Capstone Phase 4: Final Report

Performance Tracking Hockey Puck

Team #2

A Report Presented to

The Department of Electrical & Computer Engineering

Concordia University

In Partial Fulfilment of the Requirements of

ELEC/COEN 490

By

Eric Chin	*ID: 40007868*
David Hine	*ID: 40000840*
Leonidas Kyropoulos	*ID: 40004935*
Michael West	*ID: 40002131*

Project Supervisor

Dr. Glenn Cowan

Concordia University

September 2018

Abstract

The purpose of the final phase of this project is to complete the design implementation of our product, as scheduled in Phases 1 and 2. Building on the design of our performance tracking hockey puck demonstrated in Phase 3, we addressed the technical issues encountered in our design and corrected them with appropriate solutions. Also, validation and testing was performed on the puck to verify that it met the criteria and specifications outlined in the previous phases. Upon completion of this phase, the puck will be handed off to the McGill Ice Hockey Research Group for use in their future studies.

Table of Contents

Abstract 3

List of Figures 6

List of Tables 7

1. Objectives and Requirements 8

1.1 Requirements 8

1.2 Design Specifications 9

2. Description of the Project 10

2.1 Overview 10

2.2 Initial Design 11

2.2.1 Microcontroller and Wireless communication 11

2.2.2 Mechanical Assembly 12

2.3 Development 13

2.3.1 Electrical System 13

2.3.2 Microcontroller and Wireless Communication 17

2.3.3 User Interface 22

2.3.4 Mechanical Assembly 26

2.4 Issues Encountered and Solutions 28

2.4.1 Electrical System 28

2.4.2 Microcontroller and Wireless Communication 30

2.4.3 User Interface 31

2.4.4 Mechanical Assembly 32

3. Results Obtained 33

3.1 Electrical System 33

3.2 Accelerometer 33

3.3 Gyroscope 36

3.4 Wake up from lower power mode 36

3.5 Battery life 37

3.6 Charging time 37

3.7 Battery protection 37

3.8 Wireless Range 38

4. Feedback on Lessons Learned 38

5. Conclusions 39

6. References 39
7. Graduate Attributes: Real-time Technology Assessment 40
Appendix A – Technical User Manual 43
1. Setup for User 43
1.1 Installing nRFConnect for Desktop 43
1.2 Configuring the dongle 43
2. Setup for Developer 44
2.1 Installing SES 44
2.2 Programming the nRF52840 Dev Kit 45
2.3 Programming puck using the nRF52840 Dev Kit 46
3. Using the Puck 48
4. User Interface 48
5. GitHub Repository 50
Appendix B – Bill of Materials 51

List of Figures

Figure 1: System Block Diagram 10
Figure 2: Preliminary Electrical System Design 11
Figure 3: Openable Puck Design 12
Figure 4: Final Electrical System 15
Figure 5: Top and bottom views of the PCB layout 16
Figure 6: GATT Transaction Hierarchy 18
Figure 7: Graphical User Interface 25
Figure 8. Alternative Puck Design 26
Figure 9: Final Physical Assembly 27
Figure 10: Internal charging cable wires 29
Figure 11: Re-mapping of internal wires 29
Figure 12: Samples over 0.04s 34
Figure 13: Voltage measured from accelerometer 34
Figure 14: Calculated g-force from voltage 35
Figure 15: High vertical acceleration test 36
Figure 16: Connections on the Dev Kit for programming puck 47
Figure 17: Pins on the PCB Figure 47
Figure 18: Pins to short 47
Figure 19: Connections between Dev Kit and PCB (top view of PCB) 48
Figure 20: Graphical User Interface 50

List of Tables

Table 1: Requirements 8

Table 2: Design Specificaitions 9

Table 3: Final BOM 51

1. Objectives and Requirements

1.1 Requirements

The objectives of the performance tracking hockey puck are to measure its own linear acceleration and contact time with the hockey stick during a shot, then to wirelessly transmit the data to a program in real time, all while maintaining very similar physical properties of a normal ice hockey puck.

There are a few constraints that must be kept in mind during the design and development of the prototype. Data measurement testing must take place in a simulated hockey rink environment.

The table below displays all of the demands regarding the hockey puck's requirements. The functional requirements show what the puck is supposed to do and the non-functional requirements define a few specifications preferred by the customer.

Table 1: Requirements

Functional	**Non-Functional**
Measure linear acceleration	Wireless
Measure angular velocity (spin)	Indication of low battery (rechargeable)
Calculate contact time of puck with stick	Suitable for use on ice and on shooting pad
Calculate peak puck velocity	
Record data in trials and record in separate files	

Due to some harsh conditions that this device will need to withstand, the following constraints will be considered:

1. The device must be able to operate in cold temperatures. The temperature of the ice in a hockey arena is approximated -10 °C.
2. Due to condensation and friction with a cold surface, the puck must tolerate high humidity.

3. The whole system must match the physical characteristics (weight, size, center of gravity) of a standard hockey puck.
4. The puck must have high durability to withstand an elite player's slapshot.

1.2 Design Specifications

The table below displays the basic numerical design specifications of the performance tracking hockey. As mentioned previously, the puck should reproduce the dimensions of a standard regulation hockey ice hockey puck (specifications 1-3). Note that the puck will be designed and tested for use in standard temperature/humidity conditions as well as those of an ice hockey rink.

For the acceleration, we only need a measurement within 1g because the values we are concerned with are the average acceleration over the duration of the shot. We are not concerned with directly measuring small values of g force, so ±1g should suffice. Also, when integrating the acceleration to get the peak linear velocity of the puck, not requiring small values of acceleration either.

Table 2: Design Specifications

#	Description/ Parameter	Test Conditions	Value			Unit
			Min.	Typ.	Max.	
1	Diameter			76		mm
2	Thickness			25		mm
3	Mass		156		170	g
4	Battery life	Normal[1], Ice hockey rink[2]	1			hr
5	Measured acceleration	Normal[1], Ice hockey rink[2]	1		500	g (force)
6	Acceleration tolerance	Normal[1], Ice hockey rink[2]		±1		g (force)
7	Linear velocity	Normal[1], Ice hockey rink[2]			175	km/h

8	Linear velocity tolerance	Normal[1], Ice hockey rink[2]		±1		m/s
9	Measured angular rate	Normal[1], Ice hockey rink[2]			4000	°/s
10	Angular rate tolerance	Normal[1], Ice hockey rink[2]		±10		%
11	Operational temperature	Normal[1], Ice hockey rink[2]	-10		30	°C
12	Signal range	Normal[1], Ice hockey rink[2]			35	m

Note 1: Under standard room temperatures and relative humidity levels

Note 2: Tested on an ice hockey rink

2. Description of the Project

2.1 Overview

The performance tracking hockey puck is a battery-powered, microcontroller-based system which acquires data from an analog accelerometer and a digital gyroscope sensor and transmits them to a receiving microcontroller on board a USB dongle, through which the data can be recorded and accessed from a PC graphical user interface. Figure 1 displays the general flow between the system's main components.

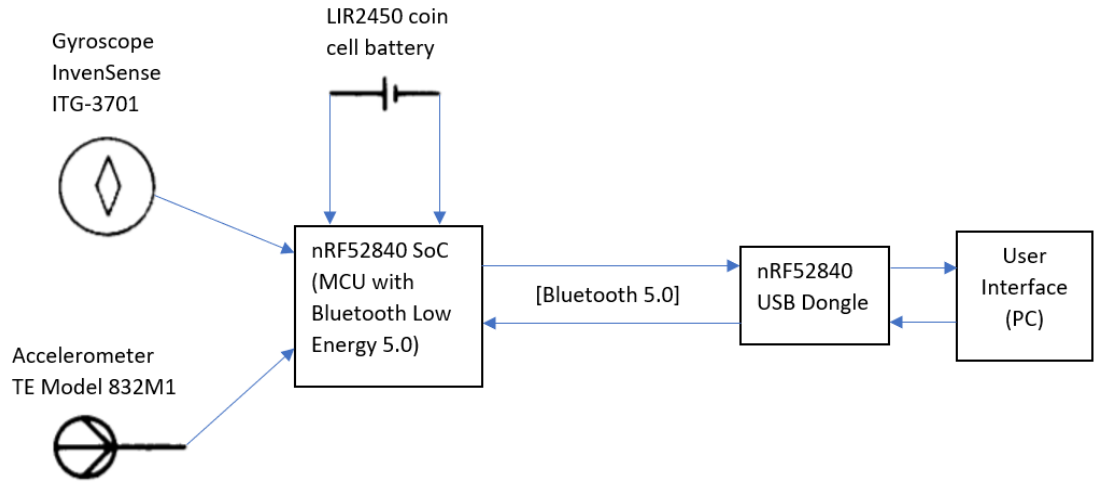

Figure 1: System Block Diagram

2.2 Initial Design

2.2.1 Microcontroller and Wireless communication

Figure 2 shows the initial electrical system design from Phase 2 of the project, designed with the KiCAD software. This consisted of the nRF52840 microcontroller, the TE832-M1 accelerometer, the ITG-3701 gyroscope, the coin cell battery, the antenna, as well as the required resistors/capacitors/inductors.

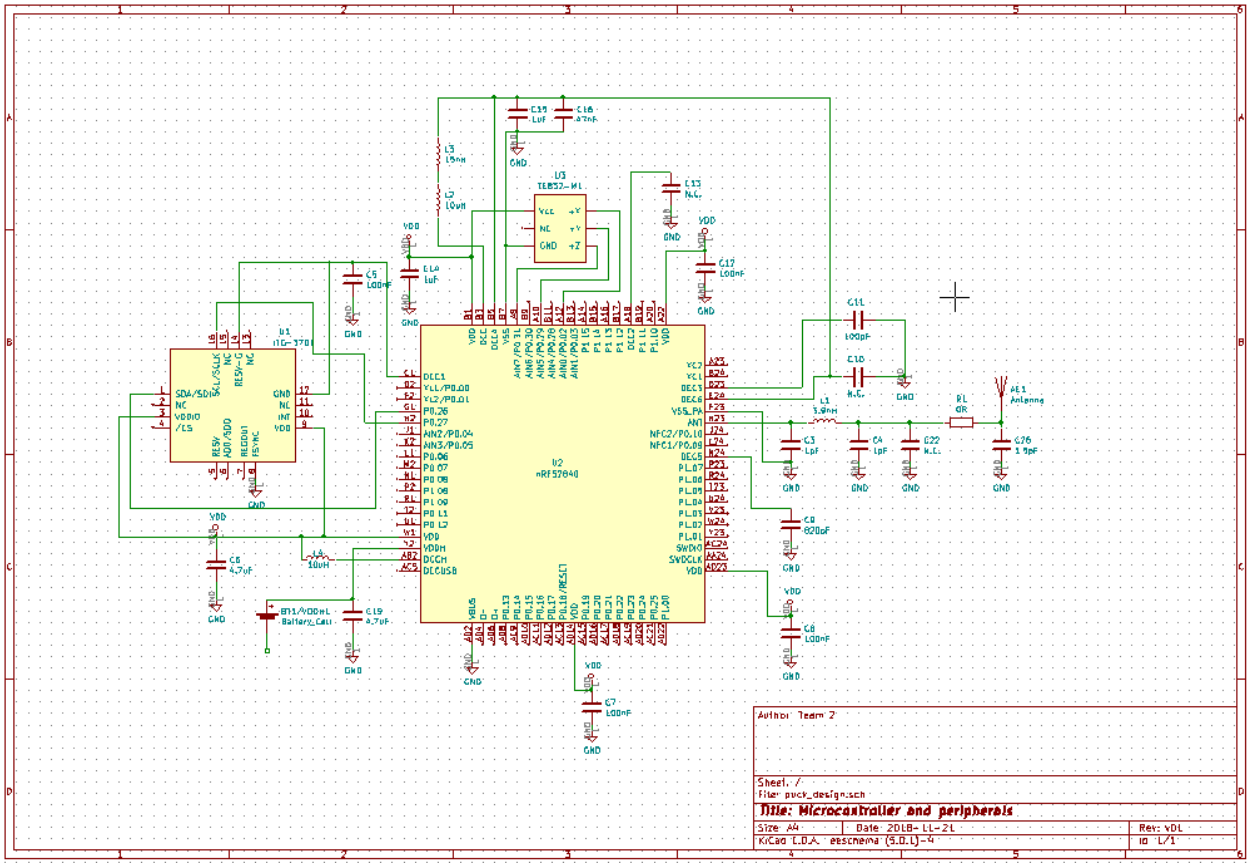

Figure 2: Preliminary Electrical System Design

2.2.2 Mechanical Assembly

Two options were considered for the structural assembly of the puck. The first option is to have an openable enclosure that would be sealed using screws, as shown in the figure below.

