

ENG 4000
FINAL REPORT

PREPARED BY



Bio Animal Big Data
YORK UNIVERSITY
LASSONDE SCHOOL OF ENGINEERING

MEMBERS

Min Jae Kim

Rishabh Bhatnagar

Xin Chen

Xingkai Wu

1 Table of Contents

1 Table of Contents	1
2 Table of Figures	5
3 Acknowledgements	7
4 Executive Summary	8
5 Section I - Technical Volume: As-Built System	9
5.1 Final Scope	9
5.2 Requirements Review	10
5.2.1 User Requirements and Verification	10
5.2.2 System Requirements and Verification by Tests	12
5.2.2.1 Battery	12
5.2.2.2 Embedded System	14
5.2.2.3 Firebase	17
5.2.2.4 Application	18
5.2.2.5 Regulatory	20
5.2.2.6 Safety	20
5.2.3 Verification Workflow	22
5.2.4 Project Stakeholders	23
5.3 Laws, Regulations, Guidelines, Intellectual Property Laws, Best Practices	24
5.4 Ethical Testing	25
5.5 Developing Sustainable Technology	25
5.3 As-Built Design	26
5.3.1 Embedded System	26
5.3.2 Server	30
5.3.3 Application	32
5.3.4 Enclosure	35
5.4 As-Built Design Compliance Analysis	37
5.4.1 Remark About Non-Conformance Report	37
5.4.2 Embedded System	38

5.4.2.1 As-Built Design Usage Restriction	38
5.4.2.2 Battery	39
5.4.2.2.1 Compliance Analysis Chart	39
5.4.2.2.2 Verification Status	40
5.4.2.3 Micro-controller	42
5.4.2.3.1 Compliance Analysis Chart	42
5.4.2.3.2 Verification Status	44
5.4.2.4 Accelerometer	45
5.4.2.4.1 Compliance Analysis Chart	45
5.4.2.5 Temperature Sensor	47
5.4.2.5.1 Compliance Analysis Chart	47
5.4.2.5.2 Verification Status	47
5.4.2.6 Enclosure	48
5.4.2.6.1 Compliance Analysis Chart	48
5.4.2.6.2 Verification Status	48
5.4.3 Application	55
5.4.3.1 MVP Design Pattern	55
5.4.3.2 Recyclerviews with Different Types of Views and Abstractions	56
5.4.3.3 Real-Time Graph	59
5.4.3.4 Application Structure	61
5.4.3.5 Ensuring Compliance (Verification Status)	63
5.4.4 Firebase Server	65
6 Section II - Financial and Management Volume: Supporting Documents	67
6.1 As-Built Work Breakdown Structure	68
6.2 As-Built Work Package Descriptions: Work Breakdown Structure	70
6.3 As-Built Resource Allocation Matrix	71
6.3.1 Originally Planned Resource Allocation Matrix	71
6.3.2 As-Built Resource Allocation Matrix	72
6.3.3 Summary & Changes since the PDR	74
6.4 As-Built Project Schedule	75

6.5 As-Built Project Procurement/Equipment/Travel List	77
6.6 Preliminary Business Case	78
6.6.1 Strength Weakness Opportunity Threat Analysis (SWOT)	79
7 Section III - Lessons Learned Volume: Supporting Document	80
7.1 Deviations From Plan	80
7.2 Failure Report	82
7.3 Lessons Learned	83
7.3.1 Defining the Problem to Solve	83
7.3.2 Finding the Solution to the Problem	83
7.3.3 Prototyping	83
7.3.4 Product Development	84
7.3.5 Product Testing	84
7.3.6 Back-end Migration From MySQL and PHP to Firebase	85
7.4 Possible Improvements in the Problem Solving Process	86
7.5 Recommendations for Possible Improvements	88
8 Deliverables and Setup	89
8.1 Setting up BioABD Android Application	89
9 Section IV - Project Self-Evaluation Matrix	91
9.1 Compliance with Laws, Regulations, Intellectual Property Rights	91
9.2 Employing Strategies of Reflection, Assessment, Self Assessment, of Team Goals in Multidisciplinary Settings	93
9.3 Adhering to Written Instructions in a Professional Context	93
9.4 Evaluating Critical Information in Reports and Design Documents	94
9.5 Appraise Possible Improvements in the Problem Solving Process	95
9.6 Justifying the strengths and Limitations of the Solution and Making Potential Solutions	96
10 Appendix 1: Application Screenshots	98
11 Appendix 2: Parsing code for values from Firebase	101
12 Appendix 3: Interfaces updating graph in real-time	103
13 Appendix 4: FireBase Crash Tracking	106
14 Appendix 5: Settings and Global Variables	107

15 Sources	109
15.1 ESP8266 Specification Sheet	109
15.2 Li-Po Battery Technology Specification	109
15.3 ADXL345 Accelerometer Data Sheet	109
15.4 Dog Sports Harness	109
15.5 BioABD Android Application GitHub Link	109
15.6 Android vs. iOS Market Share	109
15.7 Using Acceleration to Estimate Distance and Velocity	110
15.8 Agile vs. Waterfall	110
15.9 MP Android Chart - Android Charts	110
15.10 Android Resource Monitoring Application	110
15.11 Android MP Prolific Calendar	110
16 Components	111
16.1 Microcontroller	111
16.2 Battery	111
16.3 Temperature Sensor	111
16.4 Accelerometer	111
16.5 Remark	111

2 Table of Figures

Figure 1 TR Youtube Video	21
Figure 2 Workflow for verifying the system	22
Figure 3 As-Built system diagram	26
Figure 4 Microcontroller High Level Diagram	27
Figure 5 Cost of individual components	29
Figure 6 Firebase web interface	30
Figure 7 Diagram of high level server structure showing input and output	31
Figure 8 Diagram of high level application and firebase connections	32
Figure 9 Global market share of Android, iOS, and other mobile OSs. By Statista	32
Figure 10 Dashboard of the application	34
Figure 11 3D CAD file of the enclosure (Angle 1)	35
Figure 12 3D CAD file of the enclosure (Angle 2)	35
Figure 13 Verification setup	40
Figure 14 Oscilloscope output	40
Figure 15 Verification of battery under low temperature environment	41
Figure 16 Verification of battery under high temperature environment	41
Figure 17 Schematic of the ESP-8266 Board	43
Figure 18 Verification (From TR Video)	44
Figure 19 ADXL345 pins	45
Figure 20 Verification (From TR video)	46
Figure 21 Cooling down and warming up the temperature sensor	47
Figure 22 System with harness	53
Figure 23 The harness use cases	54
Figure 24 Header view for dashboard, the dummy view, the number 60 is a placeholder, not a real value.	55
Figure 25 Hard Coded XML showing two fonts taken from the official developers.android.com website	57
Figure 26 Photo of Recyclerview. This is how every screen in the BioABD Application actually looks.	57
Figure 27 Creating a new page in the apps	59
Figure 28 Contract shared between the micro-controller, server, and application.	60
Figure 29 Raw string received from Firebase.	60
Figure 30 Parsed variable of raw JSON string from Firebase.	60
Figure 31 Diagram of the different activities and their purposes.	61
Figure 32 Diagram of inheritance of activities as well as many levels of abstractions. Note: the Extends goes from left to right, so Dashboard Activity extends Abstract Data Activity	62
Figure 33 Diagram of Adapters	62

Figure 34 Diagram global variables	62
Figure 35 Table showing compliance	63
Figure 36 Application APK size	63
Figure 37 CPU and RAM usage for application	64
Figure 38 Contract shared between the micro-controller, server, and application.	65
Figure 39 Summary of testing of Firebase	66
Figure 40 WBS (part 1)	68
Figure 41 WBS (part 2)	69
Figure 42 Resource allocation matrix from the PDR	71
Figure 43 As-Built Resource allocation matrix (part 1)	72
Figure 44 As-Built Resource allocation matrix	73
Figure 45 Project Schedule (part 1)	75
Figure 46 Project Schedule (part 2)	76
Figure 47 Project expense list	77
Figure 48 SWOT diagram	79
Figure 49 Angle Error	80
Figure 50 Usage of the database in the past 30 days as of 1/4/2018 (DD/MM/YYYY).	86
Figure 51 Command for cloning master branch of BioABD app from Github.	89
Figure 52 Option to import project into Android Studio.	89
Figure 53 Run option	90
Figure 54 Select device to build to	90
Figure 55 Code for parsing values from Firebase	101
Figure 56 Contract for a correctly parsed value from Firebase.	102
Figure 57 Code for announcer variable from Appendix 2.	103
Figure 58 Listener that adds value to the graph	104
Figure 59 Listener that adds value to the data for the graph to then plot	105
Figure 60 Evidence of crashless application	106
Figure 61 Code that sets the global variable to the desired value	107
Figure 62 Code that sets the global variable to the desired value	107
Figure 63 Code that checks if x range should be limited, creating a floating graph	107
Figure 64 Floating Graph	108
Figure 65 Non-floating graph	108

3 Acknowledgements

The BioABD team would like to thank Professor Ebrahim Ghafar-Zadeh for his vital guidance on this project and his commitment to the group. As well as his vital knowledge of technical electrical engineering concepts, allowing us to complete the project.

The team would like to thank Professor Suzanne MacDonald and Amanda Nickerson for providing us guidance on animal testing, granting us the ability to test on animals, and for her commitment to helping us complete the project with specific requirements.

4 Executive Summary

The BioABD project aims to give animal psychologists, animal trainers, and other professionals a way to monitor statistics of their animals. BioABD is intended to be used by animal psychology professor Suzanne MacDonald who intends to utilize the BioABD system in an animal behavioural study. The BioABD system will track the three-axis acceleration (showing movement and the level of activity of the animal), temperature, and the heart rate of the animal using the ESP8266 micro-controller. The heart rate monitor will be implemented in the future.

The ESP8266 microcontroller sends the temperature and three-axis acceleration data to the back end server at a sample rate of one measurement per second. BioABD then stores this information on the google hosted back end service called Firebase and shares this information with professionals in a real-time chart on an Android application. Firebase is intended to be used as a real-time database so the storage and retrieval of information on Firebase is straight forward when developing a mobile application. BioABD additionally aims to be inexpensive by using inexpensive micro-controllers and sensors.

The potential uses for this project outside of its current scope are the addition of more sensors. The project is built in a way that the addition of multiple sensors is simple. An example of this is a heart rate sensor added to BioABD allowing animal psychology professors to track heart rate as well (Note that this heart rate sensor would have been included in the final BioABD product but the sensor was backlogged and was not available in time). Another example is adding an additional sensor for velocity allowing for the tracking of speed accurately and adding more utility for animal trainers. This project can also be marketed to the general public, not just professionals due to the low cost of the microcontroller and sensor. This means that the initial “buy-in” to the tracking ecosystem is very low and with a simple internet connection many statistics about a pet can be tracked easily.

5 Section I - Technical Volume: As-Built System

5.1 Final Scope

BioABD set out to solve the problem of allowing professionals such as animal trainers or animal psychologists to track certain characteristics about their animal. The final scope of BioABD represents this. While it may differ from the original scope, it represents the requests of the primary stakeholders (Animal Psychology professor Doctor Suzanne MacDonald) perfectly. The final scope of BioABD was to measure the acceleration the dog experiences in 3-dimensions, along with temperature, using the ESP8266 micro-controller. The micro-controller would be powered by a battery to be portable, read the sensor data connected to it, calibrate the values received to meaningful information, then send this information to the server in a specific format. Firebase would be responsible for storage of information and the management of the real-time database. Firebase would notify any applications active that new data has been added. Firebase would support multiple users. Lastly, the application would receive this information from firebase, parse this information, and then display it. The application would download older information that was on the server as well as newer information in the scenario where the microcontroller was sending data to the server and the application was started late.

5.2 Requirements Review

Given the purpose of BioABD, to help professionals including animal phycologists and trainers, and give them real-time access to 3-dimensional acceleration and temperature data, the system requirements of the individual components of BioABD are as follows.

5.2.1 User Requirements and Verification

Requirement	Description	Verification
Reliability	<ol style="list-style-type: none"> 1. Long battery life. 2. Uninterrupted data stream. 3. Accurate data measurement. 	Verified in the System Requirements and Verification section (next section).
Functionality	<p>Data measured by all the sensors.</p> <ol style="list-style-type: none"> 1. Data received by the micro-controller from the sensors. 2. Data interpreted, formatted, and published to the Firebase server. 3. Data retrieved from the Android device. 	Verified in the System Requirements and Verification section (next section).
User interface	<ol style="list-style-type: none"> 1. Intuitive and easy to use user interface (UI). 	The application design was based off Google Material Design standards proven to be intuitive and easy to use. In addition to this, some application designers from TD were consulted for the UI and positive feedbacks were given.
Placement of the device	<ol style="list-style-type: none"> 1. The micro controller must be placed on the dog in a way that it can easily be attached 	Compatible harness with mounting points will be provided with the product.

	and removed (for example, on a harness).	
Maintenance	<ol style="list-style-type: none">1. Battery must be re-charged when needed.2. Wi-Fi must be provided by either:<ol style="list-style-type: none">a. Installed Wi-Fi access points in the environment or,b. Wi-Fi hotspot provided by a mobile phone.	<p>Apply $1.2A_{max}$ and 4.2V from DC power supply, battery can be fully charged in 1 hour and 40 min from empty.</p> <p>Wifi connection manual will be provided with the product.</p>

5.2.2 System Requirements and Verification by Tests

5.2.2.1 Battery

Requirement	Description	Verification
Supply voltage	1. The battery should supply constant 3.7V DC voltage.	Measuring the positive and negative terminal of the battery by using oscilloscope. A straight line was shown on the oscilloscope screen with the value of 3.7V. Thus this requirement is satisfied
Microcontroller can be remotely powered up by using battery	1. The battery can constantly supply 3.7V DC voltage to microcontroller to keep the microcontroller in ON state 2. The 3.3V Pin should supply 3.3V to power on the accelerometer and the temperature sensor.	By using oscilloscope. Measuring the voltage between the battery Pin and the Ground, the voltage is constant 3.7V DC. Measuring the voltage between the 3.3V pin and the Ground, the voltage is 3.3V DC. Thus this requirement is satisfied
Battery performance does not change(remains stable) within the rated temperature	1. By turning on the high power fan (used to lower the temperature of high Voltage/ Current MOSFET) to decrease the temperature of the battery. 2. By turning on the the high power heat gun to 200 °F (93°C) to increasing the surrounding temperature so that battery's temperature increases to a relatively high	In both extreme conditions, the battery's voltage remains 3.7V DC and 3.3V Pin gives 3.3V output. Therefore the requirement is satisfied.

	temperature(40°C, measured using temperature sensor).	
Battery can be rechargeable when the supply voltage is not sufficient to turn on the microcontroller	1. The battery's rating is 2000mAh, therefore, by applying 1.2A _{max} and 4.2V from DC power supply, battery can be fully charged in 1 hour and 40 min from empty.	It took around 1 hour for the battery to be fully charged from empty to full at 4.2V @ 1.2A _{max} from DC power supply. Therefore this requirement is satisfied.

5.2.2.2 Embedded System

Requirement	Description	Verification
Portability	<ol style="list-style-type: none"> 1. Compact and small enough to be mounted onto a dog leash (harness). 2. Powered by a battery. 3. Data must be transferred wirelessly. 	<p>ESP micro-controller was used due to its compactness. Battery's dimension was similar to the micro-controller thus, was able to be stacked. 3D printed enclosure supports enough space for the microcontroller, the battery, and the accelerometer. When mounted on to a dog harness, it is small enough to be negligible for the dog's movement. Moreover, the weight is less than 400g. The micro-controller connects to the Wi-Fi network and all the data acquired is transferred wirelessly. This is verified by looking at the data being published to the firebase server by the microcontroller.</p>
Measurement of acceleration	<ol style="list-style-type: none"> 1. Measurement of X, Y, and Z axis component of the acceleration separately. 2. Measurement of meaningful acceleration data (up to 8 times the force of gravity for extreme measures). 3. Must be accurate. 4. Sample rate must be high (at least one per second). 5. Transferred using a widely used standard protocol (ex. 	<p>Accelerometer X, Y, Z axis data is verified by measuring the data when the sensor is stationary (or not accelerating). This should measure the force of gravity. Moreover, when the sensor is at free fall, the measured acceleration should be zero for all three components. The TR verified all these properties. Moreover, the sensor's manufacturer states that the</p>

	<p>I2C)</p> <p>6. Power efficiency.</p>	<p>sensor can read upto 16g (16 times the force of gravity). The accuracy of the data is also verified by the same TR section mentioned above. The transferred data can be verified by monitoring the I2C input and this was also verified. The power efficiency was tested and it shows that the controller can measure more than 6 hours on a single charge.</p>
Measurement of temperature	<ol style="list-style-type: none"> 1. Temperature measurements should be accurate in the range of body temperature of an animal. 2. Transferred using a widely used standard protocol 3. Power efficiency. 	<p>The temperature sensor 's accuracy was verified using the human body temperature. The data was successfully received by the micro-controller using one of the GPIO (general purpose input/output) pin. This was verified by monitoring the GPIO pin's data input on the microcontroller's COM terminal. The power efficiency was tested and it shows that the controller can measure more than 6 hours on a single charge.</p>
Calibration of sensors	<ol style="list-style-type: none"> 1. Sensors must agree with a known standard of units. <ol style="list-style-type: none"> a. Temperature sensor must agree with 0 degrees as freezing point, 36-37 degrees as a healthy human 	<p>The temperature sensor's calibration was done from the manufacturer's factory. The sensor showed 36-37 degrees when measuring the human body temperature, while it measured the correct temperature when it</p>

	<p>body temperature, and 100 degrees as boiling temperature.</p> <p>b. Accelerometer must agree with the force of gravity as 9.8 m/s^2 and free-fall as 0 m/s^2.</p>	<p>was placed in a climate controlled room. The accelerometer also agreed with the forces stated in the description (verified in the TR).</p>
Interpretation of acquired data from sensors	<p>1. Raw data from each sensor (ex. 32342) must be interpreted as a data with a standard unit (ex. 5.34 m/s^2 or 34 degrees Celsius).</p>	<p>Raw data was interpreted using the manufacturer's specified values. The TR's output values verifies its validity.</p>
Formation of interpreted data to meet the firebase query requirements	<p>1. Interpreted data given a tag (ex. Xacceleration, Yacceleration, Zacceleration, temperature, time).</p>	<p>Each data was given a tag to distinguish between the different types of data. Without the proper tags, published data to the server would not be handled. This was verified by observing the Firebase server.</p>
Publishment of acquired and interpreted data to the Firebase server	<p>1. Send the data to the Firebase server.</p>	<p>The data sent was verified by observing the received data on the Firebase server (mentioned above).</p>

5.2.2.3 Firebase

Requirement	Description	Verification
Connection	1. Establishes connection with the micro-controller and the application.	This was verified by powering up the completed microcontroller. The microcontroller then sent acceleration and temperature information to Firebase. Since Firebase received the information, this test passed.
Reception	1. Receive formatted information from the microcontroller and store it in the same format.	There was a format coded into the microcontroller, it would send the stored Unix Time, Acceleration data, and temperature data. This would be sent with specific key/value pairs. When the microcontroller was powered up, the correct data was received in the correct format. Thus this was verified.
Notification	1. Notify application when data is added.	This was verified when the application was partially completed. The application had implemented a listener that would be notified when new data was added to a specific node on the database. Then some arbitrary information was added to Firebase. After this, there was a breakpoint added to the code to see if the app would receive this data. Since the application did receive the data, this requirement was verified.

5.2.2.4 Application

Requirement	Description	Verification
Connection	1. Establishes a connection to a specific node on Firebase.	When integrating Firebase into an Android application, there is no built in method to check if the connection is live, if there is no connection the application queues the request. So to test this, the reverse was done. Data was added to Firebase. If there was a connection, the callback listener would be called inside the app. So some data was added by the microcontroller and log (ADB Logcat) statements were used to verify it was being given to the application.
Parsing	1. Parses the data given from the Firebase.	Sample data was added to Firebase and the callback sent it to the application. The application then used the model class to parse this into the appropriate objects. Thus this requirement has been met.
Display	1. Displays the data existing on Firebase. 2. Displays the latest updated data from the Firebase.	When the application opens the graph page, all old data existing on the server is downloaded. After this, as new data is added, that is also downloaded. This means that the existing as well as the new data being added to the

		server is given to the application and parsed. Then the graph is given this data one-by-one to graph. This is best seen by the full system test in the TR. This requirement has been met.
Stability	1. App does not easily crash.	Firestore allows for tracking of the application. Through all of testing, the application has never crashed. Please refer to Figure 3 for evidence. This requirement has been met.

5.2.2.5 Regulatory

Requirement	Description	Verification
Permission for animal testing	1. Acquire permission from York's Animal Care Committee for animal testing.	Doctor Suzanne MacDonald is one of the heads of the committee so she granted us permission for animal testing as long as she supervises.

5.2.2.6 Safety

Requirement	Description	Verification
Safe to use device	<ol style="list-style-type: none"> 1. Must be within a reasonably safe temperature to prevent burns. 2. Within a safe electrical rating to prevent shocking or electrocuting the animal. 	<p>All the electrical circuit and wiring are enclosed in the 3D-printed case, no exposed wires or testing points have direct contacts to the animal.</p> <p>Battery temperature testing, electrical rating testing were passed during the testing procedure (TR).</p>

The verification video can be seen in the following link (Same video as the TR):

<https://www.youtube.com/watch?v=UIK-N0HM-rw>



TR Video
Unlisted

Figure 1 TR Youtube Video

Now that the system and user requirements have been defined, it must be shown how BioABD meets these requirements.

