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		Final Revision

Table of Contents

List of Tables

List of Figures

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*

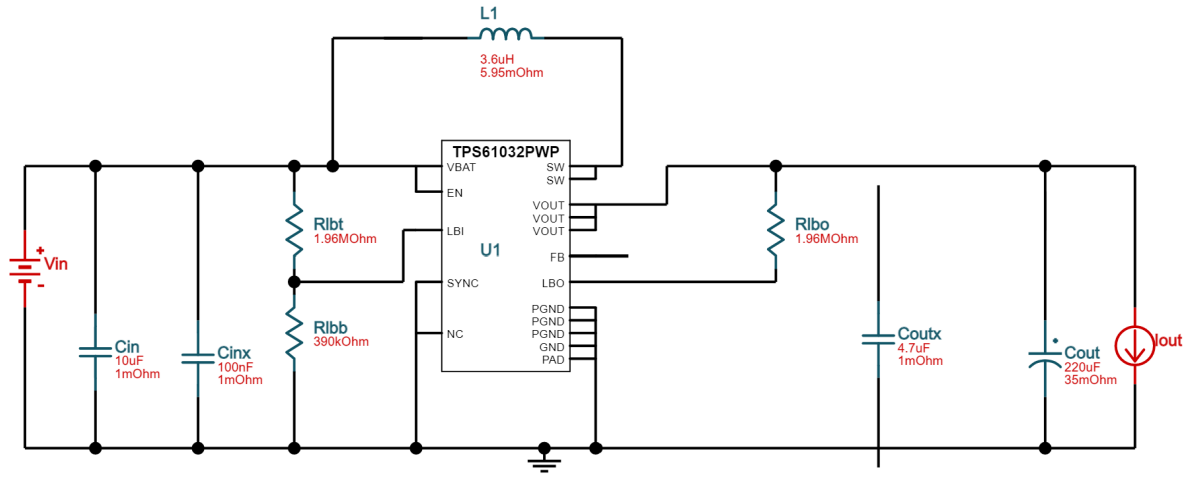
The power subsystem of the smart cricket bat consists of a lithium-ion polymer battery (LiPo), a DC-DC boost converter circuit, and a battery charger. A LiPo battery is rechargeable and can be charged with a battery charger circuit. A boost converter is used to raise the voltage from the LiPo battery in order to supply the power to the microcontroller, IMU, and bluetooth module which are part of the control subsystem. The boost converter and the battery charging circuit were 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 500 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, IMU, and bluetooth module 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. The MCP73831 chip was used for the battery charging circuit that employs a constant-current/constant-voltage charge algorithm for the battery. Figure 1 shows the schematic of a 5V DC-DC boost converter circuit.

Figure 1: Schematic of Boost Converter Circuit



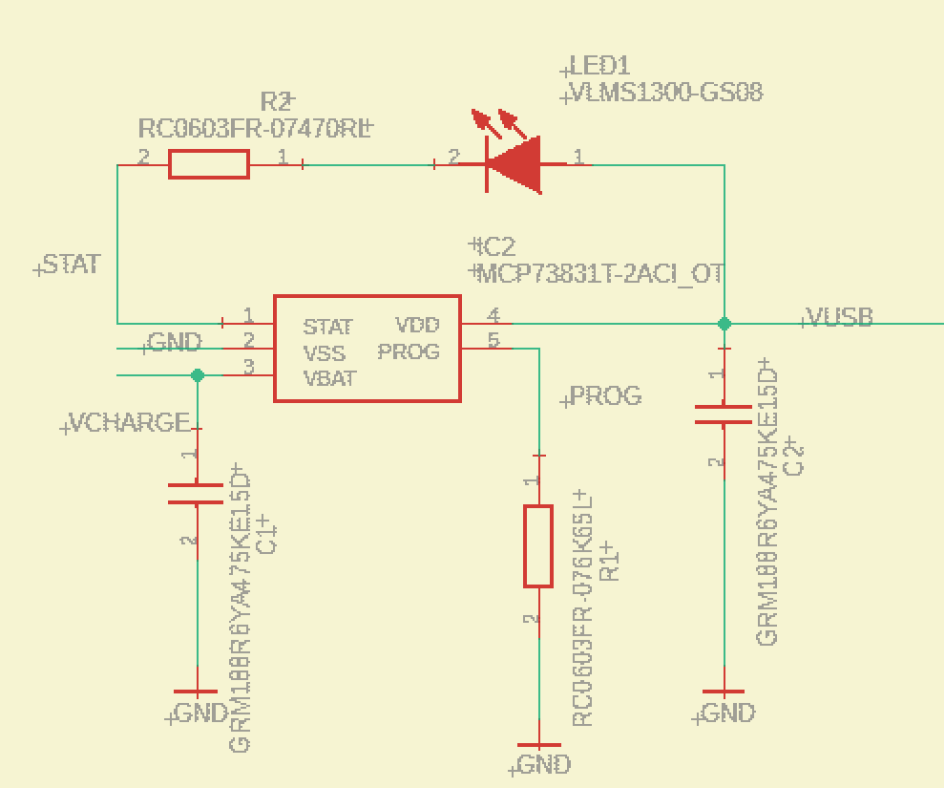
A boost converter circuit was first built on a breadboard by connecting all the components according to the schematic and the output voltage was tested to confirm that it performed according to the design. The circuit was then moved from breadboard to a perfboard for a more robust and stable design,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

For the battery charging circuit, the MCP73831 charge controller IC was used which employs a constant-current/constant-voltage charging method. The constant voltage regulation is at 4.2 V. The circuit was designed to provide a maximum charging current of 150 mA. So, a program resistor R_{PROG} = 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:

$$R_{PROG}(k\Omega) = 1000\,V / I_{REG}(mA)$$

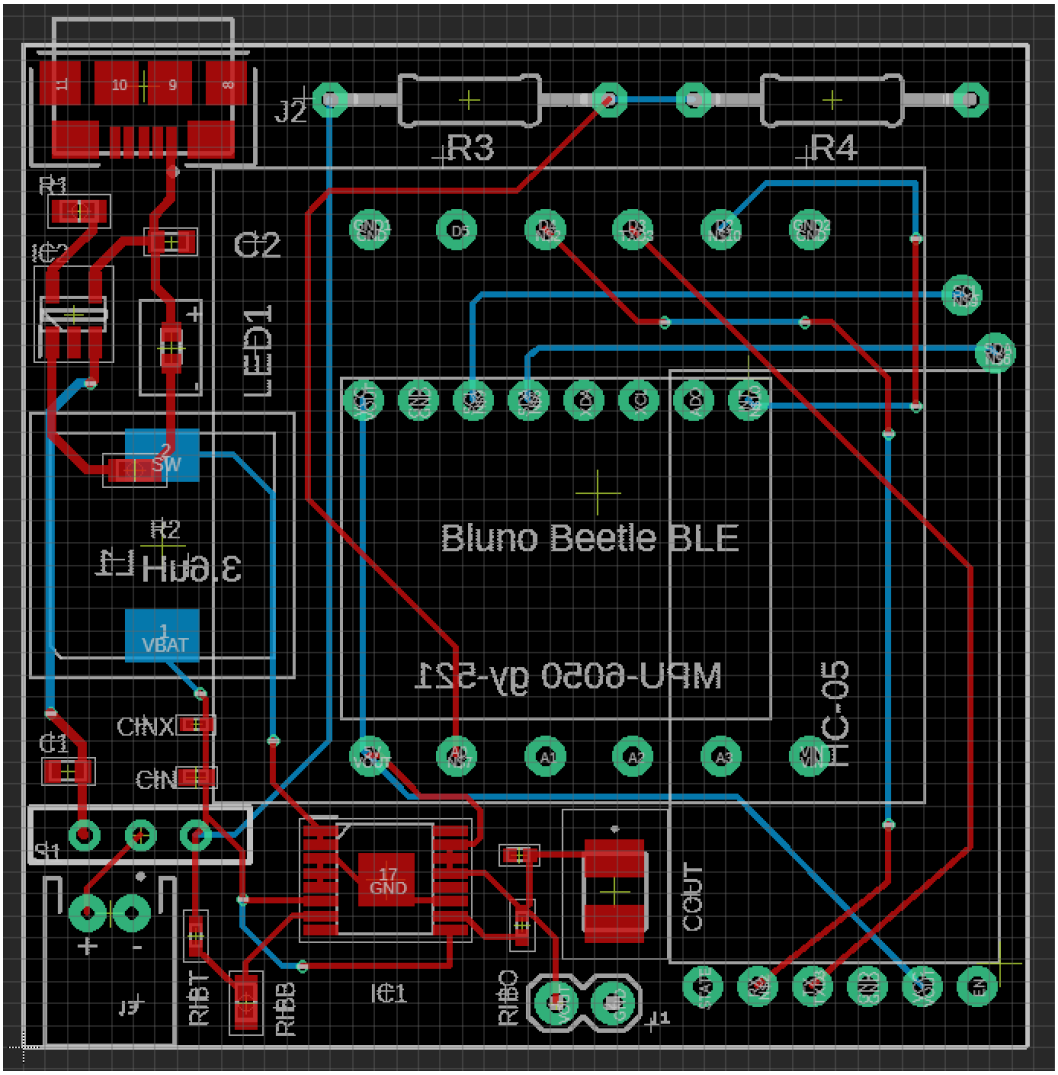
So in order to limit the charging current of the battery to 150 mA, a program resistor should be 6.67 kΩ. The red LED on the board turns on when the battery is charging and will turn off when the battery is fully charged. A schematic for the charging circuit is shown below in Figure 2.

Figure 2: Schematic of Battery Charging Circuit



A PCB was designed to provide connections between the power and control subsystems. The entire smart cricket bat device is on a single PCB. For the power subsystem, the boost converter and battery charging circuits were designed using surface mount components. There is a microUSB port on a PCB which can be used to recharge the LiPo battery. A PCB layout was designed using EAGLE and is shown in Figure 3.

Figure 3: PCB Layout



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

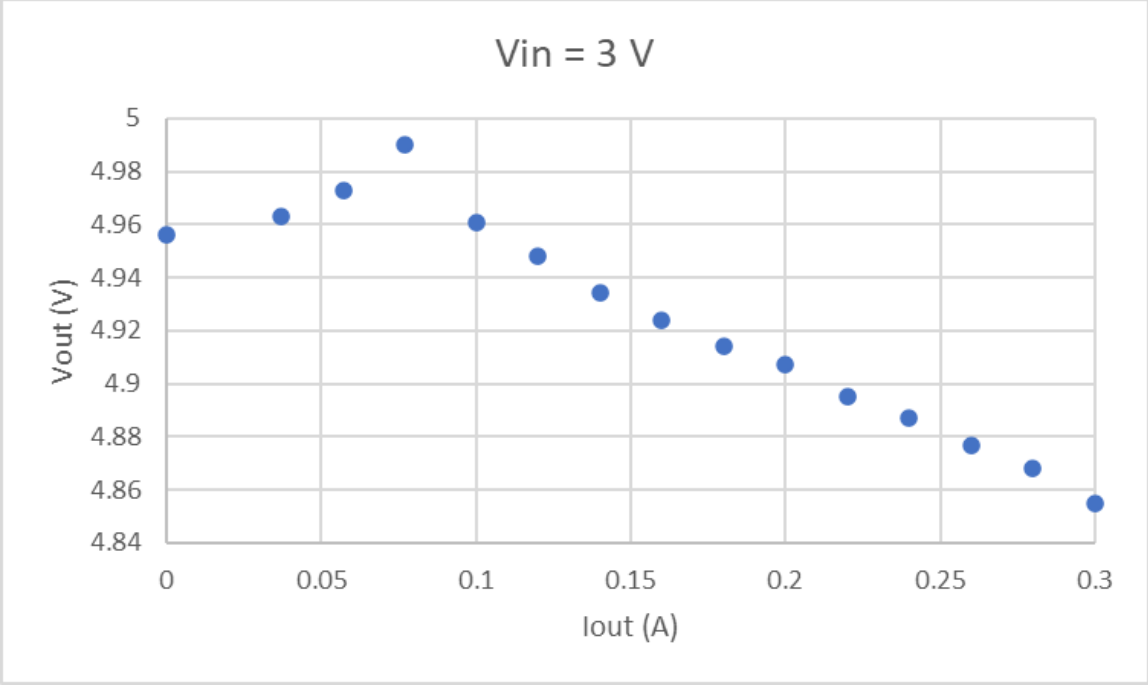


Table 2: Load Regulation with Input Voltage Vin =3.7V

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 5: Load Regulation of Boost Converter Circuit with Vin =3.7V

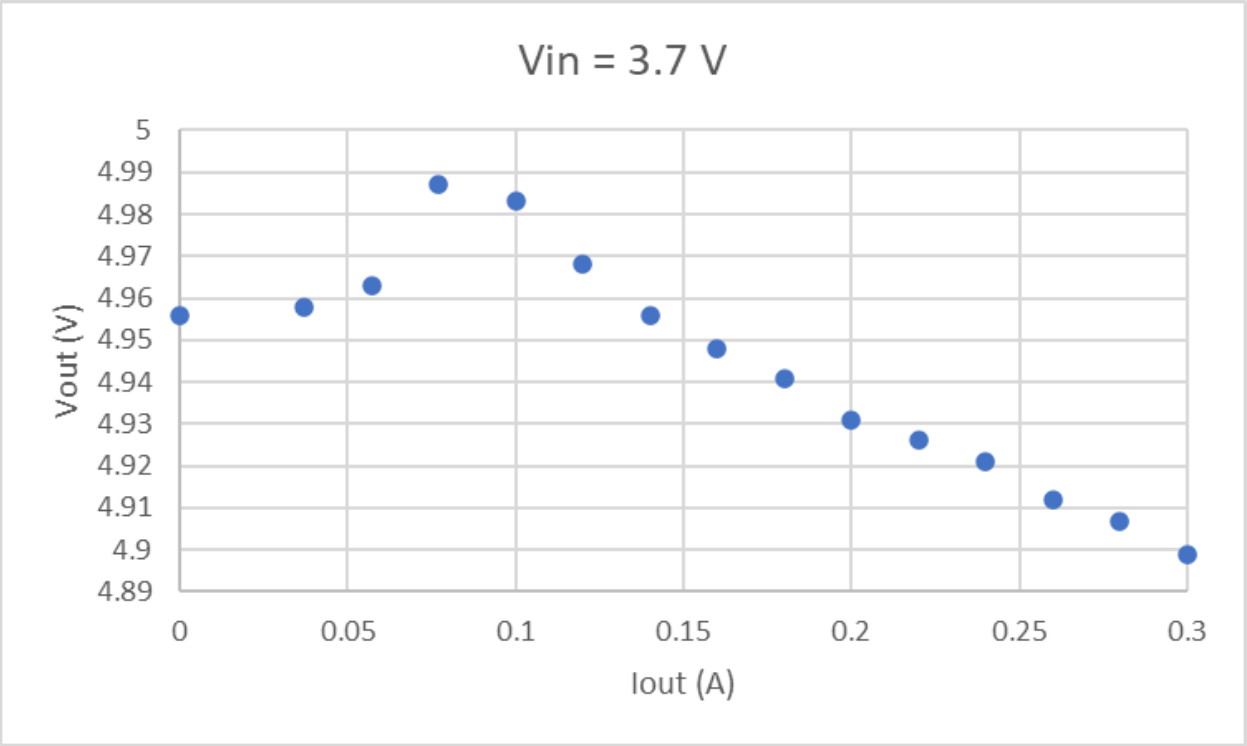
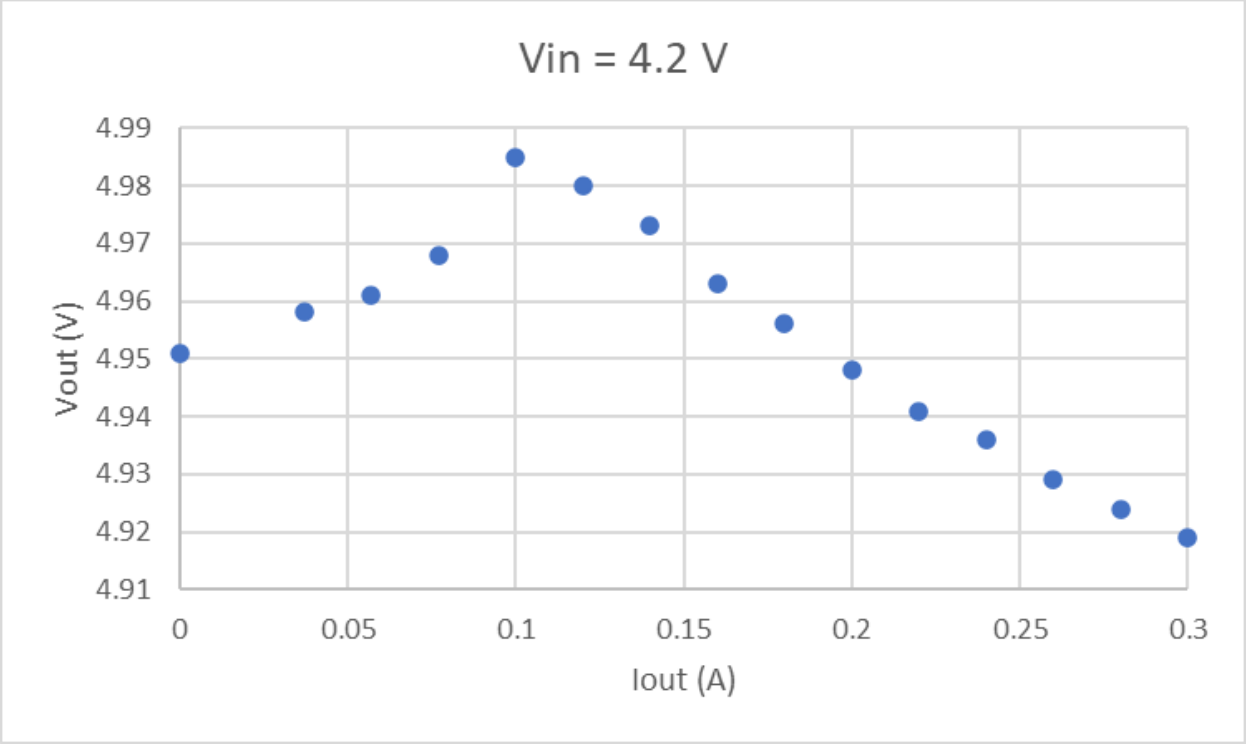


Table 3: Load Regulation with Input Voltage Vin =4.2V

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 6: 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:

$$Load\ regulation = \frac{V_{(no-load)} - V_{(full-load)}}{V_{(full-load)}} \times 100$$

The line regulation is calculated using the following formula:

$$Line\ regulation = \frac{\Delta V_{out}}{\Delta V_{in}} \times 100$$

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

$$Load\ regulation = \frac{V_{(no-load)} - V_{(full-load)}}{V_{(full-load)}} \times 100 = \frac{4.956-4.899}{4.899} \times 100 = 1.1635\ \%$$

$$Line\ regulation = \frac{\Delta V_{out}}{\Delta V_{in}} \times 100 = \frac{4.963 - 4.961}{4.2 - 3} \times 100 = 0.167\%$$

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 entire system containing MCU, IMU and bluetooth module 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.