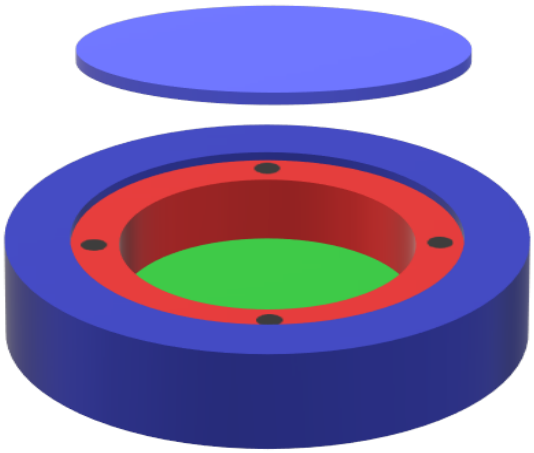

Figure 3: Openable Puck Design

This design grants us good accessibility to all the components inside, allowing us to easily make changes to the circuitry inside the puck, while maintaining a respectable degree of durability. The second option is a sealed enclosure. Once all components are correctly placed inside and functional, a layer of vulcanized rubber will be added on top of the components, which will fill all the empty space and completely sealing the puck. This will ensure stronger durability and maintain the physical properties of a puck as much as possible

The sealed enclosure option was chosen as the better design. However, debugging is very important in the early development process of the project. Therefore, we would need frequent access to the interior components of the puck for updates and alterations.

Due to these circumstances, we will be using an openable enclosure during development of our capstone project.

2.3 Development

2.3.1 Electrical System

Although the preliminary design provided the basic functionality, there were still many components to be added to form a complete system. The following changes or additions were made to the design:

- **Antenna:** It was decided that designing our own antenna would be quite a challenging task, which none of the team members had sufficient knowledge or experience for. As such, the Fanstel BT840F, a custom version of the nRF52840 microcontroller including a PCB trace antenna, was used.
- **Battery charging:** As per our initial design, the puck should be rechargeable while remaining completely sealed. Two options were considered for charging: by cable or wireless (inductive). We were not confident in the feasibility of inductive charging to provide a reasonable charging time for the battery. As such, it was decided to proceed with standard charging via a cable. As will be explained further on, the required circuitry and hardware was added to the design.
- **Battery Protection:** Any battery-powered system should have circuitry to protection the battery from over-current discharging or over-discharging (undervoltage protection). Therefore, appropriate circuitry was added to accomplish this.

Due to the very small size of components and general lack of available breakout boards for testing them, it would be impractical to attempt to implement the electrical system on a breadboard first. As such, more time was spent in ensuring that the designed system was complete and could be implemented on a printed circuit board straight away.

The final electrical system is shown in Figure 4. Each main component and its functionality are described below.

- **Microcontroller (U2):** The Fanstel BT840F MCU module includes the nRF52840 microcontroller with integrated Bluetooth Low Energy capability, as well as a PCB trace antenna. As will be explained in detail further on, it acquires the data from the accelerometer and gyroscope and transmits them to the user interface PC over Bluetooth Low Energy. Additionally, its internal voltage regulator is used to provide and maintain a constant 3.3V voltage to the sensors. Note that a 0Ω resistor was placed to temporarily disconnected the battery circuitry from the microcontroller while the former was being tested.

- **Accelerometer:** The TE 832-M1 500g-range accelerometer provides an analog voltage signal for each of the 3 axes (X, Y, Z directions) with a resolution of 2.5 mV/g. The 0g reference is equal to its excitation voltage divided by 2. These signals are relayed to the microcontroller's ADC inputs.

- **Gyroscope (U1):** The ITG-3701 gyroscope provides digital signals for 3 axes of rotation (around X, Y, Z). Using SPI communication, the microcontroller can acquire the rotational data for each axis.

- **Charge management controller (U4):** The MCP73832-2 charge management controller chip controls the charging voltage and current for the coin cell battery. The 20kΩ programming resistor sets the constant charging current to 50 mA, which is appropriate for the type of battery used. The STAT pin provides a signal corresponding to the charging status, which is either low (battery charging) or high impedance (charge complete). For the latter case, the signal is tied to the microcontroller's level via a 10kΩ pull-up resistor.

- **Battery protection chip (U5):** The BQ29700 integrated circuit chip protects the battery from excessive discharging or discharging current. It works by sensing the voltage across two MOSFETS (one for charging and one for discharging) and adjusting their gate voltage to control whether the return path to the battery is open or closed.

PCB Design

Once the final electrical system was designed, a printed circuit board was designed. The main constraints dictating the size, shape and design of the PCB were the following:

- The board should have a single layer of copper, with all the components placed on the top side such that reflow soldering can be performed.
- Since it must measure the angular velocity of the puck, the gyroscope should be placed in the center of the board.
- It should have a circular shape for more even distribution and stability within the puck. Its radius should be as small as possible.
- There should be as little board material underneath the antenna as possible, as to maximize the signal range.

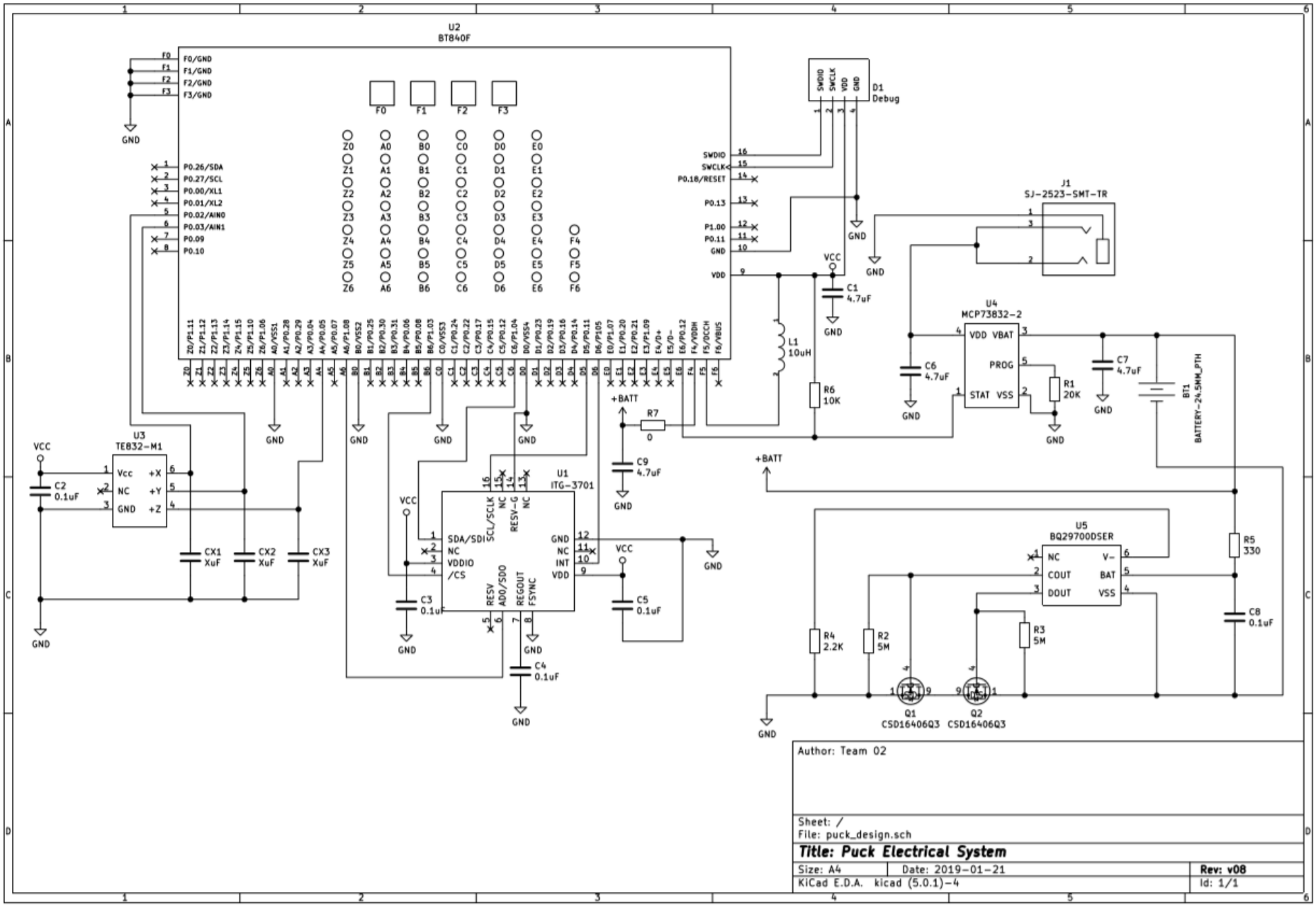

Figure 4: Final Electrical System

Given these factors, the PCB shown in Figure 5 was designed using the KiCAD software. Some notable features are the following:

- A header providing a connection point to the 4 pins required to program the microcontroller unit was added at the edge of the board.
- In general, power signal paths were routed on the bottom side of the board, and ground paths were routed on the top side.
- Decoupling capacitors were placed as close as possible to the power output pins, to reduce noise in the circuit.
- The radius is 1.1 inches, limited by the size of the battery holder, which cannot cover the center due to the gyroscope.
- The majority of smaller components were placed on the opposite side as the larger, heavier components in order too help balance the weight.
- Two mounting holes were included to be able to screw the board into the puck.