5.2.3 Verification Workflow

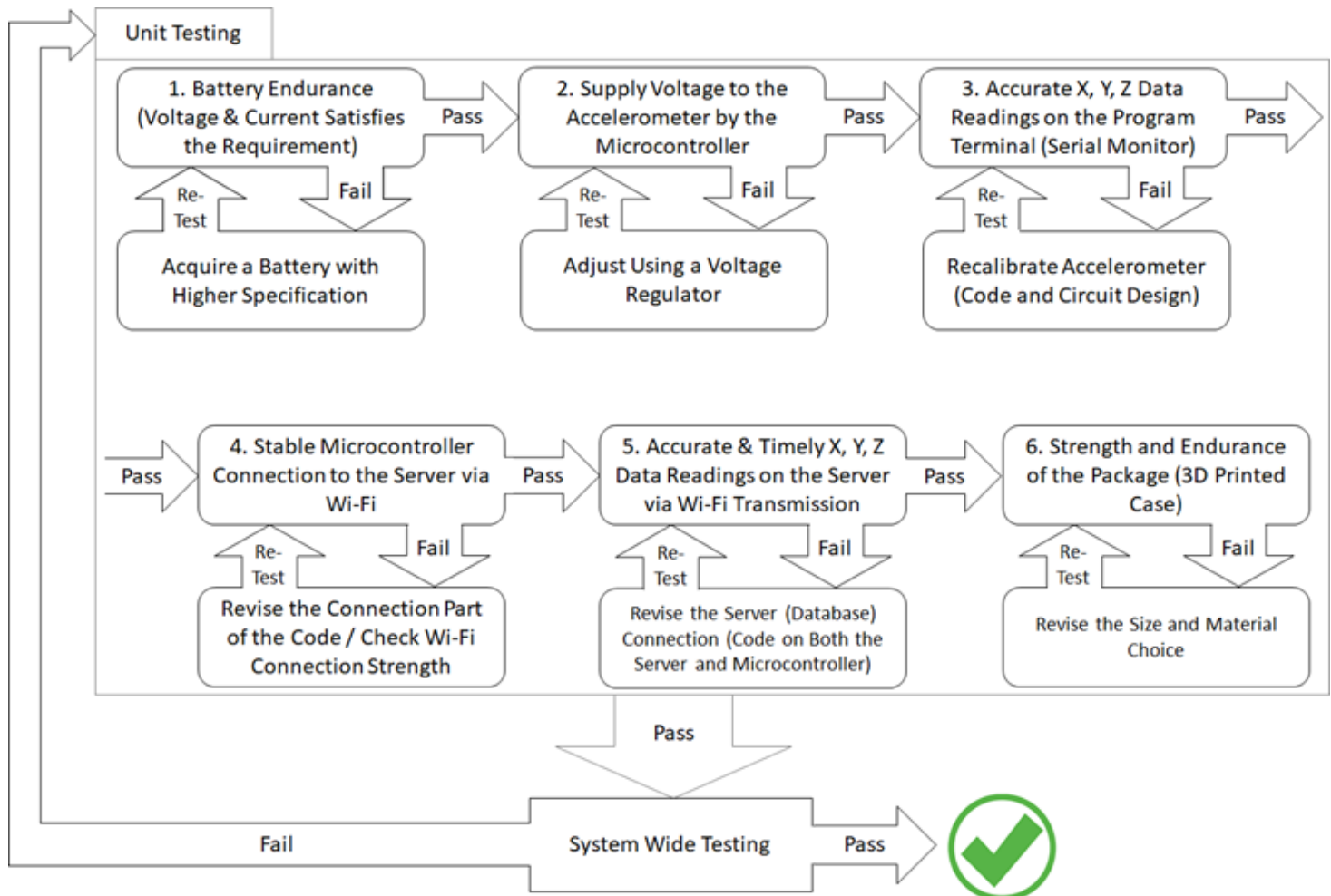


Figure 2 Workflow for verifying the system

5.2.4 Project Stakeholders

The following were the main stakeholders in BioABD. These stakeholders had a major impact on the project design and requirements. They also played a large impact during the development of BioABD as well as provided vital input.

1. Animal Psychologists and Big Data Researchers (Client + Users) - This group has an interest given the research and data analysis that can be performed using this project. They have had a major input on the scope of the project.
2. Dog trainers and professionals (Users) - This group has an interest in the project due to the team's ability to provide a service and a product that can help them track different attributes of their dog and potentially improve the physical performance of their animals.
3. Professor *Ebrahim Ghafar-Zadeh* - Professor Ghafar-Zadeh has a tremendous effect on the project. He has given us a lot of advice on methodology of how to do the technical side of the project.
4. BioABD Team - The team has a vested stake in the success of this project as the success of the project means that the successful application of the theories learnt throughout Engineering as well as proving that the team can deliver regardless of tight deadlines and changing requirements.

5.3 Laws, Regulations, Guidelines, Intellectual Property Laws, Best Practices

Laws, regulations, guidelines, intellectual property laws, and best practices are vital to follow. This is because they are intended to hold a certain standard in products. Following these rules ensures the safety of both the user, the developing company, and the environment. An example is the regulations that prevent lead to be used in pipes for homes. Lead in pipes could cause serious harm to humans. This should obviously not happen. This is an extreme example of why following laws, guidelines, regulations, best practices and more is necessary and important. Intellectual property is extremely important as well. It allows the creator of an innovative technology to use that technology with little fear of having others copy it, thus incentivising more people to create innovative technology.

BioABD follows all laws, regulations, guidelines, intellectual property laws, and best practices that the team is aware of. An example of this is the intellectual property of the graph and chart used in the application. Anyone is free to use them as long as they are cited appropriately and people do not steal the code and claim it to be their own. BioABD has cited MP Android Chart and Prolific Calendars used.

5.4 Ethical Testing

This was a very important topic for the group. It is an important topic to the group as they would not like to inflict any harm to the animal (physical or physiological). The group went to extensive lengths to ensure that the animal testing done was ethical. The help of Animal Psychology professor Suzanne MacDonald was taken for this. She ensured the group followed the guidelines set out by York's Animal Care Committee and that BioABD had the appropriate approvals. Her and her TA personally ensured that the team followed these guidelines.

5.5 Developing Sustainable Technology

When developing a product, it is very important to look at how sustainable the manufacturing and the usage of the product is. A product must use minimal resources. This is good for the user, as he or she will have to charge the device less but is also good for the environment. BioABD is built to be sustainable. It uses a microcontroller and a sensor. These are the only components that are required to be purchased, and thus they are the only required components of BioABD that must be manufactured. The other components are firebase and the application, which run on servers and devices the user likely already owns. Due to the small size of the microcontroller and the sensors, and the reusable nature of BioABD (since the current implementation can be taken off one harness and placed on another), BioABD has minimal effect on the economy. An example of this, if BioABD is purchased as it is now, the end user can remove the encasing for the microcontroller and sensors and be placed on another.

5.3 As-Built Design

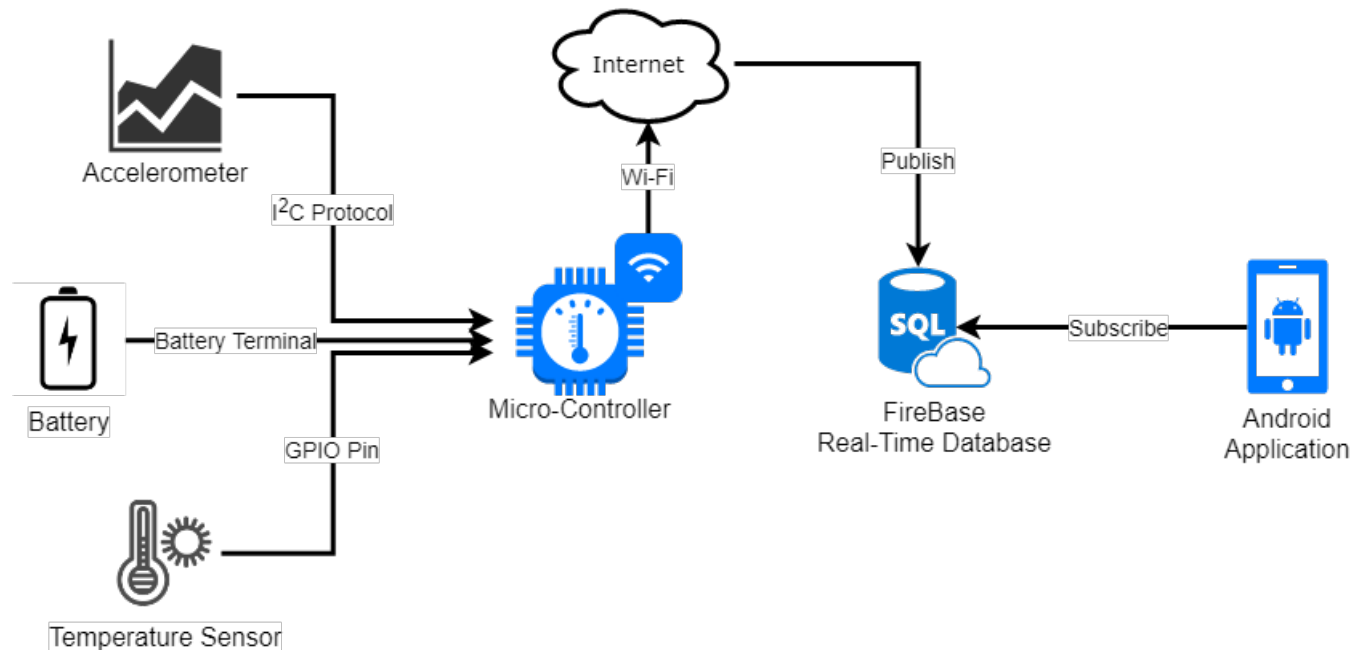


Figure 3 As-Built system diagram

5.3.1 Embedded System

The main hardware (embedded system) used to collect the acceleration and temperature data was the ESP-32 micro-controller board and micro-controller board. The ESP series board was used as it supported the I2C communication protocol (used to communicate between the micro-controller and the accelerometer) and GPIO pins (used to receive data from the temperature sensor). Moreover, the micro-controller supported Wi-Fi, with its built in antenna and Wifi controller chip. The ESP8266 is the predecessor to the ESP-32. While the predecessor is more affordable, the successor has more features with a more powerful computing processor. Since both boards were compatible with the same sensors, codes, and batteries, the team acquired both boards and tested the difference. We found that the additional processing power did not have any impact on the performance on the BioABD application so thus, the team decided to use the ESP8266 micro-controller board.

The embedded system (board with the accelerometer and the temperature sensor) is mounted on an animal's harness. The micro-controller takes the acceleration and temperature data from the respective sensors and send it to the Firebase server using its Wi-Fi capability.

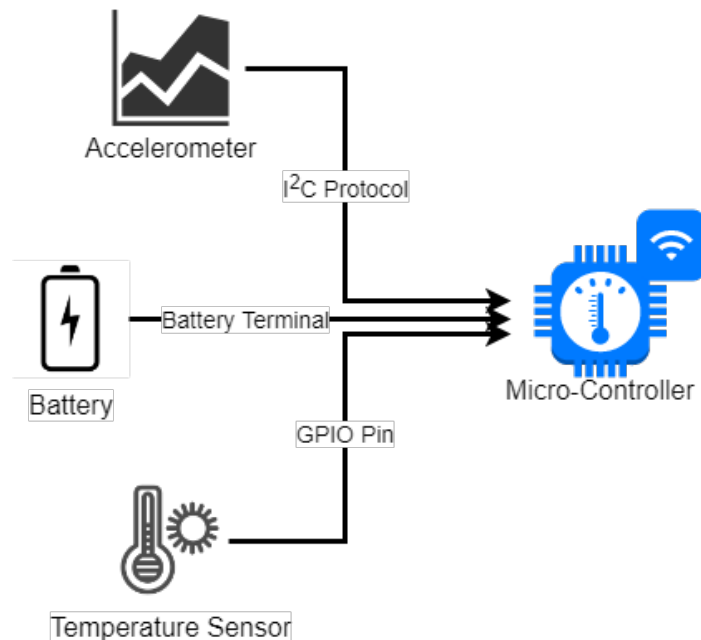


Figure 4 Microcontroller High Level Diagram

The micro controller then takes these values and calibrates them to useful numbers (for example, 234 to $7.45 \frac{m}{s^2}$). This information was then structured in a format that will be accepted by the server, and the information is then sent as a package to the server.

The following code shows the Wi-Fi connection, Firebase server connection, acceleration request, temperature request, and transmission of data to the Firebase server.

```
// Import Libraries
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL345_U.h>
#include <OneWire.h>
#include <DallasTemperature.h>

// Define Constants
#define FIREBASE_HOST "bioabd-662cf.firebaseio.com"
#define FIREBASE_AUTH "Cq08PSmZMvpMkMH0xIhVbajug5Cc5Y0tSoYcGIMX"
#define WIFI_SSID "MJK"
#define WIFI_PASSWORD "A12086477198990"

// Assign ID to the accelerometer
Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345);

// Assign pin for the temperature sensor
#define ONE_WIRE_BUS 2
// Setup for the temperature sensor
```

```
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

void setup() {
  Serial.begin(9600);
  // Attempt connection with given SSID and Password
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("connecting");

  // Connect to Wi-Fi
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }

  Serial.println();
  Serial.print("connected: ");
  Serial.println(WiFi.localIP());

  // Connect to Firebase server with the given host name and authentication code
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  #ifndef ESP8266
  while (!Serial);
  #endif

  // Sensor initialization
  if(!accel.begin())
  {
    // Sensor detection error for the accelerometer
    Serial.println("No ADXL345 detected. Check the wiring.");
    while(1);
  }
  // Maximum measured force @ 16g
  accel.setRange(ADXL345_RANGE_16_G);

  // Start the temperature sensor
  sensors.begin();
}

void loop() {
  // Request the temperature
  sensors.requestTemperatures();
  sensors_event_t event;

  // Request the acceleration
  accel.getEvent(&event);
  StaticJsonBuffer<200> jsonBuffer;
  JsonObject& root = jsonBuffer.createObject();

  // Measure the time in ms and send it to the server
  root["unixTime"] = millis();
  // Measure the x y z acceleration values and send it to the server
  root["xValue"] = event.acceleration.x;
  root["yValue"] = event.acceleration.y;
  root["zValue"] = event.acceleration.z;
  // Measure the temperature and send it to the server
  root["tempValue"] = sensors.getTempCByIndex(0);
  // Push the measured data to the server
  String name = Firebase.push("/adx1345", root);
  // Error handler
```

```
if (Firebase.failed()) {  
  Serial.print("pushing /logs failed:");  
  Serial.println(Firebase.error());  
  return;  
}  
// Delay for 1 sec and repeat  
delay(1000);  
}
```

The embedded system is one of the two main parts of the BioABD tracking ecosystem (other being the UI provided by the app). Affordable components were used to keep the BioABD product appealing. The cost breakdown is shown on the following chart:

Component	Cost
Microcontroller	\$24.00
Battery - 3 pack	\$12.50
Temperature Sensor	\$5.00
Accelerometer	\$15.00
Total	\$56.50

Figure 5 Cost of individual components

5.3.2 Server

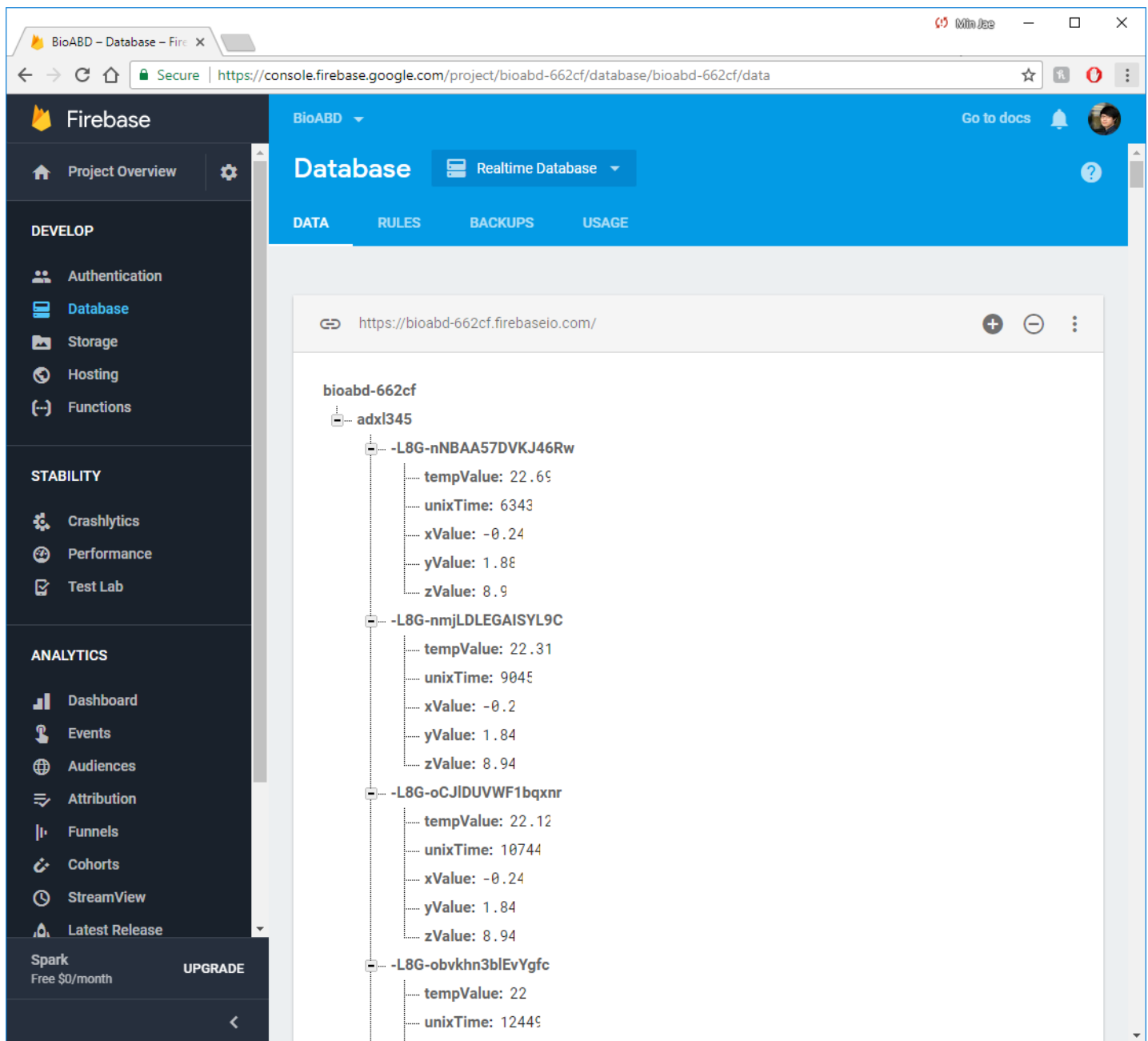


Figure 6 Firebase web interface

Firebase was chosen as the server. Firebase is a Google hosted solution that is specifically meant to integrate into applications. While Firebase is not specific to mobile, the integration between the Firebase database and the application is robust. Firebase is a document-oriented database, which is a non-relational and unstructured database that provides supreme performance compared to popular structured relational databases such as MySQL and Oracle.

The database is commonly used for real-time applications that deal with massive volumes of changing data types. Firebase also implemented its own WebSocket library called TubeSock to provide real-time event-based communication. For the BioABD system, the Firebase database is used as the main real-time database and an data archive for future data accesses. When an application opens, a socket type connection is opened between the application and Firebase. This allows for easy sending of information to Firebase from the micro-controller. It also allows the real-time database to notify any mobile applications open about the addition, removal, or editing of data on the server.

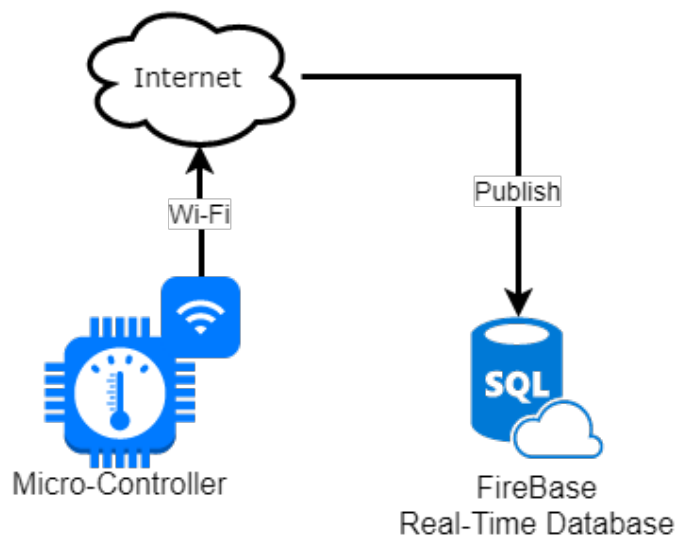


Figure 7 Diagram of high level server structure showing input and output

5.3.3 Application

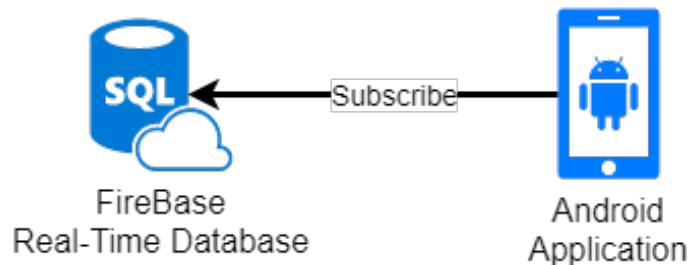


Figure 8 Diagram of high level application and firebase connections

The first two components of the project involved gathering information and then storing it. This part of the project allows for the end user to observe the data uploaded to the server. The server will rapidly update with real-time acceleration and temperature data. This application must present its data to the user in an easy-to-use and intuitive manner. To solve the problem, the team chose to code an Android application which subscribes to the Firebase server (Android application subscribes to the Firebase Server) and to use a real-time updating graph to show the real-time data. This allows for the application easy-to-use and shows any changes in their data. It is also worth noting that there are languages like Flutter (owned by Google), react, react native, and more, which allow for building one application which runs on both iOS and Android with one code-base. However, these are still buggy and in early development (react native being in beta, and flutter being in early alpha). This combined with the intensive nature of a real-time graph, the application was coded for native Android. Android was chosen because globally, it has 87.7% of the market share as of the 2nd quarter of 2017 (data according to Statista). The project can be later extended to iOS and web based application.

