DESIGN OF BLUETOOTH COMPATIBLE TEMPERATURE AND PHOTOPLETHYSMOGRAM (PPG) MONITORING SYSTEM

by

WONG KENG MUN
(16388)

Dissertation submitted in partial fulfilment of
the requirements for the
Bachelor of Engineering (Hons)
(Electrical and Electronic)

JANUARY 2016

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
32610 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Design of Bluetooth Compatible Temperature and Photoplethysmogram (PPG) Monitoring System

by

Wong Keng Mun
16388

A project dissertation submitted to the
Electrical and Electronic Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of requirement for the
BACHELOR OF ENGINEERING (Hons)
(ELECTRICAL AND ELECTRONIC)

Approved by,

____

(Professor Dr. Varun Jeoti)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
January 2016

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein here not been undertaken or done by unspecified sources or persons.

____

WONG KENG MUN

ABSTRACT

A Bluetooth Compatible Temperature and Photoplethysmogram (PPG) Monitoring System is designed to allow continuous monitoring of body temperature and heart beat per minute (BPM) of the user. The system is specifically designed to all Bluetooth compatible Android device user to convenient the user and to alert the responsible person when an emergency happens especially to the patient or the elderly. The system integrates both Digital Temperature sensor (DS1820) and Photoplethysmogam pulse sensor using Arduino and sends all the detected information to the Android device via the Bluetooth Module (HC-06). All the data that send using the system have all verified by using another certified Digital Thermometer and OMRON HEM-7203 Automated Blood Pressure Monitor. Under the optimal condition where the sensors have successfully detected the information, the percentage error of temperature and beat per minute are 1.92% and 4.44% respectively. For temperature, the percentage error is caused mainly because the DS1820 have ± 0.5℃ increment while the temperature sensor has ± 0.1℃ increment. Also, DS1820 does not have a proper temperature insulation to the surrounding. For heart pulses, the system displayed almost perfect plotting of PPG data, however, the comparison of BPM does not seem to be perfect because the OMRON Blood Pressure Monitor using the different concept to detect a pulse.

ACKNOWLEDGMENT

First of all, the author would like to express his gratitude for all the help and guidance received from his supervisor, Professor Dr. Varun Jeoti. His brilliant insight and significant feedback provided to the author throughout the whole Final Year period has made the project successful.

The author also like to take this opportunity to thanks, everyone especially Dr. Aamir Malik, Mr. Asif Iqbal and Mr. Zubair who supported me directly or indirectly during the designing process.

Lastly, the author would like to express his highest gratitude toward his father, Mr. Wong Kim Tack and his late mother, Mrs. Tan Siew Wee for their uncountable encouragement and support in all direction during his study in Universiti Teknologi PETRONAS.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLE

LIST OF EQUATIONS

CHAPTER 1

INTRODUCTION

1.1 Background Study

The human body is actually very sensitive and intelligent as it often sends us some early signal or indication about our body health condition through all kind of vital signs. In Medical, vital signs consist of heart rate, body temperature, respiration rate and blood pressure. All these parameters are very important for the medical diagnosis of the patient. Often these data are recorded as the reference during the first visit and compared with the future diagnosis to determine the patient situation[1]. However in this project, heart rate and body temperature are the only parameters that studied due to time constraint.

Heart rate refers to the number of the cardiac cycles that our cardiovascular system able to perform in every minute. As such, heart rate assessment usually records in beats per minute (BPM). During the assessment, the heart rate can be calculated by either recording number of pulses felt in 10 seconds or 15 seconds and multiply by either 6 or 4 [2].

As a healthy adult, the heart rate is typically 70-80 BPM, but the standard can be vary due to variables such as gender, age, current physical activity, body temperature, and emotion. In summary, the average heart rate of male is 68 BPM while the female is 75 BPM. However, the value also depends on the age of the human as the heart rate of young people are typically higher than an adult. Another parameter is due to fitness of the human as the heart rate is slightly different for those who exercise regularly and those who not use to exercise. Besides, body temperature is proportional to the heart rate due to the metabolic rate of the cardiac cell. A common symptom such as fever or hypothermia will cause the heart rate become faster or slower. Lastly, emotion also plays a role in affecting heart rate. Feeling such as excitement, fear or stress will generally increase our heart rate which caused the heart rate assessment inaccurate[2].

Human body temperature is considered as another important vital sign for a human. Body temperature is controlled by a part of our brain which called as Hypothalamus. To be specific, Hypothalamus is regulating the amount of heat produced and the release of heat to the environment to maintain our body temperature. However, the heat at our body is not evenly distributed. Our body cell has metabolized differently and generate a specific amount of heat when they carry out a specific type of activity[3]. Below are the different parts of human body which generally is used to measure our body temperature for medical purposes.

1. Rectal Temperature
2. Virginal Temperature
3. Oral Temperature
4. Axillary Temperature
5. Tympanic Temperature
6. Skin Temperature
7. Bladder Temperature

Although there are different part of our body to take body temperature measurement, rectal temperature is considered the most accurate which represent the current temperature of the human[4].

1.2 Significant of vital sign monitoring

When treatment involves, vital signs are important parameters when they are used to diagnose a patient. All these data serve as a record of the patient exactly like a marker of health to the patient. The sudden change of vital signs usually indicates the patient condition can be worse and emergency notification is needed to inform the respective doctor. At this point, all data recorded in those seconds are extremely useful for the doctor to make an accurate decision.

Besides monitoring the patient in the hospital, vital signs monitoring can be useful in monitoring the health of the one you love especially to all the elderly, individuals who have certain disease recorded. For example, heart rate monitoring can provide valuable information about the cardiovascular system of the client who have bradycardia or tachycardia medical record and notify for emergency medical attention. Bradycardia refers to the irregular heart rate which is too slow while Tachycardia refers to the opposite.

Another application of this device is related to multiple studies of diseases such as sleep apnea and congestive heart failure by scientist to understand their behavior. This is particularly useful in improving the treatment process by giving more reliable data about the disease.

1.3 Problem statement

As one of the biggest factor of people suffering incurable disease is due to the failure of early detection. Instead of regular body checking which usually carry out once in a half of year or yearly, people should have their basic vital sign constantly monitored by the vital sign monitor.

1.4 Objectives

The goal of this study is to design a continuous temperature and heart rate monitor which involve the following process.

- To study and identify the appropriate sensor and method to measure heart rate and body temperature accurately
- To integrate both measuring method in a system and implement Bluetooth device to transfer the data
- To study and implement algorithm of processing the measured data and necessary computation

1.5 Scope of study

To compensate the limited time frame of the project which is only seven months the scope of the study is focused on body temperature monitoring and heart rate monitoring system. This monitoring system will involve in detail study of the method of taking information of the body temperature and heart rate. Then, the parameters will be transferred to the receiving end to perform real-time monitoring over the body temperature and pulse rate of the patient. At the same time, all these data can be saved at any time as a record for future use.

CHAPTER 2

LITERATURE REVIEW AND THEORY

2.1 Temperature Measurement

Body temperature measurement should be done as accurate as possible. This is important because even one small degree variation of body temperature will result in the different diagnosis and treatment as different body temperature can indicate the problems in our body[3]. For the past few decades, glass thermometer which utilizes the principle of expansion and contraction of the substance (mercury) is used to take the temperature in our Rectal, Oral, and Axillary. However, the mercury thermometer is needed to be handled carefully as the glass is easy to be broken and harmful to the environment and human[5]. Slowly afterward, the use of mercury glass thermometer is being replaced with a digital thermometer because easy to use, free from parallel error and getting stable and reliable data. In today market, there are mainly 5 types of temperature sensor which include of Thermocouples, Resistance Temperature Detectors (RTD), Thermistors, infrared and semiconductor sensors[6]. At the end of the temperature measurement review, digital temperature sensor DALLAS DS18B20 which categorized as the semiconductor sensor is used in this project.

2.1.1 Thermocouple

Thermocouple utilizes the principle in which different type of metals which soldered together in one end will produce different net thermal emf when they exposed into two different temperature. Because of this principle, an extremely wide range of temperature can be measured and usually they are implemented in heavy industry which involve high temperature. Thermocouple is indeed a good thermometer as they are low cost, rugged and available in smaller size. However, the major disadvantage of the thermocouple are due to the output signal is too low and they have linearity error. To utilize this thermocouple in the digital controller, the signal must be regulated through conditioning and higher order of polynomial equation is needed as a software calibration to avoid linearity problems[7].

2.1.2 Resistive Temperature Device

Resistive Temperature Device (RTD) as the name suggest, it is a temperature sensing device which able to vary its resistance when it detects different temperature. The principle lies on the material chosen such as Platinum, Nickel or Copper which particular sensitive to temperature changes. The Thermal Response and Typical Material Resistivity is listed in Table (1).

TABLE 1: Thermal response and typical material resistivity table.

RTD Material	Thermal Response	Typical Material Resistivity
Platinum	0.00385 Ω/Ω/˚C (IEC 751)	9.81 x 10-6 Ω cm
Nickel	0.00672 Ω/Ω/˚C	5.91 x 10-6 Ω cm
Copper	0.00427 Ω/Ω/˚C	1.53 x 10-6 Ω cm

To measure the resistance across an RTD, a constant current is applied and measure the resulting voltage to determine the RTD resistance. RTDs exhibit fairly linear resistance to temperature curves over their operating regions, and any nonlinearity is highly predictable and repeatable. Due to this, it has higher linearity compared to the Thermocouple. However, despite the effect of the mechanical stress which can have a long-term effect on the repeatability on the sensor, the RTD has one major problem on its accuracy. This is because the self-heat generated as the result of the high current passing through it. As mentioned above, high current excitation is needed to convert the resistance into a voltage and it generate a high power dissipation in the form of heat which will artificially increase the resistance of RTD and make the data less accurate.

2.1.3 Thermistors

The principle of Thermistors is similar to the RTD except the material used by it is semiconductor which exhibits a highly nonlinear resistance vs temperature curve. It is fair to mention that, the Thermistor able to vary its resistance largely by a very small change of temperature and thus it becomes more accurate compared to the RTD. On the downside of the Thermistor, it is also the same problem with the RTD because it required a high excitation current which makes the thermistor become extremely hot and effect the accuracy. In simple term, in compensating of high accuracy of data thanks to its sensitivity, the thermistor lose a certain amount of temperature range that it can detect and the problem cannot be simply overcome by software calibration.

2.1.4 Integrated Silicon-Based Sensor

Semiconductor sensor can be categorize based on its output parameter (either Voltage, Current, Digital or Resistance) and the diode temperature sensor[6]. As it stated above this silicon temperature sensor is actually designed to be compatible with a microcontroller with its built-in circuits such as signal processing circuitry, analog sensing circuitry, and digital input/output. The concept of semiconductor sensor is all started with BJT base-emitter voltage equation:

$$V_{BE} = \frac{kT}{q}\ln\left(\frac{I_c}{I_s}\right)$$

EQUATION 1: BJT Base Emitter Voltage Equation

Where k referred as Boltzmann’s constant, T referred as absolute temperature and I_s referred as the current of geometry. This process requires placing the identical N transistors together with the existing transistor to eliminate I_s and to calculate the temperature based on the Brakaw bandgap reference[8].

In general, all semiconductor temperature sensor working in the same mechanism. However, because this N transistor Circuit is integrated with another circuit, it can be further classified as analog semiconductor temperature sensor and digital semiconductor temperature sensor. Just as the name suggested, the digital sensor is integrated with A/D and D/A converter while analog sensor does not integrate. Both have their own usage as the analog sensor able to provide more precision data while digital sensor able to provide more stability. In our case, both sensors are applicable to implement but digital temperature sensor will be chosen as it has more stable output compared with an analog sensor.

