Marching Band Assistant
Final Report

Team Number 6

Wynter Chen (wynterc2)
Alyssa Licudine (alyssal3)
Prashant Shankar (shankar7)

ECE 445
TA: Dhruv Mathur
December 9, 2020

ABSTRACT

The Marching Band Assistant (MBA) is a wearable device that is used to calculate a user's conducting tempo via motion detection. The MBA is turned on using a switch within a plastic encasing. When on, the MBA continuously performs calculations based on the user's arm motions. The MBA can be connected to a host computer via Bluetooth. The user can run an executable program that displays the device's latest output in real-time. The user display contains an option to record the data received over time and save it to a CSV file.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Objective

One of the primary responsibilities of a drum major is to conduct a consistent tempo during marching band practices and live performances. However, most high school drum majors are tasked with this responsibility with no prior training. While some high schools and colleges have access to drum major camps to train and practice fundamentals, drum majors often do not have practice tools readily available to receive feedback on their conducting.

The Marching Band Assistant (MBA) aims to create a method for drum majors to practice and analyze the consistency and tempo of their conducting. An arm attachment with an inertial measurement unit on it would be used to record acceleration from the conducting arm. The derived acceleration data processed via a microcontroller would calculate the user's average tempo after a short period of recording, as well as the standard deviation of time between conducting motions. The calculations would be transmitted via Bluetooth to a GUI on a laptop, where the user could both have data displayed in real time and view data afterwards to observe any inconsistencies in his or her conducting.

1.2 Background

A drum major is the leader of a marching band, a common entertainment organization that plays musical numbers at the sporting events of most high schools and colleges. A drum major's responsibilities include relaying vocal commands, communicating with band members, and conducting effectively [1]. Conducting establishes the tempo of a musical number, and the drum major is the only visual source of tempo during performances.

Thus, metronome devices such as the Dr. Beat are used in personal or full-band practices to commit tempos to memory [2]. However, there are issues with using an audio source due to the spaced-out nature of marching bands combined with the relatively slow speed of sound. Additionally, there is currently no way to tell if one's conducting motions are on time without the use of a metronome. Software such as SmartMusic serve a similar purpose to our solution; it receives input via microphone and provides correctional feedback to musicians, but this only extends to singing or playing an instrument [3].

To our knowledge, there is no commercial device that records the movements of a conductor's arms to determine the tempo or consistency of conducting. Personal interviews with Metea Valley High School Band Director Glen Schneider, Marching Illini member Daniel Dresser, and University of South Carolina Drum Major Kelley Powell indicate that the tool would be a valuable asset in high school marching bands and practice environments.

Having a tool where drum majors can record their motions would help them verify if they truly committed a tempo to memory, and if they can keep a consistent tempo over a long period of time. Our tool intends to graphically display conducting data both during and after the recording session ends, so that drum majors can analyze their behavioral patterns and catch inconsistencies. Our expectation is to deliver real-time data with a delay of less than 500 ms. We believe this is an appropriate limit because while the user does not need instant feedback since the long-term behavior of conducting is more useful information than a single beat, the user should be capable of identifying within approximately two measures of music if his or her tempo has changed.

1.3 High-Level Requirements

- The MBA must correctly identify conducting tempos of 80 BPM-160 BPM, with an error margin of +/- 10% BPM.
- The MBA should transmit the tempo calculations wirelessly to a receiving computer and display the latest data with a delay of less than 500 ms.
- The MBA should be capable of wireless usage without charging for at least four hours.

2. DESIGN

2.1 Block Diagram

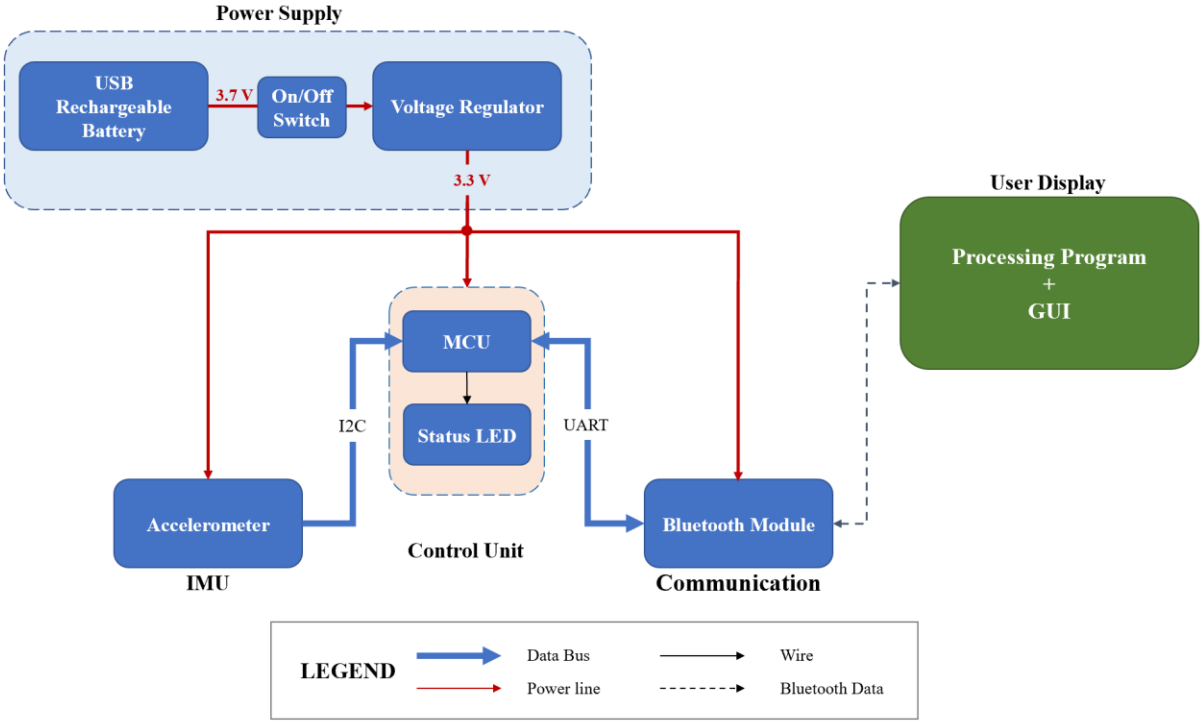
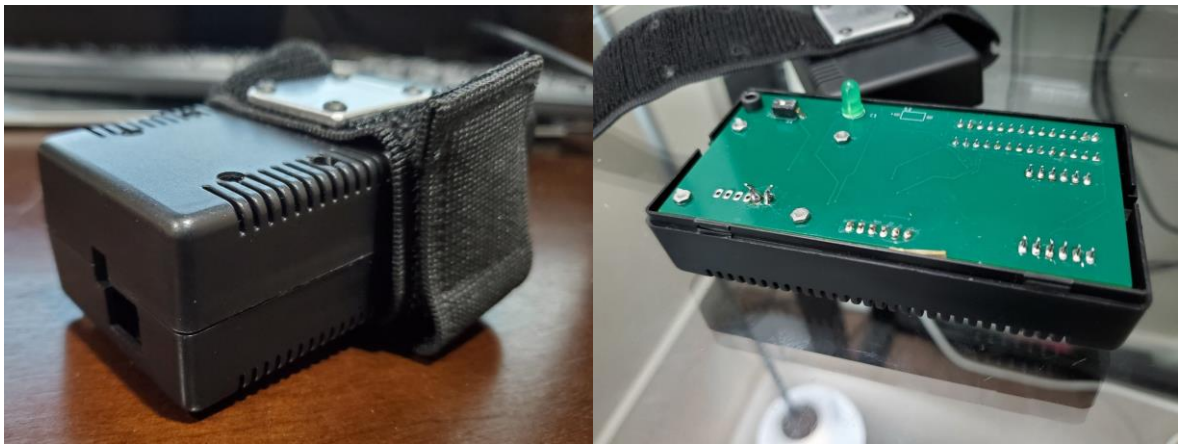


Fig 1. A comprehensive block diagram of the Marching Band Assistant.

The MBA consists of five main subsystems: A power supply, the inertial measurement unit (IMU), the control unit (CU), communication, and the user display. The power supply drives the steady operation of the sleeve, powering all its components for up to four hours, which is the high-end typical length of a band rehearsal. The IMU contains the accelerometer, which captures the acceleration of the user's arm motions. This data is then passed onto the microcontroller (MCU) in the control unit subsystem. The MCU parses and processes the data received from the IMU to calculate the user's average tempo. The processed data is transmitted to a receiving computer via the Bluetooth module. A Java program collects and displays the data on the user's computer via a Processing GUI.

2.2 Physical Design



(a) Outside view. *(b) Inside view with top opened.*

Fig. 2. The physical design of the Marching Band Assistant.

Our design allows the user to slip the one-size-fits-most device on his or her arm, even if thick clothing such as a marching band uniform is worn. The MBA device is enclosed in a plastic box (Fig. 2a). A plastic box was chosen because it created less Bluetooth interference and was more comfortable than a metal box. A velcro strap is attached to the top of the box, and wraps around the bottom of the upper arm. The strap is required to form a tight fit around the arm for the MBA to work properly.

The PCB is placed inside the box (Fig. 2b). Plastic material and a mounting hole is used to fix the PCB in place within the box. The top of the PCB contains the LED and switch, while the bottom contains most of the MBA's subsystem components. The PCB positioning of the IMU (located on the right side of Fig. 2b) further up the arm is crucial for recording accurate data. The battery is connected to the battery board underneath the PCB, and is safely taped to the bottom of the box. Appendices A and B respectively refer to the schematic and PCB design of the MBA.

The MBA device contains a Bluetooth module which sends data to the computer feedback system. An executable program serves as the feedback system on the computer, with options to view data in real-time or observe past data sets. As long as the device is powered on, it will continue to record data and be eligible to connect to Bluetooth-compatible devices.