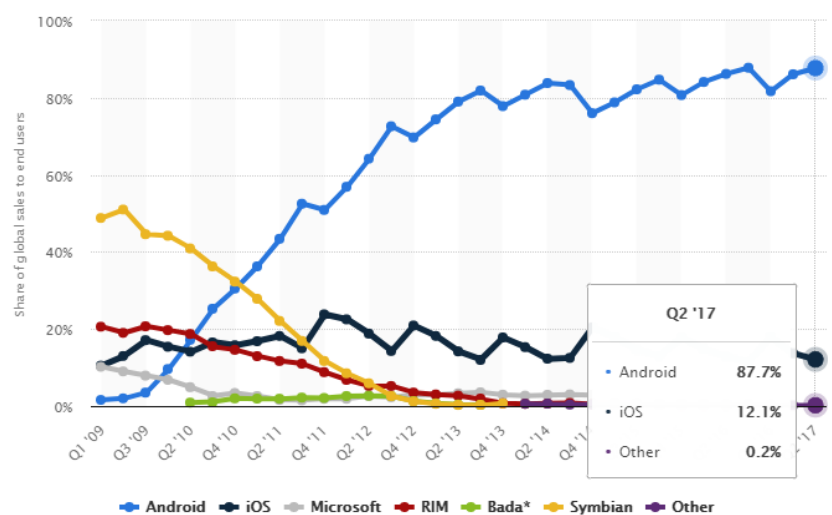


Figure 9 Global market share of Android, iOS, and other mobile OSs. By Statista

This application uses component-based programming. This means that while most other android applications have hard-coded views for most pages, BioABD application has individual components (sometimes referred to as views) added in a list (explained in the technical section).

Most companies which develop software products in the industry opt to use the Agile methodology instead of the Waterfall methodology. While the other parts of the project used the Waterfall methodology, the application was developed using Agile. Agile has a proven faster time to market and is proven to be more dynamic and adaptable than Waterfall. According to the website Delta-Matrix, Agile is up to 50% faster in time to market in enterprise schedules. It is also 25% more productive and gives fewer defects. They provide case studies to prove this. Additionally, according to Segue Technologies, Agile is also more user focused. Lastly, due to Agile being rapid and allowing for rapid prototyping, any changes can be rapidly absorbed. Any changes to requirements can be more easily handled and dealt with. With the rapidly changing requirements of the project due to the stakeholders, and the team constantly finding limitations of the Waterfall methodology, Agile was the best methodology to use for this part of the project. A more technical breakdown of how the application is designed and built is provided in the As Built Design Compliance Analysis section.

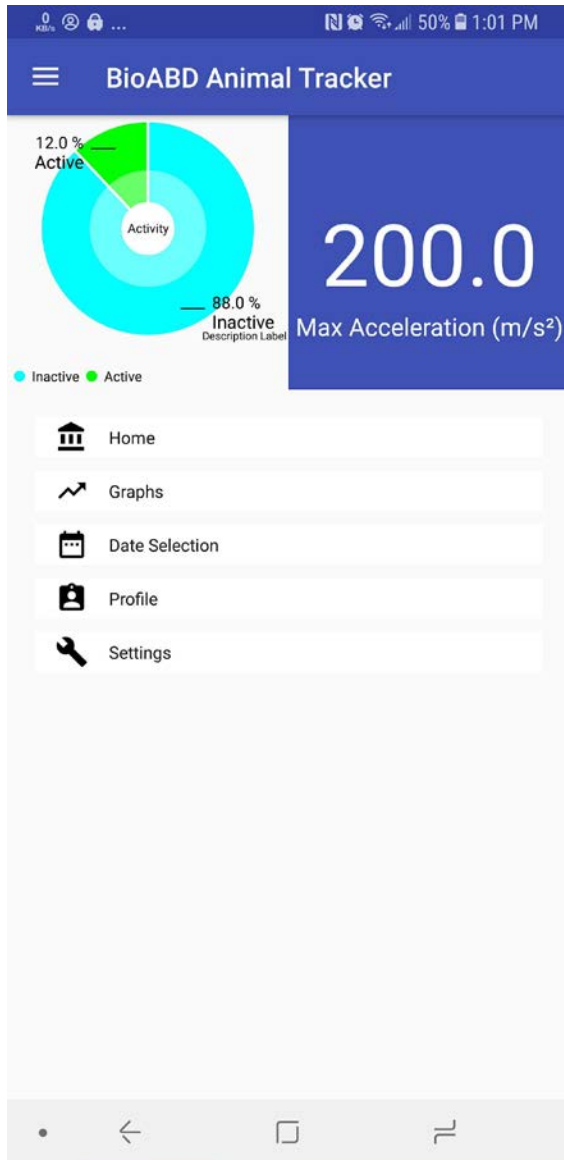


Figure 10 Dashboard of the application

More images of the application is available on the [Appendix 1: Application Screenshots](#) section.

5.3.4 Enclosure

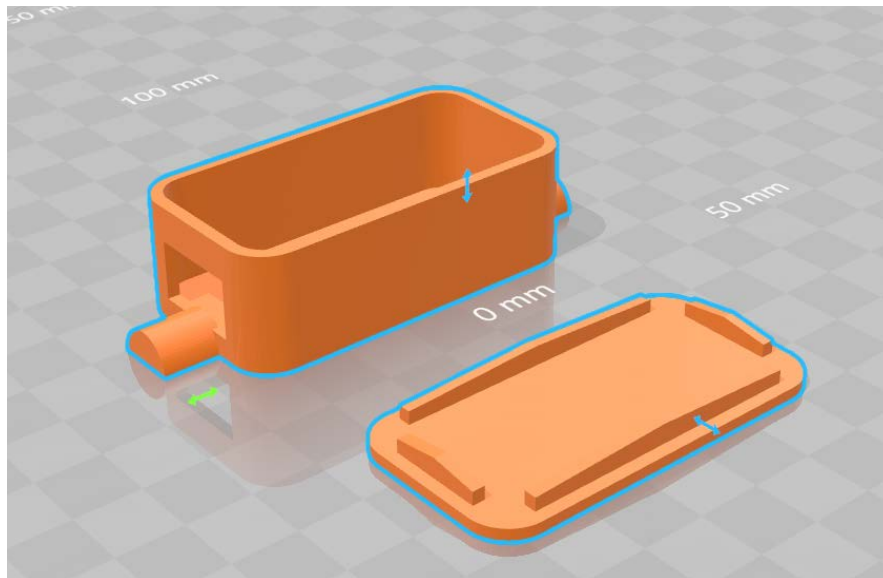


Figure 11 3D CAD file of the enclosure (Angle 1)

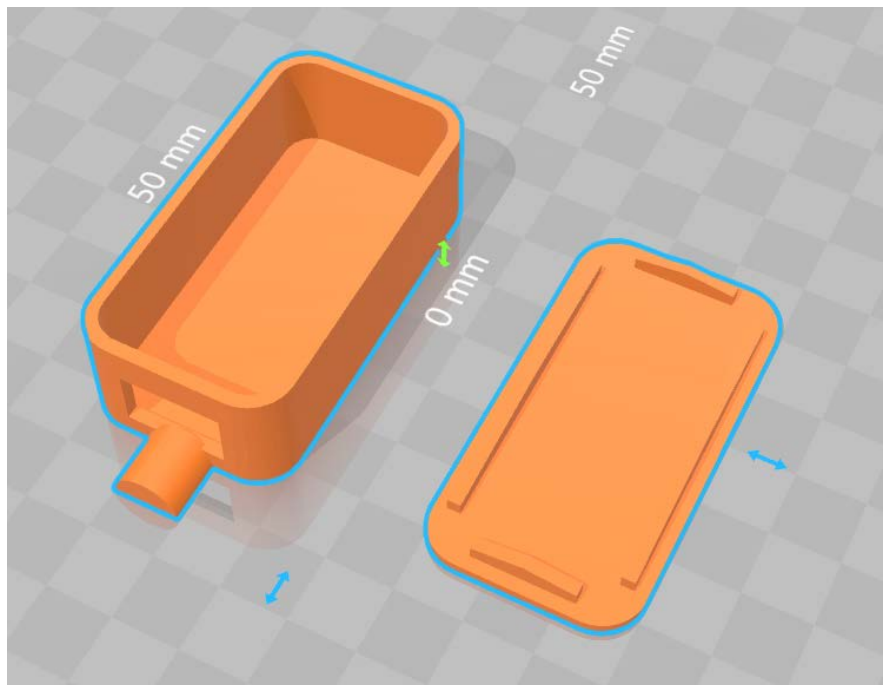


Figure 12 3D CAD file of the enclosure (Angle 2)

The enclosure was made using the TinkerCAD 3D CAD software. The dimensions for the microcontroller, accelerometer, battery, and wiring were measured. With the measurements, the 3D model for the enclosure (casing) was made. A small hole in the enclosure was made to let the temperature sensor wire out of the enclosure. The enclosure opens with a lid, where the lid requires a pry tool to be separated from the body. This ensures that the enclosure does not

open by accident. There are two small notches on both end of the enclosure and this is to mount it to the harness. The material is made from ABS (Acrylonitrile Butadiene Styrene) plastic, which is non-conductive, allowing electrical components to be mounted without any shorting or shocking concerns. The material is also durable making it suitable for the BioABD's application.

5.4 As-Built Design Compliance Analysis

5.4.1 Remark About Non-Conformance Report

Non-Conformance Report (NCR) was not used as all components complied with its stated specification.

NCR template was created but never used. The following shows the NCR template:

Part Number:	NCR Number:
Part Description:	Quantity:
Location:	Vendor:
Non-Conformity:	
Repair Request Description:	
NCR Author:	
Signature:	
Date:	

5.4.2 Embedded System

5.4.2.1 As-Built Design Usage Restriction

Component	Temperature (C)	Max Voltage Output (V)	Max Current Output (A)	Stated safety risks / warnings	Testing and Verification
ESP-32 ¹	-40 to 125	5	~0.3	-	Performance tested with battery plugged in
ADXL345	-40 to 85	-	-	-	Manufacturer specified.
LP-803860 3.7V 2000mAh Battery	-20 to 60	3.7	0.2+	Do not expose to, dispose of the battery in fire. Do not put the battery in a charger or equipment with wrong terminals connected. Avoid shorting the battery. Avoid excessive physical shock or vibration. Do not disassemble or deform the battery. Do not immerse in water. Do not use the battery mixed with other different make, type, or model batteries. Keep out of the reach of children. Battery must be charged in appropriate charger only. Never use a modified or damaged charger. Do not leave battery in charge over 24 hours. Store the battery in a cool, dry and well-ventilated area.	Specified in the Battery section in the next page.

5.4.2.2 Battery

5.4.2.2.1 Compliance Analysis Chart

No	Item	Performance	Remark	Testing and Verification
1	Capacity	2000mAh	Standard discharge after standard charge	While charging the battery from 0V to 3.7V by using 1.2A current, it takes 1 hour and 40 minutes to charge, so total capacity is $1.2A \times 1.667hr = 2000mAh$
2	Nominal voltage	3.7V	Mean operation voltage during standard discharge after standard charge	The output voltage between positive and negative voltage is 3.7V
3	Voltage at end of discharge	3.0V	Discharge cut-off voltage	The voltage is measured when battery is fully discharged in the discharging test.
4	Charging voltage	4.2V		As per specified
5	Standard charge	Constant current: 0.4A Constant voltage: 4.2V		Manufacturer's standard
6	Quick charge	Constant current: 1.2A Constant voltage: 4.2V		
9	Operation temperature range	Charge: 0~45C	60±25%R.H	Decrease/ Increase the battery temperature using high power fan/ high power heat gun.
		Discharge:- 20~60C		

5.4.2.2.2 Verification Status

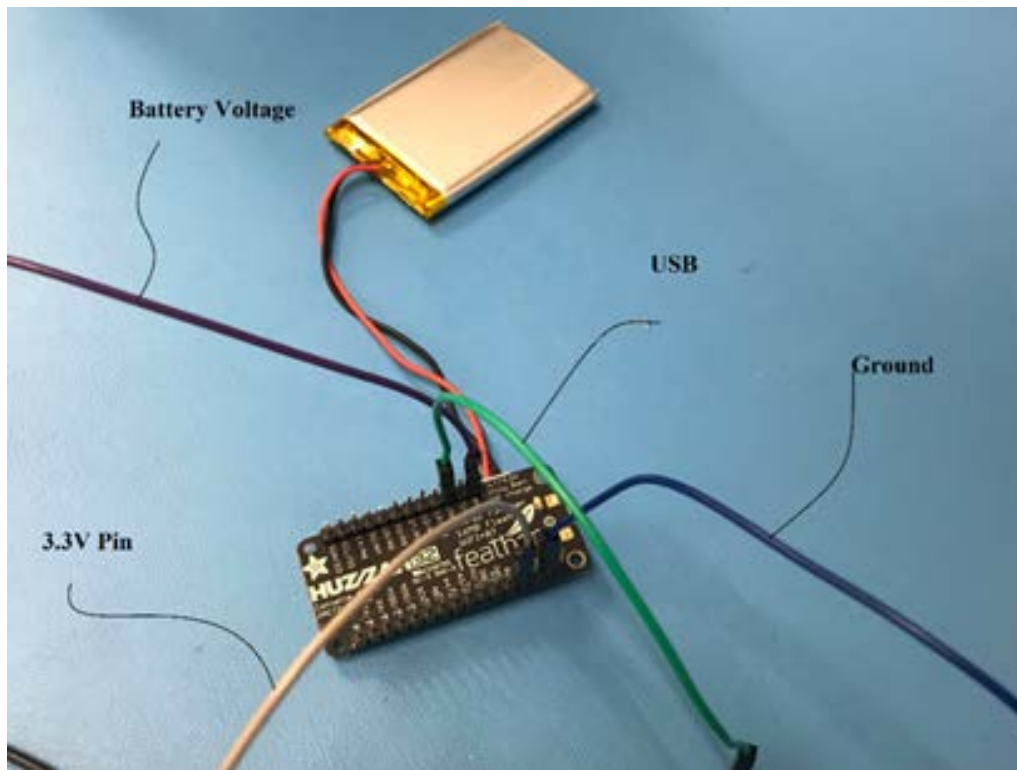


Figure 13 Verification setup



Figure 14 Oscilloscope output



Figure 15 Verification of battery under low temperature environment

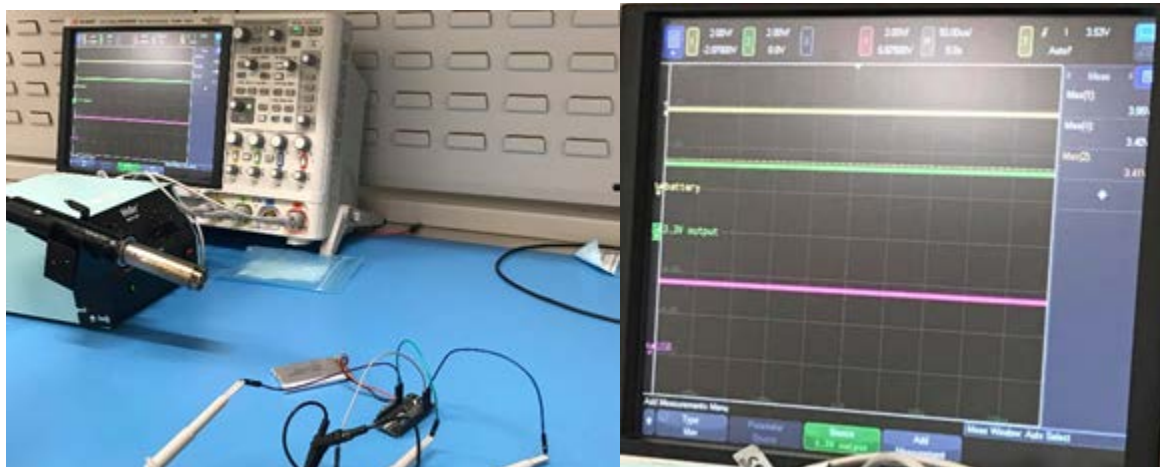


Figure 16 Verification of battery under high temperature environment

5.4.2.3 Micro-controller

5.4.2.3.1 Compliance Analysis Chart

No	Item	Performance	Remark	Testing and Verification
1	Computational Resource	80MHz	With 3.3V logic/power	As per specified for internal Chip Computation speed and turn on voltage.
2	Input Power	1. 5.0V via USB 2. 3.7V via battery		Tested in the battery section in the previous page(while battery is plugged in)
3	Dimensions	2.0" x 0.9" x 0.28" (51mm x 23mm x 8mm)	Without headers soldered in	Measured using ruler while designing the 3D printed case
4	Mass	9.7 grams		As per specified and measured by using balance
5	Memory	4MB of FLASH (32 MBit)		Manufacturer's spec sheet
6	Output Power	1. 3.3V 2. 3.7V 3. 5.0V	3.3V regulator with 500mA peak current output. 3.7V directly off the battery. 5.0V directly off the USB.	Tested in the battery section in the previous page
9	Communications Link (Protocols)	1. Built in WiFi 802.11 b/g/n 2. GPIO pins	GPIO pins - can also be used as I2C and SPI	Manufacturer's spec sheet

POWER AND FILTERING

VBUS, VBAT, 3.3V, GND, AP2112-3.3, JSTPH, X1

RESET

3.3V, GND, R3, RESET, SW2

USB TO SERIAL CONVERTER

IC16\$1, CP2104, USB/UART BRIDGE, VIO, #RST, VDD, #SUSPEND, REGIN, SUSPEND, 6PI00/TXLED, GND, 6PI01/RXLED, 6PI02, 6PI03, R1, D+, D-, D-, TXD, RXD, RTS, CTS, VPP, NC, VIO, 1.8-VDD, Op. Temp: -40°-85°C, CP2104, IC16\$2, THERMAL PAD, GND

ESP8266 MODULE + AUTORESET

ESP8266, RESET, TXD, RXD, CH_PD, GPIO16, GPIO14/SCK, GPIO12/MISO, GPIO13/MOSI, UCC, CS, DO, DI, CLK, GPIO9, GND, TXD, RXD, GPIO5, GPIO4, GPIO0, GPIO2, GPIO15, GND, TXD, RXD, GPIO5/SCL, GPIO4/SDA, RTS, DTR, nmbt2222, R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, USB_D+, USB_D-

LIPO CHARGING

VBUS, VBAT, 3.3V, GND, MCP73831/2, LIPO Charger, CHG, STAT, VDD, VBAT, PROG, USS, VDDC, 3.75-6V, Temp: -40-85°C, MCP73831T-2NCL/OT, S1, R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15, R16, R17, R18, R19, R20, R21, R22, R23, R24, R25, R26, R27, R28, R29, R30, R31, R32, R33, R34, R35, R36, R37, R38, R39, R40, R41, R42, R43, R44, R45, R46, R47, R48, R49, R50, R51, R52, R53, R54, R55, R56, R57, R58, R59, R60, R61, R62, R63, R64, R65, R66, R67, R68, R69, R70, R71, R72, R73, R74, R75, R76, R77, R78, R79, R80, R81, R82, R83, R84, R85, R86, R87, R88, R89, R90, R91, R92, R93, R94, R95, R96, R97, R98, R99, R100, R101, R102, R103, R104, R105, R106, R107, R108, R109, R110, R111, R112, R113, R114, R115, R116, R117, R118, R119, R120, R121, R122, R123, R124, R125, R126, R127, R128, R129, R130, R131, R132, R133, R134, R135, R136, R137, R138, R139, R140, R141, R142, R143, R144, R145, R146, R147, R148, R149, R150, R151, R152, R153, R154, R155, R156, R157, R158, R159, R160, R161, R162, R163, R164, R165, R166, R167, R168, R169, R170, R171, R172, R173, R174, R175, R176, R177, R178, R179, R180, R181, R182, R183, R184, R185, R186, R187, R188, R189, R190, R191, R192, R193, R194, R195, R196, R197, R198, R199, R200, R201, R202, R203, R204, R205, R206, R207, R208, R209, R210, R211, R212, R213, R214, R215, R216, R217, R218, R219, R220, R221, R222, R223, R224, R225, R226, R227, R228, R229, R230, R231, R232, R233, R234, R235, R236, R237, R238, R239, R240, R241, R242, R243, R244, R245, R246, R247, R248, R249, R250, R251, R252, R253, R254, R255, R256, R257, R258, R259, R260, R261, R262, R263, R264, R265, R266, R267, R268, R269, R270, R271, R272, R273, R274, R275, R276, R277, R278, R279, R280, R281, R282, R283, R284, R285, R286, R287, R288, R289, R290, R291, R292, R293, R294, R295, R296, R297, R298, R299, R300, R301, R302, R303, R304, R305, R306, R307, R308, R309, R310, R311, R312, R313, R314, R315, R316, R317, R318, R319, R320, R321, R322, R323, R324, R325, R326, R327, R328, R329, R330, R331, R332, R333, R334, R335, R336, R337, R338, R339, R340, R341, R342, R343, R344, R345, R346, R347, R348, R349, R350, R351, R352, R353, R354, R355, R356, R357, R358, R359, R360, R361, R362, R363, R364, R365, R366, R367, R368, R369, R370, R371, R372, R373, R374, R375, R376, R377, R378, R379, R380, R381, R382, R383, R384, R385, R386, R387, R388, R389, R390, R391, R392, R393, R394, R395, R396, R397, R398, R399, R400, R401, R402, R403, R404, R405, R406, R407, R408, R409, R410, R411, R412, R413, R414, R415, R416, R417, R418, R419, R420, R421, R422, R423, R424, R425, R426, R427, R428, R429, R430, R431, R432, R433, R434, R435, R436, R437, R438, R439, R440, R441, R442, R443, R444, R445, R446, R447, R448, R449, R450, R451, R452, R453, R454, R455, R456, R457, R458, R459, R460, R461, R462, R463, R464, R465, R466, R467, R468, R469, R470, R471, R472, R473, R474, R475, R476, R477, R478, R479, R480, R481, R482, R483, R484, R485, R486, R487, R488, R489, R490, R491, R492, R493, R494, R495, R496, R497, R498, R499, R500, R501, R502, R503, R504, R505, R506, R507, R508, R509, R510, R511, R512, R513, R514, R515, R516, R517, R518, R519, R520, R521, R522, R523, R524, R525, R526, R527, R528, R529, R530, R531, R532, R533, R534, R535, R536, R537, R538, R539, R540, R541, R542, R543, R544, R545, R546, R547, R548, R549, R550, R551, R552, R553, R554, R555, R556, R557, R558, R559, R560, R561, R562, R563, R564, R565, R566, R567, R568, R569, R570, R571, R572, R573, R574, R575, R576, R577, R578, R579, R580, R581, R582, R583, R584, R585, R586, R587, R588, R589, R590, R591, R592, R593, R594, R595, R596, R597, R598, R599, R600, R601, R602, R603, R604, R605, R606, R607, R608, R609, R610, R611, R612, R613, R614, R615, R616, R617, R618, R619, R620, R621, R622, R623, R624, R625, R626, R627, R628, R629, R630, R631, R632, R633, R634, R635, R636, R637, R638, R639, R640, R641, R642, R643, R644, R645, R646, R647, R648, R649, R650, R651, R652, R653, R654, R655, R656, R657, R658, R659, R660, R661, R662, R663, R664, R665, R666, R667, R668, R669, R670, R671, R672, R673, R674, R675, R676, R677, R678, R679, R680, R681, R682, R683, R684, R685, R686, R687, R688, R689, R690, R691, R692, R693, R694, R695, R696, R697, R698, R699, R700, R701, R702, R703, R704, R705, R706, R707, R708, R709, R710, R711, R712, R713, R

