

**Department of Computer Science**

This project has been satisfactorily demonstrated and is of suitable form.

This project report is acceptable in partial completion of the requirements for the Master of Science degree in Software Engineering.

|  |  |  |
| --- | --- | --- |
| **Embedded Software Development Case Study** | | |
| Project Title (type) | | |
| Alexander Gauf |
| Student Name (type) |
| Dr. Chang-Hyun Jo |
| Advisor's Name (type) |
|  | |
| Advisor's signature | Date |
| Dr. Ning Chen |
| Reviewer's Name (type) |
|  | |
| Reviewer's signature | Date |

**Abstract**

This paper outlines the development of an embedded software system. A hybrid development method is tailored to a small-scale development team with a focus on pulling together best practices for software development and hardware development. This development method is then applied to development of an accelerometer system with android mobile integration for use in educational physics classrooms. Finally, a “lessons learned” section will note advantages, disadvantages and recommended changes to the hybrid development method.

**Keyword List:**

Embedded System, Software Development, Hardware Development, Hybrid Development, Education, Physics Education, Case Study, Physics Lab Hardware, Android Application.

Table of Contents

[**1.** **Introduction** 6](#_Toc512283076)

[**1.1.** **Description of the Problem** 6](#_Toc512283077)

[**1.2.** **Project Objectives** 6](#_Toc512283078)

[**1.3.** **Development Environment** 6](#_Toc512283079)

[**1.4.** **Operational Environment** 7](#_Toc512283080)

[**2.** **Development method for Embedded System** 8](#_Toc512283081)

[**2.1.** **The Embedded Software Development Model.** 8](#_Toc512283082)

[**Figure 2. 1 – Initial embedded software development method.** 8](#_Toc512283083)

[**2.2.** **Model Important Notes** 9](#_Toc512283084)

[**2.3.** **The Hybrid Development Method Explained** 10](#_Toc512283085)

[**2.3.1.** **System Definition** 10](#_Toc512283086)

[**2.3.2.** **System Development** 11](#_Toc512283087)

[**2.3.3.** **System Deployment** 12](#_Toc512283088)

[**3.** **Requirements Description** 13](#_Toc512283089)

[**3.1.** **Requirements Summary Table** 13](#_Toc512283090)

[**Table 3.1 – Requirements Summary Table** 13](#_Toc512283091)

[**3.2.** **System Requirements Specification** 15](#_Toc512283092)

[**3.2.1.** **Introduction** 16](#_Toc512283093)

[**3.2.2.** **Overall Description** 18](#_Toc512283094)

[**3.2.3.** **System Features** 20](#_Toc512283095)

[**3.2.4.** **Data Requirements** 23](#_Toc512283096)

[**3.2.5.** **External Interface Requirements** 24](#_Toc512283097)

[**3.2.6.** **Software Quality Attributes** 25](#_Toc512283098)

[**3.2.7.** **Analysis Models** 27](#_Toc512283099)

[**4.** **Design Description** 28](#_Toc512283100)

[**4.1.** **Decomposition and Uses view** 28](#_Toc512283101)

[**Figure 4.1- Decomposition and Uses View** 29](#_Toc512283102)

[**4.2.** **Component and Connector plus Allocation Diagram** 30](#_Toc512283103)

[**Figure 4.2 – Component and Connector Plus Allocation View** 31](#_Toc512283104)

[**4.3 Circuit Diagram** 32](#_Toc512283105)

[**Figure 4.3 – Circuit Diagram** 32](#_Toc512283106)

[**4.4** **Case and Final Design** 33](#_Toc512283107)

[**Figure 4.4 – Hardware Open 1** 33](#_Toc512283108)

[**Figure 4.5 – Hardware Open 2** 33](#_Toc512283109)

[**Figure 4.6 – Hardware Closed** 34](#_Toc512283110)

[**Figure 4.7 – Battery Compartment** 34](#_Toc512283111)

[**Figure 4.8 – Peripherals** 34](#_Toc512283112)

[**5.** **Implementation** 35](#_Toc512283113)

[**5.1.** **Organization of source file structure** 35](#_Toc512283114)

[**5.2.** **Reference list of files** 35](#_Toc512283115)

[**6.** **Test and Integration** 37](#_Toc512283116)

[**6.1.** **Test Suite IDs** 37](#_Toc512283117)

[**6.2.** **Text Cases** 37](#_Toc512283118)

[**6.2.1.** **TS000 – Requirements Acceptance Tests.** 37](#_Toc512283119)

[**6.2.2.** **TS001 -Android App Unit and Component Tests** 39](#_Toc512283120)

[**6.2.2.** **TS002 - Arduino Hardware Unit and Component Tests** 51](#_Toc512283121)

[**6.2.3.** **TS003 - Android App Integration Tests** 56](#_Toc512283122)

[**6.2.4.** **TS004 - Arduino Hardware Integration Tests** 60](#_Toc512283123)

[**6.2.5.** **TS005 - System Integration Tests** 62](#_Toc512283124)

[**6.2.6.** **TS006 - Android App Acceptance Tests** 64](#_Toc512283125)

[**6.2.7.** **TS007 - Arduino Hardware Acceptance Tests** 68](#_Toc512283126)

[**6.2.8.** **TS008 - System Acceptance Tests** 70](#_Toc512283127)

[**7.** **Installation Instructions** 73](#_Toc512283128)

[**7.1.** **Installing the android software** 73](#_Toc512283129)

[**7.2.** **Installing the Embedded System Software** 73](#_Toc512283130)

[**7.3.** **Testing the hardware with the android phone** 73](#_Toc512283131)

[**Figure 7.1 – Bluetooth Test** 73](#_Toc512283132)

[**8.** **Operating Instructions** 74](#_Toc512283133)

[**8.1.** **Running the System** 74](#_Toc512283134)

[**8.1.1.** **Starting the Hardware** 74](#_Toc512283135)

[**8.1.2.** **Shutting down the hardware** 74](#_Toc512283136)

[**8.1.3.** **Testing the Hardware to Android connection** 74](#_Toc512283137)

[**8.1.4.** **Opening the experiment pages** 75](#_Toc512283138)

[**8.1.5.** **Taking data for the experiments** 75](#_Toc512283139)

[**9.** **Recommendations for Enhancement** 76](#_Toc512283140)

[**9.1. Process Model Lessons learned** 76](#_Toc512283141)

[**9.1.1.** **Hardware Selection** 76](#_Toc512283142)

[**9.1.2.** **Difficulty of Architectural Design Refinement** 76](#_Toc512283143)

[**9.1.3.** **Hardware Interfacing** 76](#_Toc512283144)

[**9.1.4.** **Testing** 76](#_Toc512283145)

[**9.2. Process Model Recommended Adjustments** 77](#_Toc512283146)

[**9.2.1.** **Updated Model** 77](#_Toc512283147)

[**9.2.2.** **Changes** 78](#_Toc512283148)

[**9.2.3.** **Case Study System Possible Additions** 78](#_Toc512283149)

[**10.** **Bibliography** 79](#_Toc512283150)

[**10.1. Referenced** 79](#_Toc512283151)

[**10.2 Not Directly Referenced/ Used in development.** 80](#_Toc512283152)

1. **Introduction** 
   1. **Description of the Problem**

Embedded software systems are finding their way into every aspect of our lives, from the phones in our pockets to the controllers in a child’s new light up toy. Most of our computing needs are now in these devices, with the number and complexity of embedded devices only growing. [3] Unfortunately today most methods for developing these systems are still lacking, since the evolving software development methods have not yet been blended with established hardware development methods. Embedded systems require this fusion which has led to an industry need for both embedded software engineers knowledgeable on both sides and engineering methods that work consistently for the entire system, as opposed to hardware and software separately. [10]

A survey of papers shows that many different methods have been generated and applied in the development of embedded software systems, all of which attempt to deal with different consistent issues in the field. [2], [5], [6], [8], [9], [10] These issues include limited resources, tight market windows, lack of architecture support, and most importantly the difficulty of blending together both hardware and software development. [4], [7] The main differentiator between standard software development and embedded software development are the stricter constraints that hardware imposes on software projects. [9] Requirements for an embedded software system cannot be changed so readily since available hardware often have severe limitations on what they can achieve. [8] Thus there is a need for a development process using the best practices of software engineering and combining them with practices from systems engineering to ensure the best possible system is developed.

This problem of process development is being tackled by many, but one thing lacking are applied case studies for these methods that include lessons learned for the process. While many systematic reviews have been performed often it is difficult to get useful information after the project has been wrapped up with no consideration of what may have gone wrong or right with the process. [7], [8]

* 1. **Project Objectives**

There were two main objectives for this project to address the problem described in section 1.1.

First an embedded systems development method was created using results from other successful or published processes. This included finding a proper method for this embedded software development and then tailoring it using principles from other methods from software engineering to systems engineering and successful or unsuccessful embedded software projects. This involved identifying what was helpful and what may have been unnecessary for the product development to satisfy the needs of the embedded systems development world.

The next objective was to develop an embedded software system using this method to provide a case study for the process. This allowed for lessons learned and further refinement of the method to better future projects and research in the area, as well as improved the skills of the system developer.

The case study created to satisfy this second objective was an accelerometer system for use in mechanical physics labs. It is a small device that is “throwable” by students and takes and transmits acceleration data to an android phone for analysis. This data can be used to display basic physics principles such as acceleration, free fall, projectile motion, and many more.

* 1. **Development Environment**

This project involved development of an embedded software system which included both hardware and software development.

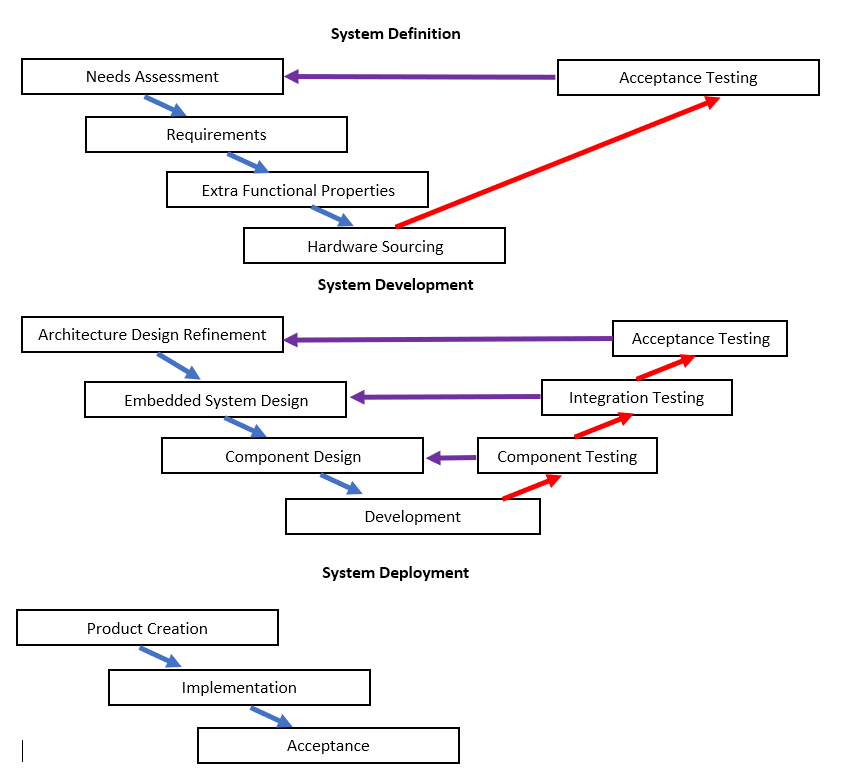
The hardware was a windows 10 desktop PC, multiple android phones running either 7 Nougat or 8 Oreo Operation system, and an Arduino Pro Micro with multiple attachments. The Arduino board has a HC-05 Bluetooth module attached to communicate with the android phone and an MMA8451 accelerometer for taking data. Component testing was mostly done on an Arduino UNO before migrating to the Arduino Pro Micro.

The software development included developing an android application and Arduino software to run the hardware system. Android app development was done using android studio 3.0.1. Arduino development was done using the Arduino IDE.

* 1. **Operational Environment**

The mobile software operated in android 7 or 8 using Java. The Arduino hardware is run by C++. The final system is to be deployed in an introductory physics classroom for use by groups of two to four college students. It may also be deployed in Demonstration environments for Physics clubs or other similar activities.

1. **Development method for Embedded System**
   1. **The Embedded Software Development Model.**

The completion of this project was done using a modified version of the hybrid development model bringing together Systems Engineering and Software engineering outlined by Suliman et al. in [10]. The method they specified is similar to the software development V-Model but ties it into a three-stage development method from systems engineering. That method was adjusted based on other successful and unsuccessful practices to create the model shown below.

**Figure 2. 1 – Initial embedded software development method.**

* 1. **Model Important Notes**

The reason this model was selected is for the focus on three stage production. Since embedded systems have a hardware component there are harsher limits on flexibility than software only projects. Thus, we have the three stages of systems engineering incorporated: System Definition involves requirements definition and hardware estimation before any development is done, System Development focuses on component development then system integration, then Deployment creates a final product and ensures final acceptance.

The creators of this method planned to apply it to a case study in their future work.

In the development phase the three steps of Partitioning, Allocating, and Scheduling noted to be important in embedded system design are grouped into “Embedded System Design.” [10] This is done to simplify the model to be closer to the SW V-model. The practices mentioned are detailed in the definition of Embedded System Design.

