]

9. Design Team Information

Member Name	Major	Project Role
Jack Durkin	Electrical Engineering	Hardware Lead
Noah Lewis	Computer Engineering	Project Lead
John Michel	Computer Engineering	Archivist/Software Lead

10. Conclusions and Recommendations

This project sought to create an easy-to-use and relatively accurate impact and concussion detection headband. This headband provides a moderate solution to the trade-off between usability and accuracy. While other concussion detection solutions such as helmets may provide more accuracy, they are not nearly as usable. Similarly, simpler wearables such as patches lack the accuracy needed for usable results. [JM]

The final implementation of this project showed a working prototype containing the primary required features for the headband device. The prototype can automatically detect impacts and record linear acceleration and angular rate data associated with the impact. The data recorded is automatically stored to a data file in an on-board microSD. The microSD can be removed from the prototype and the data can be extracted via a microSD to USB converter. The data obtained from the microSD can be analyzed by a Matlab script to analyze the impacts recorded and determine an approximate concussion risk. [JM]

The working prototype developed in this project still requires debugging or development of multiple features, including step detection, a database structure, the headband device battery system, the communication of the impact data file over Bluetooth communication, and the integration of the Matlab script into an existing mobile application. [JM]

10.1 Future Implementation

10.1.1 Step Detection

These would be calculations based on the linear acceleration for upright movement detection for step counting. To achieve this, the Signal Magnitude Area, or SMA, is calculated using the following equation:

$$SMA = \frac{1}{t} (\int_0^t |x(t)| dt + \int_0^t |y(t)| dt + \int_0^t |z(t)| dt) \quad (6)$$

This equation categorizes the data into two sections: walking and jogging. Walking is in the range between 0.135g's and 0.8g's, while jogging is above 0.8g's. [NL]

The calculations can be improved further by using adaptive timing threshold to eliminate false positives. The following equation measures the timing threshold:

$$t_l = f_s * \frac{0.1}{mean(SMA)} \quad (7)$$

The adaptive timing threshold calculation uses the sampling frequency of the accelerometer that is used in the calculation, as well as the mean of the SMA calculated in equation (6) above, to determine the timestamp between each heel-strike to the ground. If the timestamp is not big enough, or is too small for a regular step, then the system determines the accuracy of the step that was detected. This can be used to decrease the error range of the calculations. [NL]

10.1.2 SQLite Database Structure

ConcAccelData	
sensor#	INT
accelX	REAL
accelY	REAL
accelZ	REAL
timeStamp	TEXT
linAccel	REAL

ConcGyroData	
sensor#	INT
gyroPitch	REAL
gyroRoll	REAL
gyroYaw	REAL
timeStamp	TEXT
angAccel	REAL

IMUData	
accelX	REAL
accelY	REAL
accelZ	REAL
gyroPitch	REAL
gyroRoll	REAL
gyroYaw	REAL
timeStamp	TEXT

Figure 10.1.2.1: SQLite database schema for the mobile application. [NL]

Above would be the SQLite database schema for the mobile application. The three relations we will include will be the ConcAccelData (for all of the data from the three accelerometer sensors), ConcGyroData (for all of the data from the three gyroscope sensors), and the IMUData table. The ConcAccelData table will include sensor#, which is the number corresponding to the accelerometer/gyroscope pairing, the X, Y, and Z accelerations, the timestamp and the linAccel, which is the linear acceleration calculated using (3) from above. The ConcGyroData is similar to the first table, except instead of coordinates, the table records the pitch, roll and yaw, along with the angAccel, which is the angular acceleration calculated using (4) from above. The final table, IMUData, records the IMU's accelerometer data (accelX, accelY, and accelZ) and the gyroscope data (gyroPitch, gyroRoll, and gyroYaw), then attaches a timestamp to the data. [NL]

On the next page is the code to create the tables in SQLite:

```
1 CREATE TABLE ConcAccelData(  
2   sensor# INT,  
3   accelX REAL,  
4   accelY REAL,  
5   accelZ REAL,  
6   timeStamp TEXT PRIMARY KEY,  
7   linAccel REAL  
8 );  
9  
10 CREATE TABLE ConcGyroData(  
11   sensor# INT,  
12   gyroPitch REAL,  
13   gyroRoll REAL,  
14   gyroYaw REAL,  
15   timeStamp TEXT PRIMARY KEY,  
16   angAccel REAL  
17 );  
18  
19 CREATE TABLE IMUData(  
20   accelX REAL,  
21   accelY REAL,  
22   accelZ REAL,  
23   gyroPitch REAL,  
24   gyroRoll REAL,  
25   gyroYaw REAL,  
26   timeStamp TEXT PRIMARY KEY  
27 );
```

Figure 10.1.2.2: Creation of the SQLite tables in the mobile application. [NL]

10.1.3. Battery System

The battery system was originally designed to have a 3.7 V battery source. To charge a battery there needs to be a greater voltage across the battery than its charge. Due to this to use the charging circuit the batteries had to be separated from one another and charged separately. This meant that the charging circuit could not be circuited directly onto either protoboard. For future improvements, a new voltage regulator would be selected that could handle a large current draw. An LD1117V3.3 would work, with a maximum current draw of 800mA and output of 3.3V. With this voltage regulator a single battery pack would suffice. This would be advantageous with regards to size as well. With this regulator the original battery charging circuit could be implemented, as well as the coulomb counter. [JD]