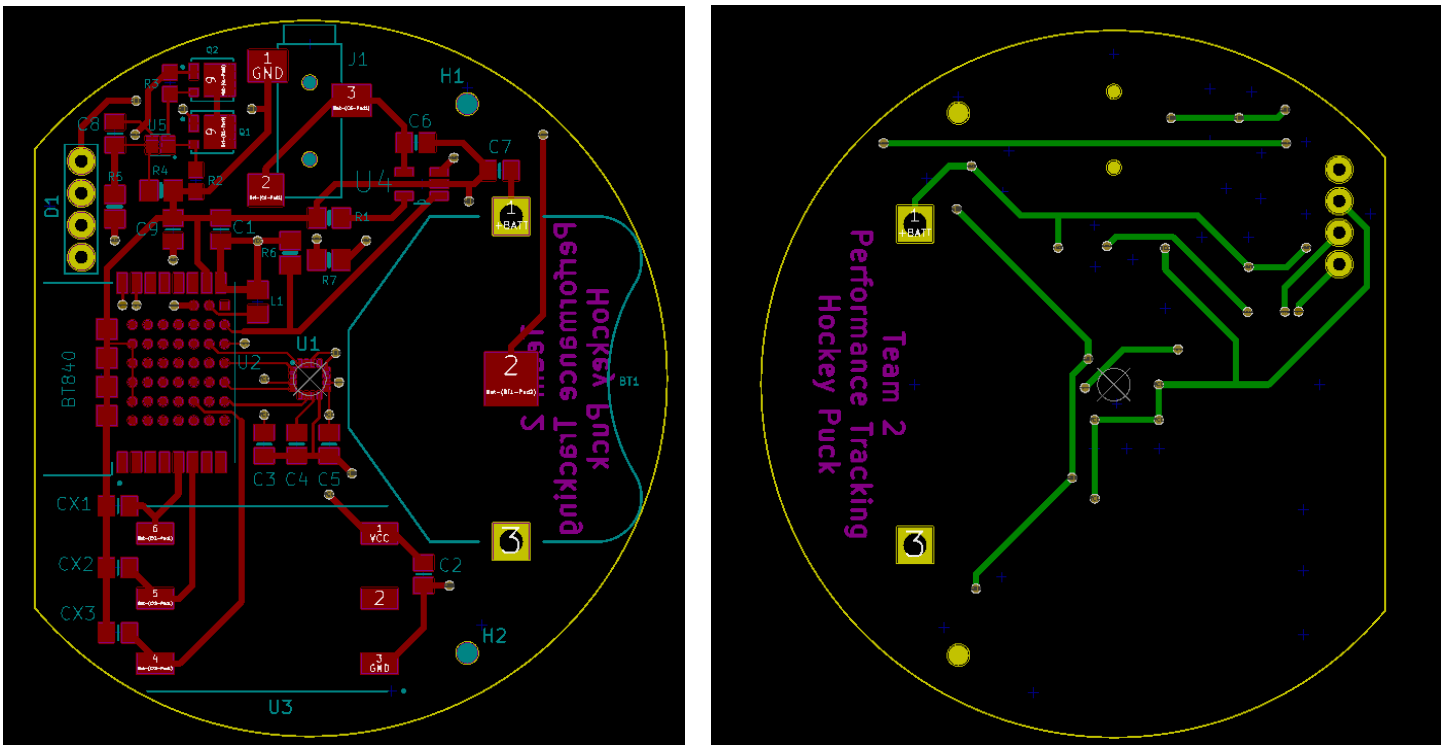

Figure 5: Top and bottom views of the PCB layout

Once the design was completed and reviewed, the Gerber/drill files were generated and provided to ALLPCB, who manufactured the boards with the copper traces, vias and pads.

Assembly

In addition to the unpopulated PCBs, a stencil was also ordered from ALLPCB. This made it possible to accurately apply solder paste of appropriate thickness to the pads. Then, the board was populated with all the components, except for the isolation resistor (R7), the inductor (L1) and the accelerometer (U3). The board was placed in a reflow oven and underwent a complete reflow cycle of appropriate temperatures for all the components. Finally, the board was checked for any errors and continuity checks were performed. The accelerometer was soldered manually (as recommended by the manufacturer) and remaining components were soldered using an air gun once the battery circuitry had been tested.

2.3.2 Microcontroller and Wireless Communication

Microcontroller Program Structure

Bluetooth Services and Communication

In order to transmit the data wirelessly from the puck to the user, we had chosen to use the newest Bluetooth technology, Bluetooth 5 Low Energy (BLE). The microcontroller we chose in Phase 2 was the nRF52840, made by Nordic Semiconductor. This microcontroller was used to both transmit and receive the data, as it is embedded in the BT840F module on the PCB as well as the nRF52840 dongle that the user connects to the computer.

According to BLE specifications, there are two device roles in BLE, peripheral and central. Peripheral devices can connect to and transmit data to one central device. In our product, the puck was the peripheral device and the PC connected to the dongle received the data being transmitted from the puck.

The Generic Attribute Profile (GATT) is the way in which BLE devices transmit data using services and characteristics, within profiles. A profile is a collection of services, which in turn are a collection of characteristics. Services break up different types of data and define unique functionality. Characteristics define a single data type. Both services and characteristics are defined by a unique ID (UUID). The central BLE device can receive data from many characteristics and distinguishes the data by UUID. There

are services that have already been created by the Bluetooth Special Interest Group (SIG) standards organisation and can be used by the public. Additionally, custom services can be created by BLE users.

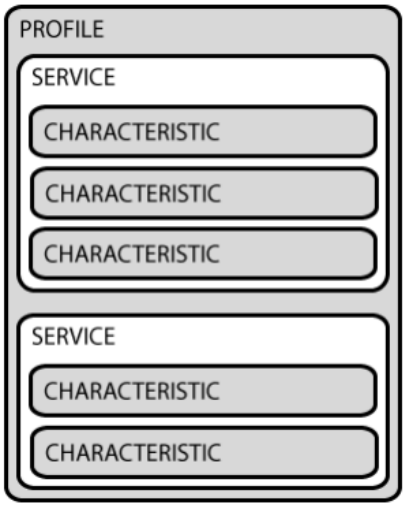

Figure 6: GATT Transaction Hierarchy [1]

Note:

Only the peripheral device (puck) was programmed while the central device (dongle) was simply used as a receiver. Therefore, all subsequent references to the program refer to the peripheral side.

Our project consisted of three services:

1) Battery Service

The battery service is a GATT service defined by the SIG and was used in our puck program to read the input voltage on the PCB, VDD. The service characteristic is comprised of the battery level characteristic which is read by our central device to determine battery level. The service returns the battery level in percentage.

2) SAADC Service

The SAADC service is a custom service written by Team 2 to make use of the SAADC (successive approximation analog-to-digital converter), which is the ADC converter used to sample the analog accelerometer. The sampling rate was set to 2 kHz, as will be explained in section 3.

This service is composed of a characteristic named “SAADC Values”. Writing a value of 1 to this characteristic enables the ADC and samples the accelerometer, while writing a 0 to SAADC Values stops sampling the accelerometer.

3) Nordic UART Service (NUS)

The Nordic UART service (NUS) is a custom service created by Nordic Semiconductor. It composed of two characteristics, namely RX and TX, which are used to receive and transmit data over BLE. The data collected from the SAADC service as well as the battery service are transmitted from the puck to the central PC using this service.

Program Sequence

The steps followed by the program are as follows. When the program first starts, it initializes all of its services and then advertises that it is ready to connect to the central device. Once the central device initiates the connection, the program is then ready to be run. On the central side, notifications are enabled automatically by the user interface, allowing for the user to receive the data.

The program is then controlled by the central user interface. Writing a value of 1 to the SAADC Values characteristic starts the sampling and writing a value of 0 stops the sampling.

The user can then disconnect from the puck. If there is no connection initiated within 60 seconds, the puck enters low power mode, where it can then be woken up. Upon wake-up, the program resets and restarts once again.

Transmission of data

There are two types of data that we are transmitting, namely the battery level and the ADC value from the accelerometer. Once collected, the data was sent in the form of notifications from the NUS service to the central PC. In order for the central device to connect to the puck, the puck first advertises that it is ready for connection. The central device then connects to the puck through the user interface and the two devices are then paired.

Enabling the notifications on the NUS service allows for the data to be transmitted. When the SAADC service is enabled, the battery service is disabled and vice versa. For the battery, the battery level

is sent in a 2-byte packet holding the battery percentage. For the SAADC data, the raw data collected from the accelerometer is sent, and will be interpreted by the user interface. The calculations for interpreting the data will be further explained in this section.

Analog to Digital Conversion (ADC)

As mentioned in the SAADC service section, we are using a custom service to convert the analog readings from the accelerometer and to send them digitally to the user. In order to provide maximum accuracy in the measurements of the sampled voltages from each axis of the accelerometers, a 10-bit ADC resolution was used.

The range of input voltages for the ADC is 0-VDD (0-3.3 V) as a result of the internal regulator of the input voltage to the MCU. The maximum voltage the ADC can convert is therefore 3.3V, and a 10-bit resolution results in 2^{10} = 1024 possible values over this voltage range. The size of the step is given by

$$\frac{3.3V}{1024} = 3.222\ mV$$

Our accelerometer has a sensitivity of 2.5 mV/g, therefore a 10 bit ADC resolution provides measurements within ±1g of the actual accelerometer measurement.

Although the ADC is not exact, it meets our initial specification of acceleration tolerance within ±1g. Also, for the purpose of the MIHRG's research, they are interested in high-g situations such as a slap shot, where g-force is in the range of 100-500g's, and where small deviations of ±1g will not impact the results obtained.

Sleep and Wake-up

To save power when the puck is not being used for testing, the MCU goes into sleep, or low power, mode where it consumes minimal current. The puck will enter its low power mode when it remains disconnected from the user interface for 60 seconds. In order to wake the puck up, the user must simply strike it vertically on a surface. This was achieved by setting a low power comparator (LPCOMP) on the analog pin connected to the z-axis of the accelerometer. LPCOM compares the input voltage on the pin to a reference voltage. The reference voltage we set was VDD*9/16.

Striking the accelerometer excites it and produces a voltage from 0.4V to 2.9Vfor a range of ±500 g's of force. Depending on the state of charge of the battery, VDD will change and the force needed to wake up the puck is in the range of 70-85 g's.

Upon a wake-up from sleep, the program resets and the puck begins to advertise again. The user can once again connect to the puck within 60 seconds of waking up before it goes into low-power mode once again.

Voltage Regulation

The MCU is supplied by a battery with a nominal voltage of 3.6V. The maximum MCU input voltage (VDD) is 3.3V, therefore a register was set in the MCU to regulate the voltage at 3.3V. When the battery discharges and the voltage drops below 3.3V, the voltage regulator output follows the battery voltage until the battery is cut out by the protection circuitry at 2.8V.

Battery Level Definition

The battery level is transmitted from the puck to the central device based on the following calculation:

$$BAT\_LEVEL\_IN\_PERCENT = 100 - \frac{(3200 - BAT\_LEVEL\_IN\_mV) * 100}{400}$$

Any value above 3200 mV was defined to be 100% because of voltage the regulation of the input voltage to the MCU (VDD). The percentage does not correspond directly to actual battery voltage levels since we are sampling VDD which is regulated at 3.3V (nominally around 3.26 V). The equation is simply used as an indication of when the battery drops below 3.2V and should be charged.

Calculations

In order to convert the SAADC values transmitted by the NUS service into a voltage (and then a g-force measurement), the following calculation procedure was used in the User Interface code before saving the data to a CSV file:

- SAADC produces an integer from 0-900 representing excitation voltage (ADC_VALUE) produced by the accelerometers

- The accelerometer data can be converted into a voltage by

$$ADC\_RESULT\_IN\_VOLTS = [(ADC\_VALUE) * 600 / 1024 * 6 / 1000]$$

With constants:

600 -> Reference voltage (in mV) used by ADC while doing conversion.