When designing this method a few best practices were taken from other authors. Michael Smith et al. stressed the importance of test driven development in these systems, so this will be integrated in each step [9]. Also, multiple authors found prototyping to be useful in developing embedded systems as it helps bridge the gap between software and hardware [6], [9]. Taking this into consideration leads to defining the development stage into conceptual design, then prototype creation is recommended for development, followed by testing then repetition of the design and possible creation of multiple prototypes. It’s worth noting that for major projects this may become expensive fast depending on the cost of parts for the prototype; however, the existence of the prototype addresses a concern found by Rong et al. where they found conflicts between hardware and software testing environments were a major issue in embedded systems development [7]. By creating component prototypes and integrating them the testing can be done directly on the system so hardware limits can be assessed, and software can be tested directly in hardware.

Finally, it is worth noting that this method could be easily extended to an agile style hybrid. For example, the development of prototypes allows for frequent deliverables to product owners. It is noted though that in embedded systems it can be more difficult and costly to change hardware selections during development. So contrary to Agile’s need for flexible requirements some limitations would have to be agreed to before any development is done, otherwise costs may explode for rapidly changing hardware requirements.

* 1. **The Hybrid Development Method Explained**

This section details the parts of the development method shown in Figure 2.1. This is written to be referred to by anyone wishing to apply this method to their development. Checklists and notes were included in this section for that purpose.

* + 1. **System Definition**

This stage of development should make sure that the correct product is being developed as defined at the time of conception. The following checklist should be completed in this phase.

* + Needs are assessed.
  + Requirements are clearly defined.
  + Acceptance tests defined.
  + Hardware sources selected.
  + Extra-functional requirements ranked and compromised on.
  + Needs and Requirements revisited after source selection until all initial acceptance tests pass.
    - 1. **Assessment of Needs**

This is where the product vision is first described. Define:

* What type of product is needed and what solutions to provide
* Define gaps between current systems if they exist and the desired system.

For more thorough documentation of this phase a Vision and Scope document is recommended.

* + - 1. **Requirements Definition**

Requirements are detailed and documented. Best practices such as interviewing end users or workshops are recommended. This step should end with

* A requirements specification document such as an System Requirements Specification, see section 3.2 of this report for an example.
* Definition acceptance tests defined from requirements.
  + - 1. **Identifying and Ranking Extra-Functional Properties**

This step defines the system limits. Since the hardware of the system can strictly limit capabilities the range of customer wants should be determined early.

Prioritize the list of EF properties, a possible list is given here but there may be others for certain projects:

1. Reliability
2. Usability
3. Efficiency
4. Maintainability
5. Memory Consumption
6. Modifiability

When prioritizing be sure to keep in mind the deployment environment, it is highly recommended to work with users on this step.

* + - 1. **Hardware Source Selection**

Once requirements and EF properties are decided the hardware to achieve the products goals should be determined and documented here. It’s important to ensure thorough documentation so that should the hardware need to be changed or rebought during the prototyping step it can be easily referenced. Software languages should also be determined based on the hardware chosen

* + - 1. **Requirements Acceptance Testing**

Before moving on from System Design Acceptance testing should be done to be sure that the defined product fits user needs. Any issues in acceptance testing should lead to going back to step 1.1 and repeating this phase until all acceptance tests pass. It is likely that requirements and EF priority may need to be readdressed after source selection based on budget and source limitations.

* + 1. **System Development**

Once project limits have been set on the product to be created system development can begin. The following checklist should be completed in this phase.

* Get the big picture and making proper architecture choices.
* The embedded system should be designed following a method described below which involves breaking the system into components and designing those components.
* System integration tests determined.
* Components should be designed
* Component tests determined.
* Component development, prototype creation recommended
* Component testing, looping back to component design and possible new prototypes
* Integration testing, looping back into the embedded system and repeating component/prototyping
* Acceptance testing looping back to architecture choices.
  + - 1. **Refine Conceptual Architecture**

Proper architecture should be determined to fill the big picture of the system. This will be the step that takes significant experience and will be refined over multiple iterations.

* + - 1. **Embedded System Design**

This step should end with clearly defined components, including what functions each component performs and when they perform it. The following three steps are recommended for good ES design.

* + - Partitioning

Partition the functions into small units needed to be performed by any hardware.

* + - Allocating

The partitioned functions should be allocated to the hardware determined in system definition.

* + - Scheduling

The partitioned functions should be allocated to the involved hardware determined in the System Definition sourcing.

* + - 1. **Component Design**

Software is designed and written for the components accordingly to be implemented in development and prototypes.

* + - 1. **Development**

This step should involve development in the best method determined for the project. Research results in a strong recommendation for testable prototype creation. The following are recommended:

1. Begin by constructing components in the most independent way possible to allow component testing.
2. On the second pass after components are tested the prototype should be assembled so that integration testing can be performed.
3. Prototyping should not be determined complete until all acceptance tests are passed.
   * + 1. **Development Testing**

All three forms of testing are involved in this step.

* + - Components should be tested leading back to component design until component tests pass
    - Integration testing should be performed leading back to the embedded system design.
    - Before moving on to deployment all acceptance tests should pass.
    1. **System Deployment**

The results of development should now be put into a working environment. First the final product should be created from the final prototype then implemented. All defects or shortcomings should be documented here but at this phase development has concluded and these will have to be put into a future project.

* + - 1. **Product Creation**

The end of the development phase should result in either a working prototype or a well-defined method to create the final product that has passed all acceptance tests. This should now be built into the final product that will be released to the users.

* + - 1. **Implementation**

After creation the product should be put into use in a trial setting for final acceptance.

* + - 1. **Acceptance**

Since at this point development has concluded the results of the trial run are documented and if the product is acceptable it is deployed. At this point all user requirements should have been thoroughly tested and satisfied so any shortcomings will be pushed to a future version or project if they are planned.

1. **Requirements Description** 
   1. **Requirements Summary Table**

|  |  |
| --- | --- |
| **Table 3.1 – Requirements Summary Table** | |
| **ID No.** | **Requirement Summary** |
| R001 | Bluetooth connection with Hardware can be tested within the application |
| R002 | When the user does something “wrong” the application notifies them of what they did and how to correct it. |
| R003 | A freefall experiment is available to view acceleration of a falling object |
| R004 | A freefall experiment is available to view speed of a falling object |
| R005 | A freefall experiment is available to view position of a falling object |
| R006 | A freefall experiment poses questions to students about free fall |
| R007 | A freefall experiment explains a procedure for users to follow within the app |
| R008 | The experimental data can be viewed during the experiment. |
| R009 | There’s an option to report issues or give feedback on the application |
| R010 | As data is collected a graph is shown so users can associate data with the plots |
| R011 | If more analysis is required a table of taken data can be viewed by user after data completion |
|  |  |
| R012 | A hardware device provides acceleration data to the application |
| R013 | The hardware can be inserted into a ball and tossed around without breaking. |
| R014 | The hardware device can be used with common physics tools such as carts and springs. |
| R015 | The system has room and is modifiable to include more features later. |
| R016 | The Hardware device is powered so that either batteries can be changed or recharged easily. |
| R017 | The system is “understandable” for a physics student. |
| R018 | The application operates for a full experiment without crashing. |
|  |  |
| USE-1 | After initial installation, the SW shall always be usable by the user, regardless of internet connection. |
| PE-1 | System will remain operable without internet connection |
| PE-2 | Any information requested from databases or tickets sent to CS shall be stored until a connection is reestablished |
| SE-1 | No Personal information shall be accessed or stored by the application |
| SAF-1 | The HW system shall not function if it is not in safe condition |
| ROB-1: | Should the application crash all taken data should be erased and the user returned to the main page with notification |
| ROB-1.1; | If any crash should occur the user should be given the option to submit a feedback report with fillable information |
| MOD-1: | The application should be modifiable to accept further experiments with the same equipment. |
| MOD-2: | The hardware should be modifiable to integrate more experiments later. |

* 1. **System Requirements Specification**

The following document was prepared in the Definition phase of developing the Motion Ball embedded software system.

System Requirements Specification

for

Android Motion Ball

Release 1.0

**Prepared by Alexander Gauf**

**February 21, 2018**

* + 1. **Introduction** 
       1. **Purpose**

This SyRS describes the software functional and nonfunctional requirements for version 1.0 of the Android Motion Ball System. This document is intended to be used by the members of the project team who will implement and verify the correct functioning of the system. Unless otherwise noted all requirements specified here are committed for release 1.0

* + - 1. **Document Conventions**

|  |  |
| --- | --- |
| Term | Description |
| Motion Ball App (MBA) | The Android application for this system |
| Motion Ball HW (MBHW) | The Embedded System hardware for this system |

* + - 1. **Concept**

The motion ball is an educational device used to help students connect the three areas of projectile motion: Theory, Graphs, and Reality. First students will be able to throw this device from one person to the other, or just across a room. As they do this the motion will be plotted on their android smart phone. Finally, students will be able to access the motion data and perform standard lab experiments that are usually done just looking at pictures or videos. On top of this the motion ball offers a consistent source of acceleration data, unlike phone accelerometers that widely vary and are mostly far too inaccurate.

* + - 1. **Problem, Solution Analysis**

|  |  |
| --- | --- |
| Problem | Motion Ball Solution |
| Current projectile motion labs are often non-interactive | A device is provided that can be tossed around |
| Projectile motion equipment can have a very high cost | This simple device costs under $50 and has potential to extend to other experiments. |
| Disconnect between reality and theory | Students will see the motion of the ball plotted right as they throw it, helping them both understand what graphs mean and associate them with the real world. |
| Clunky Data taking | Simple data tables can be accessed right from the user interface on the android smart phone. Students can take time, position, and acceleration data directly from these tables. |
| One use equipment | Unlike standard lab equipment the motion ball has the potential to extend to other experiments such as momentum and energy without any more needed equipment. |
| Phone equipment unreliable | By using a cheap external device, we can guarantee consistent data, where current phone-only applications have a wide range of possible outcomes ranging from none to okay. |

* + - 1. **Scope**

The Android Motion Ball is designed for physics instructors at any level than involves basic to advanced projectile motion. Its purpose is to provide interactive equipment to help students visualize and analyze projectile motion. Release 1.0 can be thrown between students, send acceleration data to an android smart phone, and data can be graphed and viewed by students.

* + 1. **Overall Description**
       1. **Product Perspective**

The Motion Ball system is a two-part system involving an Android Software Application (MBA) and an Arduino Embedded System (MBHW). The MBA accepts data from the MBHW system and can display and plot this data. The MBHW takes accelerometer data and transmits it to the Android software via Bluetooth for analysis. The context diagram in figure 3.1 shows external entities and interfaces for release 1.0. The system has potential to be expanded to other experiments

* + - 1. **User Classes and Characteristics**

|  |  |
| --- | --- |
| **User Class** | **Description** |
| Student User | A Student user is any person who has downloaded the application and wishes to record and analyze projectile motion data. |
| Instructor User | An instructor user is any person who may oversee student users and may have an elevated need to locally test or diagnose the system. |
| Administrator | As of release 1.0 an administrator is any person employed to manage the back end of the software, focusing mainly on customer support. |

* + - 1. **Operating Environment**

OE-1: The Android Motion Ball Application shall run on Android 6.0 and Android 7.0.

OE-2: The Android Motion Ball shall be fully functional offline and does not require any data connection

OE-3: The Android Motion Ball Embedded System shall run Arduino 3.0.

OE-4: MBA shall communicate with the MBHW via Bluetooth.

* + - 1. **Design and Implementation Constraints**

None

* + - 1. **User Documentation**

UD-1: The system shall provide instructions in each section of the application as needed.

* + - 1. **Assumptions and Dependencies**

AS-1: All users will have an android phone

AS-2: Android phone will have functioning Bluetooth.

AS-3: Motion ball HW will be properly powered.

DE-1: Motion Ball App must have the Motion Ball HW to take data.

* + - 1. Context Diagram

#### **Figure 3.1: Context Diagram**

Motion Ball HW

Record Data

Request data collection or display

Display data in user requested form

User

* + 1. **System Features**
       1. **Test Connection**
          1. **Description**

A user can connect to the MBHW from the MBA and display live data values to verify functioning hardware.

* + - * 1. **Details**

|  |  |
| --- | --- |
| Connect.Enable  . Fail | Enable the Bluetooth device on the Android for connection  The app shall proceed to Connect.fail |
| Connect.Discover  . Fail | The application will search for pairable devices  If nothing is found proceed to Connect.Fail |
| Connect.SelectDevice | After a device is found it can be selected from a list |
| **Connect.Request:**  .Fail: | Request Connection to the Selected device  The app shall proceed to Connect.Fail |
| **Connection.Done**:  .Notify: | MBES has connected to MBA  App notifies user of successful connection |
| Connect.GetData  .Fail | Data can be collected and shown to verify functioning device  Proceed to Connect.Fail |
| **Connect.Fail:** | Connection could not be established |
| .Notify: | User is notified that connection failed along with reason for failure |

* + - 1. **Free Fall Experiment**
         1. **Description**

A user can connect to the MBHW from the MBA and display acceleration and time data on a graph or table view.