The battery charging circuit was tested by measuring the battery voltage and charge current going into the battery. The circuit is designed to limit the maximum charge current at 150 mAh and and the voltage at 4.2 V. When the voltage is supplied to the charging circuit, the charge current is at about 150 mAh as designed. When a certain voltage is reached, the voltage is regulated until current goes to zero. When this happens, the battery is fully charged and the red LED turned off. Table 4 shows the battery voltage and charge current. Figure 6 shows the plot corresponding to the data shown in Table 4.

Table 4: Battery Voltage and Charge Current

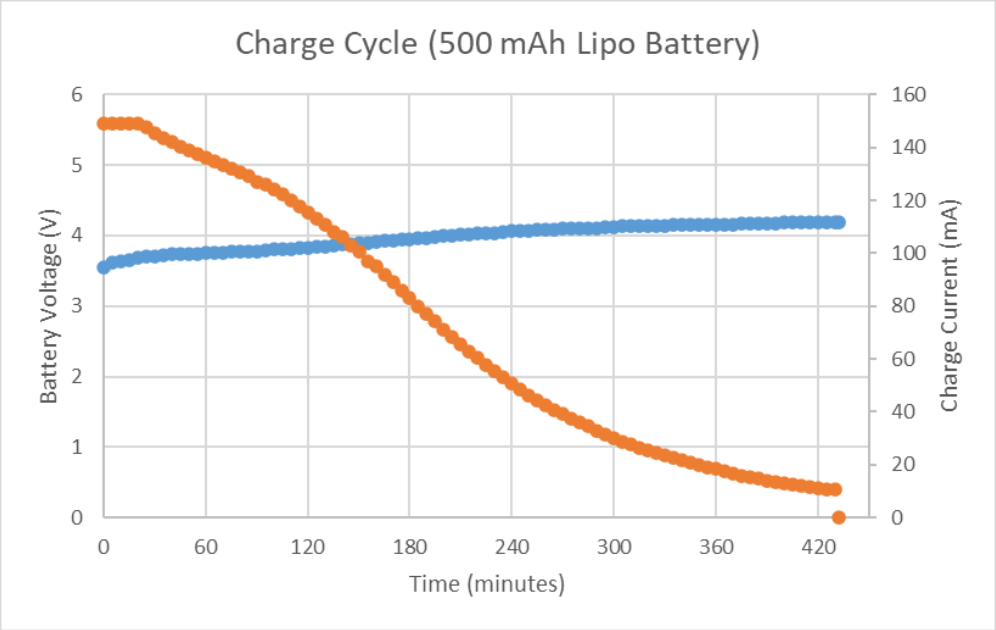
t (minute)	v (V)	I (mA)
0	3.55	149.1
5	3.61	149.1
10	3.64	149.1
15	3.66	149.1
20	3.68	149.1
25	3.7	147.6
30	3.71	145.3
35	3.72	143.5
40	3.73	141.9
45	3.73	140.3
50	3.74	138.8
55	3.74	137.5
60	3.75	136
65	3.75	134.6
70	3.76	133.3
75	3.77	132
80	3.77	130.5
85	3.78	129
90	3.78	127.1
95	3.79	125.9
100	3.8	124.2
105	3.8	122.3
110	3.81	120
115	3.82	117.6
120	3.83	115.4
125	3.84	113.1
130	3.85	110.7
135	3.86	108.1
140	3.87	106
145	3.88	103.1
150	3.89	100.5
155	3.9	97
160	3.91	95.1
165	3.92	92
170	3.93	88.9
175	3.94	85.9
180	3.95	83
185	3.96	80
190	3.97	77
195	3.98	74.1
200	3.99	71.2
205	4	68.4
210	4.01	65.7
215	4.02	63
220	4.03	60.4
225	4.03	57.9
230	4.04	55.4
235	4.05	53
240	4.06	50.7
245	4.06	48.5
250	4.07	46.4
255	4.08	44.4
260	4.08	42.6
265	4.09	40.8
270	4.1	39.1
275	4.1	37.6
280	4.1	36
285	4.11	34.5
290	4.11	33
295	4.12	31.5
300	4.12	30.2
305	4.13	28.9
310	4.13	27.7
315	4.13	26.6
320	4.14	25.5
325	4.14	24.4
330	4.14	23.5

335	4.15	22.6
340	4.15	21.7
345	4.15	20.8
350	4.16	19.9
355	4.16	19.2
360	4.16	18.4
365	4.16	17.4
370	4.16	16.8

375	4.17	16
380	4.17	15.4
385	4.17	14.7
390	4.17	14.1
395	4.17	13.6
400	4.18	13
405	4.18	12.6
410	4.18	12

415	4.18	11.6
420	4.18	11.2
425	4.18	10.8
430	4.18	10.5
432	4.18	0

Figure 7: Complete Charge Cycle (500 mAh Lipo Battery)



On the plot above, the blue color shows the battery voltage and the orange color shows charge current going into the LiPo battery. As can be seen from the table and the plot above, the LiPo battery can be recharged with the charging circuit. The charging circuit is confirmed to work according to the design.

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 about 5V at the output of the converter to power the entire device. A

battery is able to last for approximately 6 hours when the device is on and can be recharged with a microUSB via the charging circuit.

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, bluetooth module, and the control subsystem as a whole, operates correctly. To accomplish this, the microcontroller was programmed and wired to establish a connection to both the IMU and Bluetooth module. Once the two were interfacing with the microcontroller, each of the two axes of the IMU, the 3-axis gyroscope and 3-axis accelerometer, and the Bluetooth module were tested to validate that it performed as the manufacturer described.

3.2. *Microcontroller and Bluetooth*

3.2.1. Overview

The hardware of the control system supports interfacing between the IMU sensor and the commercial user application via a HC-05 Bluetooth module. 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 used as the control system to deliver data from the IMU sensor to the app.The Bluno Beetle has a sufficient number of pins and supply voltages to meet the requirements of the Smart Cricket Bat. The Bluno Beetle does have a Low-energy Bluetooth 4.0 module as well, however for the sake of ease and time, an HC-05 Bluetooth 3.0 Module was used instead.

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 and HC-05. This is more than enough to interface with all required modules, the single IMU sensor needs power, a pair of SDA and SCL pins, and a single digital pin for the interrupt. And the HC-05 module only requires power and a pair of RX and TX pins (however due to the design and placement of the RX and TX pins on the Bluno, 2 digital pins were used as replacement utilizing the SoftwareSerial library in Arduino IDE). 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 phone application that connects to the cloud which holds the machine learning algorithm, it must be able to maintain this connection for the duration of a training session. Utilizing the HC-05 Bluetooth module, communication between the MCU and the user app

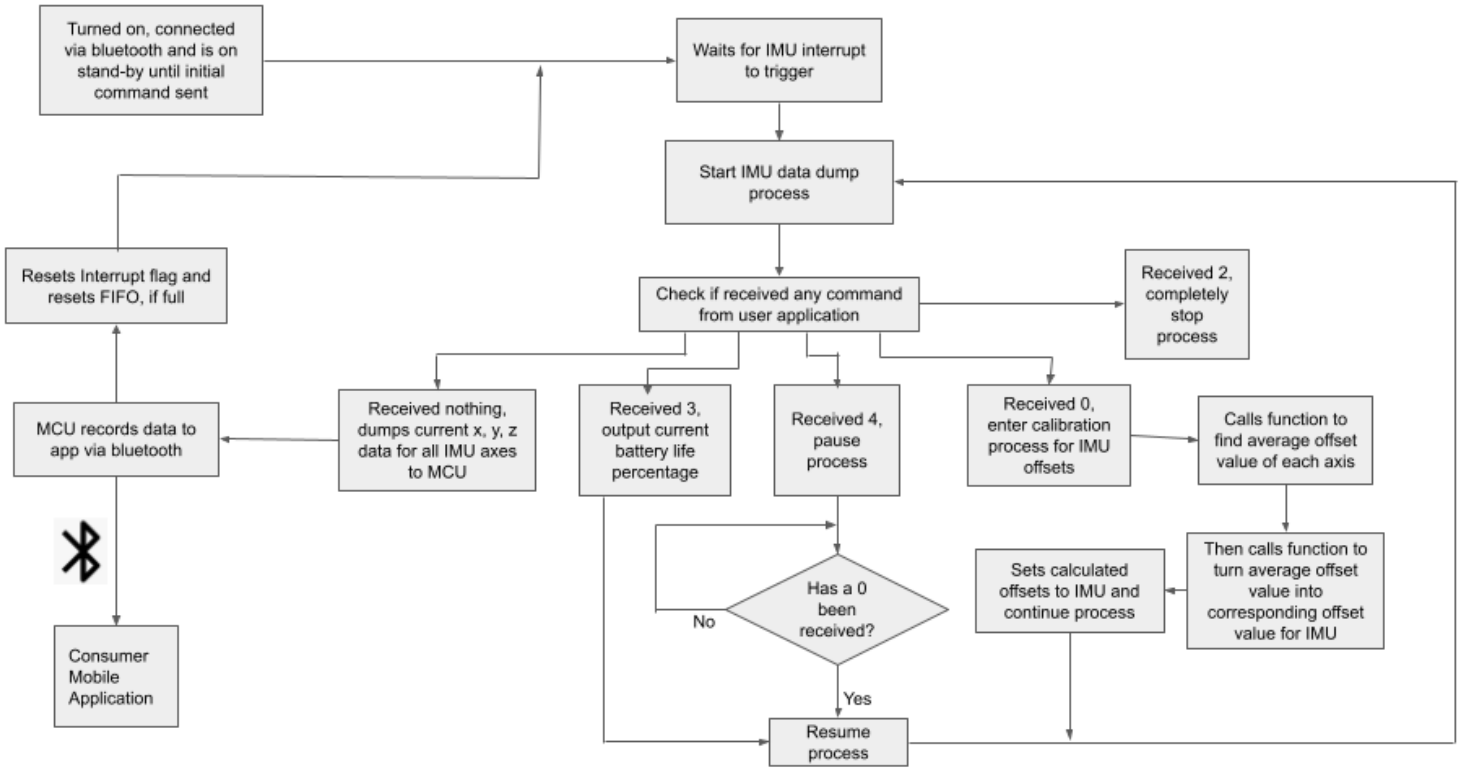
is stable and is able to reach a maximum distance of 240 feet (or approximately 74 meters). A PCB connects all control parts into a secure, robust design which is able to withstand the force of hits up to 200 gs. This is an important aspect of the Bluetooth design, as any physical connection issues on the board can affect the data transfer of any of the 3 main components, MCU, IMU, or the Bluetooth module. Specifically any connection issues on the RX/TX pins will cause discrepancies in the data sent from IMU to the application, leading to issues when trying to be processed by the machine learning algorithm. The PCB itself has dimensions of 45.7mm by 45.7mm, with only 2 layers, top and bottom.

3.2.3. MCU 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 a laptop to confirm the HC-05 bluetooth module was operating as intended. Then to confirm the code written for the MCU to receive and send data works as intended with standard bluetooth, a dummy sketch was written to send dummy values when requested by phone via bluetooth.

The flowchart in figure 8 shows the logical operation of how the microcontroller goes about gathering the IMU data, interpreting user input, and delivering the data to the consumer application.

Figure 8: 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.

3.2.4. Bluetooth Validation

The HC-05 Bluetooth module's connection distance was validated by writing a dummy sketch that simply connects the device to a laptop, and when prompted by a user input sends the current distance between the device and the laptop. This is done in intervals of 10 ft and was corroborated by walking with a measuring tape to insure exact distance that the bluetooth loses connection which was at approximately 220 ft to 240 ft, this test was done multiple times hence the range of give or take 20 ft. The received messages from the device at each 10 ft interval can be seen in figure 9, the figure also acts as the evidence to show the effect of a connection issue on the RX and TX pins of the Bluetooth module. One can notice the gaps in the strings caused by insufficient connections being unable to fully send the expected data.

Figure 9: Bluetooth Distance Validation

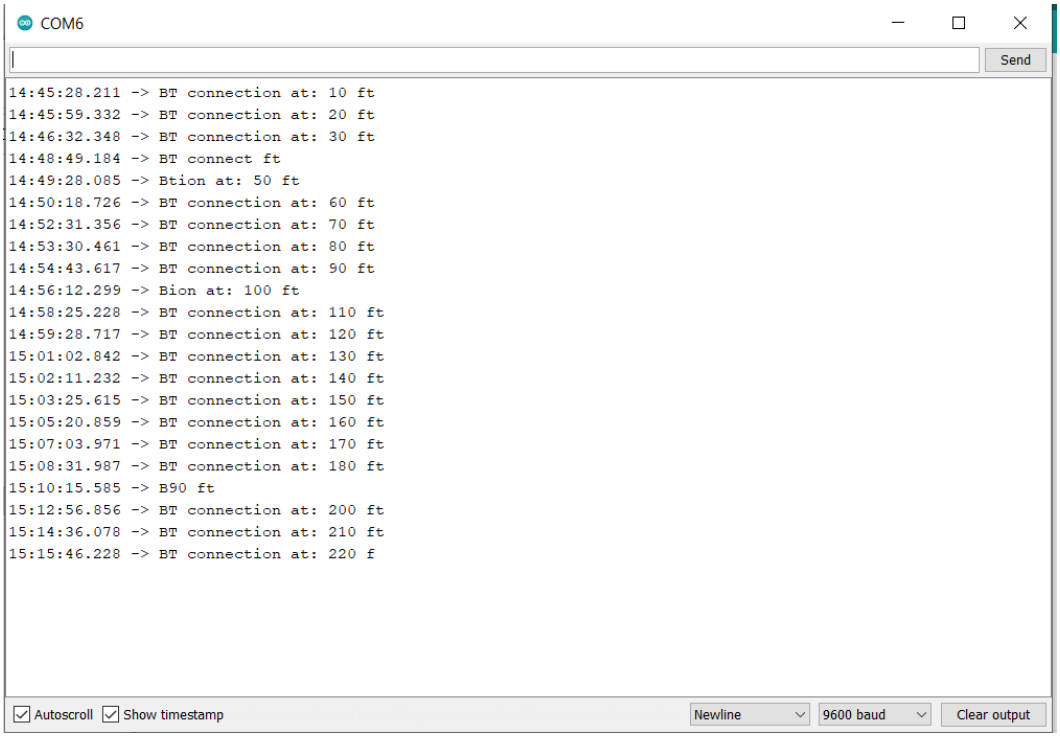


Figure 10: Bluetooth Data Transfer Validation

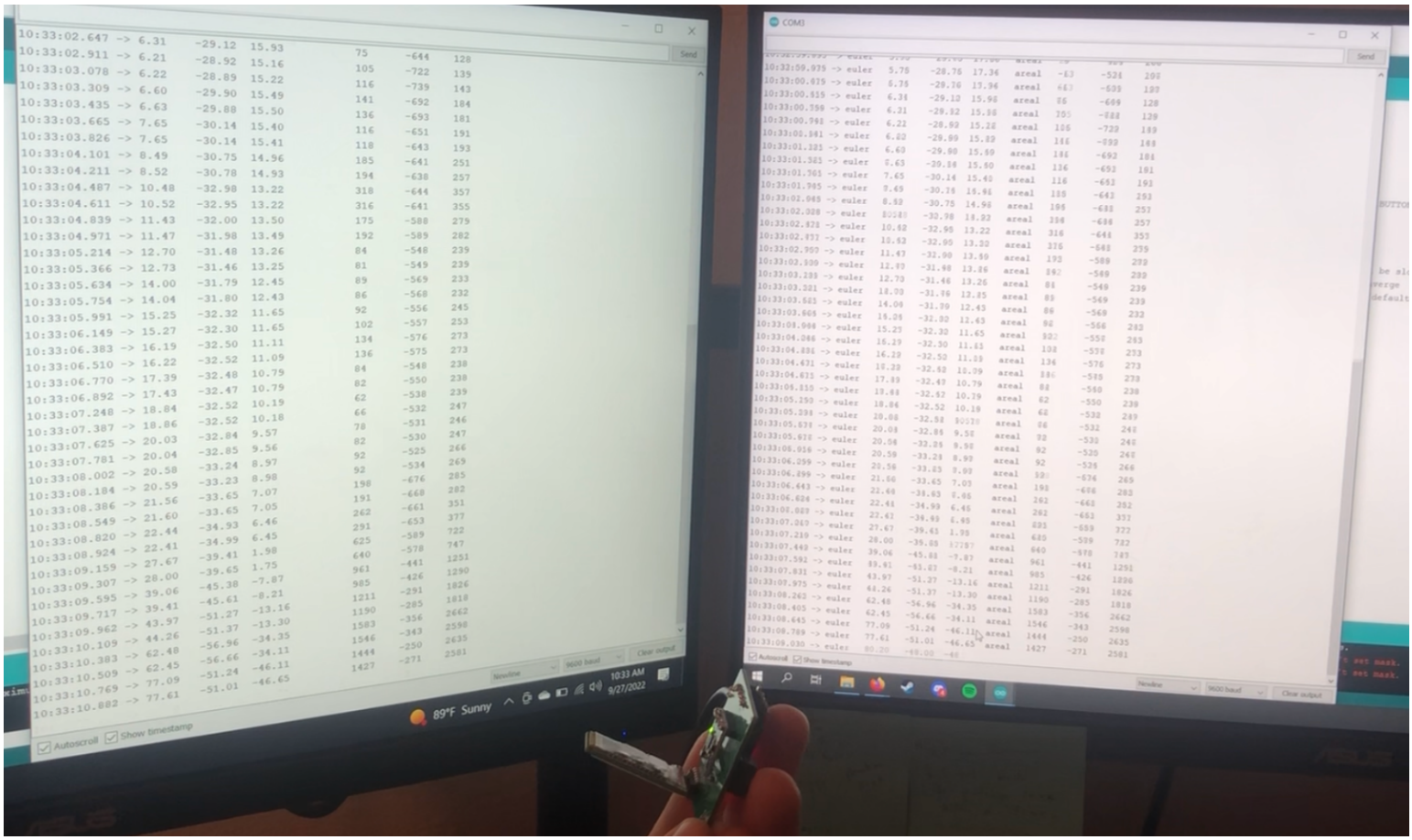


Figure 10 shows the data being sent via bluetooth to a laptop and the right shows the data being received through a mircoUSB via the serial interface. These two communication windows are receiving the same information with the left being at a delay of approximately 2 micro seconds. This shows the data sent via bluetooth has no discrepancies in data and is being received as expected.