1024 -> 10-bit ADC resolution

6 -> ADC is configured to use VDD with a 1/6 pre-scaling unit, so multiply by 6

1000 -> Conversion from mV to V

- To convert the battery level from percent to volts:

$$BAT\_LEVEL\_IN\_V = \frac{3200 - 400 + 4(BAT\_LEVEL\_IN\_PERCENT)}{1000}$$

- From the accelerometer datasheet, this equation is given to calculate g-force (using the last transmitted battery level). 400 is

$$G\ force = 400 * (ADC\_RESULT\_IN\_VOLTS - \frac{BAT\_LEVEL\_IN\_V}{2})$$

2.3.3 User Interface

A Graphical User Interface (GUI) for Windows 10 desktops was developed to accompany the puck. The main purpose of the interface was to allow the user to control the puck's functionality. This included connecting to the puck, indicating when a shot begins and ends, reading information from the puck, displaying this information of the screen and saving readings to CSV files

The user interface is designed and styled using HTML and CSS, while the functionality was written in JavaScript. This structure is common to web application technologies. The JavaScript drivers for the puck

are provided by Nordic Semiconductors. They run on Node.js which is a command line runtime. In order to make a native desktop application, Electron framework was used to run parallel to the Node.js drivers to allow the creation of a GUI window. The desktop application will save a CSV file with all the samples of the recorded shot. Each shot generates a new CSV file.

GUI Design

An overview of the GUI is shown in Figure 7.

- **(1) Home button:** Reloads the page.
- **(2) Help button:** Opens the User Manual that is attached in the Appendix of the report. It is saved as a PDF file.
- **(3) Trial name label:** Label identifying the trial name. The text shown here is also what the name of the CSV file will be when saved.
- **(4) Connect button:** When pressed, the desktop application will scan for an advertising signal from the nRF52840 microcontroller. The text displayed in the button will change to "Scanning…" to indicate that the application is searching for the advertising signal. An adapter listener is created with this button which allows Bluetooth communication. The button must be pressed before any transmission takes place.
- **(5) Disconnect button:** Disconnects the desktop application from the microcontroller in the puck. This button closes the adapter listener created by the connect button.
- **(6) Connection status:** This status bar indicates when there is a connection between the nRF52840 dongle and the nRF52840 microcontroller. The status bar is red when the desktop application is not connected to the puck and the status bar becomes green once a connection is established.
- **(7) Battery level status:** This status bar indicates whether the battery level is high or low. The status bar is blue when a reading has not yet been made, green when the battery level is above 20 percent and red when the battery level drops below 20 percent. Note that the battery level is updated in 10 second intervals.
- **(8) Shot overview section:** This section offers a summarized view of the most recent shot. It displays the peak acceleration in G force, the average velocity in meters per second and the contact time in seconds of the last shot. Note that these values are meant to offer a quick

summary of the last shot. The intent when designing the puck was to gather as much information as possible, so analyzing the complete raw data in the CSV file offers a deeper understanding of shot taken.

- **(9) New Trial button:** This button opens the window shown in number (12). A new trial should be started when the user wants to rename the CSV files.
- **(10) Start and Stop buttons:** The start button is pressed before the shot is taken and the stop button is pressed once the shot has ended. The buttons remain disabled until the desktop application connects to the microcontroller in the puck. The start and stop button alternate between enabled and disabled states depending on whether the microcontroller is sending data.
- **(11) New Trial window:** This window appears when the New Trial button (9) is pressed. The associated CSV file will be the name entered in the text box with a number appended to it. For this reason, special characters such as asterisks and slashes are not accepted in the text box. After each registered shot a new CSV file is created, so naming trials is a useful feature.
- **(12) Save changes button:** Confirms the name of the trial and CSV file. User can confirm what text was entered after the window closes in the trial name label (3).

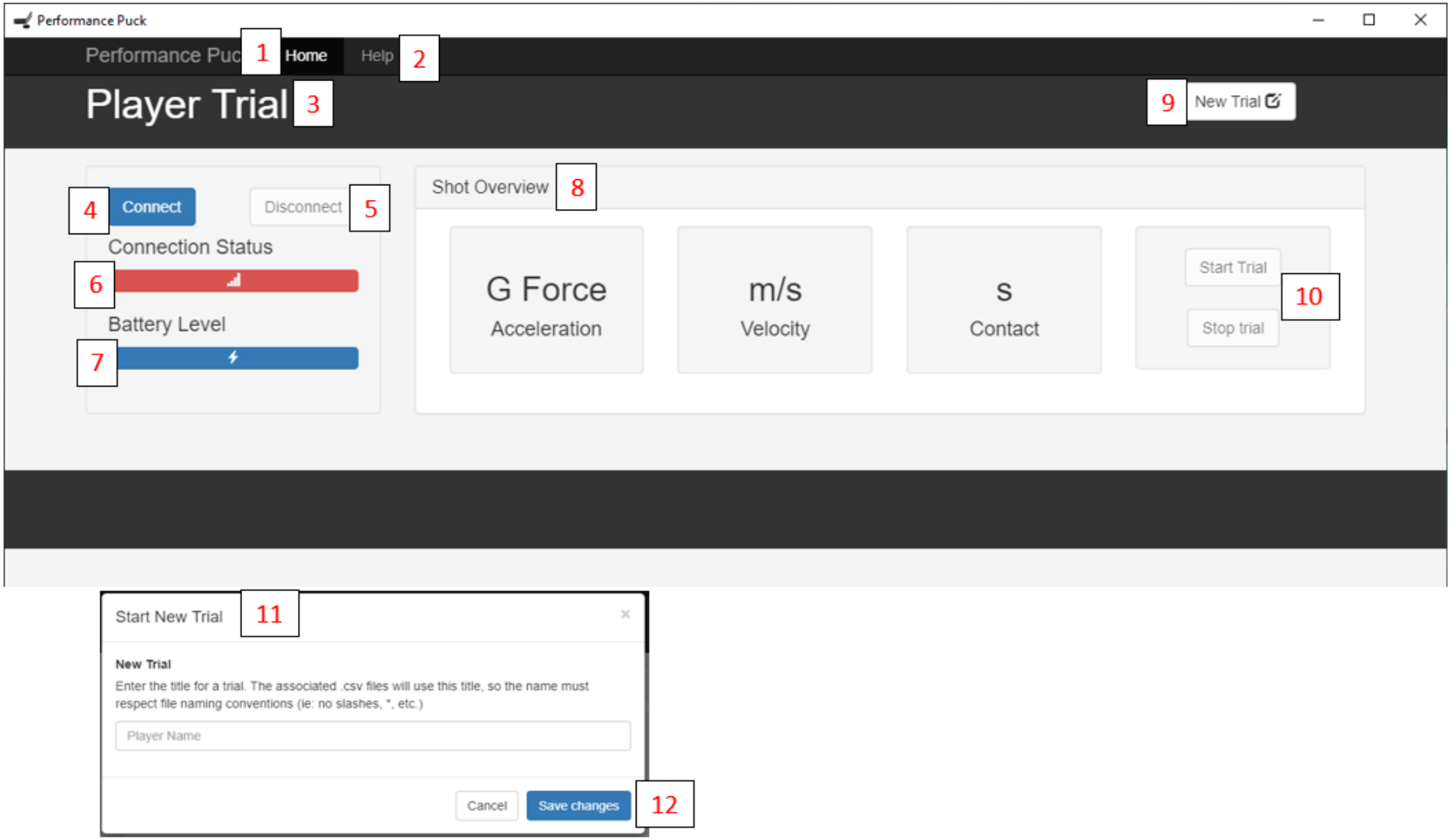

Figure 7: Graphical User Interface

2.3.4 Mechanical Assembly

The first prototype of the puck's exterior consisted of a 2.25-inch diameter hole milled in the center. Originally, we wanted to place our PCB in the complete center of the puck to conserve its center of mass, which would be at 12.5 mm deep. However, it was discovered that it would be a problem removing that much material, as it would cause the puck to be too flexible and less shock absorbent. Due to this issue, we milled a hole only 9 mm deep into the puck. The first concern with this design was the PCB was too close to the top and could be more susceptible to damage on impact.

In an attempt to solve this problem, a second design was brought to attention. This design consisted of making an incision in the middle of the puck, cutting it in half, then milling a hole on both halves of the puck, to allow us to place the PCB in the center. Both halves would then be screwed together. This is shown in the diagram below:

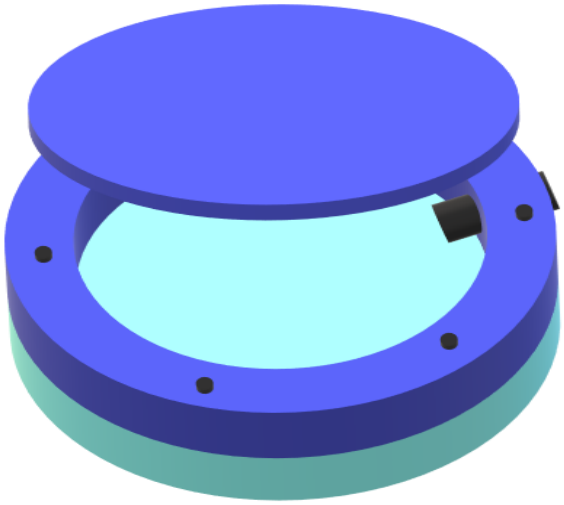

Figure 8. Alternative Puck Design

Similar to the first design, the mechanical department was not keen on acting on this design, as the same issues of removing too much material and the increased flexibility would arise once again. Therefore, we decided to keep working with our first prototype's design.

The next step was accommodating the PCB inside the puck. Two small holes were drilled into the base of the milled area to allow the board to be screwed in. Furthermore, two small areas were milled a little deeper to make room for some solder paste keeping the circuit's battery holder in place. Finally, one final hole was drilled into the side of the puck, that would act as a charging port for the puck's electrical system.

The last step into completing the puck's physical assembly, was milling a slightly larger area on the top of the puck, creating a lip for the cover to rest on. A circular piece of plexiglass will be used as the seal and rests perfectly on the top of the lip area. The glass cover can then be screwed or glued onto the puck, sealing it and protecting our board.

Final Design

Figure 9: Final Physical Assembly

2.4 Issues Encountered and Solutions

2.4.1 Electrical System

A few issues were encountered during the implementation and testing of the electrical system.

- After the initial assembly of the components on the PCB, it was found that our DC power signal was behaving abnormally, since there was a sawtooth waveform with an amplitude of approximately 20% of the signal's level. In addition, the battery protection IC (U5) was not functioning properly. After extensive tests were conducted, it was found that the chip was improperly connected on the board (rotated by 180°). After correcting this issue, the circuit performed as expected.

- Upon removing the battery from the circuit and then re-inserting it, it was observed that the MOSFETS controlled by the protection IC (U5) were not conducting. However, as per the chip manufacturer's datasheet, it may be necessary to short across the two MOSFETS in order to enable the chip's discharge operational mode. This turned out to be the case, and was taken into consideration each time the battery was taken out and put back in. This information is included in the user manual for our product, should the client wish to modify the circuit and remove the battery.

- Since the 2.5mm to USB cable used for charging did not have a datasheet, and was chosen after completing and populating the PCB, it was necessary to re-wire the internal wires of the cable to match the 2.5mm connector's signals with those expected by the charging jack. The cable was cut in the middle, and the outer insulation was stripped back along with that of each internal wire, as shown in Figure 10. Then, the cable was rewired as described in Figure 11.