* + - * 1. **Details**

|  |  |
| --- | --- |
| **Experiment.Explain:** | The Free Fall experiment is explained to users within the app |
| **Experiment.Question:** | Experimental Questions are provided to help with student learning |
| **Experiment.Procedure** | The experimental procedure is explained within the application |
| **Data.Request:**  **.Time:**  **.Live:**  **.Fail:** | Request data collection to begin  The app will get the start time  The user can have the recording run while the app is open to be stopped manually  The app shall proceed to Data.Fail |
| **Data.Done:**  **.Notify:**  **.Fail:** | Collection has completed and the application shall do that following  App notifies user of collection completion  Proceed to Data.Fail. |
| **Data.Graph**  **.Fail** | A Graph of data taken can be seen as the data is taken  Proceed to Data.Fail. |
| **Data.TableView**  **.Fail** | After data is collected users can access the data for analysis purposes  Proceed to Data.Fail. |
| **Data.Fail:**  **.Notify:** | Collection has stopped or did not work for unexpected reasons  User is notified that collection failed along with reason for failure |

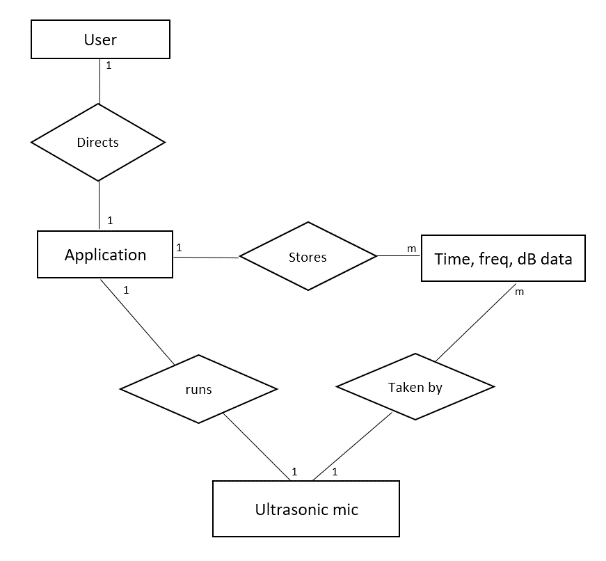
* + - 1. **Extra-functional**
         1. **Description**

Once data has been collected the user can browse files to delete, rename, view, or graph them.

* + - * 1. **Details**

|  |  |
| --- | --- |
| **Update.Check** | User may check for application updates. |
| .Notify: | User is notified that updates are available only after checking |
| .Fail: | User is notified if the app could not check for updates |
| **Update.Apply** | User may apply updates if found in Check |
| .Notify: | User is notified if update is successful |
| .Fail: | User is notified if app cannot be updates with an explanation |
| Feedback.Write: | User may fill out a form to note issues |
| .Close: | User requests to close view and is returned to browse |
| .Fail: | User is notified if feedback form cannot be opened |
| Feedback.Send | User may send issues to be reviewed by system maintenance team |
| .Fail | User is notified if feedback cannot be sent with an explanation |
| ComingSoon.View: | Upcoming features are displayed with “Coming Soon” |
| ComingSoon.Notify | Selecting any feature that is not yet available will notify the user that it is coming soon and where to go for further information |
| Hardware.Battery | Hardware batteries are accessible and replaceable by user |
| Hardware.Switch | The Hardware has a switch to power it on and off to maximize battery time |

* + 1. **Data Requirements**
       1. **Logical Data Model**



Acceleration and time data.

MBHW

MBA

**Figure 3.2 Logical Data Model**

* + - 1. **Partial Data Dictionary**

**Table 3.2 – Partial Data Dictionary**

|  |  |  |
| --- | --- | --- |
| **Data Element** | **Description** | **Composition or Data Type** |
| Time Data | Data taken by MBHW for time values from start | Numeric, Double |
| Acceleration Data | Data taken by MBHW for acceleration values at each time. | Numeric, Double Array “{Ax,Ay,Az” |
| Acceleration magnitude | Transformation of raw acceleration data. | Numeric, Double  “Amag=sqrt(Ax^2+Ay^2+Az^2)” |
| Speed | Transformation of raw acceleration data | Numeric, Double  “V(i) = V(i-1)+Ax\*(t(i)-t(i-1))” |
| Speed | Transformation of raw acceleration data and speed data | Numeric, Double  “X(i) = X(i-1)+V(i-1)\* (t(i)-t(i-1))”+0.5\*Ax\*(t(i)-t(i-1))^2” |

* + - 1. **Reports**

No Reports are generated by this application.

* + - 1. **Data Integrity, Retention, Disposal**

|  |  |
| --- | --- |
| **ID** | **Description** |
| DR-1 | The application shall delete all taken data upon closure, nothing is stored by the application. |
|  |  |

* + 1. **External Interface Requirements**
       1. **User Interfaces**

|  |  |
| --- | --- |
| **ID** | **Description** |
| UI-1: | The user interface will be displayed using an android device. |
| UI-2: | The interface shall be completely useable and navigable by touch. |

* + - 1. **Software Interfaces**

|  |  |
| --- | --- |
| **ID** | **Description** |
| SI-1: | Android Phone |
| SI-1.1: | Motion Ball app shall accept data from the motion ball device as required. |
| SI-2: | Update |
| SI-2.1: | The system shall interface with an update server to apply updates, fixes, and later releases as required. |

* + - 1. **Hardware Interfaces**

|  |  |
| --- | --- |
| **ID** | **Description** |
| HI-1: | Motion Ball Embedded System MBHW |
| HI-1.1: | The system shall run the MBHW to take and display data as requested by user |
| HI-1.2: | The system shall poll the MBHW to ensure proper compatibility and working condition. |

* + - 1. **Communication Interfaces**

|  |  |
| --- | --- |
| **ID** | **Description** |
| CI-1: | User notifications |
| CI-1.1: | The user shall be notified of insufficient memory errors, both RAM and HD. |
| CI-1.2 | The user shall be notified with explanation for all known possible user errors, such as improper procedure |

* + 1. **Software Quality Attributes**
       1. **Usability**

|  |  |  |
| --- | --- | --- |
| **ID** | **Description** | **Rationale** |
| USE-1 | After initial installation, the SW shall always be usable by the user, regardless of internet connection. | System likely to be used outside of service data areas, often happens in classrooms |
|  |  |  |

* + - 1. **Performance**

|  |  |  |
| --- | --- | --- |
| **ID** | **Description** | **Rationale** |
| PE-1 | System will remain operable without internet connection | See USE-1 |
| PE-2 | Any information requested from databases or tickets sent to CS shall be stored until a connection is reestablished | See USE-1,  For the user to avoid inconvenience |

* + - 1. **Security**

|  |  |  |
| --- | --- | --- |
| **ID** | **Description** | **Rationale** |
| SE-1 | No Personal information shall be accessed or stored by the application | Bluetooth connection vulnerable to security concerns, no sensitive data should be accessible through application |

* + - 1. **Safety**

|  |  |  |
| --- | --- | --- |
| **ID** | **Description** | **Rationale** |
| SAF-1 | The HW system shall not function if it is not in safe condition | Avoid damage to persons or phone from improper hardware |

* + - 1. **Availability**

|  |  |  |
| --- | --- | --- |
| **ID** | **Description** | **Rationale** |
| AVL-1 | See USE-1 | Usability overlaps availability here |

* + - 1. **Robustness**

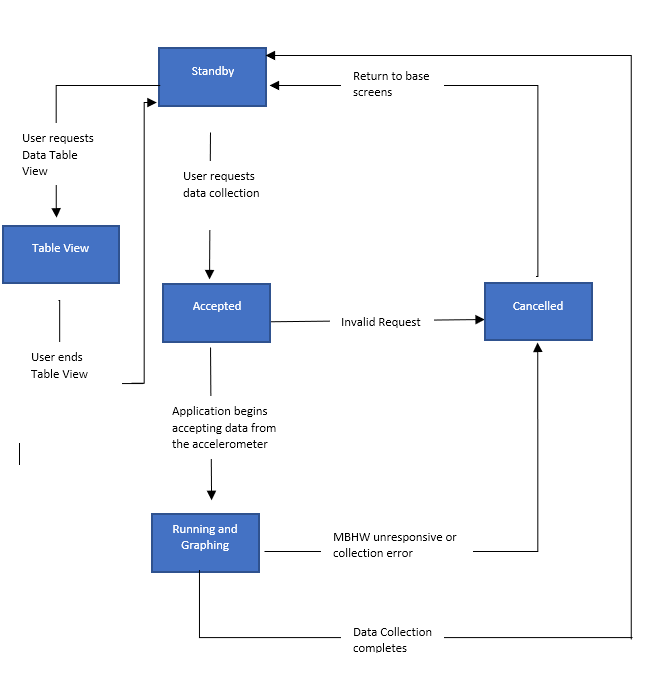
|  |  |  |
| --- | --- | --- |
| **ID** | **Description** | **Rationale** |
| ROB-1: | Should the application crash all taken data should be erased and the user returned to the main page with notification | Avoid data issues and provide user explanation |
| ROB-1.1; | If any crash should occur the user should be given the option to submit a feedback report with fillable information | User quality |

* + - 1. **Modifiability**

|  |  |  |
| --- | --- | --- |
| **ID** | **Description** | **Rationale** |
| MOD-1: | The application should be modifiable to accept further experiments with the same equipment. | Device is meant to be a cheap alternative to multiple experiments. |
| MOD-2: | The hardware should be modifiable to integrate more experiments later. | Device is meant to be a cheap alternative to multiple experiments. |

* + 1. **Analysis Models**
       1. **State Transition Diagram**

**Figure 3.2- State Transition Diagram.**



1. **Design Description**

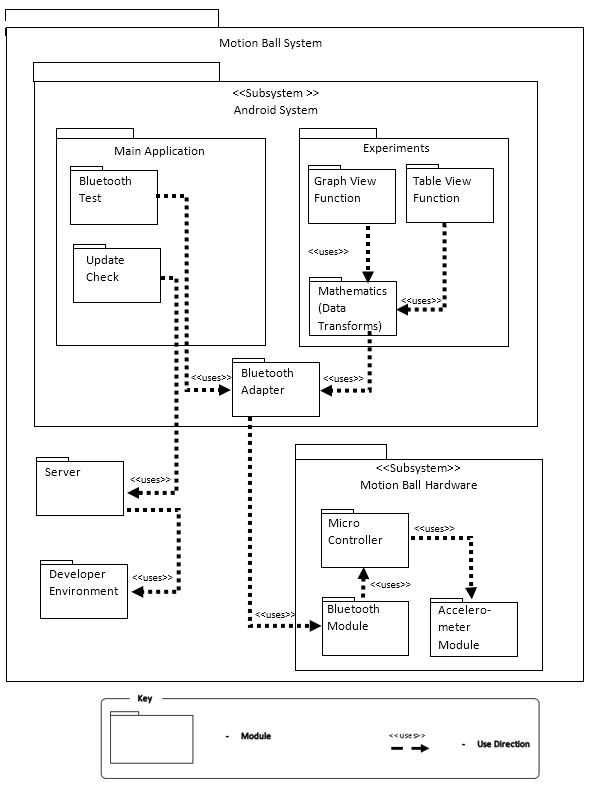
Since this system involves both a Software Application and Embedded software device the Design is presented in four sections. The software system design of this project is presented with two views. First a decomposition and uses view to show module breakdown and dependencies, then a Component and connector plus allocation view to further describe the full system and interfaces. Next to show the physical design a circuit diagram and graphic of the device will be presented. These four views combined should provide an accurate and thorough description of the full system for use in further development.

* 1. **Decomposition and Uses view**

The following diagram shows a high-level description of all modules shown by the folder icons. This is combined with a uses view denoted by the directional dotted <<uses>> arrows. Any module contained within another module is a submodule of the outer one up to the full Motion Ball system.

This view shows that the system is broken into two major subsystems that communicate through their respective Bluetooth modules, the Bluetooth adaptor of the Android system, and the HC-05 Bluetooth module for the embedded system. A specific note from the use view of this design is that the Android Bluetooth module uses the MBHW Bluetooth module to cause the microcontroller to get data from the accelerometer module. Thus, without working Bluetooth communication the system would not function, showing that the Bluetooth modules were very highly ranked in design considerations.

The component and connection plus allocation view of section 5.2 will show these dependencies more thoroughly.

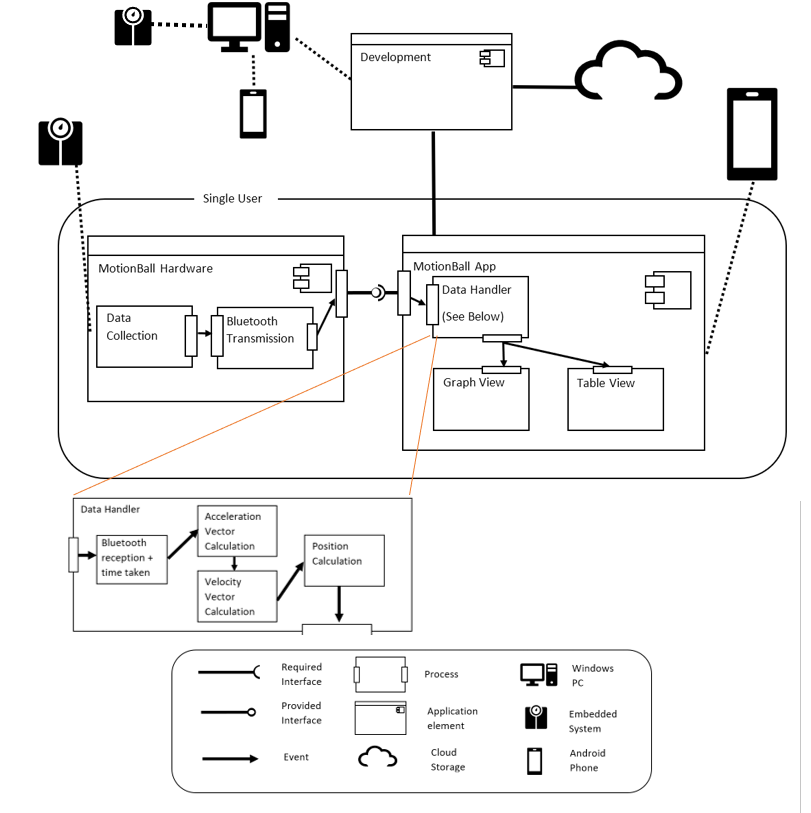
**Figure 4.1- Decomposition and Uses View**

* 1. **Component and Connector plus Allocation Diagram**

The following diagram shows a slightly more detailed view of the components, connections, event flow, and allocation of the MotionBall system. A couple important points to note from this diagram are:

1. The motion ball app requires a connection to the hardware Bluetooth transmission for the Data handler to function.
2. Modifiability (Requirement 013) was maximized in this design by allocating all data mathematics to the data handler in the android application. By making the hardware only transmit raw data we allow for other features to be added later that may transform this raw data in different ways. As of right now only one experiment is implemented which periodically takes raw data from Bluetooth, adds time values to the data, then transforms it to acceleration vectors, velocity vectors, and positions before sending it out to the Graph View. Then after the Graph View has been completed a table view can call the data from the graph and display it.
3. Also notice that the call for updates currently directly connects to the developer’s application to check version compatibility on the google drive cloud storage. If this system were to go to a larger scale a version controller would need to be created and implemented to separate the development environment and versions from the user’s stable version.
4. Finally note that communications form the Hardware to the Application are one directional. As soon as the connection is established data is taken and sent by the Hardware, it is up to the data handler to choose which data to keep and to analyze it. This is another architectural choice to maximize performance of the embedded system and modifiability of the Android App so that it doesn’t have to wait for input or handle establishing communication with the Android application.

