Department of Computer Science & Engineering
The University of Texas at Arlington

Detailed Design Specification
CSE 4317: Senior Design II
Fall 2020

Team Hydro
Bluetooth Hydrometer

Andrew Duong
Rebecca Byun
Gam B. Garbuja
Ryan Tyler

Revision History

Revision	Date	Author(s)	Description
0.1	09.18.2020	AD	document creation
0.2	09.23.2020	GG	complete introduction and system overview
0.3	09.24.2020	GG	complete subsystem definitions & data flow
0.4	09.23.2020	RB,RT	complete system layers and subsystems
1.0	09.25.2020	AD	combined document together and convert to TeX

CONTENTS

List of Figures

List of Tables

1 Introduction

Fermentation is an important process of brewing beer. Key part of brewing beer with a certain taste and density requires fermenting beer at the right temperature for a certain period. For homebrewers, it can be tedious to keep track of temperature and density of beer during the fermentation process.

"Bluetooth Hydrometer" is a device designed to help homebrewers keep track of temperature and density of beer during the fermentation process. It floats on the beer inside the fermenting vessel while sending data to a smartphone via Bluetooth.

Bluetooth Hydrometer consists Arduino-nano with 9-axis inertial measurement unit (IMU) and temperature sensor. Temperature sensor reads temperature while IMU reads relative position when the device is floating. Arduino gets analog input from sensors and process data to get actual temperature and specific gravity of beer. Specific gravity is obtained from relative position of hydrometer inside the fermentation vessel. Once the process is done, data are sent to a smartphone via Bluetooth.

Temperature and density data are stored in a database for future reference and analysis. Mobile app provides visual interface to homebrewers at real time providing current temperature and specific gravity of beer.

2 System Overview

Overall system consists of three major layers, Sensors, Controllers (Hardware/Software) and UI/UX. Each layer has separate functions and interface with each other for data input and output. Sensors layer provide analog data to Controllers for further processing. After analyzing data from sensors, Controllers provide digital data to UI/UX for providing relevant information to user with good visual interface.

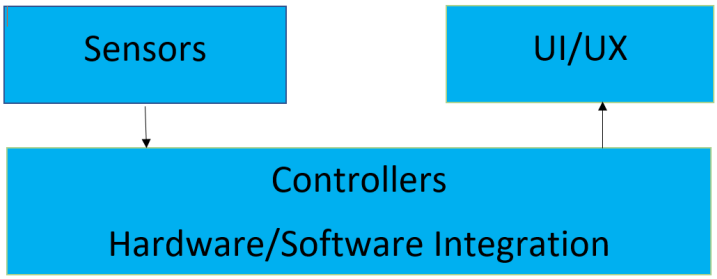

Figure 1: A simple architectural layer diagram

2.1 Sensors

Sensors layer for Bluetooth Hydrometer device mainly consists of two sensors, temperature sensor and 9-axis IMU sensor. Sensors are essential parts of Bluetooth Hydrometer which measure temperature and specific gravity of beer, which are the main requirements of the project. Temperature sensor provides analog temperature read whereas, IMU provides relative position of Bluetooth Hydrometer during floatation. Temperature data and position data are read by Arduino nano in Controllers layer. Sensors, therefore, provides necessary input data to Controllers.

Arduino nano 33 BLE sense had built-in temperature and IMU sensors which makes it easier to read temperature and relative position of Arduino without additional circuitry. IMU sensor contains ac-

celerometer and gyroscope that reads rotation and orientation of Arduino board. Temperature sensors reads temperature, it can also read pressure and humidity if needed.

2.2 CONTROLLERS

Arduino nano 33 BLE sense is the heart of Bluetooth Hydrometer device. Nano is low-powered Bluetooth enabled microcontroller which can read analog as well as digital inputs from sensors. Sensors provide critical analog data for temperature and relative position of hydrometer. Nano then process analog data and provides actual temperature and specific gravity (depending on relative position) to UI/UX. Nano is programmed to handle analog data from sensors and provide output to UI/UX layer.

2.3 UI/UX

UI/UX is another important layer that provides user interface to user by providing actual data. Temperature and specific gravity data are visually and graphically presented to user through a mobile app or through a website. Data's from Controller are stored in a database and are analyzed through a software.