2.3 Block Descriptions

Subsystem 1: Inertial Measurement Unit

Before explaining the specifications of our inertial measurement unit, we will explain why we are collecting acceleration data in the first place. A fundamental requirement of the MBA is to determine when the user has conducted a beat. A "beat" is conducted when the user stops their arm in a certain place in between motions in order to establish a certain tempo. Regardless of where the arm stops, the common trend is that a beat is observed when there is a sudden decrease in velocity of the arm.

Acceleration data can help determine these sudden shifts in velocity. When an object in motion comes to a sudden stop, the magnitude of acceleration drastically increases, then drastically decreases shortly afterwards since the object is at rest for a non-zero period of time. Therefore, we needed to detect anomalies in acceleration data to determine when a beat is conducted.

The inertial measurement unit (IMU) extracts acceleration information from the user's arm motions. Acceleration along one axis causes displacement on the corresponding proof mass, and the capacitive sensors in the IMU detect the differential displacement. Consumer IMUs such as the ICM-20948 use proof masses for each of its three angular rate axes [4]. The ICM-20948's three sensors are an accelerometer, a gyroscope, and a magnetometer, but we used only the accelerometer functionality of the ICM-20948. We found that the accelerometer collected enough data to detect tempos, and disabling the gyroscope and magnetometer saved power.

Each sensor has a sigma-delta analog-to-digital converter (ADC) that produces digital outputs, with an adjustable sensitivity of $\pm 2g$, $\pm 4g$, $\pm 8g$, or our selection of $\pm 16g$ for the accelerometer. The data outputted from the IMU specifies the acceleration in each axis [4]. A Digital Motion Processor (DMP) computes the data it acquires from the IMU using motion processing algorithms. The data is sent in digital form, utilizing I2C protocol after sampling, digitizing, and packeting the data. The packeted data is 16 bits long. Power is supplied to this module via the analog VDD pin, and its data is transferred to the MCU through its data line pins.

Instead of using the ICM-20948 on its own, we opted to use the SparkFun 9DoF IMU Breakout board (Fig. 3). It contains the ICM-20948, level shifters to regulate and supply voltage to the chip, and two Qwicc connectors which remained unused in our design [5]. The breakout board was usable on both initial breadboard designs and our final PCB design because male header pins could be soldered onto the through holes. The most significant through holes used were "DA" and "CL," which were necessary pins for transmitting IMU data at an appropriate frequency to the ATMega328P.

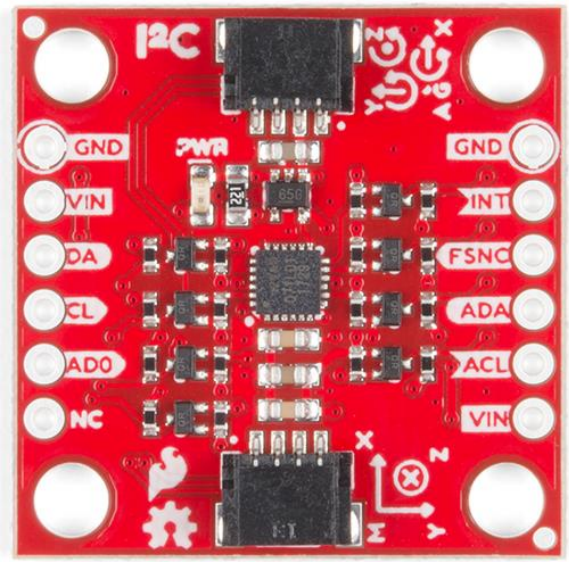


Fig. 3. The SparkFun 9DoF breakout board. The ICM-20948 can be found at the center of the board.

Since the maximum output data rate is 1.125 kHz, the maximum baud rate of the IMU is 1125 kBd. Given that the ATMega328P MCU can run with a baud rate of at least 1 MBd without issue, we had no issues running the IMU at the maximum output data rate [6].

Subsystem 2: Control Unit

The control unit is driven by a microcontroller and interacts with the user through its IMU inputs, Bluetooth output, and an LED light that indicates whether the sleeve is off or on. The microcontroller executes all the interpretation and processing of the raw data from the data registers of the IMU. Its flash storage contains the data forwarded to the Bluetooth module via a Universal Asynchronous Receiver/Transmitter (UART). The computer then receives the final values for the beats per minute to be outputted to the GUI via Bluetooth.

Microcontroller

The microcontroller is calibrated to receive and parse raw data from the IMU's digital motion processor (DMP) in real-time. It is programmed with calculations that filter out noise, perform moving averages of the tempo derived from acceleration spikes, and output the results to Serial.

The ATMega328P (Fig. 4) was chosen due to its flexibility in usage. The chip is able to run on an 8 MHz clock when powered at 3.3V. It is capable of receiving data from sensors via its SDA/SCL pins, communicating with serial ports via its TX/RX pins, and can drive voltage through numerous digital pins. Additionally, its range of baud rates accommodate for the data rate our application needs, which is at least 1.125 kbps. The potential rates of the ATMega328P range from 2400 bps to 230.4 kbps. The ATMega328P has considerable documentation and

application to electronics projects, which proved to be useful for reference when designing and constructing the overall circuit for the device.

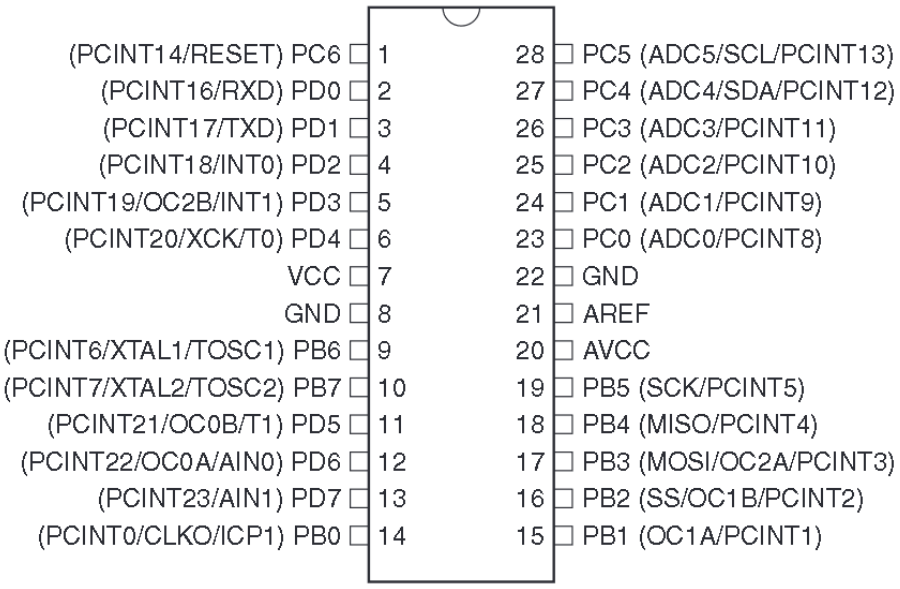


Fig. 4. The ATMega328/P Pinout [7].

In order to use the ATMega328P as a standalone chip on the PCB, we had to bootload the chip so that it was capable of running a program without the help of an Arduino. The process of bootloading involves first wiring a second ATMega328P and clock to pins on an Arduino and running code that makes the chip capable of running independently. Once completed, the first ATMega328P chip is taken out from the Arduino and code can be uploaded to the standalone chip [8].

The ATMega328P runs on an 8 MHz clock as opposed to a typical 16 MHz clock. This is because the ATMega328P on our PCB is only supplied 3.3V, and 8 MHz is a standard clock speed to run at that voltage level since we would need a 5V supply for a 16 MHz clock.

The program for the microcontroller was written using the Arduino IDE. The purpose of the program was to continuously collect data from the IMU, perform calculations that determine whether a beat is conducted, and output the average rolling tempo to the serial port immediately after a new beat was detected. The tempo is calculated by saving a timestamp when the beat was detected, storing that timestamp in an array containing the last 10 timestamps, taking the average difference between timestamps, and using the equation 60000/(avg. difference) to convert from milliseconds to BPM. Storing every timestamp entry from when the device was turned on would result in a tempo averaged throughout the entire session of use, which is not very helpful. However, storing only the latest two timestamps would result in volatile tempo results, when the reality is that conducting a tempo isn't a process that is established in only two motions. We

found through trial and error that storing the latest 10 timestamps painted a reasonable picture of the user's recent conducting tempo.

The PeakDetection library by Github user leandcesar was essential for the code to work. The library uses the principle of dispersion, detecting peaks in rolling data by checking if the newest data is a specified amount of standard deviations away from the rolling average [9]. The algorithm is modifiable, including variables such as amount of rolling samples stored and z-score, which determines how influenced the algorithm is by data anomalies. Finding the correct parameters that suited our needs was largely a subjective process, as when a conducting beat "starts" varies from person to person. Fig. 5 shows an example of how the PeakDetection algorithm is able to detect anomalies in acceleration by outputting a positive value when peaks are detected. For our purposes, we ended up settling on having a 72 sample window, a standard deviation threshold of 1, and a z-score of 0.1, meaning the algorithm is not heavily influenced by data spikes. The rising edge of the peaks is used as the reference of when a beat is conducted.

