Smart Cricket Bat

Gavin Dahl, Jiakai Hu, Pablo Barron, Nolapat Pipitvitayakul

**SUBSYSTEM REPORTS**

REVISION – 1

20 November 2022

SUBSYSTEMS REPORT

FOR

Smart Cricket Bat

Prepared by:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Author Date

Approved by:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Project Leader Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

John Lusher II, P.E. Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

T/A Date

**Change Record**

| **Rev.** | **Date** | **Originator** | **Approvals** | **Description** |
| --- | --- | --- | --- | --- |
| **-** | 4/20/2022 | Gavin Dahl |  | Draft Release |
| **1** | 11/20/2022 | Gavin Dahl |  | First Revision |

**Table of Contents**

**Table of Contents** [**3**](https://docs.google.com/document/d/1KEMfujLp_TKIFv6GNQsRDHfS5j0_PNNK/edit#heading=h.3znysh7)

**List of Tables [5](https://docs.google.com/document/d/1KEMfujLp_TKIFv6GNQsRDHfS5j0_PNNK/edit#heading=h.4d34og8)**

**List of Figures 6**

**1. Introduction 7**

**2**[**.**](https://docs.google.com/document/d/1KEMfujLp_TKIFv6GNQsRDHfS5j0_PNNK/edit#heading=h.lnxbz9) **[Power Subsystem Report 8](https://docs.google.com/document/d/1KEMfujLp_TKIFv6GNQsRDHfS5j0_PNNK/edit#heading=h.lnxbz9)**

2.1. Subsystem Introduction 8

2.2. Subsystem Details 8

2.3. Subsystem Validation 10

2.4. Subsystem Conclusion 18

**3. Control Subsystem Report 19**

3.1. Control Introduction 19

3.2. Microcontroller 19

3.2.1. Overview 19

3.2.2. Operation 19

3.2.3. Validation 20

3.3. Gyroscope 21

3.3.1. Operation 21

3.3.2. Validation 21

3.4. Accelerometer 23

3.4.1. Operation 23

3.4.2. Validation 24

3.5. Housing Unit 29

3.6. Subsystem Conclusion 29

**4**[**.**](https://docs.google.com/document/d/1KEMfujLp_TKIFv6GNQsRDHfS5j0_PNNK/edit#heading=h.lnxbz9) **[App Subsystem Report 30](https://docs.google.com/document/d/1KEMfujLp_TKIFv6GNQsRDHfS5j0_PNNK/edit#heading=h.lnxbz9)**

4.1. Subsystem Introduction 30

4.2. Subsystem Details 30

4.2.1. Bluetooth Communication 30

4.2.2. User Interaction / Dashboard 31

4.3. Subsystem Validation 32

4.4. Subsystem Conclusion 33

**5**[**.**](https://docs.google.com/document/d/1KEMfujLp_TKIFv6GNQsRDHfS5j0_PNNK/edit#heading=h.lnxbz9) **[Machine Learning Subsystem Report 34](https://docs.google.com/document/d/1KEMfujLp_TKIFv6GNQsRDHfS5j0_PNNK/edit#heading=h.lnxbz9)**

5.1. Machine Learning Introduction 34

5.2. Machine Learning Details 34

5.2.1. Bat Region Dividing 34

5.2.2. Data Gathering, Inputting, and Plotting 35

5.2.3. Finding the Impact 36

5.2.4. FFT and PSD 37

5.2.5. Peak Detection 38

5.2.6. Making Feature and Label 38

5.2.7. Training and Testing Sets 38

5.2.8. Model and Feature Importance 39

5.3. Machine Learning Validation 40

5.4. Subsystem Conclusion 41

**List of Tables**

Table 1: Load Regulation with Input Voltage Vin =3V 11

Table 2: Load Regulation with Input Voltage Vin =3.7V 13

Table 3: Load Regulation with Input Voltage Vin =4.2V 15

Table 4: Battery vs Time with Load Current =50mA 17

Table 5: Accelerometer Measured Vs Calculated Acceleration 28

**List of Figures**

Figure 1: Schematic of Boost Converter Circuit 8

Figure 2: PCB Layout for Boost Converter Circuit 10

Figure 3: Load Regulation of Boost Converter Circuit with Vin =3V 12

Figure 4: Load Regulation of Boost Converter Circuit with Vin =3.7V 14

Figure 5: Load Regulation of Boost Converter Circuit with Vin =4.2V 16

Figure 6: Battery Discharge Curve with Load current =50mA 18

Figure 7: Flow Diagram of Microcontroller Logic 20

Figure 8: X-Axis Gyroscope Validation 22

Figure 9: Y-Axis Gyroscope Validation 22

Figure 10: Z-Axis Gyroscope Validation 23

Figure 11: 5 Foot Fall Accelerometer Validation 25

Figure 12: 10 Foot Fall Accelerometer Validation 26

Figure 13: 15 Foot Fall Accelerometer Validation 27

Figure 14: Bluetooth UI 30

Figure 15: User Feedback 31

Figure 16: HC-05 Bluetooth Module Circuit 32

Figure 17: Bluetooth validation data and arduino code 33

Figure 18: Bat Division 34

Figure 19: Data Plotting 35

Figure 20: Impact Window 36

Figure 21: FFT & PSD 37

Figure 22: Peak Detection on FFT over X-axis Gyroscope of Region 12 38

Figure 23: Feature and Label 38

Figure 24: Feature Importance 1 39

Figure 25: Feature Importance 2 40

Figure 26: Training and Testing Accuracy 41

**1. Introduction**

The smart cricket bat will acquire data during the user's swing and give feedback to the user on how to further improve their swing. The system gathers data through the inertial measurement unit mounted at the handle of the bat, where it is transmitted to the consumer app via a microcontroller with a Bluetooth module. The data transferred to the app is then uploaded to the machine learning algorithm to be processed and then returns to the user the characteristics of the swing and what can be done to improve. The system is broken down into the power, control, app, and machine learning subsystems, each of which was designed and rigorously tested. Since each subsystem was validated to be working correctly and fulfilling all requirements, there is a clear path to integration for these subsystems into the full system specified in the Conops, FSR, and ICD.

**2. Power Subsystem Report**

**2.1. *Subsystem Introduction***

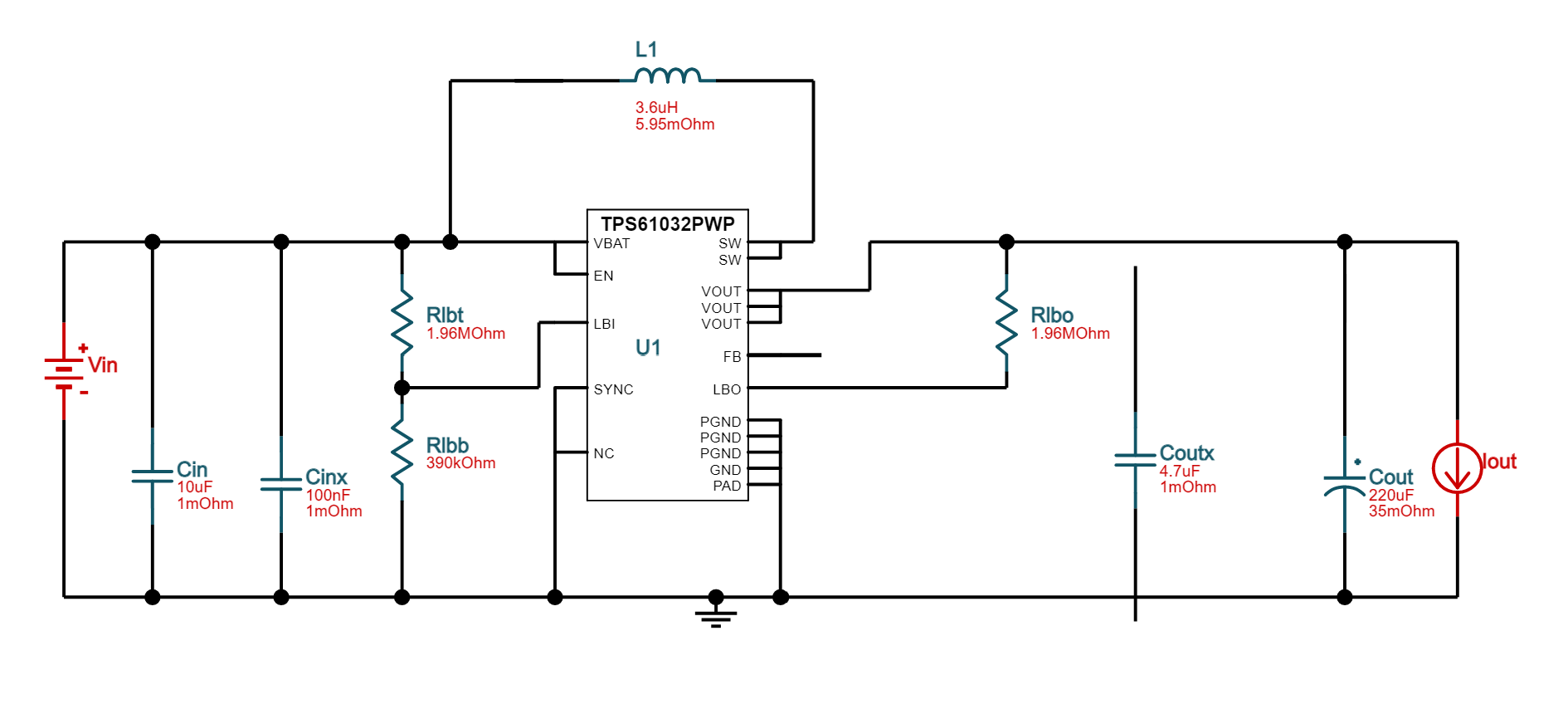
The power subsystem of the smart cricket bat consists of a lithium-ion polymer battery (LiPo), a DC-DC boost converter circuit, and a battery charger. A LiPo battery is rechargeable and can be charged with a battery charger circuit. A boost converter is used to raise the voltage from the LiPo battery in order to supply the power to the microcontroller which is part of the control subsystem. The boost converter was tested to confirm its stability and to validate that it performed correctly according to the design.

**2.2. *Subsystem Details***

The source of power for the smart cricket bat is a 3.7V LiPo battery that has a capacity of 150 mAh. The reason for choosing LiPo battery as a power source is because it is rechargeable, small, and lightweight. These features are important due to the size and weight requirements of the smart cricket bat. The battery output ranges from 3.0V to 4.2V with a nominal voltage of 3.7V. When fully charged, the battery outputs 4.2V, and it will be completely cut out when the voltage goes below 3.0V.

A 5V DC-DC boost converter circuit was designed to raise the voltage levels of the battery that ranges from 3.0V to 4.2V to a 5V voltage level in order to power the microcontroller in the control subsystem. A boost converter circuit was designed based on the requirements using TI WEBENCH Power Designer. The chip used for the boost converter was TPS61032 from Texas Instruments. Figure 1 shows the schematic of a 5V DC-DC boost converter circuit.

**Figure 1:** Schematic of Boost Converter Circuit



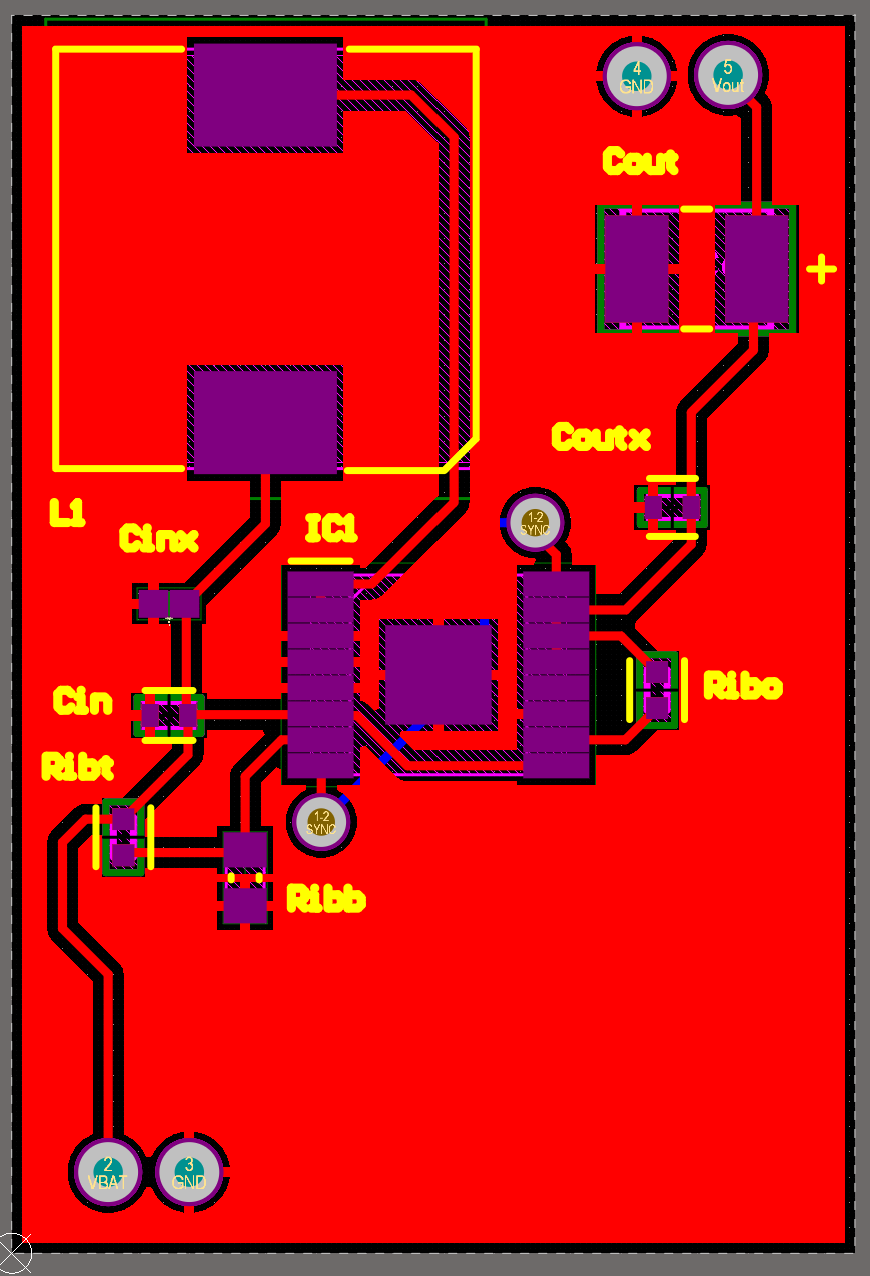
A boost converter circuit was first built on a breadboard by connecting all the components according to the schematic and the output voltage was tested to confirm that it performed according to the design. The circuit was then moved from breadboard to a perfboard for a more robust and stable design after which it is tested to validate the load and line regulation of the boost converter circuit.