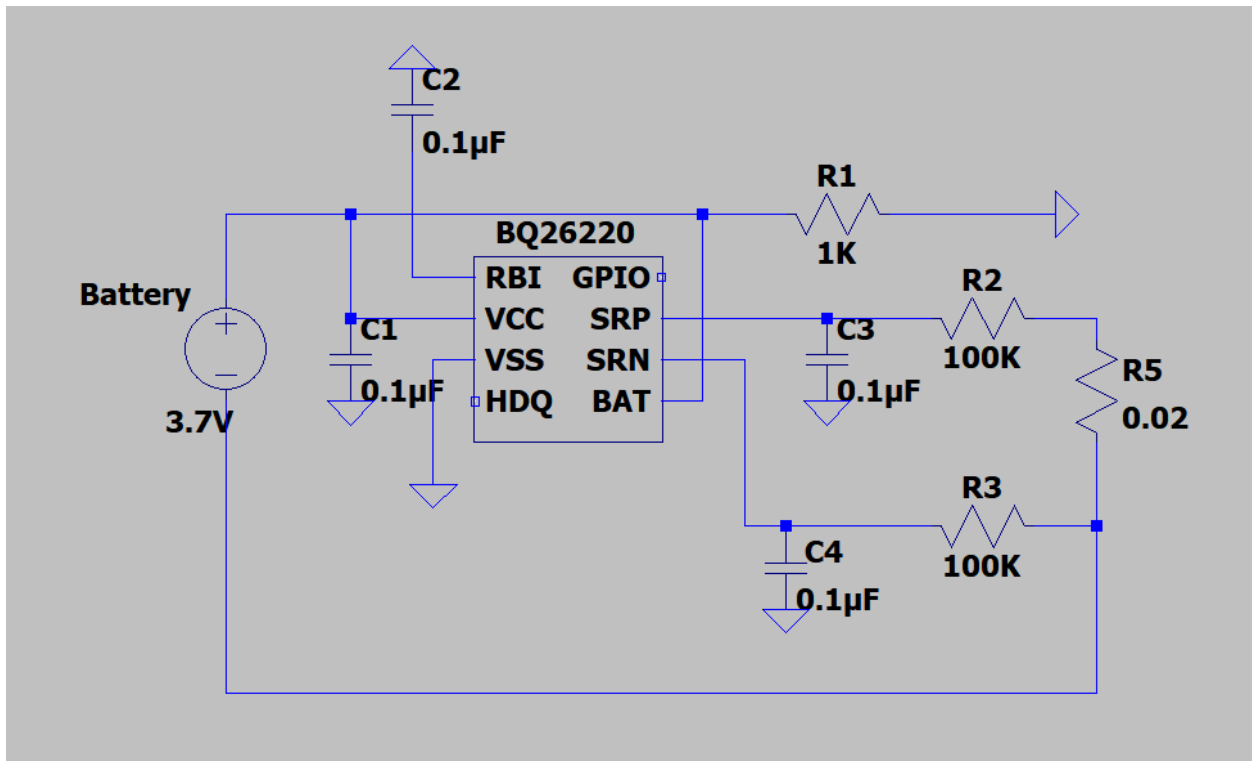


Figure 10.1.3: Coulomb Counting Circuit [JD]

10.1.4. Impact Data File Read and Bluetooth Communication Integration

The final implementation of this project intended to read the data file from the microSD and retransmit that data over the Bluetooth connection for the mobile application to receive. This part of the project was nearly complete; dummy strings could be sent over the connection, and at the correct time. The only issue occurred when reading the data file, where an unknown error caused the Bluetooth connection to crash when attempting to transmit the read data. Any other data sources, such as a dummy string, could be sent over the connection. Future work should correctly implement the data file read so that its contents can be received by the mobile application at the correct time. The incomplete implementation is shown because the section of code in main.c lines 258 through 266 is the only section that is broken within the data transmission. See the appendices for the full code in main.c. [JM]

10.1.5. Mobile Application and Matlab Script Integration

This project intended to integrate the computations detailed in the Matlab script with the mobile application. MathWorks, the developer of Matlab, provides a Matlab Compiler SDK which compiles Matlab code into other languages such as C++ and Java. Future work should export the Matlab script into Java code and insert it into the mobile application, which is based on the source code for the Microchip Bluetooth Data application. Future work should also modify the mobile application to fit this project; the application currently contains many features and application views that are not necessary for this project. [JM]

10.2. Team Dynamics

This design process has spanned over a year and a half. The team has had to be very flexible to finish the project. COVID-19 halted in person meetings at the beginning of the implementation phase of this project. Communication over platforms like Microsoft Teams, Outlook, and text were essential. Multiple team members were lost during the project; two previous group members decided to pursue other opportunities outside of the university. This meant that our remaining members had to pick up new tasks and adjust quickly. Specifically, not having a second electrical engineer created timeline issues in completing the PCB. Communication with advisors and professors was essential to the team's success in this project. Although it was not ideal that the in-person meetings were extremely limited, the team was still able to work together. The team members were able to work separately to complete individual tasks and also work together to integrate each other's work. [JD]

11. References

- Benzel, Edward C, et al. Classification of Impacts from Sensor Data. 22 Mar. 2016.
- “Bluetooth Low Energy.” *Microchip*, Microchip Technology, Inc., Nov. 2020,
www.microchip.com/en-us/products/wireless-connectivity/bluetooth-low-energy.
- E Fortune, et al. “Step Detection Using Multi- versus Single Tri-Axial Accelerometer-Based Systems.” *Physiological Measurement*, vol. 36, no. 12, Dec. 2015, p. 1. EBSCOhost, doi:10.1088/0967-3334/36/12/2519.
- Gao, Dalong, and Charles W Wampler. “Head Injury Criterion.” *IEEE Robotics & Automation Magazine*, Dec. 2009, pp. 71–74.
- Greenwald, Richard M et al. “Head impact severity measures for evaluating mild traumatic brain injury risk exposure.” *Neurosurgery* vol. 62,4 (2008): 789-98; discussion 798.
doi:10.1227/01.neu.0000318162.67472.ad
- Hanly, Steve. “Accelerometers: Taking the Guesswork out of Accelerometer Selection.” EnDAQ, 2016, blog.endaq.com/accelerometer-selection.
- Hasawish, Herman. “React Native Chart Kit.” Npm, Inc., 14 Nov. 2020.
- “Interfacing Microcontrollers with SD Card.” OpenLabPro.com, Etiq Technologies, 2019,
openlabpro.com/guide/interfacing-microcontrollers-with-sd-card/.
- Jackson, J. Edward. *A User's Guide to Principal Components*. John Wiley & Sons, 1991.
- King, Albert I., et al. “International Research Council on Biomechanics of Injury Conference.” National Football League & Wayne State University, *Is Head Injury Caused by Linear or Angular Acceleration?*, 2003, pp. 1–3.
- Kleiven, Svein. “Why Most Traumatic Brain Injuries are Not Caused by Linear Acceleration but Skull Fractures are.” *Frontiers in bioengineering and biotechnology* vol. 1 15. 7 Nov.

2013, doi:10.3389/fbioe.2013.00015

Lam, Lawrence K, and Austen J Szypula. "Wearable Emotion Sensor On Flexible Substrate For Mobile Health Applications." 2018.

Lin, Shih-Kai, et al. "An Ultra-Low Power Smart Headband for Real-Time Epileptic Seizure Detection." 2018.

"Lithium-Ion Battery." *Clean Energy Institute*, 25 Sept. 2020,

www.cei.washington.edu/education/science-of-solar/battery-technology/.

Mack, Christoph, et al. Communication System for Impact Sensors. 31 Jan. 2017.

O'Connor, Kathryn L et al. "Head-Impact-Measurement Devices: A Systematic Review."

Journal of athletic training vol. 52,3 (2017): 206-227. doi:10.4085/1062-6050.52.2.05

Storm, Fabio Alexander. "Stop Detection and Activity Recognition Accuracy of Seven Physical Activity Monitors." Researchgate, 19 Mar. 2015,

www.researchgate.net/publication/273781177_Step_Detection_and_Activity_Recognition_Accuracy_of_Seven_Physical_Activity_Monitors.

Watson, Jeff. "MEMS Gyroscope Provides Precision Inertial Sensing in Harsh, High

Temperature Environments." *MEMS Gyroscope Provides Precision Inertial Sensing in Harsh, High Temperature Environments | Analog Devices*, 2017

Zhao, Niel. "Full-Featured Pedometer Design Realized with 3-Axis Digital Accelerometer."