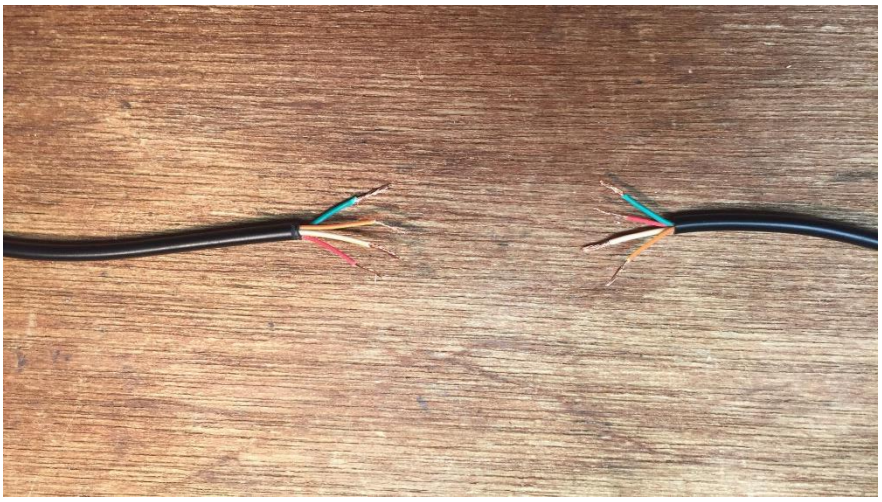

Figure 10: Internal charging cable wires

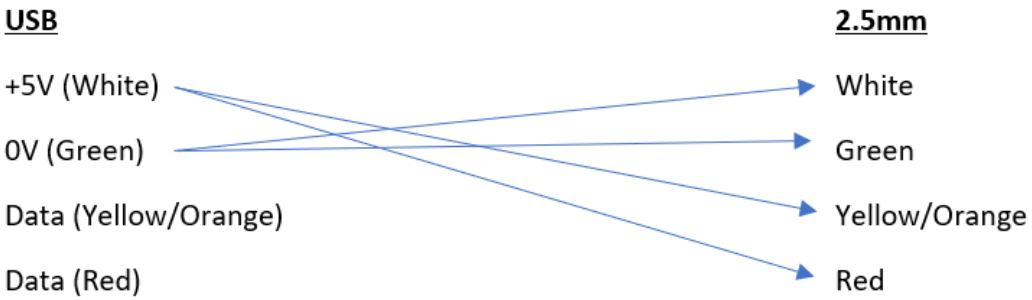

Figure 11: Re-mapping of internal wires

***Note: before performing this modification on a new cable, it should first be verified that the wire colours match the signals described above. This can easily be done by obtaining the specification of a USB type A cable and performing continuity checks for each wire with respect to the connections on the USB port.**

2.4.2 Microcontroller and Wireless Communication

When programming the puck, there we several problems encountered. The problems and their solutions are listed below.

1) Unable to send large packets

When trying to transmit the SAADC data in real-time, we were getting errors showing that the buffers storing the data before transmission were overflowing.

It is specified in the BLE (specifically Bluetooth 5) specifications that the maximum transmission packet length is 256 bytes (247 bytes are available to the user due to overhead), however we were only sending the 2 sampled bytes per axis (6 total) simultaneously, reducing the throughput significantly and hence causing the error.

The throughput needed for transmitting the accelerometer data can be calculated as follows:

$$2000\,\frac{samples}{second} \times 2\,\frac{bytes}{axis} \times 3\,\frac{axes}{sample} = 12{,}000\,\frac{bytes}{second}, excluding\ overhead\ bytes$$

Sending only 6 bytes per notification was not providing us with the throughput needed to transmit in real-time. The solution to this problem was to add a temporary buffer of 240 bytes and fill it with the accelerometer data until it was full. Once it was full, the entire buffer of 240 bytes was sent as one notification instead of having a single notification of 6 bytes.

2) Range not sufficient

Another issue encountered was the lack of range when transmitting data. It was possible to connect to the puck from a large distance but when transmitting, the range decreased significantly. This issue was still present in the Phase 4 demonstration, where we struggled to exceed 5m of range, and we would not be able to meet the requirements outlined for the customer.

After debugging, it was discovered that the transmission power was set to a default value of 0 dB, which was limiting the range. By increasing the packet transmission power to the maximum value of 8 dB, we were able to extend the range to approximately 15m line-of-sight. The increase of 8 dB in power corresponds to approximately 6 times the power, more than doubling the previous range.

3) Crashing of program when out of range

Similar to the previous issue, when the SAADC service was transmitting data and the puck exceeded the connection range, the program would crash and would not be recoverable without removing the battery. The crash was again caused by the overflow of buffers due to the large distance between the user and puck. The transmitted packets were not acknowledged on time by the central device and would therefore be re-transmitted continually until the buffers overflowed.

The solution to this crash was to put the puck to sleep when this occurred, since waking the puck up would simply cause a reset. Recovering from a resource overflow in the buffers requires a reset, therefore putting the chip to sleep avoids crash and the program can easily be restarted once it is in range again.

2.4.3 User Interface

- The original user interface pitched to the client in Phases 2 and 3 was developed using Java. The BLE drivers and API's supplied by Nordic Semiconductor were written for JavaScript. In order to build a GUI for the puck, two options were discussed at this point: use the Nordic BLE API and drivers and build a 'web-like' application or build Java Bluetooth drivers to incorporate with the existing Java interface. A complete understanding of Bluetooth protocol would be necessary to write our own code to communicate with the puck, so the final decision was to use the Nordic Semiconductor pc-ble-driver. These drivers were out of date, which led to other issues.

- The BLE drivers and API's were designed for Node.js command line use, not for making a GUI. The workaround solution to this was to use Electron framework to create a window and GUI. Nordic Semiconductors had tested an older version of Electron. Electron has had many updates in the past year, resulting in many compatibility issues. Finding ways to run on application resolving the compatibility issues took a significant part of Phase 4. This included recompiling and rebuilding the driver's C++ files, finding the right system architecture of Node and the driver

and finding an older unsupported version of Electron. The complete steps taken to solve this issue can be found in the user manual in the Appendix of the report.

- JavaScript is a scripting language, usually used for website development. For this reason, file system manipulation is not straight forward, as JavaScript applications do not usually make changes to your local drive. This caused some complications with exporting the CSV files. The final solution was to have the user specify a file name within the Puck GUI and always save it to the same location as the .exe file.

- Nordic Semiconductors also has drivers for Python development available and taken into consideration as a possible platform for development. These drivers were also not compatible with the latest version of Python and did not successfully install when cloned from the GitHub repository. For this reason, the GUI was built with JavaScript.

2.4.4 Mechanical Assembly

During the development of the physical assembly, the following issues were encountered:

1. Initially, the plexiglass cover was resting onto the PCB's battery holder. This was cause for concern since impact from the top may exert too much force on the battery. This was fixed by milling an extra layer, creating the lip for the seal to rest on.
2. The original mounting holes to attach the PCB to the puck were too small any screws to hold them. This was solved by widening the mounting hole with a drill.

3. Results Obtained

3.1 Electrical System

The first test which was performed was to ensure that the electrical circuitry worked correctly. Once the PCB was fully populated, continuity checks were carried out all throughout the circuit, which did not reveal any issues. Then, the system was powered, and the voltage was checked at appropriate points. As explained in section 2.3.1, two main problems were discovered. Once these were resolved, the voltages at all points on the circuit were correct and the system was powered properly.

3.2 Accelerometer

Sampling Rate

The sampling rate of the MCU must be sufficiently high to determine the contact time between the player's stick and the puck. The contact time will be the limiting factor for the sampling rate because it occurs over a very short period of time.

From the previous study conducted by the McGill Ice Hockey Research Group, it was determined that the contact time between the puck and the stick was approximately 0.04s and the contact time was found by taking all points above 10 g's. [2] There must therefore be enough points to appropriately distinguish between points over a span of 0.04s and to get a good waveform of g-forces

Using a sampling rate of 2 kHz we would have 80 samples over a period of 0.04s. The following graph of 80 points at a constant voltage over 0.04s was used to illustrate this.

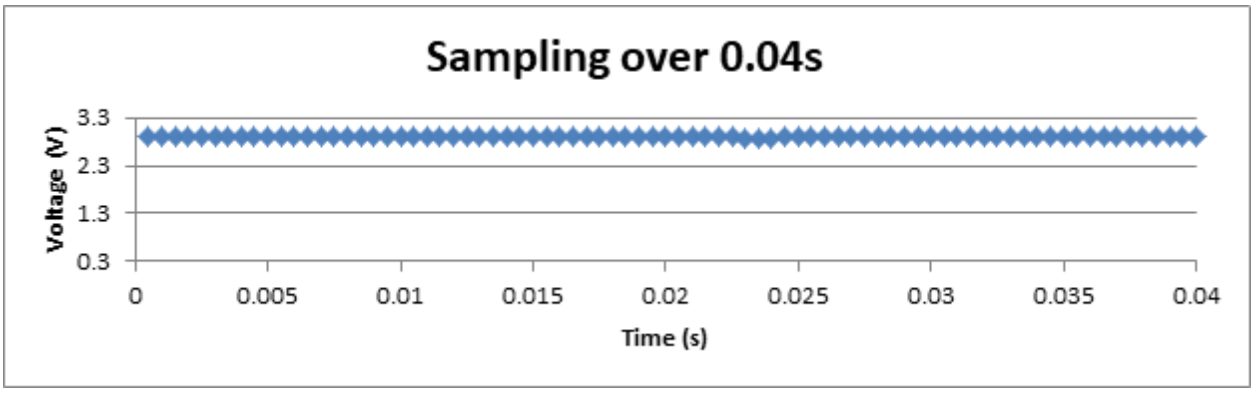

Figure 12: Samples over 0.04s

By observation, it was determined that a sampling rate of 2 kHz provided a sufficient number of points over 0.04s to determine contact time.

ADC Resolution

Shaking the puck manually would test whether an ADC resolution of 10 bits is high enough to distinguish small changes in g-force as measured by the accelerometer. We manually shook the puck in the y-axis for 4 seconds to see the range of g-force values obtained.