3.3. *Gyroscope*

3.2.1. Operation

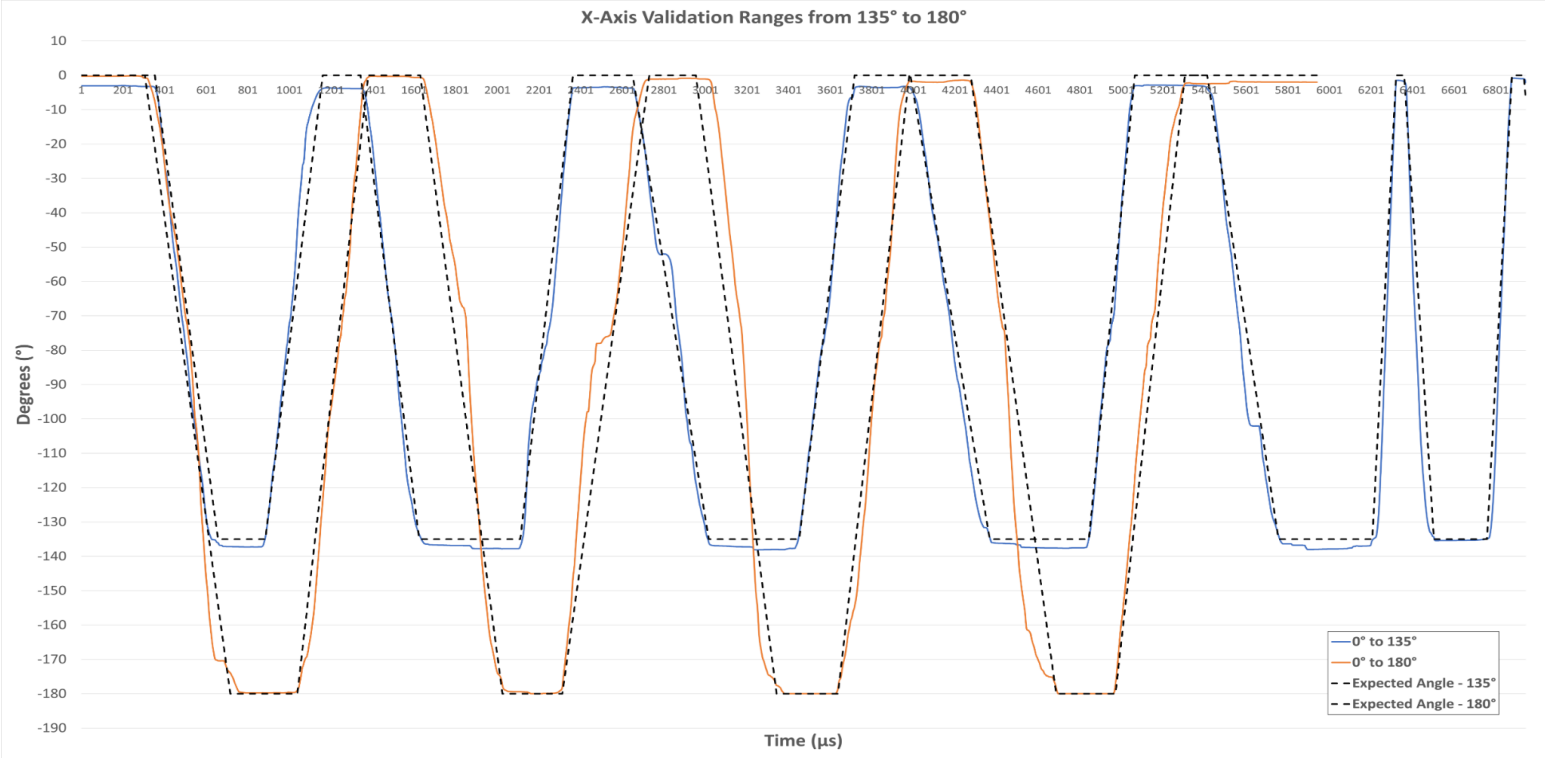
The inertial measurement unit used is a gy-521 board that uses an MPU-6050. The MPU-6050 has both a gyroscope and accelerometer inside, this was chosen for two option, one was to save on space as the design must stay small and unobtrusive, and two, its best if the gyro and accelerometer axes are lined 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.

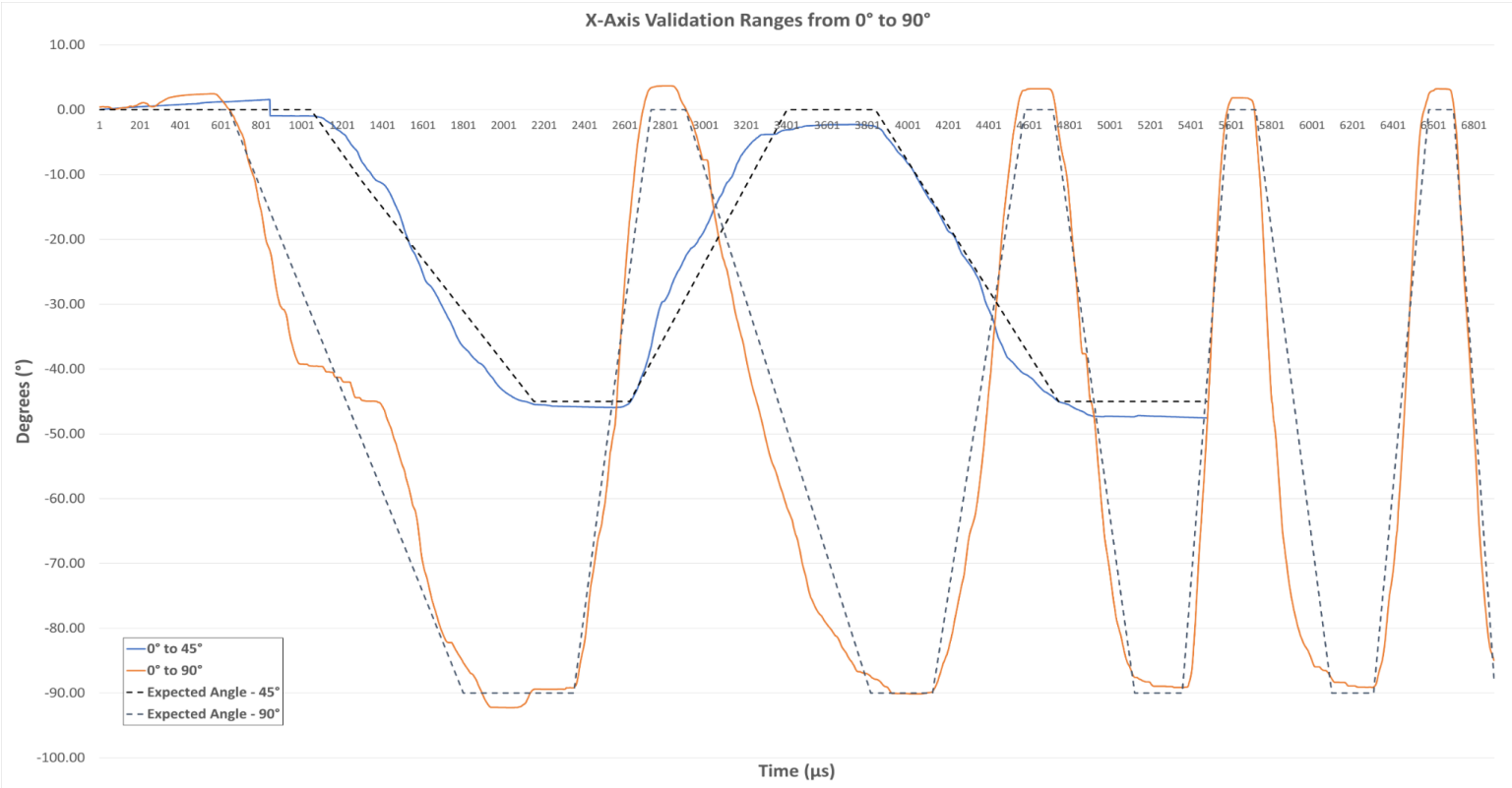
The gyroscope uses an I2C interface, which allows the microcontroller to take the measurements in via the serial clock and serial data lines. This is the standard way I2C devices communicate, by transmitting data that are 9 bits long along these 2 lines.

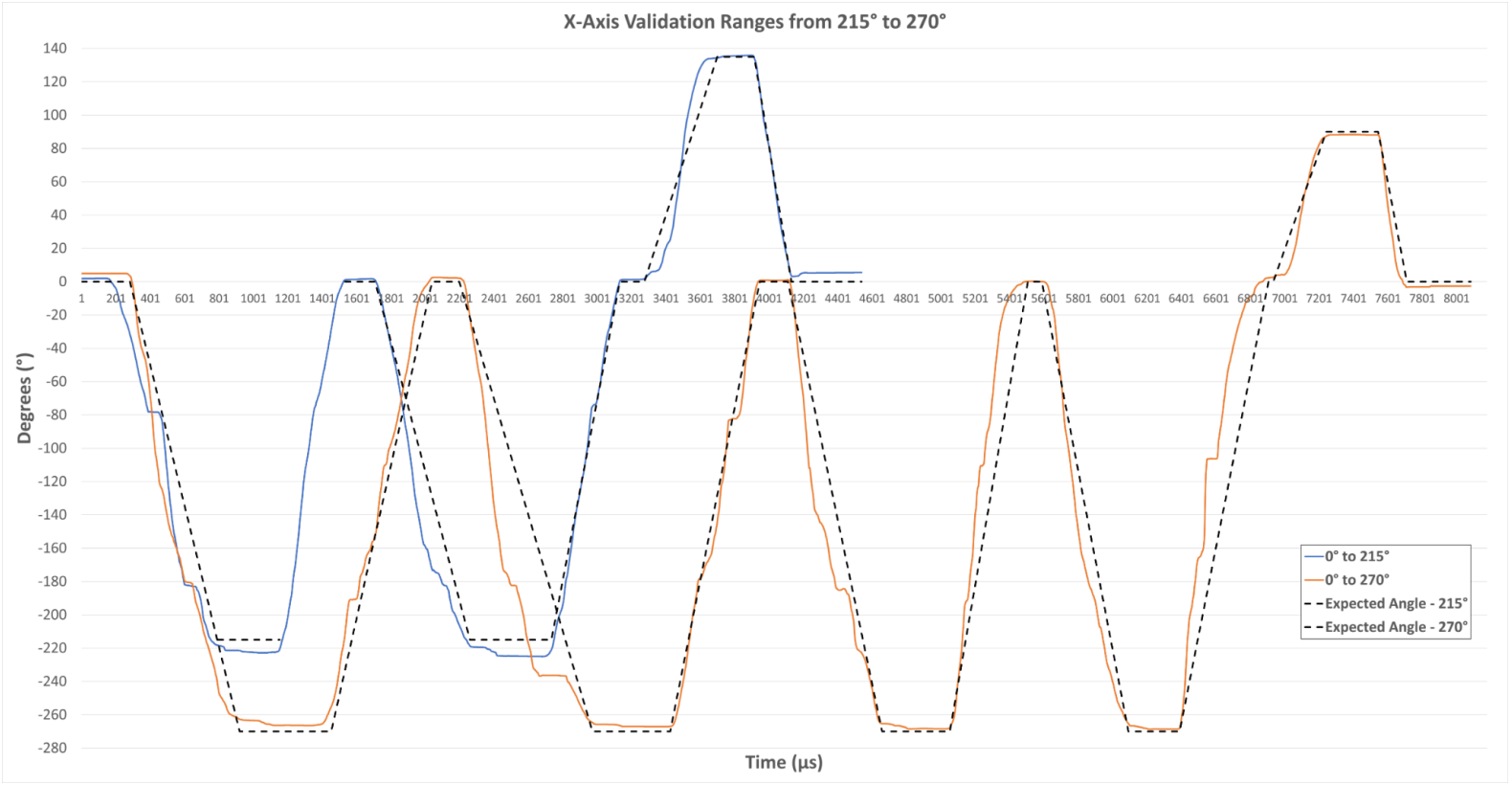
3.2.2. Validation

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, the x and z-axes from angle ranges of 0-45°, 0-90°, 0-135°, 0-180°, 0-215°, 0-270°, 0-305°, 0-360°, while the y-axis was done in the same method but only to 180° as that is the limit of that axis before the angle wraps back around. Below are all the plots of the validation, in color is the actual rotation received from the IMU and the black dashed line is the expected angle estimated from the protractor used during the process.

Figure 11: X-Axis Gyroscope Validation






X-Axis Validation Ranges from 215° to 270°
Degrees (°)
Time (µs)
0° to 215°
0° to 270°
Expected Angle - 215°
Expected Angle - 270°

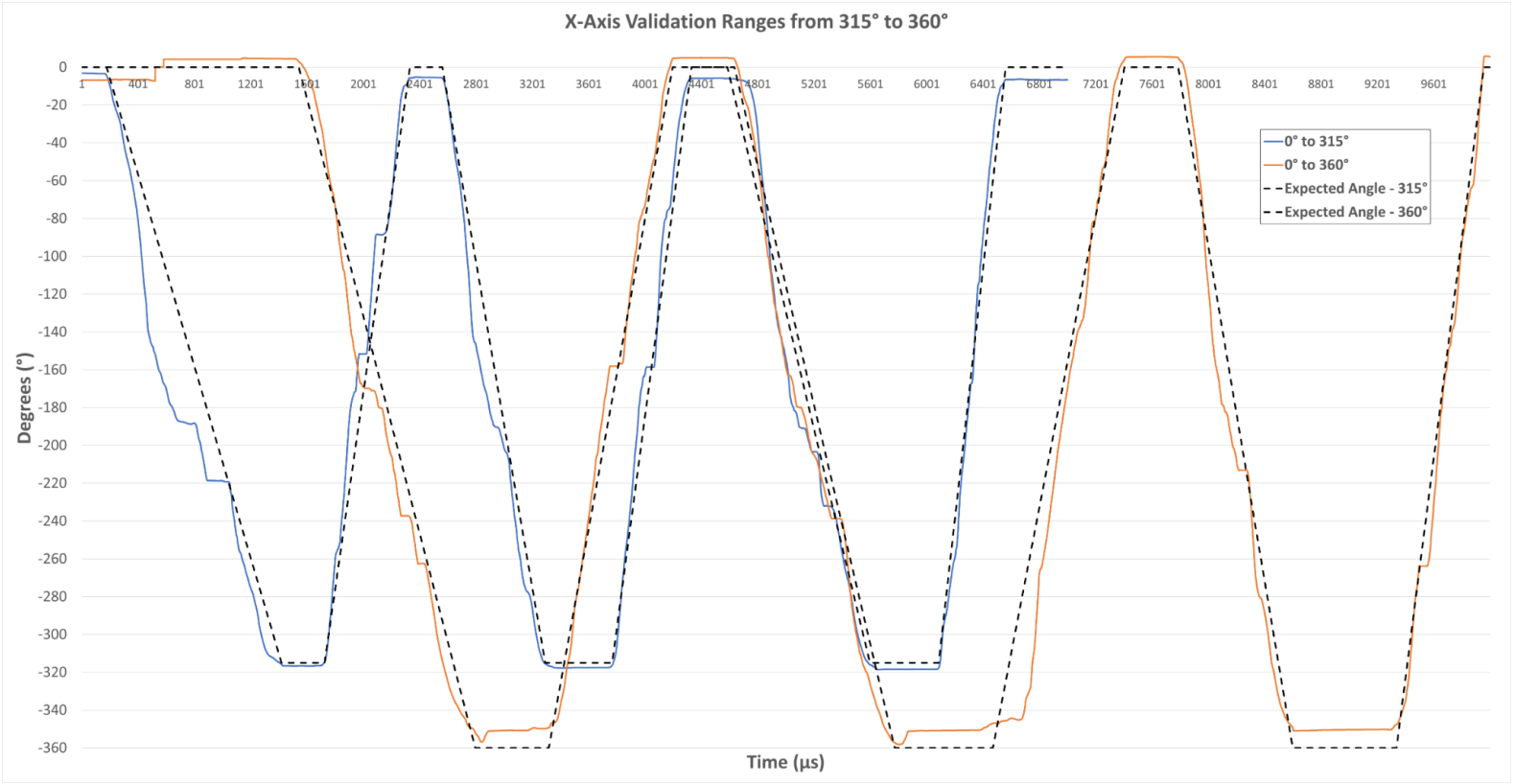
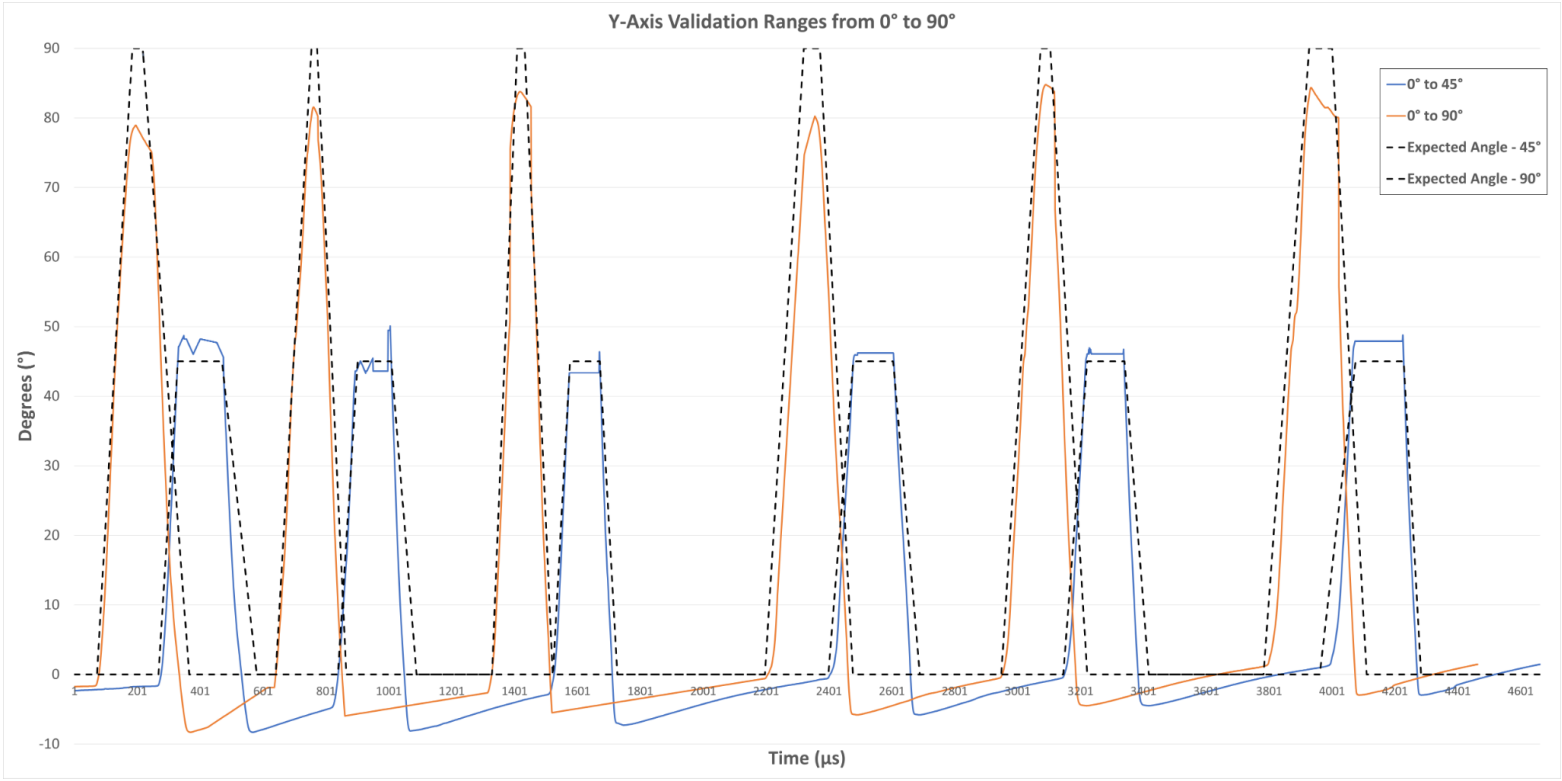