**Figure 4.2 – Component and Connector Plus Allocation View**



**4.3 Circuit Diagram**

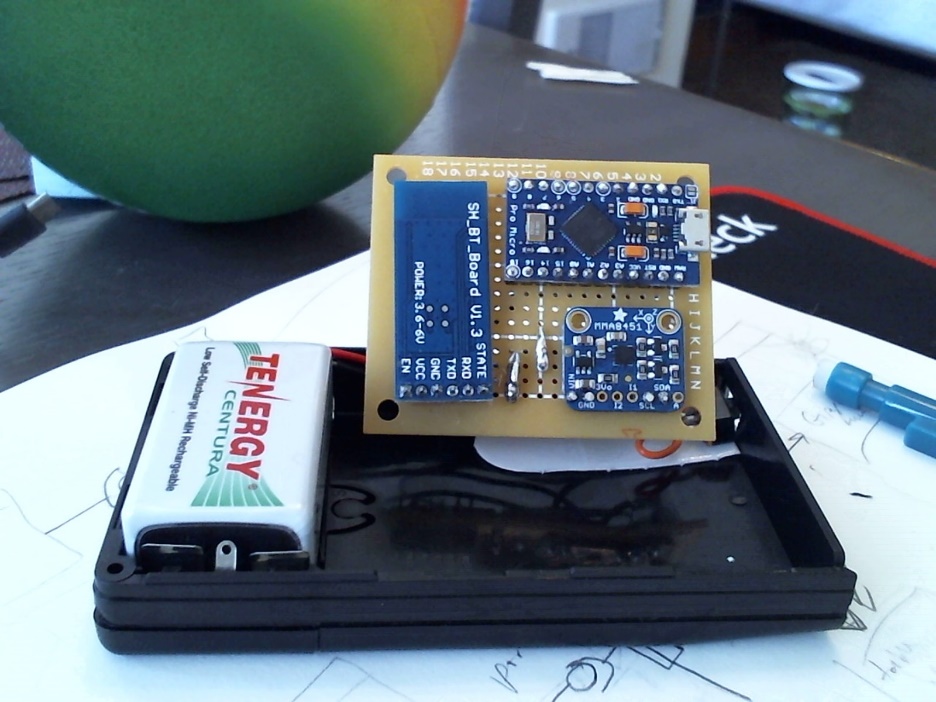
The following diagram shows the circuit inside the embedded device. A couple notes about this device include:

1. The Arduino Pro Micro is currently powered by 9V battery which leads to high energy loss since both components only pull from the 5V regulator. This leads to lower battery life but testing still confirmed it working for more than long enough for a lab procedure.
2. About 30% of the microcontrollers capacity is taken up by the software and device in this design, leaving room for further additions. Also, many data pins are also still available for other features; However, the current case may need to be upgraded if larger hardware modules were to be added.
3. Since so little space is used a smaller microcontroller may be possible but the pro micro was chosen for ease of modifiability and ease of interfacing with Bluetooth.
4. The Bluetooth module chosen (HC-05) is not a low-energy Bluetooth device. This makes this hardware unable to interface with iOS devices that only accept approved low energy Bluetooth interfaces. This choice was made to reduce cost of the system at the sacrifice of Portability. In the end of requirements gathering portability to iOS was deemed unnecessary.

### **Figure 4.3 – Circuit Diagram**

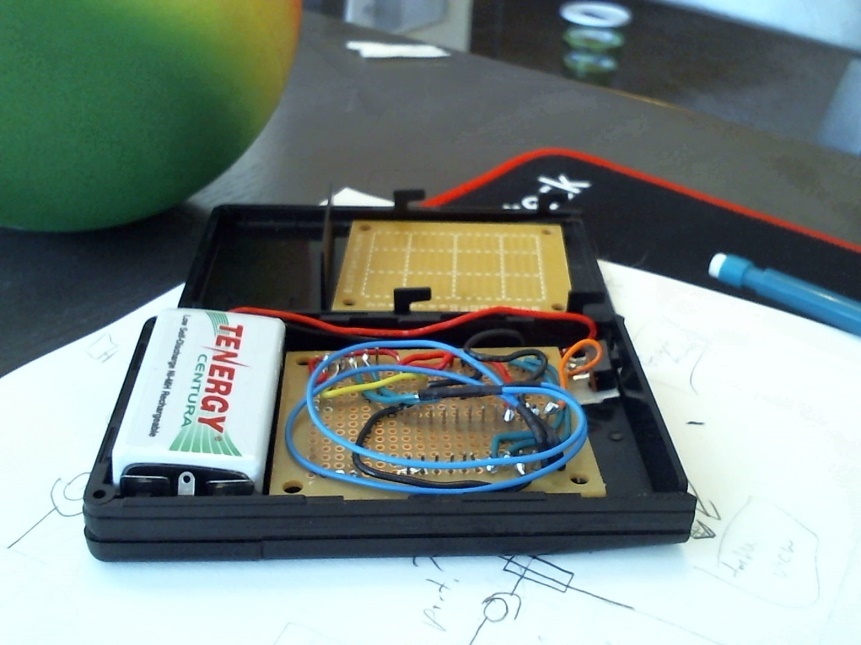
* 1. **Case and Final Design**

To finally realize this project the system had to be assembled in such a way as to be used with multiple other parts including a ball, carts and others. This finally assembly will be shown in the following figures. Since this project focuses on the software this will be only lightly discussed.



### **Figure 4.4 – Hardware Open 1**

The circuit is shown here with case open. The accelerometer is in the bottom right, Arduino Pro Micro on the top right, and Bluetooth HC-05 on the left. All powered by a rechargeable 9V battery.



### **Figure 4.5 – Hardware Open 2**

Shown here is the backside of the circuit board before assembly. The proto board shown is screwed over the wiring for protection before final assembly.

### **Figure 4.6 – Hardware Closed**

The final assembly is shown here showing the power switch at the front right of the system.



### **Figure 4.7 – Battery Compartment**

Note also design included a replaceable battery compartment to satisfy requirement 010.

### **Figure 4.8 – Peripherals**

This device fits into a couple of different devices to satisfy experiments as shown in the images below.

****

1. **Implementation**

The implementation files for this system involve an android application and Arduino software program. This section lists these two implementation files and the files for the development programs.

* 1. **Organization of source file structure**
     1. **Application Files**

The application files are broken into the android application, gradle build files, and signed APK for release 1.0. The full list of files can be found in section 5.2. The entire program was written in Android studio using Java.

* + 1. **Arduino Files**

The Arduino file is one file written in C for use on the Android Pro Micro.

* 1. **Reference list of files**

For this list gradle and other android studio common files were not detailed, these are shaded in grey. Only details specific to the application were included.

* + 1. **Android Application**

1. MotionBallApplication
   1. .gradle
   2. .idea
   3. app
      1. build
      2. libs
         1. GraphView-4.2.1.jar
      3. src
         1. androidTest
         2. main
            1. java

com

motionball

motionball

BluetoothConnect.java

BluetoothConnectionService.java

ComingSoonFragment.java

CustomListAdapter.java

DataViewActivity.java

DeviceListAdapter.java

freeFall\_DynamicGraph.java

FreeFallActivity.java

MainActivity.java

NotSupported.java

UpdateDialogueFragment.java

XYValue.java

* + - * 1. res

drawable

drawable-v24

layout

activity\_bluetooth\_connect

activity\_data\_view

activity\_free\_fall\_dynamic\_graph

activity\_freefall

activity\_main

device\_adapter\_view

listview\_row

mipmap-anydpi-v26

mipmap-hdpi

mipmap-xhdpi

mipmap-xxhdpi

mipmap-xxxhdpi

values

colors

strings

styles

* + - * 1. AndroidManifest.xml
      1. test
  1. build
  2. gradle
  3. .gitignore
  4. AStudioProjects.iml
  5. gradle.properties
  6. gradlew
  7. gradlew.bat
  8. local.properties
  9. MotionBallApplication.iml
  10. settings.gradle
      1. **Arduino Code**

ArduinoProMicroCode.ino

1. **Test and Integration**

This project utilized test driven development. Each of these test cases was written before design of components, integration, and acceptance and then design was done to fit these tests.

Note that some information was left off test cases or integrated with other parts to shorten them including who ran the test, when it was created, when it was ran and test data. Any of these that were important to note were included in the “Remarks” section.

* 1. **Test Suite IDs**

|  |  |  |
| --- | --- | --- |
| **ID** | **Type** | **Description** |
| TS000 | Acceptance | Requirements Acceptance Tests |
| TS001 | Unit/Component | Android App Unit and Component Tests |
| TS002 | Unit/Component | Arduino Hardware Unit and Component Tests |
| TS003 | Integration | Android App Integration Tests |
| TS004 | Integration | Arduino Hardware Integration Tests |
| TS005 | Integration | System Integration Tests |
| TS006 | Acceptance | Android App Acceptance Tests |
| TS007 | Acceptance | Arduino Hardware Acceptance Tests |
| TS008 | Acceptance | System Acceptance Tests |

**Definitions**

Acceptance - Subjective user satisfaction tests.

Unit - Individual function tests with no interface to other components.

Component - Unit tests using data from other system components.

Integration - Tests involving combining multiple units or components.

* 1. **Text Cases**
     1. **TS000 – Requirements Acceptance Tests.**

|  |  |
| --- | --- |
| Test Suite ID | TS000 |
| Test Case ID | TC001 |
| Test Type | Acceptance |
| Test Case Summary | Will the Android Application defined in description satisfy user needs. |
| Related Requirement | R001-R011, R015, R016,R017, USE-1,PE-1,2, SE-1, ROB-1, MOD-1,2 |
| Prerequisites | 1. System Requirements Specification Version 1 completed. |
| Test Procedure | 1. Interview possible users about satisfaction of Android application needs. |
| Expected Result | 1. All needs are addressed in the system requirements specification. |
| Actual Result | 1. Additional needs found and added to requirements specification as R017 and R018 related to quality. 2. Second repeat of test passes. |
| Status | Pass |
| Remarks |  |

|  |  |
| --- | --- |
| Test Suite ID | TS000 |
| Test Case ID | TC002 |
| Test Type | Acceptance |
| Test Case Summary | Will the Embedded Software System defined in description satisfy user needs. |
| Related Requirement | R001, R012-R016. PE-1, SAF-1, MOD-2 |
| Prerequisites | 1. System Requirements Specification Version 1 completed. |
| Test Procedure | 1. Interview possible users about satisfaction of hardware system needs. |
| Expected Result | 1. All needs are satisfied by defined hardware system. |
| Actual Result | 1. Additional hardware needs found and added as R014 for battery changeability. 2. Second run of this test passes. |
| Status | Pass |
| Remarks |  |

|  |  |
| --- | --- |
| Test Suite ID | TS000 |
| Test Case ID | TC003 |
| Test Type | Acceptance |
| Test Case Summary | Will the system defined in description satisfy user needs. |
| Related Requirement | All |
| Prerequisites | 1. System Requirements Specification Version 1 completed. |
| Test Procedure | 1. Interview possible users about satisfaction of general system needs. 2. Check status of TC001 and TC002. |
| Expected Result | 1. All needs are satisfied by the defined system. 2. TC001 and TC002 pass. |
| Actual Result | 1. Some needs not satisfied in first iteration. 2. TC001, TC002 and this Test passed second iteration. |
| Status | Pass |
| Remarks | Two iterations performed of these acceptance tests. |