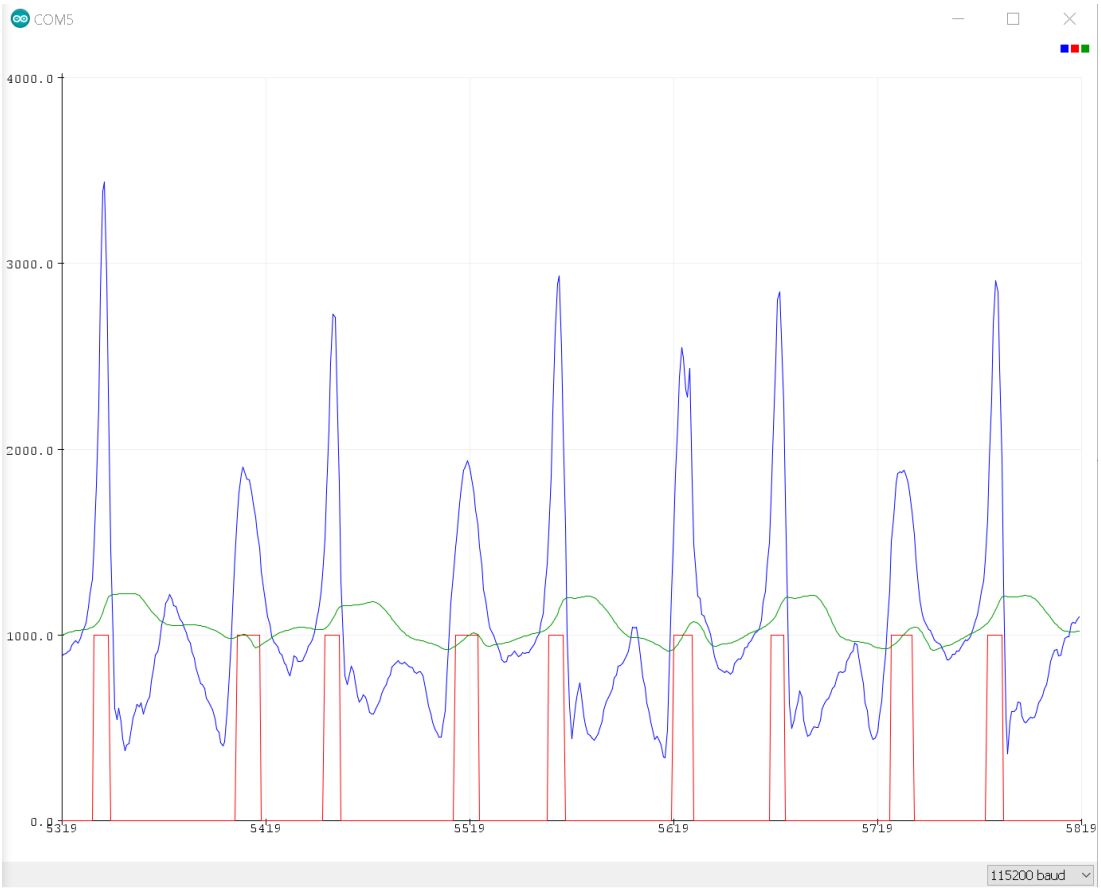


Fig. 5. A serial plotter example of the peak detection algorithm in its early stages. Blue line: magnitude of acceleration, Green line: rolling average of acceleration, Red line: +1000 when a peak is detected, otherwise 0.

Additional measures were taken to ensure that peak detection was not erroneous. We only tested for peaks if the magnitude of acceleration was greater than 1.1g, because it can be safely assumed that any values below that are noise if the device is at rest. Even though the magnitude of acceleration can dip below 1.1g, this only happens well after a beat is conducted, so we found through testing that it is not our concern. Additionally, we did not poll for peaks if another peak occurred less than 150 ms prior. This is because to have the rising edges of peaks 150 ms apart would be equivalent to conducting at 400 BPM. It is extremely uncommon to find pieces of music with a tempo of faster than 180 BPM, so it is safe to assume that new peaks detected in such a short period of time can be attributed to noise [10]. This filtering method significantly improved the accuracy of the tempo algorithm.

Status LED

The green LED turns on or off according to the operation mode of the microcontroller. When the switch is pushed into the "ON" mode with a charged battery, the light will turn on. When the switch is pushed into the "OFF" mode, the LED will not light up because there will be no current supplied to the LED or the microcontroller.

Subsystem 3: Communication

The communication subsystem includes one Bluetooth module inside the MBA attachment. The Bluetooth module allows for wireless data collection and transmission to the computer, eliminating the need for wires that may limit the user's movement. The module communicates with the ATMega microcontroller via UART. Once the Bluetooth module has been connected to the user's PC, the data from the module's TX pin will continuously be displayed on the GUI [11].

The HC-05 Bluetooth Transceiver Module was chosen because its default baud rate (9.6 kBd) is greater than that of the ICM-20948's maximum output data rate, 1.125 kBd. Compared to other Bluetooth modules, it has an expansive collection of documentation on interfacing with Python and its integration with the ATMega328P. These resources proved to be very useful while configuring and debugging the module.

Such resources were used to determine the connections that needed to be made between the HC-05 Bluetooth module and the ATMega328P for stable communication (Fig. 6) [12].

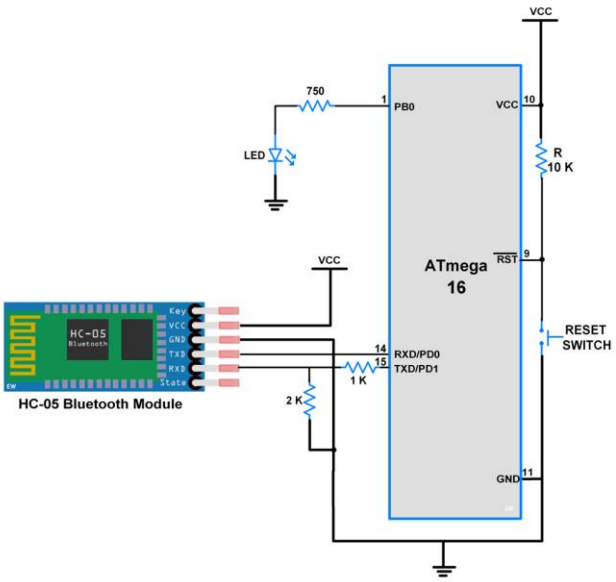


Fig. 6. HC-05 Bluetooth Module interfacing with the ATMega.

The schematic above creates a voltage divider for the TX pin because it assumes 5V will be used to supply the module. Because of our device's voltage regulator, these resistors were not included in our circuit. Every other aspect from this schematic was adopted into our final design.

Subsystem 4: User Display

The user display is our means of displaying the Bluetooth data in a clear and digestible format (Fig. 7). The user display is created using Processing, an IDE specialized in developing GUIs by utilizing Java. Processing enabled us to create an interface that displays the latest data obtained in a specified serial port, updating a meter that displays those values, and providing options to record data as it is received. The GUI is stored in an executable file that can be used without setup.

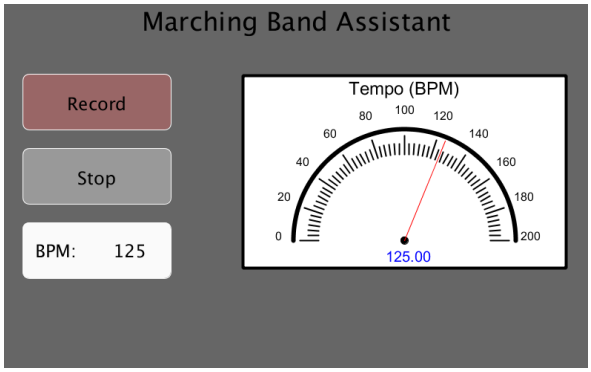


Fig. 7. The GUI display for the MBA.

It is important to note that the “Record” and “Stop” options do not turn the MBA on or off, nor does it stop the MBA from recording tempo data. Tempo data is always recorded while the MBA is powered on; the “Record” and “Stop” options are used to toggle saving incoming values to a CSV file. Once recording is complete, the user can open up a program of his or her choice to plot the data recorded over time (Fig. 8).

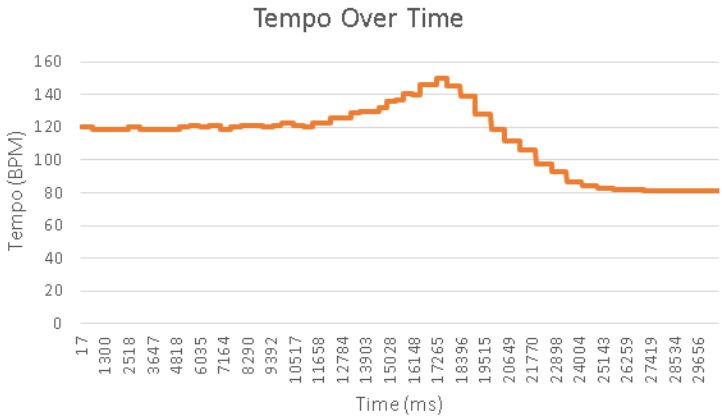


Fig. 8. An example of MBA data recorded over time. The user consistently conducted at 120 BPM, sped up, then slowed down to approximately 80 BPM in the span of 30 seconds.

The user display program polls for new serial port data every 20 ms. If it receives a new numerical value, it updates the meter with that value. Otherwise, the last valid value is pushed to the meter. If no data is received via serial port for 5 seconds, the program briefly disconnects and reconnects to the port to ensure that there are no long-term connectivity issues. This was implemented to fix bugs with serial communication, and it is worth noting that once the device reconnects, every value that was recorded while the device was disconnected is still recorded.

Subsystem 5: Power Supply Unit

The power supply provides the power necessary for steady operation of the IMU, the control unit, and the communication subsystems on the attachment. With the convenience of the user in mind, a lightweight Li-Poly battery that can be recharged via mini-USB on a battery charging board was used. Primary lithium batteries are lighter than other primary chemistries and are suitable for low current applications. An on and off DIP switch was placed between the battery and the voltage regulator to ensure that there is no excess charge on any other components.

On/Off Switch

The on/off switch is the user-control that is used to activate and deactivate the circuit. When the user switches the MBA sleeve attachment on, the whole circuit is activated and the IMU starts collecting data. When the user switches the sleeve attachment off, the battery stops supplying power to the components in the circuit, meaning the MCU, IMU, and Bluetooth module turn off as well.

Power Requirements

The battery must supply enough power to ensure full functionality of the three subsystems simultaneously for at least four hours. The operating voltage and the current drawn by each component were used to calculate the voltage and current delivery requirements of the power supply. The power requirements for all the components on the attachment are calculated in Appendix D. Each component on the sleeve requires +3.3V. The battery at full charge provides at least +3.7V, a voltage higher than what is needed for each module, so a linear voltage regulator is used to scale the voltage down to +3.3V. Because the circuit draws about 300 mA of current at full operation, the battery is estimated to last about 8 hours on one full charge.