2.2 Heart Rate Measurement

The heart is an important organ which functions as a pump for our cardiovascular system. Contraction of our heart allows the transfer of all oxygen-rich blood and nutrient to every part of our body. Technically, pulse rate and heart beat rate have the same rate but they are not the same as rate pulse is slightly delayed. This is due to the fact that, human only feel the pulse after the blood flow all the way via millions of vein in our body. As one of the vital sign of human, pulse rate is as important as body temperature because pulse rate also can reflect our physical condition. Various factor such as emotion, infection and performing physical activities can affect our pulse rate. However, with proper measurement, those behaviors can be identified and be used medical study. Traditionally, arterial pulse rate being measured by putting fingers on the vein around the wrist or neck and calculate the amount of pulse feel over one minute. As the technology advanced, various heart rate sensor such as Electrocardiograph (ECG) and Photoplethysmogram (PPG) is used to measure heart rate. As we compare ECG and PPG, ECG will have higher accuracy compared with PPG but they are more costly. In this project, we choose PPG pulse sensor because PPG is easier to be integrated with temperature sensor.

2.2.1 Electrocardiograph (ECG)

Electrocardiograph (ECG) is a method of detecting electrical signal through the skin of a human. Usually, ECG consists of two part which is the transmitter and receiver. The transmitter is typically placed on the chest near to our heart and transfers the ECG signal which follows the heart function closely while the receive simply receives the signal and compute the data. In this method, heart rate is calculated by counting the amount of small squares between R-wave-to-R-wave (RR). The recorded number is then used as the denominator of 1500 which provides the Heart Rate with unit Beats per Minute. This nominator of 1500 is an outcome due to the ECG paper runs at 25mm/sec at the monitor. Below is the formula to calculate Heart Rate.

$$Heart\ Rate = \frac{25\ \left({}^{mm}/_{sec}\right) \times 60\ \left({}^{sec}/_{min}\right)}{number\ of\ squares} = \frac{1500}{number\ of\ squares}\left(\frac{Beats}{Minute}\right)$$

EQUATION 2: ECG Pulse Rate Calculation

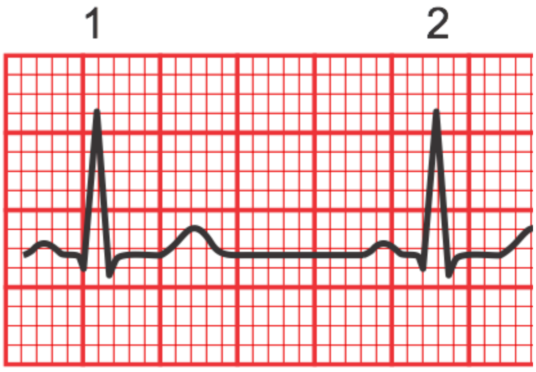


FIGURE 1: ECG R-wave-to-R-wave Interval

2.2.2 Photoplethysmogram (PPG)

Another Heart Rate sensing technology is by utilizing the concept of Photoplethysmogram (PPG) to measure the pulse rate of our heart. In simple words, Photoplethysmogram is the graph light intensity which measured by an integrated device called as a pulse oximeter. Our blood vein appears to be blue because our skin filter out most of the wavelength of other color and only the blue color wavelength are reflected back to our eye. The pulse rate can be measured by calculating how much absorption of the green light with photodiodes placed at the sensor. In this pulse sensor, a green color light-emitting diode (LED) is used to transmit light source toward our vein which rich in capillary tissue while the photodiode is used to capture every single data related to the intensity of the green light that near the LED. Thus, by observing the intensity of light (Photoplethysmogram) of each intensity cycle and the frequency (Pulse Rate) can be calculated [9].

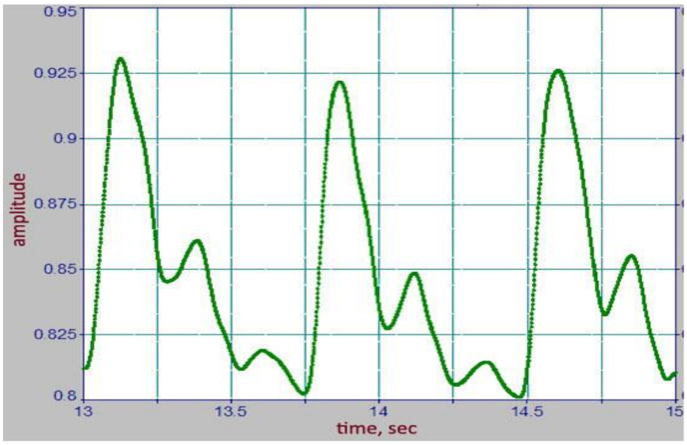


FIGURE 2: Typical Photoplethysmogram (PPG) Waveform

CHAPTER 3

METHODOLOGY

The design of this application mainly divided into 3 parts which are design concept, programming methodology, and verification methodology. The design concept will be mainly discussing the subsystem involved in designing the monitoring system. Whereas programming methodology will be discussing software used and programming method and the verification methodology will be focused on the method to verify the obtained data.

3.1 Design Concept

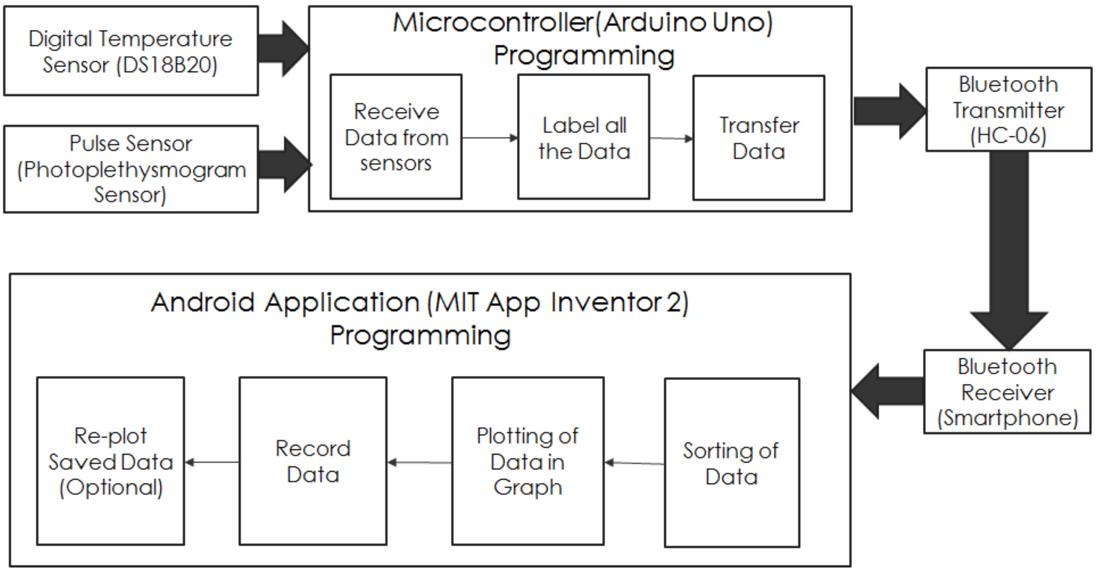


FIGURE 3 System Block Diagram of the Monitoring System

The design concept is started by conducting research on appropriate sensor to work in this project. After comparing all types of temperature sensor available in the market, the digital temperature sensor DS18B20 and PPG pulse sensor are selected as the sensor for this project.

DS18B20 is a typical digital semiconductor based sensor with an operating voltage of 3V to 5.5V. Although semiconductor sensor has a lower range of temperature measurement (-55 °C to +125 °C) but its range is ideal for measuring the body temperature of a human. Additionally, this sensor has ±0.5°C accuracy which is able to provide more accurate and stable output to the microcontroller.

As for heart rate measurement, PPG is easier to be integrated with a temperature sensor. Although, ECG have a higher accuracy compared with PPG but they are more costly and require to measure the heart rate through certain part the only body only. This PPG heart rate sensor has an operating voltage of 3.3V to 5V which is suitable to be implemented with Arduino without additional amplifier circuit. The Pulse Rate of a human can be computed using the formula below.

$$Pulse\ Rate = Number\ of\ Pulses\ in\ 10\ seconds \times 6\ (BPM)$$

EQUATION 3: PPG Pulse Rate Equation

In our monitoring system, Arduino Uno is used as our controller to detect the Temperature and Heart Rate reading. Arduino Uno basically receives the incoming raw signal from the sensors and label them accordingly. The labeled data will be sent to the Smartphone via the Bluetooth devices HC-06. HC-06 is a Bluetooth 2.0 device which is relatively cheaper and more stable compared to Bluetooth 4.0. Besides, this device is compatible with Arduino and user-friendly compared to another method of transfer data such as ZigBee.

On the other hand, Android Smartphone is used to receive data transmitted from Bluetooth HC-06. An Android Application is developed to process the incoming data by sorting them accordingly and to plot the PPG waveform in Real Time. The user has the option to save the recorded data to re-plot back the saved data for future study purpose.

3.2 Programming Method

In this project, the programming is separated into 2 part which is the programming of the microcontroller (Arduino) and Programing of Android Application (SmartPhone).

3.2.1 Arduino

Arduino is programmed in C programming and is one most user-friendly microcontroller which allows the user to create various kind of project. The controller is programmed to capture measured data from the DALLAS 18B20 and PPG Heart Rate Sensor and transmit the data via the Bluetooth Device HC-06 to the Smartphone. The transmission of the data needs to be optimized in different timing to maintain the accuracy of the measured value. However, before the data is sent the controller will label all the data with the specific alphabet. For instances, PPG raw signal will be labeled with letter S, pulse rate will which computed in the controller will be labeled with letter B whereas body temperature will be labeled with letter T. This setting is important because multiple types of data are involved in the system and it will be useful in Android Application later.

3.2.2 Android Application

Android Application is programmed using an open source web program which known as MIT APP INVENTOR 2. The whole program utilizes graphical programming method which is easy to learn if the right amount of time is spent. However, due to the ongoing development of the program, this program still attaching new features and a lot of limitation such as not compatible with latest Bluetooth 4.0.

As for the programming, the application is set to receive the data from Arduino and process the incoming data to allow Real-Time Plotting. At this point, the user has the choice to start and stop at any time and record down the data inside the Smartphone.

After that, the user will have a choice to re-plot the saved data and export the data is text format or the picture of the graph that saved in .PNG format.

Below are the summary for design methodology.