The SparkFun LiPo Charger is currently what is used as a LiPo battery charger where the battery can be charged with a micro USB cable as a charger input. It uses MCP73831 charge controller IC which employs a constant-current/constant-voltage charging method. The board is set to charge the battery at 500 mA by default, however, the LiPo battery currently used has a capacity of 150 mAh and has to be charged with a rate of 1C or less. This means that the charging current should be 150 mAh or less. So, a program resistor RPROG = 2 kΩ on the board was desoldered and replaced with a resistor RPROG = 6.67 kΩ to limit the charging current to 150 mA. The formula for the program resistor and charging current is found from the MCP73831 datasheet and are calculated using the following equation:

So in order to limit the charging current of the battery to 150 mA, a program resistor should be 6.67 kΩ. The blue LED on the board turns on when the battery is charging and will turn off when the battery is fully charged.

A PCB for a boost converter circuit will replace a circuit on a perfboard in which all components on a PCB will be surface mount components. This will make the boost converter circuit to be even smaller so that it will be able to connect with the control subsystem in a housing unit that will be attached to the bottom of the cricket bat’s handle. A PCB layout was designed using Altium and will later be sent to the PCB manufacturer for fabrication and assembly next semester. The PCB layout for the boost converter circuit is shown in Figure 2.

**Figure 2:** PCB Layout for Boost Converter Circuit



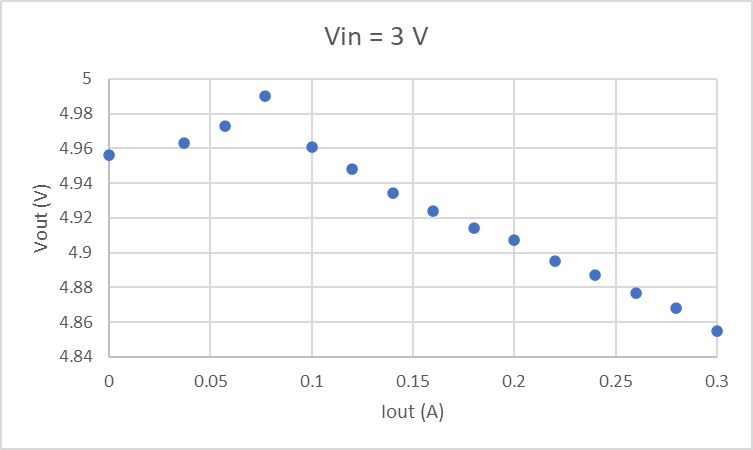
**2.3. *Subsystem Validation***

The 5V DC-DC boost converter was tested for line regulation and load regulation. Line regulation is the ability of the power supply to maintain its specified output voltage over changes in input voltage. Load regulation is the ability of the power supply to maintain its specified output voltage over changes in the load. In this case, the output voltage of the boost converter should be able to maintain around 5V. Since a LiPo battery voltage ranges from 3V to 4.2V with a nominal voltage of 3.7V, the load and line regulation were tested for input voltages of 3V which is the minimum, 3.7V the nominal voltage, and 4.2V the maximum voltage of the battery. The load current was varied from 0 to 300 mA using the electronics load equipment and the output voltages were measured. Table 1 to Table 3 shown below are the data collected from the test, and each table has a corresponding plot.

**Table 1:** Load Regulation with Input Voltage Vin =3V

| Vin (V) | Vout (V) | Iout (A) |
| --- | --- | --- |
| 3 | 4.956 | 0 |
| 3 | 4.963 | 0.037 |
| 3 | 4.973 | 0.057 |
| 3 | 4.99 | 0.077 |
| 3 | 4.961 | 0.1 |
| 3 | 4.948 | 0.12 |
| 3 | 4.934 | 0.14 |
| 3 | 4.924 | 0.16 |
| 3 | 4.914 | 0.18 |
| 3 | 4.907 | 0.2 |
| 3 | 4.895 | 0.22 |
| 3 | 4.887 | 0.24 |
| 3 | 4.877 | 0.26 |
| 3 | 4.868 | 0.28 |
| 3 | 4.855 | 0.3 |

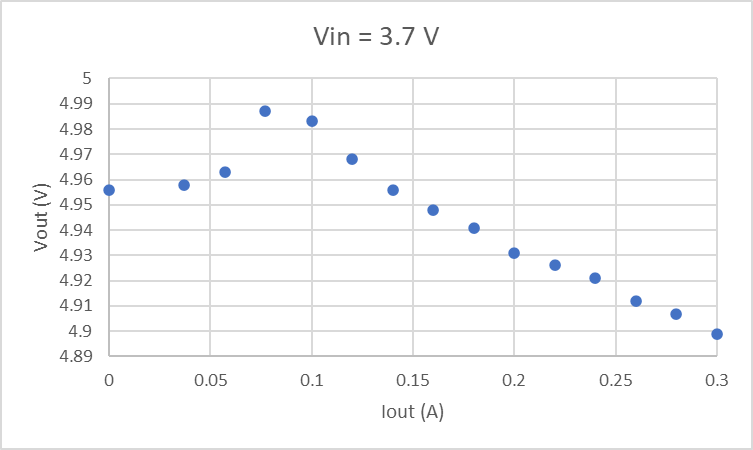
**Figure 3:** Load Regulation of Boost Converter Circuit with Vin =3V

****

**Table 2:** Load Regulation with Input Voltage Vin =3.7V

| Vin (V) | Vout (V) | Iout (A) |
| --- | --- | --- |
| 3.7 | 4.956 | 0 |
| 3.7 | 4.958 | 0.037 |
| 3.7 | 4.963 | 0.057 |
| 3.7 | 4.987 | 0.077 |
| 3.7 | 4.983 | 0.1 |
| 3.7 | 4.968 | 0.12 |
| 3.7 | 4.956 | 0.14 |
| 3.7 | 4.948 | 0.16 |
| 3.7 | 4.941 | 0.18 |
| 3.7 | 4.931 | 0.2 |
| 3.7 | 4.926 | 0.22 |
| 3.7 | 4.921 | 0.24 |
| 3.7 | 4.912 | 0.26 |
| 3.7 | 4.907 | 0.28 |
| 3.7 | 4.899 | 0.3 |

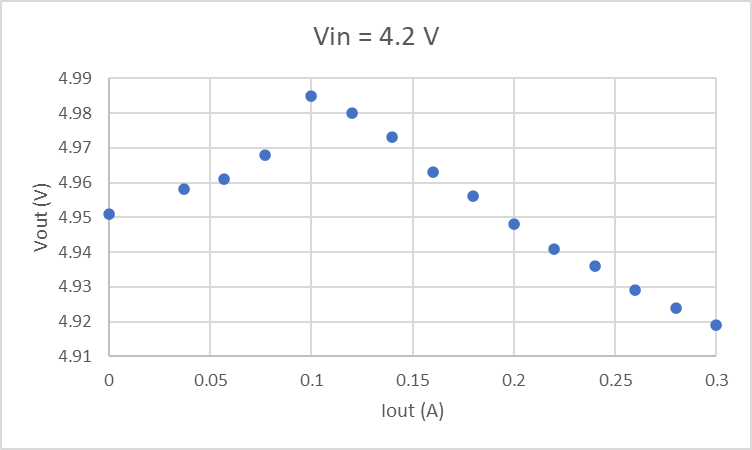
**Figure 4:** Load Regulation of Boost Converter Circuit with Vin =3.7V

****

**Table 3:** Load Regulation with Input Voltage Vin =4.2V

| Vin (V) | Vout (V) | Iout (A) |
| --- | --- | --- |
| 4.2 | 4.951 | 0 |
| 4.2 | 4.958 | 0.037 |
| 4.2 | 4.961 | 0.057 |
| 4.2 | 4.968 | 0.077 |
| 4.2 | 4.985 | 0.1 |
| 4.2 | 4.98 | 0.12 |
| 4.2 | 4.973 | 0.14 |
| 4.2 | 4.963 | 0.16 |
| 4.2 | 4.956 | 0.18 |
| 4.2 | 4.948 | 0.2 |
| 4.2 | 4.941 | 0.22 |
| 4.2 | 4.936 | 0.24 |
| 4.2 | 4.929 | 0.26 |
| 4.2 | 4.924 | 0.28 |
| 4.2 | 4.919 | 0.3 |

**Figure 5:** Load Regulation of Boost Converter Circuit with Vin =4.2V

****

As can be seen from the tables and figures above, the output voltages are very close to 5V even when the input voltages are varied from 3V to 4.2V and the load currents are varied from 0 to 300 mA. The lowest output voltage is 4.855V when Vin = 3 V and load current is at the maximum 300mA, which is still close to 5V. From the data collected, the load and line regulation were calculated. The load regulation is calculated using the following formula:

The line regulation is calculated using the following formula:

Using the above formula and the data above, the load and line regulation are

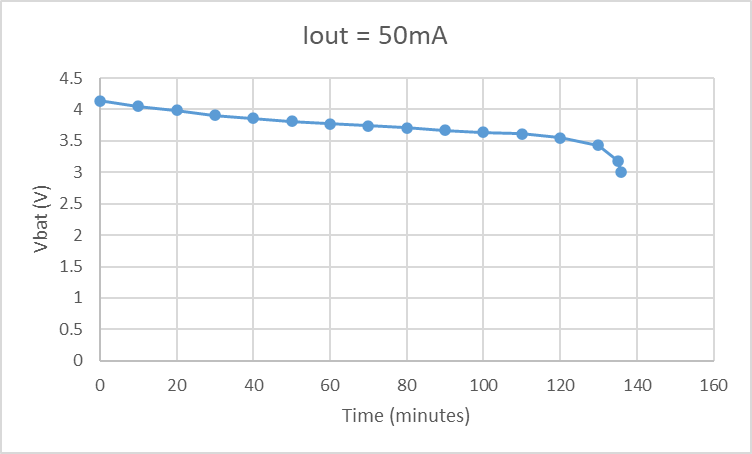
The values obtained from the load and line regulation calculations are small which means that the boost converter is well regulated. Thus, the ability for the boost converter to power the microcontroller was confirmed by the data collected and the calculations above, regardless of changes in the input voltage from 3V to 4.2V and load current from 0 to 300 mA.

In order to measure how long it can run, a LiPo battery was fully charged and connected to a boost converter circuit. The output of a boost converter is then connected to a load with current of about 50 mA. The load current of 50 mA is an estimate of the max current drawn for the entire system, including microcontroller, IMU, and bluetooth based on the information on the datasheets. The power was left running and the battery voltage was measured every 10 minutes until it went down to a cutoff voltage of 3V.

**Table 4:** Battery Voltage vs. Time with Load current =50mA

| Time (min) | Vbat (V) |
| --- | --- |
| 0 | 4.14 |
| 10 | 4.05 |
| 20 | 3.99 |
| 30 | 3.91 |
| 40 | 3.86 |
| 50 | 3.81 |
| 60 | 3.77 |
| 70 | 3.74 |
| 80 | 3.71 |
| 90 | 3.67 |
| 100 | 3.64 |
| 110 | 3.61 |
| 120 | 3.55 |
| 130 | 3.43 |
| 135 | 3.18 |
| 136 | 3 |

**Figure 6:** Battery Discharge Curve with Load current =50mA

****

Notice that the battery voltage went down from a maximum of about 4.2V to 3V in about 135 minutes or 2:15 hours when the load current is 50mA. So, the requirement that the system be able to run on a battery for at least 2 hours is validated.

**2.4. *Subsystem Conclusion***

The DC-DC boost converter works as designed and was able to raise the voltage signal from a LiPo battery to 5V at the output of the converter. A battery was able to last for more than 2 hours as specified by the sponsor and can be charged with a LiPo charger. Overall, the power subsystem works as expected and reached the goal of this semester. Next step is to send a PCB layout to a manufacturer for fabrication and assembly of the converter and to integrate the power subsystem with the control subsystem in the next semester.

**3. Control Subsystem Report**

**3.1. *Subsystem Introduction***

Since the purpose of the sensing unit is to relay swing data to the machine learning for accurate analysis, it is critical to confirm that the inertial measurement unit, and the control subsystem as a whole, operates correctly. To accomplish this, the microcontroller was programmed and was wired to establish a connection to the IMU. Once the two were interfacing, each of the two axes of the IMU, the 3-axis gyroscope and 3-axis accelerometer, were tested to validate that it performed as the manufacturer described.

**3.2. *Microcontroller***

**3.2.1. Overview**

The hardware of the control system supports interfacing between the IMU sensor and the commercial user application. And as the bridge between the two, is essential in the data collection process, so it must run continuously to support the entire system.