Voltage Regulator

Because the IMU breakout board, the Bluetooth module, and the microcontroller require the same voltage inputs of 3.3V, only one voltage regulator is used to satisfy the voltage needs of each component. The LP2985 low-dropout (LDO) and low-noise regulator is used because it has stable behavior despite the output voltage being close to the input voltage value, improving its power efficiency. It is able to output our desired voltage quantity of 3.3V, and its low-noise output ensures that the incoming power supply or transients in the load will not affect the stability of the voltage being supplied to the subsystems [13].

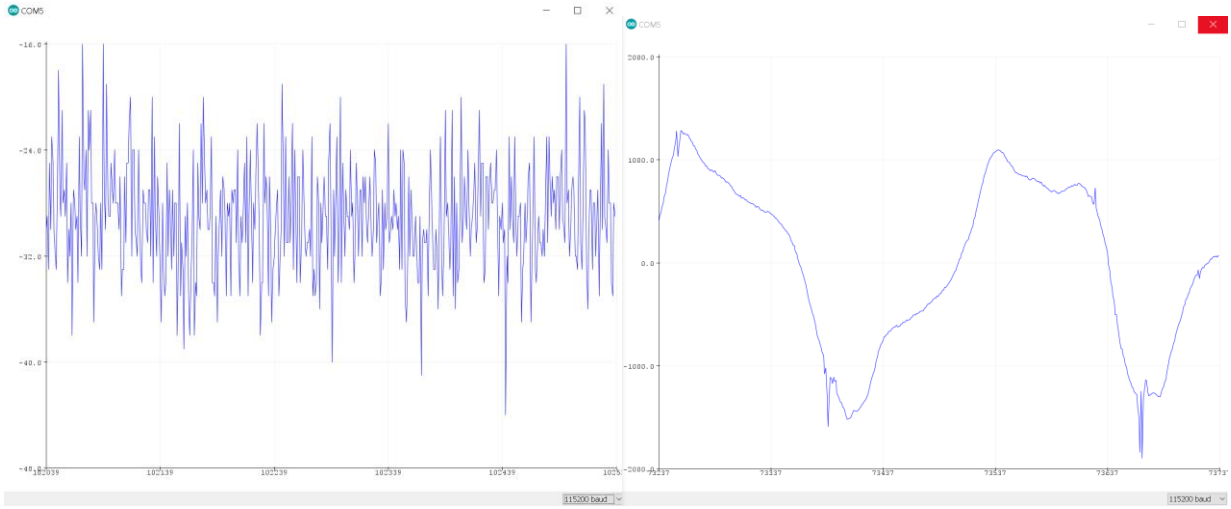
3. VERIFICATION

Subsystem 1: Inertial Measurement Unit

We were able to successfully connect the IMU to our MCU via I2C protocol. Fig. 9 shows an example of retrieving acceleration data in 3 axes through I2C protocol and Serial commands. Our verification process for observing if we retrieved usable accelerometer data in the x, y and z axes proved to be a success. While the accelerometer data is not perfectly free of noise, the amount of noise we were observing while the IMU was at rest indicated that it would not cause significant problems for our magnitude of acceleration or tempo calculations. Fig. 10.a demonstrates the x-axis acceleration while at rest via the Arduino IDE's serial plotter, while Fig. 10.b. demonstrates the x-axis acceleration while the IMU is in motion.

```
17:32:01.350 -> Scaled. Acc (mg) [ -00024.90, -00007.81,  01026.86 ]
17:32:01.350 -> Scaled. Acc (mg) [ -00023.44, -00004.88,  01023.44 ]
17:32:01.350 -> Scaled. Acc (mg) [ -00019.04, -00003.42,  01031.74 ]
17:32:01.350 -> Scaled. Acc (mg) [ -00011.23,  00000.00,  01027.34 ]
17:32:01.350 -> Scaled. Acc (mg) [ -00025.39, -00008.30,  01025.88 ]
17:32:01.350 -> Scaled. Acc (mg) [ -00021.97,  00004.88,  01023.93 ]
17:32:01.350 -> Scaled. Acc (mg) [ -00022.46,  00001.46,  01026.86 ]
```

Fig. 9. The MCU displaying three-axis IMU data via serial monitor.



(a) X-axis acceleration data at rest. (b) X-axis acceleration data in motion.

Fig. 10. X-axis acceleration data observed on a serial plotter.

Similar data that passed our requirements were observed in the Y and Z axes as well. An altered requirement/verification process was required for the Z axis due to +1*g* biasing as a result of gravity.

Subsystem 2: Control Unit

We were able to achieve the core requirement of calculating the correct tempo with a less than ten percent margin of error. Out of ten trials of the process explained in Appendix C.2 performed at each tempo, every single trial concluded within a five percent margin of error. Fig. 11 demonstrates a pair of successful trials demonstrating a perfect end result. Fig. 11.a shows the

serial monitor output when conducting at 80 BPM, Fig. 11.b is at 120 BPM, and Fig. 11.c is at 160 BPM. The final 16 samples of every trial were also all within a 5 percent margin of error as shown in the figures below, although this was not explicitly a requirement.

```
15:50:37.417 -> 79
15:50:38.176 -> 79
15:50:38.930 -> 79
15:50:39.680 -> 79
15:50:40.399 -> 79
15:50:41.177 -> 80
15:50:41.915 -> 80
15:50:42.651 -> 80
15:50:43.406 -> 80
15:50:44.153 -> 80
15:50:44.909 -> 80
15:50:45.647 -> 80
15:50:46.378 -> 80
15:50:47.171 -> 80
15:50:47.926 -> 80
15:50:48.636 -> 80
```

(a) 80 BPM.

```
15:51:51.119 -> 118
15:51:51.593 -> 118
15:51:52.106 -> 118
15:51:52.643 -> 119
15:51:53.112 -> 118
15:51:53.620 -> 119
15:51:54.148 -> 118
15:51:54.611 -> 119
15:51:55.129 -> 120
15:51:55.655 -> 120
15:51:56.120 -> 120
15:51:56.627 -> 120
15:51:57.134 -> 120
15:51:57.649 -> 120
15:51:58.141 -> 120
15:51:58.630 -> 120
```

(b) 120 BPM.

```
15:49:09.743 -> 159
15:49:10.172 -> 157
15:49:10.560 -> 158
15:49:10.926 -> 158
15:49:11.330 -> 157
15:49:11.645 -> 158
15:49:12.019 -> 158
15:49:12.380 -> 159
15:49:12.758 -> 159
15:49:13.099 -> 159
15:49:13.557 -> 160
15:49:13.880 -> 162
15:49:14.255 -> 161
15:49:14.661 -> 161
15:49:14.990 -> 160
15:49:15.416 -> 160
```

(c) 160 BPM.

Fig. 11. Calculated tempo at various speeds. Last 16 samples for each trial is shown.

The LED was proven to have a current flow and turn on when the device is powered on. However, we did not have the lab access required to verify the lux of the LED. Professors and teaching assistants have verified with the eye test that the LED is visible from a distance of two feet, but we failed to obtain quantitative results on the LED's brightness.

Subsystem 3: Communication

To verify the HC-05 can transfer data at AT least the output data rate of the MCU, we programmed the HC-05 to have a baud rate of 115200 baud and requested the baud rate from the module. We set the module to operate at 115200 baud because it is the fastest baud rate that is compatible with laptop devices and avoids data corruption that can occur with faster transmission rates.

After typing in the AT command that requests the set baud rate, the Arduino terminal shows that it is the desired 115200 baud (Fig. 12). The serial monitor must be set to "NL+CR"; otherwise no AT commands will be received (Fig. 13).

Fig. 12. Arduino serial monitor exhibiting the baud rate of the HC-05.

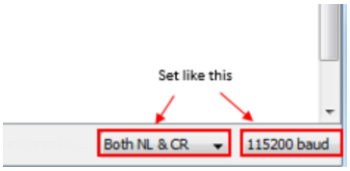


Fig. 13. Settings needed for Arduino serial monitor on HC-05 module program.

To ensure our module was successfully transmitting data to our Processing program, we connected the MBA to the user's PC and ran the program to confirm that data was being updated corresponding to the MBA's movement. This meant that the HC-05 was not only connected to the user device, but was also sending readable data from the MCU to the serial port on the PC through its TX pin.

Transmission time tests performed according to the Bluetooth module section of Appendix C.3 were successful. Fig. 14a references the local timestamps for the HC-05 serial output, while Fig. 14b references the serial monitor timestamps on the receiving computer. The serial monitors display a maximum delay of 475 ms between any two inputs to the serial port. Even accounting for the fact that the GUI program polls data received from the serial port at minimum every 20 ms, the entire process from the processing of MCU data to displaying the data on the GUI remains under 500 ms.

```
16:48:33.558 -> 971
16:48:33.651 -> 972
16:48:33.745 -> 973
16:48:33.884 -> 974
16:48:33.979 -> 975
16:48:34.073 -> 976
16:48:34.168 -> 977
16:48:34.259 -> 978
16:48:34.355 -> 979
16:48:34.449 -> 980
```

```
16:48:33.986 -> 971
16:48:34.126 -> 972
16:48:34.220 -> 973
16:48:34.313 -> 974
16:48:34.407 -> 975
16:48:34.501 -> 976
16:48:34.595 -> 977
16:48:34.689 -> 978
16:48:34.829 -> 979
16:48:34.876 -> 980
```

(a) Timestamps for HC-05. *(b) Timestamps for receiving computer.*

Fig. 14. Timestamps on origin and receiving computer. Note the difference between computers never exceeds 500 ms in real time.

Subsystem 4: User Display

We have confirmed that the user display program is both capable of communicating with the HC-05 and displaying the transmitted data on a GUI. This was demonstrated extensively during our final demonstration, although Fig. 7 and 8 showcase other examples. We also verified that the GUI is able to update at least twice a second. We referred to our CSV files and found that the GUI updated as frequently as 5 times a second, and in theory could update 50 times a second due to its update rate since it could fetch serial port values every 20 ms.