Full-Featured Pedometer Design Realized with 3-Axis Digital Accelerometer | Analog Devices, 2017

12. Appendices

12.1. MPLAB X Configurations and Code

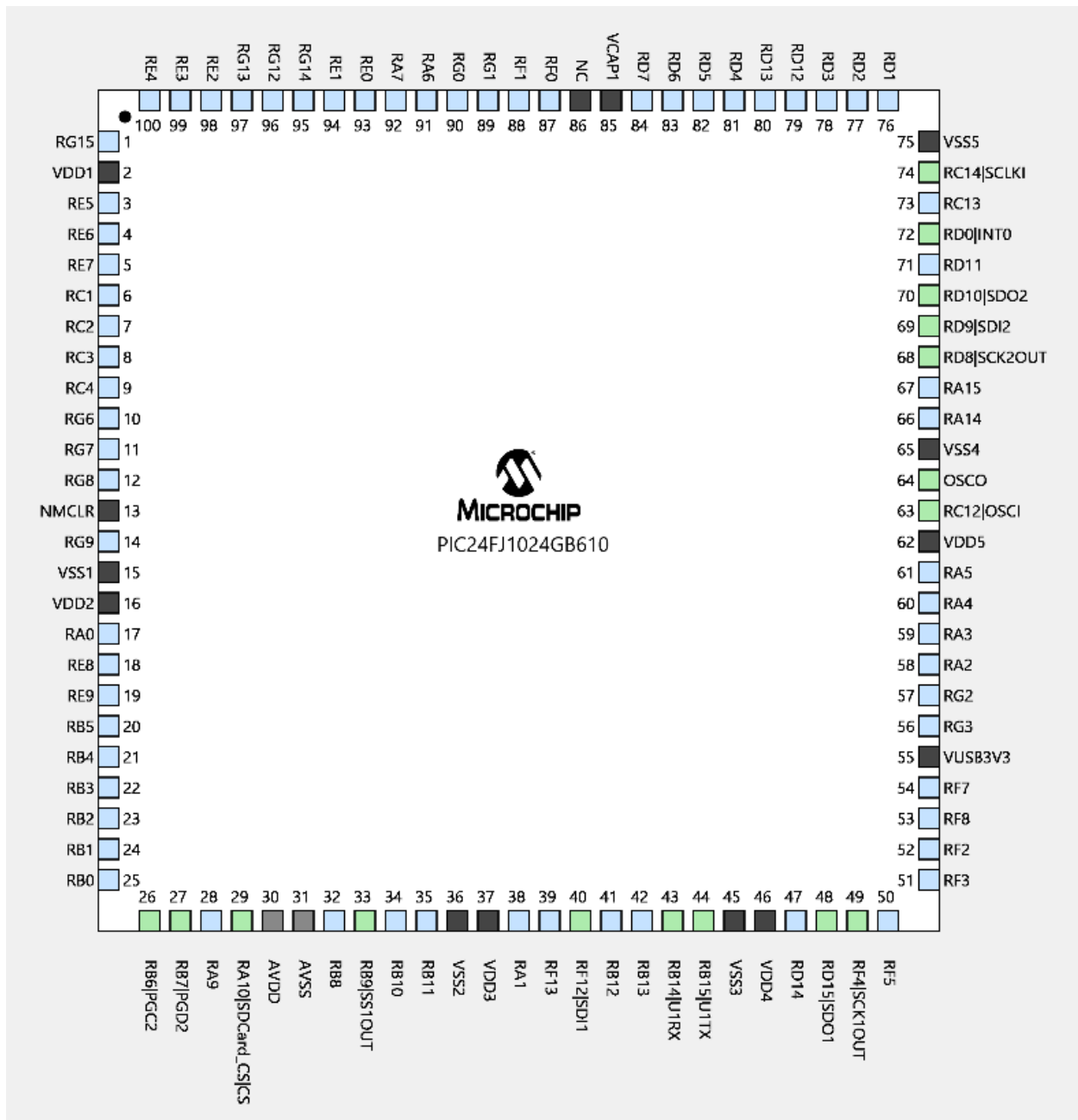


Figure 12.1.1: Package view of the PIC24FJ1024GB610 pin assignments used. [NL, JM]

System Module

The screenshot displays the 'System Module' configuration window for the PIC24FJ1024GB610 microcontroller. At the top, there are two tabs: 'Easy Setup' (selected) and 'Registers'. Below the tabs is a 'Clock' section with a dropdown arrow. The main configuration area includes:

- A frequency input field set to '8000000' Hz, with a dropdown menu set to 'Primary Oscillator' and a range '(3.5 MHz - 32 MHz) Clock Source'.
- A 'PLL Enable' checkbox, currently unchecked.
- Three input fields for 'Fosc' (8 MHz), 'Fosc/2' (4 MHz), and 'USB Clock' (empty).
- An 'Oscillator Fractional Divisor' checkbox, currently unchecked.
- A 'Clock Output Pin Configuration' dropdown menu set to 'OSC2 is general purpose digital I/O pin'.
- A 'Use Secondary Oscillator' checkbox, currently unchecked, with an input field for '(31 - 33) kHz'.
- A 'Reference Oscillator Output' checkbox, currently unchecked.
- 'Enable Clock Switching' and 'Enable Fail-Safe Monitor' checkboxes, both currently unchecked.
- An 'Auto Tuning Mode' dropdown menu set to 'Disabled'.

Figure 12.1.2: MCC System Module for the PIC24FJ1024GB610 microcontroller. This module was primarily used to set the clock rate and general features of the microcontroller. [JM]

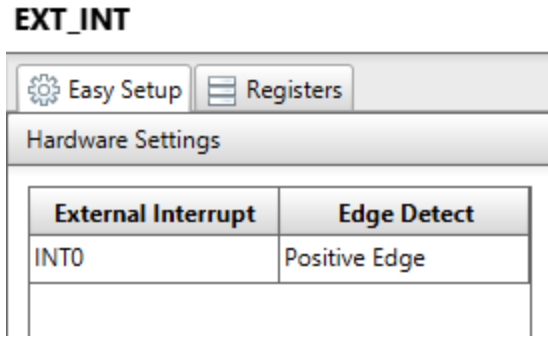


Figure 12.1.3: MCC External Interrupt module. This module sets up the code required to run an interrupt service routine for an external interrupt tied to INT0. [JM]


```

52     uint8_t writeData[BUFFER_SIZE];
53     uint8_t readData[BUFFER_SIZE];
54     int i = 0;
55
56     clearBuffer(writeData, BUFFER_SIZE);
57     clearBuffer(readData, BUFFER_SIZE);
58
59     // wait one second
60     for(i = 0; i < 10; ++i) {
61         ms_delay(100);
62     }
63
64     XL372_CS = 0; // CS to active low
65
66     // set FIFO to Disabled
67     writeData[0] = XL372_FIFO_CTL;
68     writeData[0] = writeData[0] << 1;
69     writeData[1] = 0x00 + 1; //turn FIFO to disabled
70     // plus one is for the FIFO Samples which is 0x1A4, uses bit 0 here.
71     SPI1_Exchange8bitBuffer(writeData, 2, readData);
72
73     XL372_CS = 1; // CS to inactive high
74
75     us_delay(40);
76
77     XL372_CS = 0; // CS to active low
78
79     // set INT1 to FIFO_FULL
80     writeData[0] = XL372_INT1_MAP;
81     writeData[0] = writeData[0] << 1;
82     writeData[1] = 0x04; // map FIFO_FULL interrupt onto INT1
83     SPI1_Exchange8bitBuffer(writeData, 2, readData);
84
85     XL372_CS = 1; // CS to inactive high
86
87     us_delay(40);
88
89     XL372_CS = 0; // CS to active low
90
91     // set FIFO samples to 169*3 = 507 (this gives 169 XYZ samples AFTER the trigger point)
92     // 507 = 0x1FB
93     writeData[0] = XL372_FIFO_SAMPLES;
94     writeData[0] = writeData[0] << 1;
95     writeData[1] = 0xFB; // bottom 8 bits
96     SPI1_Exchange8bitBuffer(writeData, 2, readData);
97
98     XL372_CS = 1; // CS to inactive high
99
100    us_delay(40);
101
102    XL372_CS = 0; // CS to active low