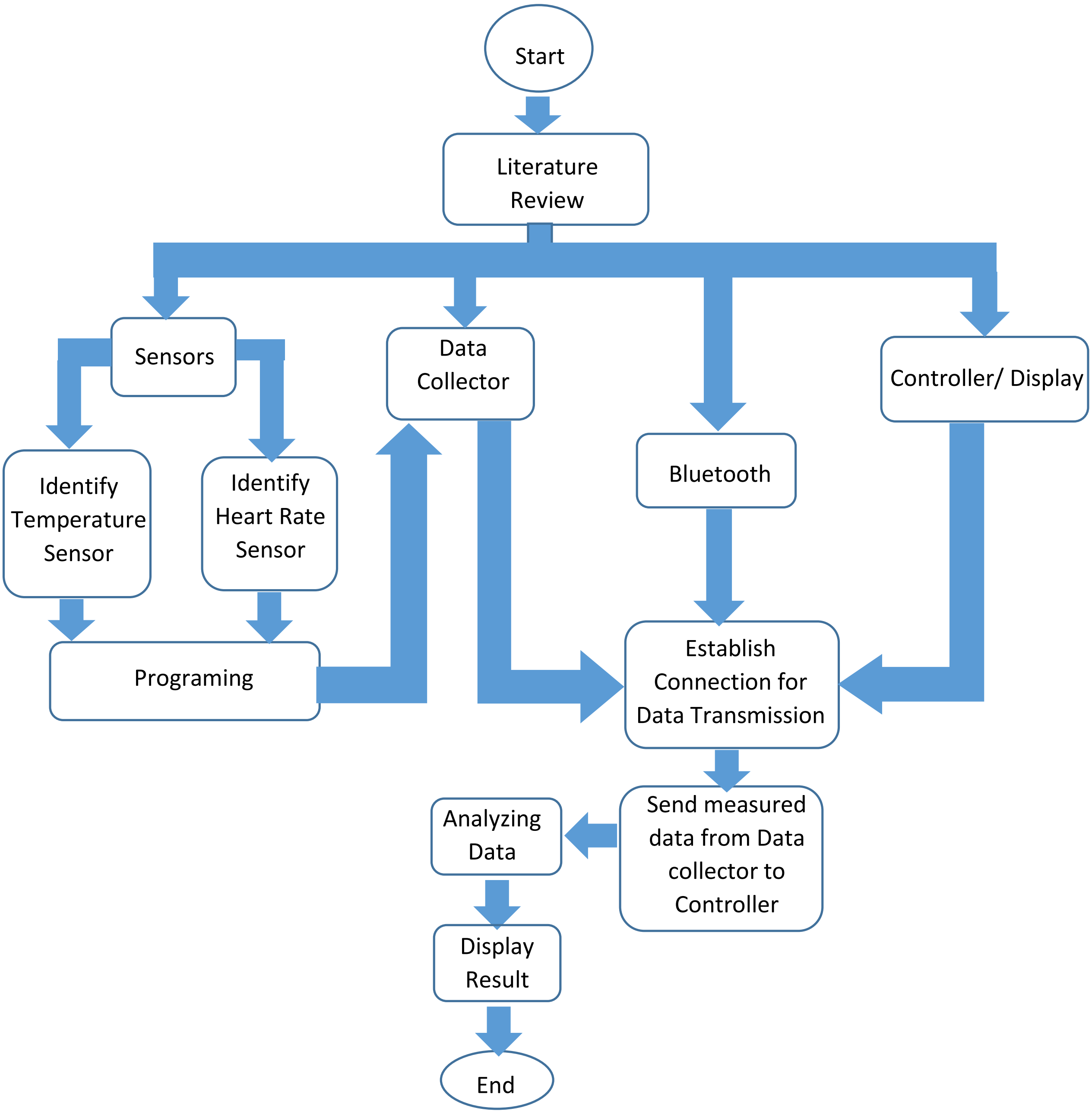


FIGURE 4: Flow Chart of Design Process

3.1 Verification Process

The verification process is to ensure the monitoring system is ready to use and also to debug and optimize if the output is not accurate. Verification conducted mainly by comparing the obtained through the system with other certified instruments. Verification of temperature would require a commercial digital thermometer that certified by the clinic. Same goes to verification of heart rate, OMRON HEM-7203 automated pressure monitor is used to compare the pulse per minute which as shown in Figure 5.

FIGURE 5 Verification Process

Verification for temperature begin with placing of sensor DS18B20 of the system and Digital Thermometer on the same part of the body for at least 1 minute to stabilize the reading of the sensors. After the both reading stabilized, the data will be recorded for every 10 seconds in the total duration of 5 minutes. After 5 minutes, all the recorded data will be tabulated using the Equation 4. This verification procedure is repeated for another 4 times to measure repetitive error.

$$Mean\ Body\ Temperature = \frac{\sum_{i=1}^{N} Temperature(i)}{N}\ °C$$

Equation 4: Mean Body Temperature Equation

Verification for pulse rate is similar to the verification of temperature. The PPG pulse sensor and the OMRON Automated Pressure Sensor are placed on the ear and upper arm respectively for at least 3 minutes to stabilize the reading. After the both reading stabilized, the data of PPG sensor will be recorded for every 10 seconds in the total duration of 10 minutes whereas the data from OMRON will be taken for every 1 minute. This is because the OMRON does not provide immediate value and it need time to stabilize its reading. After 10 minutes, the mean data will be tabulated using the Equation 5. This verification procedure is repeated for another 4 times to measure repetitive error.

$$Mean\ Pulse\ Rate = \frac{\sum_{i=1}^{N} BPM(i)}{N}\ BPM$$

EQUATION 5: Mean Pulse Rate Calculation

Below is the summary of verification methodology.

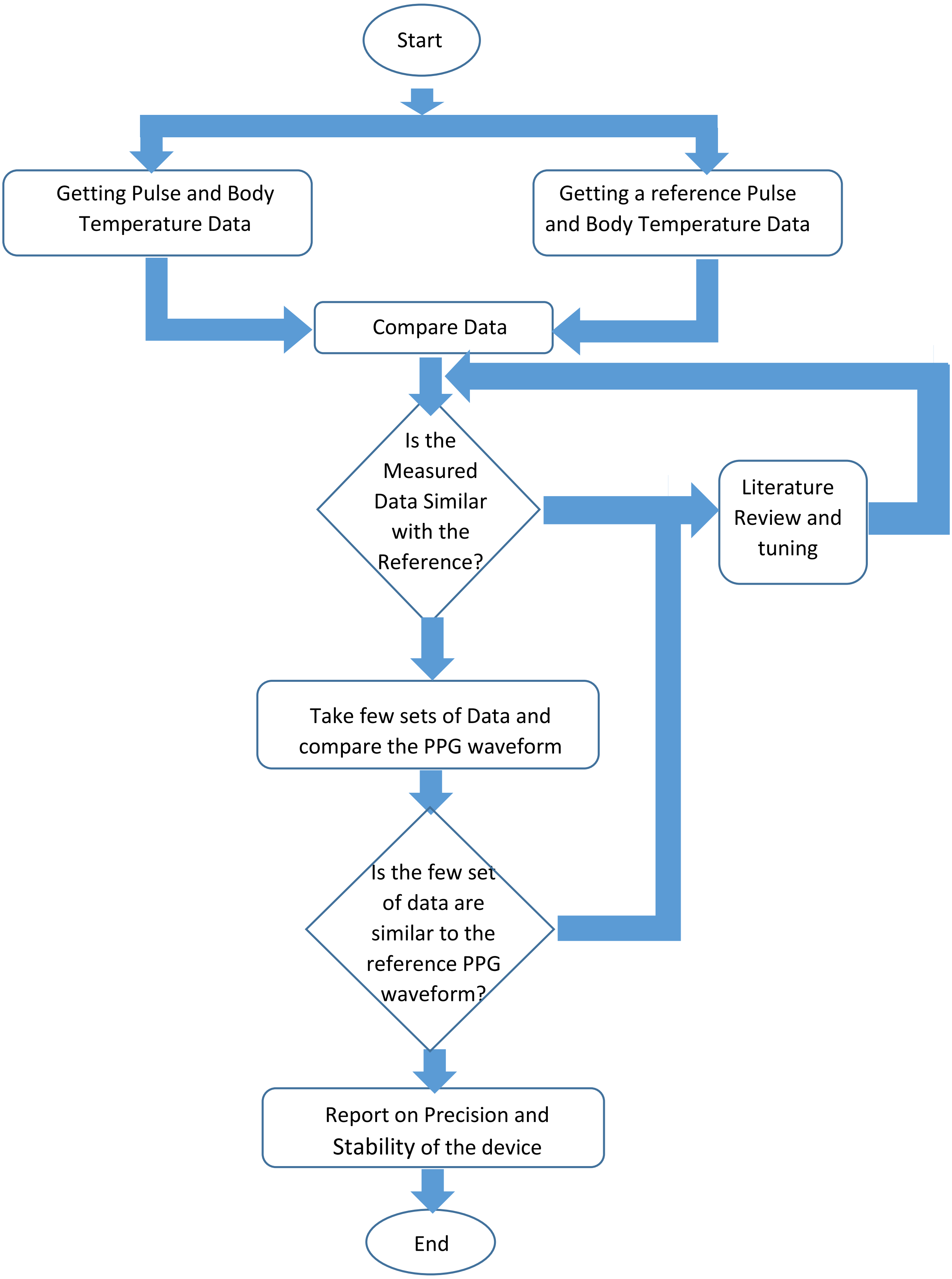


FIGURE 6: Flow Chart of Verification Process

CHAPTER 4

RESULT AND DISCUSSION

The result of this project is divided into Temperature, Heart Rate, and Photoplethysmogram sections as each of them need to be verified separately. Note that, all the verification process are assumed to be tested under an optimal environment where all other parameter such as environment condition and the subject will be constant.

4.1 Temperature Verification

Temperature verification is carried out by taking measurement at the same place and time. This is to minimize the variation body temperature while maintaining constant room temperature. All the result are tabulated in Table 2.

TABLE 2: Temperature Verification Result

System Temperature Measurement (°C)	Digital Thermometer Measurement (°C)	Percentage Error (%)
36.5	37.2	1.8
36.	36.7	1.9
36.0	36.3	0.8
37.0	37.5	1.3
38.0	38.2	0.5

4.2 Heart Rate Verification

Heart Rate verification is almost the same step as verifying temperature but using OMRON HEM-7203 Automated Pressure Monitor. All the result are tabulated in Table 3.

TABLE 3: Heart Rate Verification Result

System Hear Rate Measurement (BPM)	OMRON Puressure Measurement (BPM)	Percentage Error (%)
81	82	1.2
78	79	1.3
68	70	2.9
75	77	2.6
71	72	1.4

4.3 Mean Percentage Error Calculation

To provide meaningful data and understand the accuracy of the system, Mean percentage error for both temperature and pulse rate are calculated with Equation 6.

$$Mean\ Percentage\ Error = \frac{\sum_{i=1}^{N} Error(i)}{N} \times 100\ \%$$

EQUATION 6: Mean Percentage Error Calculation

4.4 Photoplethysmogram Verification

Photoplethysmogram (PPG) verification involve in comparing the processing software of Android Application and Computer. The PPG Processing Software in Computer was developed by the manufacturer of the PPG pulse sensor. Verification of PPG waveform is achieved by processing the raw data using computer and smartphone together in a specific time frame. The result can then be analyze as the plotted PPG waveform shown in Figure 7 and Figure 8.