**3.2.2. Operation**

A Bluno Beetle BLE microcontroller is currently what is used as the control system to deliver data from the IMU sensor to the app.The Bluno Beetle has a sufficient number of pins, supply voltages, and bluetooth modules to meet the requirements of the Smart Cricket Bat.

The Bluno Beetle BLE has 4 Digital I/O pins, 4 Analog Input pins, and a pair of Serial Clock Line and Serial Data Line pins. It is also able to provide a 5V output supply to power the IMU. This is more than enough to interface with the single IMU sensor as it only needs power, a pair of SDA and SCL pins, and a single digital pin for the interrupt. The Bluno Beetle can be powered on a range of 5 to 8 V, which will be supplied by the power subsystems boost converter.

The key feature of the microcontroller is to function as the bridge between the sensor data and the application that holds the machine learning algorithm, it must be able to maintain this connection for the duration of a training session. Bluetooth will be used to effectively do this, currently using standard bluetooth before trying to establish the low-energy bluetooth 4.0 module within the microcontroller. Code for this operation was written and successfully tested as part of the other subsystems.

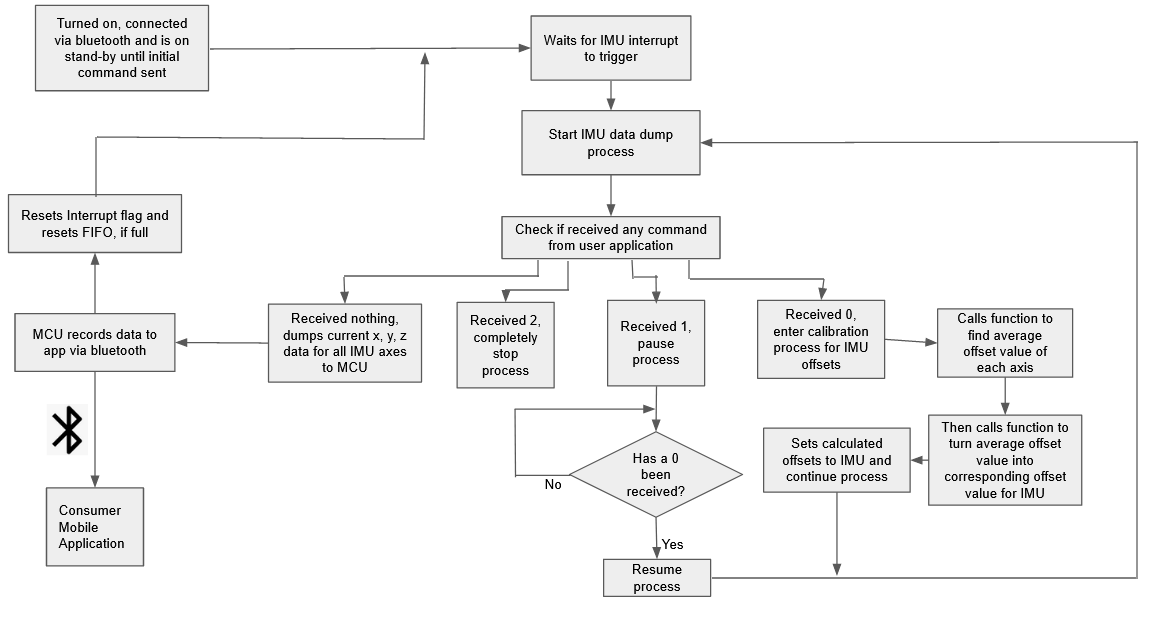
A PCB will later be added to replace the perfboard prototype and provide more secure connections between the power and control subsystems.

**3.2.3. Validation**

The control system was validated to meet all requirements. Since its primary function is to communicate with the application subsystem, it was validated by first linking and pairing with the manufacturers recommended application for bluetooth to confirm the low-energy bluetooth module onboard the MCU was operating as intended. Then to confirm the code written for the MCU works as intended with standard bluetooth, a dummy control system was made with a standard HC-05 module and connection was established and information was both sent and received. Integration will involve moving from standard bluetooth to low-energy bluetooth.

The flowchart in figure 7 shows the logical operation of how the microcontroller goes about gathering the IMU data, interpreting user input, and delivers the data to the consumer application.

**Figure 7:** Flow Diagram of Microcontroller Logic



Something not shown in the flowchart is the fact that for the IMU, the interrupt is constantly triggered after the first trigger, so within the code is a buffer step that checks for more data dumps from the IMU before the MCU checks the interrupt again allowing for quicker revolving of data. However, this on paper functions the exact same way as checking the interrupt each time, so it can be thought of the same way to avoid unnecessary confusion.

This logic was successfully executed on the Bluno Beetle with the validated IMU sensor results in turn validating the microcontroller. The only thing remaining unvalidated is the communication between the MCUs low-energy bluetooth module and the app, which will be done during the integration stage.

**3.3. *Gyroscope***

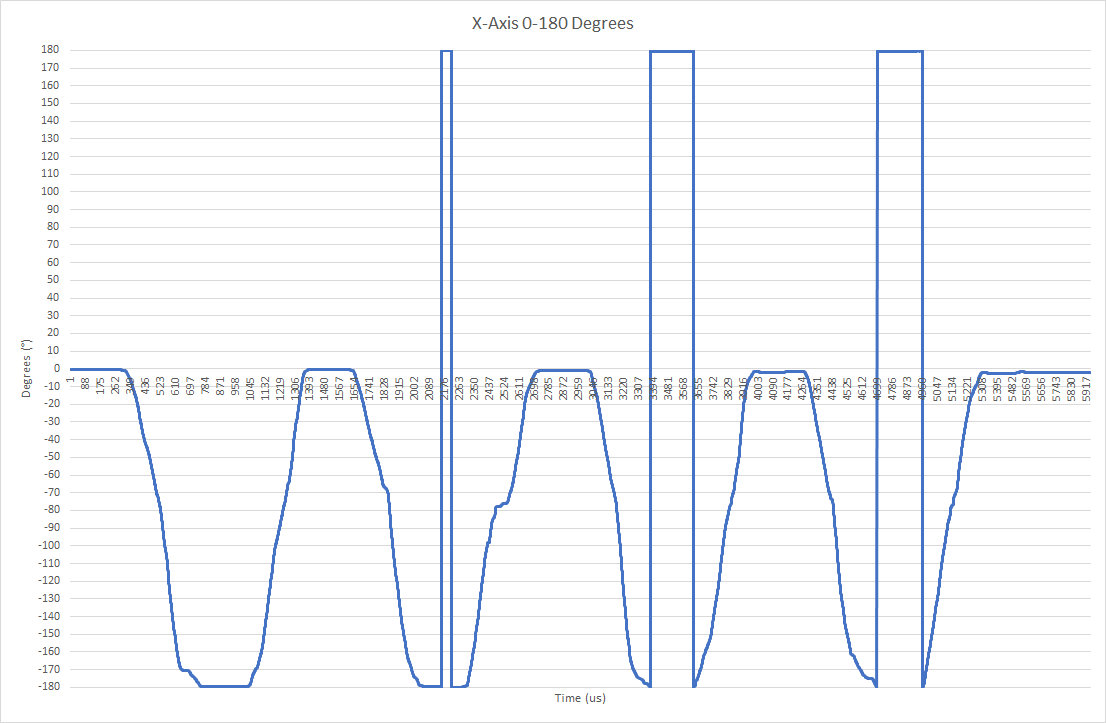
**3.2.1. Operation**

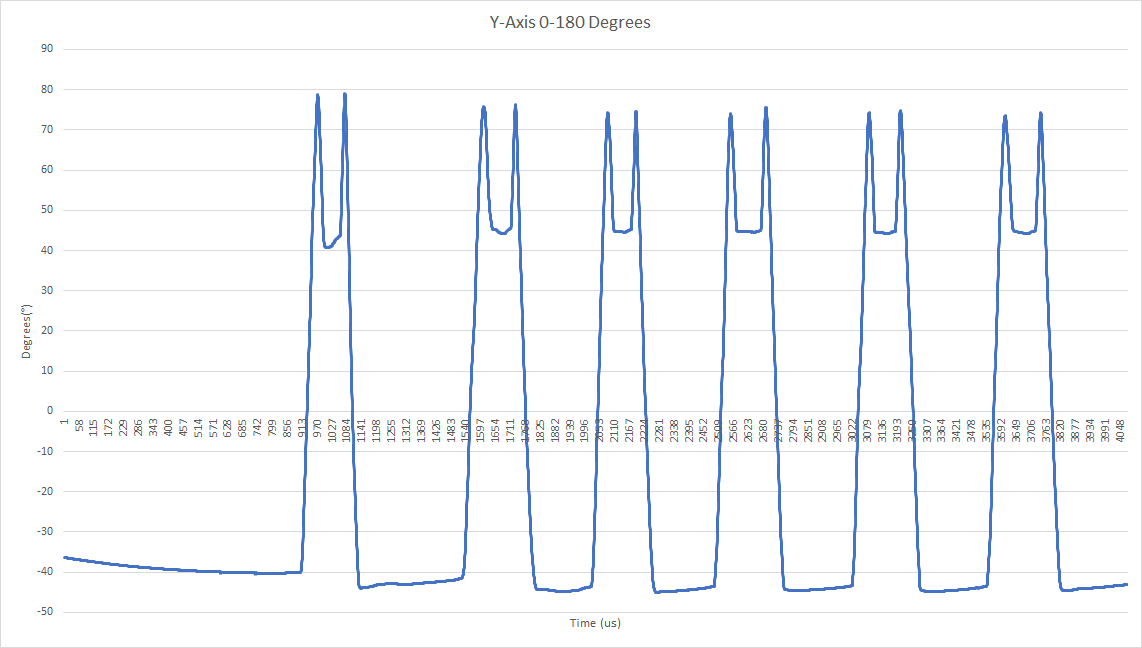
The inertial measurement unit used is a gy-521 board that uses an MPU-6050. The MPU-6050 has both a gyroscope and accelerometer inside, this was chosen for two option, one was to save on space as the design must stay small and unobtrusive, and two, its best if the gyro and accelerometer axes are lined up and this is are to do with two separate devices. For the gyroscope, it has 4 full scale ranges, ±250, ±500, ±1000, ±2000 °/sec with the sensitivity being 131, 65.5, 32.8, 16.4 LSB/°/sec, respectively. For the purposes of the Smart Cricket Bat, the range will stay at the default range of ±250 °/sec, which has a noise rate of 0.005 mdps/rtHz.

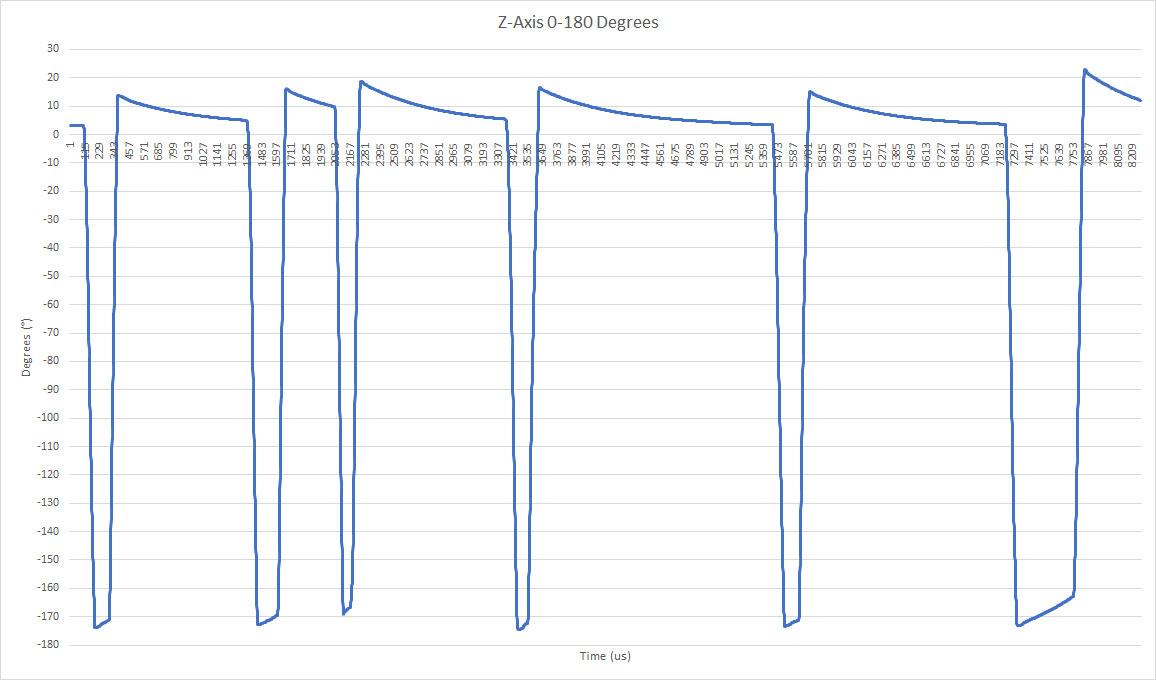
The gyroscope uses an I2C interface, which allows the microcontroller to take the measurements in via the serial clock and serial data lines. This is the standard way I2C devices communicate, by transmitting data that are 9 bits long along these 2 lines.

**3.2.2. Validation**

The gyroscope’s angle accuracy was validated by attaching the gyroscope to the center of a protractor and rotating the protractor to the desired degree and comparing that to and confirming that the gyroscope's readings were accurate. This was done for each axis of the gyroscope and was done from angle ranges of 0-45°, 0-90°, 0-135°, 0-180°, 0-215°, 0-270°, 0-305°, 0-360°. Below are some of the graphs of the received gyro values as each axis is rotated from 0-180°, all 21 other angle graphs were also made but not shown here for space purposes.