43

5.4.2.3.2 Verification Status



Figure 18 Verification (From TR Video)

5.4.2.4 Accelerometer

5.4.2.4.1 Compliance Analysis Chart

No	Item	Performance	Remark	Testing and Verification
1	Acceleration	Max acceleration: 10000g Max measured acceleration: $\pm 16g$	For any axis. Unpowered and powered. g = force of gravity	Manufacturer's spec sheet
2	Input Power	-0.3 V to +3.6 V	Operational power: 3.3V	Working when plugging into 3.3V pin on the microcontroller
3	Thermal	-40°C to +105°C	For both powered and stored	Manufacturer's spec sheet
4	Mass	3.2 grams		Measured using balance
5	Communications Link	I2C and SPI		Working when plugging into GPIO pins
6	Dimension	3 mm x 5 mm x 1 mm		Measured using ruler while designing the 3D printed case

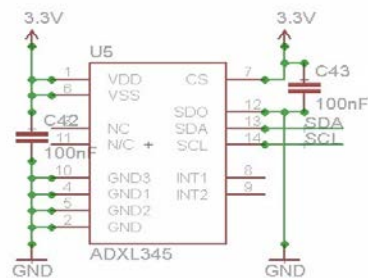


Figure 19 ADXL345 pins

Verification Status

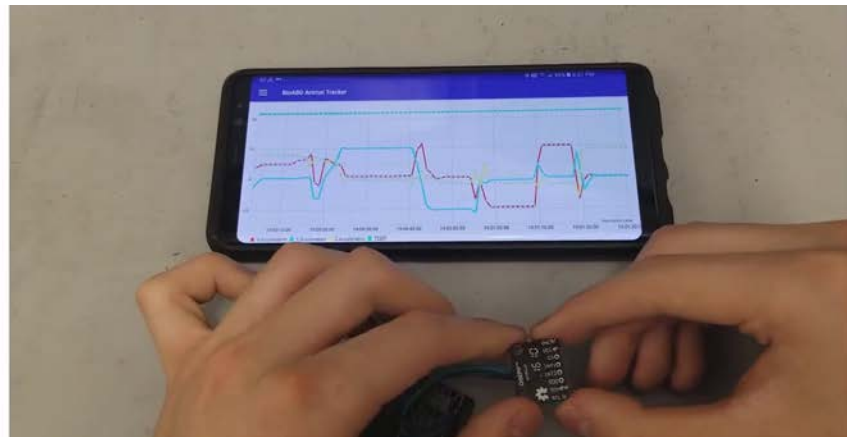


Figure 20 Verification (From TR video)

5.4.2.5 Temperature Sensor

5.4.2.5.1 Compliance Analysis Chart

No	Item	Performance	Remark	Testing and verification
1	Thermal	Measures temperatures from -55°C to +125°C	$\pm 0.5^{\circ}\text{C}$ accuracy from -10°C to +85°C	Cold water temperature, room temperature and body temperature testing.
2	Input Power	3.0V to 5.5V	Zero standby power required	Working when plugged into the 3.3 Pin of the microcontroller
3	Mass	8.8 grams	Including the data and power wires	Measured using balance
4	Communication Link	1-Wire interface		Manufacturer's spec sheet

5.4.2.5.2 Verification Status



Figure 21 Cooling down and warming up the temperature sensor

5.4.2.6 Enclosure

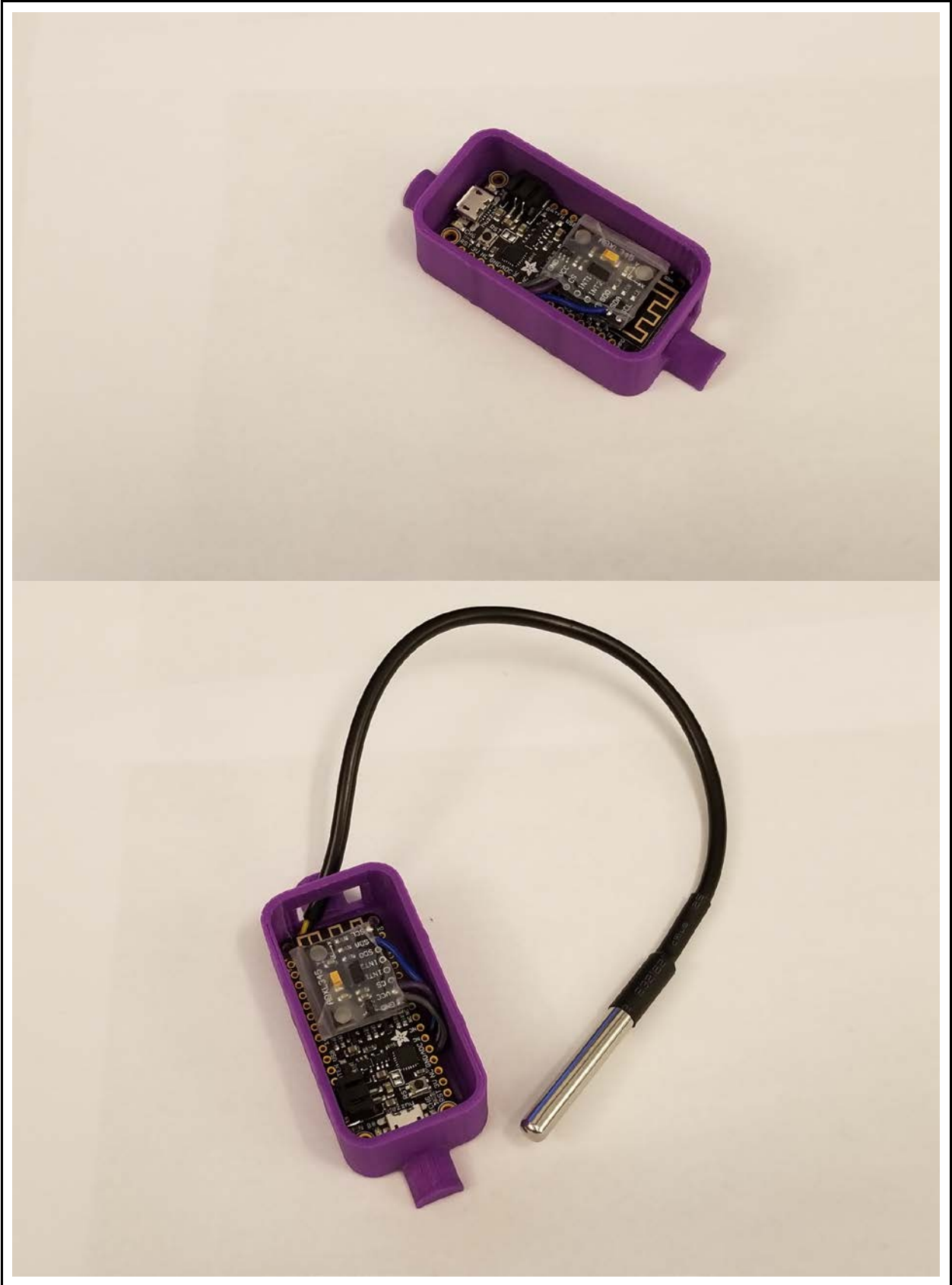
5.4.2.6.1 Compliance Analysis Chart

No	Item	Performance	Remark	Testing and verification
1	Thermal	Maximum Temperature: 80°C Minimum Temperature: -20°C Melting Point: 221°F 105°C	Source: Plastic Properties of Acrylonitrile Butadiene Styrene (ABS) Archived May 15, 2010, at the Wayback Machine. Small table of ABS properties towards the bottom.	3D printing material physical property
2	Tensile Strength of the Material	4,300 psi		
3	Mass	13.2 grams	Including the lid	Measured using balance
4	Material	ABS plastic	3D printed	3D printing service standard

5.4.2.6.2 Verification Status

The figures below shows how the system is packaged and verifies that the enclosure meets the package requirements.





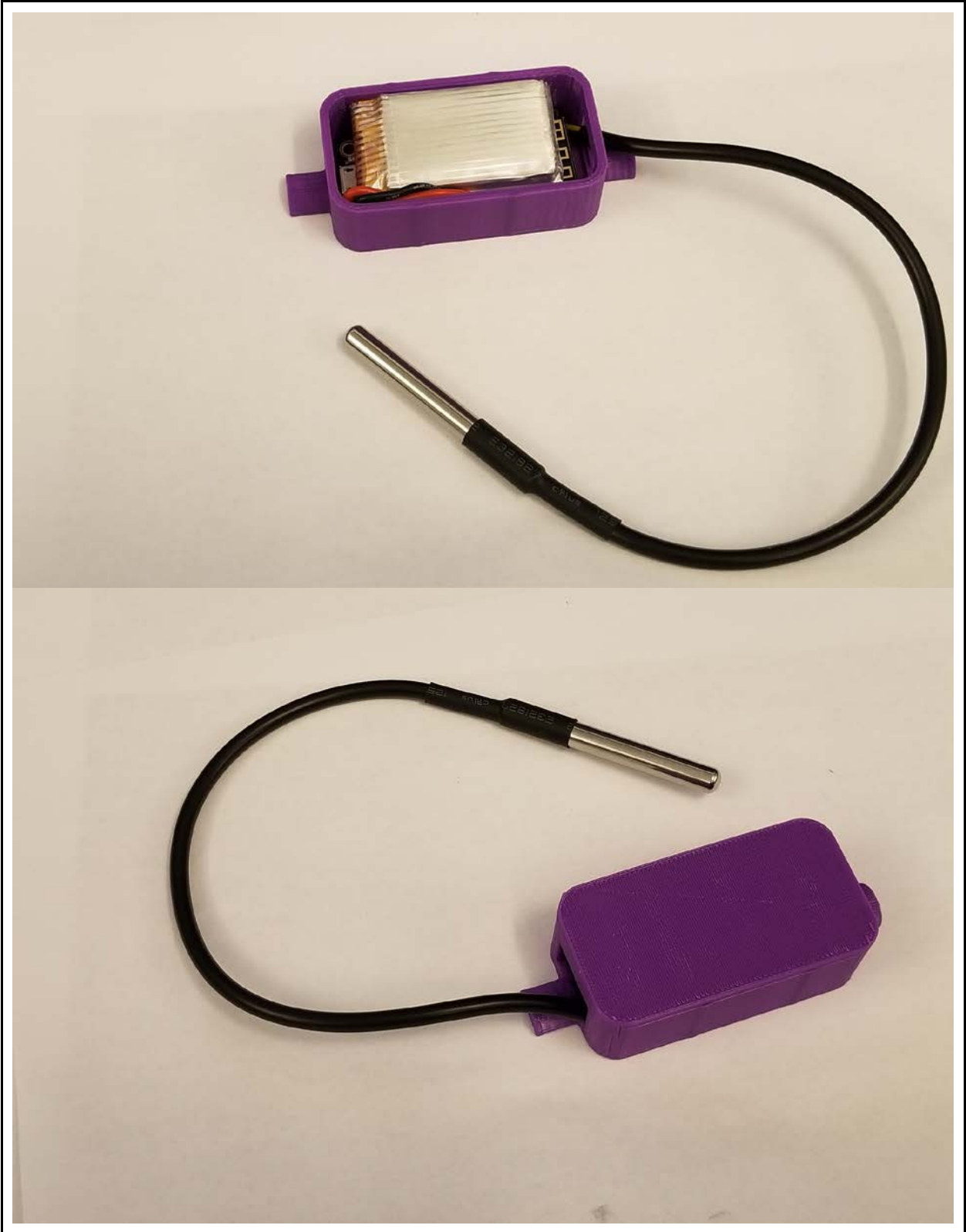






Figure 22 System with harness

The following figure shows the harness use case.



Figure 23 The harness use cases

5.4.3 Application

5.4.3.1 MVP Design Pattern

The BioABD Application is written in a relatively unique way. It uses a design pattern intended to simply how different screens are handled. The design pattern used is the Model View Presenter (MVP) design pattern (variant of Model View Controller or MVC pattern). The model view presenter design pattern allows for the separation of a complex application into many separate screens and components. For example, the BioABD dashboard would be an example of a screen. A dashboard has many different views/components that need. The BioABD application dashboard has a fancy header along with links to other parts on the app. The header is one component/view and any of the links are another component/view. The whole point of MVP is that it separates the logic into the “Presenter” class and allows for all logic for views to stay in that class. Each class has a model it must follow as well, this is what the presenter needs to provide the view in order for the view to display properly. For example with the fancy header, it is just a dummy view shown below.

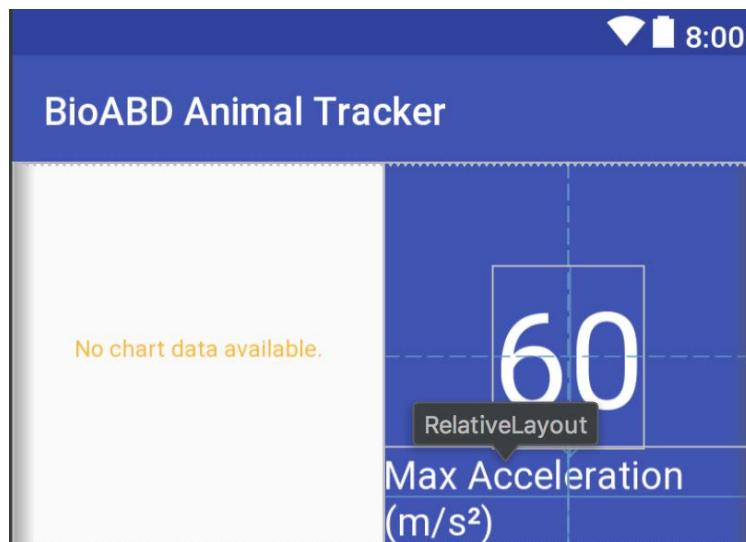


Figure 24 Header view for dashboard, the dummy view, the number 60 is a placeholder, not a real value.

The view also has a controller, or in this case, a binder. This binder binds the specific data the presenter (class with all the logic) wishes to display. The following is the constructor of the binder, showing what data the binder requires.

```
public SideBySidePieChartViewBinder(double labelNumber, @StringRes int labelResId,  
PieChartTypes type,
```



```
DataSource chartDataSource) {  
    this.type = type;  
    this.chartDataSource = chartDataSource;  
    this.labelNumber = labelNumber;  
    this.labelResId = labelResId;  
}
```

Figure 7 - Information required by the binder to bind to the view.

To add this to a page on the app is a procedure that will be explained in detail further, but a class called the Adapter is where the views are added to the screen. The following is how the header is added to the dashboard.

```
add(new SideBySidePieChartViewBinder(presenter.getMaxAcceleration(), R.string.dash_header_meter_label,  
    PieChartTypes.DASHBOARD_SMALL, new SideBySidePieChartViewBinder.DataSource() {  
        @Override  
        public HashMap<String, Float> getEntries() {  
            return presenter.getPieChartData(presenter.isMocked());  
        }  
  
        @Override  
        public String getCenterText() {  
            return context.getString(R.string.dash_header_chart_center);  
        }  
    }  
));
```

Figure 8 - How the header is added to the recyclerview, through the adapter and using a datasource interface class.

According to a Brainvire article, the MVP/MPC pattern allows for faster development, ability to have multiple views interacting separately on the same screen, and ability to change one view without affecting the whole screen. This can be shown by us modifying the binder for the top header for the BioABD Dashboard without changing the rest of the screen, as was shown above.

5.4.3.2 Recyclerviews with Different Types of Views and Abstractions

Normally, most applications have hard-coded XMLs for their screens. For example, they will have many layers of hard coded views and have View.GONE and View.VISIBLE to show and hide the views they want to show on a screen. This can easily get messy and can cause one view to conflict with another. MVP combined with the powerful property of Recyclerviews can

get around this issue. An example of normal XML coding is shown in the figure taken from the official google developers website.

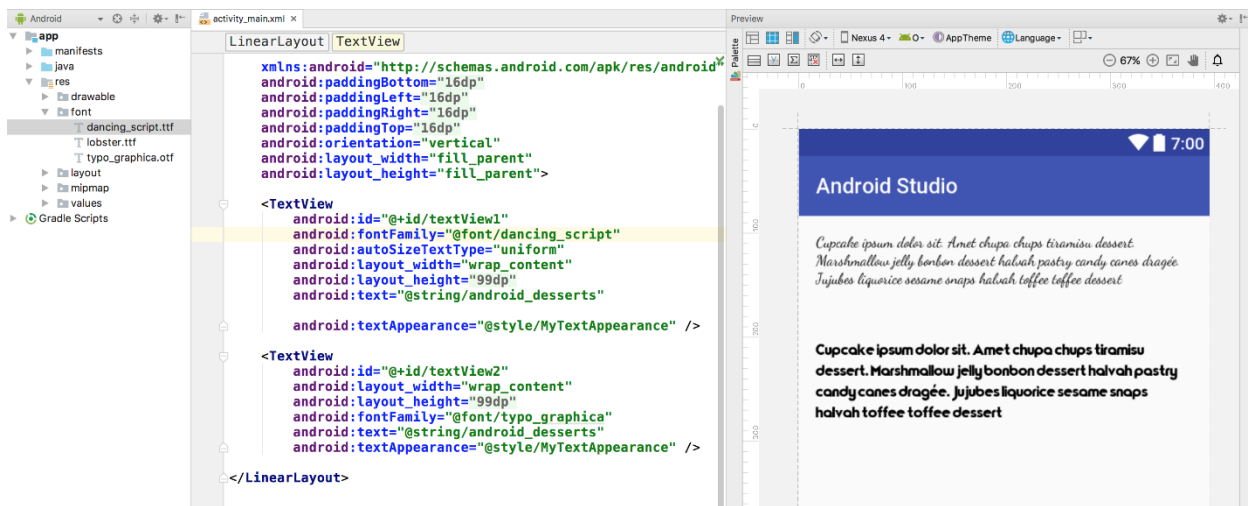


Figure 25 Hard Coded XML showing two fonts taken from the official developers.android.com website

Due to the exponentially increasing complexity of this method as pages get more complex and the limited capability of XML, a framework was developed that allowed for easy managing of views in a manner that suits MVP perfectly. Firstly, the concept of a recycler view must be understood. A recyclerview is simply a list on which views can be added.

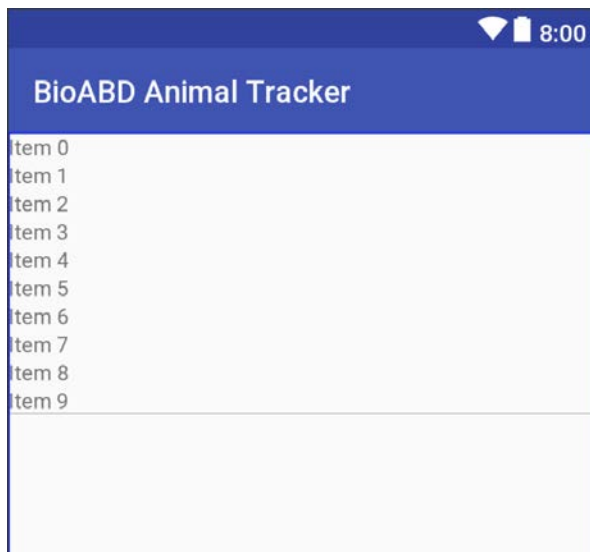


Figure 26 Photo of RecyclerView. This is how every screen in the BioABD Application actually looks.