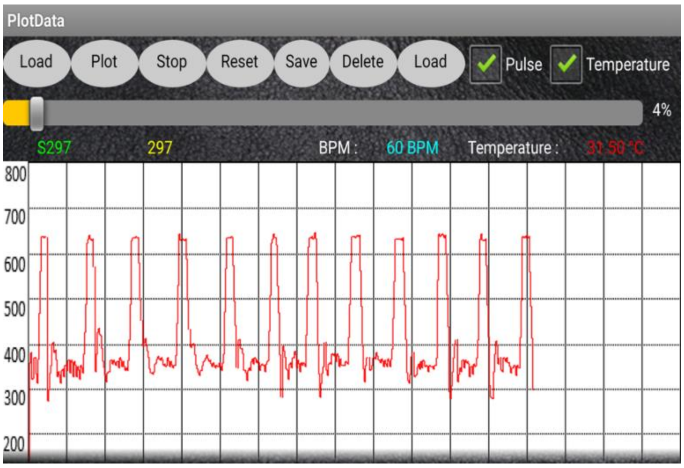


FIGURE 7: PPG Waveform plotted by monitoring application in Smartphone

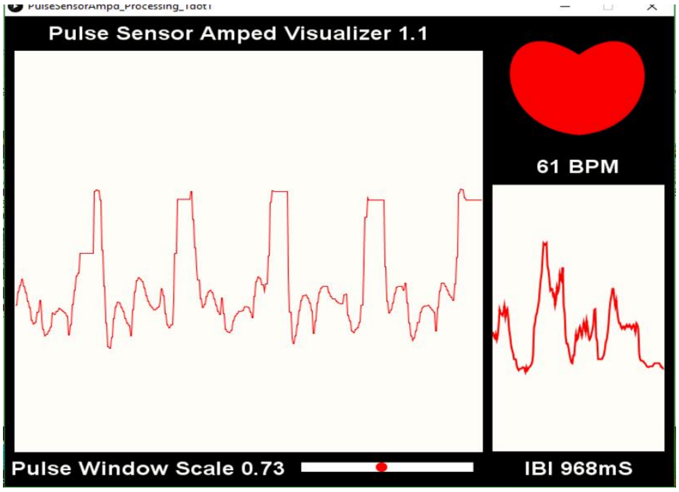


FIGURE 8: PPG waveform plotted using Computer

4.5 Discussion

In Table II of the temperature verification, the computed Mean Percentage Error of all five result is 1.30%. Although this result is very close to each other but the difference of this two devices is related to the accuracy of the device itself. This is because Digital Thermometer have ±0.1°C accuracy while DS18B20 have ±0.5°C accuracy which means our temperature sensor will only be increment if the temperature difference is more than 0.5°C. Besides, DS18B20 is not isolated properly from environment temperature while the verification is performed. This is because the temperature sensor calculation will be much more complex and the accuracy is not guaranteed

As for Table III of the heart rate verification, the computed Mean Percentage Error of all five result is 1.86 %. This result is very close to each other but the difference of this two devices might due to the different area of sensing. OMRON HEM-7203 Automated Pressure Monitor is a certified pressure monitor which designed to measure the heart rate at our upper arm. On the other hand, PPG pulse sensor can only sense the heart rate around the part of the body which is rich is capillary tissue. In this case, an ear clip is used to attach the PPG sensor and monitor over the pulse rate. Besides, these two devices have a different concept of detecting pulse and computation of our heart rate. This will also result in percentage error as it shown in Table III.

For Photoplethysmogram (PPG) verification, the plotted waveform shown in Figure 7 and Figure 8 are similar to each other. The only difference in the figures is because of different scaling when the graph is plot. As the computer processing software is developed by the manufacturer, we can conclude that the PPG Monitoring System that designed is able to achieve the same result as the manufacturer.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

As the conclusion, the main idea of the project is to design a continuous temperature and heart rate monitor. All the data that send using the system have all verified by using another certified Digital Thermometer and OMRON HEM-7203 Automated Blood Pressure Monitor. Under the optimal condition where the sensors have successfully detected the information, the percentage error of temperature and beat per minute are 1.30% and 1.86% respectively. For temperature, the percentage error is caused mainly because the DS1820 have $\pm 0.5℃$ increment while the temperature sensor has $\pm 0.1℃$ increment. Also, DS1820 does not have a proper temperature insulation to the surrounding. For heart pulses, the system displayed almost perfect plotting of PPG data, however, the comparison of BPM does not seem to be perfect because the OMRON Blood Pressure Monitor using the different concept to detect pulse rate.

5.2 Recommendation

For future analysis, a more accurate device such as Cardiac Monitor is needed to verify the performance of the system over longer times. This will help in improving the method and techniques to measure data and calibrate for more accurate reading.

Also, the analysis of PPG signal needs to be verified by accredited medical institution to prove the usefulness of this application.

REFERENCES

[1] W. Q. Lindh, M. Pooler, C. D. Tamparo, and B. M. Dahl, *Delmar's Comprehensive Medical Assisting: Administrative and Clinical Competencies*: Cengage Learning, 2009.

[2] S. Walsh and E. King, *Pulse Diagnosis: A Clinical Guide*: Elsevier Health Sciences UK, 2007.

[3] L. McCallum and D. Higgins, "Measuring body temperature," *Nursing times,* vol. 108, pp. 20-22, 2011.

[4] J. L. Fuzy, *The Nursing Assistant's Handbook*: Hartman Publishing, Incorporated, 2003.

[5] I. Blumenthal, "What parents think of fever," *Family practice,* vol. 15, pp. 513-518, 1998.

[6] A. Tarun. (2013). *Types of Temperature Sensors and Their working Principles | Features*. Available: http://www.elprocus.com/temperature-sensors-types-working-operation/

[7] B. Baker, "Temperature sensing technologies," *AN679, Microchip Technology Inc,* 1998.

[8] W. Kester, J. Bryant, and W. Jung, "SECTION 7 TEMPERATURE SENSORS," *Practical Design Techniques for Sensor Signal Conditioning, Analog Devices,* 1999.

[9] K. H. Shelley, "Photoplethysmography: beyond the calculation of arterial oxygen saturation and heart rate," *Anesthesia & Analgesia,* vol. 105, pp. S31-S36, 2007.

APPENDICES

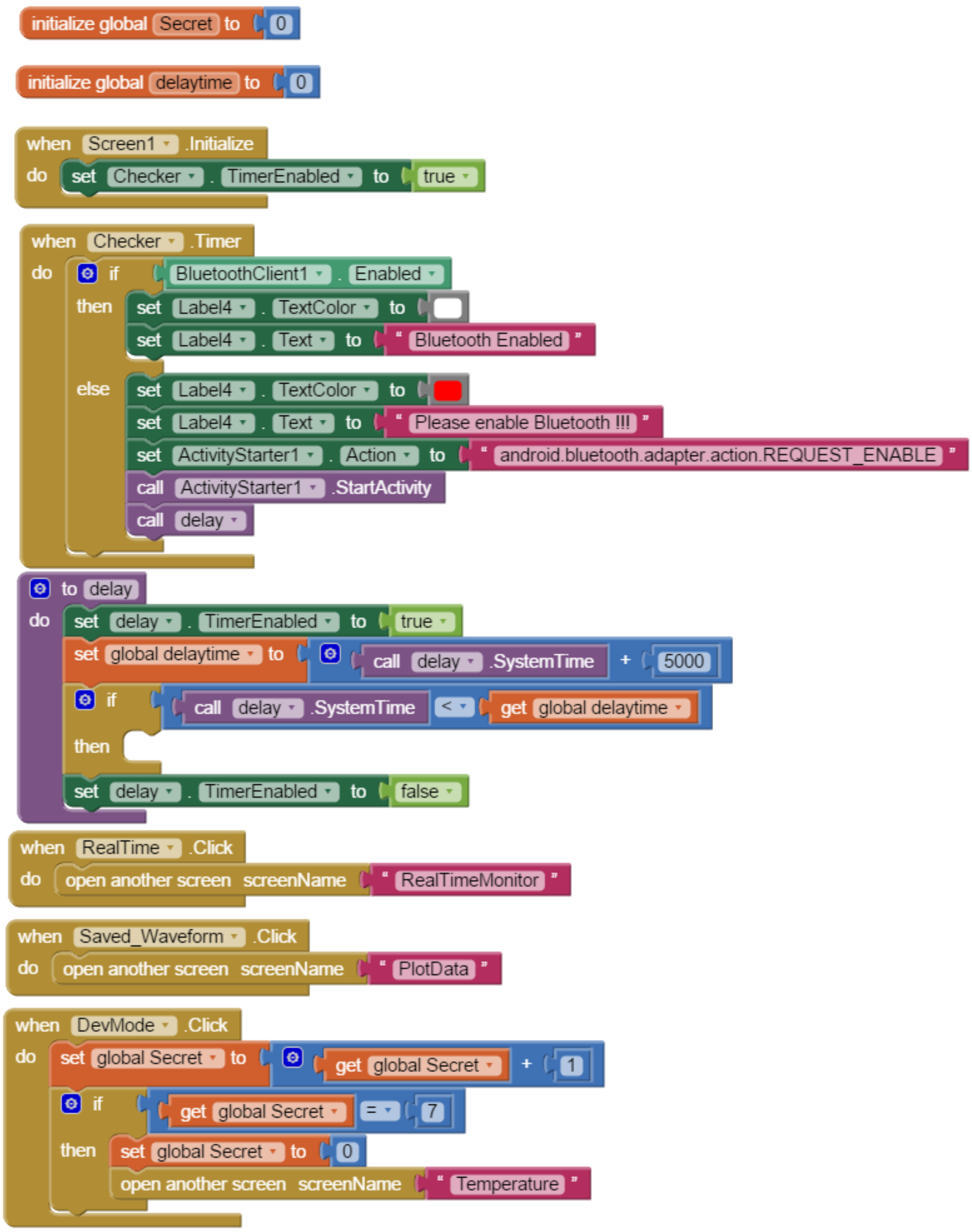
APPENDIX I: ARDUINO CODE

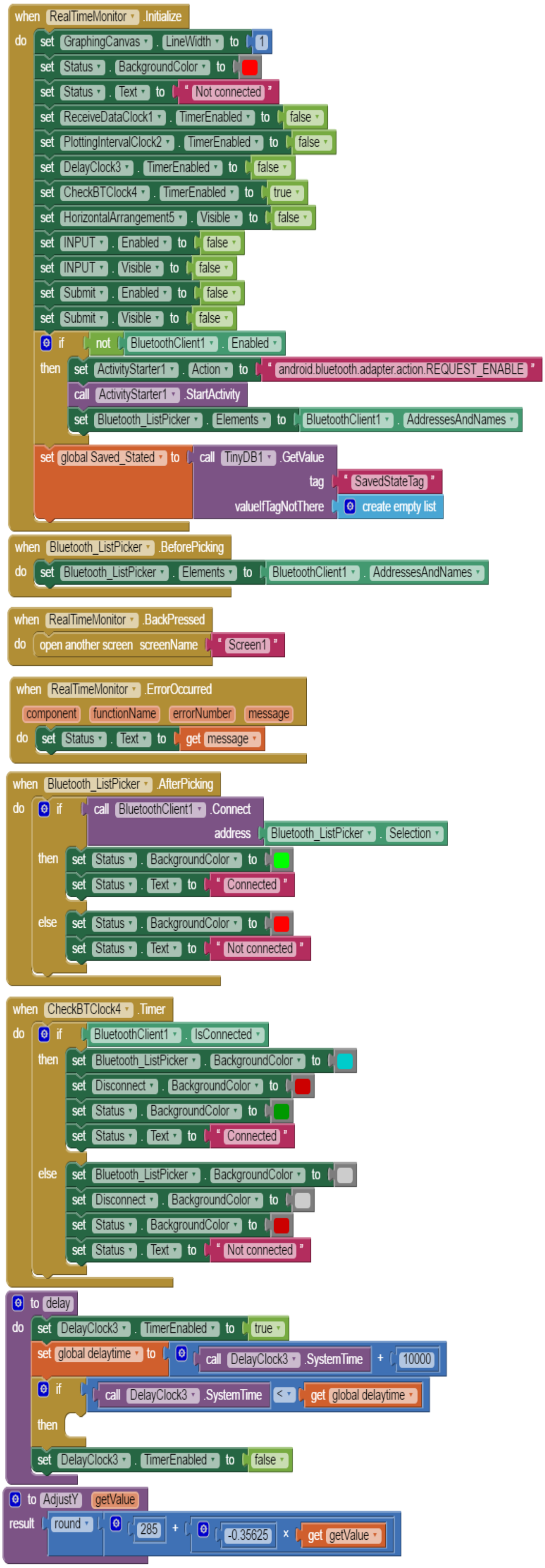
```
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

unsigned long interval = 1000;
unsigned long currentMillis = 0;
unsigned long previousMillis = 0;

#include <SoftwareSerial.h>
SoftwareSerial Keng(0,1);

int pulsePin = 0;
int blinkPin = 13;
int fadePin = 5;
int fadeRate = 0;

volatile int BPM;                 // int that holds raw Analog in 0. updated every 2mS
volatile int Signal;              // holds the incoming raw data
volatile int IBI = 600;           // int that holds the time interval between beats! Must be seeded!
volatile boolean Pulse = false;   // "True" when User's live heartbeat is detected. "False" when not a "live beat".
volatile boolean QS = false;      // becomes true when Arduoino finds a beat.

static boolean serialVisual = false;   // Set to 'false' by Default.  Re-set to 'true' to see Arduino Serial Monitor ASCII Visual Pulse

void setup(){
  pinMode(blinkPin,OUTPUT);        // pin that will blink to your heartbeat!
  pinMode(fadePin,OUTPUT);         // pin that will fade to your heartbeat!
  Serial.begin(115200);            // we agree to talk fast!
  interruptSetup();                // sets up to read Pulse Sensor signal every 2mS
  Keng.begin(115200);
  sensors.begin();
}
void loop(){
   serialOutput() ;

  if (QS == true){
      digitalWrite(blinkPin,HIGH);
      fadeRate = 255;
      serialOutputWhenBeatHappens();   // A Beat Happened, Output that to serial.
      QS = false;                      // reset the Quantified Self flag for next time
  }
  ledFadeToBeat();                     // Makes the LED Fade Effect Happen
  delay(20);                           //  take a break
  currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;
    temp();
  }
}
void ledFadeToBeat(){
    fadeRate -= 15;                         //  set LED fade value
    fadeRate = constrain(fadeRate,0,255);   //  keep LED fade value from going into negative numbers!
    analogWrite(fadePin,fadeRate);          //  fade LED
  }
void temp(){
    sensors.requestTemperatures();
    Serial.print("T");
    Serial.println(sensors.getTempCByIndex(0));
}
void serialOutput(){   // Decide How To Output Serial.
 if (serialVisual == true){
     arduinoSerialMonitorVisual('-', Signal);   // goes to function that makes Serial Monitor Visualizer
 } else{
      sendDataToSerial('S', Signal);     // goes to sendDataToSerial function
 }
```