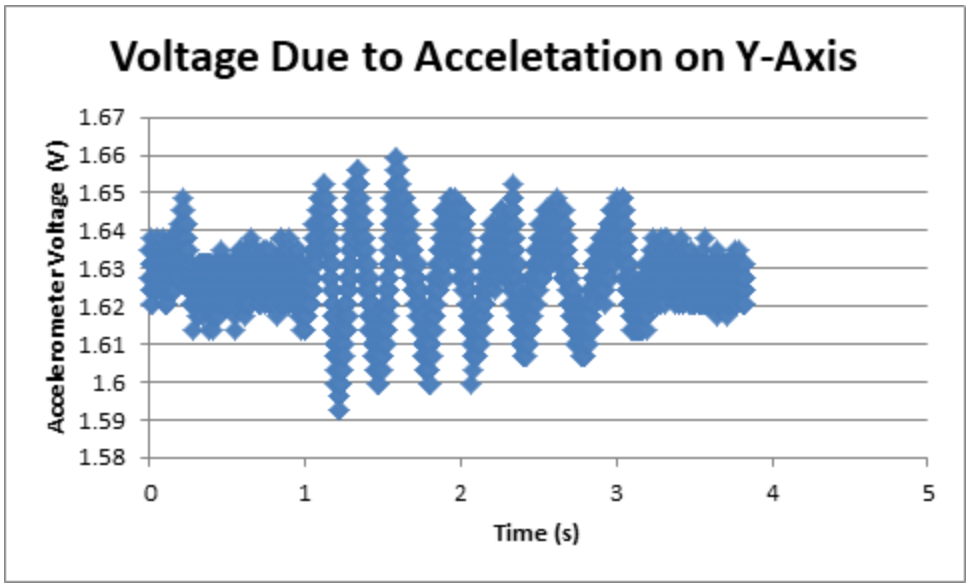

Figure 13: Voltage measured from accelerometer

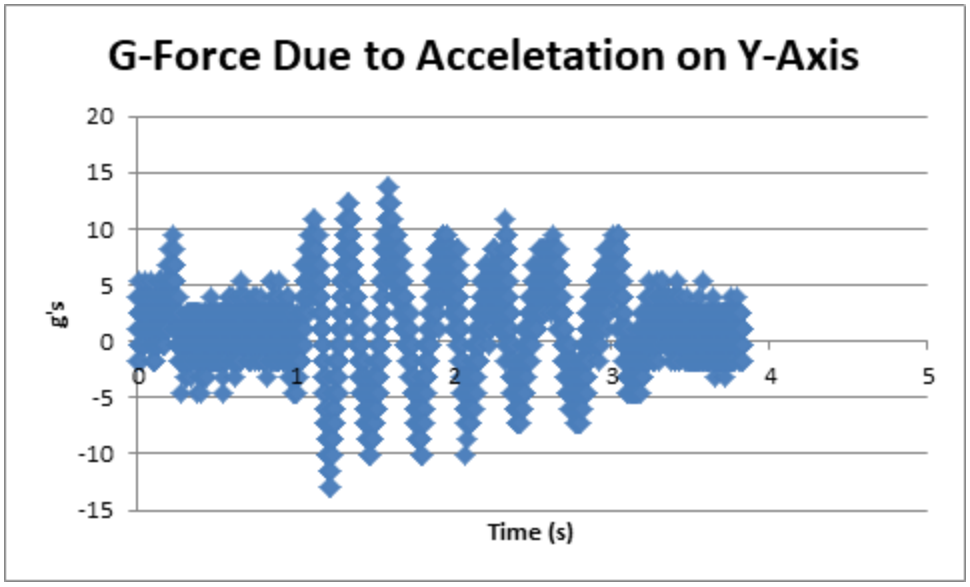

Figure 14: Calculated g-force from voltage

Observing the range of g-force values in Figure 14 we can see that the ADC resolution is sufficient to detect very small changes in g-force. Note that the negative g-force values represent forces in the negative y-axis direction.

Full range of g-force

The next test consisted of verifying that the accelerometer was capable of measuring the full range of values that would be relevant to study (i.e. up to 500 g's). Striking the puck vertically on a table would produce enough g-force to simulate high acceleration without actually shooting the puck with a stick (since the components were not yet properly secured). As we can see from Figure 15, the measured acceleration was approximately 235g. Therefore, our device is able to measure high values of acceleration. However, in the future it will be necessary to perform slapshot test once the puck is ready on the structural level.

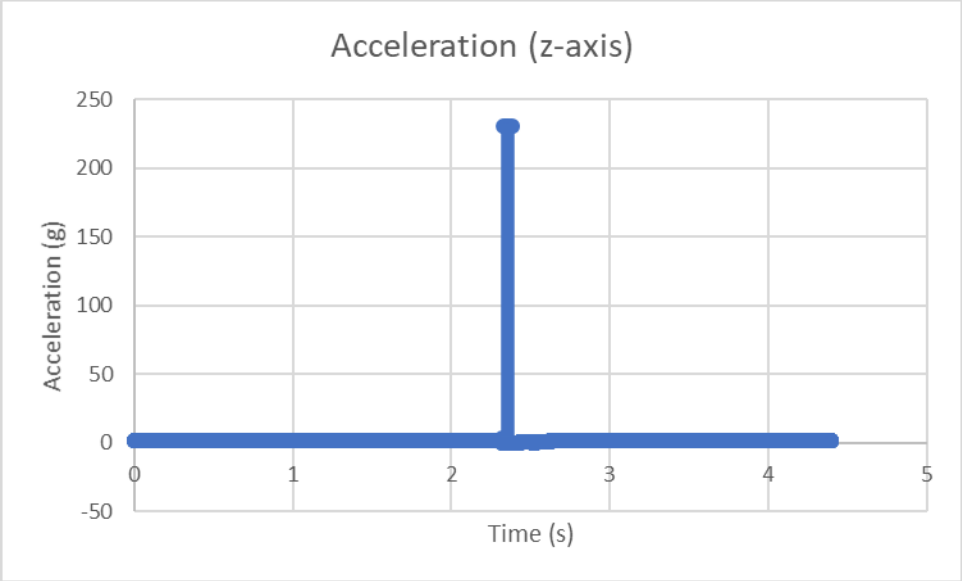

Figure 15: High vertical acceleration test

3.3 Gyroscope

Unfortunately, we were unable to complete the implementation of the gyroscope sensor. All the required connections on the PCB are already in place, but the required code for SPI communication with the microcontroller is not yet written. Therefore, although it was not possible within the time frame of our academic project, this is still a feasible task for further development.

3.4 Wake up from lower power mode

The user will have 60 seconds to connect to the puck through the UI before the puck goes into sleep mode, where it consumes minimal current. In order to wake the puck from sleep mode, the user must strike the puck vertically in the z-direction with sufficient force to excite the accelerometer to VDD*9/16, which translates to approximately 70-85 g's of force, depending on the state of charge of the battery.

To test this, we disconnected the puck in the UI and waited 60 seconds until its Bluetooth service was no longer advertising. We were then not able to discover the puck or connect to it. By striking it on a table, we were again able to see that it was advertising again and we successfully connected to it.

3.5 Battery life

A test was conducted in order to obtain an estimate of the system's battery life. When the battery was approximately 40-50% charged (~3.4V), the system was started and was made to continuously sample and transmit at 2 kHz. After 15 minutes had elapsed, the battery voltage had decreased to 2.8V and the system went into low power mode. As batteries have better performance at full charge, it can be estimated the battery life for continuous use if at least 30 minutes. Therefore, assuming there is generally a delay between trials when the puck is typically used, the expected battery life can be at least 1 hour.

3.6 Charging time

A rough estimate of the battery charging time was obtained by recording the time required to charge the battery several times. In theory, the fully depleted battery should take around 3.2 hours to charge, since it is rated at 160mAh and the charging current is 50mA. This roughly corresponded to what was observed, since for each test, the battery was very close to being fully charged before 4 hours had elapsed. However, it was not possible to obtain an exact charging time, since the charging status pin (indicating full charge) on the controller chip (U4) had not been configured with the microcontroller. In future, this can be done to obtain a better estimate of the required charging time.

3.7 Battery protection

The battery life test described previously also allowed us to verify the undervoltage protection function. After continuously sampling and transmitting at 2 kHz, the battery voltage eventually dropped down to 2.8V, at which point the discharge protection MOSFET opened and prevented the battery from further discharging to the system, confirming the expected behaviour.

3.8 Wireless Range

The range of the wireless communication with the microcontroller was tested by continuously sampling and transmitting in real-time at 2 kHz from the puck and gradually increasing the distance between the receiver (USB dongle). This was performed in an environment allowing for line of sight path with minimal interference, similar to that of the McGill Ice Hockey Research Group's lab. As described previously, the initial range was very short (only a few meters). However, by increasing the transmitting power, it was possible to increase this range to around 12-15 meters. Although this is not as large as originally required (35 m), it is still enough to allow the research group to perform trials in relative proximity to the hockey net.

4. Feedback on Lessons Learned

This project has provided us with invaluable experience when it comes to working on complex technical team projects, on a technical level as well as a teamwork and organization level.

- We learned that without already having the required knowledge or experience, it is essential to do enough research to ensure that the design choices are definitely the correct ones. For example, our chosen wireless communication method was Bluetooth with long range. However, we later discovered that Bluetooth is not suitable long-range communication for high throughput.
- This project has demonstrated the importance of design the product with the entire system in mind, ensuring that there are no missing links between components. For example, we had designed the puck for Bluetooth communication with the possibility of storing data temporarily before transmitting. However, since the chosen microcontroller did not have any EEPROM memory on board, we were unable to explore this option within the time frame of the project, as the PCB was already complete.

- Communication between team members is crucial to the success of a project. Not only does it help ensure a good design from the start, it also allows the team to identify areas of the project that unexpectedly require more effort than expected, and to shift the focus towards them.

5. Conclusions

During the final phase of the project, the implementation of the Performance Tracking Hockey Puck was completed. This followed the project definitions established in phase 1, the initial design work done in phase 2, and the development work performed in phase 3. The electrical system, microcontroller code, wireless communication, graphical user interface and mechanical assembly were all developed to produce a complete system. After resolving any issues encountered along the way, the product was tested in order to determine its level of compliance with the originally established requirements and constraints. Although there is still work to be done to completely satisfy some requirements, we have been successful in providing the McGill Ice Hockey Research Group with a working system that can be perfected for use in their research. Overall, this project has provided all of us with invaluable experience for working on complex engineering team design projects, which will serve us well in our future careers.

6. References

[1] Introduction to Bluetooth Low Energy, https://learn.adafruit.com/introduction-to-bluetooth-low-energy/introduction, accessed March 29th, 2019.

[2] A.Villaseñor, R.A. Turcotte, and D.J. Pearsall. *Recoil Effect of the Ice Hockey Stick During a Slap Shot*. 2006. https://pdfs.semanticscholar.org/70ec/acc51725d210954b1f5ed35109ec2fc52940.pdf, accessed March 29th, 2019.

7. Graduate Attributes: Real-time Technology Assessment

ANALOGICAL CASE STUDIES

Using technology as a way to review plays has become quite common in professional sports today. Examples of these systems are cricket's decision review system (DRS) and soccer's video assistance referee (VAR) and goal line technology. Although these technologies were deemed quite useful, they have been a major source of frustration for supporters for a few reasons.

With this technology, a certain amount of dependency has developed over the years and people believe it will work 100% of the time. There have been a few instances where the VAR miss crucial plays that alter the outcome of the entire match. These incidences include missing an illegal act that should have caused a penalty and missing an offside call to an on-field delay. [1]

To add to the controversy, many people believe that sports should maintain a certain level of simplicity when being played. [2] It can be argued that technology can slow down the pace of any game by a significant amount if a play requires additional review that a referee can not decide on. This can ruin momentum for both teams as well as decrease excitement from the fans. For these reasons, many people, especially of older generations, would like to keep technology limited to eliminate any complications or unfair advantages during competitive matches.

MAPPING CONTEMPORARY PRACTICES

Referring to the ELS(EE) Implications of Engineering Design, we can notice that some aspects are often prioritized over others. The sports industry today is a very successful business as society pays tons of money for game tickets or team merchandise. Because of this, the economical aspect is often the most focused on. Large team franchises will often neglect ethical or social problems if it brings them a lot of money. A problem with this, is that not all teams have an equal amount of monetary assets. This means that a certain amount of technology available to them may differ from team to team, creating possible unfair advantages. This is why focusing on the economical aspect of the rising technologies can compromise any ethical or social standing.