3 Subsystem Definitions & Data Flow

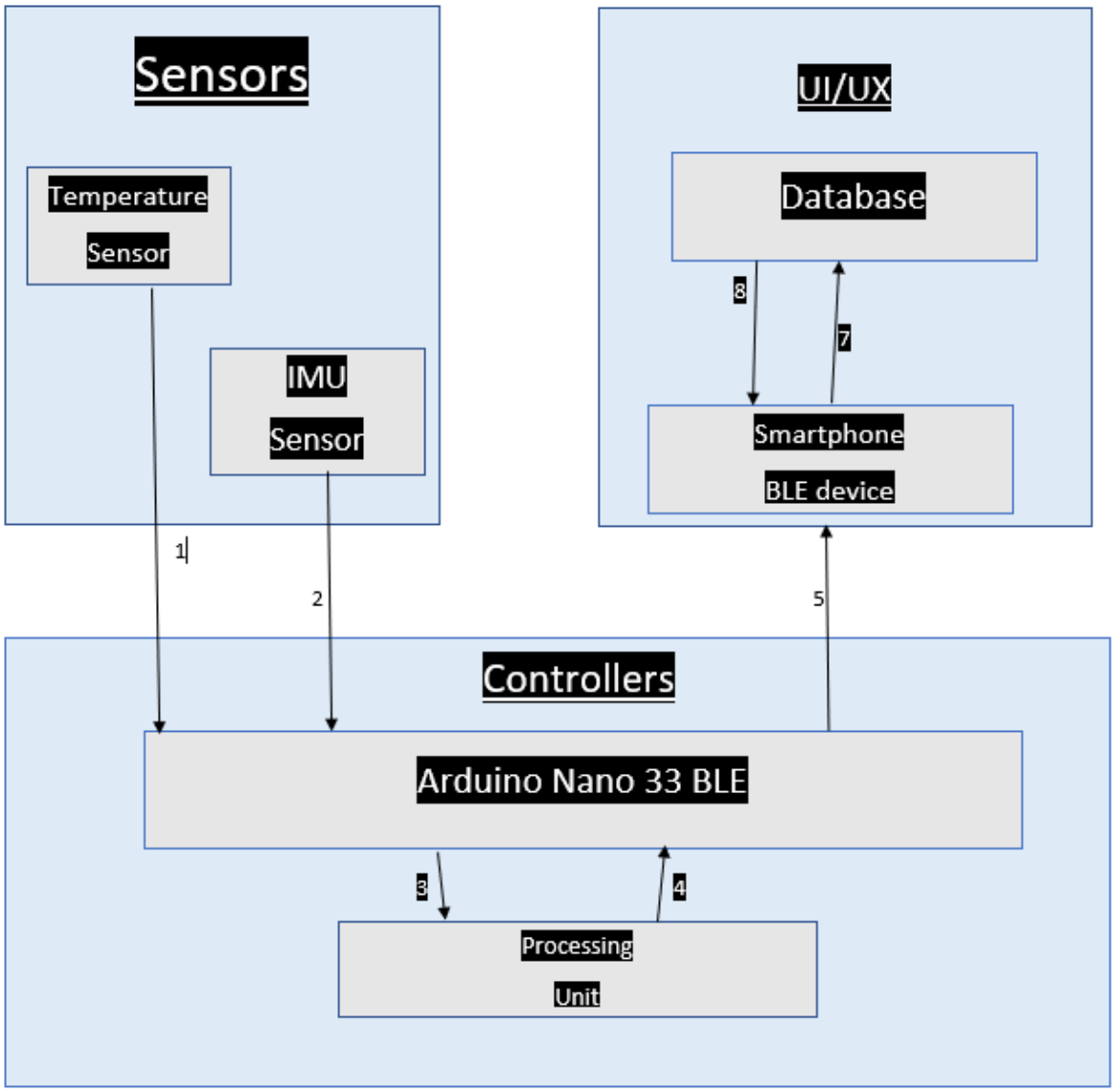

Figure 2: A simple data flow diagram

4 SENSORS

4.1 TEMPERATURE SENSOR

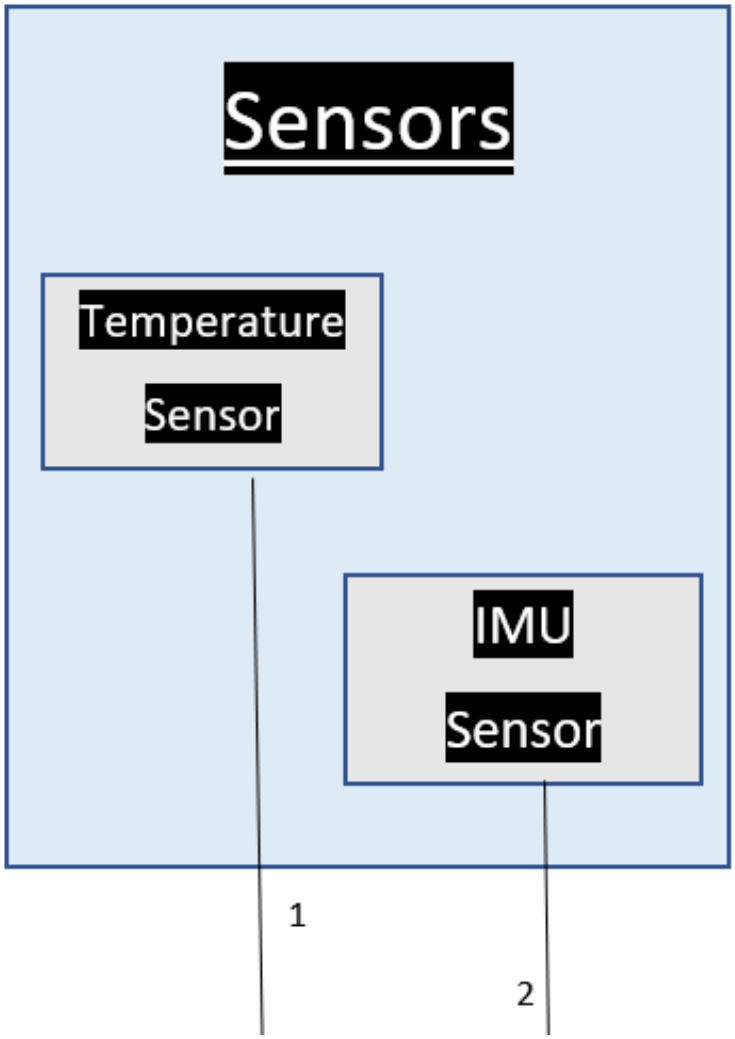

Figure 3: Sensor subsystems description

4.1.1 ASSUMPTIONS

Temperature sensors are built in on Arduino Nano 33 BLE sense which can have small error during the temperature read which can be adjusted though Arduino Programming.

4.1.2 RESPONSIBILITIES

Arduino nano built in temperature sensor reads temperature. Arduino software provides HTS221 library to read temperature from Arduino nano 33 BLE sense. HTS221.h handles temperature readings from Arduino nano board and it must be included during Arduino programming.

```
#include <Arduino_HTS221.h>   //Arduino library to be included
HTS.begin()        //Initializes temperature sensor
-  Returns true of false depending on the initialization
HTS.readTemperature()      //Function to read temperature in degree Celsius
       -  Returns temperature in degree Celsius (default)
HTS.readTemperature(FAHRENHEIT)
-  Returns temperature in Fahrenheit if needed
```

4.1.3 SUBSYSTEM INTERFACES

Each of the inputs and outputs for the subsystem are defined here. Create a table with an entry for each labelled interface that connects to this subsystem. For each entry, describe any incoming and outgoing data elements will pass through this interface.