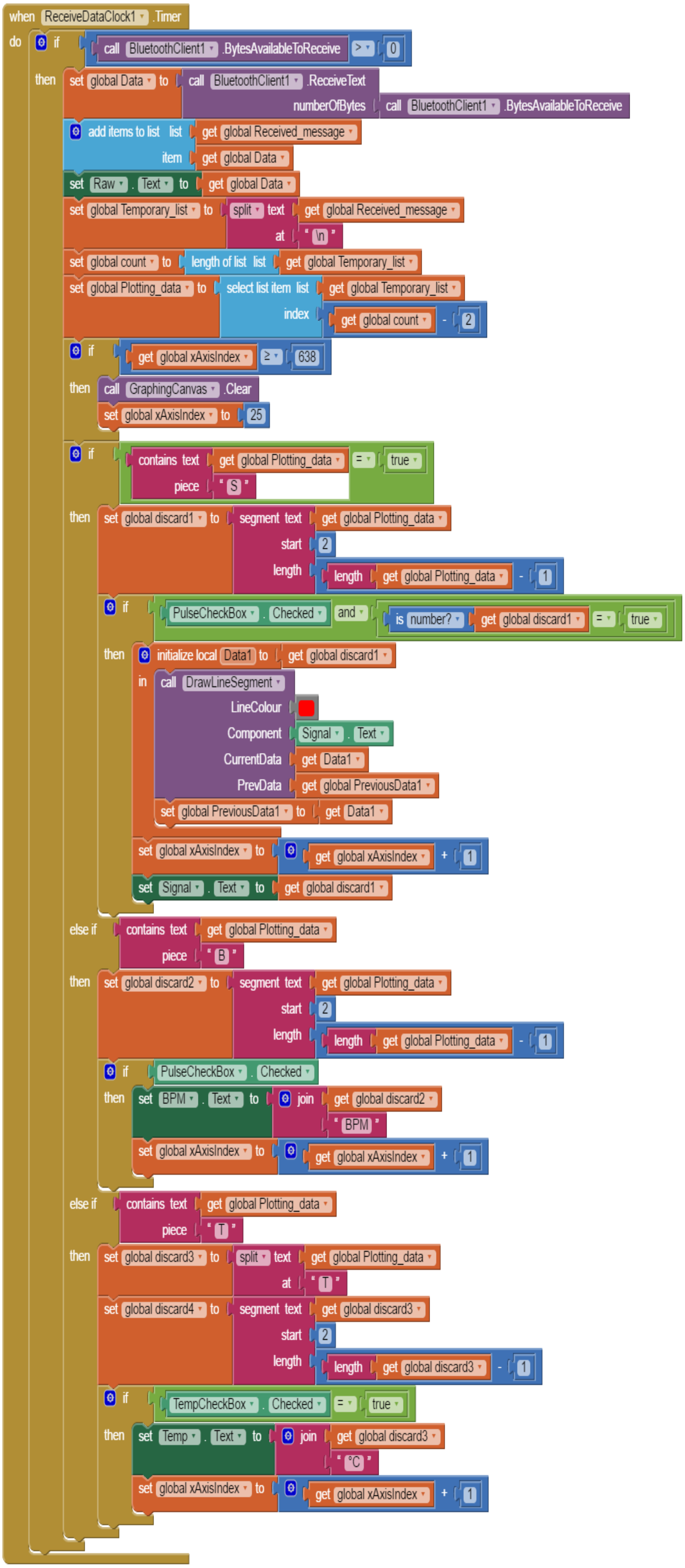
```
}
void serialOutputWhenBeatHappens(){
 if (serialVisual == true){            //  Code to Make the Serial Monitor Visualizer Work
    Serial.print("*** Heart-Beat Happened *** ");
    Serial.print("BPM: ");
    Serial.print(BPM);
    Serial.print("  ");
 } else{
        sendDataToSerial('B',BPM);   // send heart rate with a 'B' prefix
        sendDataToSerial('Q',IBI);   // send time between beats with a 'Q' prefix
 }
}
void sendDataToSerial(char symbol, int data ){
    Serial.print(symbol);
    Serial.println(data);
  }
void arduinoSerialMonitorVisual(char symbol, int data ){
const int sensorMin = 0;      // sensor minimum, discovered through experiment
const int sensorMax = 1024;    // sensor maximum, discovered through experiment
int sensorReading = data;
int range = map(sensorReading, sensorMin, sensorMax, 0, 11);
  switch (range) {
  case 0:
    Serial.println("");
    break;
  case 1:
    Serial.println("---");
    break;
  case 2:
    Serial.println("------");
    break;
  case 3:
    Serial.println("---------");
    break;
  case 4:
    Serial.println("------------");
    break;
  case 5:
    Serial.println("--------------|-");
    break;
  case 6:
    Serial.println("--------------|---");
    break;
  case 7:
    Serial.println("--------------|-------");
    break;
  case 8:
    Serial.println("--------------|----------");
    break;
  case 9:
    Serial.println("--------------|----------------");
    break;
  case 10:
    Serial.println("--------------|-------------------");
    break;
  case 11:
    Serial.println("--------------|-----------------------");
    break;
  }
}

volatile int rate[10];                    // array to hold last ten IBI values
volatile unsigned long sampleCounter = 0;          // used to determine pulse timing
volatile unsigned long lastBeatTime = 0;           // used to find IBI
volatile int P =512;                      // used to find peak in pulse wave, seeded
volatile int T = 512;                     // used to find trough in pulse wave, seeded
volatile int thresh = 525;                // used to find instant moment of heart beat, seeded
volatile int amp = 100;                   // used to hold amplitude of pulse waveform, seeded
volatile boolean firstBeat = true;        // used to seed rate array so we startup with reasonable BPM
volatile boolean secondBeat = false;      // used to seed rate array so we startup with reasonable BPM

void interruptSetup(){
TCCR2A = 0x02;     // DISABLE PWM ON DIGITAL PINS 3 AND 11, AND GO INTO CTC MODE
TCCR2B = 0x06;     // DON'T FORCE COMPARE, 256 PRESCALER
OCR2A = 0X7C;      // SET THE TOP OF THE COUNT TO 124 FOR 500Hz SAMPLE RATE
```

```
  TIMSK2 = 0x02;     // ENABLE INTERRUPT ON MATCH BETWEEN TIMER2 AND OCR2A
  sei();             // MAKE SURE GLOBAL INTERRUPTS ARE ENABLED
}

// THIS IS THE TIMER 2 INTERRUPT SERVICE ROUTINE.
// Timer 2 makes sure that we take a reading every 2 miliseconds
ISR(TIMER2_COMPA_vect){                       // triggered when Timer2 counts to 124
  cli();                                      // disable interrupts while we do this
  Signal = analogRead(pulsePin);              // read the Pulse Sensor
  sampleCounter += 2;                         // keep track of the time in mS with this variable
  int N = sampleCounter - lastBeatTime;       // monitor the time since the last beat to avoid noise

    //  find the peak and trough of the pulse wave
  if(Signal < thresh && N > (IBI/5)*3){       // avoid dichrotic noise by waiting 3/5 of last IBI
    if (Signal < T){                          // T is the trough
      T = Signal;                             // keep track of lowest point in pulse wave
    }
  }

  if(Signal > thresh && Signal > P){          // thresh condition helps avoid noise
    P = Signal;                               // P is the peak
  }                                           // keep track of highest point in pulse wave

  if (N > 250){                               // avoid high frequency noise
    if ( (Signal > thresh) && (Pulse == false) && (N > (IBI/5)*3) ){
      Pulse = true;                           // set the Pulse flag when we think there is a pulse
      digitalWrite(blinkPin,HIGH);            // turn on pin 13 LED
      IBI = sampleCounter - lastBeatTime;     // measure time between beats in mS
      lastBeatTime = sampleCounter;           // keep track of time for next pulse

      if(secondBeat){                         // if this is the second beat, if secondBeat == TRUE
        secondBeat = false;                   // clear secondBeat flag
        for(int i=0; i<=9; i++){              // seed the running total to get a realisitic BPM at startup
        rate[i] = IBI;
        }
      }

      if(firstBeat){                          // if it's the first time we found a beat, if firstBeat == TRUE
        firstBeat = false;                    // clear firstBeat flag
        secondBeat = true;                    // set the second beat flag
        sei();                                // enable interrupts again
        return;                               // IBI value is unreliable so discard it
      }
      // keep a running total of the last 10 IBI values
      word runningTotal = 0;                  // clear the runningTotal variable
      for(int i=0; i<=8; i++){                // shift data in the rate array
        rate[i] = rate[i+1];                  // and drop the oldest IBI value
        runningTotal += rate[i];              // add up the 9 oldest IBI values
      }
      rate[9] = IBI;                          // add the latest IBI to the rate array
      runningTotal += rate[9];                // add the latest IBI to runningTotal
      runningTotal /= 10;                     // average the last 10 IBI values
      BPM = 60000/runningTotal;               // how many beats can fit into a minute? that's BPM!
      QS = true;                              // set Quantified Self flag
    }
  }
  if (Signal < thresh && Pulse == true){   // when the values are going down, the beat is over
   digitalWrite(blinkPin,LOW);            // turn off pin 13 LED
    Pulse = false;                         // reset the Pulse flag so we can do it again
    amp = P - T;                           // get amplitude of the pulse wave
    thresh = amp/2 + T;                    // set thresh at 50% of the amplitude
    P = thresh;                            // reset these for next time
    T = thresh;
  }
  if (N > 2500){                           // if 2.5 seconds go by without a beat
    thresh = 512;                          // set thresh default
    P = 512;                               // set P default
    T = 512;                               // set T default
    lastBeatTime = sampleCounter;          // bring the lastBeatTime up to date
    firstBeat = true;                      // set these to avoid noise
    secondBeat = false;                    // when we get the heartbeat back
  }
  sei();                                   // enable interrupts when youre done!
}
```