So far, all the technology discussion has been based on competitive matches, however, technology is also on the rise for training as well. If we use our project as an example, coaches can use the

Performance Tracking Hockey Puck to track statistics from each player. This can be controversial because coaches may only use the puck's data to determine the skill level of players instead of actually watching them. [3] This could potentially affect the decision of taking a player on a team or assigning specific roles in the team's lineup. Also, there is always the issue of falsifying data, which is facilitated by this type of technology.

In this case, the participants, which are the players, can be affected because they can be unfairly judged. The utility the puck provides may also cause the user, the coach, to overlook other aspects of the player that would be seen through the traditional "eye test".

PUBLIC PERCEPTION AND EARLY WARNING

This issue may also potentially arise from our performance tracking puck. Currently, the puck is being designed strictly for research purposes, however, the McGill Ice Hockey Research Group has discussed future plans to make this product available to the public. The main customer for this specific puck will be coaches who will use it as a training tool, being able to analyze each player's statistics and also use them for tryouts.

Based on the reaction from fans regarding the case study, we can see that the new era of sports technology is not entirely trusted within the community. Due to certain cases where the camera review systems were deemed unreliable, there was confusion amongst the spectators why decision were being made on behalf of the technological results alone.

For these reasons, we recommend that our puck be used solely for research and training purposes, and not for determining skill level of players or making any decisions for tryouts or role assignment.

TECHNOLOGY ASSESSEMENT AND CHOICE

Currently, the decisions we make regarding the design of the puck are only based on the criteria given by the research group, only for research purposes. Therefore, our prototype will not be available to the public.

If it is decided to make this device available to the public as a training tool, some changes may have to be made to satisfy the ethical and social conditions to avoid all possible controversy or unfair advantages.

We must assure that the technology and purpose of the smart puck will not grant competitive advantages and try to make it as accessible as possible. For the time being, this puck will not be used during competitive matches, therefor only the controversies related to training should be addressed.

REFERENCES

[1] “Why sport’s relationship with technology is a crucial juncture, James Willoughby https://thenewdaily.com.au/sport/sport-focus/2018/05/07/sport-technology/

[2] “Should Technology in Sports be Limited?”, Natalie Proulx https://www.nytimes.com/2018/02/15/learning/should-technology-in-sports-be-limited.html

[3] “Sports Technology has a Problem: End User Programming”, Brad Stenger https://www.sporttechie.com/sports-technology-problem-end-user-programming/

Appendix A – Technical User Manual

1. Setup for User

This section of the user manual outlines the necessary steps to follow before the user can use the user interface and puck.

1.1 Installing nRFConnect for Desktop

Summary

1. nRFConnect is a tool for testing Nordic's Bluetooth Low Energy products.
2. It can be used as a debugging tool in the place of the UI we have created.
3. Also provides the drivers necessary for the nRF52840 dongle.

Procedure

- Download and install here:

  https://www.nordicsemi.com/Software-and-Tools/Development-Tools/nRF-Connect-for-desktop

- Once installed, under "Add/remove apps" tab, install "Bluetooth Low Energy"
- The "Bluetooth Low Energy" app can be used to connect to the dongle (central device), and scan and connect to the puck (peripheral device)
- See documentation online (above link) for how to use

1.2 Configuring the dongle

Summary

- The nRF52840 dongle is used to gather the data transmitted by the puck.

Prerequisites

- nRFConnect installed

Procedure

- Insert the nRF52840 Dongle in a USB port on your computer. The status light (LD2) starts pulsing red, indicating that the Dongle is powered up and is in bootloader mode. After a few seconds, the computer will recognize the Dongle as a USB composite device. The driver needed for the nRF52840 USB DFU feature is also installed.

2. Setup for Developer

To change the software loaded onto the puck, some tools are required. The development environment is SEGGER and the nRF52840 Development Kit (Dev Kit) is used to program the MCU onboard the puck.

2.1 Installing SES

Summary

- SEGGER Embedded Studio (SES) is an IDE used for programming the nRF52840 chip (and hence the puck) using Nordic Semiconductor's software development kit (SDK).
- SES is free to use for Nordic Semiconductor users.

Procedure

- Download and extract zip to desired location (e.g.: C:\SEGGER\)

  http://segger.com/downloads/embedded-studio/embeddedstudio_arm_nordic_win_x64
- Run C:\SEGGER\arm_segger_embedded_studio_v414_win_x64_nordic\bin\emStudio.exe
- Can alternatively create a shortcut to emStudio.exe on Desktop and run that

2.2 Programming the nRF52840 Dev Kit

Summary

- If the program on the puck needs to be changed, it can first be tested directly on the nRF52840 Dev Kit.
- Instead of loading the program onto the puck, it is loaded to the MCU onboard the Dev Kit
- The pins of the Dev Kit can be used to simulate those of the puck's peripherals (accelerometer, etc. See KiCad schematic.)
- Use SES to write and load programs

Prerequisites

- SES installed
- Pull the folder "BLE Projects" from GitHub repo. This contains the SDK as well as the project running on the puck

  https://github.com/David-Hine/Capstone

Procedure

- Under "BLE Projects", SDK version 15.2.0 (nRF5_SDK_Current) can be found. The project solution (ble_puck) is located in the same folder. To run and modify the project, simply open the .emProject file using SEGGER.
- The documentation for SDK version 15.2.0 can be found online or in the attached folder nRF5_SDK_15.2.0_offline_doc.
- Find useful tutorials on how to develop programs on SES and build Bluetooth Low Energy (BLE) applications (attached along with datasheets).

2.3 Programming puck using the nRF52840 Dev Kit

Summary

- Programming the puck also uses SES
- The Dev Kit is used to program the puck
- 5 pins on the PCB must be connected to the Dev Kit in order to program the puck
- Once these 5 pins are connected, programs will be loaded directly onto the puck instead of the MCU onboard the Dev Kit.

Procedure

a. Remove battery from puck

b. Plug in dev kit for power

c. Turn off dev kit using switch

d. Connect the 5 pins on the Dev Kit as shown in the figure below. Cables needed are for VTG, SWDIO, SWDCLK (female to male) and VDD, GND (male to male).

e. Turn on dev kit and make sure all 3 switches are in the correct position (ON, VDD, DEFAULT) as shown in the figure below.

f. Connect the pins to the four pins on the PCB

g. Should now be able to load program the puck using SEGGER

h. Once program is loaded to puck, unplug all wires and insert the battery into the battery holder

i. The last step is to enable battery discharging by shorting (for a brief instant, 1-2s) between the two vias shown in Figure 18 using a thin wire. This must be done whenever the battery is removed and re-inserted, as described in section 2.3.1.

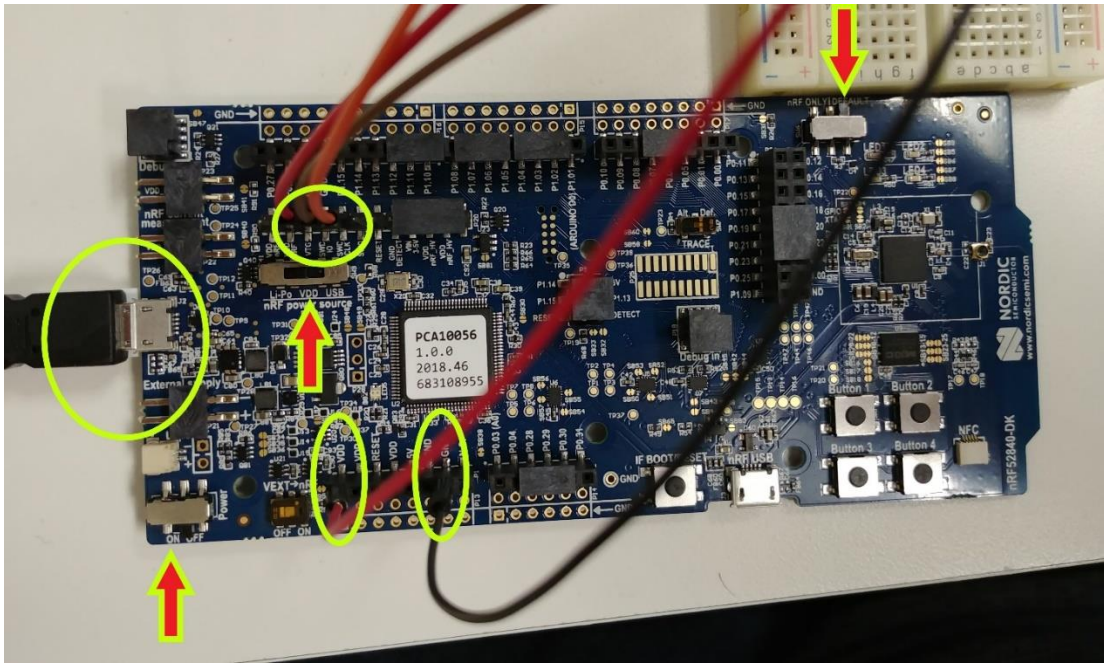

Figure 16: Connections on the Dev Kit for programming puck

Figure 8: Pins on the PCB Figure

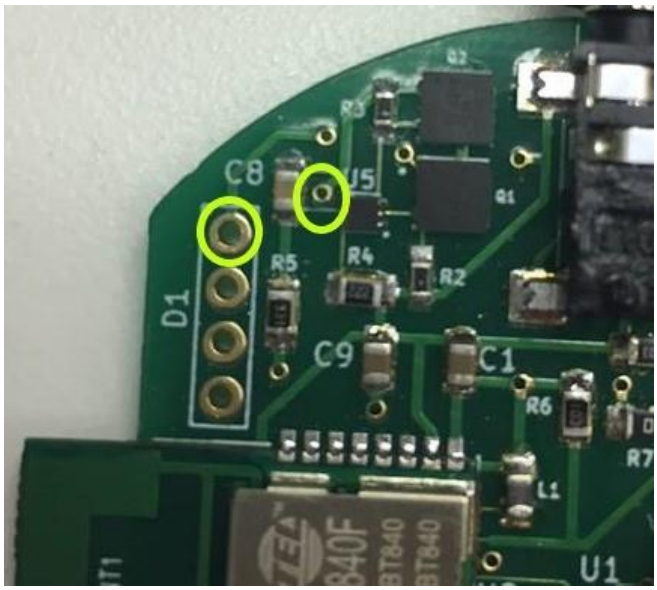

Figure 18: Pins to short

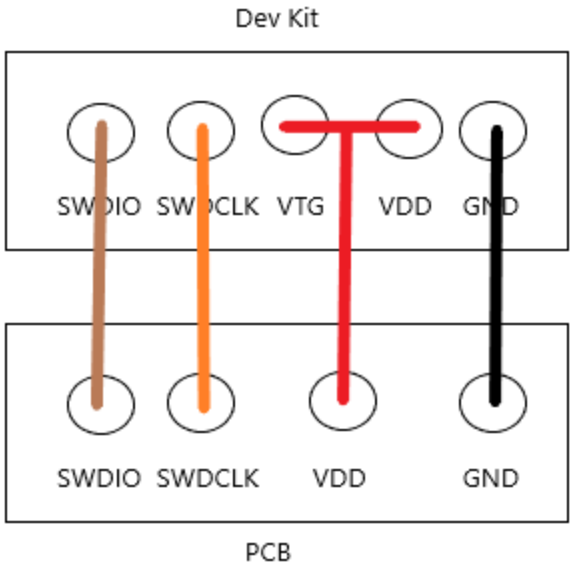

Figure 19: Connections between Dev Kit and PCB (top view of PCB)

3. Using the Puck

Using the puck is very simple. When disconnected, simply strike it on a surface vertically to wake it up. The puck will begin advertising by Bluetooth and can be connected to the UI. Once the puck is no longer needed, disconnect it from the UI and it will go back to it low power sleep mode.