```



```

103
104 // set ODR to 3200 Hz
105 writeData[0] = XL372_TIMING;
106 writeData[0] = writeData[0] << 1;
107 writeData[1] = 0b01100000; // set ODR
108 SPI1_Exchange8bitBuffer(writeData, 2, readData);
109 XL372_CS = 1; // CS to inactive high
110
111 us_delay(40);
112
113 XL372_CS = 0; // CS to active low
114
115 // set to instant on mode, low threshold (do this last)
116 writeData[0] = XL372_POWER_CTL;
117 writeData[0] = writeData[0] << 1;
118 writeData[1] = 0x02;
119 SPI1_Exchange8bitBuffer(writeData, 2, readData);
120 XL372_CS = 1; // CS to inactive high
121
122 ms_delay(20); // delay for >16 ms for change to instant on mode
123
124 XL372_CS = 0; // CS to active low
125
126 // set FIFO to oldest saved mode
127 writeData[0] = XL372_FIFO_CTL;
128 writeData[0] = writeData[0] << 1;
129 writeData[1] = 0x06 + 1; //turn FIFO to oldest saved mode
130 // plus one is for the FIFO Samples which is 0x1A4, uses bit 0 here.
131 SPI1_Exchange8bitBuffer(writeData, 2, readData);
132
133 XL372_CS = 1; // CS to inactive high
134
135 ITG3701_CS = 0; // CS set to active
136
137 // write to int_enable
138 writeData[0] = ITG3701_INT_ENABLE;
139 writeData[1] = 0x10; //enable FIFO overflow interrupt
140 SPI1_Exchange8bitBuffer(writeData, 2, readData);
141
142 ITG3701_CS = 1;
143 us_delay(40);
144
145 ITG3701_CS = 0; // CS set to active
146
147 // Configure CONFIG
148 writeData[0] = ITG3701_CONFIG;
149 writeData[1] = 0x40; // FIFO mode to 1 (oldest saved mode)
150 SPI1_Exchange8bitBuffer(writeData, 2, readData);
151
152 ITG3701_CS = 1;
153 us_delay(40);

```

```

154     ITG3701_CS = 0; // CS set to active
155
156     // Configure GYRO_CONFIG
157     writeData[0] = ITG3701_GYRO_CONFIG;
158     writeData[1] = 0x18; // Sets full scale range (FS_SEL) to +- 4000 degrees/s
159     SPI1_Exchange8bitBuffer(writeData, 2, readData);
160
161     ITG3701_CS = 1;
162     us_delay(40);
163     ITG3701_CS = 0;
164
165     // Configure FIFO_EN
166     writeData[0] = ITG3701_FIFO_EN;
167     writeData[1] = 0x70; // enables x, y and z axes to be written to the FIFO buffer
168     SPI1_Exchange8bitBuffer(writeData, 2, readData);
169
170     ITG3701_CS = 1;
171     us_delay(40);
172     ITG3701_CS = 0;
173
174     // Configure USER_CTRL
175     writeData[0] = ITG3701_USER_CTRL;
176     writeData[1] = 0x50; // enables FIFO, disables I2C, enables SPI
177     SPI1_Exchange8bitBuffer(writeData, 2, readData);
178
179     ITG3701_CS = 1;
180     us_delay(40);
181     ITG3701_CS = 0;
182
183     // Configure PWR_MGMT_1
184     writeData[0] = ITG3701_PWR_MGMT_1;
185     writeData[1] = 0x08; // disables temp sensor
186     SPI1_Exchange8bitBuffer(writeData, 2, readData);
187
188     ITG3701_CS = 1;
189     us_delay(40);
190     ITG3701_CS = 0;
191
192     // Configure PWR_MGMT_2
193     writeData[0] = ITG3701_PWR_MGMT_2;
194     writeData[1] = 0x00; // Standby mode off
195     SPI1_Exchange8bitBuffer(writeData, 2, readData);
196
197     ITG3701_CS = 1;
198
199     // set up Bluetooth
200     // set reset to 1 since it is active low
201     TRIS_RST_N = 0;
202     ms_delay(1);
203     PORT_RST_N = 1; // active low
204

```

```

205 // set up pins for Bluetooth test (OFF/OFF/ON) -> 1/1/1 (ROM App) (switches are !P2_0 and !P2_4 and EAN)
206 TRIS_P2_0 = 0; // set P2_0 to output
207 TRIS_P2_4 = 0; // set P2_4 to output
208 TRIS_EAN = 0; // set EAN to output
209 ms_delay(1);
210 PORT_P2_0 = 1;
211 PORT_P2_4 = 1;
212 PORT_EAN = 1;
213
214 // set up indication pin connections
215 TRIS_P0_4 = 1; // P0_4
216 TRIS_P1_5 = 1; // P1_5
217
218 // set power to on for BM78
219 // this section is currently unused
220 TRIS_SW_BTN = 0;
221 ms_delay(1);
222 PORT_SW_BTN = 1; // power to on, leave as is
223 // end unused section
224
225 // reset and then begin
226 PORT_RST_N = 0; // active low
227 ms_delay(1); // apply an on pulse for at least 63 ns
228 PORT_RST_N = 1;
229
230 ms_delay(100);
231
232 char buffer[16];
233 unsigned int actualLength = 16;
234
235 // main loop
236 while (1)
237 {
238     while(PORT_P0_4 && PORT_P1_5); // wait for a connection to happen
239
240     // disable external interrupt until Bluetoothing is done
241     EX_INT0_InterruptDisable();
242
243     // wait 10 seconds just to be safe (give user time to select that they want to transfer data)
244     for(i = 0; i < 100; ++i) {
245         ms_delay(100);
246     }
247
248     // read data from file into buffer
249     if (f_mount(&drive, "", 1) == FR_OK) {
250
251         if (f_open(&file, "TestRun.TXT", FA_OPEN_EXISTING || FA_READ) == FR_OK) {
252
253             f_lseek(&file, 0); //set file pointer to start of file
254
255             // read through file in a loop