Subsystem 5: Power Supply Unit

There were three important requirements for the power supply that we verified through multiple trials. To ensure our device is able to run in full operation for each of these requirements, the MBA was turned on, connected to the user's computer over Bluetooth, and the GUI was run. Data must be actively sent over Bluetooth when testing because the TX pin draws approximately 250 mA of current when transmitting data, compared to the 30 mA that is drawn when no data is being sent to the computer.

The first verification confirmed that the power supply unit was able to supply enough power to the circuit for at least four hours on a full charge. To do so, the voltage of the battery was recorded for 4 hours at 30-minute intervals. Through multiple trials, we see that the battery reliably powered our circuit for four hours (Fig. 15). A similar test on the output of the regulator was executed to ensure it was consistently held at approximately 3.3 V, which was the voltage needed for the IMU, MCU, and Bluetooth module. It was verified that this was the case through our trials (Fig. 16).

To ensure the temperature of the battery did not exceed the operating temperature that can damage the user's skin (44°C), a thermistor that was connected to a DMM was held flush against the battery for 4 hours at 30-minute intervals (Fig. 17).

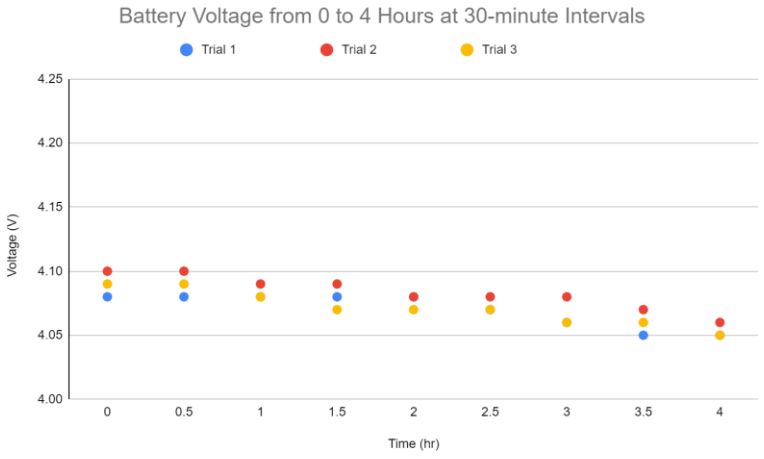


Fig. 15. 3 Trials of Battery Voltage from 0 to 4 Hours at 30-minute intervals.

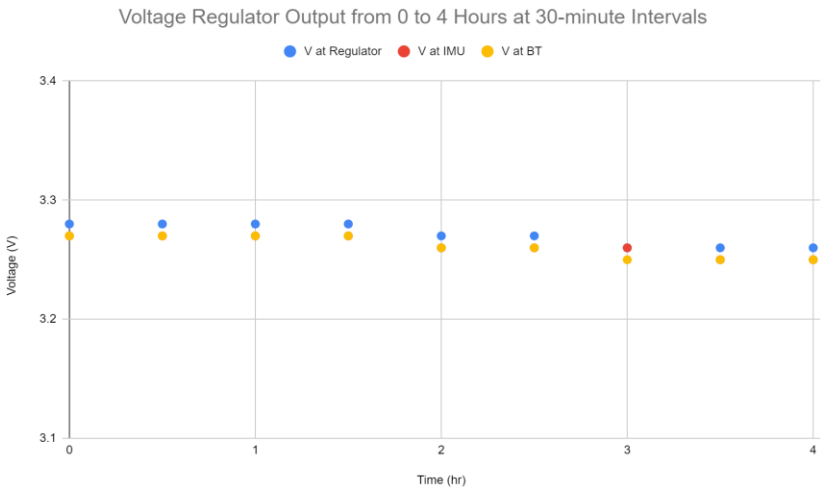


Fig. 16. 3 Trials of Voltage Regulator Output from 0 to 4 Hours at 30-minute intervals.

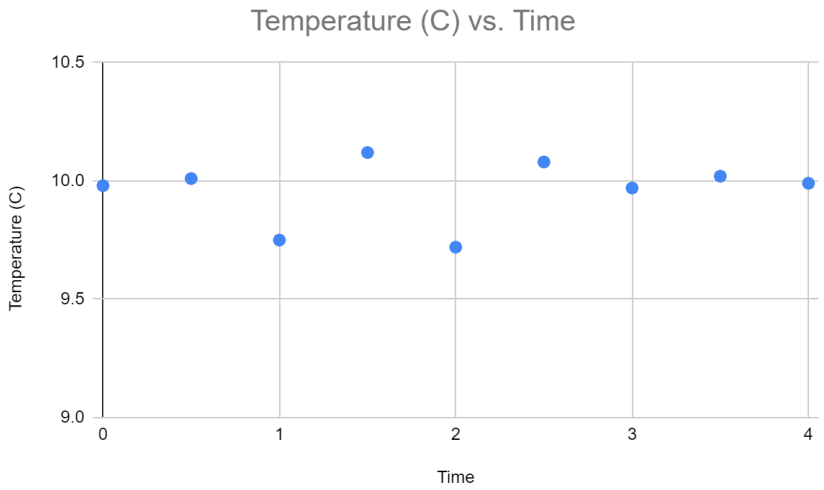


Fig. 17. Temperature of the Battery from 0 to 4 Hours at 30-minute intervals.

The switch successfully turns the circuit on and off, but we were not able to test if it requires 1000 grams of force. However, it appears to need at least this amount of force and is unlikely to be accidentally toggled, which is the basis of our verification test.

4. COST ANALYSIS AND SCHEDULE

4.1 Final Costs of Labor

Our research shows that the average salary nationwide for a Consumer Electronics Engineer is $75,591 as of September 24, 2020 [14]. Assuming an engineer worked 40 hours a week for 52 weeks, their hourly wage would be $36.34. On average, we each worked on 1 device for 10 hours a week for 10 weeks. Additionally, we will need to account for 2.5 times the amount of cost as originally expected to account for overhead. Therefore, we estimate the total labor costs to be 3 people * 36.34 dollars an hour * 10 hours * 10 weeks * 2.5 = **$27,255**.

4.2 Final Costs of Parts

The final costs of all parts necessary in our project can be found in Appendix E. Our total cost to build one device was **$68.31**. It is worth noting that the amount we personally spent was greater, due to ordering extra parts and currently unnecessary components.

4.3 Total Costs

Table 1: Total costs of project, including labor and parts.

Section	**Cost**
Labor	$27,225.00
Parts	$68.31
Total	**$27,293.31**

4.4 Schedule

Our first mission after completing the design document was to conduct further research into methods that would help our project succeed. Wynter did research on the optimal way to package the device, Alyssa studied how to interface the ATMega328P and the HC-05, and Prashant researched peak detection. We all contributed to constructing the first and all further PCB designs. Once the team obtained parts one week later, Prashant worked extensively on the microcontroller program and created the core functionality of the GUI. Alyssa configured the HC-05 module to work at the specified baud rate and collaborated with the Machine Shop in fabricating an enclosure, and Wynter continued to develop the attachment's physical design.

The final two weeks of the project were the most time-consuming. Alyssa and Prashant worked on soldering all the components onto the PCB and spent multiple days debugging the circuit. Once the circuit was complete, Alyssa conducted verification tests on the power supply while Wynter completed the GUI. All three of us ensured that we met as many verifications processes as we could before the final demonstration.

5. CONCLUSION

5.1 Accomplishments

We were successful in meeting our original objective of creating a device that was capable of detecting conducting motions and giving the user feedback on his or her tempo. The device meets all of our high-level requirements, resulting in a robust and user-friendly marching band assistant that correctly identifies conducting tempos between 80 BPM and 160 BPM and transmits the data wirelessly to the user's personal computer, on which an executable program displays and records the tempo they are conducting at. Some requirements and verification processes had to be altered, but we have demonstrable proof that most of our requirements were met. The device has no electrical components exposed and is enclosed in a compact plastic casing that fits any arm with the help of a velcro strap.

5.2 Uncertainties

The range reached by the Bluetooth module was not tested exhaustively because it was not a requirement, but the MBA was tested within a range an average user would be able to still see the feedback display; the distance between the MBA and the computer was at most about 5 feet away during testing.

The MBA packaging was larger than intended because a plastic enclosure that fit the dimensions of our system could not be found. The excess space at the top of the box may cause the PCB to experience additional vibrations from movement and consequently interfere with the IMU's acceleration readings. Although this has not been an issue during testing, a smaller casing would be ideal for user comfort and mechanical integrity.

There are potential bugs with the user display that have not been extensively tested. The behavior of the executable program when selecting a non-available serial port is unknown. Additionally, the program could face potential issues when pressing "Record" or "Stop" while the program is establishing reconnection.

5.3 Ethical Considerations

This device incorporates technology into musical training in a new and innovative way, in order to "contribute to society," as stated in the ACM Code of Ethics [15]. The goal of this device is to assist drum majors by providing a learning tool; both drum majors and marching band members would benefit from it. The IEEE Code of Ethics states that members have a responsibility "to improve the understanding by individuals and society of the capabilities and societal implications of conventional and emerging technologies" [16]. If this product were to be commercialized, an instruction manual would be provided to help users understand the capabilities of this new technology and how it is used.

The project utilizes a Bluetooth module to transmit data. Data theft is a potential security risk associated with this Bluetooth [17]. While the attachment does not transmit sensitive or personal data, the recieving computer could have personal data vulnerable to a Bluetooth security breach. Therefore, it would be recommended that the device be used in a trusted place without significant wireless interference. Additionally, the device (and with it, the Bluetooth module) should be powered off when not in use.

The safety of the user is the top priority of this device, as the IEEE Code of Ethics clarifies that it "holds paramount the safety, health, and welfare of the public" [16]. Since the device is placed directly on the user's arm, it is important that the electrical components do not overheat or shock the user. The Li-Poly battery is flammable, so additional precautions were taken; regulators were implemented to prevent the battery voltage from decaying below 3.0 V/cell or exceeding 4.2 V/cell [18]. The whole system, including the battery, were secured in a plastic enclosure as an added layer of protection for the user. Nylon was used for the attachment because it is a relatively heat-resistant and insulating fabric.