**Figure 8:** X-Axis Gyroscope Validation

**Figure 9:** Y-Axis Gyroscope Validation

**Figure 10:** Z-Axis Gyroscope Validation

Notice, since the graphs for the y and z axes drift a bit more, which causes them to lose track of where they started, being a consistent 5° and 10° off on return, respectively. However, this won’t be an issue because, one, we will primarily be using the x and z axes to find the angle of the bat on impact, which has the least drift. And two, since most swings won’t start at 0° anyway, the machine learning algorithm will already be subtracting the starting degree from the finished degree to get the true angle of the bat.

Through extensive validation, the gyroscope within the IMU is confirmed to be working as intended and gives the accurate degrees of change for the system. However, replacing the gyroscope (and by extension the IMU) with one with more consistency, less tendency to drift, and overall less noise is being considered.

**3.4. *Accelerometer***

**3.4.1. Operation**

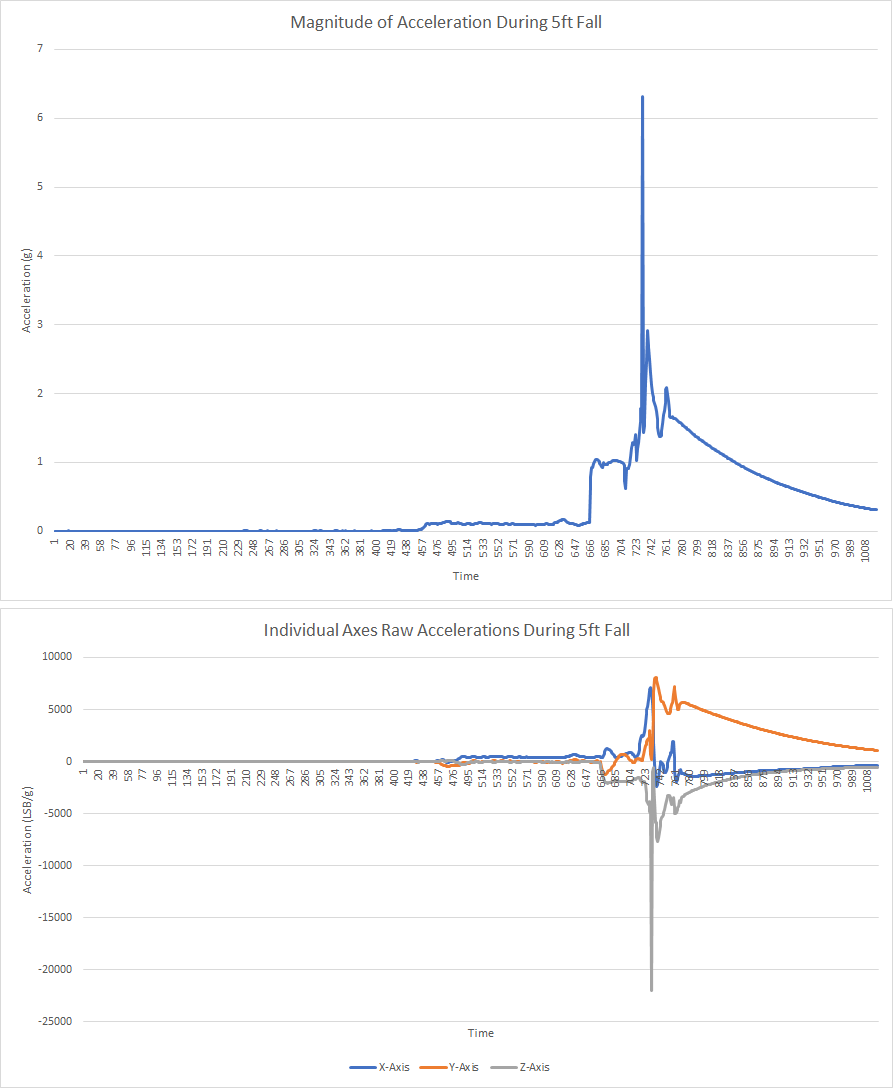
The inertial measurement unit used is a gy-521 board that uses an MPU-6050. For the accelerometer, it has 4 full scale ranges, ±2, ±4, ±8, ±16 g with the sensitivity being 16384, 8192, 4096, 2048 LSB/g, respectively. The raw data received from the accelerometer is in terms of least significant bit per g (LSB/g) and must be divided by the sensitivity to get the data in terms of g’s. For the purposes of the Smart Cricket Bat, the range will be the range of ±8 g, which has a noise rate of 0.005 mdps/rtHz. This range and sensitivity may be subject to change as we integrate the subsystems, it depends on what gets the best results out of the machine learning algorithm.

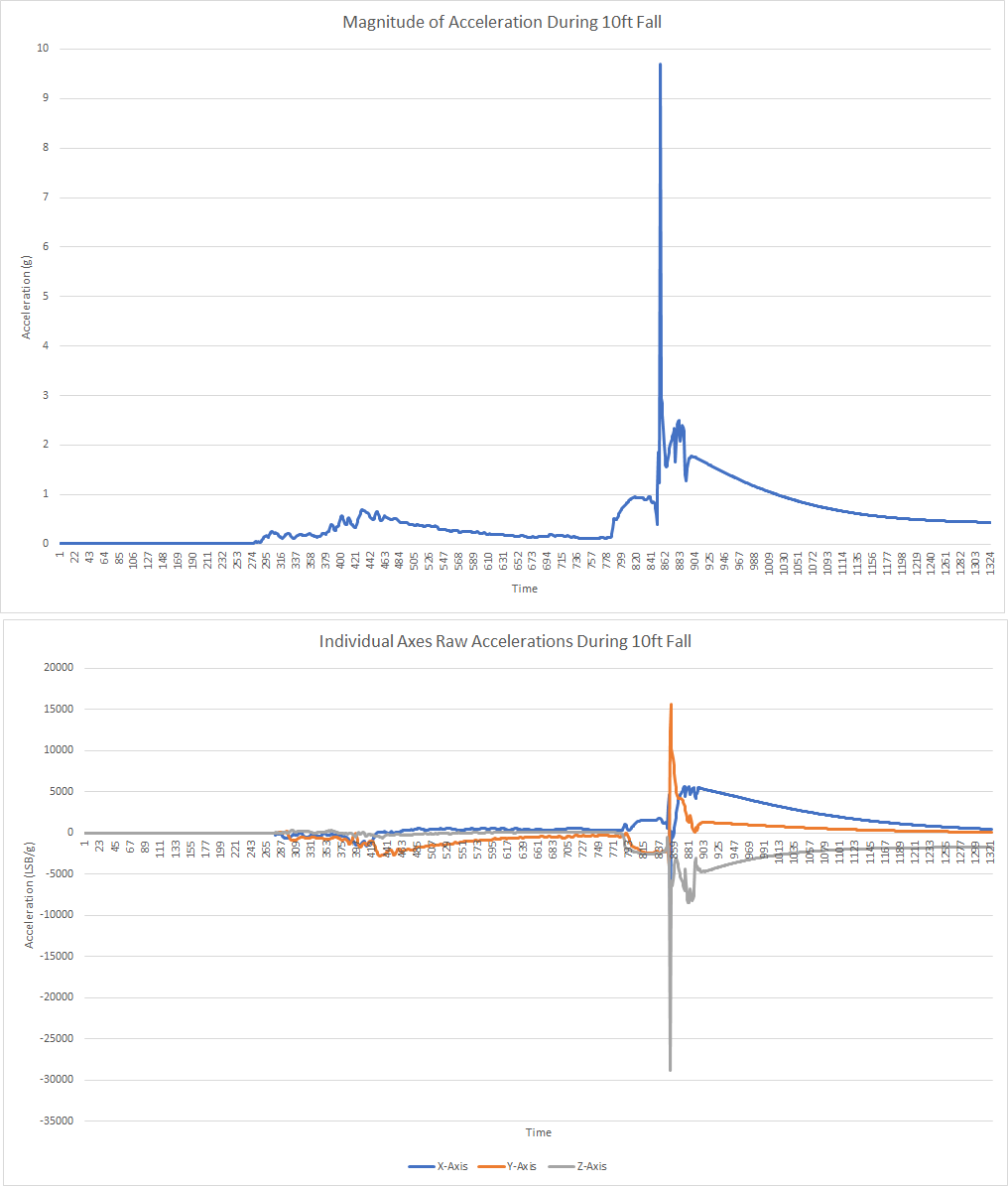
The accelerometer uses an I2C interface, which allows the microcontroller to take the measurements in via the serial clock and serial data lines. This is the standard way I2C devices communicate, by transmitting data that are 9 bits long along these 2 lines.

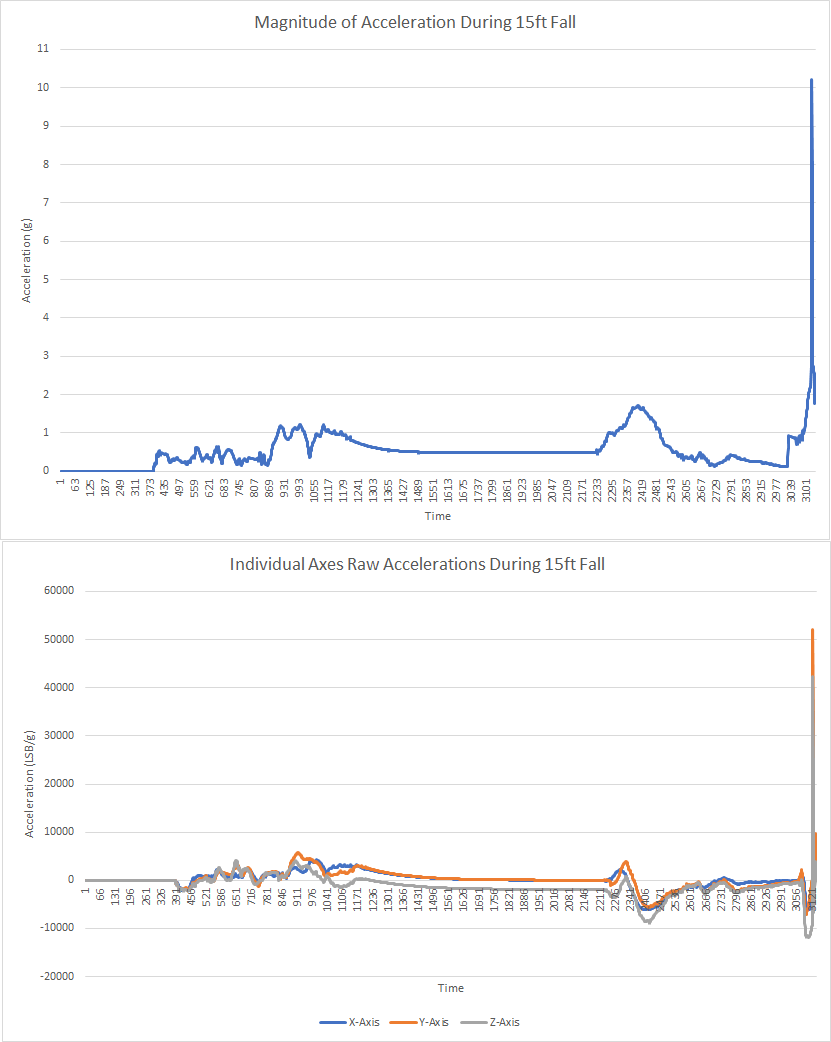
**3.4.2. Validation**

To validate the accelerometer, the device was dropped from 4 different heights ranging from 1 to 15 feet. The heights were restricted to a max of 15 feet because the bluetooth module cannot currently be used and the power system is not in a state to be dropped from so high, so the length of the microUSB being used to power and receive info from the device is its limiting factor. There were also problems with the microUSB’s connection being unstable after the first couple of drops, so the data would sometimes stop right after the collision, specifically in the case of 15 feet. With this in mind, the following two kinds of graphs are produced from the acceleration of the x, y, and z axes when falling and hitting concrete from heights at 5, 10, and 15 feet. The first being the average magnitude of the acceleration of the entire unit during the fall and the other is the raw accelerations for each axis.

**Figure 11:** 5 Foot Fall Accelerometer Validation



**Figure 12:** 10 Foot Fall Accelerometer Validation 

**Figure 13:** 15 Foot Fall Accelerometer Validation 

Compare these values to what the expected acceleration during the collision should be, based on calculations done using an estimated collision time of 10ms and the assumption that the device loses about 75% of its acceleration after the collision. The values obtained are all about 10 times larger than the values received from the IMU, there could be a few reasons for this. One, the IMU automatically removes gravity from given values which may affect total gs, two, the calculations were done incorrectly leading them to be larger, or three, the IMU is giving wrong values. However, out of these options the most likely to affect the validation process is the removal of gravity, as it is unclear how this factors into the calculations, and could be the cause for the calculated accelerations being almost a factor of 10 larger.

**Method Used to Calculate gs**

**Table 5:** Accelerometer Measured Vs Calculated Acceleration

| Dropped Height | Calculated Acceleration | Measured Acceleration |
| --- | --- | --- |
| 5 ft | 69.673 g | 6.3214 g |
| 10 ft | 98.534 g | 9.6987 g |
| 15 ft | 120.6804 g | 10.223 g |

After comparing the two data sets it can be inferred that, despite the values from the accelerometer being off by about a multiple of 10, the accelerometer gives consistent data that is representative/follows the trend of the overall changes in acceleration. This is more than passable for the purposes of the Smart Cricket Bat, as it only needs the relative gs enacted on the bat by the ball in a given section, relative to the other sections. However, replacing the accelerometer (much like the gyroscope) with one with more consistency, accuracy, and sensitivity is being considered.