```

```

256         while(actualLength == 16) {
257
258             // read from file
259             f_read(&file, buffer, sizeof(buffer)-1, &actualLength);
260
261             // code used in testing to determine if data was being read
262             /*char dataOut[2];
263             sprintf(dataOut, "%d", actualLength);
264             ms_delay(1);
265             printf(dataOut);*/
266
267             ms_delay(1);
268
269             // output buffer to Bluetooth
270             printf(buffer);
271             ms_delay(1);
272
273         }
274         f_close(&file);
275     }
276
277     f_mount(0, "", 0);
278 }
279
280 ms_delay(333);
281
282 //reset and then restart loop
283 PORT_RST_N = 0; //active low
284 ms_delay(1); //apply an on pulse for at least 63 ns
285 PORT_RST_N = 1;
286
287 // enable external interrupt now that we're done
288 EX_INT0_InterruptEnable();
289 }
290
291 return 1;
292 }
293
294 void ms_delay (int ms)
295 {
296     T2CON = 0x8020; // Timer 2 on, TCKPS<1,0> = 10 this 1:64
297     TMR2 = 0;
298     while(TMR2 < ms*63); // 1/(4MHz/(64*63)) = 0.001024
299     //close to 1ms
300 }
301
302 void us_delay (int us)
303 {
304     T2CON = 0x8000; // Timer 2 on, TCKPS<0,0> = 00 this 1:1
305     TMR2 = 0;
306     while(TMR2 < us*4); // 1/(4MHz/(1*4)) = 1e-6

```

```

307 |     //close to lus
308 | }
309
310 | void CHARLCD1_PutNumber (int num, int chars) {
311 |     char dataOutput[chars]; //declare chars
312 |     sprintf(dataOutput, "%x", num); //convert from number to chars
313 | }
314
315 | void clearBuffer (uint8_t* buff, int size) {
316 |     int i = 0;
317 |     for (i = 0; i < size; ++i) {
318 |         buff[i] = 0x0;
319 |     }
320 | }

```

Figures 12.1.4: Code in main.c for the MPLAB X project used to code the microcontroller. This file handles initialization and also handles Bluetooth communication once the Bluetooth module has obtained a connection. [JM]

```

1
2  /**
3     EXT_INT Generated Driver File
4
5     @Company:
6         Microchip Technology Inc.
7
8     @File Name:
9         ext_int.c
10
11    @Summary
12        This is the generated driver implementation file for the EXT_INT
13        driver using PIC24 / dsPIC33 / PIC32MM MCUs
14
15    @Description:
16        This source file provides implementations for driver APIs for EXT_INT.
17        Generation Information :
18            Product Revision  : PIC24 / dsPIC33 / PIC32MM MCUs - 1.170.0
19            Device            : PIC24FJ1024GB610
20        The generated drivers are tested against the following:
21            Compiler          : XC16 v1.61
22            MPLAB             : MPLAB X v5.45
23    */
24
25  /**
26     (c) 2020 Microchip Technology Inc. and its subsidiaries. You may use this
27     software and any derivatives exclusively with Microchip products.
28
29     THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
30     EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED
31     WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
32     PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION
33     WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.
34
35     IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
36     INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
37     WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
38     BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
39     FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN
40     ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,
41     THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.
42
43     MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE
44     TERMS.
45  */
46
47  /**
48     Section: Includes
49  */
50
51  #include "ext_int.h"

```

```

52
53 //***User Area Begin->code: Add External Interrupt handler specific headers
54 #include "../XL372.h"
55 #include "spil.h"
56 #include "fatfs/ff.h"
57 #include "../pins.h"
58 #include "../ITG3701.h"
59 FATFS drive;
60 FIL file;
61
62 UINT actualLength;
63 char newline[] = "\n";
64 char comma[] = ",";
65 uint8_t writeData[1024];
66 uint8_t readData[1024];
67 uint8_t readDataGyro[1024];
68 //***User Area End->code: Add External Interrupt handler specific headers
69
70 /**
71  * Section: External Interrupt Handlers
72  */
73
74 void __attribute__((weak)) EX_INT0_Callback(void) {
75     // Add your custom callback code here
76     // INT0 triggered -> FIFO_FULL triggered
77
78     const int BUFFER_SIZE = 1024;
79     const int SAMPLE_COUNT_ACCEL = 169;
80     const int SAMPLE_COUNT_GYRO = 84;
81
82     // set back to off mode to stop FIFO data writing
83     XL372_CS = 0; // CS to active low
84
85     // set FIFO to Disabled
86     writeData[0] = XL372_POWER_CTL;
87     writeData[0] = writeData[0] << 1;
88     writeData[1] = 0x00 + 1; //turn FIFO to disabled
89     // plus one is for the FIFO Samples which is 0x1A4, uses bit 0 here.
90     SPI1_Exchange8bitBuffer(writeData, 2, readData);
91
92     XL372_CS = 1; // CS to inactive high
93
94     us_delay(40);
95
96     // get gyro FIFO data first since it is currently active
97     // get FIFO data
98     writeData[0] = ITG3701_FIFO_R_W;
99     writeData[0] += 0x80; //read
100
101     ITG3701_CS = 0; // CS to active low
102

```

```

103 // read 169*3*2 bytes of data (2 bytes per sample, 3 axes, 169 total samples per axis). +1 is for read command
104 SPII_Exchange8bitBuffer(writeData, SAMPLE_COUNT_GYRO*3*2 + 1, readDataGyro);
105
106 ITG3701_CS = 1; // CS to inactive high
107
108 // get accel FIFO data
109 int i = 0;
110 for (i = 0; i < BUFFER_SIZE; ++i) {
111     writeData[i] = 0x0;
112 }
113 writeData[0] = XL372_FIFO_DATA;
114 writeData[0] = writeData[0] << 1;
115 ++writeData[0]; //read
116
117 XL372_CS = 0; // CS to active low
118
119 // read 169*3*2 bytes of data (2 bytes per sample, 3 axes, 169 total samples per axis). +1 is for read command
120 SPII_Exchange8bitBuffer(writeData, SAMPLE_COUNT_ACCEL*3*2 + 1, readData);
121
122 XL372_CS = 1; // CS to inactive high
123
124 // write data to file
125 if (f_mount(&drive,"",1) == FR_OK) {
126
127     if (f_open(&file, "TestRun.TXT", FA_OPEN_APPEND | FA_WRITE | FA_READ ) == FR_OK) {
128
129         char dataOutput[16]; // declare chars
130         sprintf(dataOutput, "%02x", readData[1]); //convert from number to chars
131
132         // write accel data to file
133         for (i = 1; i <= SAMPLE_COUNT_ACCEL*3*2; i+=2) {
134
135             // sum the relevant values. MSB first, then LSB
136             uint16_t origData = readData[i] * 256;
137             origData += readData[i+1];
138             origData = origData / 16;
139             if((origData & 0x0800) != 0) {
140                 origData = origData | 0xF000; // add leading ones if needed
141             }
142             else { // handle positive
143                 origData = origData & 0x0FFF; // add leading zeroes if needed
144             }
145
146             short int twosComp = *(short int*)&origData;
147             sprintf(dataOutput, "%06d", twosComp); // convert from number to chars
148             f_write(&file, dataOutput, 6, &actualLength);
149             f_write(&file, comma, 1, &actualLength);
150
151         }
152
153         // write gyro data to file