5.4 Future Work

There are quite a few improvements we would still like to make to improve the performance of the MBA and enhance the user's experience with the device. The microcontroller program may be capable of more refined peak detection, so additional testing, calibration, and improvements to the algorithm can be made to better model signals specific to conducting motions.

A basic utility that would be beneficial for the GUI is built-in metronome so that the user would not need an external device to practice conducting to a specific tempo. An option for increasing and decreasing the sensitivity of peak detection on the GUI is also desirable, as it would allow MBA to adjust for more sharp or more fluid conducting. Furthermore, a feature that allows the user to conduct along with pieces of music (similar to SmartMusic) would expand the versatility of the device.

A practical feature we would like to implement is an LED on the PCB that blinks or changes color when the battery is low. We currently only have an LED that would turn off if the battery is no longer able to supply the power necessary for full operation of the circuit. Finally, miniaturization of the device through the elimination of unused components (such as JST ports), the use of the smaller surface-mount variant of the ATMega328P, and installation of the circuitry in a smaller customized box would lead the device to be more compact and comfortable for the user.

REFERENCES

[1] R. Raymond and C. Aliga. *Today's Drum Major | Central Iowa Color Guard and Drum Major Camp 2016.* (2016). Accessed: Dec. 9, 2020. [Online]. Available: http://www.centraliowacolorguard.com/upload/398755/documents/Drum%20Major%20Handbook%202016.pdf

[2] *DB-90: Dr. Beat*. BOSS. Accessed: Dec. 9, 2020. [Online]. Available: https://www.boss.info/us/products/db-90/.

[3] *Smartmusic*. MakeMusic. Accessed: Dec. 9, 2020. [Online]. Available: https://www.smartmusic.com/.

[4] *ICM-20948*. TDK. (2017). Accessed: Dec. 9, 2020. [Online]. Available: https://product.tdk.com/info/en/documents/catalog_datasheet/imu/DS-000189-ICM-20948-v1.3.pdf

[5] *SparkFun 9DoF IMU (ICM-20948) Breakout Hookup Guide*. Sparkfun. (2019). Accessed: Dec. 9, 2020. [Online]. Available: https://learn.sparkfun.com/tutorials/sparkfun-9dof-imu-icm-20948-breakout-hookup-guide

[6] *AtMega328P Microcontroller*. Components101. (2018). Accessed: Dec. 9, 2020. [Online]. Available: https://components101.com/microcontrollers/atmega328p-pinout-features-datasheet

[7] *ATmega48A/PA/88A/PA/168A/PA/328/P megaAVR Data Sheet.* Microchip. (2020). Accessed: Dec. 9, 2020. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf

[8] *From Arduino to a Microcontroller on a Breadboard.* Arduino. (2018). Accessed: Dec. 9, 2020. [Online]. Available: https://www.arduino.cc/en/Tutorial/BuiltInExamples/ArduinoToBreadboard

[9] leandcesar. *PeakDetection.* Github. (2020). Accessed: Dec. 9, 2020. [Online]. Available: https://github.com/leandcesar/PeakDetection

[10] J. Kopstein. *Marching Speeds.* Altissimo! Recordings. Accessed: Dec. 9, 2020. [Online]. Available: https://militarymusic.com/blogs/military-music/13516233-marching-speeds

[11] *HC-05 Bluetooth Module*. Components101. (2018). Accessed: Dec. 9, 2020. [Online]. Available: https://components101.com/wireless/hc-05-bluetooth-module

[12] *HC-05 Bluetooth Module Interfacing with AVR ATmega16/ATmega32: A….* ElectronicWings. Accessed: Dec. 9, 2020. [Online]. Available: https://www.electronicwings.com/avr-atmega/hc-05-bluetooth-module-interfacing-with-atmega1632.

[13] A. Veeravalli, S.M. Nolan. *Introduction to Low Dropout (LDO) Linear Voltage Regulators.* Design & Reuse. Accessed: Dec. 9, 2020. [Online]. Available: https://www.design-reuse.com/articles/42191/low-dropout-ldo-linear-voltage-regulators.html

[14] *Consumer Electronics Engineer Salary.* Ziprecruiter. (2020). Accessed: Dec. 9, 2020. [Online]. Available: https://www.ziprecruiter.com/Salaries/Consumer-Electronics-Engineer-Salary

[15] D. Gotterbarn. *ACM Code Of Ethics And Professional Conduct*. (2020). Accessed: Dec. 9, 2020. [Online]. Available: https://www.acm.org/code-of-ethics

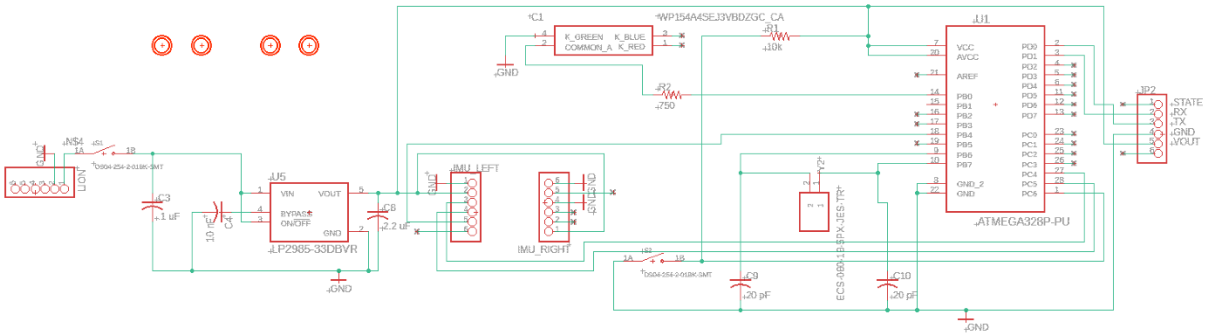
[16] *IEEE Code Of Ethics*. IEEE. (2020). Accessed: Dec. 9, 2020. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html

[17] *Wireless Connections and Bluetooth Security Tips.* Federal Communications Commission. Accessed: Dec, 9, 2020. [Online]. Available: https://www.fcc.gov/consumers/guides/how-protect-yourself-online

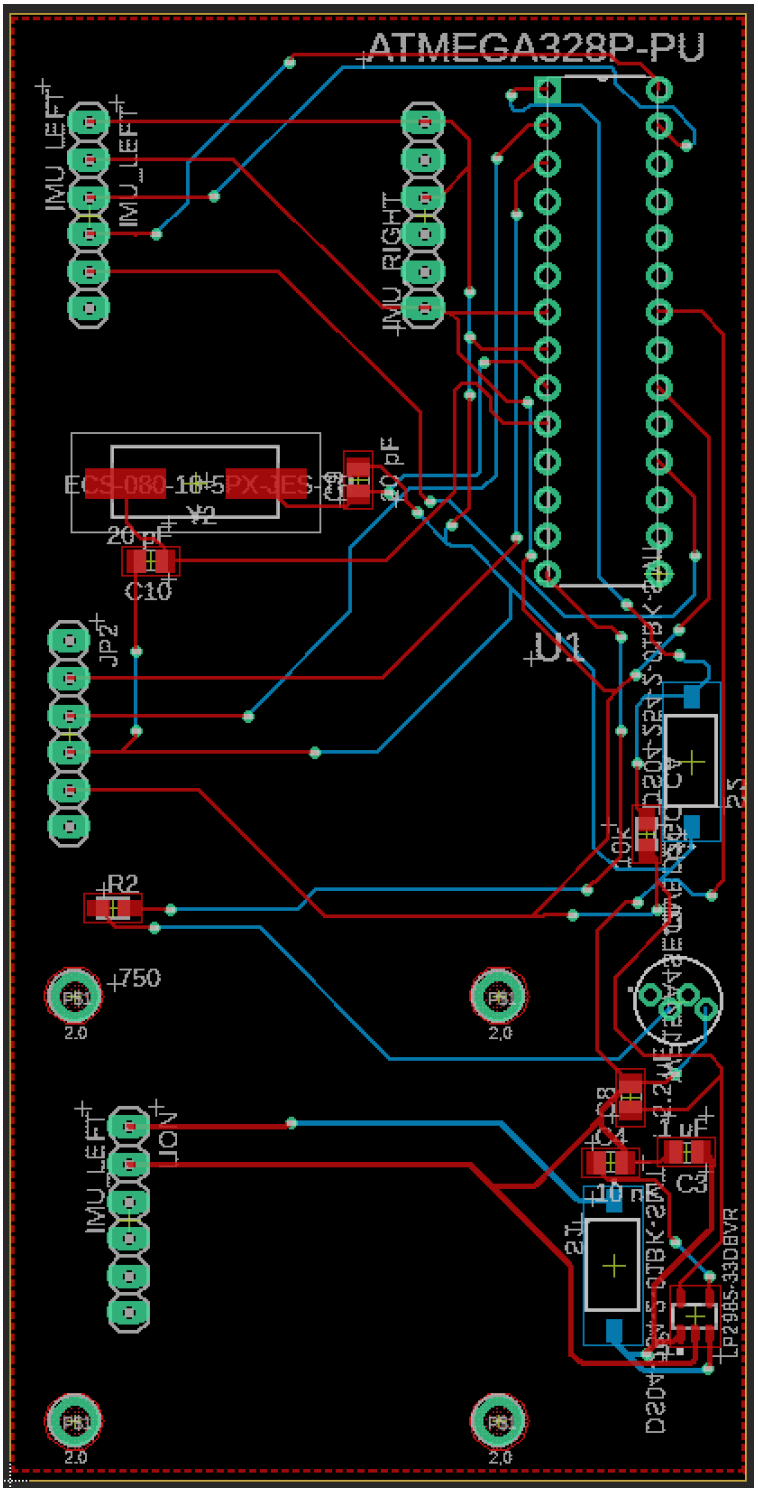
[18] Spring 2016 Course Staff, Champaign, IL, USA. *Safe Practice For Lead Acid And Lithium Batteries*. 2016. Accessed: Dec. 9, 2020. [Online]. Available: https://courses.engr.illinois.edu/ece445/documents/GeneralBatterySafety.pdf