X-Axis Validation Ranges from 315° to 360°
Degrees (°)
Time (µs)
0° to 315°
0° to 360°
Expected Angle - 315°
Expected Angle - 360°

Figure 12: Y-Axis Gyroscope Validation



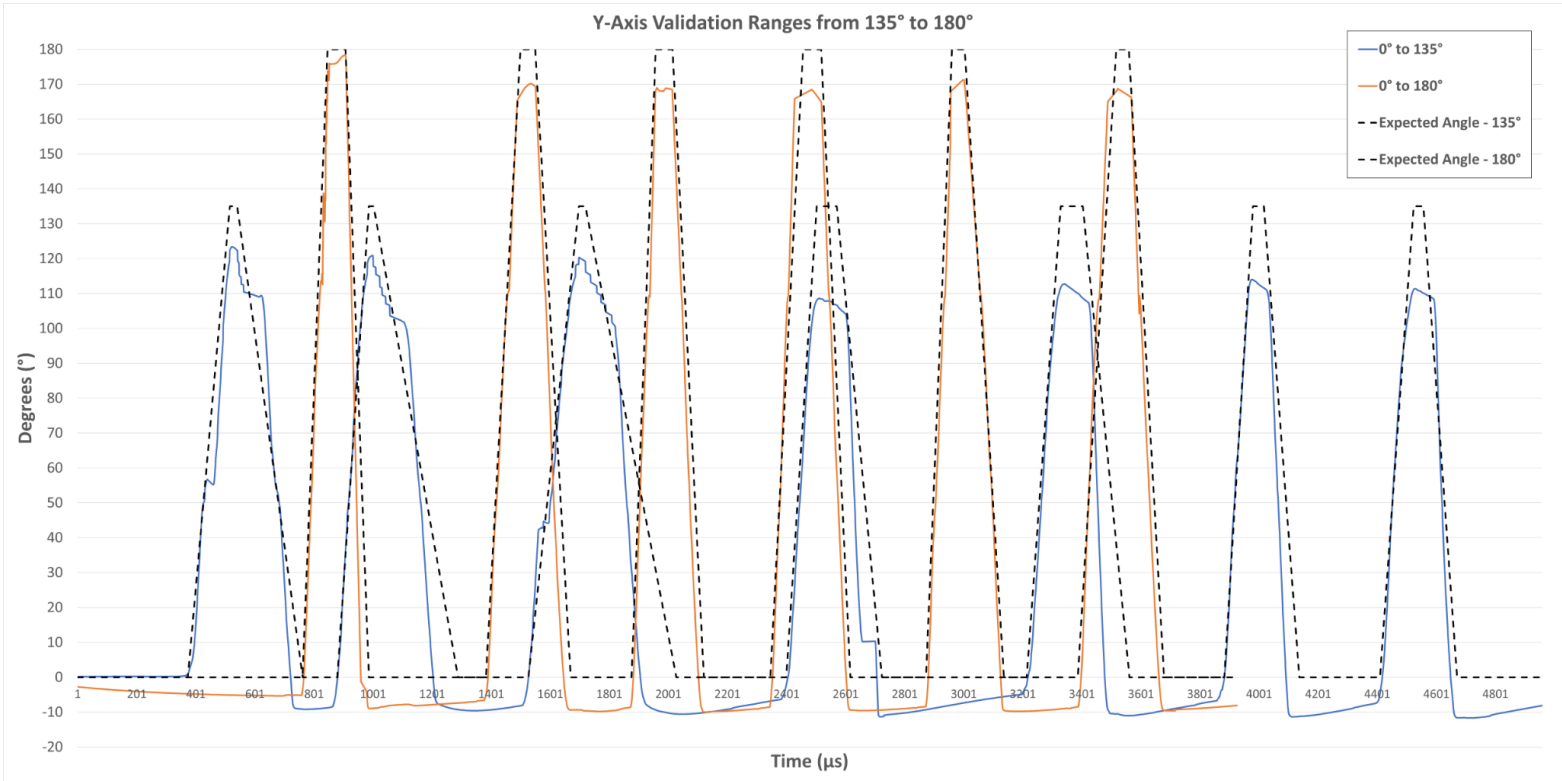
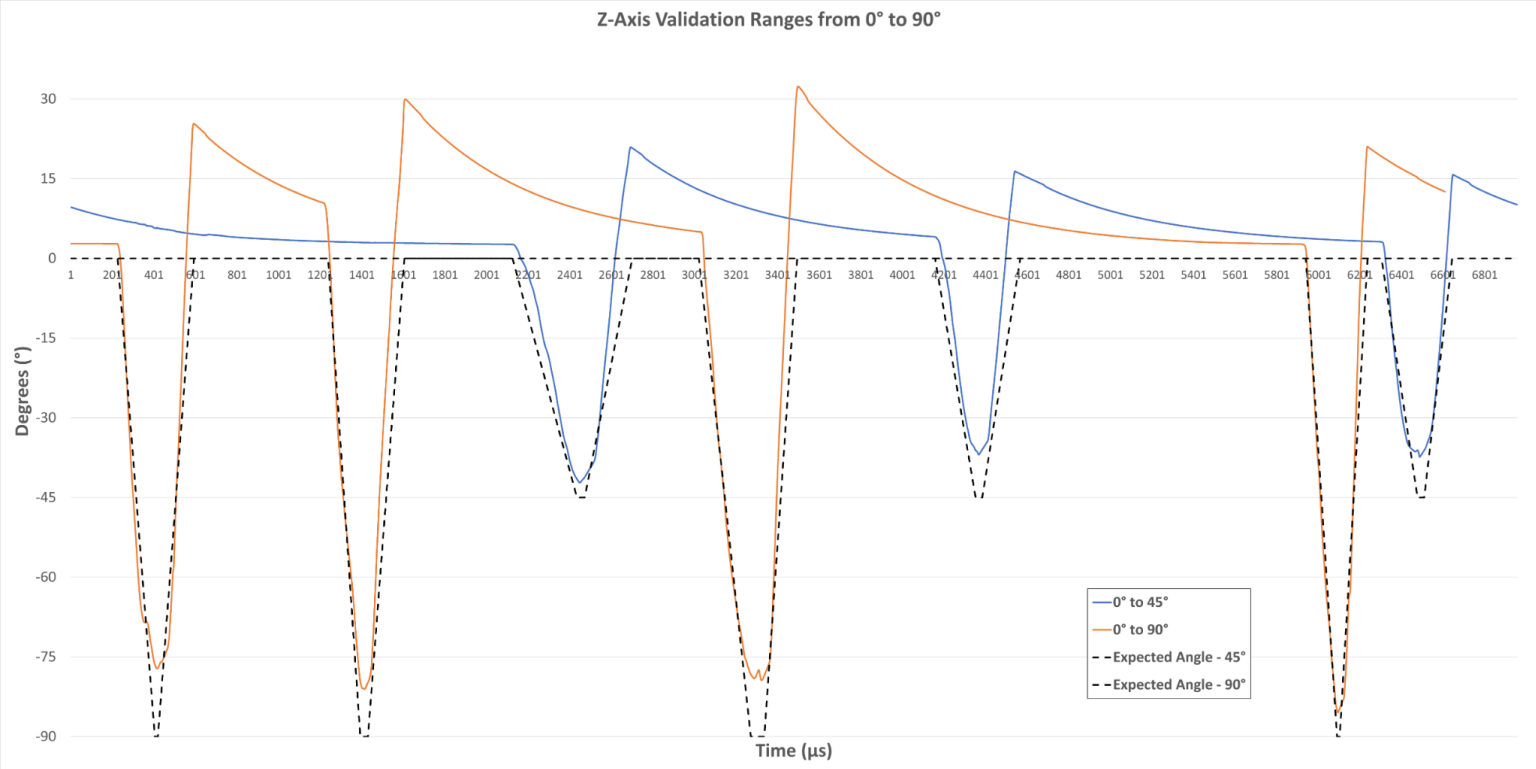
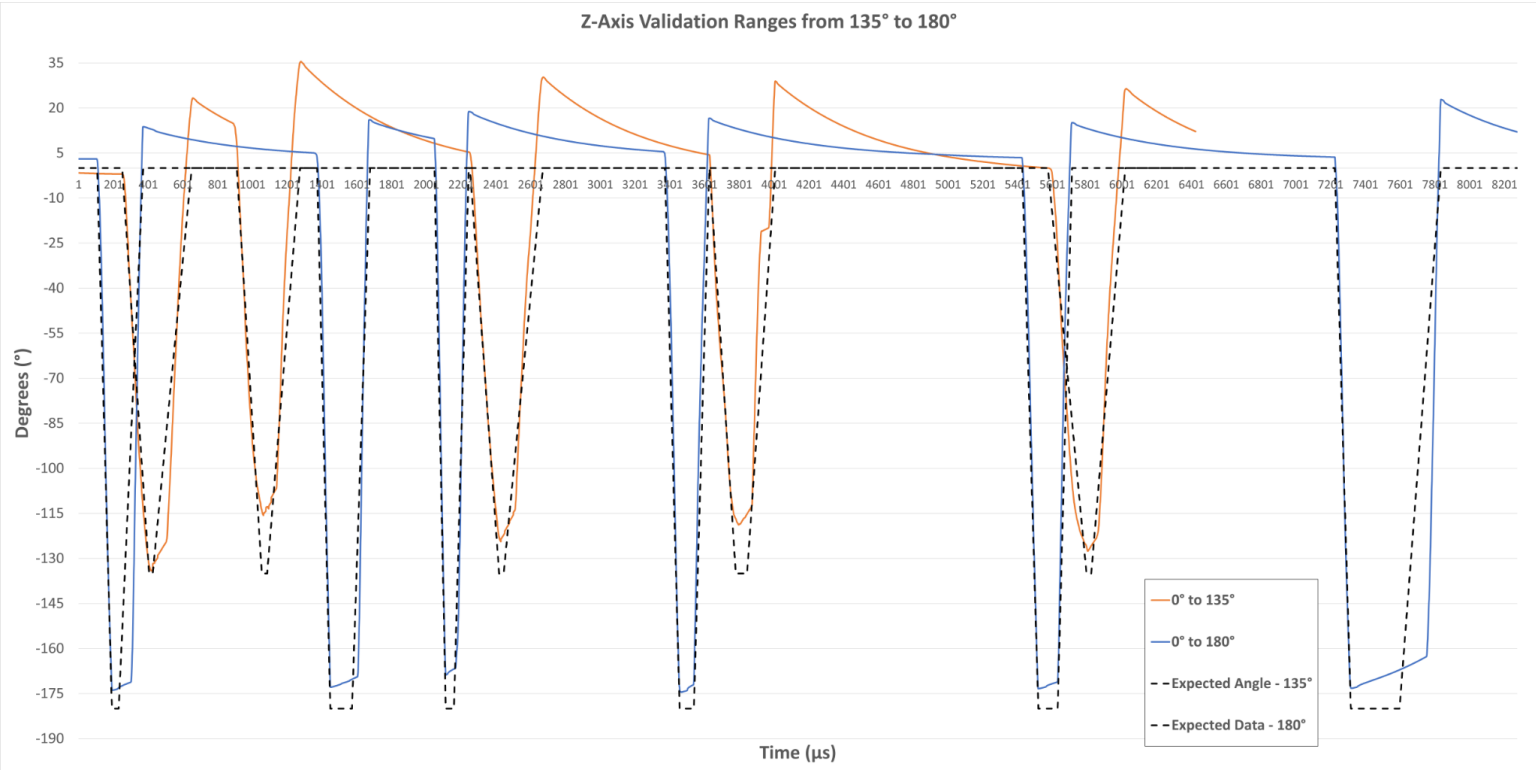
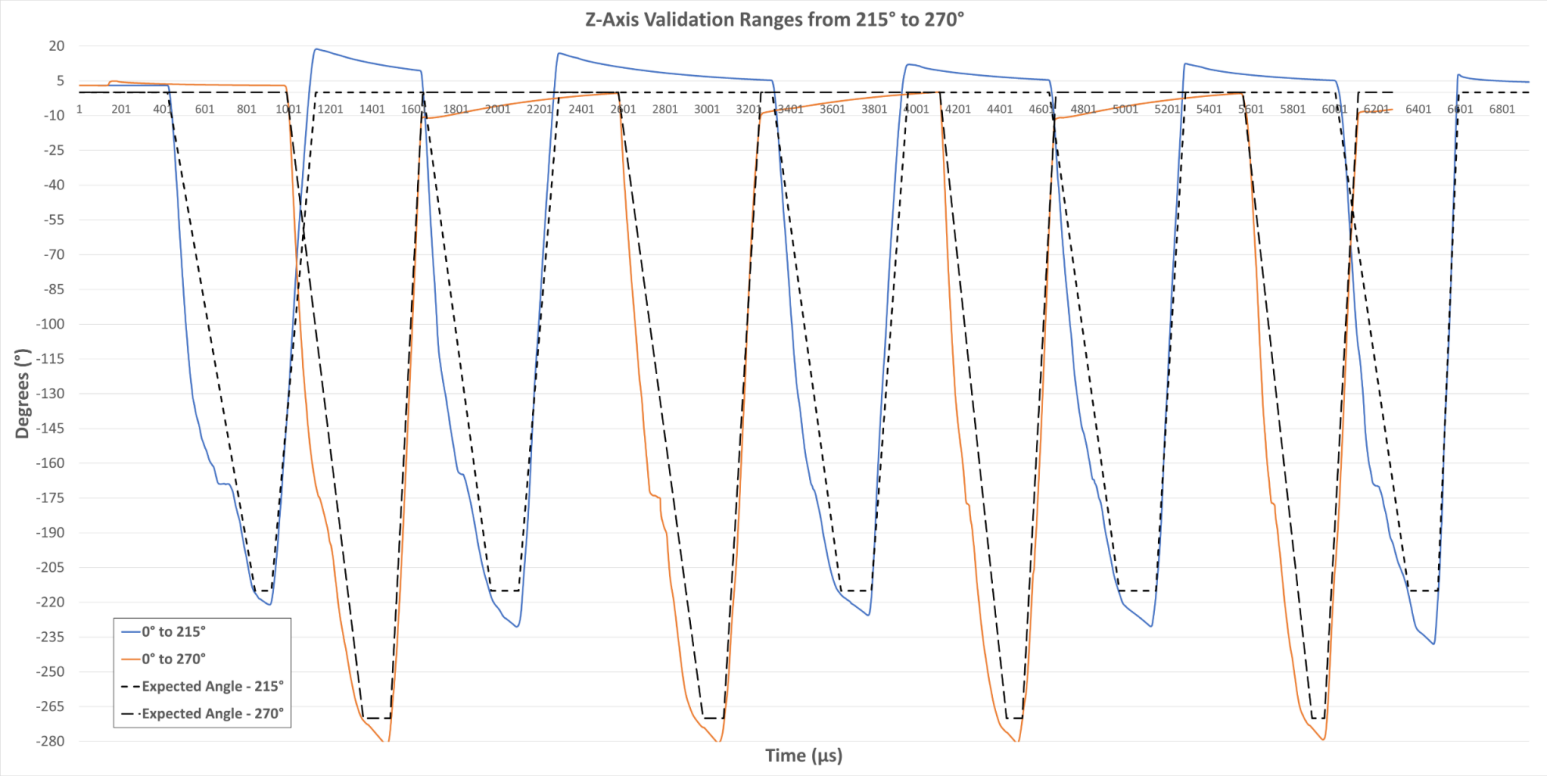
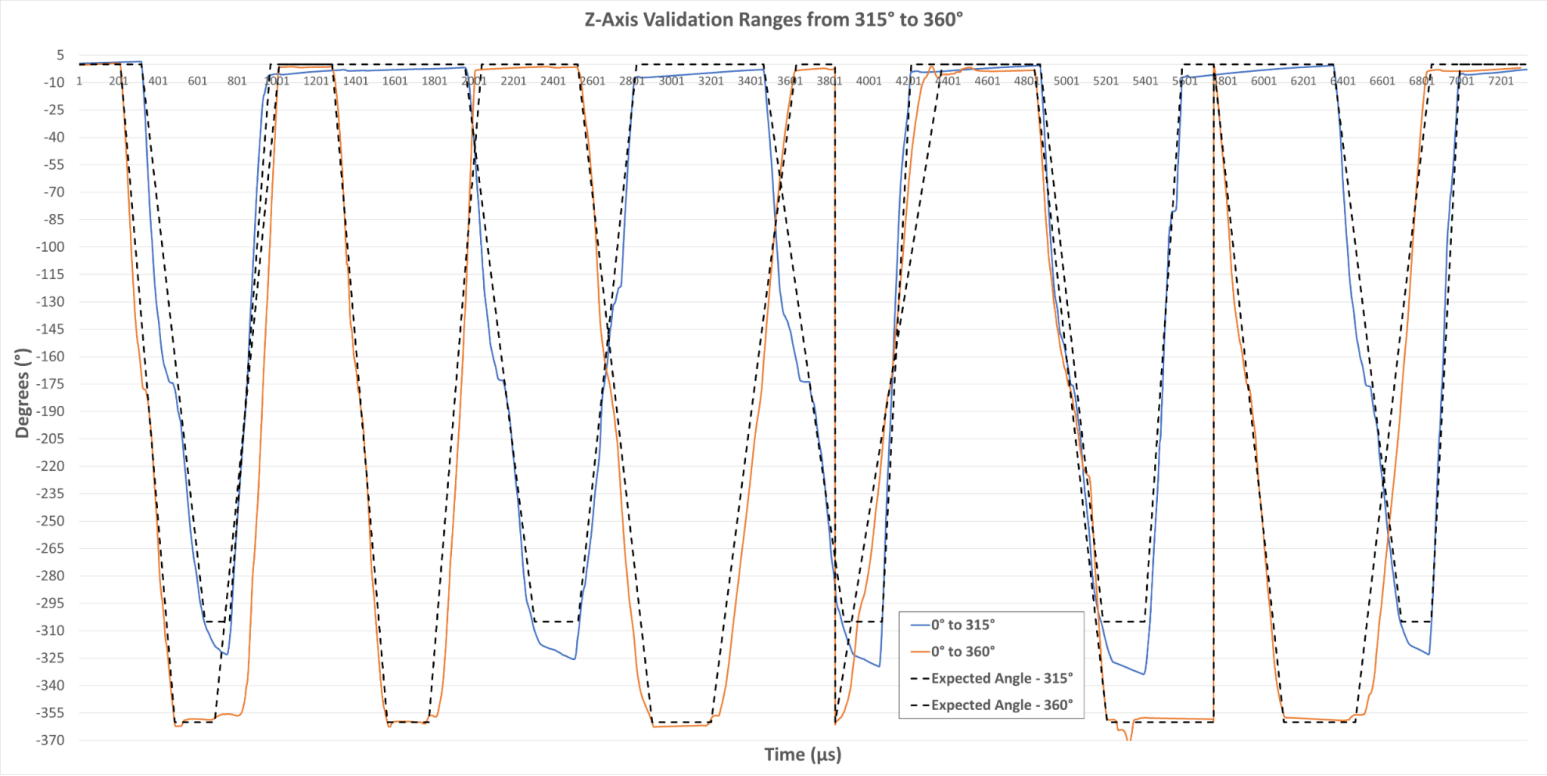