4. User Interface

Summary

There is a graphical user interface that made to control the puck's connection and when it is tracking a shot.

Prerequisites

- Performance Puck .exe file installed on the desktop

Procedure

- Open the PerformancePuck.exe. User will be prompted to select a COM port that the USB dongle is installed to.
- The GUI shown in Figure 20 will load. Ensure that the nRF52840-dongle is inserted in a USB port. Hit the puck to wake it up and press the connect button. The desktop application will scan for the puck for 30 seconds before timing out.
- The connection bar status should update to green when the desktop application connects to the puck. The battery level status will also update over 10 second intervals. Once a connection is established, the puck is ready for testing.
- Press the 'New Trial' button to name your trial. The name entered will be the name of the saved CSV file. Upon confirmation the GUI should reflect the input name.
- Press the 'Strat Trial' button before the shot is taken and the 'Stop Trial' button to end the trial. A CSV file containing all the recorded samples can be found in the same folder as the PerformancePuck.exe file. It is recommended to move the CSV from this folder as soon as it is created in order to avoid cluttering the folder or overwriting the CSV file.
- Also note that once the connection is established, the user can perform as many trials as desired, or until the puck's battery dies. You can rename the trial and take multiple shots without disconnecting.
- When all trials are complete, press the 'Disconnect' button. After 3 minutes the puck will go to low energy mode.
- Another note is that the user interface may not be able to connect, but the timeout is not triggered either. In this case, press the home button to reload the page and restart this procedure until a connection is established.

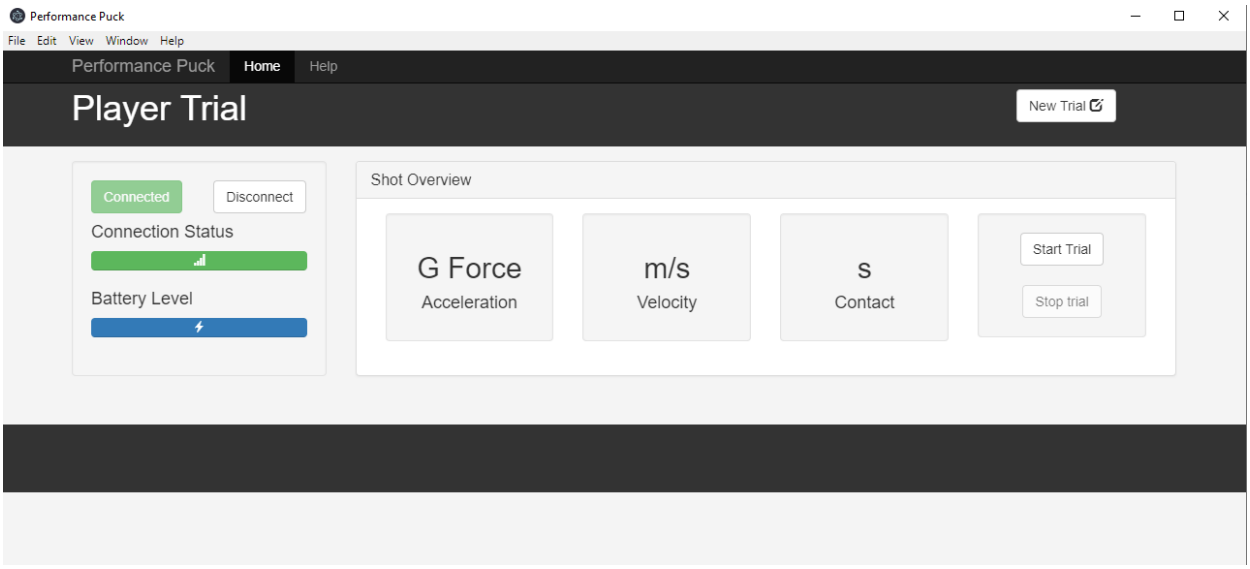

Figure 20: Graphical User Interface

5. GitHub Repository

The bulk of the code used in this project was saved to a GitHub repository for future use. This repository is split into three sections: BLE Projects, JS Code and KiCad.

BLE Projects: This folder contains the C code and headers. This code is used to program the microcontroller. The main code is in the main.c file. Segger embedded studio was used to modify these functions.

JS Code: This folder contains the JavaScript Code and associated HTML and CSS files. The file called puckRenderer,js contains the JavaScript code that controls the functionality behind the UI like connecting to the puck and calculations for exporting acceleration data to CSV files. The index.html file contains the code for the Graphical User Interface. Bootstrap 4.0 was used to style the HTML and is not included in the GitHub repository.

KiCad: The KiCad folder contains the files related to the electrical system design with the KiCAD software. These including the schematic, component symbols/footprints and PCB layout.

Appendix B – Bill of Materials

Table 3 is the final Bill of Materials for the project, consisting of all the components and materials purchased. This includes development tools necessary for designing the puck and spare components in case of any damage. The cost of materials for one puck is approximately $400 CAD. Note that over half this cost is attributed to the accelerometer ($232 CAD).

Table 3: Final BOM

Item	Design Reference	Description	Value	Footprint	Quantity	Unit	Manufacturer	Manufacturer P/N	Supplier	Supplier P/N	Unit Cost	Total Cost
1	BT1	Coin Cell Holder - 24.5 mm	PRT-08863		2	pcs	Sparkfun	PRT-08863	ABRA	PRT-08863	1.33	2.66
2	C1, C6, C7, C9	Capacitor, X7R, ±10%	4.7µF	805	5	pcs	Samsung	CL21B475KPFNNNE	Digi-Key	1276-2972-1-ND	0.39	1.95
3	C2, C3, C4, C5, C8	Capacitor, X7R, ±10%	0.1µF	805	5	pcs	Samsung	CL21B104KCFNNNE	Digi-Key	1276-6840-1-ND	0.16	0.80
4	J1	2.50mm (0.094", 3/32", Sub Mini, Miniature) - Headphone Jack Connector Solder	SJ-2523-SMT-TR		2	pcs	CUI Inc.	SJ-2523-SMT-TR	Digi-Key	CP-2523SJCT-ND	1.38	2.76
5	L1	FIXED IND 10UH 100MA 600 MOHM	LQM21FN100M70L	805	2	pcs	Murata	LQM21FN100M70L	Digi-Key	490-4029-1-ND	0.25	0.5
6	Q1, Q2	MOSFET N-CH 25V 60A 8-SON	CSD16406Q3		4	pcs	Texas Instruments	CSD16406Q3	Digi-Key	296-24251-1-ND	1.84	7.36
7	R1	20 kOhms ±1% 0.125W, 1/8W Chip Resistor 0805 (2012 Metric) Moisture Resistant Thick Film	20K	805	4	pcs	Samsung	RC2012F203CS	Digi-Key	1276-5352-1-ND	0.15	0.60
8	R2, R3	5 MOhms ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) High Voltage Thick Film	5M	603	4	pcs	Ohmite	HVC0603T5004FET	Digi-Key	HVC0603T5004FETCT-ND	3.04	12.16
9	R4	2.2 kOhms ±1% 0.125W, 1/8W Chip Resistor 0805 (2012 Metric) Moisture Resistant Thick Film	2.2K	805	4	pcs	Samsung	RC2012F222CS	Digi-Key	1276-5290-1-ND	0.15	0.60
10	R5	330 Ohms ±1% 0.125W, 1/8W Chip Resistor 0805 (2012 Metric) Moisture Resistant Thick Film	330	805	4	pcs	Samsung	RC2012F331CS	Digi-Key	1276-5246-1-ND	0.15	0.60
11	R6	10 kOhms ±1% 0.125W, 1/8W Chip Resistor 0805 (2012 Metric) Moisture Resistant Thick Film	10K	805	2	pcs	Samsung	RC2012F103CS	Digi-Key	1276-5332-1-ND	0.16	0.32
12	R7	0 Ohms Jumper 0.125W, 1/8W Chip Resistor 0805 (2012 Metric) Moisture Resistant Thick Film	0	805	2	pcs	Yageo	RC0805JR-070RL	Digi-Key	311-0.0ARCT-ND	0.16	0.32
13	U1	MEMS Gyroscope 3 axis 16QFN	ITG-3701		2	pcs	TDK Invensense	ITG-3701	Digi-Key	1428-1050-1-ND	15.65	31.3
14	U2	Bluetooth Modules (802.15.1) nRF52840 Bluetooth 5, Thread, Zigbee module	BT840F		2	pcs	Fanstel	BT840F	Mouser	308-BT840F	20.05	40.10
15	U3	Accelerometer X, Y, Z Axis ±500g 6kHz SMD	832M1-0500		1	pcs	TE Connectivity	832M1-0500	Mouser	824-832M1-0500	232.23	232.23
16	U4	Charger IC Lithium-Ion/Polymer SOT-23-5	MCP73831T-2ATI/OT		2	pcs	Microchip Technology	MCP73831T-2ATI/OT	Digi-Key	MCP73831T-2ATI/OTCT-ND	0.88	1.76
17	U4	Charger IC Lithium-Ion/Polymer SOT-23-5	MCP73832T-2ATI/OT		2	pcs	Microchip Technology	MCP73832T-2ATI/OT	Digi-Key	MCP73832T-2ATI/OTCT-ND	0.91	1.82
18	U5	Battery Battery Protection IC Lithium-Ion/Polymer 6-WSON (1.5x1.5)	BQ29700DSER		3	pcs	Texas Instruments	BQ29700DSER	Digi-Key	296-43985-1-ND	1.05	3.15
19		Bluetooth Development Tools (802.15.1) Dev kit for NRF52840 Bluetooth 5	nRF52840-DK		1	pcs	Nordic Semiconductor	nRF52840-DK	Mouser	949-NRF52840-DK	63.83	63.83
20		Bluetooth Development Tools (802.15.1) USB Dongle for Eval of NRF52840	nRF52840-Dongle		1	pcs	Nordic Semiconductor	nRF52840-Dongle	Mouser	949-NRF52840-DONGLE	13.80	13.80
21		Coin Cell Battery Rechargeable - 24.5mm	PRT-10319		3	pcs	Sparkfun	PRT-10319	ABRA	PRT-10319	4.13	12.39
22		Standard hockey puck			12	pcs	Canadian Tire	083-0175-6	Canadian Tire	083-0175-6	1.42	17.04
23		Single-layer PCB + stencil			5 + 1	pcs	ALLPCB		ALLPCB		88.08	88.08
24		USB to 2.5mm Charging Cable, 3ft			1	pcs	Ancable	CS-JBL-1	Amazon	B071S6JH2M	12.95	12.95
25		USB to 2.5mm Charging Cable, 3ft			1	pcs	Ancable	CA-JBL-1	Amazon	B01KO3UPHU	12.95	12.95
		Total cost (including taxes and shipping & handling)										654.33