**3.5. *Housing Unit***

The housing unit consists of 2 elements, the mounting mechanism and the housing unit for the power system and control system. The current housing unit prototype is 3-D printed with PLA material and is 65.5 mm in length and 51mm in radius. The prototype, primarily being used to collect data for the machine learning, holds the perfboard MCU and IMU design, has a small slit to power the MCU, and is attached to the handle via 4 pronged teeth that easily slide onto the bat and clip to it for security. Through the process of collecting data, the prototype has proven to be robust enough to withstand the average hits and shocks that are to be expected from a cricket bat while in use. The prototype is also fairly unobtrusive to the user when swinging, which is a high priority for the final design.

For the final housing unit design, the same material and mounting mechanism will be used, however the overall size and design of the control and power system housing will be changed. Changing from perfboard to PCBs will allow for a smaller compartment and a sleeker design overall. It will be approximately half the prototype's height, will contain a single slot that allows the device to be charged when not in use, and a power button to turn the device on and off.

**3.6. *Subsystem Conclusion***

The Bluno Beetle BLE microcontroller works as designed and is able to utilize a classic bluetooth module to link and pair with a mobile device. Each of the sensors on the MPU-6050 operate as designed. Despite some small discrepancies in the behavior of the IMU, the sensors function well within the scope of what the Smart Cricket Bat will need them to. However, the decision to switch to a more robust IMU is still an option, and it depends upon the needs of the machine learning. The microcontroller and sensors are a critical part of the overall system, and their ability to integrate with the consumer application ensures that they will be able to continuously collect data for processing.

**4. App Subsystem Report**

**4.1. *Subsystem Introduction***

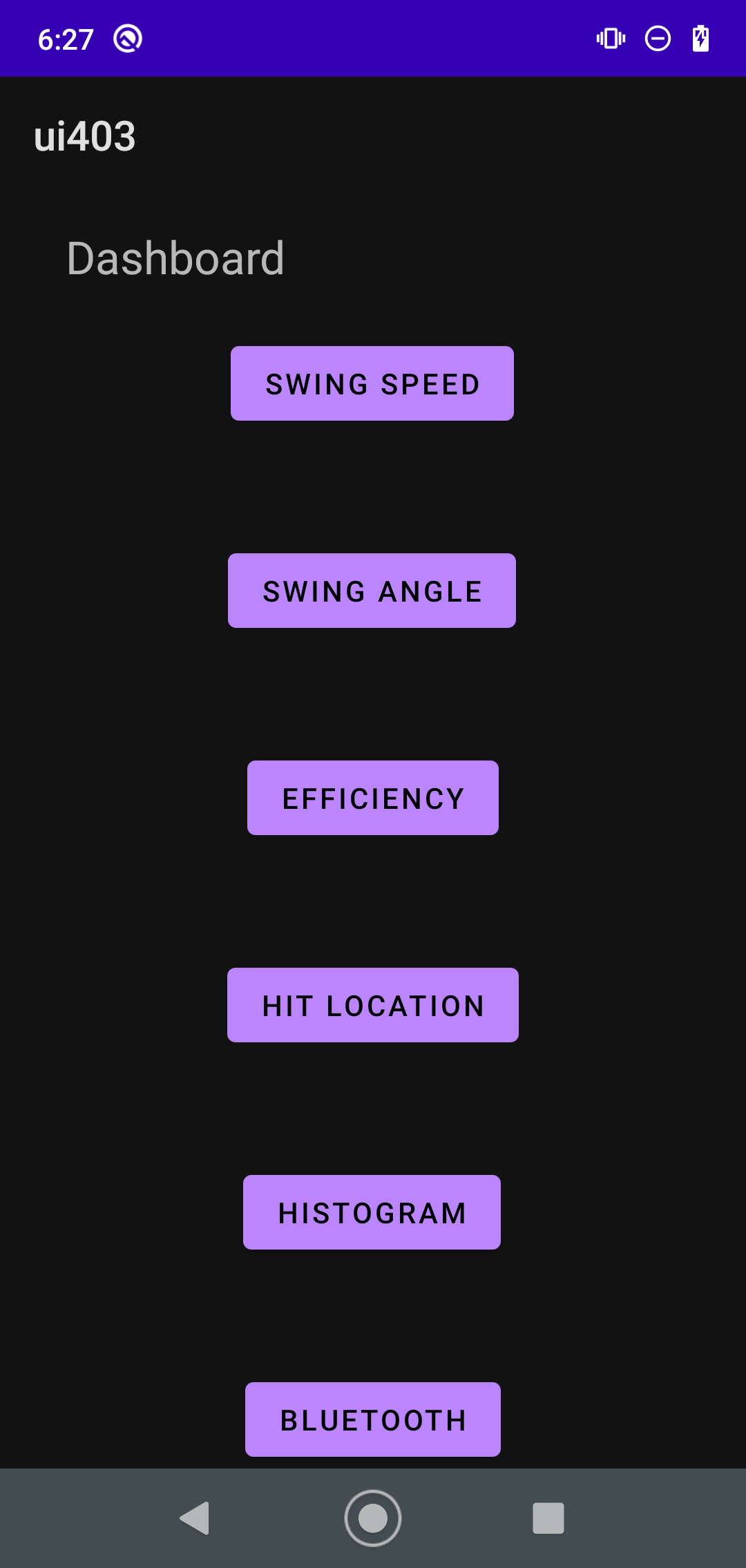
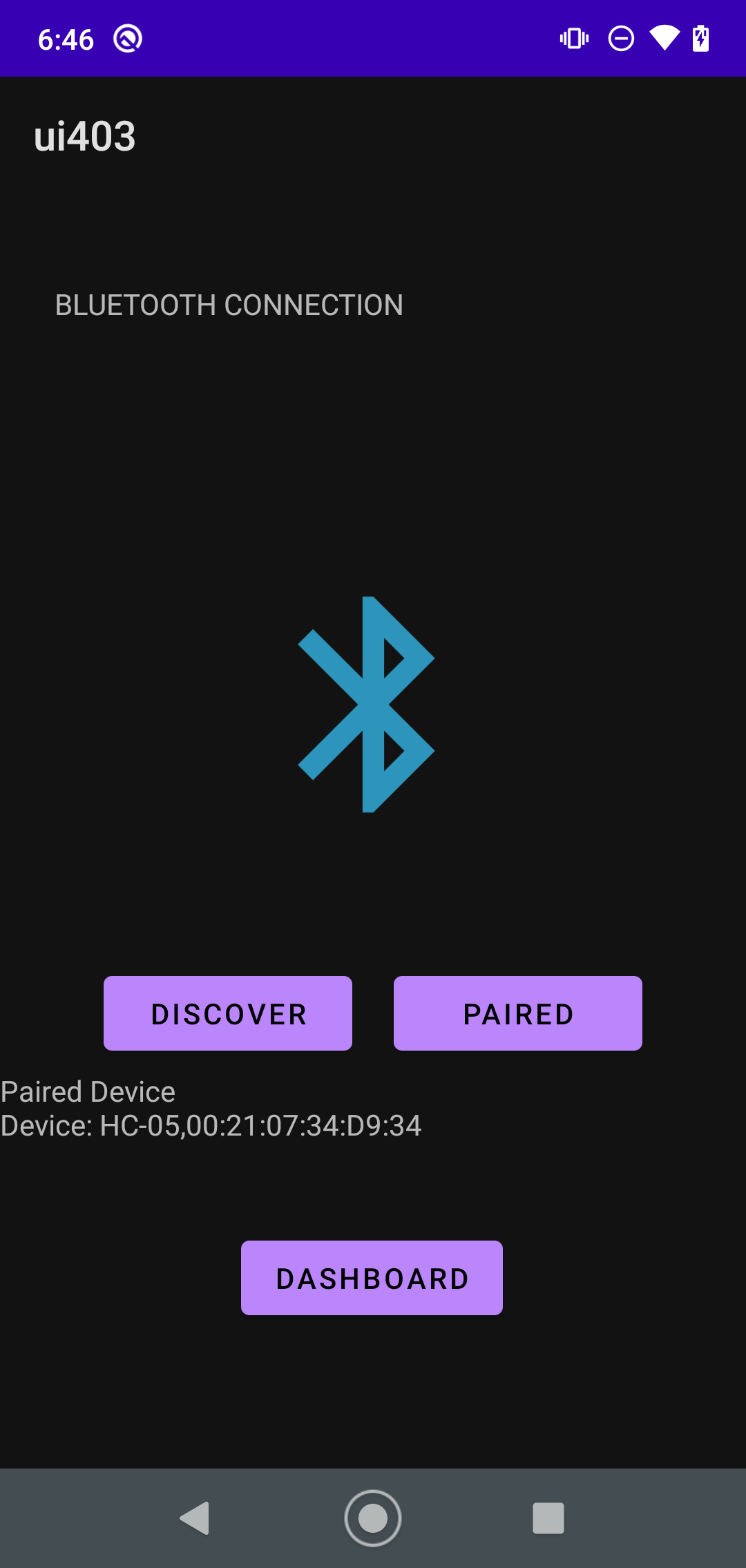
The android app has two main functions: communication between the control subsystem and the ML algorithm and handling user interaction. The app will receive the swing data gathered by the MCU via a bluetooth connection which will then be sent to the ML algorithm, that will run on the app, to be processed. Once processed, the user can navigate through different pages of the app by clicking various buttons on the dashboard to view their results. Each page will consist of a certain data output i.e. hit location, efficiency, or swing speed. The app is user friendly and does not require any advanced technical skill to allow any Cricket player to use our product.

**4.2. *Subsystem Details***

**4.2.1. Bluetooth Communication**

As previously mentioned, the android app will handle the communication between the ML algorithm that will run on the app and the control subsystem via a bluetooth connection. To establish a bluetooth connection between the MCU, we will be using an HC-05 Bluetooth module. After each swing the user will press the designated bluetooth button on the dashboard that will then send a signal to trigger the MCU to send the swing data. An image of the dashboard and bluetooth page is shown below. The user is now free to navigate through the app to view their results. The Bluetooth page will also allow the user to make sure they are connected to the correct device by displaying its name and MAC address.