Figure 13: Z-Axis Gyroscope Validation









Notice, the graphs for the y and z axes drift a bit more when compared to expected angle, which causes them to lose track of where they started, being 5° to 10° off on return most of the time, with the outlier of an offset of nearly 35° for the z-axis during its 90° test. However,

this won't be much of an issue as most swings won't start at 0° anyway and most swings will be moving fast enough that the drift will not have time to start taking effect. And the machine learning algorithm will already be subtracting the starting degree from the finished degree to get the true angle of the bat instead of what's given. The angle is also only important for giving the simple tilt of the bat on impact and to calculate the torque seen by the bat.

Through extensive validation, the gyroscope within the IMU is confirmed to be working as intended and gives the accurate degrees of change, within a given margin, for the system.

3.4. *Accelerometer*

3.4.1. Operation

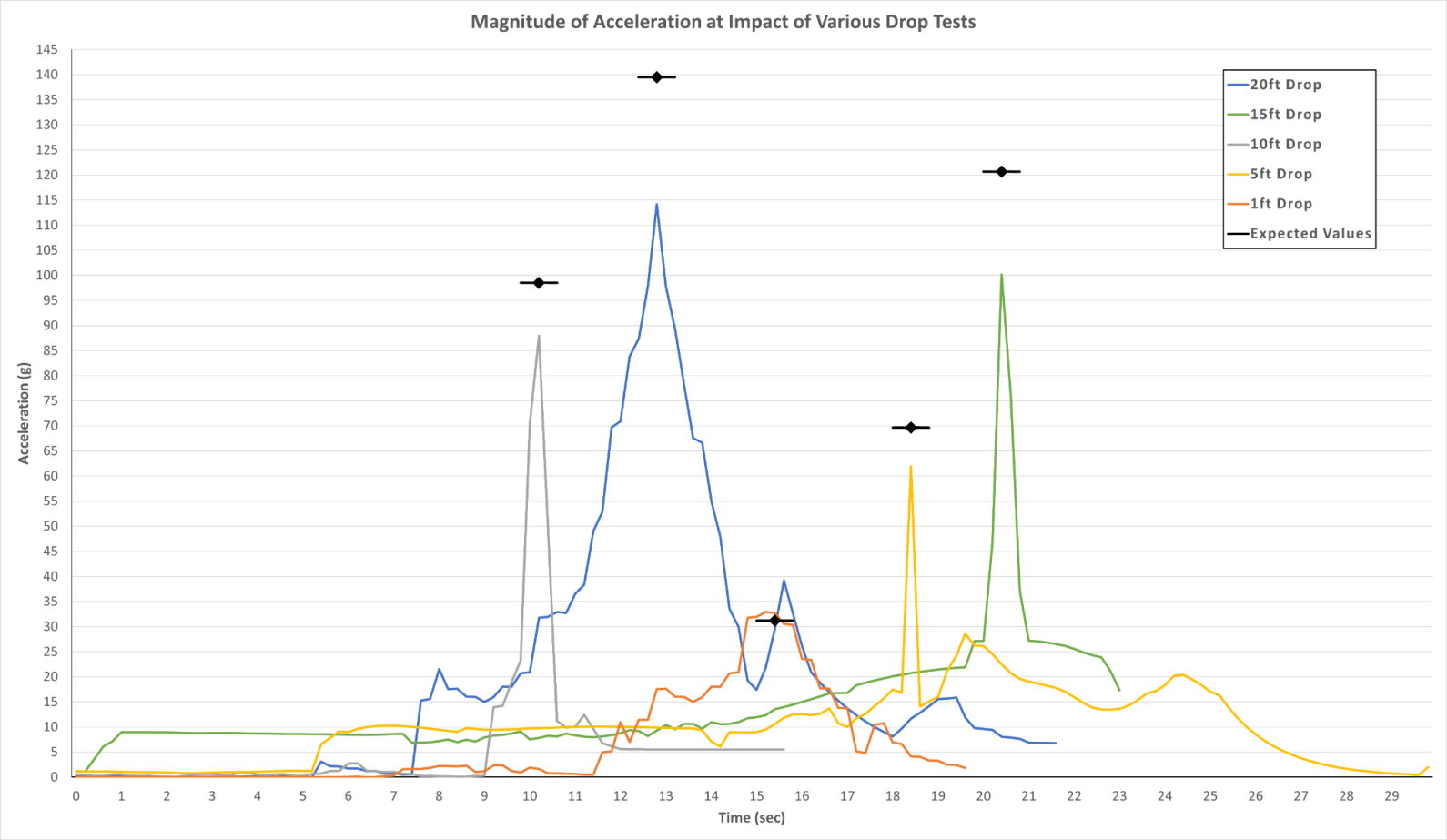
The inertial measurement unit used is a gy-521 board that uses an MPU-6050. For the accelerometer, it has 4 full scale ranges, ±2, ±4, ±8, ±16 g with the sensitivity being 16384, 8192, 4096, 2048 LSB/g, respectively. The raw data received from the accelerometer is in terms of least significant bit per 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 was decided to give the best accuracy for our machine learning algorithm after testing all individually.

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 5 different heights ranging from 1 to 20 feet, in intervals of 5ft. The device was put into a padded box and dropped from each height respectively while sending the IMU data during the fall to the laptop to record said data. With this in mind, the following graph was produced from the magnitude of the acceleration of the x, y, and z axes when falling and hitting concrete from heights at 1, 5, 10, 15, and 20 feet. The collared plots are of the actual data gathered from the IMU during its fall and the black line is the expected value of each drop found through calculations.

Figure 14: Falls from Various Heights 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. When performing the validation previously there was a problem with the acquired values being a magnitude of 10 smaller than the expected values, this problem was rectified by redoing the conversion calculations from the IMUs LSB/g unit to meters per second and finding that a step involving multiplying by gravity was missing. With the corrected data one can notice that the measured results get further and further from the expected results the impact force increases, likely due to the sudden change that causes said impact becoming too quick for our IMU to accurately measure. Evenstill, the acquired data is much closer and within an acceptable range of the calculated expected value to claim it gives accurate enough reading for our purposes.

Method Used to Calculate gs

$$Use\ Kinetic\ Energy\ to\ find\ velocity\ right\ before\ collison:$$

$$KE = \frac{1}{2}mv^2 \rightarrow v_1 = \sqrt{2gh} = \sqrt{2 * 9.81 * h}$$

$$Use\ velocity\ to\ find\ acceleration\ during\ collision,\ assume\ t = 10\ ms\ \&\ v_2 = \frac{v_1}{4}:$$

$$a = \frac{v_2 - v_1}{t} = \frac{\frac{\sqrt{2*9.81*h}}{4} + \sqrt{2*9.81*h}}{0.01}$$

$$Finally,\ convert\ acceleration\ from\ m/s^2\ to\ gs:$$

$$= a/9.81 \rightarrow gives\ acceleration\ during\ collision\ in\ terms\ of\ g$$

Table 5: Accelerometer Measured Vs Calculated Acceleration

Dropped Height	Calculated Acceleration	Measured Acceleration
1 ft	31.191 g	32.88 g
5 ft	69.673 g	61.95 g
10 ft	98.534 g	87.94 g
15 ft	120.681 g	100.19 g
20 ft	139.494 g	114.19 g

After comparing the two data sets it can be inferred that the accelerometer gives consistent data that is representative of the overall changes in acceleration, within a margin of error. 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.

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 housing unit is 3-D printed using PLA material and is 50mm in height (plus 30mm as the height of the mounting unit, which will attach to part of the bat handle) with a radius of 34mm. The housing unit holds the PCB, which includes the MCU, IMU, HC-05 Bluetooth Module and the battery boost/charging circuit, has a small slit for access to the mircoUSB to charge said battery, and is attached to the handle via a connector with 2 prongs to firmly connect the two pieces. Through the process of collecting data, validation, and various drop/shock tests, the housing unit 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 housing unit is moderately unobtrusive, however depending on the hand placement of the user and swing follow-through, the device might be considered in the way. This is mostly a consequence of switching to the HC-05 module as it has a height

of 38mm which inflates the overall height of the housing unit by 30mm, as without it the PCB only has a height of approximately 8mm. The device as a whole weighs approximately 85.91 ± 0.01 grams, which is within range specified for unobtrusive weight, which was no more than 100 grams.

The housing unit also has a few features for the user's convenience. The most important is that the unit has an alignment divot used to properly attach the device to the bat, the divet should align with the backside line of the bat to ensure machine learning can accurately calculate the position of the hit. The device also has an easy to use switch with two settings, on and charging. The unit’s charging port is to the left of the alignment divot and above it is a window to be able to confirm the device is either on or charging. The housing unit it a closed box that the user will not have easy access to, however for image purposes the lid is taped to the unit to show a top down perspective.

Figure 15: Picture of Housing Unit On and Off Bat

3.6. *Subsystem Conclusion*

The Bluno Beetle BLE microcontroller works as designed and is able to utilize a classic bluetooth module, the HC-05, 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. The housing unit is robust and as unobtrusive as possible within the current limitations of the PCB. 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*

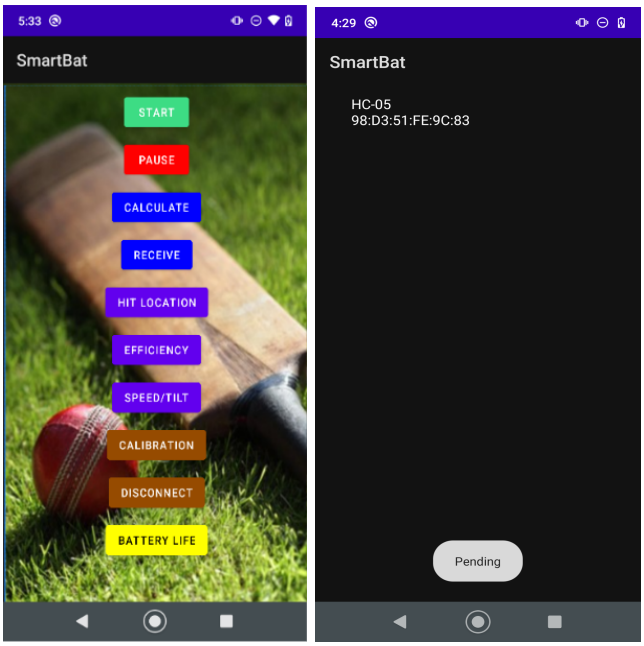
The android app has two main functions: communication between the control subsystem and the ML algorithm and handling user interaction. The app will receive the swing data gathered by the MCU via a bluetooth connection which will then be sent to the ML algorithm, that will run on a AWS ec2 instance, to be processed. Once processed and results have been received, 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 requires minimal technical skill 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 AWS cloud computer(EC2) and the control subsystem. To establish a bluetooth connection between the MCU, we will be using an HC-05 Bluetooth module. As soon as the user opens up the app, a list of their paired devices will appear on their screen to allow the user to connect to our device. The device list will show the device name as well as its MAC address. Once connected the paired device list will collapse to display the dashboard. To begin data collection, Before each swing, the user will press the designated “start” button on the dashboard that will then send a signal to trigger the MCU to send the swing data. Once the user has completed their swing, they press the “Stop” button to conclude their data collection. Apart from receiving data from the MCU, that app will also send trigger signals via bluetooth to the MCU to begin system calibration and determine the battery life of the device. Every time the device is first powered on the device must be calibrated by pressing the “Calibration” button on the dashboard. The app will then prompt the user with the calibration status and when the calibration is complete. To view the battery life percentage of the device, the user must press the “Battery Life” button on the dashboard. An image of the dashboard and bluetooth page is shown below. An image of the dashboard and paired devices list is shown in Figure 16.

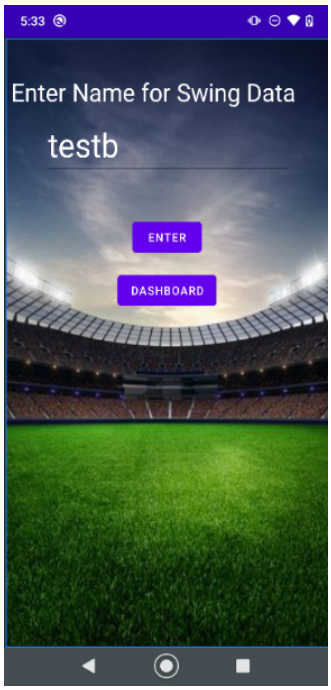
Figure 16: Dashboard and Paired Devices list



4.2.2. Sending Gathered Data to Cloud

Once the user has concluded their data collecting for the swing, they can press the calculate button to open up a window to allow them to enter a name they would like to give the data. The user has been given control of what name they would like to name their swing data to be able to know which data they are viewing the AWS cloud database. The Data sending page along with an example of a name is shown in Figure 17. Once they have entered a name, the user can press the “Enter” button and they will be prompted that data was sent to the AWS cloud database. The user can now press the dashboard button to return to the dashboard.