```



```

154         for (i = 1; i <= SAMPLE_COUNT_GYRO*3*2; i+=2) {
155
156
157             // sum the relevant values. MSB first, then LSB
158             uint16_t origData = readData[i] * 256;
159             origData += readData[i+1];
160             short int twosComp = *(short int*)&origData;
161             sprintf(dataOutput, "%06d", twosComp); // convert from number to chars
162             f_write(&file, dataOutput, 6, &actualLength );
163             f_write(&file, comma, 1, &actualLength );
164
165         }
166
167         f_write(&file, newline, 1, &actualLength);
168         sprintf(dataOutput, "%02x", readData[1]); // convert from number to chars
169         f_close(&file);
170     }
171
172     f_mount(0, "", 0);
173 }
174
175 // set back to instant on mode
176 // this will also set up for the next impact
177 // set to instant on mode, low threshold (do this before reading the data)
178 writeData[0] = XL372_POWER_CTL;
179 writeData[0] = writeData[0] << 1;
180 writeData[1] = 0x02;
181 XL372_CS = 0; // CS to active low
182
183 SPI1_Exchange8bitBuffer(writeData, 2, readData);
184
185 XL372_CS = 1; // CS to inactive high
186
187 us_delay(40);
188
189 // read status register to clear the FIFO_FULL interrupt
190 XL372_CS = 0; // CS to active low
191
192 // read status
193 writeData[0] = XL372_STATUS;
194 writeData[0] = writeData[0] << 1;
195 ++writeData[0]; // read
196 SPI1_Exchange8bitBuffer(writeData, 2, readData);
197
198 XL372_CS = 1; // CS to inactive high
199
200 us_delay(40);
201
202 ms_delay(20); // delay for >16 ms for change to instant on mode
203
204 }

```

```

205
206  /**
207     Interrupt Handler for EX_INT0 - INT0
208  */
209  void __attribute__((interrupt, no_auto_psv)) _INT0Interrupt(void) {
210     /***User Area Begin->code: INT0 - External Interrupt 0***
211
212     EX_INT0_CallBack();
213
214     /***User Area End->code: INT0 - External Interrupt 0***
215     EX_INT0_InterruptFlagClear();
216  }
217  /**
218     Section: External Interrupt Initializers
219  */
220
221  /**
222     void EXT_INT_Initialize(void)
223
224     Initializer for the following external interrupts
225     INT0
226  */
227  void EXT_INT_Initialize(void) {
228     /***
229     * INT0
230     * Clear the interrupt flag
231     * Set the external interrupt edge detect
232     * Enable the interrupt, if enabled in the UI.
233     ***/
234     EX_INT0_InterruptFlagClear();
235     EX_INT0_PositiveEdgeSet();
236     EX_INT0_InterruptEnable();
237  }

```

Figures 12.1.5: Code in ext_int.c for the MPLAB X project used to code the microcontroller.

This file contains MCC generated code as well as user-written code for handling the external interrupt on INT0, which is triggered by the accelerometer's FIFO queue reaching its capacity.

The interrupt service routine reads data from the accelerometer and gyroscope and stores it to the microSD. It also resets the configurations for the accelerometer as needed. [JM]

```

1  /*-----
2  File Name      : XL372.h
3  Author        : MPD Application Team
4  Version       : V1.0
5  Date          : 12/03/2016
6  Description   : XL372 Registers and bits values
7  File ID      : $Id: XL372.h,v 1.1.1.1 2011/12/02$
8
9  Analog Devices ADXL 372 digital output accelerometer
10 with advanced digital features.
11
12 Based on preliminary data sheet Rev l2b
13
14 (c) 2011 Analog Devices application support team.
15 xxx@analog.com
16
17 -----
18
19 The present firmware which is for guidance only aims at providing
20 customers with coding information regarding their products in order
21 for them to save time. As a result, Analog Devices shall not be
22 held liable for any direct, indirect or consequential damages with
23 respect to any claims arising from the content of such firmware and/or
24 the use made by customers of the coding information contained herein
25 in connection with their products.
26
27 -----*/
28
29 #ifndef __XL372_H
30 #define __XL372_H
31
32 /* ----- Register names ----- */
33 #define XL372_DEVID_AD          0x00
34 #define XL372_DEVID_MST        0x01
35 #define XL372_PARTID           0x02
36 #define XL372_REVID            0x03
37 #define XL372_STATUS           0x04
38 #define XL372_STATUS2         0x05
39 #define XL372_FIFO_ENTRIES2    0x06
40 #define XL372_FIFO_ENTRIES     0x07
41
42 #define XL372_XDATA_H          0x08
43 #define XL372_XDATA_L          0x09
44
45 #define XL372_YDATA_H          0x0A
46 #define XL372_YDATA_L          0x0B
47
48 #define XL372_ZDATA_H          0x0C
49 #define XL372_ZDATA_L          0x0D
50
51 #define XL372_MAXPEAK_X_H      0x15

```

```

52 #define XL372_MAXPEAK_X_L          0x16
53
54 #define XL372_MAXPEAK_Y_H          0x17
55 #define XL372_MAXPEAK_Y_L          0x18
56
57 #define XL372_MAXPEAK_Z_H          0x19
58 #define XL372_MAXPEAK_Z_L          0x1A
59
60 #define XL372_OFFSET_X             0x20
61 #define XL372_OFFSET_Y             0x21
62 #define XL372_OFFSET_Z             0x22
63
64 #define XL372_THRESH_ACT_X_H        0x23
65 #define XL372_THRESH_ACT_X_L        0x24
66
67 #define XL372_THRESH_ACT_Y_H        0x25
68 #define XL372_THRESH_ACT_Y_L        0x26
69
70 #define XL372_THRESH_ACT_Z_H        0x27
71 #define XL372_THRESH_ACT_Z_L        0x28
72
73 #define XL372_TIME_ACT              0x29
74
75 #define XL372_THRESH_INACT_X_H      0x2A
76 #define XL372_THRESH_INACT_X_L      0x2B
77
78 #define XL372_THRESH_INACT_Y_H      0x2C
79 #define XL372_THRESH_INACT_Y_L      0x2D
80
81 #define XL372_THRESH_INACT_Z_H      0x2E
82 #define XL372_THRESH_INACT_Z_L      0x2F
83
84 #define XL372_TIME_INACT_H           0x30
85 #define XL372_TIME_INACT_L           0x31
86
87 #define XL372_THRESH_ACT2_X_H        0x32
88 #define XL372_THRESH_ACT2_X_L        0x33
89
90 #define XL372_THRESH_ACT2_Y_H        0x34
91 #define XL372_THRESH_ACT2_Y_L        0x35
92
93 #define XL372_THRESH_ACT2_Z_H        0x36
94 #define XL372_THRESH_ACT2_Z_L        0x37
95
96 #define XL372_HPF                    0x38
97
98 #define XL372_FIFO_SAMPLES            0x39
99 #define XL372_FIFO_CTL                0x3A
100
101 #define XL372_INT1_MAP                0x3B
102 #define XL372_INT2_MAP                0x3C