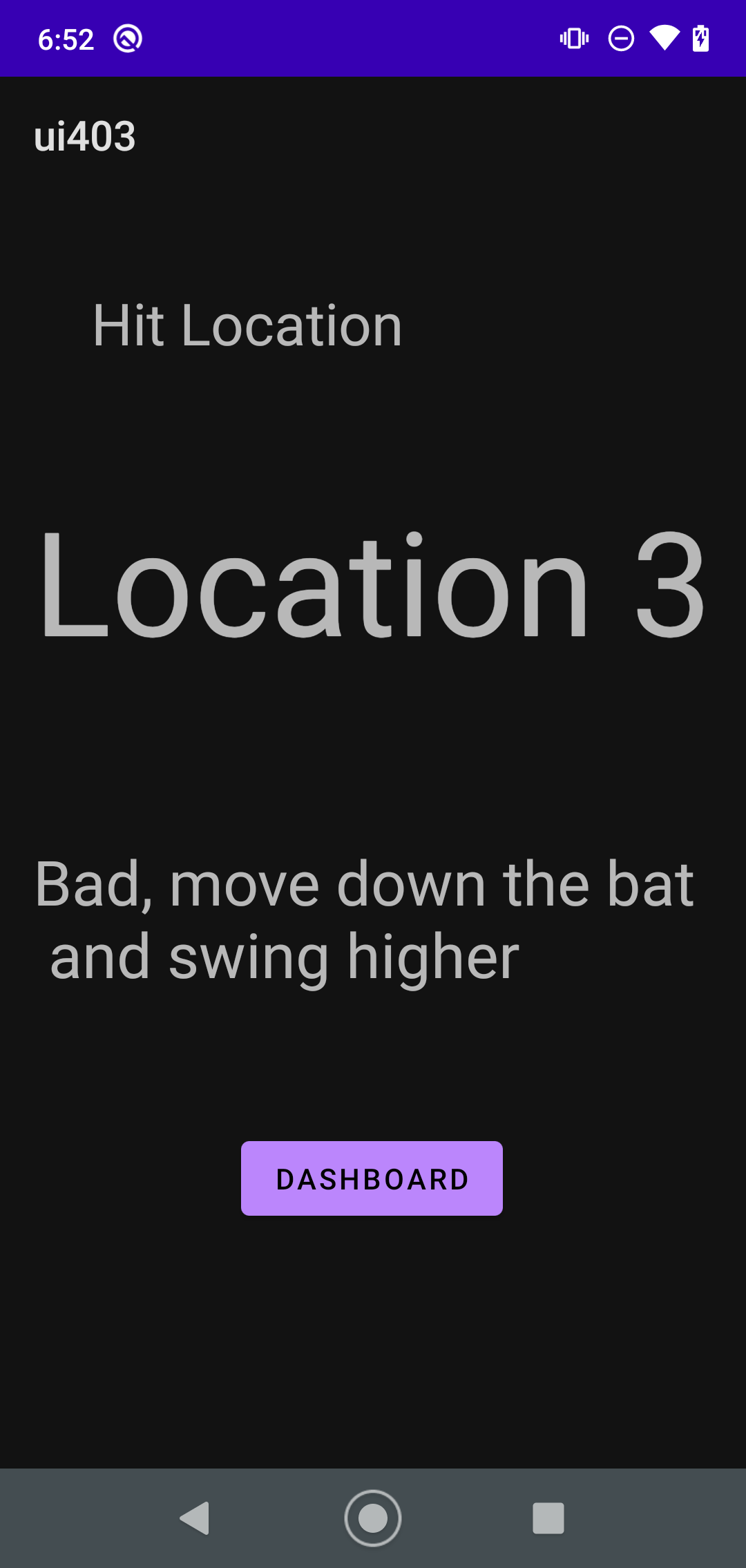
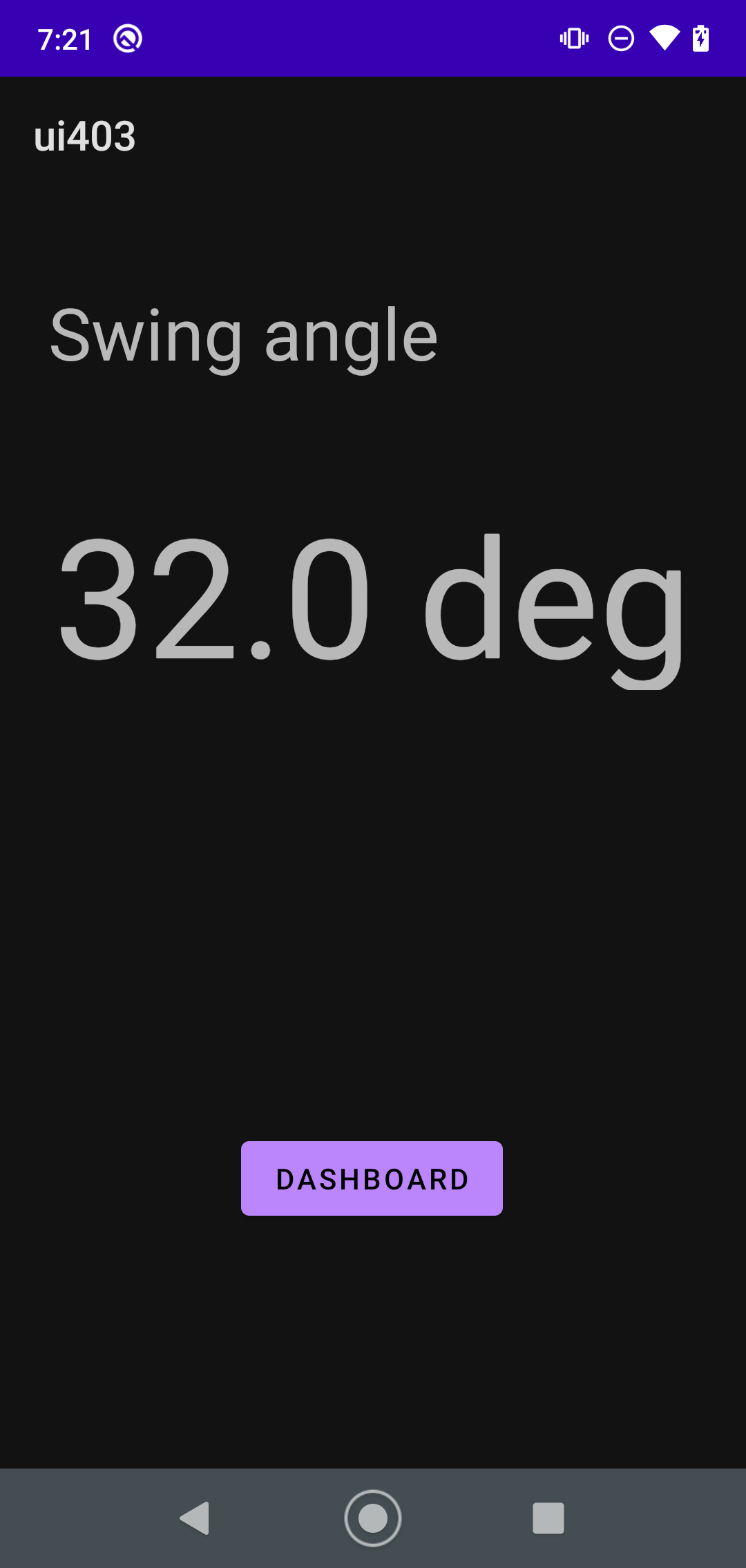
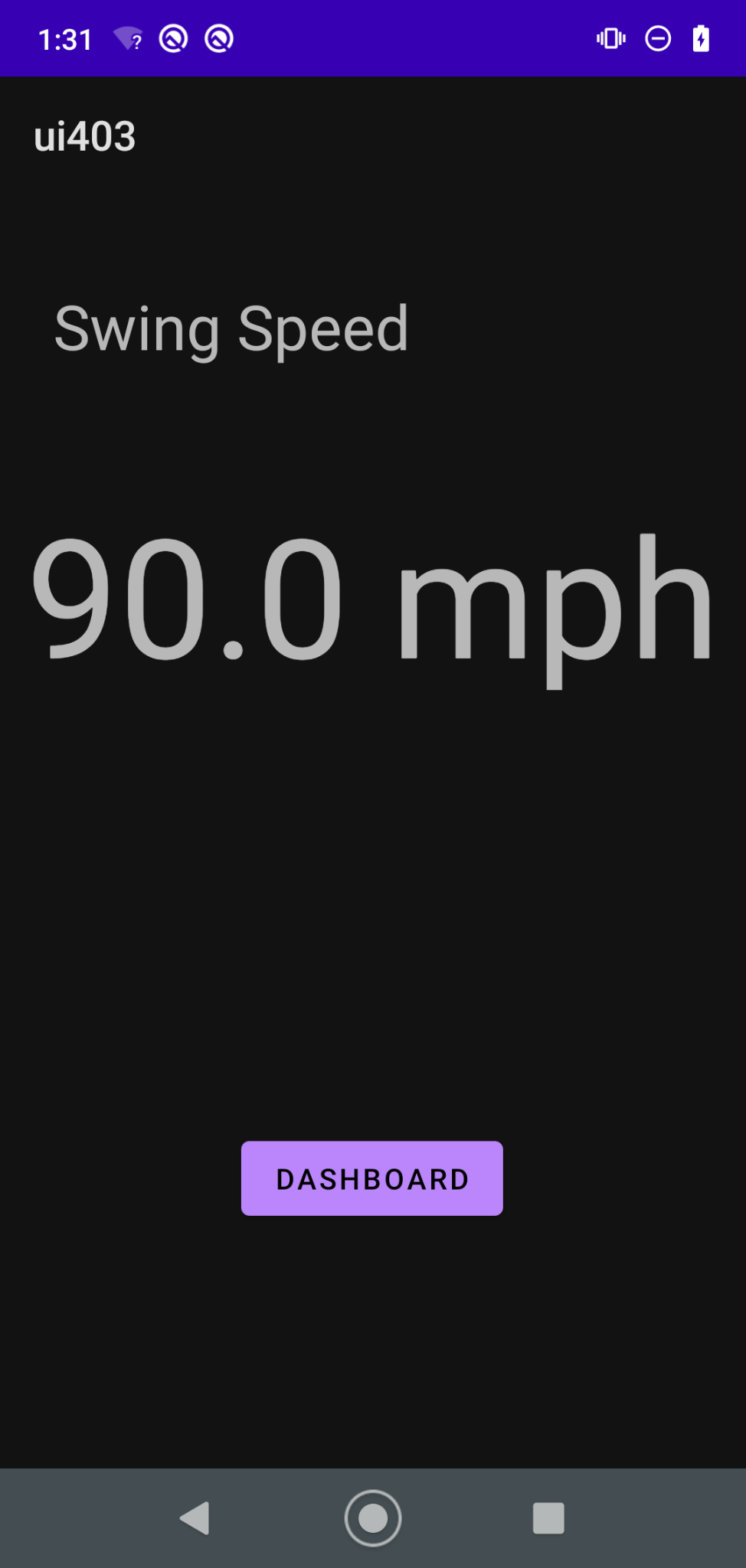
**Figure 14:** Bluetooth UI

**4.2.2. User Interaction / Dashboard**

The user interaction is controlled by the dashboard seen above. Each button is named corresponding to the result type it will output. The dashboard was designed to gather all the results calculated by the ML algorithm and send them to the individual pages once the user presses a button. The result types are as follows: Swing speed, Swing angle, Efficiency, Hit Location, and a heat map showing where the user is predominantly hitting the bat. The most important page is the hit location. In this page the user will be given the region where the collision with the ball occurred. The regions are numbers 1 through 15, where 1 is near the tip of the bat and 15 is closer to the handle of the bat. These regions will be marked on the bat. Depending on which region the ball hit, the user will be given advice as to how to get closer to the “sweet spot'' of the bat. An example of the swing results being displayed in their corresponding page is shown below. In this example the user struck the bat near the tip of the bat on the upper edge, therefore the advice in the hit location page states “move down the bat and swing higher”. The location was above the sweet spot and on the upper edge, so the player needs to swing higher and get the collision lower on the bat to get closer to the center of the bat. The user can then navigate back to the dashboard by clicking on the dashboard button displayed on each page.