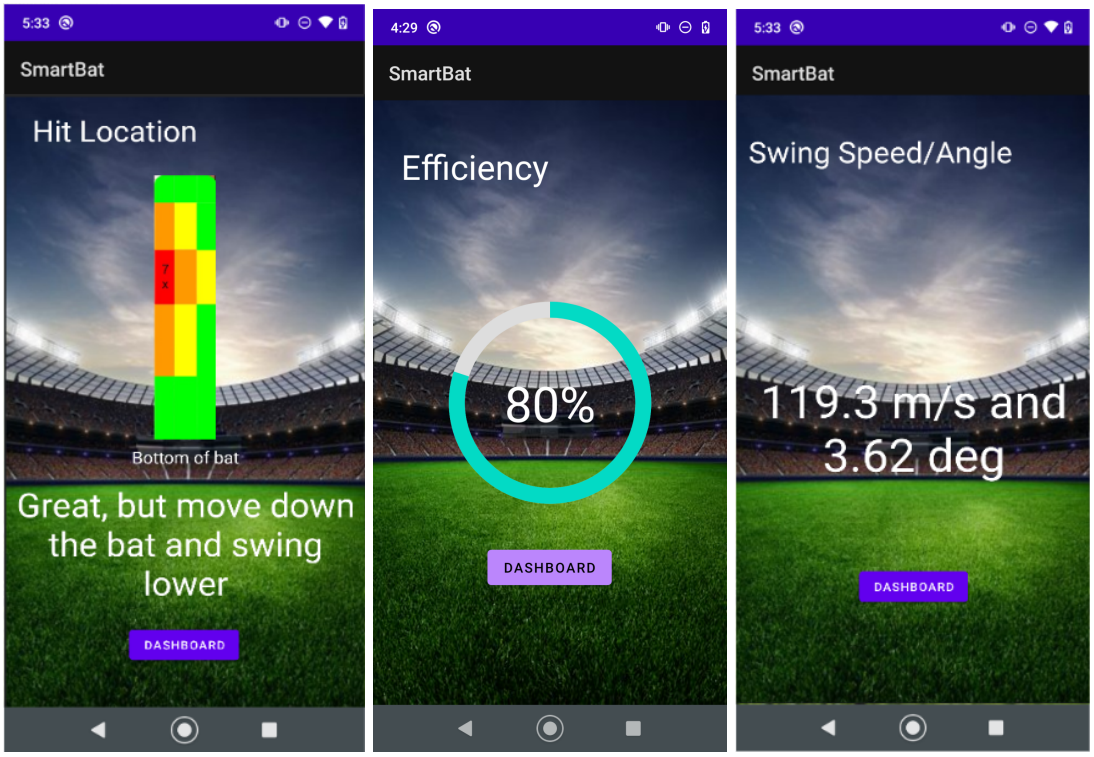
Figure 17: Data sending to AWS cloud page



4.2.3. User Interaction / Dashboard

The user interaction is controlled by the dashboard seen in Figure 16. 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 the “Receive” button. The user can now view their results by pressing the designated buttons. The result types are as follows: Hit Location, Efficiency, Swing Speed and Tilt. 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 in the form of a heat map where the most probable hit location is shown in red. 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 in Figure 18. In this example the user struck region 7 with a speed of 119.3 m/s and tilt of 3.62 degrees. This region is near the “sweet spot” of the bat on the lower edge, therefore the advice in the hit location page states “Great, but move down the bat and swing lower”. The location was near the sweet spot but on the lower edge, so the player needed to swing lower to get closer to the center of the bat. Since the collision was close to the “sweet spot" of the bat (region 8), but a bit off, its corresponding efficiency is 80%. The user can then navigate back to the dashboard by clicking on the dashboard button displayed on each page.

Figure 18: User Feedback/Results

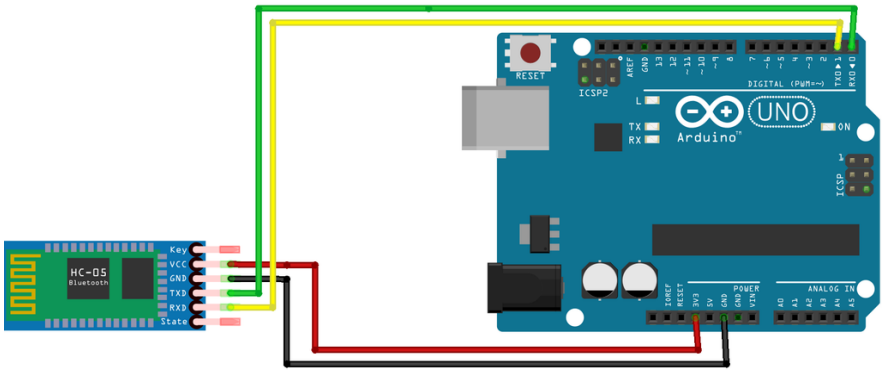


4.3. *Subsystem Validation*

4.3.1. Validation From 403

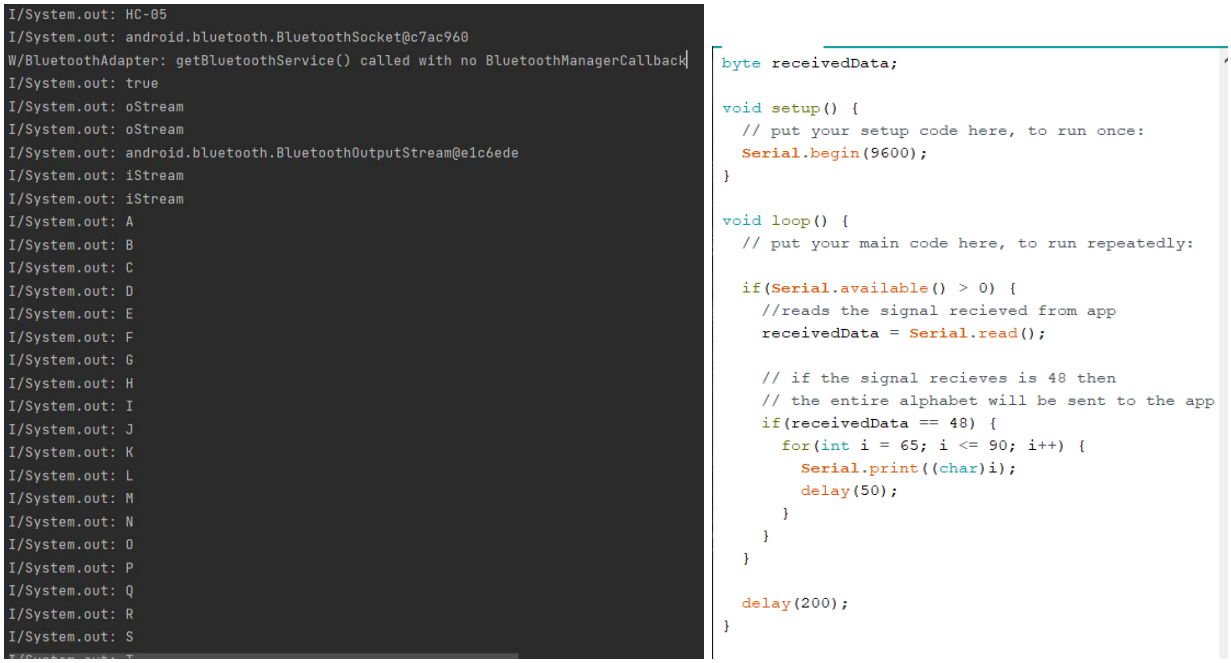
Since the MCU, the Bluno Beetle, uses the Arduino IDE and schematic, I separated the HC-05 bluetooth module and connected it to an Arduino Uno to be able to validate without needing the control subsystem. The circuit that will be used to connect the HC-05 module with the Arduino and Bluno Beetle is shown in Figure 19.

Figure 19: 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. 20. 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. 20.

Figure 20: Bluetooth Validation Data and Arduino Code



```
I/System.out: HC-05
I/System.out: android.bluetooth.BluetoothSocket@c7ac960
W/BluetoothAdapter: getBluetoothService() called with no BluetoothManagerCallback
I/System.out: true
I/System.out: oStream
I/System.out: oStream
I/System.out: android.bluetooth.BluetoothOutputStream@e1c6ede
I/System.out: iStream
I/System.out: iStream
I/System.out: A
I/System.out: B
I/System.out: C
I/System.out: D
I/System.out: E
I/System.out: F
I/System.out: G
I/System.out: H
I/System.out: I
I/System.out: J
I/System.out: K
I/System.out: L
I/System.out: M
I/System.out: N
I/System.out: O
I/System.out: P
I/System.out: Q
I/System.out: R
I/System.out: S
```

```
byte receivedData;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:

  if(Serial.available() > 0) {
    //reads the signal recieved from app
    receivedData = Serial.read();

    // if the signal recieves is 48 then
    // the entire alphabet will be sent to the app
    if(receivedData == 48) {
      for(int i = 65; i <= 90; i++) {
        Serial.print((char)i);
        delay(50);
      }
    }
  }

  delay(200);
}
```

4.3.2. Validation From 404

4.3.2.1. Bluetooth Validation

To validate the bluetooth connectivity and data sending between the MCU and the android app, a temporary page was added to the app to view the data that is being sent via bluetooth once the “Send” button is pressed on the temporary page. An image of the device next to the app displaying the data being sent via bluetooth in the temporary page is shown in Figure 19. The app was also programmed to display the data received to the Android Studio console as another form of validation. The data received should be composed of seven columns. First column is the time in milliseconds. The next three columns are the XYZ data of the Gyroscope, and the last three columns are XYZ data of the Accelerometer. A screenshot of the Android Studio console displaying the proper seven data columns is shown in Figure 20.

Figure 21: Bluetooth Validation with Temporary Page

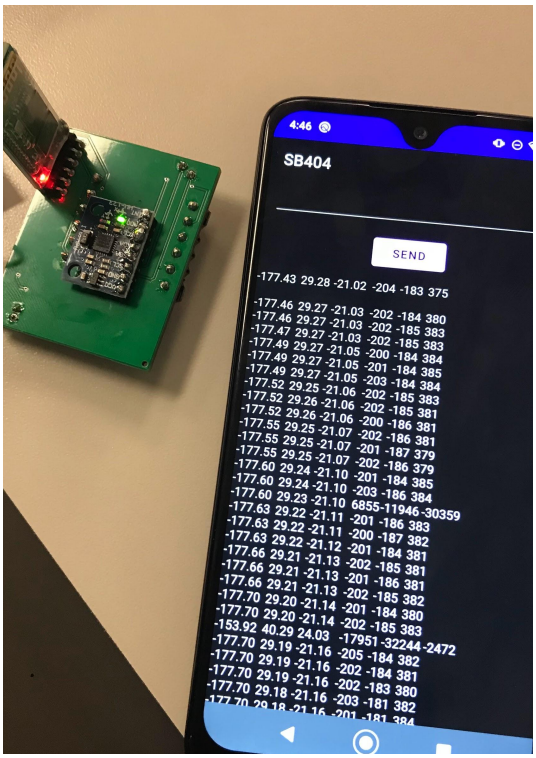


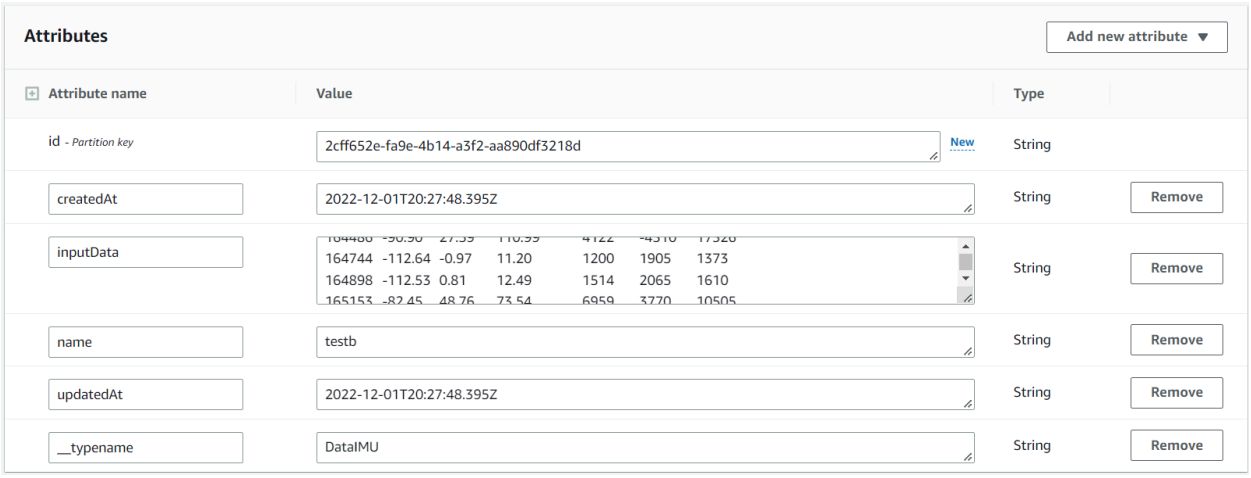
Figure 22: Android Studio Console Displaying Proper Data

	Time	Gyro: XYZ			Acce: XYZ		
I/System.out	100414	0.00	-0.24	-0.20	1135	-2052	-5713
I/System.out	100414	0.00	-0.24	-0.20	1135	-2052	-5713
I/System.out	100770	-0.88	-9.51	12.95	2307	-4828	819
I/System.out	100414	0.00	-0.24	-0.20	1135	-2052	-5713
I/System.out	100770	-0.88	-9.51	12.95	2307	-4828	819
I/System.out	100924	-0.91	-9.69	13.21	2282	-4792	830
I/System.out	100414	0.00	-0.24	-0.20	1135	-2052	-5713
I/System.out	100770	-0.88	-9.51	12.95	2307	-4828	819
I/System.out	100924	-0.91	-9.69	13.21	2282	-4792	830
I/System.out	101178	-2.09	-15.34	21.53	1530	-3710	1341
I/System.out	100414	0.00	-0.24	-0.20	1135	-2052	-5713
I/System.out	100770	-0.88	-9.51	12.95	2307	-4828	819
I/System.out	100924	-0.91	-9.69	13.21	2282	-4792	830
I/System.out	101178	-2.09	-15.34	21.53	1530	-3710	1341
I/System.out	101434	-2.70	-18.05	25.50	1203	-3226	1660
I/System.out	100414	0.00	-0.24	-0.20	1135	-2052	-5713
I/System.out	100770	-0.88	-9.51	12.95	2307	-4828	819
I/System.out	100924	-0.91	-9.69	13.21	2282	-4792	830
I/System.out	101178	-2.09	-15.34	21.53	1530	-3710	1341
I/System.out	101434	-2.70	-18.05	25.50	1203	-3226	1660
I/System.out	101691	-3.09	-20.16	28.49	969	-2872	1932

4.3.2.2. Data Sending to S3 bucket Validation

To validate and ensure proper data sending to AWS Amplify API that stores our data in the cloud, the user was given control of what they would like to name the data they collected after each swing. This will allow them to match the name they entered to the name of data shown in AWS Amplify API. A screenshot of the data sending to AWS Amplify API app page with a name of “testb” for the data is shown in Figure 15. Once the user presses the “Enter” button, the App will then send the collected data via bluetooth to AWS Amplify API which can then be exported to our EC2 instance for our ML model to calculate the results. A screenshot of the AWS Amplify page showing our data from MCU and the name of “testb” the user entered is shown in Figure 21 signifying successful data sending to AWS Amplify API.

Figure 23: AWS Amplify Page Showing Data Received



4.4. *Subsystem Conclusion*

The android app is easy to use with minor technical skills needed. 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 able to communicate and receive data from the MCU via a bluetooth connection and send the data to the AWS cloud to be processed by our ML model. A link to the github repository containing the app source code is given below.

https://github.com/pablobarron7/SmartBat_Capstone_2022_Team22

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.

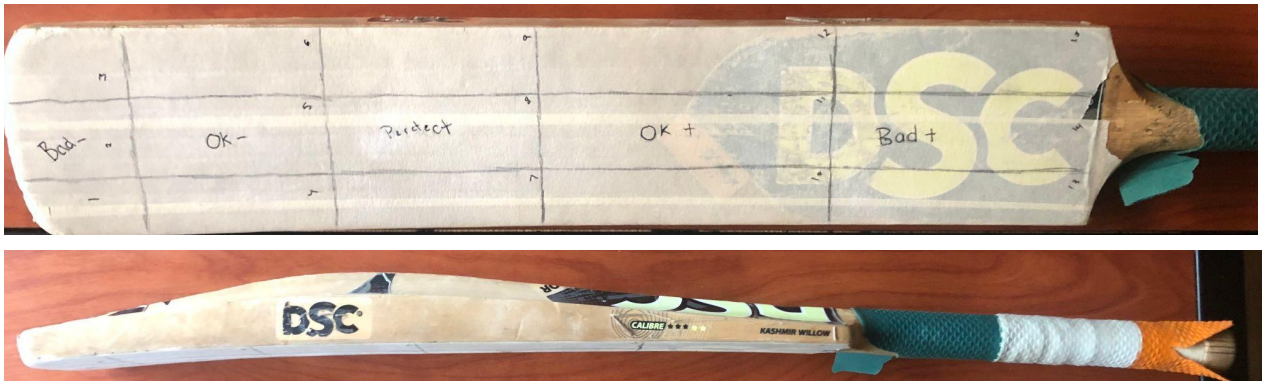
5.2. *Machine Learning Details*

The Machine Learning subsystem is done on python jupyter with tensor and scikit-learn environments then it is implemented to a python file called sv.py and put on the AWS server.