```

```

103
104 #define XL372_TIMING                0x3D
105 #define XL372_MEASURE               0x3E
106 #define XL372_POWER_CTL            0x3F
107 #define XL372_SELF_TEST            0x40
108 #define XL372_RESET                0x41
109
110 #define XL372_FIFO_DATA             0x42
111
112
113 /*-----
114  Bit field definitions and register values
115  -----*/
116 /* register values for Analog Devices ID */
117 /* The Analog Devices ID should always read this value,
118    The customer does not need to use this value but it can be used to
119    check that the device can communicate */
120
121 #define ADI_ID          0xad
122
123 /* register values for Analog Devices MEMS ID */
124 /* The Analog Devices MEMS ID should always read this value,
125    The customer does not need to use this value but it can be used to
126    check that the device can communicate */
127
128 #define MEMS_ID         0x1d
129
130 /* register values for part ID */
131 /* The part ID should always read this value,
132    The customer does not need to use this value but it can be used to
133    check that the device can communicate */
134
135 #define PART_ID        0xfa
136
137 /* register values for revision ID */
138 /* This register contains a product revision ID, beginning with
139    0x00 and is changed for each subsequent revision */
140
141 #define REV_ID         0x02
142
143 #define XL372_CS PORTFbits.RF5
144
145 #endif /* __XL372_H */

```

Figures 12.1.6: Code in ADXL372.h used in the microcontroller MPLAB X project. This file contains definitions related to the ADXL372 accelerometer, including register mappings and the chip select pin. [JM]

```

2 | * File: ITG3701.h
3 | *
4 | * Created on March 28, 2021, 10:21 PM
5 | */
6 |
7 | #ifndef ITG3701_H
8 | #define ITG3701_H
9 |
10 | // See also ITG3701 Register Map and Descriptions
11 | // http://www.invensense.com/mems/gyro/documents/RM-000001-ITG-3701-RM.pdf
12 | //
13 | ///////////////////////////////////////////////////////////////////
14 | // ITG3701 Gyro Registers //
15 | ///////////////////////////////////////////////////////////////////
16 | #define ITG3701_XG_OFFS_TC_H 0x04
17 | #define ITG3701_XG_OFFS_TC_L 0x05
18 | #define ITG3701_YG_OFFS_TC_H 0x07
19 | #define ITG3701_YG_OFFS_TC_L 0x08
20 | #define ITG3701_ZG_OFFS_TC_H 0x0A
21 | #define ITG3701_ZG_OFFS_TC_L 0x0B
22 | #define ITG3701_XG_OFFS_USRH 0x13 // User-defined trim values for gyroscope
23 | #define ITG3701_XG_OFFS_USRL 0x14
24 | #define ITG3701_YG_OFFS_USRH 0x15
25 | #define ITG3701_YG_OFFS_USRL 0x16
26 | #define ITG3701_ZG_OFFS_USRH 0x17
27 | #define ITG3701_ZG_OFFS_USRL 0x18
28 | #define ITG3701_SMPLRT_DIV 0x19
29 | #define ITG3701_CONFIG 0x1A
30 | #define ITG3701_GYRO_CONFIG 0x1B
31 | #define ITG3701_FIFO_EN 0x23
32 | #define ITG3701_INT_PIN_CFG 0x37
33 | #define ITG3701_INT_ENABLE 0x38
34 | #define ITG3701_INT_STATUS 0x3A
35 | #define ITG3701_TEMP_OUT_H 0x41
36 | #define ITG3701_TEMP_OUT_L 0x42
37 | #define ITG3701_GYRO_XOUT_H 0x43
38 | #define ITG3701_GYRO_XOUT_L 0x44
39 | #define ITG3701_GYRO_YOUT_H 0x45
40 | #define ITG3701_GYRO_YOUT_L 0x46
41 | #define ITG3701_GYRO_ZOUT_H 0x47
42 | #define ITG3701_GYRO_ZOUT_L 0x48
43 | #define ITG3701_USER_CTRL 0x6A
44 | #define ITG3701_PWR_MGMT_1 0x6B // Device defaults to the SLEEP mode
45 | #define ITG3701_PWR_MGMT_2 0x6C
46 | #define ITG3701_FIFO_COUNTH 0x72
47 | #define ITG3701_FIFO_COUNTL 0x73
48 | #define ITG3701_FIFO_R_W 0x74
49 | #define ITG3701_WHO_AM_I 0x75 // Should return 0x68
50 |
51 | #define ITG3701_CS PORTBbits.RB9
52 |
53 | #endif /* ITG3701_H */

```

Figure 12.1.7: Code in ITG3701.h used in the microcontroller MPLAB X project. This file contains definitions related to the ITG3701 gyroscope, including register mappings and the chip select pin. [JM]

```

1  /*
2  * File:   pins.h
3  *
4  * Created on February 23, 2021, 12:39 PM
5  */
6
7  #ifndef PINS_H
8  #define PINS_H
9
10 #ifdef __cplusplus
11 extern "C" {
12 #endif
13
14 // pin setup
15 #define TRIS_SW_BTN TRISAbits.TRISA3
16 #define PORT_SW_BTN PORTAbits.RA3
17 #define TRIS_P2_0 TRISAbits.TRISA2
18 #define PORT_P2_0 PORTAbits.RA2
19 #define TRIS_P2_4 TRISGbits.TRISG2
20 #define PORT_P2_4 PORTGbits.RG2
21 #define TRIS_EAN TRISGbits.TRISG3
22 #define PORT_EAN PORTGbits.RG3
23 #define TRIS_BM78_RX TRISBbits.TRISB15
24 #define PORT_BM78_RX PORTBbits.RB15
25 #define TRIS_BM78_TX TRISBbits.TRISB14
26 #define PORT_BM78_TX PORTBbits.RB14
27 #define TRIS_RST_N TRISBbits.TRISB13
28 #define PORT_RST_N PORTBbits.RB13
29 #define TRIS_WAKE TRISBbits.TRISB12
30 #define PORT_WAKE PORTBbits.RB12
31 #define TRIS_P0_4 TRISBbits.TRISB11
32 #define PORT_P0_4 PORTBbits.RB11
33 #define TRIS_P1_5 TRISBbits.TRISB10
34 #define PORT_P1_5 PORTBbits.RB10
35
36 #ifdef __cplusplus
37 }
38 #endif
39
40 #endif /* PINS_H */
41

```

Figure 12.1.8: Code in pins.h used in the microcontroller MPLAB X project. This file contains register mapping definitions related to the BM78 Bluetooth Module. [JM]