A recyclerview efficiently manages memory and only renders what is on screen to save resources. However, a RecyclerView normally only displays one type of view (so only graphs or only headers). This is where the unique nature of what BioABD comes into play. Using the

power of abstract classes (classes with partial implementations that must be subclassed using inheritance) and generics (programming with types that will be specified during runtime), many different types of views can be added on to one page. This means the BioABD dashboard can have the fancy header and links to different parts of the app all inside one `recyclerview`. This means one presenter for the whole page also, so all logic is contained and is independent of other parts of the app and all the individual components are independent of all other components on the dashboard. `Recyclerviews` require an adapter (class that adds the different views to the `recyclerview`) and a `viewholder` (class that holds the view with binded data). Both of these can be abstracted and generics can be used such that the view itself can be passed into that generic and thus, the `recyclerview` becomes independent of the view type itself. Please refer to `AbstractDataBinder` and `AbstractDataBindAdapter` classes for this implementation.

Many different levels of abstractions were made. Since all the screens were `recyclerviews`, an abstract activity class called `AbstractDataActivity` was created. This made the act of creating a new page extremely simple. To create a new page in the app, you would be forced to follow MVP. You would need a presenter (the logic class) and the adapter (the class that put views into the `recyclerview`).

```
/**
 * This is the activity for the main Graphing activity.
 */
public class GraphsActivity extends AbstractDataActivity implements GraphingView {
    private GenericLineChartAnnouncer<GraphPoint> announcer = new
GenericLineChartAnnouncer<>();
    private GraphingPresenter presenter;
    private GraphingAdapter adapter;

    @Override
    public AbstractDataBindAdapter getAdapter() {
        adapter = new GraphingAdapter(this, presenter);
        return adapter;
    }

    @Override
    public AbstractPresenter getPresenter() {
        presenter = new GraphingPresenter(this, announcer);
        return presenter;
    }

    @Override
    public void reloadPage() {
        adapter.reload();
    }
}
```

Figure 27 Creating a new page in the apps

To create a new page, just provide the presenter and adapter in an overloaded method and have a method to reload the page when the data changes significantly (although this is rarely used since it reloads the whole page).

5.4.3.3 Real-Time Graph

The real-time graph implemented on the application is done similarly to how the dashboard is implemented however it has a few key differences. For the real-time graph, when the activity is

started, the presenter is instantiated. During this instantiation, there is a listener added to the Firebase database. This listener does exactly what it sounds like, it reaches to the server and subscribes itself to one specific node. The micro-controller sends data to the node “adx1345” so whenever there is any data added to that node, a custom callback function (described shortly) gets called and my presenter gets notified of this change, as well as the new point being passed as a JSON object. Appendix 2 shows how the object is parsed into a usable format. Below the contract (or model) the micro-controller was sending to the server and what the application was receiving.

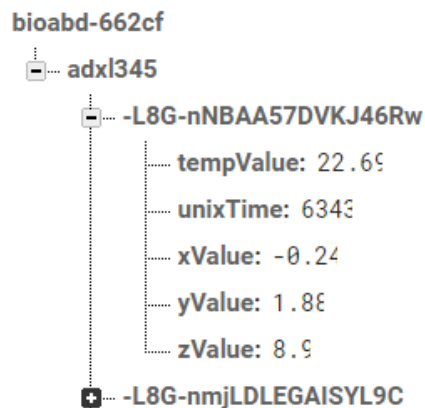


Figure 28 Contract shared between the micro-controller, server, and application.

```
dataSnapshot = (DataSnapshot@6555) "DataSnapshot { key = -L8G-nNBAA57DVKJ46Rw, value = {tempValue=22.69, zValue=8.9, unixTime=6343, xValue=-0.24, yValue=1.88} }"
```

Figure 29 Raw string received from Firebase.

```
point = {GraphPoint@6564}
  tempValue = {Float@6566} 22.69
  unixTime = {Long@6567} 6343
  xValue = {Float@6568} -0.24
  yValue = {Float@6569} 1.88
  zValue = {Float@6570} 8.9
  shadow$_klass_ = {Class@4693} "class com.bioabd.ngta.graphs.models.GraphPoint"... Navigate
  shadow$_monitor_ = -1917952187
```

Figure 30 Parsed variable of raw JSON string from Firebase.

Once this is parsed, certain checks happen to ensure the parsed object is valid and will not crash the application.

The graph API being used is MP Android Chart. It has built-in capabilities for real-time functionality. The issue however is that the built in reloading capabilities of a recyclerview will not reload just one portion of the graph, but will reload the whole page. This is not nearly as

efficient with bigger datasets or if there are multiple views on the page, it is not effective with those either. To deal with this, a class was added in the binder to allow for the graph native real-time data. However, how would one notify the graph to update for a new graph point every time the presenter had a new point to graph without reloading the page? To accomplish this, an interface was used. An interface is simply a method that can be implemented in one class but called from another class. When an implemented interface class is called, it goes and runs the code where it is implemented. In this case, there are two interfaces. One interface to notify the graph of the new data so it can prepare to call the graph-native refresh functions. The second takes the data contained within the graph back to the presenter and adds the new point, then gives the new data set back the graph to show to the user. Thus keeping with MVP pattern, all logic for the graph is within the presenter. The view simply relies on generic interfaces to update itself. This is explained in even further details in Appendix 3, which goes into code-specific details.

5.4.3.4 Application Structure

Now that the specifics of the foundation of the Android application are complete, it is time to talk about the structure of the application. This will talk about the different activities inside the app, how global variables are handled, how the slide-out/fly-out menu is handled, and more.

General Activity Structure:

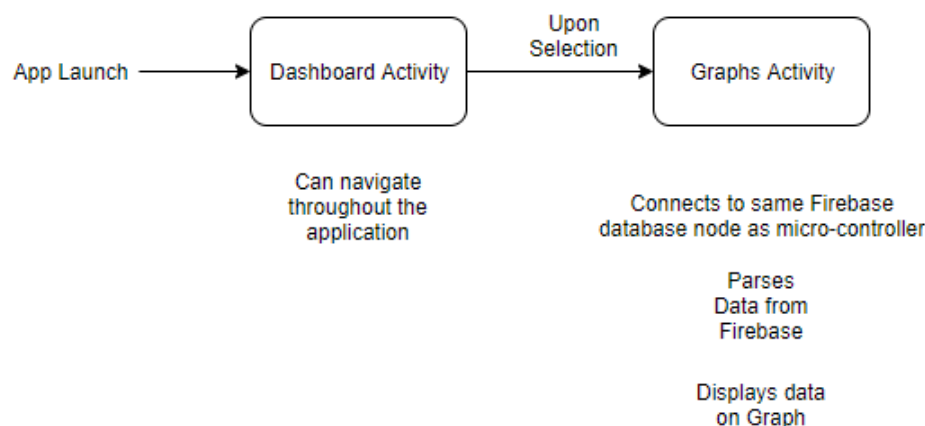


Figure 31 Diagram of the different activities and their purposes.

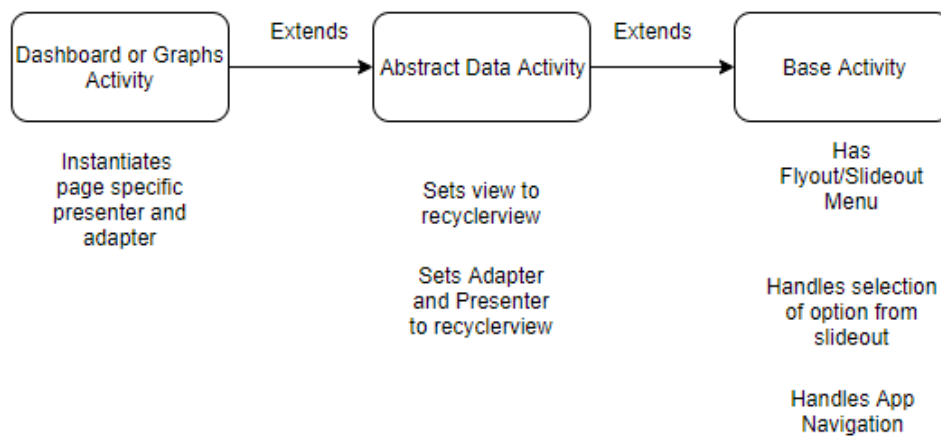


Figure 32 Diagram of inheritance of activities as well as many levels of abstractions. Note: the Extends goes from left to right, so Dashboard Activity extends Abstract Data Activity

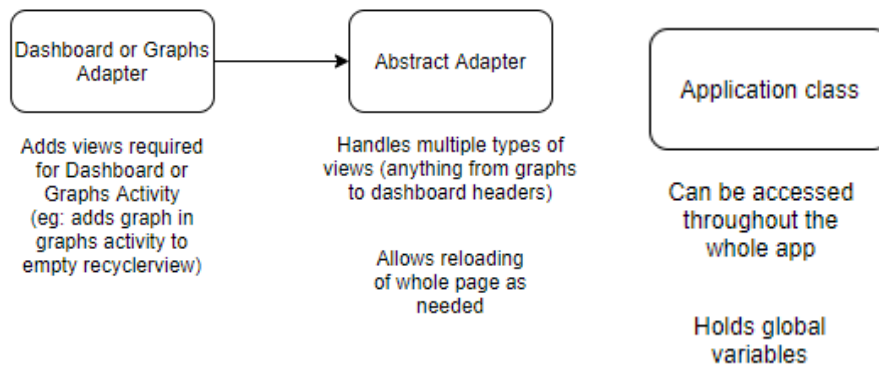


Figure 33 Diagram of Adapters

Figure 34 Diagram global variables

Please note, Appendix 5 has a more comprehensive breakdown of how global variables are stored on the BioABD application.

5.4.3.5 Ensuring Compliance (Verification Status)

To summarize, the application complies with all the functional and user requirements set out. To show this, the following chart was made.

Requirement	Test Case from TR
Establishes a connection to a specific node on Firebase.	Please refer to Test Case 1 in the Application section of TR.
Parses the data given from the Firebase.	Please refer to Test Case 2 in the Application section of TR.
Displays the latest updated data from the Firebase.	Please refer to Test Case 3/4/5 in the Application section of TR.
App does not crash easily crash	Please refer to Appendix 3 as well as Test Case 6 in the Application section of TR.

Figure 35 Table showing compliance

The application has a size of 3 MB.



Name	Date modified	Type	Size
 app-debug.apk	3/30/2018 3:36 PM	APK File	3,001 KB
 output.json	3/30/2018 3:36 PM	JSON File	1 KB

Figure 36 Application APK size

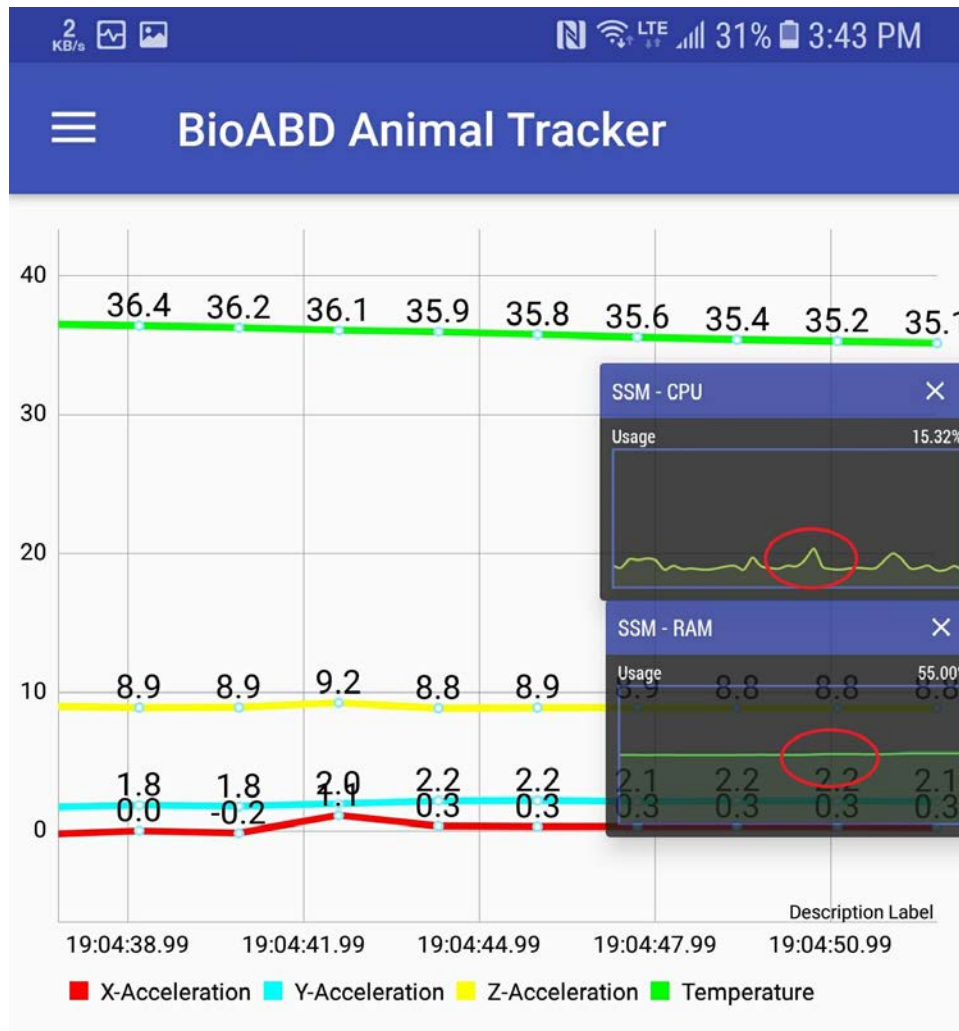


Figure 37 CPU and RAM usage for application

The system resources used when running the real-time graph can be seen above. An application called Simple System Monitor (linked in the sources section) was used to monitor this. This was ran on a Galaxy Note 8. As can be seen the increase in RAM is negligible and there is a approximately 10% spike in CPU usage when plotting the graphs. It is worth noting that while the increase in RAM usage is negligible, it will be more pronounced when there is more data stored on the server. However even then, it should not be a major issue. Additionally the CPU usage increase was also minimal.

5.4.4 Firebase Server

The firebase server had two intentions. It would act as storage for the microcontroller, and it would act as a real-time database for the application. The data is stored as a JSON and is sent as such. Firstly, the microcontroller and application agreed upon a “contract.” This contract would be the format the microcontroller would send data in and it would also be the format the application would parse. When the microcontroller would send data to Firebase, the name of the data would be auto-generated by firebase, but the values of the data would be those sent by the microcontroller. This is why in the figure below the name of the datapoint seems random (L8G-nB...) but the key/value pairs inside the object (tempValue, unixTime, xValue, yValue, zValue) are not.

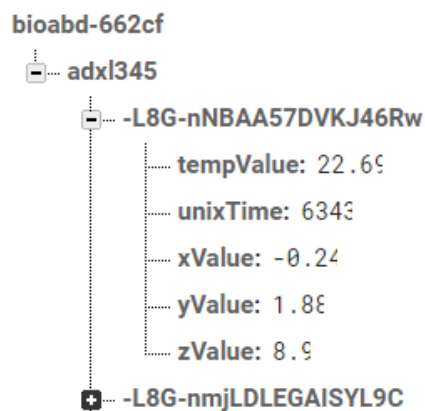


Figure 38 Contract shared between the micro-controller, server, and application.

The microcontroller would send the data to the node “adx1345” and the application would subscribe to that specific node to listen for data. As the data is added, anyone monitoring the server can see the data in real-time as can be seen by the video lined in the “System Requirements and Verification by Tests” section (also linked below).

<https://www.youtube.com/watch?v=UIK-N0HM-rw>

Please refer to Appendix 2 for the specifics of how the application parses the values received from Firebase.

Requirement	Verification
Establishes connection with the micro-controller and the application.	Please refer to test case 1, 2, and 3 from the TR from the Firebase section.
Receive formatted information from the microcontroller and store it in the same format.	Please refer to test case 1 from the Firebase section from the TR or please refer to the full system test.
Notify application when data is added.	Please refer to test case 3 from the firebase section or test case 1 from the application section.

Figure 39 Summary of testing of Firebase

As can be seen from above, the BioABD firebase database meets the requirements set out in the requirements section.

6 Section II - Financial and Management Volume: Supporting Documents

6.1 As-Built Work Breakdown Structure

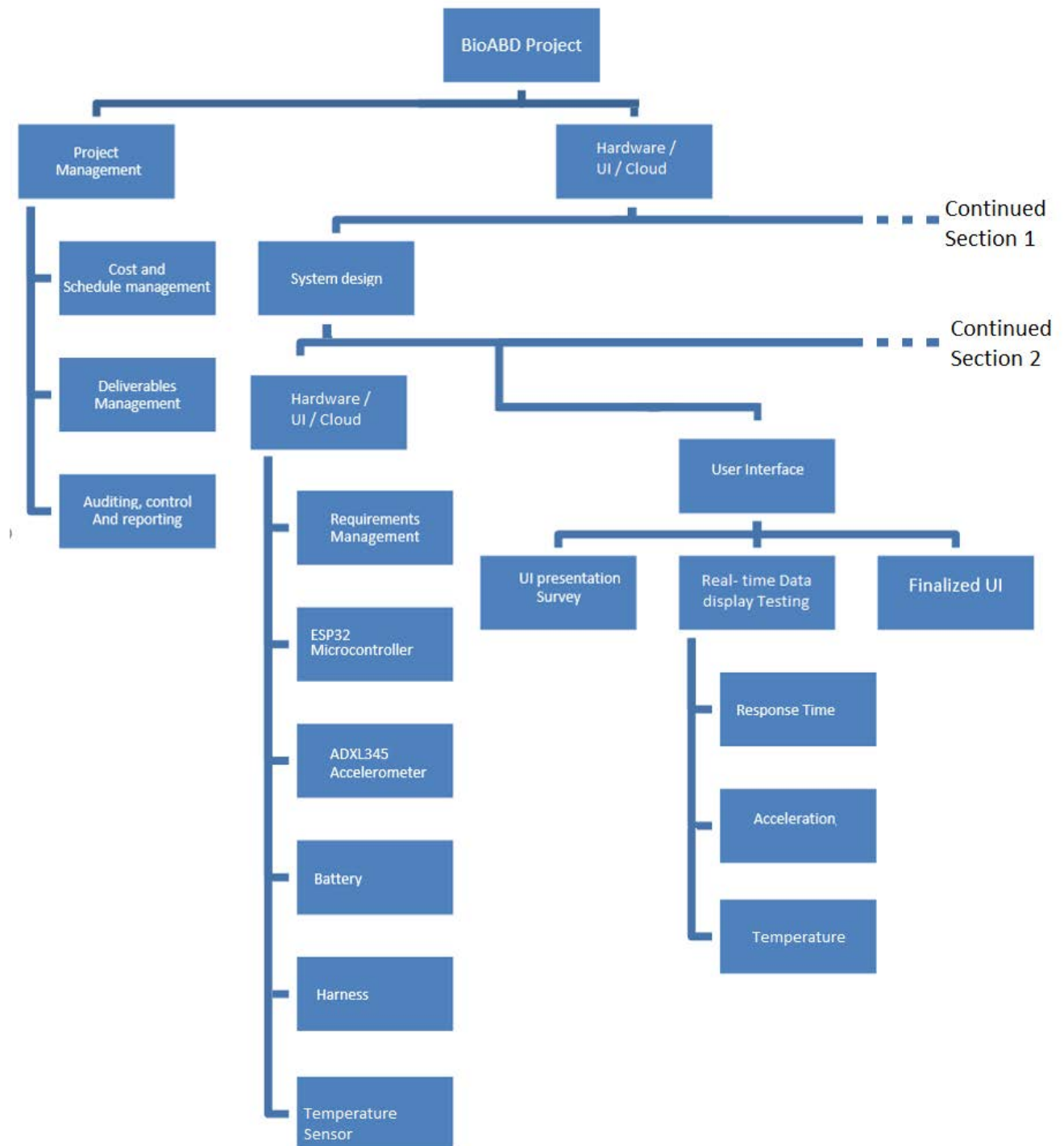


Figure 40 WBS (part 1)

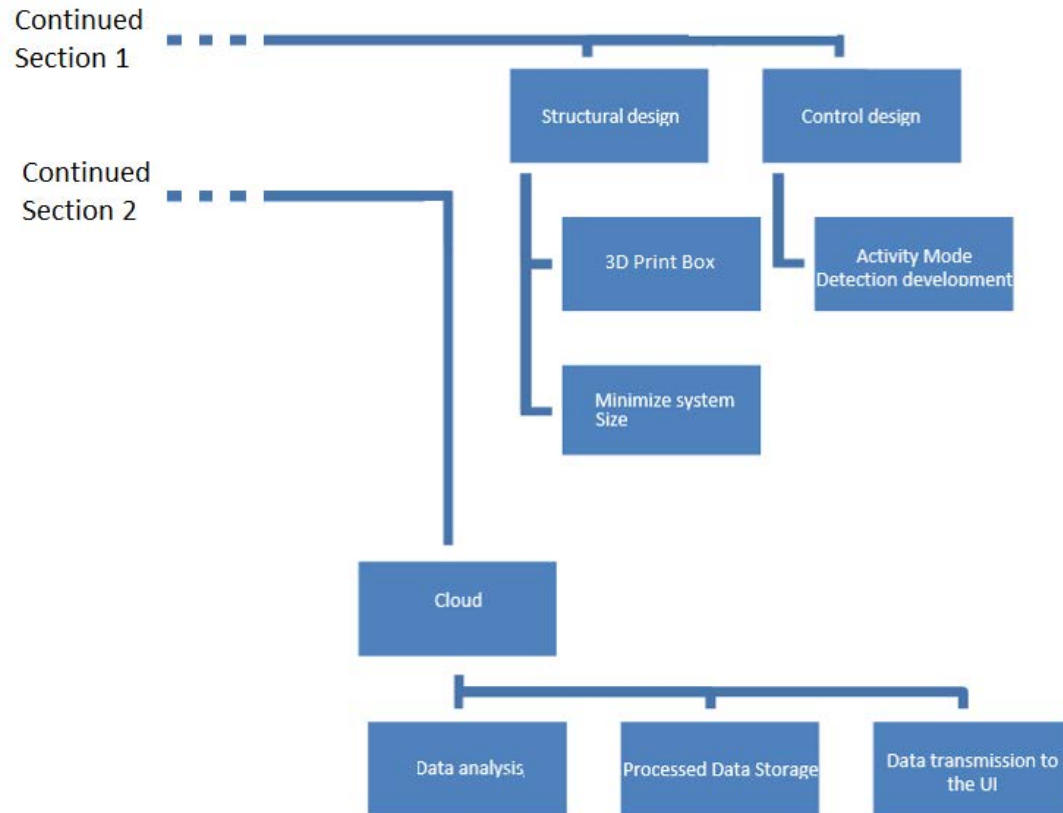


Figure 41 WBS (part 2)

6.2 As-Built Work Package Descriptions: Work Breakdown Structure

The BioABD system's WBS was created with two main components; Project management and Hardware/UI/Cloud (the product). Project management was splitted into three subsections; cost/schedule management to manage the team's budget and time restrictions, deliverable management to manage the scope of the project and changes required to the project, and finally, auditing, control and reporting.