[19] *Modify the HC-05 Bluetooth Module Defaults Using AT Commands.* Instructables Circuits. (2013). Accessed: Dec. 9, 2020. [Online]. Available: https://www.instructables.com/Modify-The-HC-05-Bluetooth-Module-Defaults-Using-A/

APPENDIX A: MBA SCHEMATIC



APPENDIX B: MBA PCB LAYOUT



APPENDIX C: Requirement and Verification Tables

C.1. Inertial Measurement Unit

Requirement	Verification	Verified?
Inertial Measurement Unit 1) IMU must be capable of sending acceleration data to the MCU via I2C.	Inertial Measurement Unit 1) Check that the MCU receives any packet of data from the IMU. *Verification Process:* 1) Load program that pings the ICM-20948 IMU via I2C onto the ATMega328P MCU. 2) Connect the MCU to the IMU. 3) Power on both devices with any appropriate power supply and run the microcontroller script. If any data packet from the IMU is returned through the MCU via serial, the test was successful.	Y
2) IMU must record correct acceleration data in the x and y axes.	2) The IMU must read 0 m*g* ± 100 m*g* when resting, and must read at least 800 m*g* when moving for the x and y axes. *Verification Process:* 1) Load program that samples x-direction acceleration data onto the ATMega328P MCU. 2) Connect the ICM-20948 IMU to the MCU. 3) Lay the IMU on a flat, stationary surface. 4) Collect data over three seconds. 5) Check that the x-acceleration data does not ever exceed ±100 m*g* while stationary. 6) Hold IMU in the palm of hand. 7) Move hand back and forth to a metronome set at 160 BPM at least 2 feet along the IMU's x-axis. Continue the process for at least 10 seconds. 8) Check that the x-acceleration exceeds |800 m*g*| at least once. 9) Repeat steps 7 and 8 along the y-axis, loading in a y-sampling program instead and performing tests in the IMU's y-axis. *(Continued on next page)*	Y

3) IMU must record correct acceleration data in the z axis.	3) The IMU must read 1000 m*g* ± 10% when at rest, and must read at least 1800 m*g* for the z-axis. *Verification Process:* 1) Load program that sample z-direction acceleration data onto the ATMega328P MCU. 2) Connect the ICM-20948 IMU to the MCU. 3) Lay the IMU on a flat, stationary surface. 4) Collect data over 3 seconds. 5) Check that the z-axis acceleration data reads 1000 m*g* ± 100m*g* while stationary. 6) Hold IMU in palm of hand. 7) Move hand back and forth to a metronome set at 160 BPM at least 2 feet along the IMU's z-axis. Continue process for at least 10 seconds. 8) Check that the z acceleration exceeds 1800 m*g* at least once.	Y

C.2. Control Unit

Requirement	Verification	Verified?
<u>Microcontroller</u> 1) MCU program must correctly identify tempo of 80 BPM.	<u>Microcontroller</u> 1) Check that Serial Monitor displays accurate tempo when attempting to conduct at 80 BPM. *Verification Process:* 1) Upload program that calculates the rolling tempo average using IMU peak detection onto the ATMega328P MCU. 2) Connect the MCU to the ICM-20948 IMU. 3) Hold MCU/IMU (either on sleeve or in hand) and bring it up to the air. Keep arm/hand completely still until Step 5. 4) Power on both devices with any appropriate power supply and upload the microcontroller script. 5) Every 0.75 seconds, alternate between moving arm at least 1 ft down as fast as possible and moving arm at least 1 ft up as fast as possible. Arm must be stopped at each position for at least 0.25 seconds, so the motion can take no longer than 0.5 seconds. This process simulates conducting at 80 BPM in 2/4 time. If necessary, keep a metronome for 240 BPM to ensure results are valid. Perform Step 5 for 20 seconds starting from the first movement. 6) Read the Serial display of the MCU. If the latest output line displayed on the Serial display reads between 72 and 88 inclusive, the test was a success.	Y
2) MCU program must correctly identify tempo of 120 BPM.	2) Check that Serial Monitor displays accurate tempo when attempting to conduct at 120 BPM. *Verification Process:* 1) Upload program that calculates the rolling tempo average using IMU peak detection onto the ATMega328P MCU. 2) Connect the MCU to the ICM-20948 IMU. *(Continued on next page)*	Y

	3) Hold MCU/IMU (either on sleeve or in hand) and bring it up to the air. Keep arm/hand completely still until Step 5. 4) Power on both devices with any appropriate power supply and upload the microcontroller script. 5) Every 0.5 seconds, alternate between moving arm at least 1 ft down as fast as possible and moving arm at least 1 ft up as fast as possible. Arm must be stopped at each position for at least 0.25 seconds, so the motion can take no longer than 0.25 seconds. This process simulates conducting at 120 BPM in 2/4 time. If necessary, keep a metronome for 240 BPM to ensure results are valid. Perform Step 5 for 20 seconds starting from the first movement. 6) Read the Serial display of the MCU. If the latest output line displayed on the Serial display reads between 108 and 132 inclusive, the test was a success.	
3) MCU program must correctly identify tempo of 160 BPM.	3) Check that Serial Monitor displays accurate tempo when attempting to conduct at 160 BPM. *Verification Process:* 1) Upload program that calculates the rolling tempo average using IMU peak detection onto the ATMega328P MCU. 2) Connect the MCU to the ICM-20948 IMU. 3) Hold MCU/IMU (either on sleeve or in hand) and bring it up to the air. Keep arm/hand completely still until Step 5. 4) Power on both devices with any appropriate power supply and upload the microcontroller script. *(Continued on next page)*	Y

	5) Every 0.375 seconds, alternate between moving arm at least 1 ft down and up as fast as possible. Arm must be stopped at each position for at least 0.1875 seconds, so the motion can take no longer than 0.5 seconds. This process simulates conducting at 160 BPM in 2/4 time. If necessary, keep a metronome for 320 BPM to ensure results are valid. Perform Step 5 for 20 seconds starting from the first movement. 6) Read the Serial display of the MCU. If the latest output line displayed on the Serial display reads between 144 and 176 inclusive, the test was a success.	
<u>Status LED</u> 1) LED must be able to turn on.	<u>Status LED</u> 1) Ensure that the LED is able to turn on. *Verification Process:* 1) Attach DMM probes to respective leads. 2) Turn switch "OFF". 3) Measure current going through the LED while the switch is off. 4) Turn switch "ON". 5) Measure current going through LED while switch is on. 6) Verify that the currents measured are in the correct operating regions on the LED.	Y
2) LED must be visible from two feet away.	2) The lux of the LED will be measured for >= 30 lux while in a room with under 15 lux. *Verification Process:* 1) Connect the two ends of a 1% tolerance photoresistor to the DMM probes. 2) Turn off lights in verification lab. 3) Measure lux of the lab when lights are off, by placing the photoresistor two feet away from the LED and measuring the resistance. 4) Turn switch "ON" to turn on the LED. 5) Measure lux of the lab while LED is on and lights are off, by placing the photoresistor two feet away from the LED and measuring the resistance. 6) Subtract the lux of the lab when lights are off from the lux of the lab when LED is on and	N

	lights are off to obtain the lux of the LED. 7) Check that the lux measured is above 30.	

C.3. Communication

Requirement	Verification	Verified?
<u>Bluetooth Module</u> 1) The Bluetooth module must have a baud rate greater than 1.125 kBd, the output data rate of the MCU.	<u>Bluetooth Module</u> 1) The Bluetooth module's baud rate will be obtained from and displayed by the Arduino IDE. *Verification Process:* 1) Insert Bluetooth module header pins into breadboard. Make the following connections from the module to an Arduino Uno with jumper wires (connect wire in same row as respective pin on the breadboard): HC-05 GND ---> Arduino GND Pin HC-05 VCC (5V) ---> Arduino 5V HC-05 TX ---> Arduino Pin 10 (soft RX) HC-05 RX ---> Arduino Pin11 (soft TX) HC-05 Key (PIN 34) ---> Arduino Pin 9 2) Load and compile the Arduino program HC-05.ino (Appendix F) in the IDE by clicking "Verify" [19]. 3) Before connecting the Arduino to the USB, remove the VCC wire from the HC-05 so it is not getting any power from the Arduino. 4) Connect the Arduino Uno to the USB cable extended from a PC. Upload the HC-05.ino program to the board by clicking "Upload" in the IDE. 5) Reconnect the Arduino Uno 5V wire to the HC-05's VCC pin. The HC-05 LED will blink on and off at about 2 second intervals, indicating that the HC-05 is in AT command mode and ready to accept commands. 6) Open the Serial Monitor from the Arduino IDE, type "AT", and click SEND. "OK" should appear on the terminal to confirm the HC-05 is properly connected to the PC via Bluetooth. 7) Type "AT+UART" in the Serial Monitor to see the baud rate the module is operating at. *(Continued on next page)*	Y

	8) Confirm the baud rate is greater than 1125 Bd. If not, type “"AT+UART=9600,1,0" to explicitly set the baud rate (9600 is the default). Upload the program to the Arduino again and repeat steps 6 and 7. 9) Confirm the baud rate is >= 1125 Bd.	
2) The Bluetooth module must transmit data between the MBA and a host machine with a delay of less than 500 ms.	2) The transmission time will be calculated by taking the timestamp differences on two computers. *Verification Process:* 1) Upload [ping.ino] onto MCU that writes an incrementing number to the Serial Monitor every 100 ms. 2) Power on MCU using a host computer. Host computer must be running on Windows 7+. 3) Connect the Bluetooth module to the MCU. 4) Connect to Bluetooth module wirelessly using a second host computer. Host computer must be running on Windows 7+. 5) Open up the Arduino IDE Serial Monitor on the computer that the MCU is hooked up to. Enable “Show timestamp” feature. 6) Open up Arduino IDE Serial Monitor on first host computer. Enable “Show timestamp” feature. 7) Disable auto-scrolling and observe any ten consecutive samples of the same data on both monitors, then calculate the difference of timestamp. If every difference is less than 500 ms, the test was successful.	Y