* + 1. **TS001 -Android App Unit and Component Tests**

|  |  |
| --- | --- |
| Test Suite ID | TS001 |
| Test Case ID | TC004 |
| Test Type | Unit |
| Test Case Summary | Application opens to main view successfully. |
| Related Requirement | R018 |
| Prerequisites | 1. App is downloaded and installed on Android phone |
| Test Procedure | 1. Open android to app selection screen 2. Tap the MotionBall app icon |
| Expected Result | 1. App Opens and Displays Main Activity properly |
| Actual Result | 1. Main activity properly displayed |
| Status | Pass |
| Remarks | May want to add a test for repetition of this test case |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS001 |
| Test Case ID | TC005 |
| Test Type | Unit |
| Test Case Summary | All “Coming Soon” buttons lead to a “Coming soon” dialog box that explains that the feature is not yet implemented. |
| Related Requirement | R015, MOD-1 |
| Prerequisites | 1. App is downloaded and installed on Android phone 2. App is opened to main activity |
| Test Procedure | 1. Select a button with “coming soon” in its text 2. Repeat for all buttons with “coming soon” |
| Expected Result | 1. Each button should inform the user that the feature is not yet implemented |
| Actual Result | 1. 4 Buttons with coming soon label 2. Each showed dialog “More features are in the works, visit our website for more information” |
| Status | Fail |
| Remarks | Buttons do not specifically say the feature they tried does not exist, but does inform that it may be coming soon |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS001 |
| Test Case ID | TC006 |
| Test Type | Unit |
| Test Case Summary | The “check for updates” button properly informs user if there is an available update for the application |
| Related Requirement | R015, MOD-1 |
| Prerequisites | 1. App is downloaded and installed on Android phone 2. App is open to main activity screen |
| Test Procedure | 1. Tap “check for updates” button |
| Expected Result | 1. A dialogue opens informing the user if there is an update or not |
| Actual Result | 1. Popup dialogue states “No available updates” |
| Status | Pass |
| Remarks |  |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS001 |
| Test Case ID | TC007 |
| Test Type | Unit |
| Test Case Summary | The “Report Issue” button opens a dialog box with a text box |
| Related Requirement | R009 |
| Prerequisites | 1. App is downloaded and installed on Android phone 2. App is open to main activity screen |
| Test Procedure | 1. Tap “Report Issue” button |
| Expected Result | 1. A dialogue box opens with a text box |
| Actual Result | 1. Dialogue box is displayed with a text box |
| Status | Pass |
| Remarks |  |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS001 |
| Test Case ID | TC008 |
| Test Type | Unit |
| Test Case Summary | Text entered in the “Report Issue” dialogue box can be sent |
| Related Requirement | R009 |
| Prerequisites | 1. App is downloaded and installed on Android phone 2. App is open to main activity screen |
| Test Procedure | 1. Tap “Report Issue” button 2. Enter text in box 3. Send text. 4. Repeat with cases “no text”, numbers, text. |
| Expected Result | 1. Report dialogue box closes without issue for all inputs |
| Actual Result | 1. Box closed and could be reopened without issue |
| Status | Pass |
| Remarks |  |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS001 |
| Test Case ID | TC009 |
| Test Type | Unit |
| Test Case Summary | A button exists to enable and disable the android devices Bluetooth. |
| Related Requirement | R001 |
| Prerequisites | 1. App is downloaded and installed on Android phone |
| Test Procedure | 1. Locate button for enable/disable Bluetooth within application |
| Expected Result | 1. Button exists |
| Actual Result | 1. Button located in Bluetooth connection screen. |
| Status | Pass |
| Remarks |  |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TC001 |
| Test Case ID | TC010 |
| Test Type | Component |
| Test Case Summary | The Enable/Disable Bluetooth button properly disables the Bluetooth when it is on or turns it on if Bluetooth is off. |
| Related Requirement | R001 |
| Prerequisites | 1. Application is installed and opened. 2. Application is opened to Bluetooth test screen. |
| Test Procedure | 1. Check that Bluetooth is enabled or enable it from device. 2. Press Enable/Disable button. 3. Check that Bluetooth is disabled or disable it from device. 4. Press Enable/Disable button. |
| Expected Result | 1. If Bluetooth was enabled, then Bluetooth is disabled. 2. If Bluetooth was disabled, then Bluetooth is enabled. |
| Actual Result | 1. Bluetooth enabled properly when off with toast confirmation. 2. Bluetooth disabled properly when on with toast confirmation. |
| Status | Pass |
| Remarks | Toast notification not explicitly required but confirms the action well. |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS001 |
| Test Case ID | TC011 |
| Test Type | Unit |
| Test Case Summary | A method exists on Bluetooth Test page to discover new pairable Bluetooth devices |
| Related Requirement | R001 |
| Prerequisites | 1. App is downloaded and installed on Android phone 2. Bluetooth test page is open |
| Test Procedure | 1. Locate method to discover new devices |
| Expected Result | 1. A method is found. |
| Actual Result | 1. “Discover” button enables Bluetooth discovery |
| Status | Pass |
| Remarks |  |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS001 |
| Test Case ID | TC012 |
| Test Type | Unit |
| Test Case Summary | The Bluetooth discovery on Bluetooth Test page can find other devices. |
| Related Requirement | R001 |
| Prerequisites | 1. App is downloaded and installed on Android phone. 2. Bluetooth test page is open. 3. Bluetooth is enabled. |
| Test Procedure | 1. Locate “Discover” button. 2. Tap it. 3. Wait for devices to be discovered. |
| Test Data | NA |
| Expected Result | 1. Discovered devices are shown to user. |
| Actual Result | 1. A list at the bottom of the screen is populated with pairable devices |
| Status | Pass |
| Remarks |  |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS001 |
| Test Case ID | TC013 |
| Test Type | Unit |
| Test Case Summary | Once a device is discovered it can be selected to pair |
| Related Requirement | R001 |
| Prerequisites | 1. App is downloaded and installed on Android phone. 2. Bluetooth test page is open. 3. Bluetooth is enabled. 4. A device has been discovered. |
| Test Procedure | 1. Locate a test paired device on screen. 2. Select it. 3. Verify it has been paired through phone settings. |
| Test Data | NA |
| Expected Result | 1. Selecting the device begins the pairing process. |
| Actual Result | 1. Selecting a device showed a “paired successfully” toast. 2. Device showed in Android device settings as paired. |
| Status | Pass |
| Remarks | After pairing there was no visible notification for exactly which device was paired, may hurt quality. |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS001 |
| Test Case ID | TC014 |
| Test Type | Unit |
| Test Case Summary | Once a device is paired it a connection can be established with it |
| Related Requirement | R001 |
| Prerequisites | 1. App is downloaded and installed on Android phone. 2. Bluetooth test page is open. 3. Bluetooth is enabled. 4. A device has been discovered. 5. A device has been paired. |
| Test Procedure | 1. Locate method of connection. 2. Start connection. |
| Expected Result | 1. Connection with a paired device is established. |
| Actual Result | 1. Progress box appeared showing “Connecting” 2. Toast displayed showing “connected successfully” |
| Status | Pass |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS001 |
| Test Case ID | TC015 |
| Test Type | Unit |
| Test Case Summary | “Discover” button on Graph View page successfully finds and displays Bluetooth devices |
| Related Requirement | R001 |
| Prerequisites | 1. App is downloaded and installed on Android phone. 2. App is opened to Graph View page. 3. Bluetooth enabled on device. |
| Test Procedure | 1. Tap “Discover” button on Graph View page. |
| Expected Result | 1. Bluetooth devices are displayed to user. |
| Actual Result | 1. Bluetooth devices displayed in list at bottom of page. |
| Status | Pass |
| Remarks | Can only view two devices at a time, may be problematic in large groups. |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS001 |
| Test Case ID | TC016 |
| Test Type | Unit |
| Test Case Summary | The “Connect” button on graphing page properly connects to a selected discovered device |
| Related Requirement | R001 |
| Prerequisites | 1. App is downloaded and installed on Android phone. 2. App is opened to Graph View page. 3. Bluetooth enabled on device. 4. Device has been discovered and selected. |
| Test Procedure | 1. Locate method of connection. 2. Connect. |
| Expected Result | 1. Connection is established with device |
| Actual Result | 1. Progress Dialog shows “Connecting” 2. Toast pops up and shows “Connected Successfully” |
| Status | Pass |
| Remarks | No visual indicator which item has been connected to, consider highlighting the device when connected. |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS001 |
| Test Case ID | TC017 |
| Test Type | Unit |
| Test Case Summary | “Graph” button properly starts the data analysis function. |
| Related Requirement | R008 |
| Prerequisites | 1. App is downloaded and installed on Android phone. 2. App is opened to Graph View page. |
| Test Procedure | 1. Locate and select the “Graph” button. |
| Expected Result | 1. Data collection function should begin. |
| Actual Result | 1. With no data a notification was shown saying “No data available to graph”. 2. After inserting a test data stub, the graph began plotting points. |
| Status | Pass |
| Remarks | Had to create a test data sample for this test. |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS001 |
| Test Case ID | TC018 |
| Test Type | Unit |
| Test Case Summary | “Graph” periodically updates the Graph View with values |
| Related Requirement | R008, R010 |
| Prerequisites | 1. App is downloaded and installed on Android phone. 2. App is opened to Graph View page. |
| Test Procedure | 1. Locate and select the graph button |
| Expected Result | 1. Graph should be periodically populated with data points. |
| Actual Result | 1. Data points are shown on graph. |
| Status | Pass |
| Remarks | There were gaps between points, not a consistent time interval for plotting function. Noted as defect. |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS001 |
| Test Case ID | TC019 |
| Test Type | Unit |
| Test Case Summary | The “Stop” button on graphing page successfully stops data collection |
| Related Requirement | R008, R010, R011 |
| Prerequisites | 1. App is downloaded and installed on Android phone. 2. App is opened to Graph View page. |
| Test Procedure | 1. Start data collection by pressing “Graph” button. 2. Attempt to stop data collection with “Stop” button. |
| Expected Result | 1. Data collection stops. 2. Data can still be viewed. |
| Actual Result | 1. Data collection stopped. 2. Data was still shown on graph. |
| Status | Pass |
| Remarks |  |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS001 |
| Test Case ID | TC020 |
| Test Type | Unit |
| Test Case Summary | “Clear” button on graphing page completely removes all taken data values. |
| Related Requirement | R010 |
| Prerequisites | 1. App is downloaded and installed on Android phone. 2. App is opened to Graph View page. 3. Data is currently displayed on graph. |
| Test Procedure | 1. Tap “Clear” button and observe graph. |
| Expected Result | 1. All data points are erased. 2. all data arrays cleared. |
| Actual Result | 1. Data points were cleared. 2. Data remains in arrays, re-graphing put all points back. |
| Status | Fail |
| Remarks | Created defect log and new test case for clearing data. |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS001 |
| Test Case ID | TC021 |
| Test Type | Unit |
| Test Case Summary | “Clear” button on graphing page completely removes all taken data values from data arrays. |
| Related Requirement | R010 |
| Prerequisites | 1. App is downloaded and installed on Android phone. 2. App is opened to Graph View page. 3. Data is currently displayed on graph. |
| Test Procedure | 1. Tap “Clear” button. 2. Check data arrays for values. |
| Expected Result | 1. All data arrays cleared. |
| Actual Result | 1. All data was cleared. |
| Status | Pass |
| Remarks | Test case created after part of TC020 failed. |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS003 |
| Test Case ID | TC022 |
| Test Type | Unit |
| Test Case Summary | Table view successfully displays a table of data |
| Related Requirement | R011 |
| Prerequisites | 1. App is downloaded and installed on Android phone. 2. App is opened to Graph View page. 3. Data is currently displayed on graph. |
| Test Procedure | 1. Select “Table” button and observe. |
| Expected Result | 1. All data values should be displayed in a full table. |
| Actual Result | 1. All data values shown. 2. If no data was available app displayed a toast saying “Take data first with the Graph button” |
| Status | Pass |
| Remarks |  |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS001 |
| Test Case ID | TC023 |
| Test Type | Unit |
| Test Case Summary | For the experiment Questions can be displayed to students. |
| Related Requirement | R006 |
| Prerequisites | 1. App is downloaded and installed on Android phone |
| Test Procedure | 1. Locate Questions that should be displayable to students. |
| Expected Result | 1. Questions found. |
| Actual Result | 1. “Questions” button on experiment page displays questions for the students. |
| Status | Pass |
| Remarks |  |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS001 |
| Test Case ID | TC024 |
| Test Type | Unit |
| Test Case Summary | For the experiment there is an explanation of the procedure to follow to take data. |
| Related Requirement | R007 |
| Prerequisites | 1. App is downloaded and installed on Android phone. |
| Test Procedure | 1. Locate Procedure for freefall experiment. |
| Expected Result | 1. Procedure for freefall experiment is located and displayed. |
| Actual Result | 1. “Procedure” button brings up dialog with explanation how to proceed. |
| Status | Pass |
| Remarks | Perhaps include diagrams in this procedure. Currently all text. |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS001 |
| Test Case ID | TC025 |
| Test Type | Unit |
| Test Case Summary | The Free Fall Explanation page describes the experiment to be performed. |
| Related Requirement |  |
| Prerequisites | 1. App is downloaded and installed on Android phone. |
| Test Procedure | 1. Locate a description of the freefall experiment within the experiment pages. |
| Expected Result | 1. A description of the experiment is found. |
| Actual Result | 1. Text is displayed on the main experiment page that describes the experiment being done. |
| Status | Pass |
| Remarks |  |
| Test Environment | * Android OS |