Table 2: Subsystem interfaces

ID	Description	Inputs	Outputs
	Temperature interface	N/A	output 1

4.2 IMU Sensor

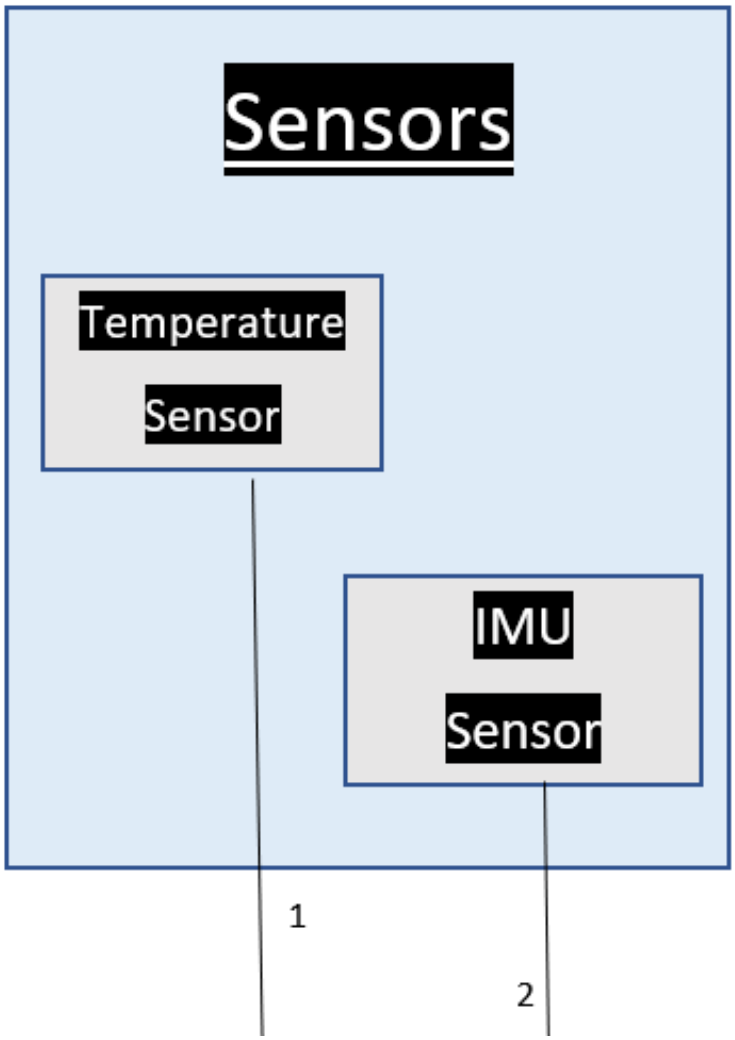

Figure 4: Sensor subsystems description

4.2.1 Assumptions

Depending on the Bluetooth Hydrometer case, there might me some fluctuation in IMU readings which can be adjusted within the Arduino software.

4.2.2 Responsibilities

Arduino nano 33 BLE sense is an advance microcontroller, it consists of 9-axis Inertial Measurement Unit (IMU) which refers to built-in gyroscope, accelerometer and magnetometer. For our project, we will be using gyroscope and accelerometer only. Accelerometer and gyroscope reading are handled by LSM9DS1 library in Arduino. LSM9DS1 library must be included during Arduino nano programming. LSM9DS1 has built-in functions to read from accelerometer and gyroscope.

```
#include <Arduino_LSM9DS1.h>    //Arduino library to be included for IMU
IMU.begin()          //Initialize IMU
-  Returns 1 on success and 0 on failure
```

Accelerometer helps to read the device relative orientation on X, Y and Z-axis. It returns values of X, Y and Z based on tilt as shown in figure 5.

```
IMU.accelerationAvailable()
-  Returns 0 if no new acceleration data is available
-  Returns 1 if new acceleration data is available

IMU.readAcceleration(x, y, z)
-  1 on success and 0 on failure
-  X, y and z are float variables where acceleration values will be stored
```

IMU acceleration read step shown below

```
If(IMU.accelerationAvailable()){
      IMU.readAcceleration(x, y, z);
```

Gyroscope reads the relative rotation of Arduino board on X, Y and Z-axis. It returns values of X, Y and Z based on angle of rotation of axis, i.e. upside down or upside position. Y-axis value is used for the project to determine the rotation of Arduino nano. X and Z-axis value is not prioritized as Arduino should be floating in upright position while reading values from Arduino (Y-axis matters). Gyroscope read is same as accelerometer read.

```
IMU.gyroscopeAvailable()
-  Returns 0 if no new gyroscope data sample is available
-  Returns 1 if new gyroscope data sample is available
IMU.readGyroscope(x, y, z)
-  1 on success, 0 on failure
-  X, y, z are float variables where the gyroscope values will be stored
```

IMU gyroscope read steps shown below,

```
If(IMU.gyroscopeAvailable()){
      IMU.readGyroscope(x, y, z);
```

4.2.3 SUBSYSTEM INTERFACES

Each of the inputs and outputs for the subsystem are defined here. Create a table with an entry for each labelled interface that connects to this subsystem. For each entry, describe any incoming and outgoing data elements will pass through this interface.