C.4. User Display

Requirement	Verification	Verified?
<u>User Display</u> 1) The user display can verify that a connection to the Bluetooth module was made.	<u>User Display</u> 1) Check if the backend of the user display recognizes connection to device via Bluetooth. *Verification Process:* 1) Load Processing script that attempts to connect to a specified Bluetooth module and returns a print message (e.g. "Success") when connection is successful. 2) Load script onto computer that supports Bluetooth capability. 3) Power on HC-05 Bluetooth module within 5 m of the host computer. 4) Connect to HC-05 on computer via Bluetooth. 5) Run the Processing script. If the specified print message is outputted, the test was successful.	Y
2) IMU data sent by the MCU is visible on the user display in any numerical form. (This includes calculations performed on IMU data.)	2) Check if raw MCU data is displayable by GUI. *Verification Process:* 1) Load Processing script that prints the raw output data of the ATMega328P MCU onto a GUI as soon as data is received on recieiving computer. 2) Connect MCU to ICM-20948 IMU, and connect MCU/HC-05 to host computer via Bluetooth. The ATMega328P must have script that loaded onto it. 3) Run the Processing script for at least 10 seconds. If the raw data received from the MCU is displayed on the GUI in any numerical form (e.g. int, float, etc.), the test was successful.	Y
3) User Display screen updates with new data at least two times a second.	3) Check using a counter if GUI is updating at an acceptable rate. *Verification Process:* *(Continued on next page)*	Y

	1) Repeat Step 1 of Requirement 2 of the User Display, and add a counter that keeps track of how many times the GUI has updated each second (counter does not have to be visible on GUI). 2) Connect MCU to ICM-20948 IMU, and connect MCU to host computer via Bluetooth. 3) Run the Processing script for at least 10 seconds while performing Step 4. 4) Simulate conducting at 160 BPM as described in the IMU subsystem requirements, Req 4, Step 5. If the counter is at least equal to 2 for every second the script was run, the test was successful.	

C.5. Power Supply

Requirement	**Verification**	**Verified?**
<u>Li-Poly Battery</u> 1) Supply +3.7V ± 5% power and stores 2500 mAh ± 5% of charge.	<u>Li-Poly Battery</u> 1) The current output will be confirmed to be within the acceptable range with a digital multimeter. *Verification Process:* 1) Ensure the battery has been fully charged (reads +3.7V ± 5%). 2) Attach a 11 Ohm resistor to the battery, the equivalent resistance of the entire circuit. 3) Attach DMM probes to respective JST connector pins of the battery. 4) Measure and record current at 5 minute intervals for 60 minutes. 5) Confirm that less than 2375 mA was extracted.	Y
2) Lifetime per charge of the sleeve should be at least 4 hours of usage.	2) The lifetime of the sleeve will be measured while it is turned on. *Verification Process:* 1) Ensure the battery has been fully charged (reads +3.7V ± 5%). 2) Attach an 11 Ohm resistor to the battery, the equivalent resistance of the entire circuit. 3) Attach DMM probes to respective JST connector pins of the battery.	Y

3) Must not exceed an operating temperature that can damage the user's skin (44°C) [17].	4) Measure and record I and V at half hour intervals for 4 hours. 5) Confirm the voltage of the battery is at least 3.7 V. 3) The temperature of the battery will be measured with a thermistor. *Verification Process:* 1) Ensure the battery has been fully charged (reads +3.7 V ± 5%). 2) Attach a 11 Ohm resistor to the battery, the equivalent resistance of the entire circuit. 3) Attach the thermistor to DMM probes and hold against battery. 4) Measure and record the temperature at 30 minute intervals. 5) Terminate verification process when the fourth hour is reached.	Y
<u>Linear Voltage Regulator</u> 1) Voltage must be regulated to +3.3V ± 5% for the Bluetooth module, microcontroller, and the VDD pin on the IMU	<u>Linear Voltage Regulator</u> 1) Stable voltage outputs at the desired values will be measured and observed through oscilloscope waveforms. *Verification Process:* 1) Attach the oscilloscope GND probe to GND of the PCB and the signal probe to the VDD input pin of the Bluetooth module. 2) Supply regulator with 3.7V DC from a power supply. 3) Ensure output voltage remains 3.3V. 4) Repeat steps 1-3 for the microcontroller and the VDD pins on the IMU.	Y
<u>On/Off Switch</u> 1) Switch must have an operating force above 1000 grams [8, 9].	<u>On/Off Switch</u> 1) Check that the operating force of the switch is above 1000 grams. *Verification Process* 1) Connect switch in series with a resistor and battery. 1) Attach DMM probes parallel to the resistor. 2) Ensure the switch is initially in the "OFF" position by checking that the voltage of	N

	the resistor is 0V. 3) Stack weights on top of the switch, until the switch is turned on and a non-zero voltage is measured across the resistor. 4) Check the size of the weights to ensure that it is over 1000 grams.	

APPENDIX D: Current and Supply Voltage Requirements

Component	Current Requirements	Voltage Requirements (Typical Operating Voltage)
ICM-20948 9DoF IMU Breakout Board	Max: 3.11 mA	3.3 V
Bluetooth Module (HC-05)	250 mA in full operation (transmitting data). 30 mA when in standby.	3.3 V
ATMega328P Microcontroller	1.5 mA	3.3 V
1-position DIP Switch	Max: 25 mA	Max: 24 V
Status RGB LED	R: 30 mA (max) G: 25 mA (max) B: 30 mA (max)	R: 2.2 V G: 3.3 V B: 3.3 V
LP2985 Low-Noise Low-Dropout Voltage Regulator	Output: 150 mA	Supply Input Voltage: Min: 2.2 V Max: 16 V Relevant Fixed Output Options: +3.3V
Lithium Ion Polymer Rechargeable Battery	Typical: 2500 mAh	Nominal: 3.7 V

To calculate the battery capacity in mA-hours needed for the subsystems on the sleeve, we add the maximum currents drawn by each subsystem:

$$3.11\,mA\ +\ 250\,mA\ +\ 1.5\,mA\ +\ 25\,mA\ +\ 30\,mA\ =309.61\,mA$$

If we want the fully charged MBA sleeve to last 4 hours, we utilize the following equation:

$$\frac{BatteryCapacity(mA\cdot hours)}{CurrentDraw(mA)} = BatteryLife(hours)$$

It is good practice to assume our battery will have less than ideal battery life. This can be compensated with the assumption that a real-world battery life will be 75% of its theoretical value. The parameters used for calculating the battery capacity on the sleeve are 309.61 mA for the current draw and 5.5 hours (approximately 1.5 hours more than we would expect) for battery life. This results in a battery capacity requirement of at least 1702.86 mAh. Because the largest

voltage needed for these components is 3.3 V, we needed a battery that can supply at least this amount as well. We chose a Li-Poly battery with a nominal voltage supply of 3.7 V and a typical supply current of 2500 mAh to extend the lifetime of the sleeve to approximately 8 hours before charging is needed.

APPENDIX E: Parts Cost Table

Name	Manufacturer	Part Number	Quantity	Unit Price ($)
Rechargeable Li-Poly Battery 3.7V	Adafruit	LIPO785060	1	11.49
USB Li-Ion/Li-Poly Charger v1.2	Adafruit	MCP73833/4	1	19.42
Mini B to USB Adapter Cable	Amazon	N/A	1	6.28
LDO Voltage Regulator	Texas Instruments	LP2985-33DBVR	1	0.51
IMU Breakout Board	Invensense	ICM-20948	1	5.91
HC-05 Bluetooth Module	DSD TECH	HC-05	1	8.99
ATMega328P MCU	Microchip	ATMega328P-PU	1	2.08
DIP Switch	CUI Devices	DS04-254-SMT	1	0.70
Green LED	SparkFun	COM-10633 ROHS	1	0.55
Velcro Fastener Strips	VELCRO	N/A	1	5.79
JST Connector	Elechawk	JJRC H36 H67	1	0.80
.01 uF Capacitor	KEMET	C0805X103K5RAC3316	1	0.33
.1 uF Capacitor	KEMET	C0805C104K5RAC7411	1	0.25
2.2 uF Capacitor	KEMET	C0805C225K4REC7210	1	0.30
20 pF Capacitor	KEMET	C0805X200J5GACTU	2	0.38
750 Ohm Resistor	TT Electronics	PFCW0805LF037500B	1	0.99
10 Kohm Resistor	Vishay	MCU0805PD1002DP500	1	0.66
Plastic Project Box	Zulkit	B07WCKF6P4	1	2.00
PCB	PCBWay	N/A	1	0.50
Total				**$68.31**

APPENDIX F: HC-05.ino Program

```
/*
AUTHOR: Hazim Bitar (techbitar)
DATE: Aug 29, 2013
LICENSE: Public domain (use at your own risk)
CONTACT: techbitar at gmail dot com (techbitar.com)
*/


#include <SoftwareSerial.h>


SoftwareSerial BTSerial(10, 11); // RX | TX


void setup()
{
  pinMode(9, OUTPUT);  // this pin will pull the HC-05 pin 34 (key pin) HIGH to switch module
to AT mode
  digitalWrite(9, HIGH);
  Serial.begin(9600);
  Serial.println("Enter AT commands:");
  BTSerial.begin(38400);  // HC-05 default speed in AT command more
}


void loop()
{


  // Keep reading from HC-05 and send to Arduino Serial Monitor
  if (BTSerial.available())
    Serial.write(BTSerial.read());

  // Keep reading from Arduino Serial Monitor and send to HC-05
  if (Serial.available())
    BTSerial.write(Serial.read());
}
```