5.2.1. Bat Region Dividing

Based on the research “Determination of the “Sweet Spot” of a Cricket Bat using COMSOL Multiphysics” by The University of West Indies, 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. The thickest portion of the bat was marked as “perfect", the area which encompasses the “Sweet Spot” that batters usually refer to. For the rest of the regions, they were named 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 24: Bat Division



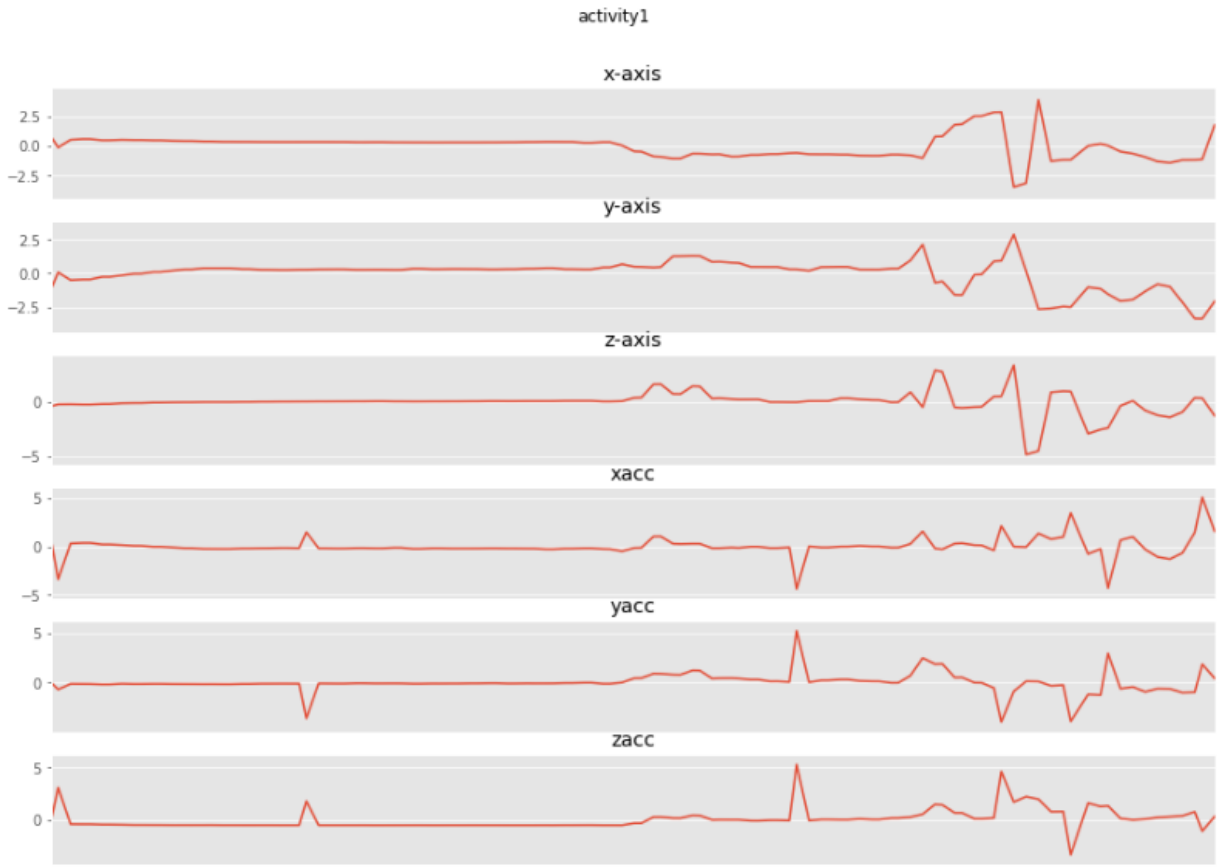
5.2.2. Data Gathering, Inputting, and Plotting

Our sponsor Pranav helped us do around 200 professional swings after we found a way to keep the orientation, but most hits are located in region 5 and 8. 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 “prefect 8”.

Figure 25: Data Plotting

```
C:\Users\hjk0811\Downloads\data3\8\8-3.txt
     timestamp  x-axis  y-axis  z-axis   xacc   yacc   zacc
0       620680   34.01  -30.90  -20.30   5239    724   1768
1       620833   -7.43   -2.88  -13.45 -15682  -3540  19920
2       621093   21.12  -15.29  -13.11   2510   -235    705
3       621347   23.60  -14.27  -13.86   2800   -273    657
4       621502   23.53  -14.21  -13.90   2808   -301    657
..         ...     ...     ...     ...    ...    ...    ...
111     644038  -64.01  -25.80  -56.24  -5416  -3159   4636
112     644297  -53.59  -50.40  -38.78  -2295  -5411   5055
113     644556  -53.14  -76.45    8.02   8047  -5189   7189
114     644712  -51.20  -76.73    6.94  25956  11383  -2965
115     644970   74.95  -49.15  -50.71   8735   3068   4690

[116 rows x 7 columns]
```

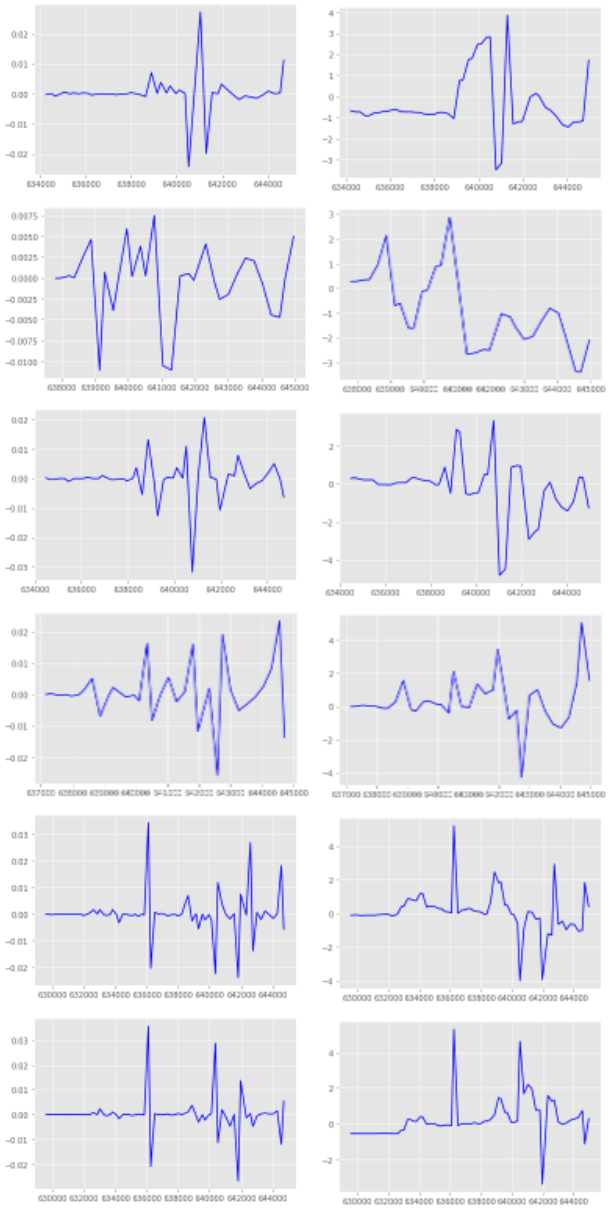


In the image, “x-axis”, “y-axis”, and “z-axis” are corresponding to the gyroscope and “xacc”, “yacc”, and “zacc” are corresponding to the accelerometer.

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, 32 values taken before and 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 26: Impact Window

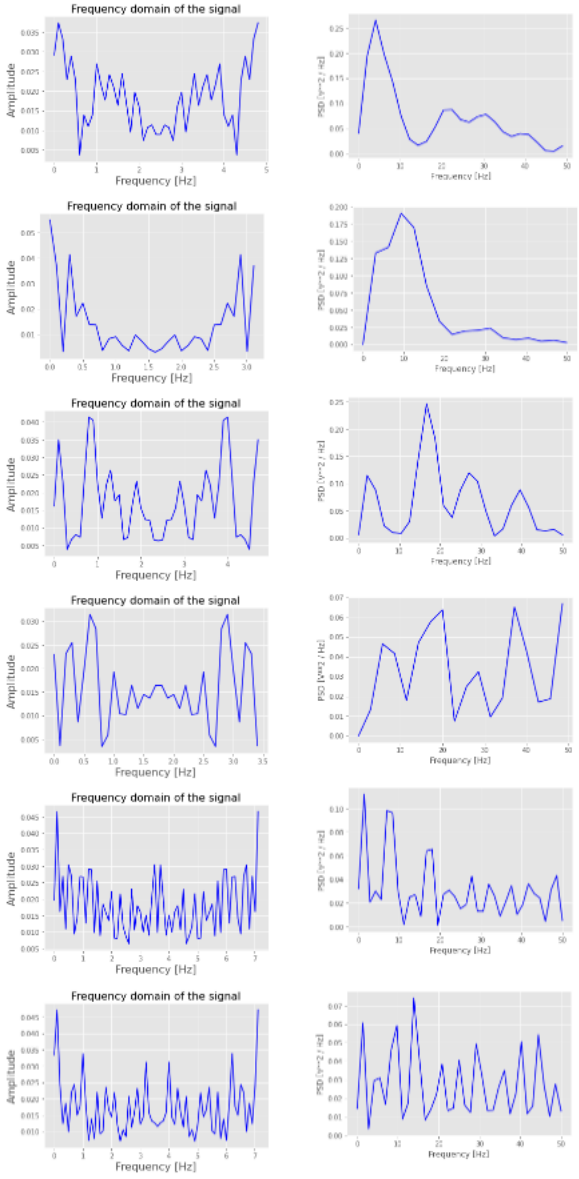


Timestamp used to be transferred here from hour, minute, sec, microsec format to total microseconds. Now, it's directly taking microseconds as input and saves a version in seconds format for other calculations.

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 data to get started on identifying the data. From all the FFT we got, we also calculate the total energy to be later used as a feature. In figure 25, left group of graphs is the FFT and the right is the PSD.

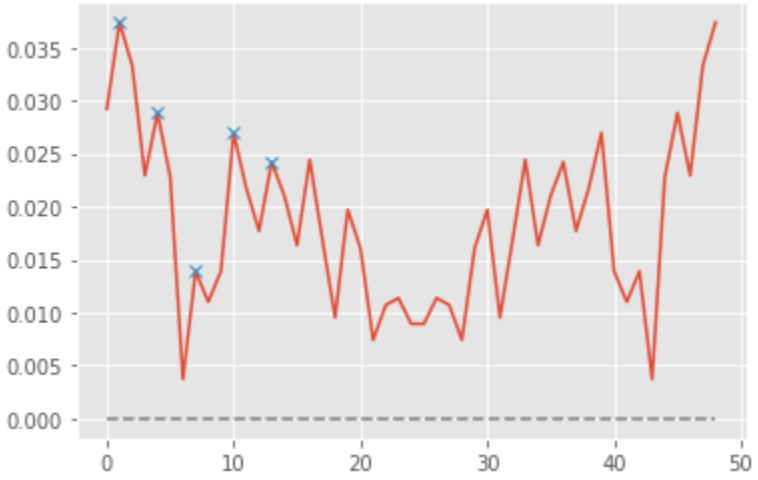
Figure 27: FFT & PSD



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. This is our peak energy detection that forms our first 120 feature which are 5 peaks * 2 coordinates for each peak (X and Y) * 2 graphs for each axis (FFT and PSD) * 6 axis from IMU.

Figure 28: Peak Detection on FFT over X-axis Gyroscope of Region 8



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 first 120 features and the labels in the image shown below. Total energy for all 6 axes from IMU and total impulse for all x,y, and z axis for the bat, were also added to total up to 129 features.

Figure 29: Feature and Label

```
[1, 0.03735847897165938, 3, 0.04141884322596024, 1, 0.03496258046753683, 3, 0.025450269479166454, 1, 0.04651838322108
504, 1, 0.04716110252071357, 2, 0.26583810778298l9, 3, 0.19079147220392945, 1, 0.11478825800110669, 2, 0.046572492268
47999, 1, 0.11213897977967563, 1, 0.0607882974112672, 4, 0.028845927924546152, 5, 0.022262168046083466, 5, 0.00800910
0562404958, 6, 0.03145026668201492, 3, 0.026909104805321087, 4, 0.01859886415411125, 11, 0.08692972605561729, 10, 0.0
2355781111805536, 8, 0.2460503126932096, 7, 0.06375506511569687, 3, 0.029905584783063755, 4, 0.03081831506092497, 7,
0.013899779957530566, 10, 0.0090515465534876 94, 8, 0.041340431735207 99, 10, 0.01924892827331978, 5, 0.030398169548108
228, 7, 0.024351506839236696, 15, 0.07742929773665966, 13, 0.009094472410564613, 13, 0.11958060573633612, 10, 0.03248
336217151577, 5, 0.09819986386691953, 7, 0.05931398066210309, 10, 0.026944467860569567, 13, 0.009786821695673964, 13,
0.026333704214750852, 13, 0.01637910969661312, 9, 0.026840290882389813, 10, 0.033871003403002566, 19, 0.0389740238301
6928, 15, 0.0060331461461047495, 19, 0.08813048914783374, 13, 0.0651924530369745, 10, 0.02689192930973262, 10, 0.0741
3198806403157, 13, 0.0241993010989335, 19, 0.009786821695673964, 15, 0.01935020891102702, 15, 0.01446116426258974, 1
2, 0.02899764474889994, 13, 0.013792303844802262, 19, 0.03897402383016928, 15, 0.0060331461461047495, 23, 0.015499770
760184646, 13, 0.0651924530369745, 13, 0.0653979502231611, 15, 0.03849105984173307, 0.020275511784260104, 0.012301219
864403698, 0.020087663971489648, 0.010805404898733266, 0.029236123085759264, 0.026547743916735915, 14426753.09724816,
8861387.32354223, 23015076.64287367]
8
```

The same process is done for all 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 70% for training and 30% for testing through trial and error.

5.2.8. Training the model

The data will be trained over Random Forest Classifier. We have done more than 6 models and over 20 combinations like different portions of Train Test Split, or random shuffle with Random Forest, Decision Tree, Logistic Regression, svm with kernel, rbf, linear or ploy and many more. From the results and the pattern of the accuracy increasing and decreasing, the solution was to simply get more training data. Because for most regions, there were only 5 to 10, or even fewer, swings while some regions like 5 and 8 have the most amount of data, at around 30 swings. As a result, this result with Random Forest Classifier with 70% training and 30% testing is the best split available with our amount of data, and it can predict region 5 and 8 very well thanks to the amount of data available. In the image shown below the top is the accuracy on training and test set, in the middle is the specific precision and details for prediction over each region. In the bottom part is the predicted results and the actual result.

Figure 30: Training Results

```
Accuracy on training set is : 0.6428571428571429
Accuracy on test set is : 0.24074074074074073
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         2
           2       1.00      0.25      0.40         8
           3       0.00      0.00      0.00         2
           4       0.22      0.33      0.27         6
           5       0.30      0.60      0.40        10
           7       1.00      0.20      0.33         5
           8       0.09      0.50      0.15         4
           9       0.00      0.00      0.00         1
          11       0.00      0.00      0.00         5
          12       0.00      0.00      0.00         1
          13       0.00      0.00      0.00         2
          14       0.00      0.00      0.00         6
          15       0.00      0.00      0.00         2

    accuracy                           0.24        54
   macro avg       0.20      0.14      0.12        54
weighted avg       0.33      0.24      0.21        54

[8 5 8 8 4 5 8 4 5 4 5 4 5 8 8 8 5 5 4 5 8 5 4 8 8 8 8 5 8 5 8 8 5 4 2 8 5
 5 5 5 8 2 8 8 5 7 8 8 8 4 4 5 5 5]
[ 5  4  8  4 11  5  2  2 14  5  3  8  2  2  4 13 14 14  4  5  5  8  4 12
 11 14  2  5 15 11  7  4  2  9  2  5  5 11  5  3 13  2  8  1 14  7  7 14
  1  7  7  5 15 11]
```

A confusion matrix was then generated to spot whether they could predict the results correctly.

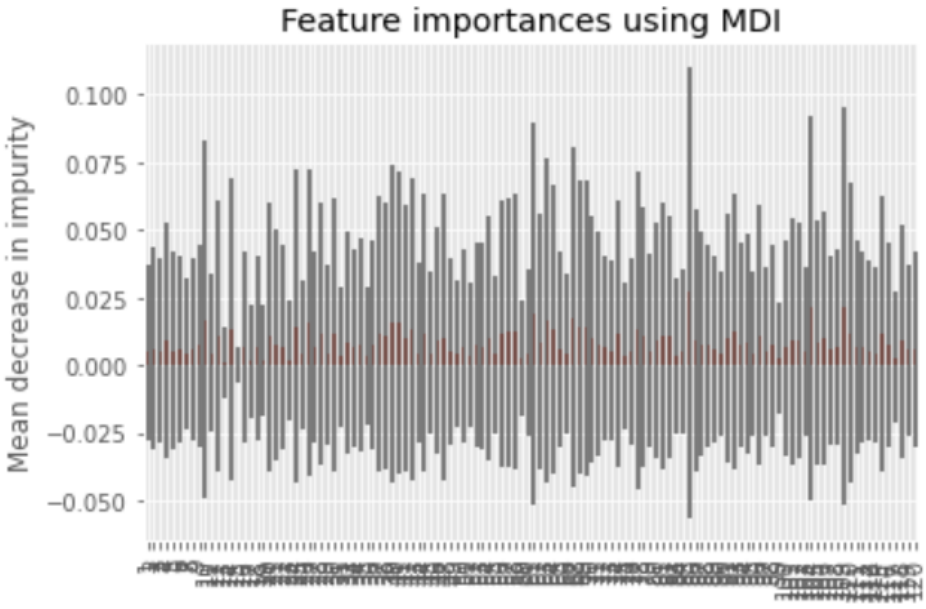
Figure 31: Confusion Matrix

```
[[0 0 0 0 0 2 0 0 0 0 0 0 0]
 [0 0 5 2 0 2 0 0 0 0 0 1 0]
 [0 0 3 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 3 0 4 0 0 0 0 0 2 0]
 [0 0 0 0 2 2 0 0 0 0 0 0 0]
 [0 0 1 0 0 3 0 0 0 0 0 1 0]
 [0 0 0 1 0 0 0 0 0 0 0 1 0]
 [0 0 0 1 0 3 0 0 0 0 0 0 0]
 [0 0 2 2 0 2 0 0 0 0 0 0 0]
 [0 0 1 0 0 1 0 0 0 0 0 1 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 1 1 0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 0]]
```

5.2.9. Feature Importance

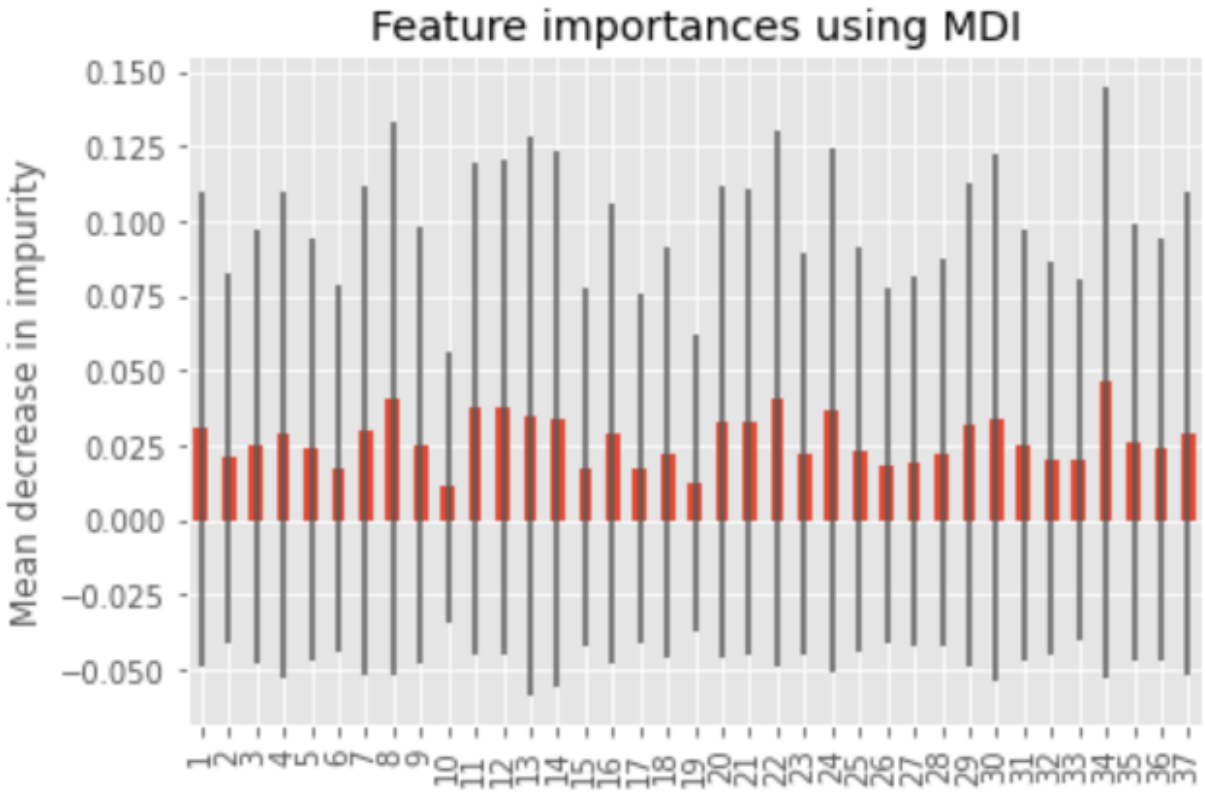
Then the ML will use feature importance to find out all the important features to increase accuracy.