**6.2.2. TS002 - Arduino Hardware Unit and Component Tests**

|  |  |
| --- | --- |
| Test Suite ID | TS002 |
| Test Case ID | TC026 |
| Test Type | Unit |
| Test Case Summary | An accelerometer provides raw 3D acceleration data. |
| Related Requirement | R012 |
| Prerequisites | None |
| Test Procedure | 1. Locate a suitable accelerometer. |
| Expected Result | 1. An accelerometer has been located. |
| Actual Result | 1. MMA8451 module selected for design. |
| Status | Pass |
| Remarks |  |
| Test Environment | Design |

|  |  |
| --- | --- |
| Test Suite ID | TS002 |
| Test Case ID | TC027 |
| Test Type | Unit |
| Test Case Summary | The accelerometer properly functions when given enough power. |
| Related Requirement | R012 |
| Prerequisites | 1. Accelerometer hooked up to a test system. 2. Serial Monitor is open in Arduino IDE. |
| Test Procedure | 1. Create/Find test code for MMA8451 2. Upload test code to test system. 3. Observe Serial monitor. |
| Expected Result | 1. Acceleration values are displayed in Serial Monitor. |
| Actual Result | 1. Acceleration values were displayed with other orientation values in Serial monitor. |
| Status | Pass |
| Remarks | MMA has extra orientation functions that won’t be needed. |
| Test Environment | * Arduino IDE |

|  |  |
| --- | --- |
| Test Suite ID | TS002 |
| Test Case ID | TC028 |
| Test Type | Unit |
| Test Case Summary | A microcontroller exists that can interface with other components. |
| Related Requirement | R012, MOD-2 |
| Prerequisites | None |
| Test Procedure | 1. Locate a suitable microcontroller. |
| Expected Result | 1. Microcontroller that supports modifiability found. |
| Actual Result | 1. Arduino Pro Micro located in design. |
| Status | Pass |
| Remarks |  |
| Test Environment | Design |

|  |  |
| --- | --- |
| Test Suite ID | TS002 |
| Test Case ID | TC029 |
| Test Type | Unit |
| Test Case Summary | The microcontroller properly functions when given enough power. |
| Related Requirement | R012 |
| Prerequisites | 1. Microcontroller available to be tested. |
| Test Procedure | 1. Locate Pro Micro test code. 2. Upload code to board. 3. Observe Serial Monitor. |
| Expected Result | 1. All pins test to be in working order and usable. |
| Actual Result | 1. LEDs activate properly on all data pins. 2. RX and TX pins properly communicate. 3. All voltage and ground pins in working order. |
| Status | Pass |
| Remarks |  |
| Test Environment | * Arduino IDE. * Design Environment. |

|  |  |
| --- | --- |
| Test Suite ID | TS002 |
| Test Case ID | TC030 |
| Test Type | Unit |
| Test Case Summary | A Bluetooth device exists to communicate with Android Bluetooth devices. |
| Related Requirement | R008 |
| Prerequisites | None |
| Test Procedure | 1. Locate Bluetooth device from design. |
| Expected Result | 1. A Bluetooth device is found with the ability to communicate with Android devices. |
| Actual Result | 1. HC-05 Bluetooth device is located that can communicate with android |
| Status | Pass |
| Remarks | This component will not work with iOS. |
| Test Environment | * Design |

|  |  |
| --- | --- |
| Test Suite ID | TS002 |
| Test Case ID | TC031 |
| Test Type | Unit |
| Test Case Summary | The Bluetooth device properly functions given enough power. |
| Related Requirement | R012 |
| Prerequisites | 1. Bluetooth device available for testing 2. Device connected to test module. |
| Test Procedure | 1. Locate/create test code for Bluetooth module. 2. Upload code to test module. 3. Observe outcome. |
| Expected Result | 1. Some confirmation is given that the Bluetooth is successfully transmitting. |
| Actual Result | 1. Values are shown in Serial monitor for Bluetooth test. 2. One Arduino Pro Micro destroyed attempting to upload code while HC-05 hooked up. |
| Status | Pass |
| Remarks | Must disconnect Bluetooth device before uploading code to the microcontroller, or the microcontroller will be destroyed. |
| Test Environment | * Arduino IDE |

|  |  |
| --- | --- |
| Test Suite ID | TS002 |
| Test Case ID | TC032 |
| Test Type | Component |
| Test Case Summary | Battery can be changed by user in a simple manner. |
| Related Requirement | R010 |
| Prerequisites | System external case available for testing. |
| Test Procedure | 1. Locate battery compartment. 2. Remove battery. 3. Replace battery. |
| Expected Result | 1. Battery replaced without issue. |
| Actual Result | 1. Battery easily removed and replaced from battery compartment. |
| Status | Pass |
| Remarks |  |
| Test Environment | None |

|  |  |
| --- | --- |
| Test Suite ID | TS004 |
| Test Case ID | TC033 |
| Test Type | Component |
| Test Case Summary | Arduino pro micro accepts data from the MMA8451. |
| Related Requirement |  |
| Prerequisites | 1. Arduino hooked up to test device to view Serial Monitor. |
| Test Procedure | 1. Write test code to run MMA and send values to Serial. 2. Upload code to Arduino. 3. Observe Serial Monitor. 4. Move MMA up and down and observe that values change accordingly. 5. Move MMA left and right and observe values change accordingly. 6. Rotate MMA and observe that values change accordingly. |
| Expected Result | 1. Acceleration values are output to Serial monitor. 2. Acceleration values change for all three motions. |
| Actual Result | 1. Acceleration values observed in the form {x,y,z 2. Moving MMA up and down increased/decreased z acceleration. 3. Moving MMA left and right increased/decreased x,y accelerations. 4. Rotating MMA changed all three acceleration values. |
| Status | Pass |
| Remarks |  |
| Test Environment | * Developer Environment. * Arduino IDE. |

|  |  |
| --- | --- |
| Test Suite ID | TS004 |
| Test Case ID | TC034 |
| Test Type | Component |
| Test Case Summary | Arduino pro micro sends data to the HC-05 Bluetooth module. |
| Related Requirement |  |
| Prerequisites | 1. Arduino pro micro hooked up to Bluetooth transmitter. 2. Bluetooth test application available to test transmission. |
| Test Procedure | 1. Write test code to transmit dummy values over Bluetooth hc-05 2. WHILE ARDUINO IS DISCONNECTED, upload the code to the Arduino. 3. Connect Bluetooth device. 4. Run Bluetooth test application. 5. Locate device. 6. Pair with device. 7. Accept data from device. 8. Check that accepted data matches code. |
| Expected Result | 1. HC-05 is found, paired, and connected to successfully. 2. Data matches data emitted by code. |
| Actual Result | 1. HC-05 is found, paired, and connected. 2. Bluetooth test software reads “test” from the connection. 3. “test” matches the dummy value sent from code. |
| Status | Pass |
| Remarks | 1. DO NOT UPLOAD CODE TO THE ARDUINO WHILE IT IS HOOKED UP TO HC-05. One device was lost from this error. 2. HC-05 may need to be changed to identify devices, a room full of HC-05 devices will be hard to work. |
| Test Environment | * Arduino IDE. * Bluetooth test application on Android device. |

**6.2.3. TS003 - Android App Integration Tests**

|  |  |
| --- | --- |
| Test Suite ID | TS003 |
| Test Case ID | TC035 |
| Test Type | Integration |
| Test Case Summary | “Test Connection” on the main screen links to Bluetooth test page successfully |
| Related Requirement | R018 |
| Prerequisites | 1. App is downloaded and installed on Android phone. 2. App is open to the main view. |
| Test Procedure | 1. Locate Bluetooth connection test link. 2. Activate it. |
| Test Data | NA |
| Expected Result | 1. A method to get to Bluetooth test is found. 2. Bluetooth test page is opened without issue. |
| Actual Result | 1. “Test Connection” button found. 2. Bluetooth test page opened properly. |
| Status | Pass |
| Remarks |  |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS003 |
| Test Case ID | TC036 |
| Test Type | Integration |
| Test Case Summary | Main page links to the freefall experiment successfully. |
| Related Requirement | R003-R007, R018 |
| Prerequisites | 1. App is downloaded and installed on Android phone. 2. Main view is open. |
| Test Procedure | 1. Locate Free Fall experiment link. 2. Activate it. |
| Expected Result | 1. A link is found to the Free Fall experiment. 2. Activating the link goes to Free Fall experiment successfully. |
| Actual Result | 1. “Free Fall Experiment” button located 2. Click leads to Free Fall experiment page. |
| Status | Pass |
| Remarks | No title on experiment page, but info implies freefall. |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS003 |
| Test Case ID | TC037 |
| Test Type | Integration |
| Test Case Summary | All actions that return to main from other activities properly load and show the main activity page |
| Related Requirement | R018 |
| Prerequisites | 1. App is downloaded and installed on Android phone |
| Test Procedure | 1. Open to main activity screen. 2. Proceed to “Test Connection” 3. Return. 4. Repeat for Free Fall experiment. |
| Expected Result | 1. After returning main activity is displayed without issue from each section. |
| Actual Result | 1. Main view displayed on return from test. 2. Main view displayed on return from Free Fall Experiment. |
| Status | Pass |
| Remarks |  |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS003 |
| Test Case ID | TC038 |
| Test Type | Integration |
| Test Case Summary | Closing the Bluetooth test page returns to the main activity without issue. |
| Related Requirement | R018 |
| Prerequisites | 1. App is downloaded and installed on Android phone 2. App is opened to Bluetooth connection screen |
| Test Procedure | 1. Close the Bluetooth test page by return button. 2. Close the Bluetooth test page by back button. |
| Expected Result | 1. App should return to main page without issue in both methods. |
| Actual Result | 1. If no actions were taken on page app successfully returned. 2. After establishing any connections app would crash on attempt to close. |
| Status | Fail |
| Remarks | App crashes if any connection actions were created before this test, a new test case should be created for this in the next iteration |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS003 |
| Test Case ID | TC039 |
| Test Type | Integration |
| Test Case Summary | Closing graph view page successfully returns to the freefall explanation page. |
| Related Requirement | R018 |
| Prerequisites | 1. App is downloaded and installed on Android phone 2. App is opened to Graph view page. |
| Test Procedure | 1. From Graph view select return or press back button. |
| Expected Result | 1. App should return to free fall explanation page |
| Actual Result | 1. Application returns to previous page then crashes |
| Status | Fail |
| Remarks | App crashes if any connection actions were created before this test, a new test case should be created for this in the next iteration |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS003 |
| Test Case ID | TC056 |
| Test Type | Integration |
| Test Case Summary | After creating a Bluetooth connection closing the Bluetooth page returns to the main activity without issue. |
| Related Requirement | R018 |
| Prerequisites | 1. App is downloaded and installed on Android phone 2. App is opened to Bluetooth connection screen |
| Test Procedure | 1. From Bluetooth test screen click the “return” button |
| Expected Result | 1. App should return to main page without issue |
| Actual Result | 1. Successfully returned to main page. |
| Status | Pass |
| Remarks | Resolved defect from TC033. |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS003 |
| Test Case ID | TC057 |
| Test Type | Integration |
| Test Case Summary | After creating a Bluetooth connection, closing graph view page successfully returns to the freefall explanation page. |
| Related Requirement | R018 |
| Prerequisites | 1. App is downloaded and installed on Android phone 2. App is opened to Graph view page. |
| Test Procedure | 1. Establish a Bluetooth connection. 2. Select return or press back button. |
| Expected Result | 1. App should return to free fall explanation page |
| Actual Result | 1. Application successfully returned to previous page. |
| Status | Fail |
| Remarks | Resolved defect from TC034. |
| Test Environment | * Android OS |

**6.2.4. TS004 - Arduino Hardware Integration Tests**

|  |  |
| --- | --- |
| Test Suite ID | TS004 |
| Test Case ID | TC040 |
| Test Type | Integration |
| Test Case Summary | Arduino pro micro sends data to the HC-05 Bluetooth module from the MMA8451. |
| Related Requirement | R012-R014 |
| Prerequisites | 1. Entire system is connected and available to be tested. |
| Test Procedure | 1. Upload final code to generate acceleration values and transmit them over Bluetooth. 2. Use Bluetooth test application to verify acceleration values are accepted. |
| Expected Result | 1. Acceleration values are read by test application. |
| Actual Result | 1. Values are read by acceleration beginning with multiple “???” then “{value,value,value{value,value,value{…” |
| Status | Pass |
| Remarks | Transmission always begins with a series of “???”, code ignores these but may cause issues later. |
| Test Environment | * Arduino IDE |

|  |  |
| --- | --- |
| Test Suite ID | TS004 |
| Test Case ID | TC041 |
| Test Type | Integration |
| Test Case Summary | Arduino pro micro sends data to the HC-05 Bluetooth module from the MMA8451 every 0.1 seconds. |
| Related Requirement | R012 |
| Prerequisites | 1. Entire system is connected and available to be tested. |
| Test Procedure | 1. Upload final code with time delay of 100ms to generate acceleration values and transmit them over Bluetooth. 2. Use Bluetooth test application to read 1s of data at a time. 3. Verify 10 values are accepted every 1s. |
| Expected Result | 1. 10 Acceleration values are read by test application every 1s. |
| Actual Result | 1. Values are read by acceleration beginning with multiple “???” then “{value,value,value{value,value,value{…” 2. 10 values are found every 1s of measurement. |
| Status | Pass |
| Remarks |  |
| Test Environment | * Arduino IDE * Bluetooth test application on Android. |