System design for the final product was broken down into Hardware (including the hardware's UI and cloud connection), UI (the application), and cloud (the Firebase server). Each component was assigned to different team members due to each person being proficient at different component of the project.

BioABD group Work Breakdown structure and work package hours has to be change due to the changed final scope and the CUPE 3903 strike. The team removed the calculation of velocity and distance traveled using the mathematical integration of the acceleration data, and added the temperature sensor. This is not a significant change since the PDR report submission. There was and still is the CUPE 3903 strike, therefore BioABD's Test Review submission was delayed. Moreover, the Final Report work time had to reduced.

Work hours for each component in the WBS is shown in the next section: [As-Built Resource Allocation Matrix](#).

6.3 As-Built Resource Allocation Matrix

6.3.1 Originally Planned Resource Allocation Matrix

Responsible, Support Required, Must be consulted, and Must be Notified was assigned with the number of hours, where being most hours assigned means “responsible” and least amount of hours assigned means “must be notified”.

Work Package	December				January			
	MJK	RB	XKW	XC	MJK	RB	XKW	XC
Cost and schedule management	3	3	3	3	3	3	3	3
Deliverables management	3	3	3	3	3	3	3	3
Auditing, control and reporting	3	6	3	3	3	6	3	3
Requirements management	2	2	2	2	2	2	2	2
UI presentation Survey	1	1	1	1	1	1	1	1
Reeal- Time data display Testing		8				8		
Finalize UI								
Data Analysis/ Process			6	6			6	6
Data storage	6			6	6			6
Data transmission to UI		6				6		
3d Print Box			6				6	
Minimize system Size			6				6	
Activity Mode Detection development	6	6	6	6	6	6	6	6
TOTAL	24	35	36	30	24	35	36	30

Work Package	February				March				April			
	MJK	RB	XKW	XC	MJK	RB	XKW	XC	MJK	RB	XKW	XC
Cost and schedule management	3	3	3	3	3	3	3	3	3	3	3	3
Deliverables management	3	3	3	3	3	3	3	3	3	3	3	3
Auditing, control and reporting	3	6	3	3	3	6	3	3	3	6	3	3
Requirements management	2	2	2	2	2	2	2	2	2	2	2	2
UI presentation Survey												
Reeal- Time data display Testing												
Finalize UI		8										
Data Analysis/ Process			5	5								
Data storage	6			6								
Data transmission to UI		6										
3d Print Box			6									
Minimize system Size			6									
Activity Mode Detection development	6	6	6	6								
TOTAL	23	34	34	28	11	14	11	11	11	14	11	11

Figure 42 Resource allocation matrix from the PDR

6.3.2 As-Built Resource Allocation Matrix

MJK = Min Jae Kim, RB = Rishabh Bhat, XKW = Xingkai Wu, XC = Xin Chen

Work Package	December				January				February			
	MJK	RB	XKW	XC	MJK	RB	XKW	XC	MJK	RB	XKW	XC
Cost and schedule management	5	0	0	0	5	0	0	0	4	0	0	0
Auditing, control and reporting	3	4	3	2	3	2	3	3	2	3	3	4
Requirements and deliverable management and documentation	20	15	16	10	21	20	22	18	19	20	19	17
UI presentation Survey (Android app development)	5	10	0	0	3	30	0	0	5	25	0	0
Real time data display testing	11	10	0	0	10	15	0	0	8	6	0	0
Finalize UI	0	0	0	0	0	0	0	0	9	10	0	0
Data analysis	0	0	0	0	0	0	0	0	0	0	0	0
Data storage (database)	3	0	4	11	0	0	5	6	0	0	8	7
Data acuriment transmission (embedded system design)	10	0	5	2	25	0	6	3	30	0	4	2
3D enclosure design and manufacturing	0	0	0	0	0	0	0	0	8	0	5	0
Minimize system size	0	0	0	0	0	0	0	0	0	0	0	0
Total	47	44	32	30	42	70	42	34	54	68	44	32

Figure 43 As-Built Resource allocation matrix (part 1)

Work Package	March				April (planned)			
	MJK	RB	XKW	XC	MJK	RB	XKW	XC
Cost and schedule management	6	0	0	0	5	0	0	0
Auditing, control and reporting	0	0	0	0	0	0	0	0
Requirements and deliverable management and documentation	50	37	38	33	15	15	15	15
UI presentation Survey (Android app development)	6	40	0	0	0	0	0	0
Real time data display testing	5	5	0	0	0	0	0	0
Finalize UI	15	18	0	0	0	0	0	0
Data analysis	5	4	5	4	0	0	0	0
Data storage (database)	0	0	5	9	0	0	0	0
Data acuriment transmission (embedded system design)	10	0	0	0	0	0	0	0
3D enclosure design and manufacturing	6	0	12	0	0	0	0	0
Minimize system size	5	0	7	7	0	0	0	0
Booth preparation	0	0	6	7	25	25	25	40
Total	108	104	73	60	45	40	40	55

Figure 44 As-Built Resource allocation matrix

6.3.3 Summary & Changes since the PDR

Min Jae Kim is responsible for all hardware supply purchase and the development of the embedded system because of his experience with working with the ESP-8266 and ESP-32 boards, Temperature sensor, and Accelerometer in the past.

Xingkai Wu is responsible for the data analysis and system size minimization and some sensors in the embedded system due to his electrical engineering background.

Rishabh Bhat is responsible for the mobile application design and the UI implementation within the application.

Xin Chen is responsible for the data processing and cloud storage (Firebase server).

All members in BioABD will prepare for the booth presentation and bring the best results to its audiences.

The team removed the calculation of velocity and distance traveled using the mathematical integration of the acceleration data, and added the temperature sensor, due to the change of stakeholder's demand. This is not a significant change since the PDR report submission.

There was and still is the CUPE 3903 strike, therefore BioABD's Test Review submission was delayed. Work time had to be reduced.

6.4 As-Built Project Schedule

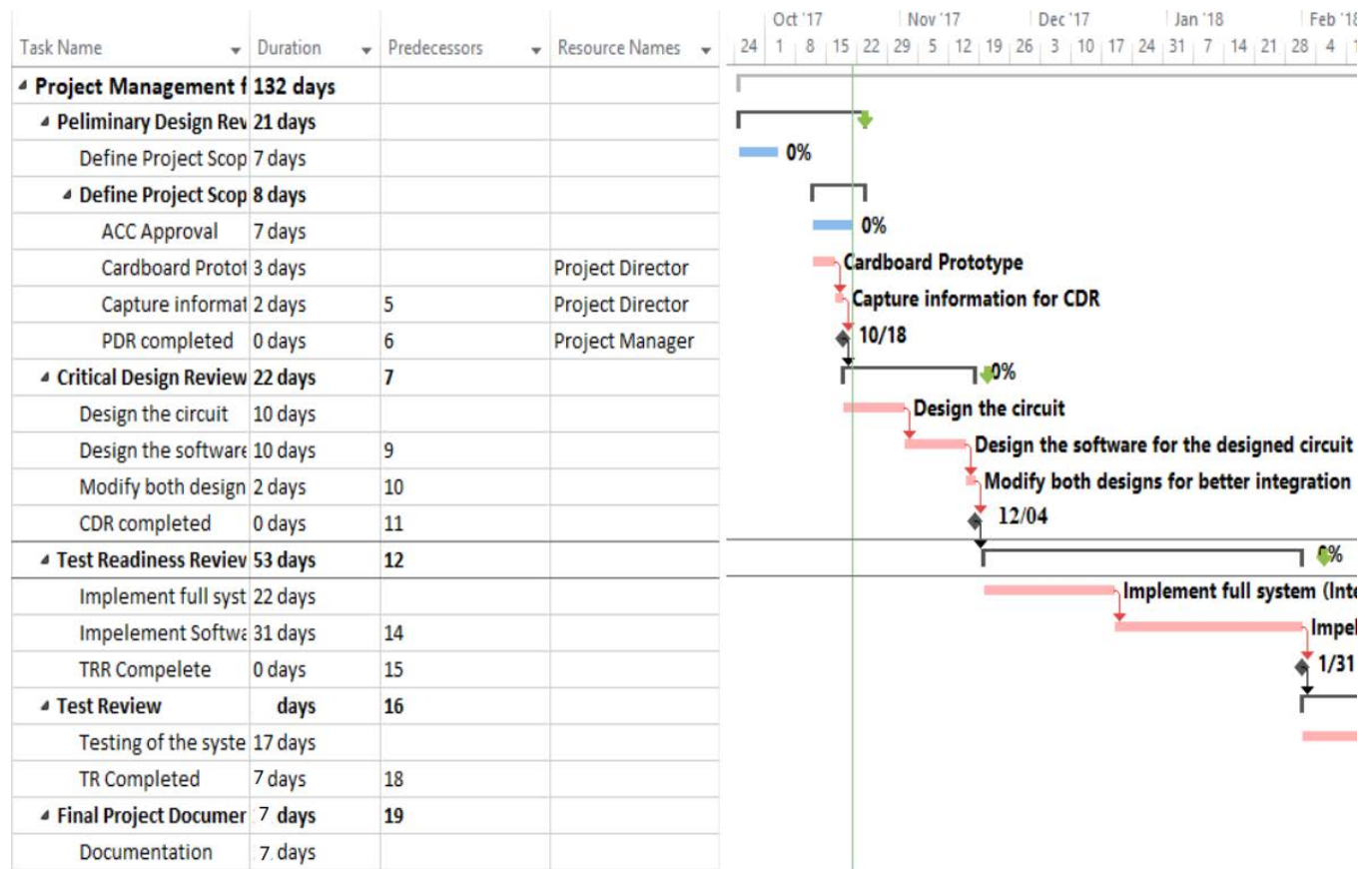
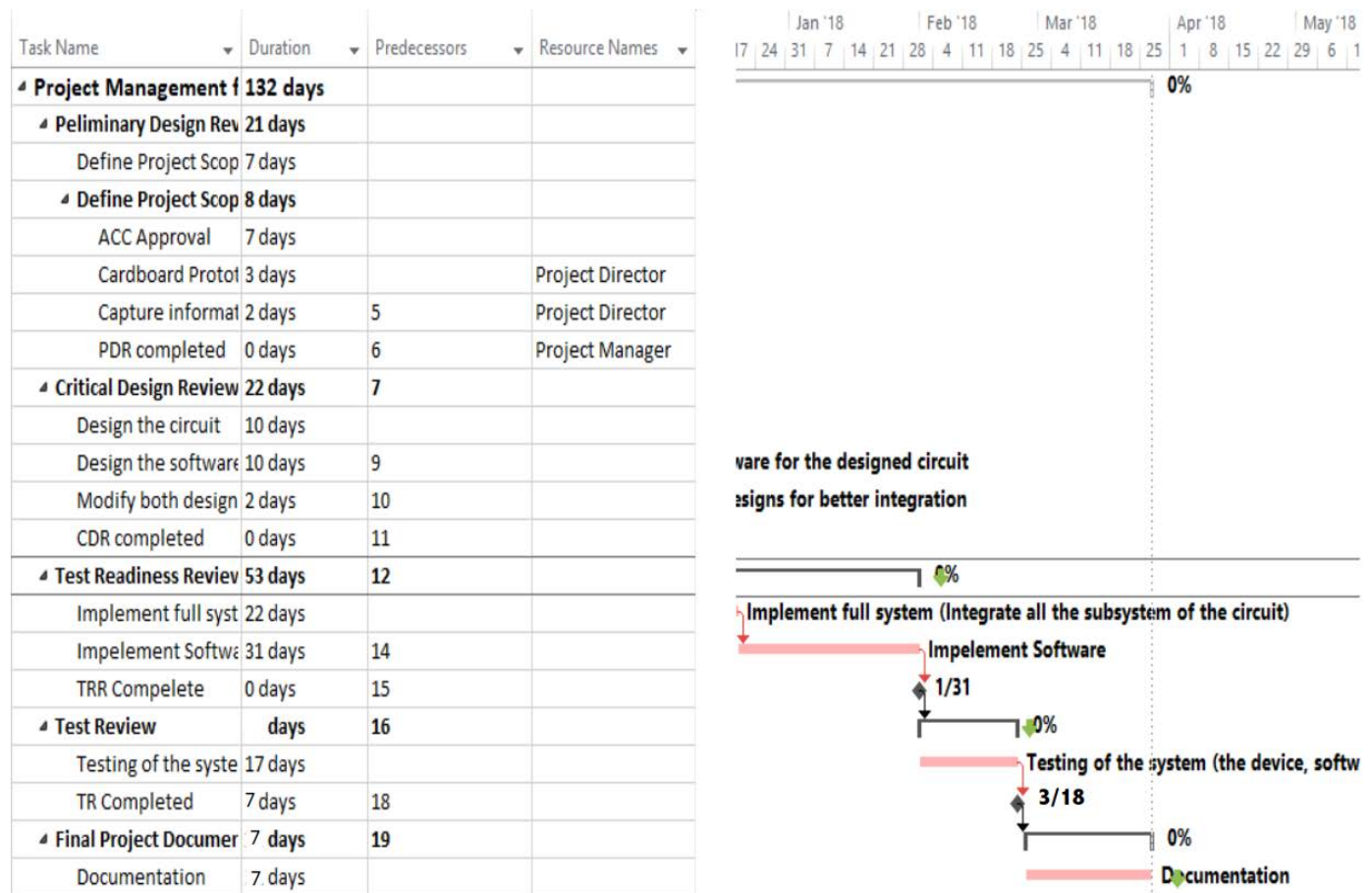


Figure 45 Project Schedule (part 1)



The project schedule has not changed much since the PDR. Some due dates were changed due to the CUPE 3903 strike and minor change in the deliverable of the project.

6.5 As-Built Project Procurement/Equipment/Travel List

Explanation		Automobile		Expenditures
Date (mm/dd/yyyy)	Description	# of KM's	Amount @ \$0.45/km	Amount (\$)
10/21/2017	Prototype Materials			13.84
10/24/2017	Microcontroller			34.31
10/24/2017	MC battery			25.85
10/24/2017	Accelerometer			33.78
11/22/2017	Dog leash			15.99
11/26/2017	On site visit	300	135.00	135.00
03/04/2018	Animal sport harness			87.61
03/06/2018	Temperature sensor			21.97
03/25/2018	Microcontroller			34.31
Total				402.66

Figure 47 Project expense list

The BioABD team will sponsor LyoFresh team (group #11), the exact amount of sponsorship has not been determined yet at this moment, but it will be in the scale of \$200 to \$300.

6.6 Preliminary Business Case

The preliminary business case for BioABD is to act as a research tool for the animal psychology research by Dr. Suzanne MacDonald of York University. They were one of the two major stakeholders for BioABD system, the other stakeholder being the professional animal trainers. Dr. Suzanne MacDonald requested a way to track the response of the animals to different stimulus. Her research involves training dogs, and she wants to research if animals are stressed in specific situations or not, and how to help them be trained with minimal stress. To deliver this, the BioABD group decided upon giving her acceleration and temperature data (with heart rate to be requested as a successor). Acceleration can help notice sudden movements that a dog may make, for example if the dog is startled, he/she may flinch similar to humans. Acceleration has the potential to be extremely useful to their research. The BioABD team also delivered a temperature sensor. This sensor detects the temperature of the dog. This is additionally important to tracking the animal as the temperature of the dog can increase or decrease depending on how stressed the dog is. This is also important as the acceleration data may differ depending on temperature. Below is a Strength, Weakness, Opportunity, Threat analysis of BioABD.

6.6.1 Strength Weakness Opportunity Threat Analysis (SWOT)

	Opportunities	Threats
Strengths	<p>Able to obtain customers in the animal research and training industry due to these strengths of the system:</p> <ul style="list-style-type: none"> • Tracking of the temperature of a an animal • Tracking of the 3-Dimensional acceleration due to movements • Real time graphical user interface • More sensors are attachable due to its modular design 	<p>Competing products which could steal the team's intellectual property, but the strengths will prevent this:</p> <ul style="list-style-type: none"> • Team and supporters will apply for legal IP protection • First to the market (First mover's advantage)
Weaknesses	<p>The heartbeat sensor was not implemented in the coded design of the embedded system due to the unavailability (back-order) of the sensor. Although this is a weakness, the team has implemented a modular design to timely add the sensor.</p> <ul style="list-style-type: none"> • The modular design will give room and opportunities for a timely implementation of other sensors in the future <ul style="list-style-type: none"> ○ Blood oxygen level sensor ○ Velocity sensor ○ Breath rate monitor 	<p>The weakness which is a threat to the BioABD system are:</p> <ul style="list-style-type: none"> • Heartbeat sensor not implemented yet • No support for multiple users on the same Firebase project. (While Firebase and the application can enable multiple users easily, this feature has yet to be implemented. Primarily due to time constraints. This means that every new user for the application requires the microcontroller code and the application code to be refactored slightly.)

Figure 48 SWOT diagram

7 Section III - Lessons Learned Volume: Supporting Document

7.1 Deviations From Plan

The scope of the project has changed from the original scope. The original BioABD system did not have the temperature sensor or planned heart rate monitor. Originally, the acceleration data was to be converted into velocity and distance values using integration calculation. The integration calculation was planned to be done on the server. During the meeting between the team and the team's stakeholders in the late February (animal behavioural researchers/ professors), the stakeholder noted that they wanted to change the scope of the project as the velocity and distance data was no longer required. Thus, the specification for the BioABD system was changed. Firstly, the stakeholders (animal psychology researchers) of the project did not need this information after much consideration, so the time it would take to finish implementation of this and test it thoroughly would be better spent on other portions of BioABD system. Secondly, there is a study by a company in Australia who has attempted to create a system which is similar to the BioABD system. There is a very high margin of error when integration is used to calculate velocity and distance even when the margin of error for the acceleration data is very small.

Angle Error (degrees)	Acceleration Error (m/s/s)	Velocity Error (m/s) at 10 seconds	Position Error (m) at 10 seconds	Position Error (m) at 1 minute	Position Error (m) at 10 minutes	Position Error (m) at 1 hour
0.1	0.017	0.17	1.7	61.2	6120	220 e 3
0.5	0.086	0.86	8.6	309.6	30960	1.1 e 6
1.0	0.17	1.7	17	612	61200	2.2 e 6
1.5	0.256	2.56	25.6	921.6	92160	3.3 e 6
2.0	0.342	3.42	34.2	1231.2	123120	4.4 e 6
3.0	0.513	5.13	51.3	1846.8	184680	6.6 e 6
5.0	0.854	8.54	85.4	3074.4	307440	11 e 6

Figure 49 Angle Error

Another change was having full talkback accessibility support for people who are vision impaired working on the application. This has fallen down to partially working while integrating the feature with due to the difficulty of having voice-over talkback functionality on a real-time graph, as a graph is an inherently visual view and talk-back would be stuck in an endless loop.

Lastly, originally a PHP server was going to be the back-end solution. Due to the scope changes made during the project, this server could be replaced by Firebase. Firebase was readily available, reduced cost, removed the extra complexity of designing custom REST APIs, firebase integrated perfectly into Android, and had a readily available real-time database. Firebase is also readily available to scale for multiple users and for a much heavier load to the realtime database.

7.2 Failure Report

FAILURE ANALYSIS REPORT	Report Number: 001
Written By: Rishabh Bhatnagar Min Jae Kim	Date: March 27, 2018
Evaluation: <p>The biggest failure during the BioABD project was also mentioned above. Due to there being multiple reasons for the abandonment of integration by the server to get velocity and distance, it was both a deviation from what was planned and a failure. Originally it was thought that just the integration algorithms would be enough and that the margin of error would be small. However during testing it was found that the margin of error is very big. Even a small calibration error could lead to big repercussions. This was not helped by the fact that the accelerometer was on a collar where the dog it was on would be constantly moving, leading to even more potential error. The lesson learnt from this was a lesson that was applied when making the application. The lesson was to make things as simple as possible. Instead of intercepting all data added to the server and performing complex calculations on it, adding a velocity and distance sensor would be much simpler. The application and microcontroller are designed in such a way that adding more sensors is quite simple, so this would be far superior, would get rid of all the error that comes along with integrating twice or even once. This concept was used when designing the application, which was done using the Model View Presenter design pattern, which specializes in keeping every component of the app simple and independent of each other, so if one fails the other components do not fail.</p>	

7.3 Lessons Learned

7.3.1 Defining the Problem to Solve

This was the stage when Eng 4K group members were in the problem space coming up with problems to solve. During this phase, the BioABD group just came up with a problem to solve and potential stakeholders. The thing the group would change would be to more clearly define the user base. During the time of defining the problem, there was a contention within the group on who would be the primary user base for the solution that the team develop. To resolve this, the group has decided that in future projects this will be discussed extensively prior to going into the solution space and defining a specific solution.