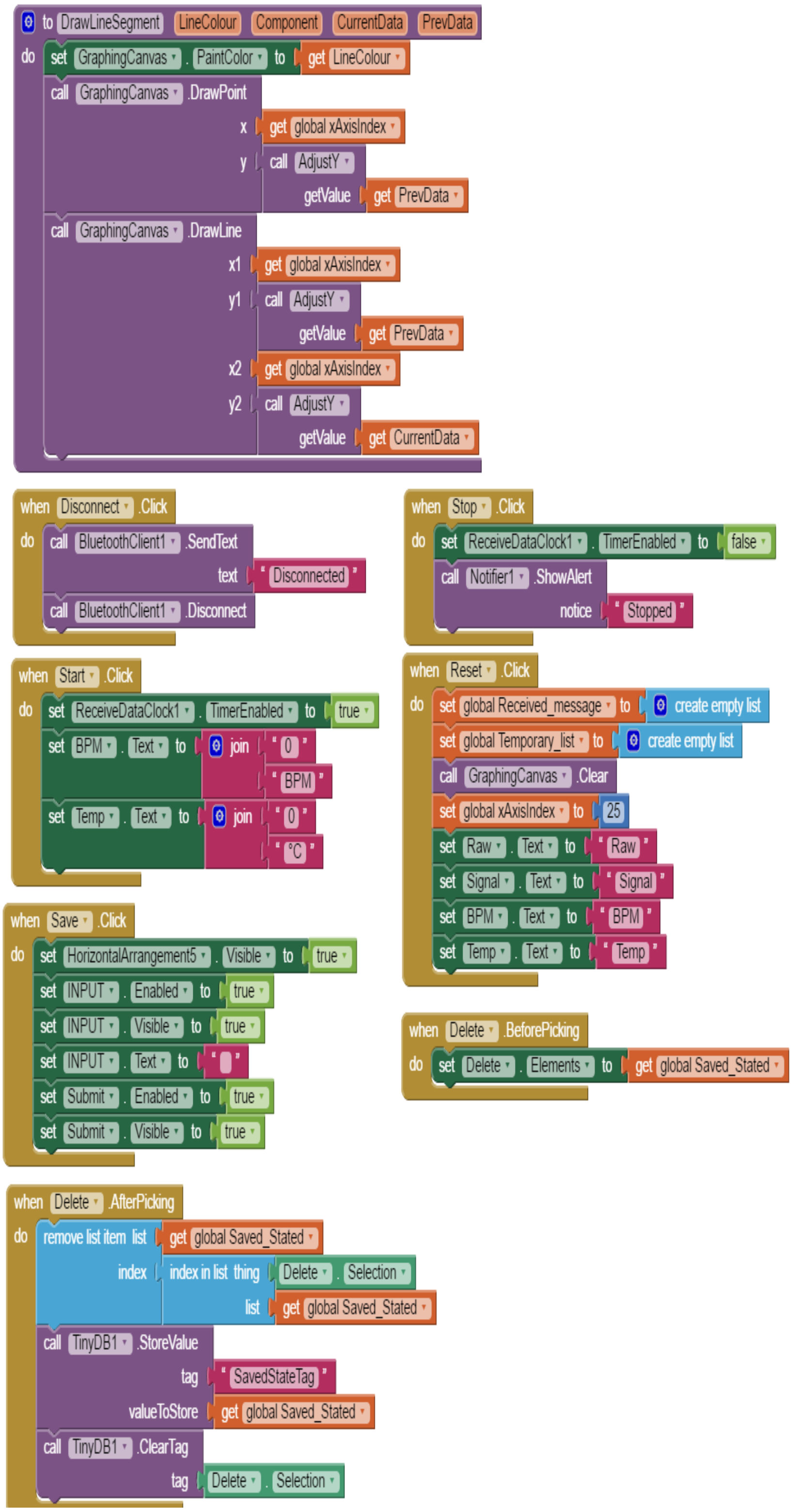
APPENDIX II: MIT APP INVENTOR CODE



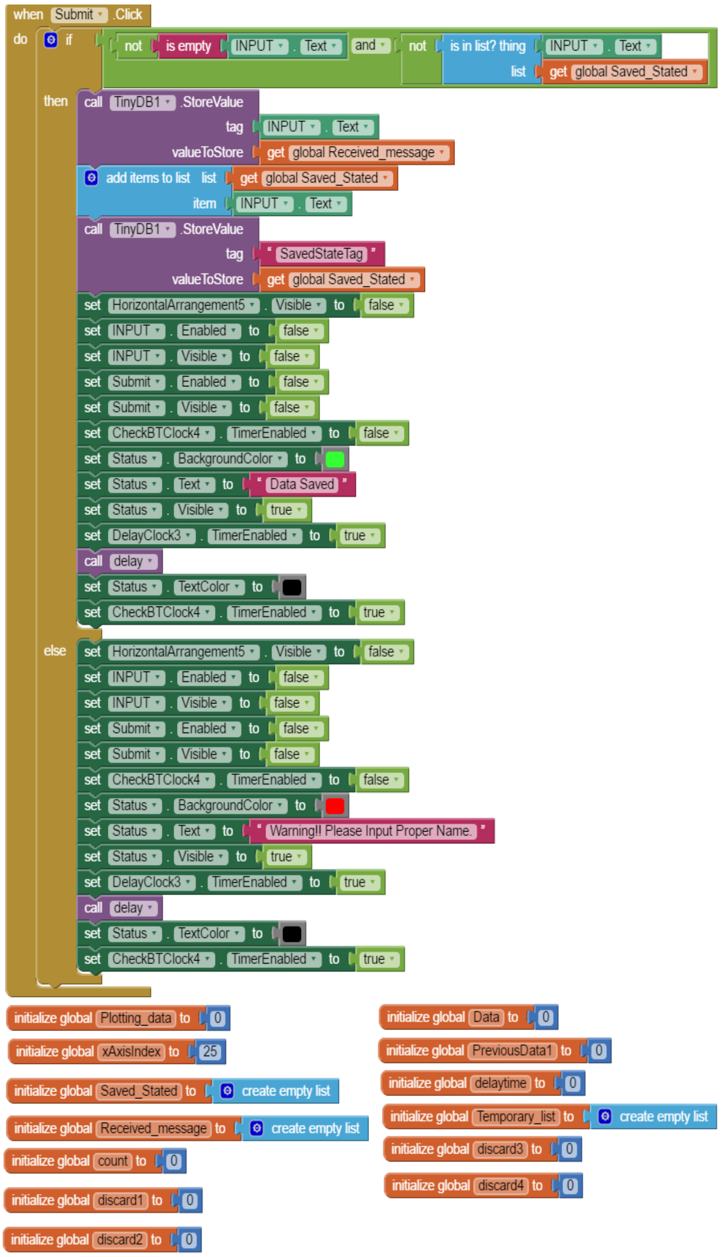

when RealTimeMonitor .Initialize
do set GraphingCanvas . LineWidth to 1
set Status . BackgroundColor to
set Status . Text to " Not connected "
set ReceiveDataClock1 . TimerEnabled to false
set PlottingIntervalClock2 . TimerEnabled to false
set DelayClock3 . TimerEnabled to false
set CheckBTClock4 . TimerEnabled to true
set HorizontalArrangement5 . Visible to false
set INPUT . Enabled to false
set INPUT . Visible to false
set Submit . Enabled to false
set Submit . Visible to false
if not BluetoothClient1 . Enabled
then set ActivityStarter1 . Action to " android.bluetooth.adapter.action.REQUEST_ENABLE "
call ActivityStarter1 .StartActivity
set Bluetooth_ListPicker . Elements to BluetoothClient1 . AddressesAndNames
set global Saved_Stated to call TinyDB1 .GetValue
tag " SavedStateTag "
valueIfTagNotThere create empty list
when Bluetooth_ListPicker .BeforePicking
do set Bluetooth_ListPicker . Elements to BluetoothClient1 . AddressesAndNames
when RealTimeMonitor .BackPressed
do open another screen screenName " Screen1 "
when RealTimeMonitor .ErrorOccurred
component functionName errorNumber message
do set Status . Text to get message
when Bluetooth_ListPicker .AfterPicking
do if call BluetoothClient1 .Connect
address Bluetooth_ListPicker . Selection
then set Status . BackgroundColor to
set Status . Text to " Connected "
else set Status . BackgroundColor to
set Status . Text to " Not connected "
when CheckBTClock4 .Timer
do if BluetoothClient1 . IsConnected
then set Bluetooth_ListPicker . BackgroundColor to
set Disconnect . BackgroundColor to
set Status . BackgroundColor to
set Status . Text to " Connected "
else set Bluetooth_ListPicker . BackgroundColor to
set Disconnect . BackgroundColor to
set Status . BackgroundColor to
set Status . Text to " Not connected "
to delay
do set DelayClock3 . TimerEnabled to true
set global delaytime to call DelayClock3 .SystemTime + 10000
if call DelayClock3 .SystemTime < get global delaytime
then
set DelayClock3 . TimerEnabled to false
to AdjustY getValue
result round 285 + -0.35625 × get getValue