**6.2.5. TS005 - System Integration Tests**

|  |  |
| --- | --- |
| Test Suite ID | TS005 |
| Test Case ID | TC042 |
| Test Case Summary | The Discover buttons can properly identify the hardware device |
| Related Requirement | R001, R012 |
| Prerequisites | 1. App is downloaded and installed on Android phone 2. Embedded system is functional. 3. Embedded system is powered on. |
| Test Procedure | 1. Open Android application to Bluetooth test screen. 2. Enable discovery. 3. Observe if “HC-05” is found. 4. Repeat on Graph View screen. |
| Expected Result | 1. HC-05 is listed on device discovery list. |
| Actual Result | 1. “HC-05” listed on Test. 2. “HC-05” listed on Graph view. |
| Status | Pass |
| Remarks |  |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS005 |
| Test Case ID | TC043 |
| Test Case Summary | Clicking the “Get raw data” button in Android app Bluetooth test shows a set of values from the hardware device. |
| Related Requirement | R001, R012 |
| Prerequisites | 1. App is downloaded and installed on Android phone. 2. Embedded system is functional. 3. Embedded system is powered on. 4. Bluetooth is paired, and a connection is created. |
| Test Procedure | 1. Press “Get raw data” button. |
| Expected Result | 1. Raw data is displayed in the form “{ax,ay,az{…” |
| Actual Result | 1. Data values are shown as “????{ax,ay,az{…” |
| Status | Fail |
| Remarks | Note taken of the question marks, they are neglected in data handler. |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS005 |
| Test Case ID | TC044 |
| Test Type | Integration |
| Test Case Summary | “Graph” button properly starts collecting data from the Arduino system |
| Related Requirement | R012 |
| Prerequisites | 1. App is downloaded and installed on Android phone. 2. Embedded system is functional. 3. Embedded system is powered on. 4. Bluetooth is paired, and a connection is created. |
| Test Procedure | 1. Press “Graph” button on Graph View screen. |
| Expected Result | 1. Data is taken from embedded system and stored. |
| Actual Result | 1. Data is taken and stored in AxArrayList, AyArrayList ,AzArrayList. |
| Status | Pass |
| Remarks |  |
| Test Environment | * Android Studio. |

|  |  |
| --- | --- |
| Test Suite ID | TS005 |
| Test Case ID | TC045 |
| Test Type | Integration |
| Test Case Summary | Graph periodically updates the GraphView with values from the Arduino device |
| Related Requirement | R008, R010, R012 |
| Prerequisites | 1. App is downloaded and installed on Android phone. 2. Embedded system is functional. 3. Embedded system is powered on. 4. Bluetooth is paired, and a connection is created. |
| Test Procedure | 1. Press “Graph” button on Graph View screen. 2. Observe Graph on page. |
| Expected Result | 1. Data is taken from embedded system and shown on graph. |
| Actual Result | 1. Data is taken, and points are shown on graph every 1 second. |
| Status | Pass |
| Remarks | Occasional values are lost, gaps are found in the graph about 1 lost every 2 seconds. |
| Test Environment | * Android OS |

**6.2.6. TS006 - Android App Acceptance Tests**

|  |  |
| --- | --- |
| Test Suite ID | TS006 |
| Test Case ID | TC046 |
| Test Type | Acceptance |
| Test Case Summary | The application is modifiable to accept more interfaces later. |
| Related Requirement | R015, MOD-1. |
| Prerequisites | 1. Android Application is functional and installed. |
| Test Procedure | 1. Open Android Application. 2. Verify methods of modifiability. |
| Expected Result | 1. Room for modification is found. |
| Actual Result | 1. “Coming Soon” button and multiple experiments labelled “Coming soon” are found. 2. Clicking any of them informs the user that they are not yet implemented. |
| Status | Pass |
| Remarks | Was performed by User representative, who is also developer in this case. |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS006 |
| Test Case ID | TC047 |
| Test Type | Acceptance |
| Test Case Summary | The application User Interface is generally understandable to the average user. |
| Related Requirement | R017 |
| Prerequisites | 1. Android Application is functional and installed. |
| Test Procedure | 1. Allow average user representative to use application. 2. Interview user about understanding of various interfaces. |
| Expected Result | 1. User finds the application interfaces understandable. |
| Actual Result | 1. User states application is easy to maneuver. 2. Buttons on Graph View page commented as “a bit small”. |
| Status | Pass |
| Remarks | Run by physics 100 student. |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS006 |
| Test Case ID | TC048 |
| Test Type | Acceptance |
| Test Case Summary | The Bluetooth testing function is understandable to the average user. |
| Related Requirement | R017 |
| Prerequisites | 1. Android Application is functional and installed. |
| Test Procedure | 1. Direct user representative to use Bluetooth test function. 2. Interview user about understanding of Bluetooth test page. |
| Expected Result | 1. User finds the Bluetooth page understandable. |
| Actual Result | 1. User found it understandable. 2. Commented that it was hard to tell if connection was working after clicking list. |
| Status | Pass |
| Remarks | Run by physics 100 student. Next version should include highlighting of connected device. |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS006 |
| Test Case ID | TC049 |
| Test Type | Acceptance |
| Test Case Summary | The Graphing and data view functions of the application are understandable to the average user. |
| Related Requirement | R017 |
| Prerequisites | 1. Android Application is functional and installed. |
| Test Procedure | 1. Direct user representative to use the graphing section. 2. Interview user about understanding of graphing and data view functions. |
| Expected Result | 1. User finds the graphing function understandable. |
| Actual Result | 1. User found graphing understandable. 2. Commented that buttons were out of order since connection had to be done before graphing. 3. Commented that nothing tells them graph is zoomable and scrollable. |
| Status | Fail |
| Remarks | Run by physics 100 student. Buttons and interface reordered.  Next version could possibly add something to say graph is interactive. |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS006 |
| Test Case ID | TC050 |
| Test Type | Acceptance |
| Test Case Summary | The application is fully available offline. |
| Related Requirement | R018, USE-1, PE-1 |
| Prerequisites | 1. Android Application is functional and installed. |
| Test Procedure | 1. With internet connection disabled start application. 2. Verify all app functions work. 3. Close application. 4. With internet connection enabled start application. 5. Disable internet connection with application opened. 6. Verify all app functions still work. |
| Expected Result | 1. App should open and run without issue if no existing connection. 2. App should continue running without issue if connection is disabled. |
| Actual Result | 1. With no internet connection application still fully functional. 2. Application undisrupted when connection was disabled. |
| Status | Pass |
| Remarks |  |
| Test Environment | * Android OS |

|  |  |
| --- | --- |
| Test Suite ID | TS006 |
| Test Case ID | TC051 |
| Test Type | Acceptance |
| Test Case Summary | If any function doesn’t work the user is notified of a reason. |
| Related Requirement | R002, R017, ROB-1 |
| Prerequisites | 1. Android Application is functional and installed. |
| Test Procedure | 1. Try various functions neglecting instructions. |
| Expected Result | 1. Anytime something is done out of order user is notified. 2. Anytime something is not available user is notified. |
| Actual Result | 1. After trying to connect to Bluetooth with no device selected a toast states to pair first. 2. Trying to view table with no available data informs user to take data first with Graph button. 3. Attempting to open any of the “Coming Soon” features notifies user that they are currently unavailable. 4. Attempting to stop data collection when data is not being collected notifies user to start taking data with “Graph” function first. |
| Status | Pass |
| Remarks |  |
| Test Environment | * Android OS |

**6.2.7. TS007 - Arduino Hardware Acceptance Tests**

|  |  |
| --- | --- |
| Test Suite ID | TS007 |
| Test Case ID | TC052 |
| Test Type | Acceptance |
| Test Case Summary | Hardware interfaces well with Ball and is possible to use with usual physics lab tools such as carts and springs. |
| Related Requirement | R013, R014 |
| Prerequisites | 1. Hardware system is assembled and functional. |
| Test Procedure | 1. Attempt to use device with provided Ball. 2. Attempt to use device with other physics materials, such as carts. |
| Expected Result | 1. Hardware device works with ball. 2. Hardware device possible to use with other materials. |
| Actual Result | 1. Hardware device fits snugly and is protected by ball. 2. Hardware device sits on top of some carts, slightly too wide for older models but still useable. |
| Status | Pass |
| Remarks | This passes from ball designed for it. It also works with some devices but requires a “saddle” for newer ones. Next version should probably be 1cm thinner. |
| Test Environment | * Arduino IDE |

|  |  |
| --- | --- |
| Test Suite ID | TS007 |
| Test Case ID | TC053 |
| Test Type | Acceptance |
| Test Case Summary | Hardware use is easy to understand at first glance. |
| Related Requirement | R017 |
| Prerequisites | 1. Hardware system is assembled and functional. |
| Test Procedure | 1. Give hardware to someone who does not know design. 2. Observe if they can figure out how to use it. |
| Expected Result | 1. Test subject can use device successfully. |
| Actual Result | 1. User able to turn on device. 2. User able to insert into ball. 3. User comments that there’s no notification of where or not device is on on the system itself. |
| Status | Pass |
| Remarks | Run by Evan Foley, friend of developer. Perhaps open case to the blinking LEDs inside to show its running. |
| Test Environment | * Arduino IDE |

|  |  |
| --- | --- |
| Test Suite ID | TS007 |
| Test Case ID | TC054 |
| Test Type | Acceptance |
| Test Case Summary | Hardware remains operable without internet connection. |
| Related Requirement | USE-1, PE-2 |
| Prerequisites | 1. Hardware system is assembled and functional. |
| Test Procedure | 1. Verify that the device runs without internet connection or does not require internet connection. |
| Expected Result | 1. Device is functional without internet connection. |
| Actual Result | 1. Device does not use internet connection. |
| Status | Pass |
| Remarks |  |
| Test Environment | * Arduino IDE |

**6.2.8. TS008 - System Acceptance Tests**

|  |  |
| --- | --- |
| Test Suite ID | TS008 |
| Test Case ID | TC053 |
| Test Type | Acceptance |
| Test Case Summary | The system does not function if the hardware is not in safe working condition. |
| Related Requirement | SAF-1 |
| Prerequisites | 1. System is functional. 2. Android application installed and running. |
| Test Procedure | 1. “Break” hardware device by pulling out battery. 2. Verify integral functions stopped. 3. Repeat while data is being taken. |
| Expected Result | 1. Data analysis functions not operable without device. 2. Data analysis functions stop without device. |
| Actual Result | 1. Bluetooth connection cannot be established. 2. Graphing function informs user that device is not available. 3. If connection was already established graphing function stops plotting points. |
| Status | Pass |
| Remarks | The only unsafe condition would be battery failure, this test covers that. |
| Test Environment | * Implementation. |

|  |  |
| --- | --- |
| Test Suite ID | TS008 |
| Test Case ID | TC054 |
| Test Type | Acceptance |
| Test Case Summary | The system can be run for the duration of a lab exercise without issue (3 hours). |
| Related Requirement | R018, USE-1, ROB-1 |
| Prerequisites | 1. System is functional. 2. Android application installed and running. 3. Hardware device is functional and running. |
| Test Procedure | 1. Establish a Bluetooth connection between devices. 2. Leave both running for 3 hours. 3. Every 30 minutes attempt to take data from devices again. |
| Expected Result | 1. Data can be taken consistently for the full time-block. |
| Actual Result | 1. Data was taken consistently. 2. Hardware got warm (~100F) over time but remained stable. |
| Status | Pass |
| Remarks | Warmth came from power loss from 9V battery. Switching to 4 AAA or AA batteries would get rid of this issue. |
| Test Environment | * Implementation |

|  |  |
| --- | --- |
| Test Suite ID | TS008 |
| Test Case ID | TC055 |
| Test Type | Acceptance |
| Test Case Summary | No personal information is shared or accessed by application. |
| Related Requirement | SE-1 |
| Prerequisites | 1. System is functional. 2. Android application installed and running. 3. Hardware device is functional and running. |
| Test Procedure | 1. Walk through code with user to verify no personal information is accessed. |
| Expected Result | 1. No personal information is accessed. |
| Actual Result | 1. App does not request any personal data permissions. 2. App does request Bluetooth access which gives coarse and fine location information, but not personal. |
| Status | Pass |
| Remarks |  |
| Test Environment | * Arduino IDE * Android Studio IDE |

1. **Installation Instructions**
   1. **Installing the android software**

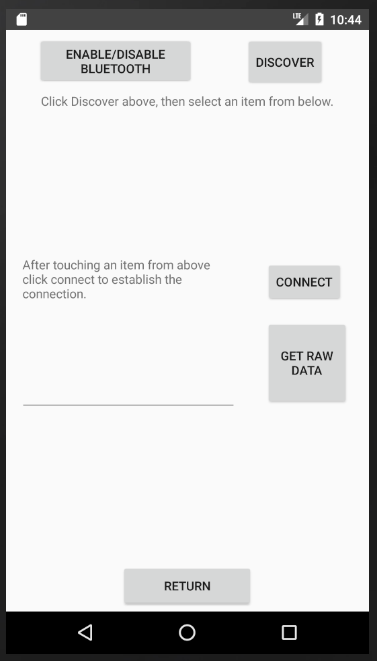
The application can be downloaded and installed from the cloud storage on google drive.

Once it is installed on the phone open it and it will be ready to work.

* 1. **Installing the Embedded System Software**

The embedded software should come preinstalled on the MotionBall system. In the case where it needs to be reinstalled do the following.

1. Open the device box to reveal the hardware stack.
2. Disconnect the blue to green jumper cable, this disconnects the data from the HC-05 Bluetooth module.
3. Connect the Arduino to PC using an appropriate cable.
4. Go to the cloud storage to download the Arduino software.
5. Upload the software using Arduino IDE.
6. Disconnect the Arduino from the PC.
7. Reconnect the blue to green jumper cables.
8. Close the device box.
   1. **Testing the hardware with the android phone**

**Figure 7.1 – Bluetooth Test**

To test the hardware, open the application then navigate to the “Test Connection” page. The figure at the right should be displayed.

Follow the instructions on the page which will be:

1. Be sure Bluetooth is enabled with the enable/disable button.
2. Discover the device using the discover button.
3. Tap on the HC-05 device when it shows up in the connect list.
4. Tap the connect button to establish the connection.
5. Tap the “collect raw data” button to view a stream of values coming from the embedded system device.
6. **Operating Instructions**

This section will detail how to run the hardware and software. Most of the application design was to be user friendly and self-explanatory, this section just details what the app tells users.