7.3.2 Finding the Solution to the Problem

This phase of the project was coming up with a solution for the problem defined in the problem space. Due to the contention in the problem space, there was a gap in vision as to who the user was, so there was a gap in vision for what the solution would be. This lead to multiple iterations of solutions, more than were needed. In the end, the group had to go back to the problem space, redefine the user base so everyone was on the same page, and then come up with the solution to the problem.

7.3.3 Prototyping

During this phase, there were key assumptions made. This came back to cause a failure. During prototyping of the integration and double integration to get velocity and distance, only the algorithms were tested. The algorithms were mathematically correct, but there were other factors such as margin of error, sensor position, and calibration which were underestimated. This caused a failure. Given the hindsight of today, the group would ensure this was thoroughly tested so this failure would be detected earlier in the lifecycle of the project. Next time, potential sources of error will be taken into account to ensure the chances of other failures are minimal.

7.3.4 Product Development

The development of BioABD was quite well done. The individual parts of the project (the application, embedded system, and the server) were all very well integrated into each other. However, the biggest change here was when the back-end was changed from a MySQL server to Firebase. The MySQL part of the project had already made very significant progress, and was almost complete when it was scrapped. This was due to the added complexity on the microcontroller and the application when using REST APIs, which had severe limitations. Given the hindsight of today, it would have been nice to use Firebase from the beginning, thus freeing up group member time for other parts of the project.

7.3.5 Product Testing

During the testing of BioABD, with the majority of the project developed, there was a stakeholder meeting with the primary stakeholders of the project. These stakeholders stated that they would prefer temperature and heart rate over velocity and distance. This caused minor panic in the group. Thankfully the project was very well designed so adding more sensors is a trivial job. However, due to the late timing of this, the sensor ordered was backlogged and thus would not be available in time for the finish date of the project. In the future, the BioABD group would like to have many more meetings with stakeholders to get very specific requirements from them to avoid this.

7.3.6 Back-end Migration From MySQL and PHP to Firebase

Initially, a MySQL PHP server was used for the project. This was because PHP is an high performance server-side scripting language, and could support the real-time integration demands of the project. MySQL is a enterprise-level SQL database that supports modelling and storage of different types of data records in database tables, and allows developer to define relationships between tables to relate different types of data. PHP has libraries for MySQL database operations, and built-in support for REST APIs which are responsible for communication between the server and the BioABD application. The PHP and MySQL combination allows for complete freedom in designing and implementing the business logic required of BioABD.

During the development of BioABD, the implementation of the MySQL server was almost complete when significant limitations were discovered in this choice of server. This combined with the change of scope of integration no longer being needed is why the MySQL was scrapped as the backend of BioABD and Firebase was used instead. The limitations of the MySQL server will be discussed below.

The main issue of the back-end with PHP and mySQL is the high degree of server concurrency when used as data storage for real-time application. The communication between the front-end and PHP + mySQL back-end is achieved via REST APIs, which rely on the HTTP protocol. The problem with exchanging information using the HTTP protocol is that there is an HTTP request made for every data record written to or retrieved from the server. The fact that sensory data are collected at a very high rate and required to be monitored with the mobile app in real-time, massive HTTP connections are established between the server and the microcontroller, as well as the server and the mobile app. Although the server has powerful processing power for handling many concurrent requests, but the processing capacity will eventually be insufficient for the BioABD project as the number of simultaneous users increases. Firebase provides a new method of information exchange using the WebSocket protocol. With this communication protocol, only one socket connection is needed between the server and the microcontroller, and another one between the server and the mobile device. The socket connection is persistent and allows information to be exchanged continuously in time.

Instead of the previous approach for data retrieval through bombarding the server with HTTP requests, Firebase provides a simpler and more efficient method for the Android application to retrieve data. Whenever new data storage occurs as an event, the server notifies and sends the data to the subscriber. The Android application subscribed to the database on the other hand,

listens for new incoming data and presents them. The method drastically alleviates the excessive concurrency imposed on the server as with the previous back-end. With Firebase, not only the deployed device can collect sensory data at a much higher frequency and present them on the Android application in real-time, but also allows a more scalable solution.

It is worth noting the limitations of Firebase also. Firebase can scale very well, but BioABD is running off the free tier. The free tier has the following limitations of usage per month. Maximum of 100 simultaneous connections (including sending and receiving), 1 GB of data stored, 10 GB of data downloaded per month, and only one database per project. This is not an issue while testing the product due to only one connection only being ran for a few hours every day. However when the product is in use 24/7, the usage may be higher. However this monthly cap should be more than enough. Below is the usage of the database during the past 30 days at the time of the writing of this report.

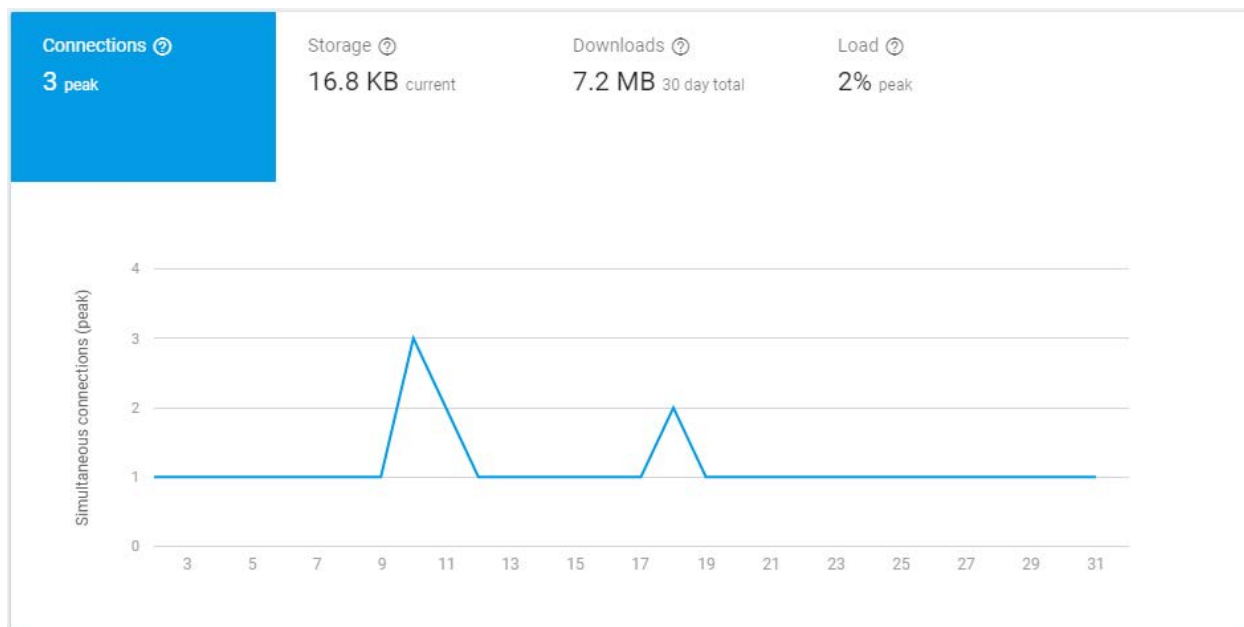


Figure 50 Usage of the database in the past 30 days as of 1/4/2018 (DD/MM/YYYY).

7.4 Possible Improvements in the Problem Solving Process

While the group believes that the problem solving used in the development of BioABD is quite good, there are some obvious improvements that need to be made. The biggest of which is the main failure in BioABD, namely the failure of the integration algorithm to function without a high margin of error. While in above sections it is mentioned that it can be replaced with a velocity

and distance sensor, the main issue that BioABD had was the late time in which the issue was found. As a result of this this potential solution to the issue could not be implemented in time.

Another potential improvement in the problem solving process for BioABD is having more regular meetings with stakeholders, as another potential feature that was missed due to time constraints was the heart-rate sensor. This was a feature that the stakeholders wanted but was not finished by the due date of this project. While this feature is still planned, it will be done after the current school year is over (since the group must keep the commitments it made).

7.5 Recommendations for Possible Improvements

Throughout the report there have been many recommendations for improvement. They will be summarized here, along with other suggestions.

- Heart rate sensor
- Velocity sensor
- Distance sensor
- Increased sample rate and update frequency
- Upgrading of Firebase tier to allow for more data storage and more data sent per month
- Adding authentication support native to the microcontroller and application, that is, not having to reprogram the app or microcontroller to switch users

To reiterate from above sections, adding sensors to BioABD is not a very difficult task due to how the software written for BioABD is architected. The difficulty is purchasing the right sensor, as the sensor must be small, low power, and light weight while also collecting correct data from the animal. Firebase and the application should support higher sample rates automatically since this is set on the microcontroller.

8 Deliverables and Setup

The deliverables for this project include:

- Code for BioABD Microcontroller
- Code for BioABD Application
- Li- Polymer Battery 2000mAh/ 3.7V
- ESP8266 microcontroller
- Temperature sensor and accelerometer
- 3D printed enclosure

8.1 Setting up BioABD Android Application

This section covers the deliverables and setup instructions for setting up the BioABD application locally. The prerequisites for this are Git and Android Studio. Open the terminal where you would like to download the github project. Type the following into the terminal window.

```
git clone https://github.com/Rishi-B/ngta.git -b master
```

Figure 51 Command for cloning master branch of BioABD app from Github.

Next, go into Android Studio and select the import project icon, navigate to the directory the application was cloned into and select it.

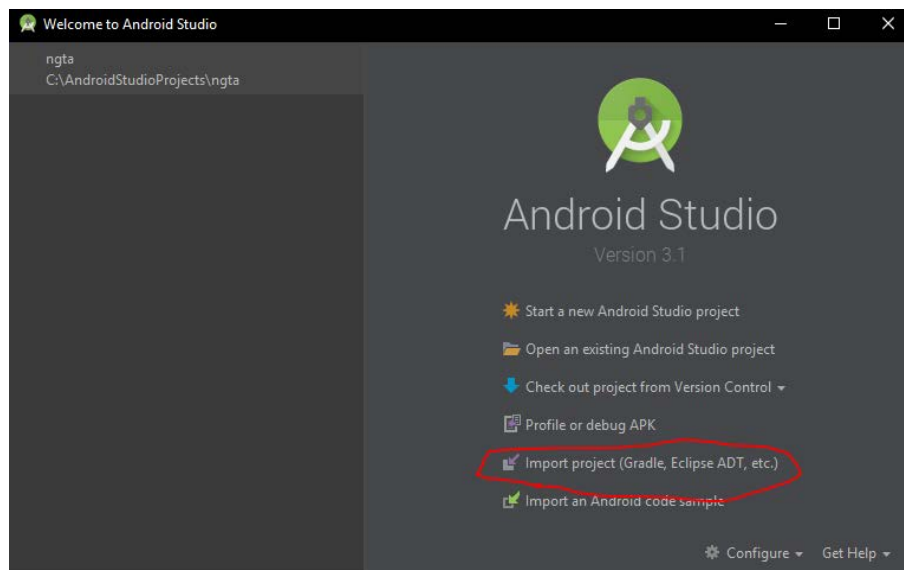


Figure 52 Option to import project into Android Studio.

The Gradle build will start, and may take some time due to the third-party dependencies involved. After this is done, connect up an android phone, click run and build to the connected device through ADB.

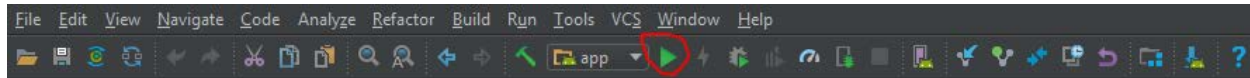


Figure 53 Run option

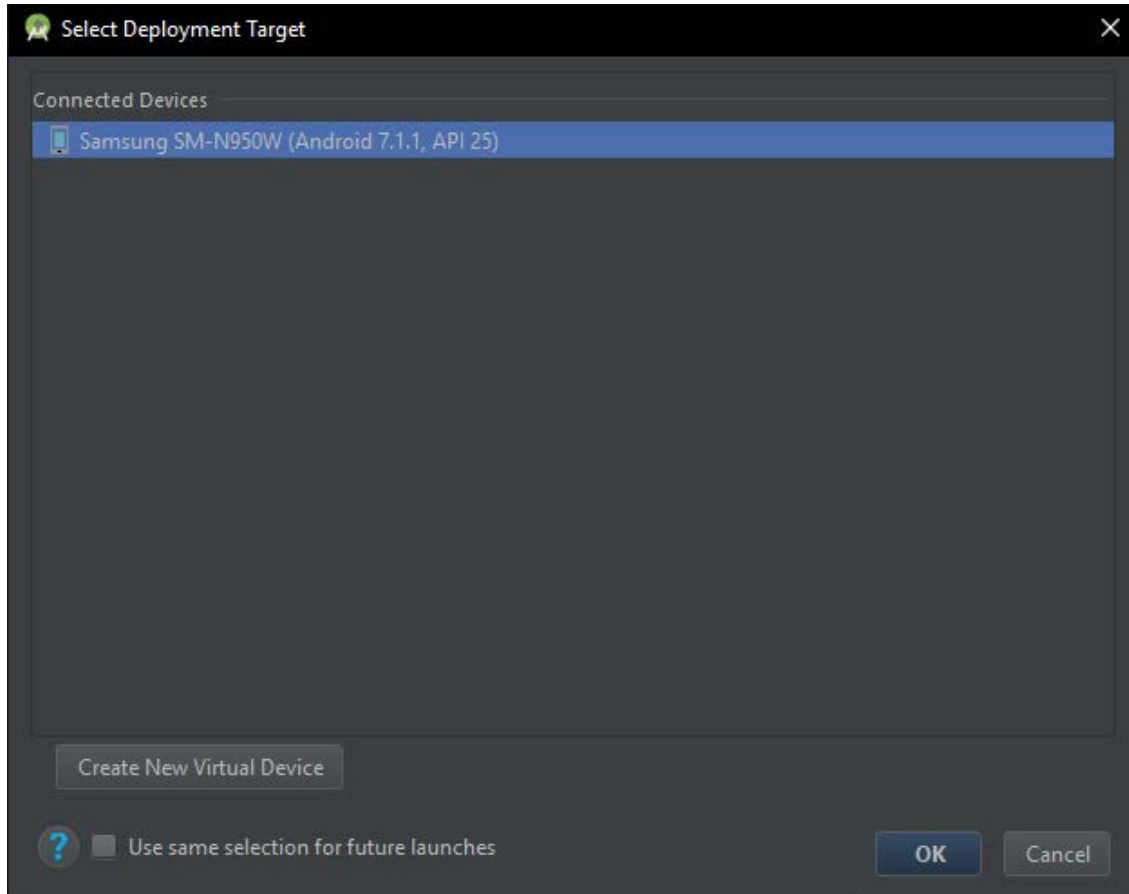


Figure 54 Select device to build to

9 Section IV - Project Self-Evaluation Matrix

9.1 Compliance with Laws, Regulations, Intellectual Property Rights

Criterion	Self-Evaluation Ranking	Justification
Explain the importance of compliance with the Professional Engineers Acts and other relevant laws, regulations, intellectual property guidelines and contractual obligations and follow best practices	Throughout BioABD project, the team has strived to meet all regulations, laws, and ethical standards that apply to engineers and for animal testing. This is why we believe we deserve an 4/4 in this department.	<p>Please refer to the Laws, Regulations, Guidelines, Intellectual Property, Best Practices, and Ethical Testing sections under Section 1 for more of an example. Following the necessary laws, regulations, obligations, and more, is very important. This is because following these laws helps keep a minimum standard for all products. For example, if regulations not to use lead in houses were not followed, it could heavily harm the residents of those houses. The BioABD group has gone above and beyond to ensure that any rules or regulations that apply have been met. An example of this, as mentioned in the requirements section, there is an explicit requirement for meeting the ethical requirements for animal testing. To ensure this was met, testing was done in the presence of Doctor Suzanne Macdonald, who has the ethical approval of York's Animal Care Committee. Additionally, the project developed is in no way intrusive to the animal, thus increasing the safety of the project.</p> <p>Following intellectual property rights allows for new inventions to not be copied directly without consequence. As a result it promotes individuals to create new innovative products using their technologies. The BioABD final report has many citations for the application for third-party APIs, to ensure credit is given where credit is due. This</p>

		can be seen by the citation of the Prolific Calendar and MP Android Charts APIs. It was very important for BioABD to follow these laws, regulations, and ethical requirements.
--	--	--

9.2 Employing Strategies of Reflection, Assessment, Self Assessment, of Team Goals in Multidisciplinary Settings

Criterion	Self-Evaluation Ranking	Justification
Employ strategies for reflection, assessment and self-assessment of team goals and activities in multidisciplinary settings	The BioABD group put a significant amount of work into looking back at the decisions made and assessing the rapidly changing goals of the team. This is why we believe we deserve an 4/4 in this department.	Since the PDR, the team has been adapting to changes in the client specification. Moreover, the team has been assessing the ever changing goals and objectives. While we have two computer engineering students and two electrical engineering students, each member was assessing each other's work. For example, Min Jae Kim, a computer engineering student, was working on the embedded system design (the circuit and code of the embedded system) while Xingkai Wu, a electrical engineering student, was assessing Min Jae Kim's work with his knowledge in electrical engineering practices.

9.3 Adhering to Written Instructions in a Professional Context

Criterion	Self-Evaluation Ranking	Justification
Adhere to written instructions in a professional context	Due to adhering to all of the required written instructions, the team deserves a 4/4.	Throughout BioABD project, the team has strived to meet all written instructions of the written specifications of the client, stakeholders, manufacturer data sheets and other documents, and the ENG4000 course.

9.4 Evaluating Critical Information in Reports and Design Documents

Criterion	Self-Evaluation Ranking	Justification
Evaluate critical information in reports and design documents	The BioABD group put a significant amount of work into the reports and design documents of BioABD. The group believes that because of the in-depth technical information included in the report and the design documents, that the mark here should be a 4/4.	<p>BioABD has gone to great lengths to ensure that there was a high amount of information in the report. This means that all of the critical information about BioABD is included here. To show this, here are a few examples.</p> <ul style="list-style-type: none"> - The requirements are clearly laid (Requirements Review section) - Each requirement has information on how it has been met (Requirements Review section or As-Built section) - There is a high amount of technical information showing how each individual component of BioABD was designed (As-Built Design Compliance section) - Appendixes walking through code showing how parts of BioABD were implemented (Appendixes) - Information on methodology used to develop components of the application (As-Built Design section Application subsection, Agile methodology and MVP Design Pattern) - Showing very technical details about the electronic components (As Built Design Compliance Section Embedded System subsection)

9.5 Appraise Possible Improvements in the Problem Solving Process

Criterion	Self-Evaluation Ranking	Justification
Appraise possible improvements in the problem solving process	The group knows that BioABD is not perfect. It can be improved and it will be in the future. This report has mentioned potential improvements many times. This is why the group believe the BioABD report deserves a mark of 4/4 in this section.	The report has a section for improving each group members' problem solving. Other than that section, the report has mentioned the potential addition of a heartbeat sensor multiple times. This can be seen by the executive summary section, the deviations from plan section, the product testing section, and more. The group strongly believes in improving the existing product and how vital this is shows their commitment to adding new features. Additionally, looking at the failure report section, it can be seen that the addition of a velocity and distance sensor was mentioned. This shows the problem solving skills of the group. Specifically, how the project was designed in such a way that a failure in one part of the project (namely the integration part) does not stop BioABD and how the group can problem solve to find other solutions which would be easy additions to the framework that BioABD already has implemented.