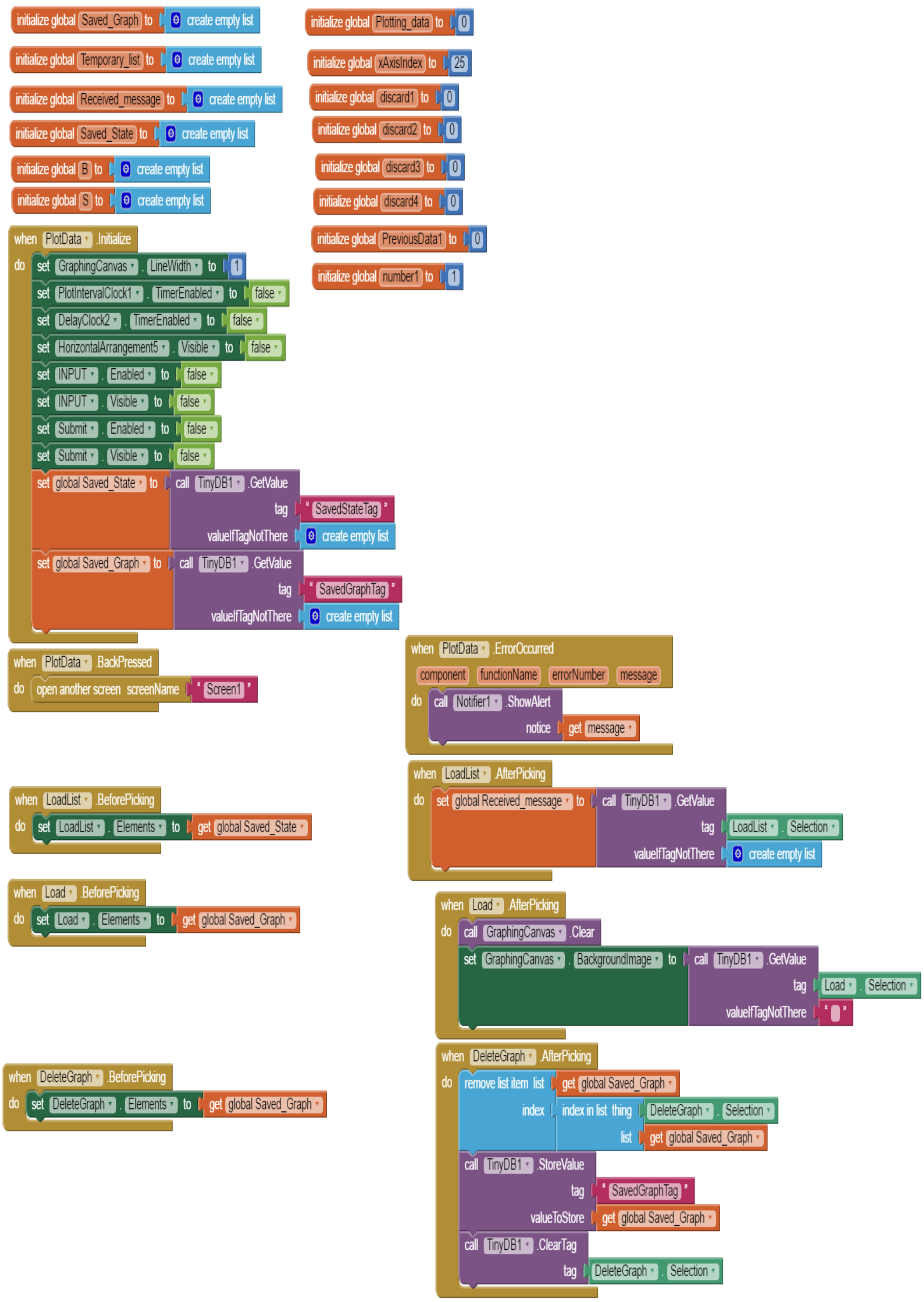
when ReceiveDataClock1 .Timer
do if call BluetoothClient1 .BytesAvailableToReceive > 0
then set global Data to call BluetoothClient1 .ReceiveText
numberOfBytes call BluetoothClient1 .BytesAvailableToReceive
add items to list list get global Received_message
item get global Data
set Raw . Text to get global Data
set global Temporary_list to split text get global Received_message
at " \n "
set global count to length of list list get global Temporary_list
set global Plotting_data to select list item list get global Temporary_list
index get global count - 2
if get global xAxisIndex ≥ 638
then call GraphingCanvas .Clear
set global xAxisIndex to 25
if contains text get global Plotting_data = true
piece " S "
then set global discard1 to segment text get global Plotting_data
start 2
length length get global Plotting_data - 1
if PulseCheckBox . Checked and is number? get global discard1 = true
then initialize local Data1 to get global discard1
in call DrawLineSegment
LineColour
Component Signal . Text
CurrentData get Data1
PrevData get global PreviousData1
set global PreviousData1 to get Data1
set global xAxisIndex to get global xAxisIndex + 1
set Signal . Text to get global discard1
else if contains text get global Plotting_data
piece " B "
then set global discard2 to segment text get global Plotting_data
start 2
length length get global Plotting_data - 1
if PulseCheckBox . Checked
then set BPM . Text to join get global discard2
" BPM "
set global xAxisIndex to get global xAxisIndex + 1
else if contains text get global Plotting_data
piece " T "
then set global discard3 to split text get global Plotting_data
at " T "
set global discard4 to segment text get global discard3
start 2
length length get global discard3 - 1
if TempCheckBox . Checked = true
then set Temp . Text to join get global discard3
" °C "
set global xAxisIndex to get global xAxisIndex + 1



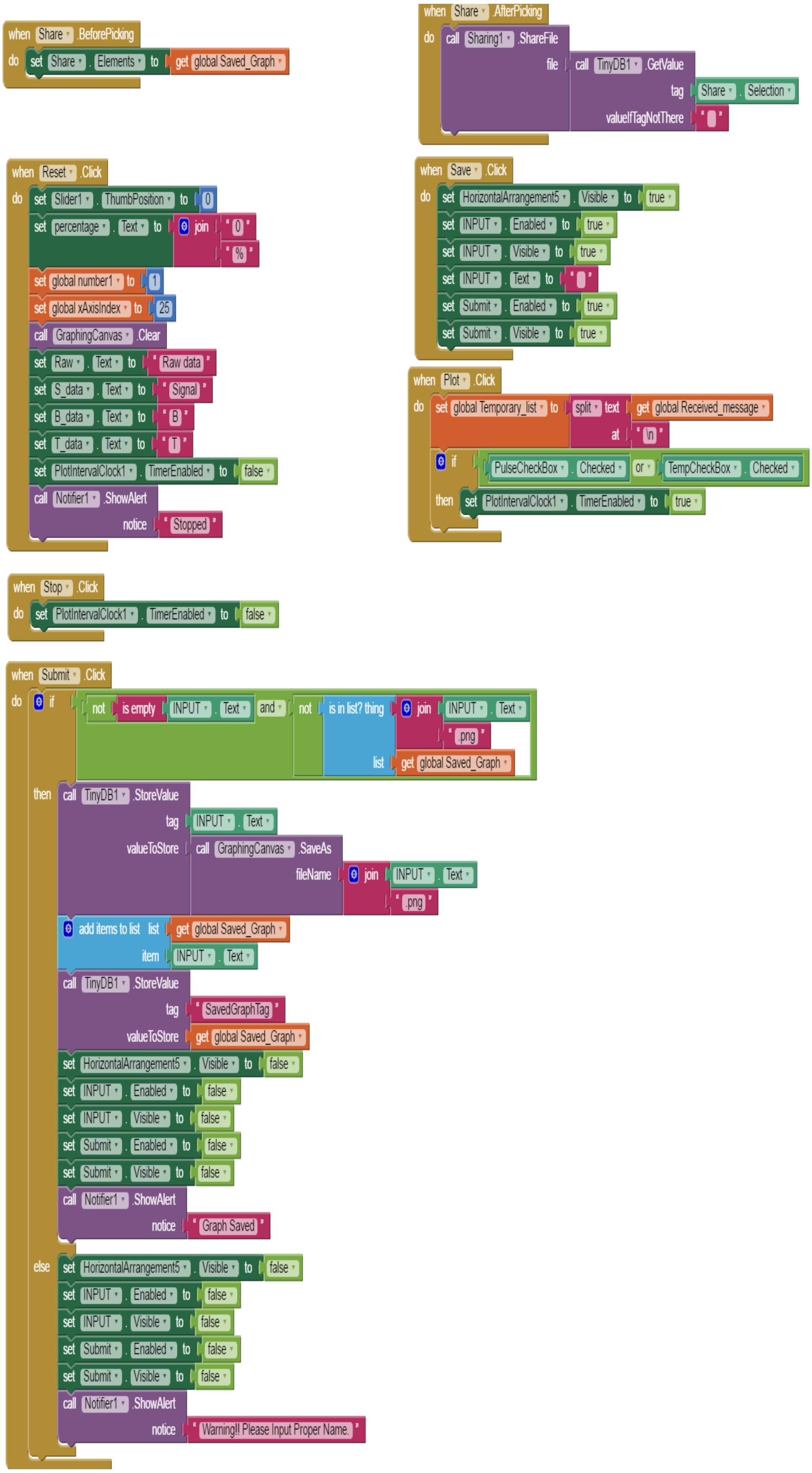
to DrawLineSegment LineColour Component CurrentData PrevData
do set GraphingCanvas . PaintColor to get LineColour
call GraphingCanvas .DrawPoint
x get global xAxisIndex
y call AdjustY
getValue get PrevData
call GraphingCanvas .DrawLine
x1 get global xAxisIndex
y1 call AdjustY
getValue get PrevData
x2 get global xAxisIndex
y2 call AdjustY
getValue get CurrentData
when Disconnect .Click
do call BluetoothClient1 .SendText
text " Disconnected "
call BluetoothClient1 .Disconnect
when Stop .Click
do set ReceiveDataClock1 . TimerEnabled to false
call Notifier1 .ShowAlert
notice " Stopped "
when Start .Click
do set ReceiveDataClock1 . TimerEnabled to true
set BPM . Text to join " 0 " " BPM "
set Temp . Text to join " 0 " " °C "
when Reset .Click
do set global Received_message to create empty list
set global Temporary_list to create empty list
call GraphingCanvas .Clear
set global xAxisIndex to 25
set Raw . Text to " Raw "
set Signal . Text to " Signal "
set BPM . Text to " BPM "
set Temp . Text to " Temp "
when Save .Click
do set HorizontalArrangement5 . Visible to true
set INPUT . Enabled to true
set INPUT . Visible to true
set INPUT . Text to " "
set Submit . Enabled to true
set Submit . Visible to true
when Delete .BeforePicking
do set Delete . Elements to get global Saved_Stated
when Delete .AfterPicking
do remove list item list get global Saved_Stated
index index in list thing Delete . Selection
list get global Saved_Stated
call TinyDB1 .StoreValue
tag " SavedStateTag "
valueToStore get global Saved_Stated
call TinyDB1 .ClearTag
tag Delete . Selection



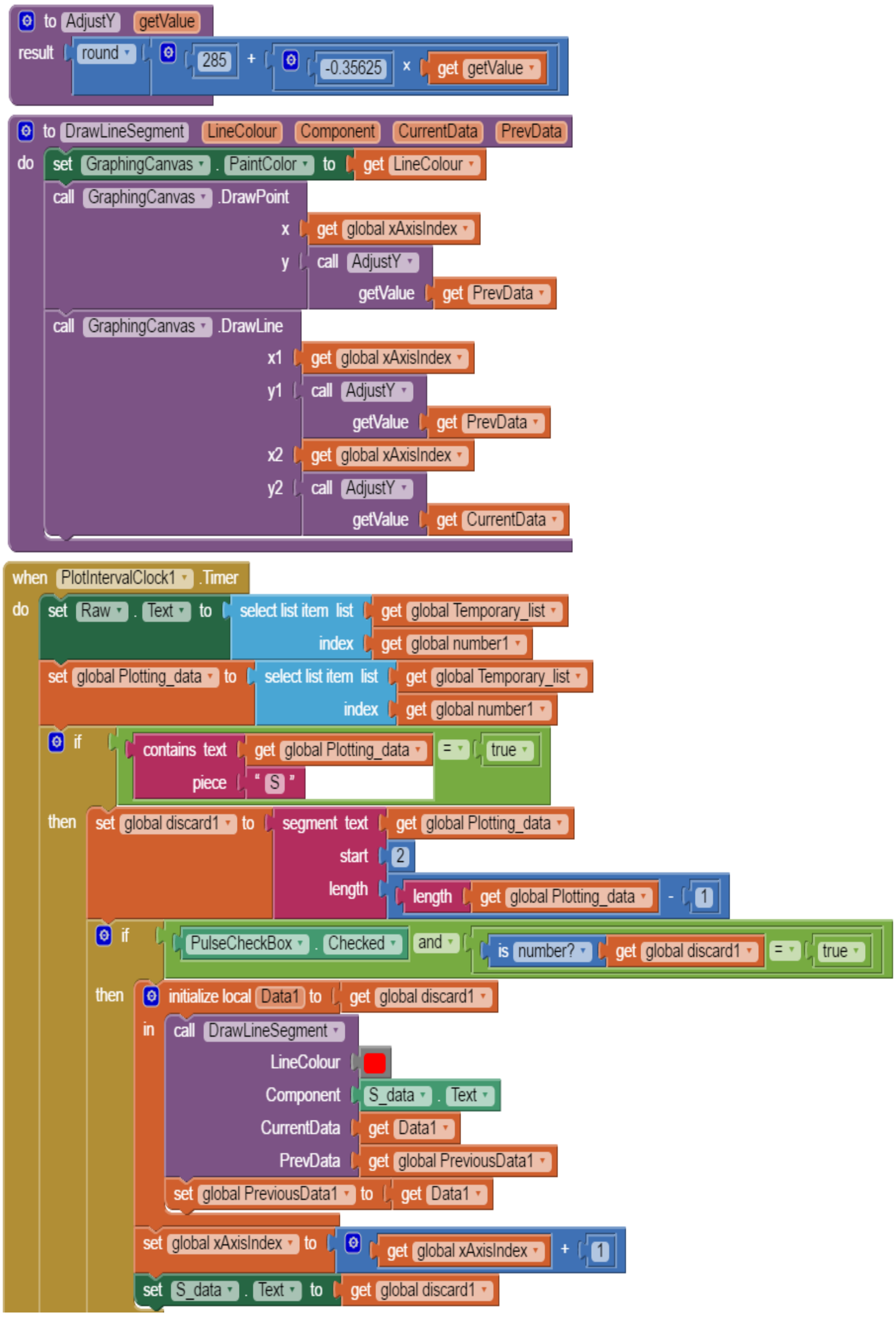
when Submit .Click
do if not is empty INPUT . Text and not is in list? thing INPUT . Text
list get global Saved_Stated
then call TinyDB1 .StoreValue
tag INPUT . Text
valueToStore get global Received_message
add items to list list get global Saved_Stated
item INPUT . Text
call TinyDB1 .StoreValue
tag " SavedStateTag "
valueToStore get global Saved_Stated
set HorizontalArrangement5 . Visible to false
set INPUT . Enabled to false
set INPUT . Visible to false
set Submit . Enabled to false
set Submit . Visible to false
set CheckBTClock4 . TimerEnabled to false
set Status . BackgroundColor to
set Status . Text to " Data Saved "
set Status . Visible to true
set DelayClock3 . TimerEnabled to true
call delay
set Status . TextColor to
set CheckBTClock4 . TimerEnabled to true
else set HorizontalArrangement5 . Visible to false
set INPUT . Enabled to false
set INPUT . Visible to false
set Submit . Enabled to false
set Submit . Visible to false
set CheckBTClock4 . TimerEnabled to false
set Status . BackgroundColor to
set Status . Text to " Warning!! Please Input Proper Name. "
set Status . Visible to true
set DelayClock3 . TimerEnabled to true
call delay
set Status . TextColor to
set CheckBTClock4 . TimerEnabled to true
initialize global Plotting_data to 0
initialize global xAxisIndex to 25
initialize global Saved_Stated to create empty list
initialize global Received_message to create empty list
initialize global count to 0
initialize global discard1 to 0
initialize global discard2 to 0
initialize global Data to 0
initialize global PreviousData1 to 0
initialize global delaytime to 0
initialize global Temporary_list to create empty list
initialize global discard3 to 0
initialize global discard4 to 0