12.2. Matlab Code

```
%Senior Design DT10
%processCSV.m

clc
clear

%sample data will be of the following format:
%one line = one impact
%first 169*3 data values: linear accelerations in gs
%next 84*3 data values: angular velocities in deg/s
num_LA = 169;
num_AA = 83;

M = readmatrix('Late2.txt');

LA_xyz = M(:, 1:num_LA*3);
AV_xyz = M(:, num_LA*3+1:num_LA*3+(num_AA+1)*3);
num_samples = size(M,1);

AV_xyz = AV_xyz ./ 16; %remove unnecessary bottom 4 bits

%divide by 10 for LA, divide by 8.2 for AV
LA_xyz = LA_xyz ./ 10;
AV_xyz = AV_xyz ./ 8.2;

%numerically differentiate angular velocities to get angular accelerations
h = 1/8000; %sample T for gyro
AA_xyz = AV_xyz(:, 1:num_AA*3);
for i = 1:3:num_AA*3-2
    AA_xyz(:,i) = (AV_xyz(:,i+3)-AV_xyz(:,i))/h;
    AA_xyz(:,i+1) = (AV_xyz(:,i+4)-AV_xyz(:,i+1))/h;
    AA_xyz(:,i+2) = (AV_xyz(:,i+5)-AV_xyz(:,i+2))/h;
end

%convert to rad/s^2
AA_xyz = AA_xyz .* pi/180;

%this gives us 169 LAs and 83 AAs for each impact
%now find the magnitude of each
LA = zeros(num_samples, num_LA);
AA = zeros(num_samples, num_AA);
for i = 1:num_LA
    for j = 1:num_samples
        LA(j,i) = norm([LA_xyz(j,3*i-2) LA_xyz(j,3*i-1) LA_xyz(j,3*i)]);
    end
end
for i = 1:num_AA
    for j = 1:num_samples
        AA(j,i) = norm([AA_xyz(j,3*i-2) AA_xyz(j,3*i-1) AA_xyz(j,3*i)]);
    end
end
```



```

%make HIC and GSI data sets *****
%since HIC is based on LA, find total sample time
accelRate = 3200;
total_sample_time = 1/3200*num_LA;

%since the sample time was 0.0528 seconds = 52.8 ms, we can do 3 HIC
%calculations. Use the largest generated HIC on the interval
dt = 15e-3;
num_HIC_samples = dt*3200; %48 is a nice even number

%calculate integrals using trapz function
spacing = 1/3200:1/3200:0.015;
int1 = zeros(num_samples, 1);
int2 = zeros(num_samples, 1);
int3 = zeros(num_samples, 1);
for i = 1:num_samples
    int1(i) = trapz(spacing, LA(i, 1:48));
    int2(i) = trapz(spacing, LA(i, 49:96));
    int3(i) = trapz(spacing, LA(i, 97:144));
end

%calculate HICs
HIC1 = dt .* (1/dt .* int1).^(5/2);
HIC2 = dt .* (1/dt .* int2).^(5/2);
HIC3 = dt .* (1/dt .* int3).^(5/2);

%take max
HIC = zeros(num_samples, 1);
for i = 1:num_samples
    HIC(i) = max([HIC1(i) HIC2(i) HIC3(i)]);
end

%make GSI data set
GSI1 = zeros(num_samples, 1);
GSI2 = zeros(num_samples, 1);
GSI3 = zeros(num_samples, 1);
for i = 1:num_samples
    GSI1(i) = trapz(spacing, LA(i, 1:48).^(5/2));
    GSI2(i) = trapz(spacing, LA(i, 49:96).^(5/2));
    GSI3(i) = trapz(spacing, LA(i, 97:144).^(5/2));
end

%take max
GSI = zeros(num_samples, 1);
for i = 1:num_samples
    GSI(i) = max([GSI1(i) GSI2(i) GSI3(i)]);
end

%make set of peak magnitudes
LA_peak = zeros(num_samples, 1);
AA_peak = zeros(num_samples, 1);
for i = 1:num_samples
    LA_peak(i) = max(LA(i, :));
    AA_peak(i) = max(AA(i, :));
end

```

```

%now that we have all our data, we need to make our data matrix on which we
%will perform PCA.
dataset = [LA_peak AA_peak HIC GSI];

%standardize the data using values obtained from relevant studies
mean_LA = 43;
sd_LA = 11.5;
mean_AA = 4029.5;
sd_AA = 1434.8;
mean_HIC = 74.7479;
sd_HIC = 46.9480;
mean_GSI = 74.7479;
sd_GSI = 46.9480;
%The LA and AA mean values are obtained from the 300,000 impact sample data
%set from Greenwald et. al. The LA and AA standard deviations were not made
%available, so the standard deviations of studies with similar means were
%used in their place. The HIC and GSI means and standard deviations are
%made based on a simulation of the LA data using the previously mentioned
%parameters. This can be done since HIC and GSI are dependent on LA. This
%work was done in a separate project and is not shown here.

col_mean = [mean_LA mean_AA mean_HIC mean_GSI];
col_std = [sd_LA sd_AA sd_HIC sd_GSI];

dataset_std = zeros(num_samples, 4);

for j = 1:4
    for i = 1:num_samples
        dataset_std(i, j) = (dataset(i, j) - col_mean(j))/col_std(j);
    end
end

%with the standardized data, use PCS equation
user_PCS_scores = 10 * (0.4718*dataset_std(:,3) + 0.4742*dataset_std(:,4) + 0.4336*dataset_std(:,1) + 0.2164*dataset_std(:,2)) + 2;

%calculate PFV at 50%, 75%, 90%
cpl = [.50; .75; .90];
N = [1687; 4389; 104370];
ppv = cpl.*17./(cpl.*17 + N);

ppv_poly = polyfit(cpl, ppv, 1); %linear approximation

%CPL as a function of PCS is a line using (35, 1) and (160, 1/17)
cplpts = [1/17; 1];
pcsppts = [160; 35];
cpl_poly = polyfit(pcsppts, cplpts, 1);

%calculate approximate ppv for each of the 100 sample points
user_CPL_vals = -0.007529*user_PCS_scores + 1.264;
for i = 1:num_samples
    if (user_CPL_vals(i) < 0)
        user_CPL_vals(i) = 0; %correct negative values
    end
end
user_PPV_vals = -0.01179*user_CPL_vals + 0.01113;

```

```

for i = 1:num_samples
    if (user_PPV_vals(i) < 0)
        user_PPV_vals(i) = 0; %correct negative values
    end
end

%calculate likelihood of concussion
concussion_risk = 1;
for i = 1:num_samples
    concussion_risk = concussion_risk * (1 - user_PPV_vals(i));
end

concussion_risk = 1 - concussion_risk;
percentage_risk = 100 * concussion_risk

%plot data from significant impacts
figure(1)
histogram(LA, 'Normalization', 'pdf')
title('Linear Accelerations Recorded')
figure(2)
histogram(AA, 'Normalization', 'pdf')
title('Angular Accelerations Recorded')
figure(3)
histogram(HIC, 'Normalization', 'pdf')
title('HICs Recorded')
figure(4)
histogram(GSI, 'Normalization', 'pdf')
title('GSIs Recorded')

```

Figures 12.2.1: Matlab code used to calculate impact measurements and approximate concussion risk. Note that the final output is given by percentage_risk and the histograms in figures 1-4. For larger impact data sets, the LA peak and AA peak values can be output instead. For a sample result using impacts recorded by the accelerometer and gyroscope on the prototype device, see Figure 5.2.8.1. [JM]