Table 3: Subsystem interfaces

ID	Description	Inputs	Outputs
	IMU interface	N/A	output 2

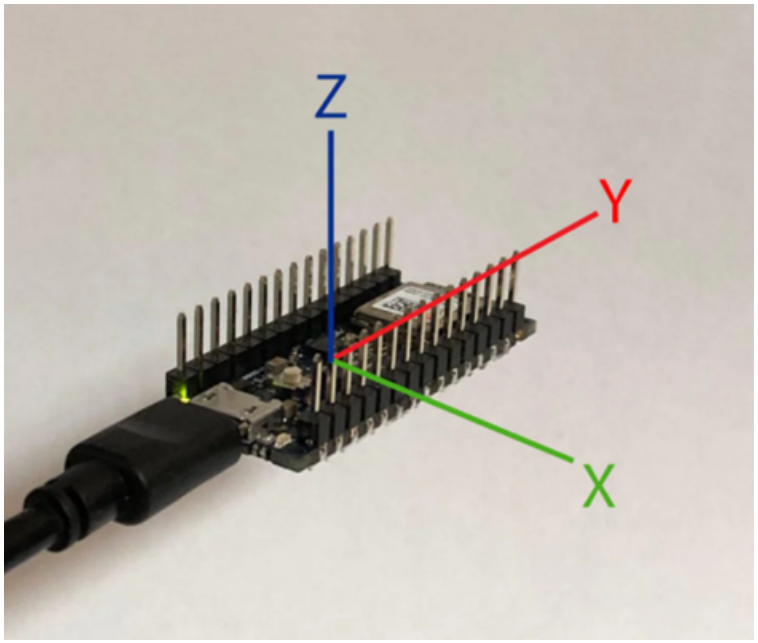

Figure 5: Arduino relative position on X, Y and Z axis's

5 Controller Subsystems

5.1 Arduino Nano 33 BLE

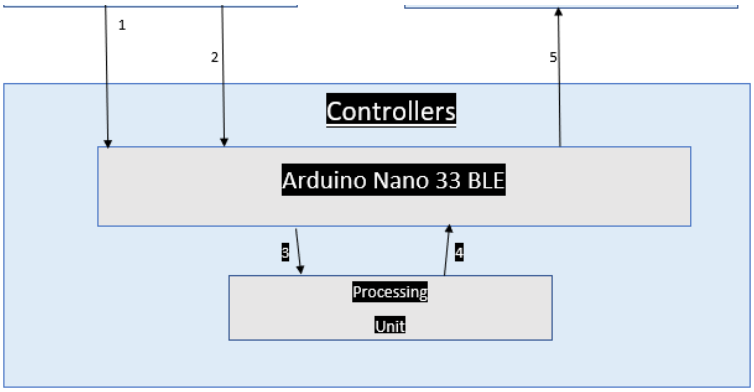

Figure 6: Controller Subsystem

5.1.1 Assumptions

Not applicable.

5.1.2 Responsibilities

The main responsibility of Arduino nano is to receive analog signals from temperature sensor and IMU sensors. CPU inside then process analog data then transfers process data to UI/UX layer via Bluetooth.

NINA-b3 (nRF52840) chip inside Arduino nano BLE 33 sense handles processing data. Arduino nano will operate at 5-3.3 V powered by 9V battery (9V will be divided with voltage divider) after the com-

pletion of Bluetooth Hydrometer.

Arduino 1.8.13 is used to program Arduino nano 33 BLE sense which is the proprietary software Arduino provides. Arduino serial monitor will be used to test the results.

5.1.3 Subsystem Interfaces

Data elements will pass through this interface.

Table 4: Subsystem interfaces

ID	Description	Inputs	Outputs
	Controller interface	input 1 input 2 input 4	output 5

5.2 Processing Unit

5.2.1 Assumptions

Hydrometer is always turned on whenever it's inside the fermentation vessel. Since, position of hydrometer is needed for specific gravity, it's assumed that the hydrometer is floating.

5.2.2 Responsibilities

60 MHz CPU is responsible for processing temperature and specific gravity data, that it receives from built-in IMU and temperature sensors. Arduino nano is programmed to handle those data whenever nano is turned on. It outputs the accurate temperature and specific gravity data after processing.

5.2.3 Processing Unit Interface

Table 5: Subsystem interfaces

ID	Description	Inputs	Outputs
	Processing Unit Interface	input 3	output 4

5.3 Bluetooth Interface

5.3.1 Assumptions

Not Applicable.