* 1. **Running the System**

There are multiple parts of the Android software that require operation. By design and requirements of the equipment each step is explained and directed within the application itself, but the following sections will cover instructions.

* + 1. **Starting the Hardware**

**Figure 8.1- Power**

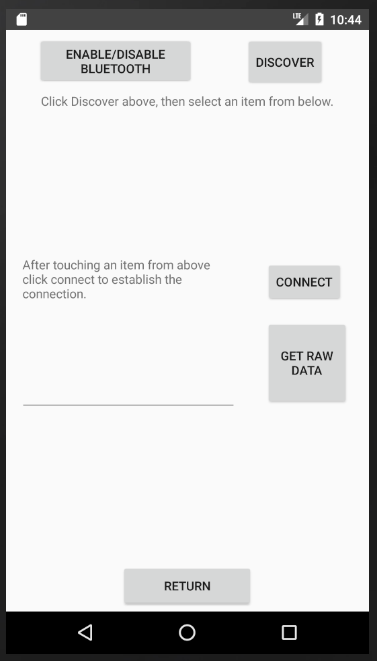
****The hardware requires no setup other than switching the power switch to the on position as shown in figure 7.2.

Once it is flipped to on the hardware will go into discovery mode to be connected to by the Android Application. No further steps are necessary to start the hardware.

* + 1. **Shutting down the hardware**

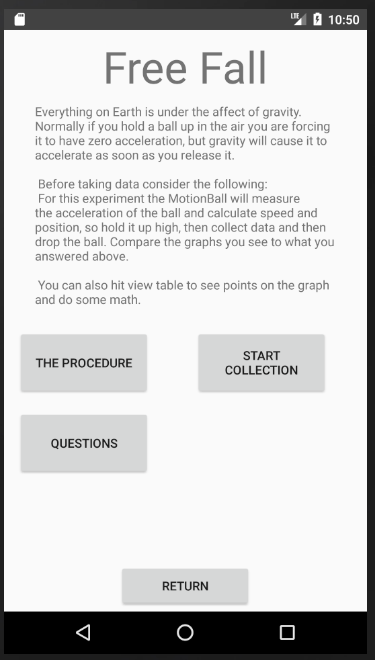
The Hardware should be shut down when not in use to conserve battery. Flip the power switch to the off position as shown at the bottom of figure 7.2 to power down the unit. No further steps are necessary to shut down the hardware.

* + 1. **Testing the Hardware to Android connection**

**Figure 8.2 – BT Test**

To test the hardware, open the application then navigate to the “Test Connection” page. The figure at the right will be displayed. Follow the instructions on the page which will be:

1. Be sure Bluetooth is enabled with the enable/disable button.
2. Discover the device using the discover button.
3. Tap on the HC-05 device when it shows up in the connect list.
4. Tap the connect button to establish the connection.
5. Tap the “collect raw data” button to view a stream of values coming from the embedded system device



* + 1. **Opening the experiment pages**

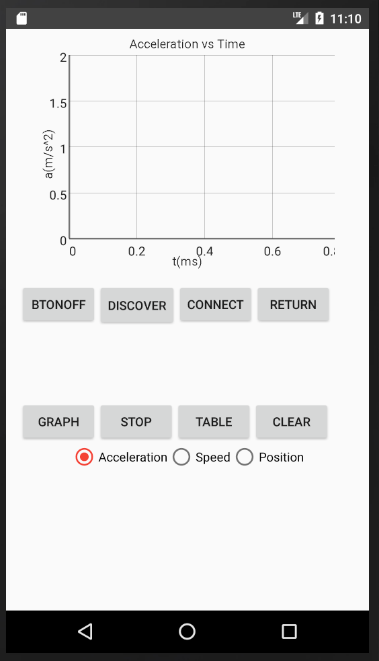
From the main application page click on any available experiment to view the description page for that experiment. For example, selecting the “Free fall experiment opens the screen to the right.

Once this view is opened select “The Procedure” or “Questions” for more information on the experiment. Select “Start Collection” to move on to data collection for the experiment.

**Figure 8.3 - Free Fall Page**

* + 1. **Taking data for the experiments**

Each experiment will have its own method of data collection, this is described in the procedure button on the experiment page. After reading this proceed to “Start Collection” which will open the experiments collection screen.

The Freefall data collection screen is shown here.

The general procedure for taking data is:

1. Turn on the hardware device.
2. Locate the device using the “Discover” button.
3. Select the correct device in the list that appears by tapping it.
4. Tap the “Collect” button to establish a connection and wait for the connecting dialogue to complete.
5. Select which data you would like to view, in this free fall case shown case we can choose Acceleration, Speed, or Position.
6. Click the appropriate button, in this case “Graph” will collect data and plot it on the graph above.
7. Data taking can always be cancelled using the “Stop” button.
8. Data can be completely cleared with the “Clear” button.
9. Data can be viewed in Table form by selecting “Table”, this will show a full array of values for more in-depth analysis.

**Figure 8.4 – Graph View**

1. **Recommendations for Enhancement** 
   1. **Process Model Lessons learned**
      1. **Hardware Selection**

The first point that came up from this project was the difficulty of selecting hardware before design. Originally the model called for coming up with the hardware in the requirements phase, but I found that until the functions were designed and ready to be allocated in the design phase it was very difficult to choose proper hardware.

* + 1. **Difficulty of Architectural Design Refinement**

This project came to a long stall of almost a month during the Architectural design phase. This was caused by lack of experience of the developer in java application, and embedded software systems. Eventually this stall was broken by just accepting where the refinement was and moving into embedded system and component/unit design. Due to the method of the model here the second approach to the architecture refinement went much more smoothly, thus for inexperienced or new teams the refinement will likely be a hinderance in the early stages.

* + 1. **Hardware Interfacing**

Getting the embedded software system to communicate with the Android Application was by far the most difficult part of this project. This was mostly caused by lack of documentation for the Bluetooth interface, the application was easy enough to write, but finding the UUID for Arduino and HC-05 insecure communication channel was challenging. Luckily the microcontroller here was well supported so eventually solutions could be found, but in the case of less user-friendly microcontrollers interfacing hardware could be a nightmare. This leads to a recommendation of allotting significantly more time to hardware-software integration than was originally planned.

* + 1. **Testing**

In this project often, unit testing had to be done before component testing, the difference being that components take data from other parts where unit testing requires test stubs.

* 1. **Process Model Recommended Adjustments**
     1. **Updated Model**

The final version of the embedded software development process is shown here.

**Figure 9.1 – Final Process Model**

**System Definition**

Acceptance Testing

Needs Assessment

Requirements

Extra Functional Properties

Hardware Estimation

**System Development**

Architecture Design Refinement

Acceptance Testing

Integration Testing

Embedded System Design

Unit and Component Testing

Component Design

Development/Prototyping

**System Deployment**

Product Creation

Implementation

Acceptance

* + 1. **Changes**

The main adjustment made was to push final hardware selections completely into the development phase and only do hardware estimations in the early phase. The main reason for this is that unless the developers are extremely familiar with any given hardware component it is very difficult to tell exactly how it will work in any system. Hardware estimation is kept in the definition phase so that the “user” can still be notified if hardware for the task then want already exists or if it needs to be developed separately as part of the project. If hardware already exists it also must be checked with the users to make sure as many nonfunctional requirements as possible are satisfied. As stated in section two this should lead to compromises since often embedded software systems cannot realize all desired extra-functional requirements.

Another change involved adding “unit” to the first stage of component testing. As stated in lessons learned often corresponding software system units had to be tested before component testing could take place. Unit and Component testing are still combined however since it is likely that hardware components can only be tested as components, some devices will not function without the other devices. An example of this is the accelerometer in this system, it could not be tested individually without being connected to a microcontroller.

Finally, “development” was changed to “development/prototyping” to stress the importance of prototyping in these systems. This importance was confirmed by multiple references and is again confirmed in this case study where some defects and issues were impossible to catch before the system was fully assembled. An example of this mentioned earlier in this report was the fact that if code was uploaded to an Arduino Pro Micro while it was hooked up to the HC-05 Bluetooth module data connection, the Arduino would be rendered unusable. If a more integrity heavy system was developed without prototyping for some issues like this, then results could be severe. Thus, while originally “development” method was left to the choice of the project team this model now strongly recommends prototyping by including the term.

* + 1. **Case Study System Possible Additions**

As shown in the design description, section 4 of this report, by putting most of the data analysis into the Android Application the final version of this application leaves room for additions. Three more experiments for using this accelerometer are already being written and work will continue on this after the conclusion of this project. Once all four experiments currently on the main page are designed and implemented I plan to put this application into use in my physics classrooms.

This could go further by creating more embedded software systems and interfacing them with the same android application. Experiments such as electronics and thermodynamics could also be achieved. Using these embedded systems instead of just the phones built in systems allows for a variety of experiments with little risk to or dependence on student’s property.

1. **Bibliography** 
   1. **Referenced**
2. M. A. Abdallah, 2008. "Implementing the component-based software engineering in embedded systems*,” International SoC Design Conference*, Busan, 2008, pp. I-399-I-402. URL: <http://ieeexplore.ieee.org.lib-proxy.fullerton.edu/stamp/stamp.jsp?tp=&arnumber=4815656&isnumber=4815559>
3. I. Crnkovic, 2005. "Component-based software engineering for embedded systems," Proceedings. 27th International Conference on Software Engineering, 2005*. ICSE 2005*., Saint Louis, MO, USA, 2005, pp. 712-713. URL: <http://ieeexplore.ieee.org.lib-proxy.fullerton.edu/stamp/stamp.jsp?tp=&arnumber=1553676&isnumber=33070>
4. M. Li, H. Wang and P. Li, 2002. "Embedded system engineering in pervasive computing era: a software approach," TENCON '02. Proceedings. *2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, 2002, pp. 1696-1699 vol.3. URL: <http://ieeexplore.ieee.org.lib-proxy.fullerton.edu/stamp/stamp.jsp?tp=&arnumber=1182660&isnumber=26489>
5. I. Meedeniya, A. Aleti, I. Avazpour and A. Amin, 2012. "Robust ArcheOpterix: Architecture optimization of embedded systems under uncertainty," *2012 Second International Workshop on Software Engineering for Embedded Systems (SEES)*, Zurich, 2012, pp. 23-29. URL: <http://ieeexplore.ieee.org.lib-proxy.fullerton.edu/stamp/stamp.jsp?tp=&arnumber=6225486&isnumber=6225480>
6. H. J. Onisto et al., 2014. "Model-driven engineering applied to the development of embedded software for B-mode ultrasound imaging systems - A case study," *2014 IEEE International Ultrasonics Symposium*, Chicago, IL, 2014, pp. 1261-1264. URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6932279&isnumber=6931723>
7. A. Polzer, S. Kowalewski and G. Botterweck, 2009. "Applying software product line techniques in model-based embedded systems engineering," *2009 ICSE Workshop on Model-Based Methodologies for Pervasive and Embedded Software,* Vancouver, BC, 2009, pp. 2-10. URL: <http://ieeexplore.ieee.org.lib-proxy.fullerton.edu/stamp/stamp.jsp?tp=&arnumber=5069132&isnumber=5069117>
8. G. Rong, T. Liu, M. Xie, J. Chen, C. Ma, D. Shao. 2014. “Processes for embedded systems development: preliminary results from a systematic review”. *ICSSP 2014 Proceedings of the 2014 International Conference on Software and System Process*, May 2014. URL: <http://dl.acm.org/citation.cfm?id=2600845>
9. M. Shen, W. Yang, G. Rong and D. Shao, 2012. "Applying agile methods to embedded software development: A systematic review," *2012 Second International Workshop on Software Engineering for Embedded Systems (SEES)*, Zurich, 2012, pp. 30-36. URL: <http://ieeexplore.ieee.org.lib-proxy.fullerton.edu/stamp/stamp.jsp?tp=&arnumber=6225488&isnumber=6225480>
10. M. Smith, J. Miller, L. Huang and A. Tran, 2009. "A More Agile Approach to Embedded System Development*," in IEEE Software*, vol. 26, no. 3, pp. 50-57, May-June 2009. URL: <http://ieeexplore.ieee.org.lib-proxy.fullerton.edu/stamp/stamp.jsp?tp=&arnumber=4814958&isnumber=4814945>
11. A. Suliman and N. Nazri, 2014."A new hybrid model of software engineering and systems engineering for embedded system development methodology," *Proceedings of the 6th International Conference on Information Technology and Multimedia, Putrajaya,* 2014, pp. 346-350. URL: <http://ieeexplore.ieee.org.lib-proxy.fullerton.edu/stamp/stamp.jsp?tp=&arnumber=7066657&isnumber=7066586>

## **10.2 Not Directly Referenced/ Used in development.**

1. 1998, IEEE Guide for Developing System Requirements Specifications," in IEEE Std 1233, 1998 Edition, vol., no., pp.1-36, Dec. 29, 1998. URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=741940&isnumber=16016>
2. IEEE Standard for Software and System Test Documentation," in IEEE Std 829-2008, vol., no., pp.1-150, July 18, 2008. URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4578383&isnumber=4578382>
3. T. Noergaard, 2005, “Embedded Systems Architecture”, Elsevier Inc.
4. J. Ivers, P. Clements, D. Garlan, R. Nord, B. Schmerl, and O. Silva, 2004, "Documenting Component and Connector Views with UML 2.0," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report CMU/SEI-2004-TR-008, 2004. URL: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7095