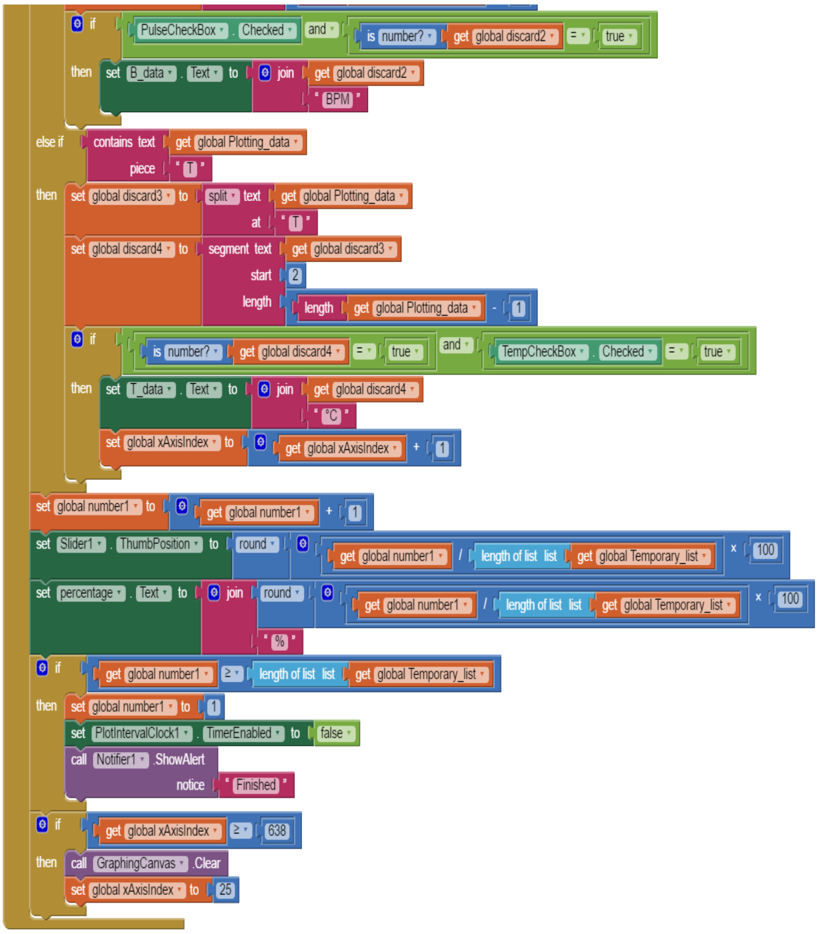


initialize global Saved_Graph to create empty list
initialize global Temporary_list to create empty list
initialize global Received_message to create empty list
initialize global Saved_State to create empty list
initialize global B to create empty list
initialize global S to create empty list
initialize global Plotting_data to 0
initialize global xAxisIndex to 25
initialize global discard1 to 0
initialize global discard2 to 0
initialize global discard3 to 0
initialize global discard4 to 0
initialize global PreviousData1 to 0
initialize global number1 to 1
when PlotData .Initialize
do set GraphingCanvas . LineWidth to 1
set PlotIntervalClock1 . TimerEnabled to false
set DelayClock2 . TimerEnabled to false
set HorizontalArrangement5 . Visible to false
set INPUT . Enabled to false
set INPUT . Visible to false
set Submit . Enabled to false
set Submit . Visible to false
set global Saved_State to call TinyDB1 .GetValue
tag " SavedStateTag "
valueIfTagNotThere create empty list
set global Saved_Graph to call TinyDB1 .GetValue
tag " SavedGraphTag "
valueIfTagNotThere create empty list
when PlotData .BackPressed
do open another screen screenName " Screen1 "
when PlotData .ErrorOccurred
component functionName errorNumber message
do call Notifier1 .ShowAlert
notice get message
when LoadList .AfterPicking
do set global Received_message to call TinyDB1 .GetValue
tag LoadList . Selection
valueIfTagNotThere create empty list
when LoadList .BeforePicking
do set LoadList . Elements to get global Saved_State
when Load .BeforePicking
do set Load . Elements to get global Saved_Graph
when Load .AfterPicking
do call GraphingCanvas .Clear
set GraphingCanvas . BackgroundImage to call TinyDB1 .GetValue
tag Load . Selection
valueIfTagNotThere " "
when DeleteGraph .BeforePicking
do set DeleteGraph . Elements to get global Saved_Graph
when DeleteGraph .AfterPicking
do remove list item list get global Saved_Graph
index index in list thing DeleteGraph . Selection
list get global Saved_Graph
call TinyDB1 .StoreValue
tag " SavedGraphTag "
valueToStore get global Saved_Graph
call TinyDB1 .ClearTag
tag DeleteGraph . Selection



when Share .BeforePicking
do set Share . Elements to get global Saved_Graph
when Share .AfterPicking
do call Sharing1 .ShareFile
file call TinyDB1 .GetValue
tag Share . Selection
valueIfTagNotThere " "
when Reset .Click
do set Slider1 . ThumbPosition to 0
set percentage . Text to join "0" "%"
set global number1 to 1
set global xAxisIndex to 25
call GraphingCanvas .Clear
set Raw . Text to "Raw data"
set S_data . Text to "Signal"
set B_data . Text to "B"
set T_data . Text to "T"
set PlotIntervalClock1 . TimerEnabled to false
call Notifier1 .ShowAlert
notice "Stopped"
when Save .Click
do set HorizontalArrangement5 . Visible to true
set INPUT . Enabled to true
set INPUT . Visible to true
set INPUT . Text to " "
set Submit . Enabled to true
set Submit . Visible to true
when Plot .Click
do set global Temporary_list to split text get global Received_message
at "\n"
if PulseCheckBox . Checked or TempCheckBox . Checked
then set PlotIntervalClock1 . TimerEnabled to true
when Stop .Click
do set PlotIntervalClock1 . TimerEnabled to false
when Submit .Click
do if not is empty INPUT . Text and not is in list? thing join INPUT . Text ".png"
list get global Saved_Graph
then call TinyDB1 .StoreValue
tag INPUT . Text
valueToStore call GraphingCanvas .SaveAs
fileName join INPUT . Text ".png"
add items to list list get global Saved_Graph
item INPUT . Text
call TinyDB1 .StoreValue
tag "SavedGraphTag"
valueToStore get global Saved_Graph
set HorizontalArrangement5 . Visible to false
set INPUT . Enabled to false
set INPUT . Visible to false
set Submit . Enabled to false
set Submit . Visible to false
call Notifier1 .ShowAlert
notice "Graph Saved"
else set HorizontalArrangement5 . Visible to false
set INPUT . Enabled to false
set INPUT . Visible to false
set Submit . Enabled to false
set Submit . Visible to false
call Notifier1 .ShowAlert
notice "Warning!! Please Input Proper Name."


to AdjustY getValue
result round 285 + -0.35625 × get getValue
to DrawLineSegment LineColour Component CurrentData PrevData
do set GraphingCanvas . PaintColor to get LineColour
call GraphingCanvas .DrawPoint
x get global xAxisIndex
y call AdjustY
getValue get PrevData
call GraphingCanvas .DrawLine
x1 get global xAxisIndex
y1 call AdjustY
getValue get PrevData
x2 get global xAxisIndex
y2 call AdjustY
getValue get CurrentData
when PlotIntervalClock1 .Timer
do set Raw . Text to select list item list get global Temporary_list
index get global number1
set global Plotting_data to select list item list get global Temporary_list
index get global number1
if contains text get global Plotting_data
piece " S "
= true
then set global discard1 to segment text get global Plotting_data
start 2
length length get global Plotting_data - 1
if PulseCheckBox . Checked and is number? get global discard1 = true
then initialize local Data1 to get global discard1
in call DrawLineSegment
LineColour
Component S_data . Text
CurrentData get Data1
PrevData get global PreviousData1
set global PreviousData1 to get Data1
set global xAxisIndex to get global xAxisIndex + 1
set S_data . Text to get global discard1


if PulseCheckBox . Checked and is number? get global discard2 = true
then set B_data . Text to join get global discard2
"BPM"
else if contains text get global Plotting_data
piece "T"
then set global discard3 to split text get global Plotting_data
at "T"
set global discard4 to segment text get global discard3
start 2
length length get global Plotting_data - 1
if is number? get global discard4 = true and TempCheckBox . Checked = true
then set T_data . Text to join get global discard4
"°C"
set global xAxisIndex to get global xAxisIndex + 1
set global number1 to get global number1 + 1
set Slider1 . ThumbPosition to round get global number1 / length of list list get global Temporary_list × 100
set percentage . Text to join round get global number1 / length of list list get global Temporary_list × 100
"%"
if get global number1 ≥ length of list list get global Temporary_list
then set global number1 to 1
set PlotIntervalClock1 . TimerEnabled to false
call Notifier1 .ShowAlert
notice "Finished"
if get global xAxisIndex ≥ 638
then call GraphingCanvas .Clear
set global xAxisIndex to 25