Figure 32: 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 33: Feature Importance 2



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

Figure 34: Feature Importance from

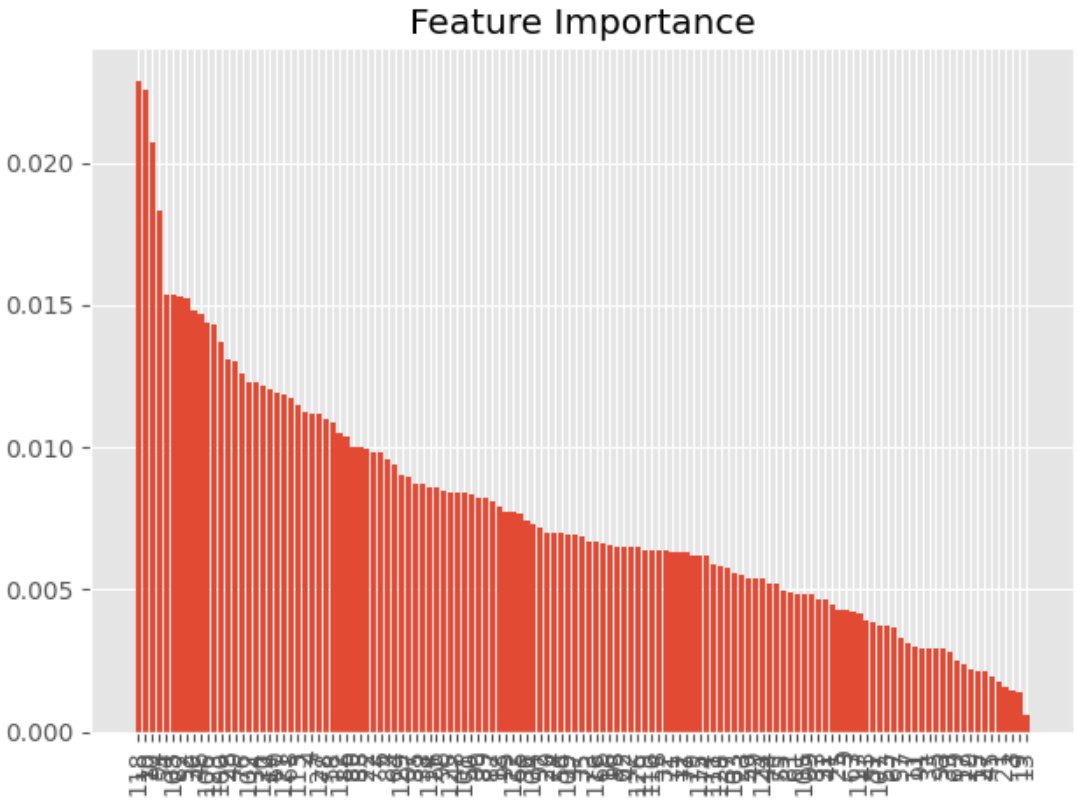
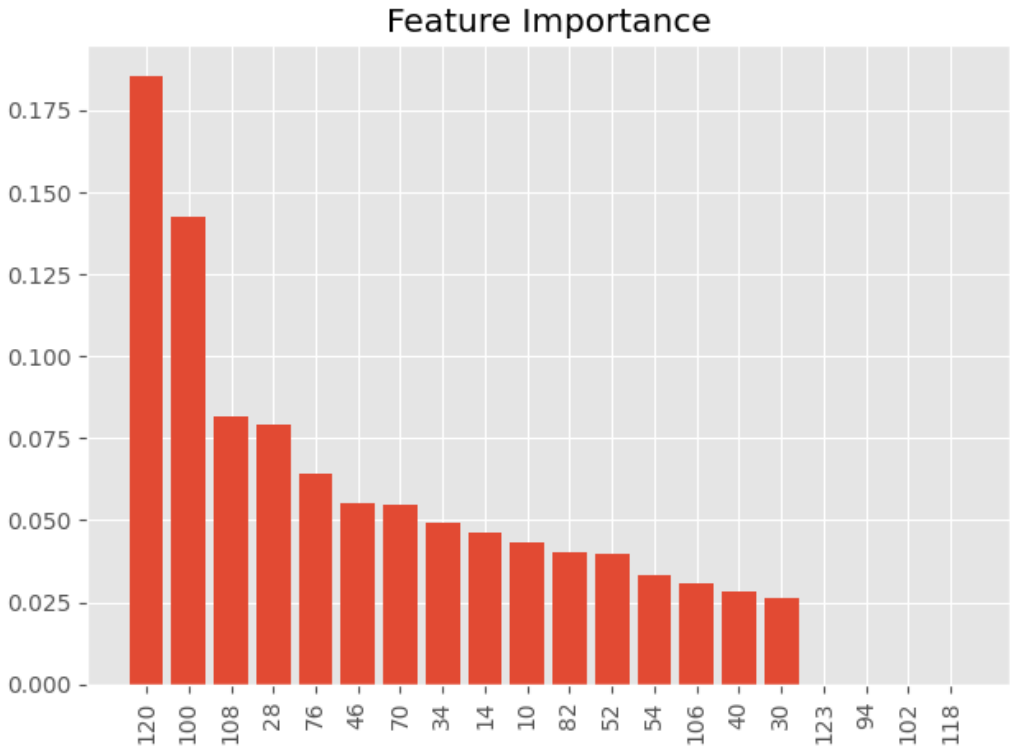


Figure 35: Feature Importance to



Feature importance was used to get the top 20 important features, which were then analyzed to see whether keeping some important features over others could improve accuracy.

5.2.10. Correlation, association, and relationships

Figure 36: Heat Map

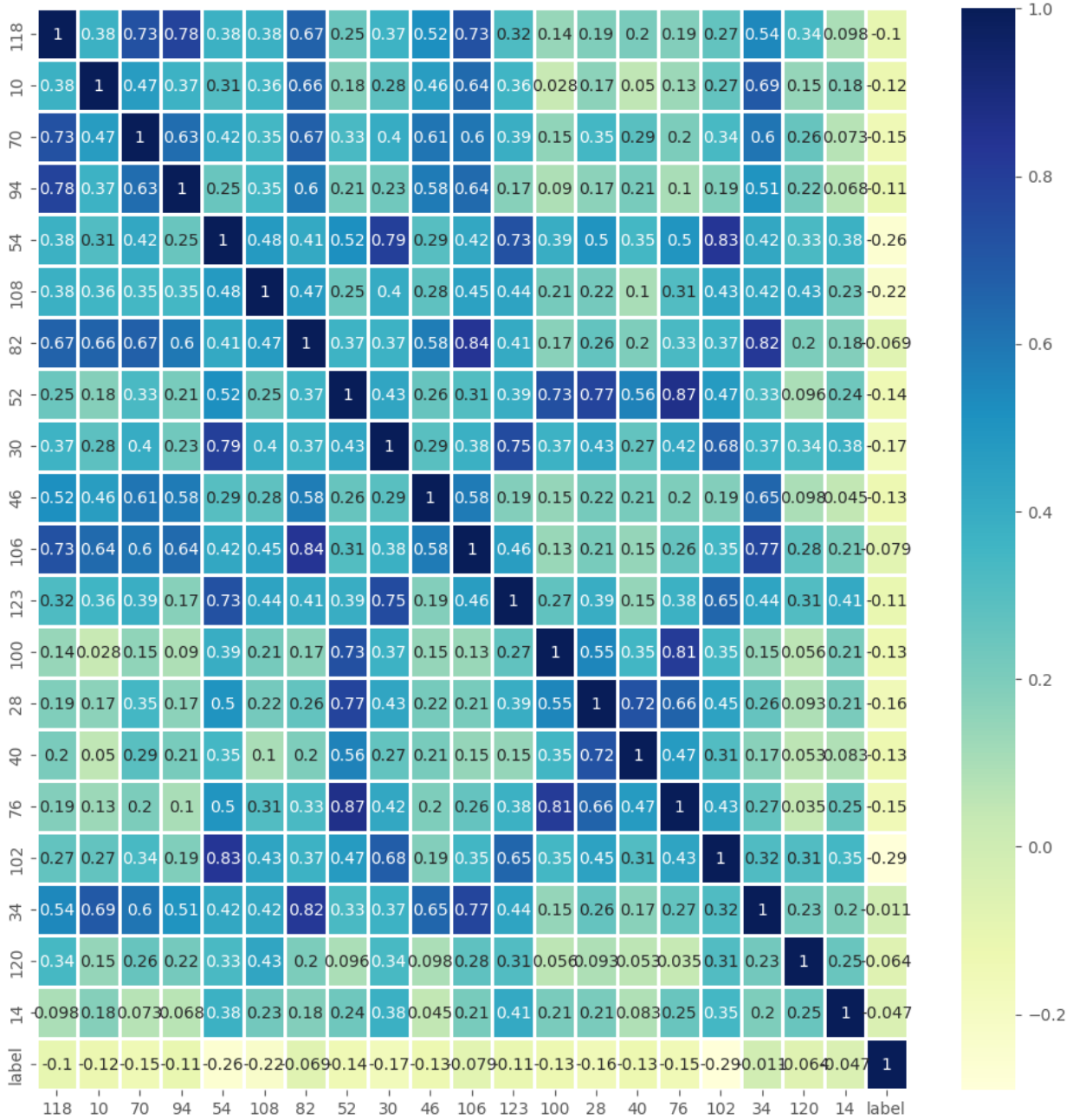


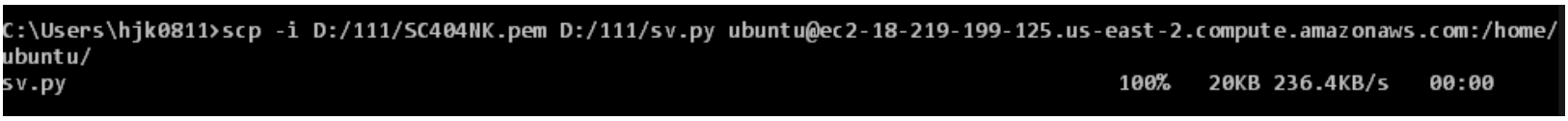
Figure 37: Seaborn Pairplot

From the seaborn pairplot and the heatmap, one can spot the correlation between features and the relationship between features and the labels. From here, one can analyze and choose better features to use. However, through all the comparisons, ther seemed to be no significant amount of increase in accuracy. So, the same model, with all 129 features, was used.

5.2.11. ML integration

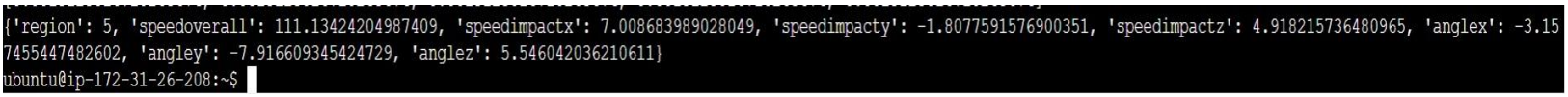
The ML model is uploaded to the AWS server for the integration with Android Studio. In order to do that, the ML model must be uploaded to the EC2 instance on the AWS server.

Figure 38: Upload ML to EC2 instance



Then a connection is set up between the S3 bucket and the EC2 instance, so we can receive the json file that Android Studio sent to S3 bucket and copy that file from S3 to EC2, then the Python code is run that uses the model we generate and the json file that just copied from S3 bucket. Then the ML model will generate a result also in json format named in sample.json with hit region, overall speed, impact speed on each axis, and angle on each axis.

Figure 39: Example output on EC2 instance



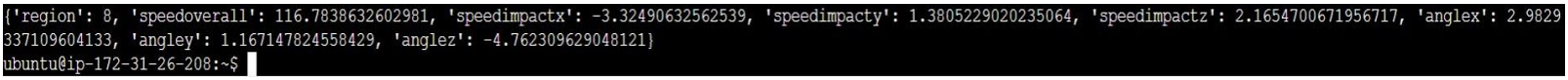
Then we will copy the generated json file from the EC2 instance to the S3 bucket.

5.3. *Machine Learning Validation*

5.3.1. Validation for actual hit on the bat

For the validation of the ML subsystem, we tried to hit the ball with the bat and chalk with the physical location that the hit was landing as the result. We validated and put one of each hit on each region on the whole system report. The ML is mostly predicting region 5 and 8 since this is the data it had the most of. Figure 40 is an example of hitting on region 8 that we did and the corresponding output that the ML output on EC2 instance of the AWS server. It correctly predicted region 8 as its collision with an overall speed of 116.78 m/s and an angle of -4.76 degrees in the z axis which runs along the bat and corresponds to the tilt of the bat.

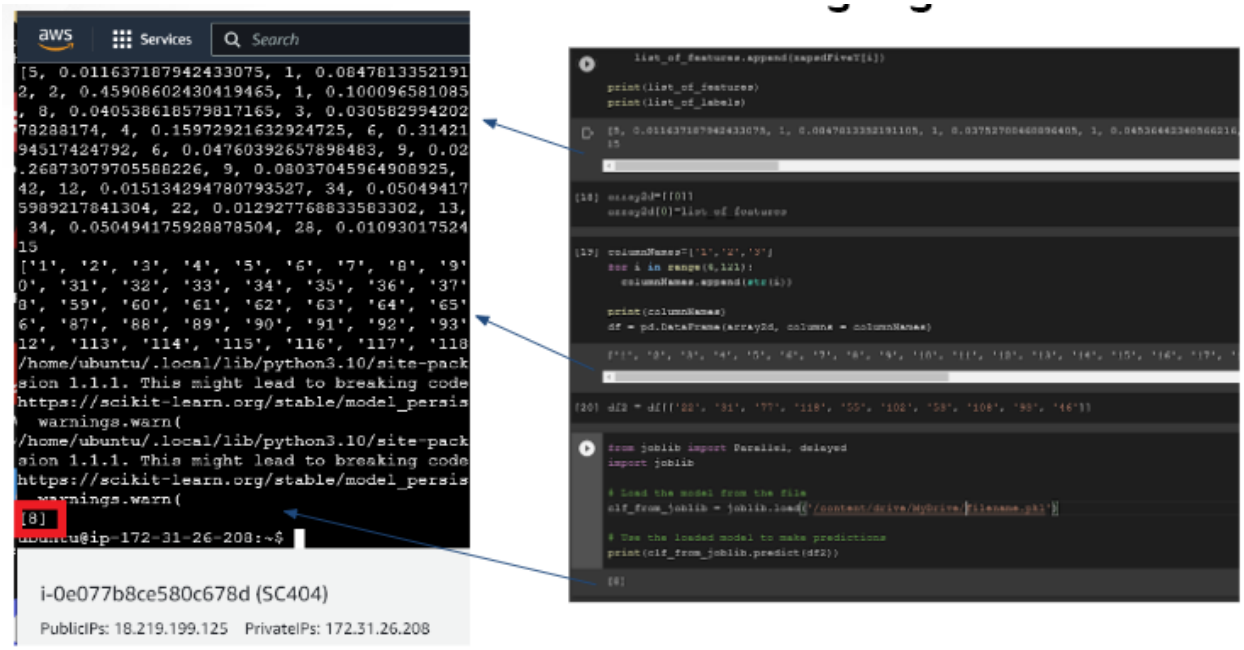
Figure 40: Hit location on the bat and ML output on EC2 console



5.3.2. ML integration on EC2 Validation

To validate if the ML successfully uploaded to EC2, we tested if the ML is outputting the same results on the EC2 server and running with Colab.

Figure 41: Same ML output on EC2 and Colab



5.4. *Subsystem Conclusion*

The Machine Learning subsystem is able to receive the data and do the process needed like normalizing, windowing, speed and angle calculation, FFT, PSD, peak energy detection,

total energy calculation, angular impulse calculation, feature extraction, training, and testing. In order to have better accuracy, we still need to do much more data. With a lot of validations we did, the ML can predict region 5 and region 8 really well, because they have the most amount of data. Region 4 and 7 are also having considerable results since they also have more data than the others. 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. The downside is it can only predict region 5 and region 8 much better than the others.