5.3.2 Responsibilities

Bluetooth will be handled by ArduinoBLE library. BLE library provides various functions to send data from Arduino to a Bluetooth device. Functions needed for Bluetooth Hydrometer are listed below. Functions will be added or removed depending on the modifications required in the future.

```
#include <ArduinoBLE.h>    //Bluetooth library needs to be included
BLE.begin()
-  Returns 1 on success and 0 on failure
```

```
BLE.central()         //For connecting Bluetooth devices (central - Bluetooth
    device)
-  Returns BLEDevice representing the central
BLEService (uuid)
BLEServide(index, uuid)
-  Returns BLEService for provided parameters
-  Index - index of service
-  Uuid - uuid(string) //BluetoothHydrometer
BLE.read()
-  True, if successful, false on failure
BLE.notify()
-  Notify change in BLECharacteristic //change in temperature or accel and gyro
    data
BLE.setLocalName(name)
-  Name: local name to be used when advertising
-  Returns: none
BLE.addService(service)
-  Service to add
-  Returns none
BLE.setAdvertisedService(bleService)
-  Returns none
-  BLEService to use UUID from
bleService.addCharacteristic(bleCharacteristic)
-  Add a BLECharacteristic to the BLE service
bleCharacteristic.writeValue(value)
-  Value - value to write
-  bleCharacteristic - temperature and position data for Bluetooth Hydrometer
BLE.advertise()
-  Start advertising
-  Returns 1 on success, 0 on failure
```

5.3.3 SUBSYSTEM INTERFACES

Table 6: Subsystem interfaces

ID	Description	Inputs	Outputs
	Bluetooth Interface	input 4	output 5

6 PHONE LAYER SUBSYSTEMS

In this section, the mobile application used to control the hydrometer will be outlined. The user will interact with the hydrometer application, which will send the request to the server.

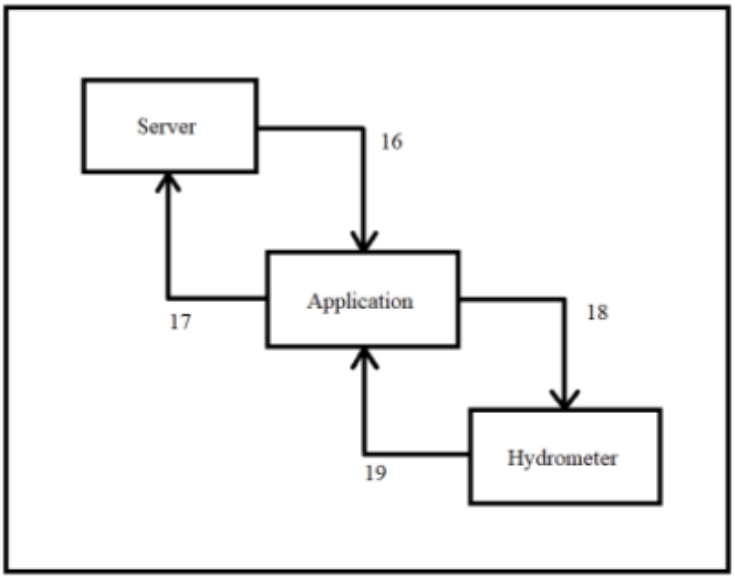

Figure 7: Mobile Phone Subsystem

6.1 LAYER HARDWARE

The phone layer will be a mobile application emulated on a physical computer operating system with the ability to connect wireless via Bluetooth to the Controller Subsystems. It will process data given to it from the Controller subsystem and transfer it to the subsystems inside the Phone Layer.

6.2 LAYER OPERATING SYSTEM

The physical system's Operating System emulating the Android application will be macOS Catalina version 10.15.6 and the emulated Operating System will be Android Ice Cream Sandwich version 4.0.

6.3 LAYER SOFTWARE DEPENDENCIES

The main dependency that will be operating is Android Studio and SDK tools version 4.0.1.

6.4 DATABASE SUBSYSTEM

The database will be where all data on an hourly/daily basis will be stored. This data will be stored for use and analysis by the user at the time of their choosing.

6.4.1 DATABASE SOFTWARE DEPENDENCIES

MySQL Workbench will be utilized as an interface for data collection and analysis.

6.4.2 DATABASE PROGRAMMING LANGUAGES

Languages used for the database will be Java and MySQL.