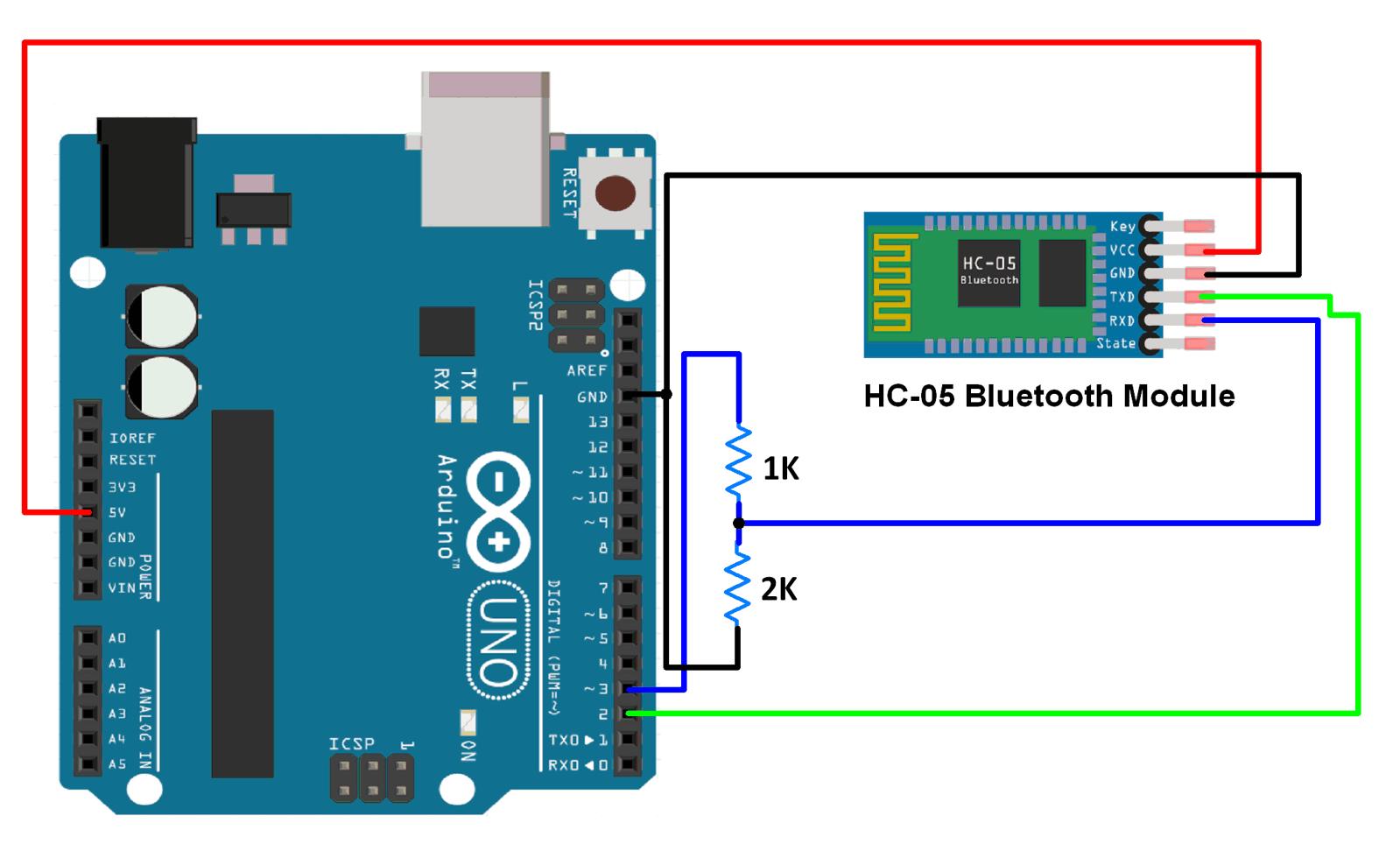
**Figure 15:** User Feedback

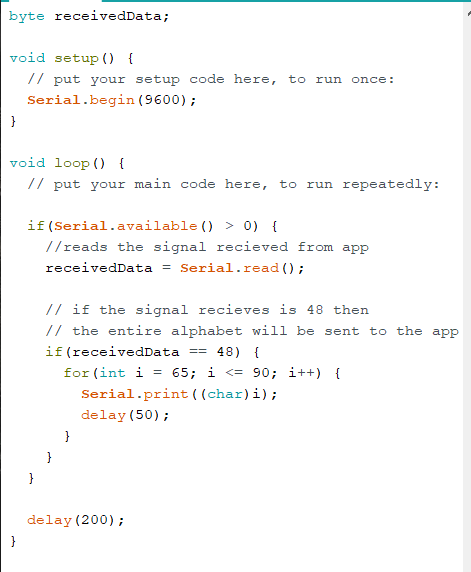
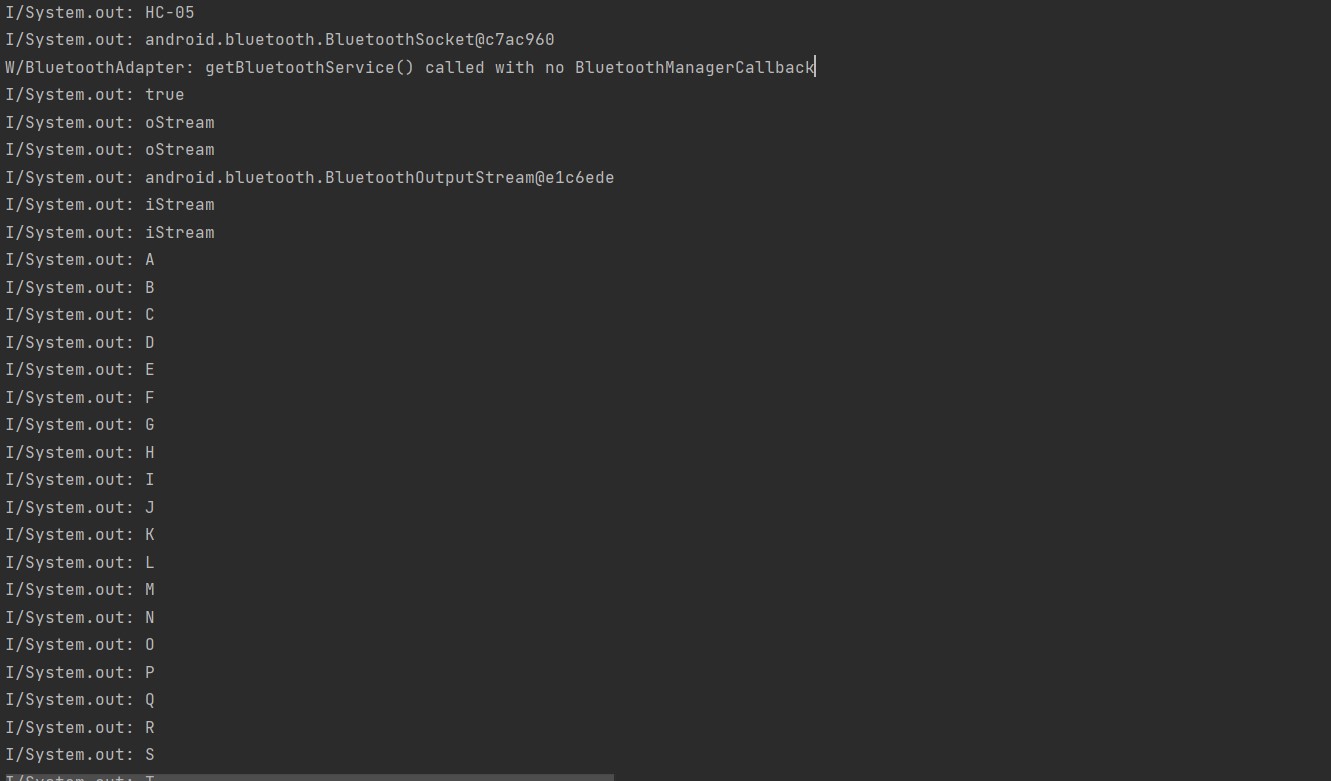
**4.3. *Subsystem Validation***

Since the MCU, the Bluno Beetle, uses the Arduino IDE and schematic, I separated the HC-05 bluetooth module and connected it to an Arduino Uno to be able to validate without needing the control subsystem. The circuit that will be used to connect the HC-05 module with the Arduino and Bluno Beetle is shown below. Since the TTL levels of the arduino is 5V while the HC-05 module TTL level is 3.3V, a voltage divider with a resistance ration of 2:1 is used to attenuate the signal level.

**Figure 16:** HC-05 Bluetooth Module Circuit



To validate the bluetooth connectivity and data sending between the MCU and the android app, the MCU was programmed to send the entire alphabet once the app sent an input of 48 signifying the ASCII code for the character ‘0’. An image of the console output showing data being received by the android app is shown in Fig. 17. Although data was able to be sent via bluetooth, random data reading errors occur when first connecting the bluetooth module. The console also shows which device the app is connected to and whether we are connected to the bluetooth socket created. An image of the simple arduino code is also shown in Fig. 17.



**Figure 17:** Bluetooth validation data and arduino code

**4.4. *Subsystem Conclusion***

The android app is easy to use without the need of an advanced technical skill. The main skill the user will need to know is how to pair their phone to a bluetooth device in their device’s settings menu. Once paired, everything will be available to the user with a simple click of a button. The android app is also able to communicate and receive data from the MCU via a bluetooth connection, although there are some random reading errors that occur when first booting up the device. A link to the github repository containing the app source code is given below.

<https://github.com/pablobarron7/Capstone_App.git>

**5. Machine Learning Subsystem Report**

**5.1. *Machine Learning Introduction***

The Machine Learning subsystem for this project is returning the region that the ball and bat impact to the user with the input of swing data. The method we are using is to divide the bat into different regions and do test swings on each region. Then ML will take the data from IMU and process them to give feedback of which region the impact happened. Then the users can look at the regions of their hit so that they can improve with their training and try to hit their “Sweet Spot” that they are looking for.

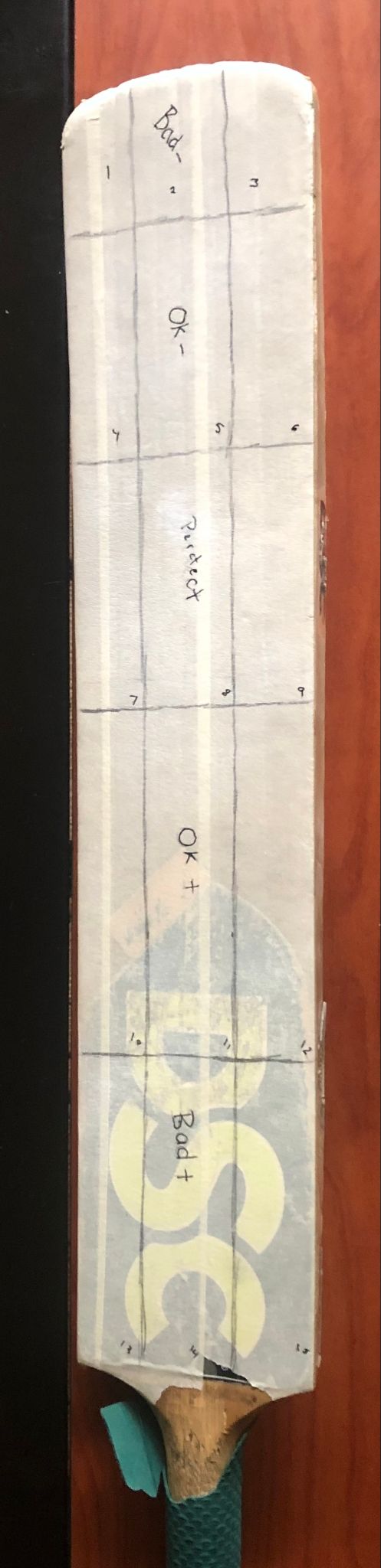
**5.2. *Machine Learning Details***

The Machine Learning subsystem is done on python jupyter with tensor environments.

**5.2.1. Bat Region Dividing**

Based on the research, different regions of the bat provide different jerks when impact, which results in different eigenfrequency for us to determine. Based on that, we divided the bat into 15 regions like the image showing below. We marked the thiccest portion of the bad into “perfect” areas which is the “Sweet Spot” that batters usually refers to. For the rest of the regions, we named them after Bad-, OK-, OK+, and Bad+. Bad- is the tip of the bat which is where batter usually avoids while Bad+ is also a region that should be avoided by batter since it is close to the handle. The OK regions are closer to the sweet spot. They are better spots to hit, but they are not optimal. We also mark + and - to divide the region where + means impact region is closer to the handle and - means impact region is closer to the tip. Then we also marked all the regions as region 1-15 for processing in ML.

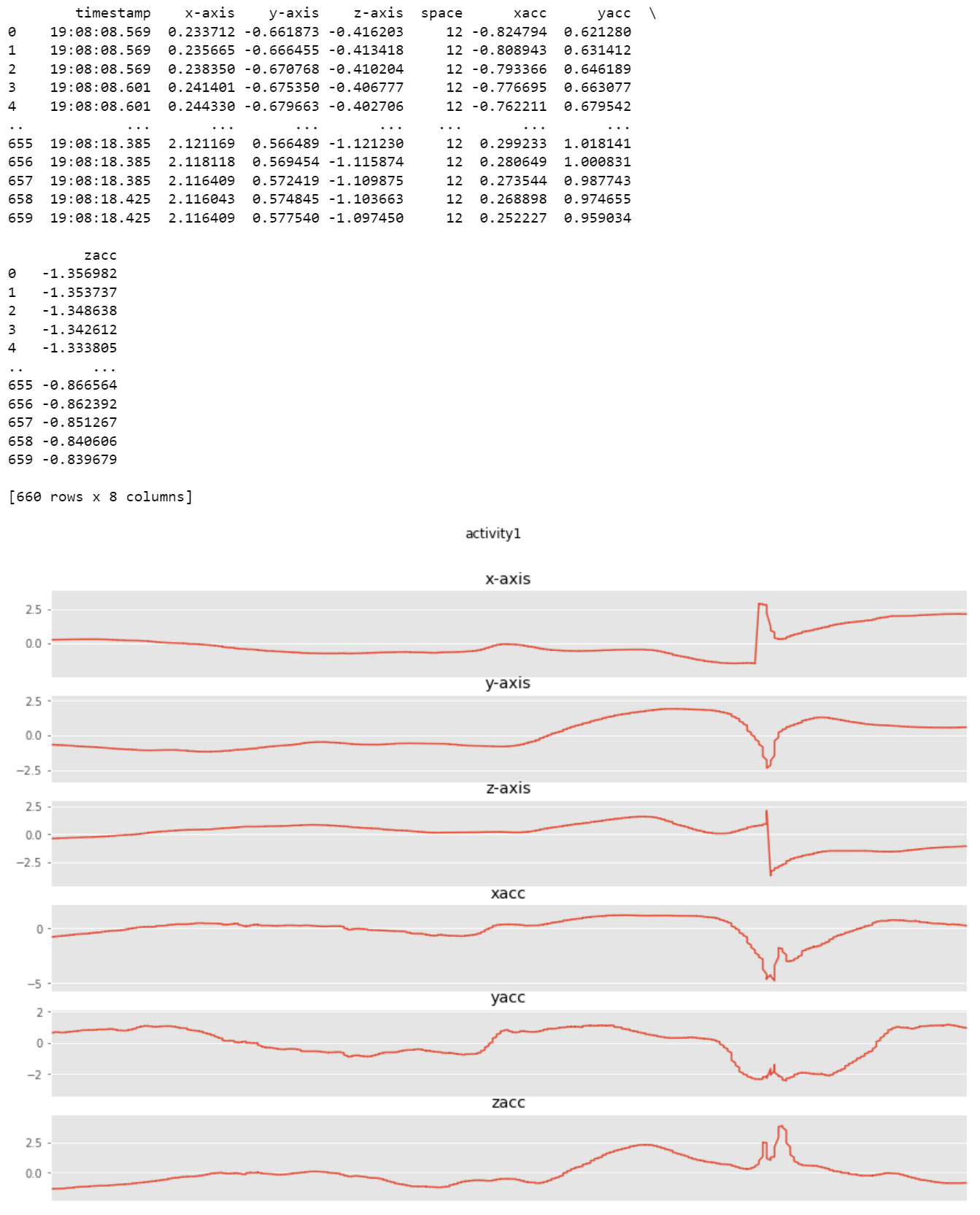
**Figure 18:** Bat Division



**5.2.2. Data Gathering, Inputting, and Plotting**

We did from 3 to 6 swings on each region and found out a total of 74 sets of data are usable. The ML takes the data gathered as input, then it does normalization on the data and plotting data over the entire duration out. The example figure shown below and all the examples later are from a set of data that corresponds to an impact on region “OK+12”.

**Figure 19:** Data Plotting

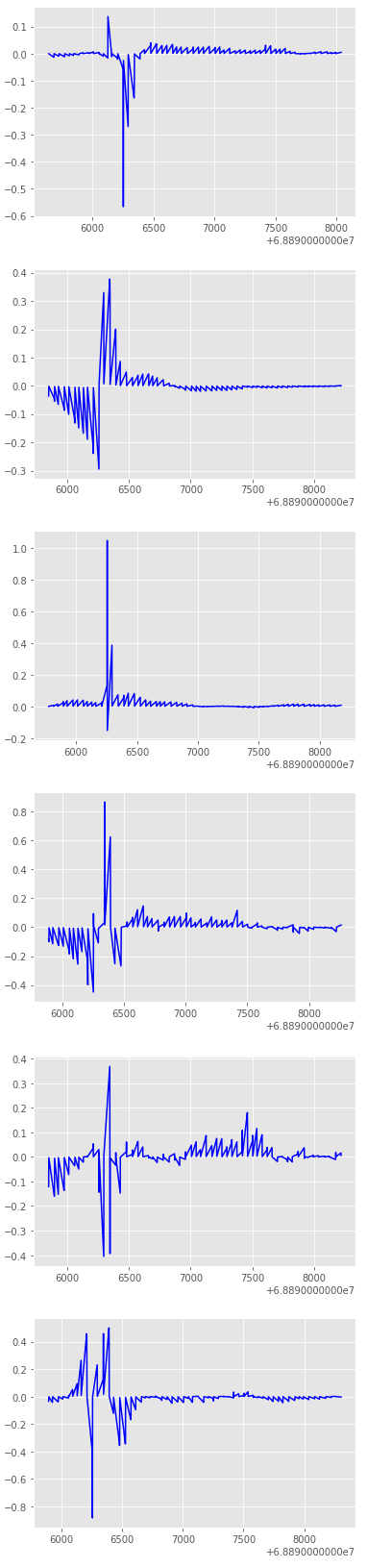
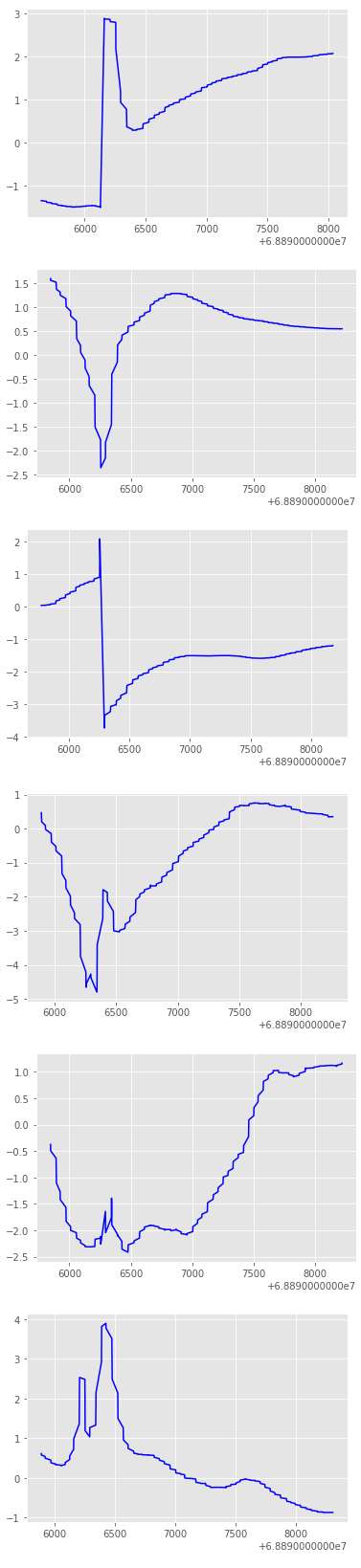


In the image, “x-axis”, “y-axis”, and “z-axis” are corresponding to the gyroscope and “xacc”, “yacc”, and “zacc” are corresponding to the accelerometer.