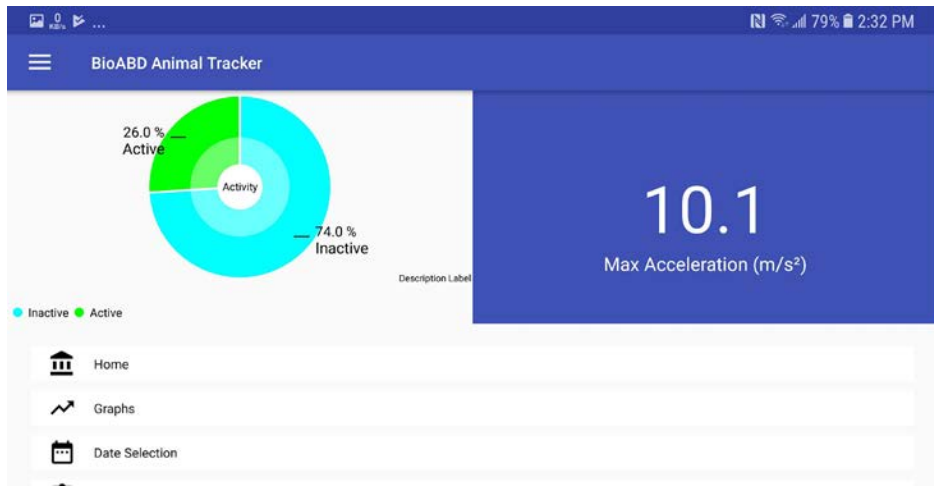
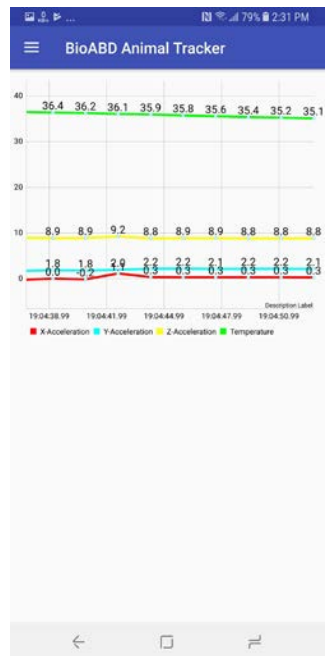
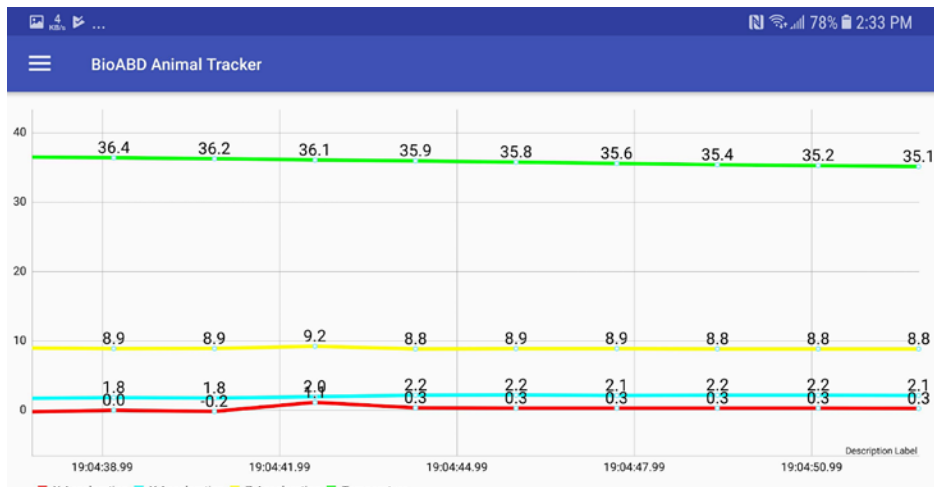
9.6 Justifying the strengths and Limitations of the Solution and Making Potential Solutions

Criterion	Self-Evaluation Ranking	Justification
Justify the strength and limitations of the solution and make recommendation for possible improvements	BioABD believes they have justified why they think certain parts of the project are done very well and which parts of the project are limitations that need to be worked on. This is why the group believes they deserve a grade of 4/4.	The group has done significant analysis. Looking at the Preliminary Business Case Strength, Weaknesses, Opportunities, and Threats (SWOT) section is one example of this. The limitations of the old solution of the PHP server can be found in the Back-end Migration from PHP to Firebase section, as well as some limitations of Firebase. The report has also mentioned, multiple times, that the heart rate is a feature that will be added and is a weakness (mainly in the SWOT section). The group has also stated a solution to the failure BioABD experienced (integration). The solution mentioned was for adding a distance and velocity sensor. This was mentioned in the Failure Report section.

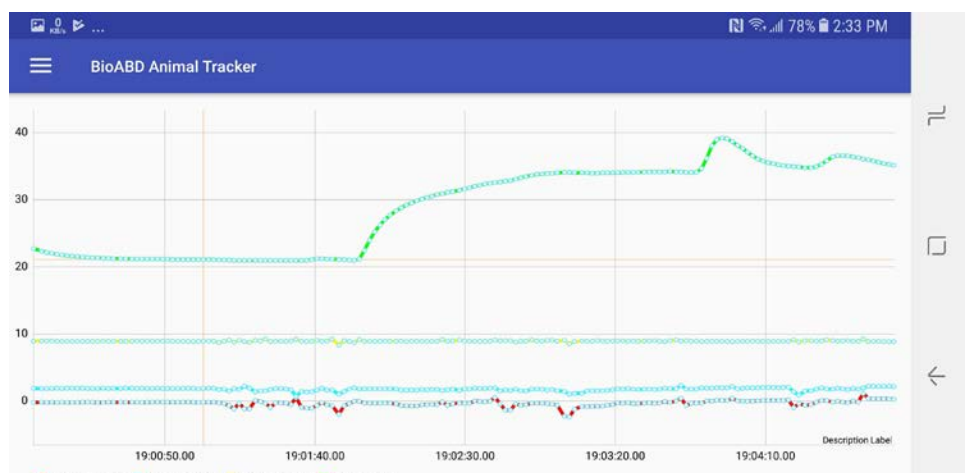
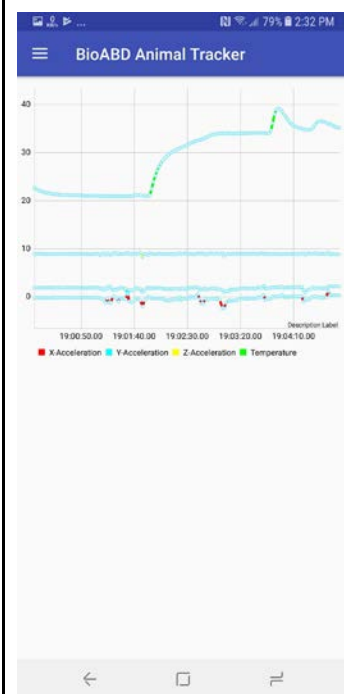
(THIS PAGE IS INTENTIONALLY LEFT BLANK)

10 Appendix 1: Application Screenshots

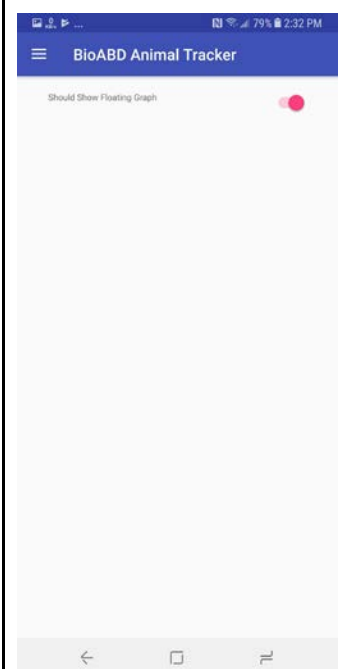
Please note that the Profile and Calendar are placeholders for future planned features.

Screen Name:	Portrait:	Landscape:
Dashboard		
Default Graph		

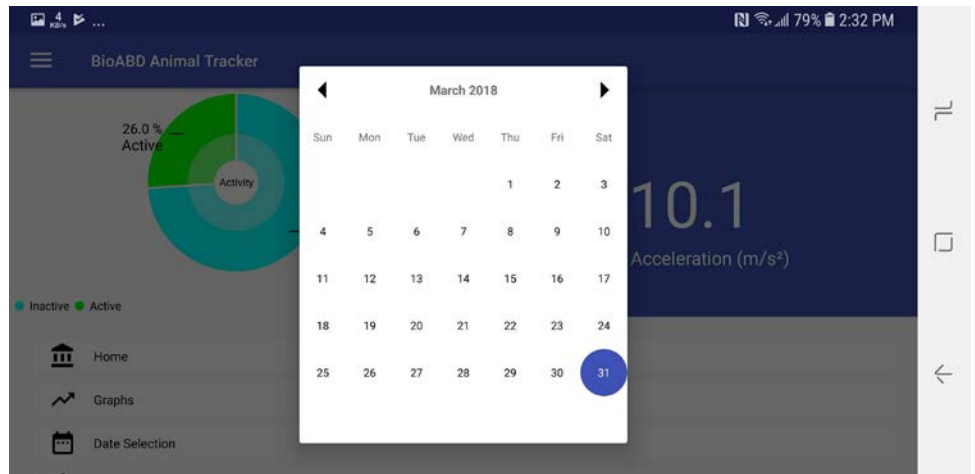
Alternative Graph



Settings:



Calendar:



11 Appendix 2: Parsing code for values from Firebase

```
database.child("adx1345").addChildEventListener(new CustomChildEventListener() {  
    @Override  
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {  
        GraphPoint point = new GraphPoint();  
        try {  
            point = dataSnapshot.getValue(GraphPoint.class);  
        } catch (DatabaseException databaseException) {  
            Log.e(TAG, "Added invalid type to database. Cannot parse. " + databaseException);  
        } catch (Exception exception) {  
            Log.e(TAG, "Something went wrong on our end. " + exception);  
        }  
        if (point != null && !point.areChildrenNull() && !point.areChildrenEmpty()) {  
            // Point is parsed and point is not empty  
            announcer.addValue(point);  
        }  
    }  
});
```

Figure 55 Code for parsing values from Firebase

The above code creates a listener (more specifically a callback listener) that is called whenever there is a child added to the adxl345 node of Firebase. This is then parsed in a try-catch statement. After this, validation is done to ensure there are no nulls, the point is not null, and the children of the parsed values are not empty. This is because android loves to crash due to null-pointer exceptions, and this prevents them in-case something goes wrong. Below is the object that is used as a model when parsing.

```
public class GraphPoint {  
    private Long unixTime;  
    private Float xValue;  
    private Float yValue;  
    private Float zValue;  
    private Float tempValue;  
  
    public GraphPoint() {
```

```
// Needed if parsing fails.
}

public GraphPoint(long unixTime, Float xValue, Float yValue, Float zValue,
    Float tempValue) {
    this.unixTime = unixTime;
    this.xValue = xValue;
    this.yValue = yValue;
    this.zValue = zValue;
    this.tempValue = tempValue;
}

public long getUnixTime() {
    return unixTime;
}

public Float getXValue() {
    return xValue;
}

public Float getYValue() {
    return yValue;
}

public Float getZValue() {
    return zValue;
}

public Float getTempValue() {
    return tempValue;
}

public boolean areChildrenEmpty() {
    return unixTime <= 0 || (xValue == 0 && yValue == 0 && zValue == 0
        && tempValue == 0);
}

public boolean areChildrenNull() {
    return unixTime == null || xValue == null && yValue == null || zValue == null
        || tempValue == null;
}
```

Figure 56 Contract for a correctly parsed value from Firebase.

12 Appendix 3: Interfaces updating graph in real-time

It is worth noting that the way that this code is written, it can be used for anything added to firebase, as all the code is generic and the only thing that would need to be changed is the model object to another model object. Thus the written code is extremely polymorphic and can be used globally by nearly anyone due to the generic nature and the minimum code changes needed to absorb any changes to the server.

Appendix 2 shows how data from firebase is parsed. When the addValue method is called within the the firebase callback function after the null and empty checks, it does the following. Firstly, the code for announcer is as follows. It is simply a HashSet of listeners. This is because if there are multiple graphs on the page, this architecture would easily allow all of them to be updated at one time.

```
public class GenericLineChartAnnouncer<T> {  
    private final HashSet<GenericLineChartListener<T>> map = new HashSet<>();  
  
    public void clearListeners() {  
        map.clear();  
    }  
  
    public void addListener(GenericLineChartListener<T> listener) {  
        map.add(listener);  
    }  
  
    public void addValue(T point) {  
        for (GenericLineChartListener<T> listener : map) {  
            if (listener != null) {  
                listener.addValue(point);  
            }  
        }  
    }  
}
```

Figure 57 Code for announcer variable from Appendix 2.

The next object is the Generic listener that goes inside the HashSet. This listener takes a generic model T. This means that the uses are not just limited to this project but to any project that has parsed objects. Below is the code for the embedded listener within the HashSet.

```
/**
 * This acts as an interface for the line chart. It is implemented in the line chart binder.
 */
public interface GenericLineChartListener<T> {

    /**
     * Notifies chart to add a point to the graph.
     * @param point
     *     Point to be added.
     */
    void addValue(T point);
}
```

Figure 58 Listener that adds value to the graph

The binder for the graph (which the class that takes the generic values and gives the view the values it needs to display) then has the implemented listener, and it is called from the presenter. This means that when the presenter gets the new point from firebase, it goes directly into the binder and updates the graph without reloading or touching any other portion of the page, as projects made in MVP are supposed to do.

However there is a second listener. This listener takes an input of the data in the graph from the binder, goes to the presenter, adds this new value from firebase to the line data object for the graph, and then passes it back to the binder for it to update the graph. Thus keeping all logic within the presenter. Below is the code for this listener. It is the opposite of the above announcer and listener. It is implemented inside the presenter and called from the binder, thus does not need to be encapsulated within an object.

```
* This takes communicates between the Presenter and Binder and adds point of model type T to the
* LineData set.
* @param <T>
*   T is the model for the graph to parse.
*/
public interface LineDataListener<T> {
    /**
     * This adds point of generic type specified to the data set of the line chart.
     * @param data
     *   Data existing on graph.
     * @param point
     *   Point to add.
     */
    void addPointToDataSet(LineData data, T point);
}
```

Figure 59 Listener that adds value to the data for the graph to then plot

13 Appendix 4: FireBase Crash Tracking

Firebase can track application crashes if they are implemented into the application. Here are the crash statistics for the BioABD Application. Please note it shows one daily active user because on this day only one person used it.

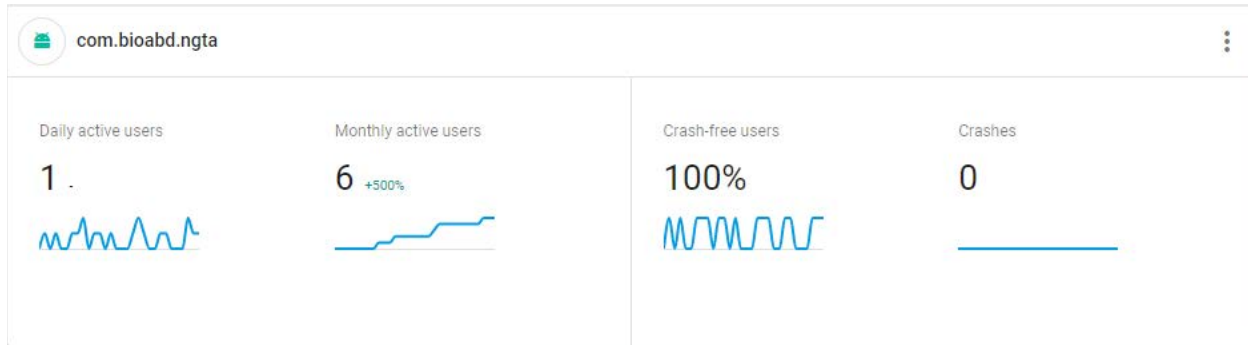


Figure 60 Evidence of crashless application

Please note that the application name is `bioabd.ngta` where `bioabd` stands for the team name. The acronym `ngta` stands for Next Generation Tracking Application, a reference to somewhere one of the team members of BioABD works.

14 Appendix 5: Settings and Global Variables

In Android, there are two ways to handle global variables that will be accessible from anywhere in the app. One is an Application class and one is a Singleton Pattern. The singleton pattern is a singleton class (class only instantiated one time) that is passed around to/from every screen on the app. This has its upsides and downsides. In BioABD the Application class was chosen. An application class is instantiated every time an app is started. It can have values stored in it. The application class in BioABD is named TrackerApplication. It only has specific variables and it only has private variables along with getters and setters. The variables are private because of encapsulation.

Inside TrackerApplication is a boolean variable (shown below) that holds a value. The default is true. When the user opens the settings page and toggles the switch, the following code is ran in the presenter for the settings page.

```
public CompoundButton.OnCheckedChangeListener getFloatingGraphCheckChangeListener() {
    return new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton compoundButton, boolean isChecked) {
            isFloatingGraphOn = isChecked;
            view.setIsFloating(isFloatingGraphOn);
        }
    };
}
```

Figure 61 Code that sets the global variable to the desired value

The view setter called inside the check change listener calls an interface that communicates between the presenter and the view. The following code sets the boolean global flag.

```
@Override
public void setIsFloating(boolean isFloating) {
    ((TrackerApplication) this.getApplication()).setGraphFloating(isFloating);
}
```

Figure 62 Code that sets the global variable to the desired value

When the graphs page is loaded, the data adding method shown in Appendix 3 does a simple check. It checks if the global flag for a floating graph is on or not. Recall that a floating graph is a graph that limits how much information can be on the x-axis at one time, having the user scroll left and right to view older and newer data respectively. Below is the check done inside the binder for the graph.

```
if (isFloating) {
    holder.chart.setVisibleXRangeMaximum(15000);
}
```

Figure 63 Code that checks if x range should be limited, creating a floating graph

Below are two screenshots showing the effect of this.

Floating Graph On:

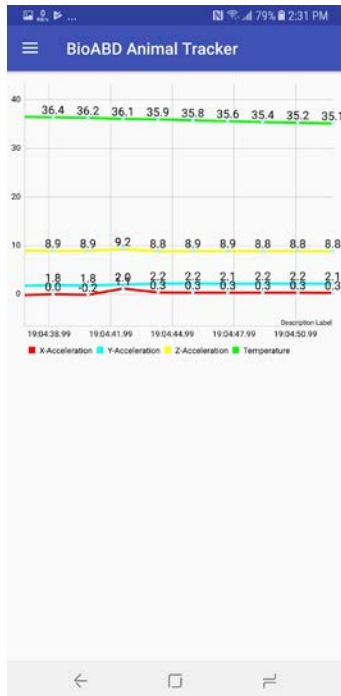


Figure 64 Floating Graph

Floating Graph Off:

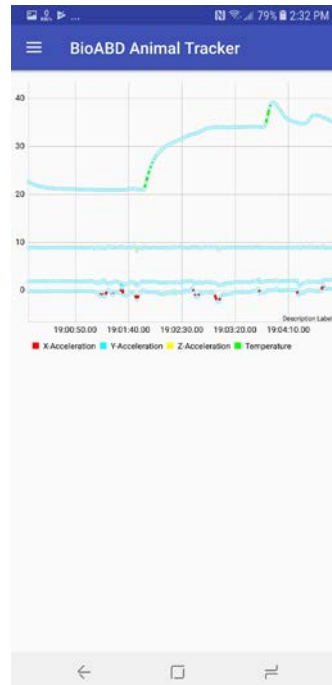


Figure 65 Non-floating graph

15 Sources

15.1 ESP8266 Specification Sheet

Adafruit. (2015, November 25). Adafruit Feather HUZZAH ESP8266 Datasheets. Retrieved March, 2018, from <https://learn.adafruit.com/adafruit-feather-huzzah-esp8266/downloads>

15.2 Li-Po Battery Technology Specification

SHENZHEN PKCELL BATTERY CO., LTD. (2014, June 03). Li-Polymer Battery Technology Specification. Retrieved March, 2018, from <https://cdn-shop.adafruit.com/datasheets/LiIon2000mAh37V.pdf>

15.3 ADXL345 Accelerometer Data Sheet

Analog Devices. (2009). ADXL345 (Rev. 0) Digital Accelerometer. Retrieved March, 2018, from <https://www.sparkfun.com/datasheets/Sensors/Accelerometer/ADXL345.pdf>

15.4 Dog Sports Harness

ComfortFlex. (n.d.). Harnesses. Retrieved March, 2018, from <http://comfortflexharness.com/>

15.5 BioABD Android Application GitHub Link

Bhatnagar, R. (2017, October 01). Rishi-B/ngta. Retrieved March, 2018, from <https://github.com/Rishi-B/ngta>

15.6 Android vs. iOS Market Share

S. (2009, January 01). Mobile OS market share 2017. Retrieved March, 2018, from <https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/>

15.7 Using Acceleration to Estimate Distance and Velocity

C. (n.d.). Using Accelerometers to Estimate Position and Velocity. Retrieved March, 2018, from <http://www.chrobotics.com/library/accel-position-velocity>

15.8 Agile vs. Waterfall

Kremic, N. (n.d.). Why is Agile Time to Market (TTM) Delivery 50% Faster? . Retrieved March, 2018, from <http://www.deltamatrix.com/why-is-agile-time-to-market-ttm-delivery-50-faster/>

B. (2017, September 07). Six benefits of using MVC model for effective web application development. Retrieved March, 2018, from <https://www.brainvire.com/six-benefits-of-using-mvc-model-for-effective-web-application-development/>

15.9 MP Android Chart - Android Charts

Jay, P. (2018, February 24). PhilJay/MPAndroidChart. Retrieved March, 2018, from <https://github.com/PhilJay/MPAndroidChart>

15.10 Android Resource Monitoring Application

Parajuli, D. (2018, February 21). Simple System Monitor - Apps on Google Play. Retrieved March, 2018, from <https://play.google.com/store/apps/details?id=com.dp.sysmonitor.app&hl=en>

15.11 Android MP Prolific Calendar

Prolific Interactive. (2017, May 16). Prolificinteractive/material-calendarview. Retrieved April 01, 2018, from <https://github.com/prolificinteractive/material-calendarview>

16 Components

16.1 Microcontroller

Cost: \$23.28

Digikey. (n.d.). Feather HUZZAH with ESP8266 WiFi. Retrieved March 31, 2018, from <https://www.digikey.ca/catalog/en/partgroup/feather-huzzah-with-esp8266-wifi/57892>

Digikey. (n.d.). HUZZAH32 – ESP32 Feather Board. Retrieved March, 2018, from <https://www.digikey.ca/catalog/en/partgroup/huzzah32-esp32-feather-board/68696>

16.2 Battery

Cost (6-pack): \$25.85

https://www.amazon.ca/gp/product/B0714JPMWM/ref=oh_aui_detailpage_o01_s00?ie=UTF8&psc=1

16.3 Temperature Sensor

Cost (3-pack): \$14.98

https://www.amazon.ca/gp/product/B012C597T0/ref=oh_aui_detailpage_o02_s00?ie=UTF8&psc=1

16.4 Accelerometer

Cost: \$14.95

https://www.amazon.ca/gp/product/B01ISMV6EU/ref=oh_aui_detailpage_o02_s00?ie=UTF8&psc=1

16.5 Remark

Please note the components can be acquired for much more affordably, but due to time limitations, the more expensive but more readily available options were chosen.