**5.2.3. Finding the Impact**

The ML took a derivative over the data and found the max point on the derivative to find out where the impact actually happens. From the max point on the derivative, we took from 32 values before the max to 128 values after the max to define the time window as impact times. Then it finds the same place on the original data to find the impact window. In the figure below, the left graphs are the original, and the right is its derivative.

**Figure 20:** Impact Window

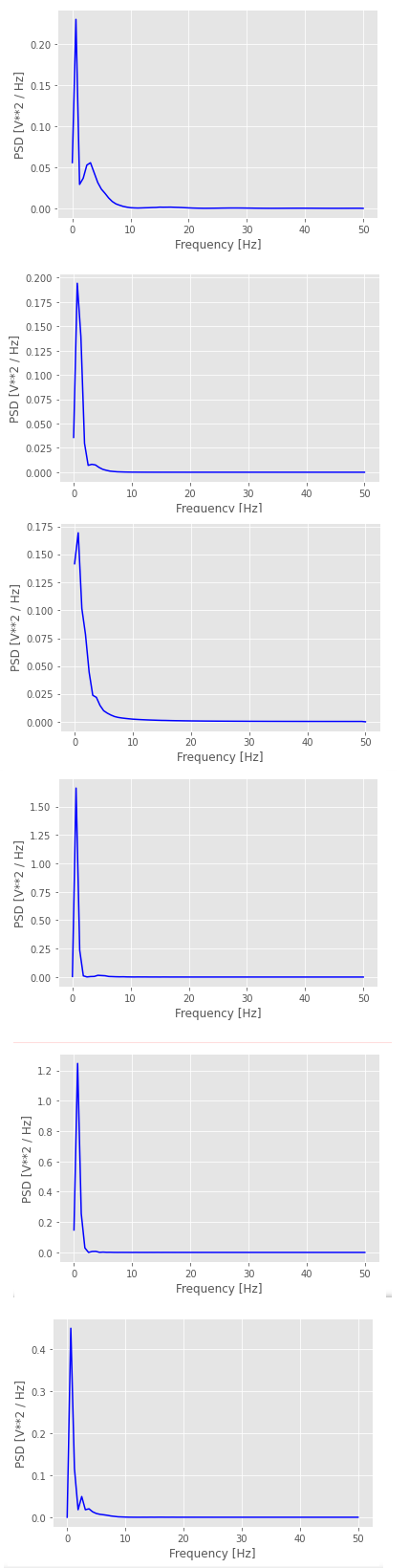
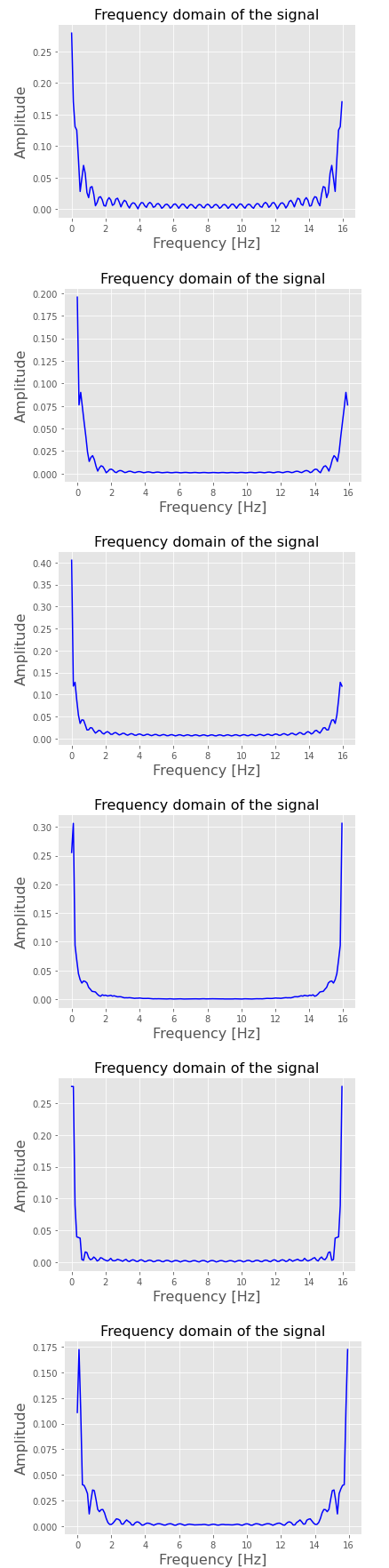


Timestamp is transformed here from hour, minute, sec, microsec format to total microseconds.

**5.2.4. FFT and PSD**

Over the window that just got sectioned out, FFT and PSD are done over these data in order to observe differences between datas to get started on identifying the datas. In figure 20, left group of graphs is the FFT and the right is the PSD.

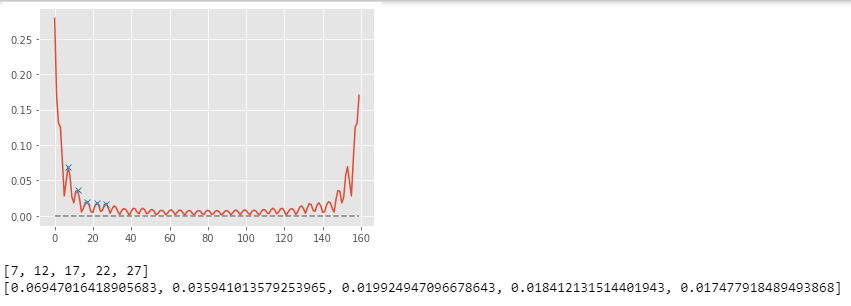
**Figure 21:** FFT & PSD



**5.2.5. Peak Detection**

The ML will try to find the first five peaks over each of the FFT and PSD that has been done and get x and y values for each of the peaks.

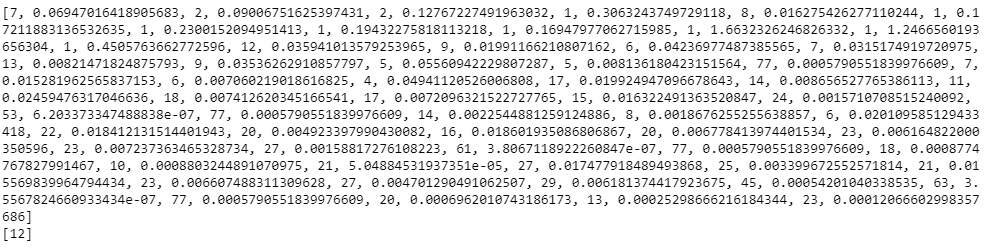
**Figure 22:** Peak Detection on FFT over X-axis Gyroscope of Region 12



**5.2.6. Making Feature and Label**

The ML combines all the first five peaks’ X and Y values of both FFT and PSD for all axes on both gyroscope and accelerometer together and labels them as the region of impact just like the image shown below.

**Figure 23:** Feature and Label



The same process is done for all 74 sets of data that was acquired and combines them as features that contain all X and Y values and Labels that contain the impact regions.

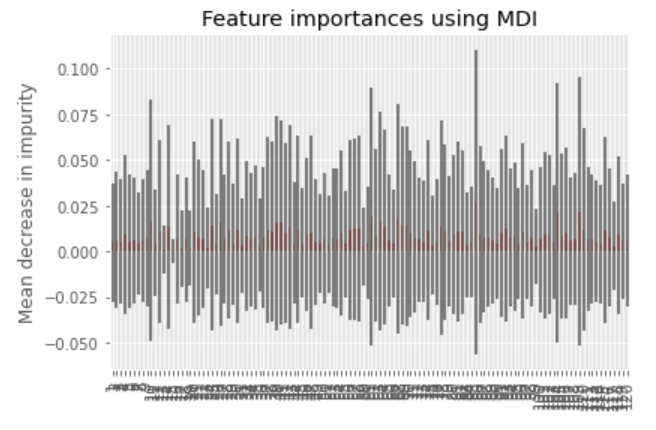
**5.2.7. Training and Testing Sets**

The data are divided into 60% for training and 40% for testing through trial and error.

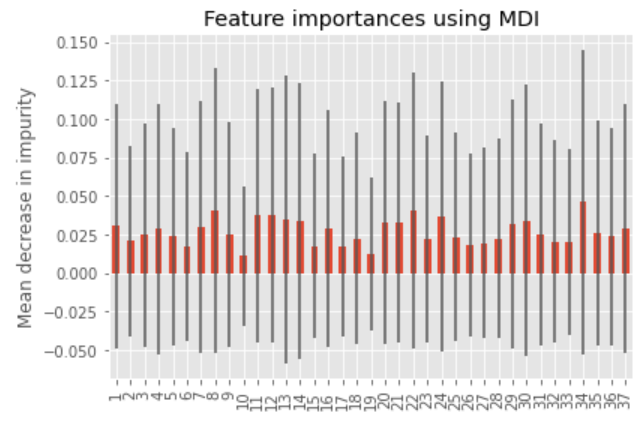
**5.2.8. Model and Feature Importance**

The data will be trained over Random Forest Classifier. Then the ML will use feature importance to find out all the important features to increase accuracy.

**Figure 24:** Feature Importance 1



The first time, a very low limit was set to take out the least important features. The training was done again and this time it is more clear to see the more important features.

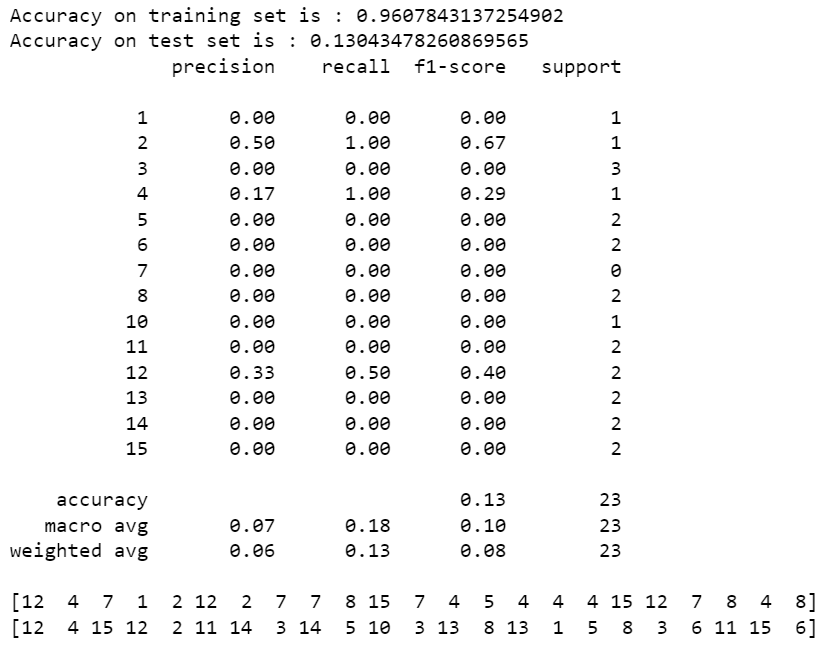
**Figure 25:** Feature Importance 2

This process is done again so we have the most important features going over the Random Forest Classifier.

**5.3. *Machine Learning Validation***

We have done more than 6 models and over 20 combinations like different portions of Train Test Split, or random shuffle with Random Forest, Logistic Regression, svm with kernel, rbf, linear or ploy and many more. From the results and the pattern of the accuracy increasing and decreasing, we found out we need more data to get further accuracy. As a result, this result with Random Forest Classifier with 60% training and 40% testing with random state of 42 is the best situation we can get with our amount of data.

**Figure 26:** Training and Testing Accuracy



**5.4. *Subsystem Conclusion***

The Machine Learning subsystem is working as expected and reached the goal of this semester. The ML is able to receive the data and do the process needed like normalizing, windowing, fft, psd, peak detection, feature extraction, training, and testing. In order to have better accuracy, we need to do much more data gathering in the next semester. As a result, the ML subsystem has the ability to transform the necessary data to give feedback for users to see their impact region on the bat.