

HEART MONITORING SYSTEM

Robotics and Mechatronics

Phebe Le

Test Plans and Test Logs

Session 1.5: 5/6/2023-23/6/23



1. Full outline of design

Aim: The main design will be segmented into several different parts. The first being an armband which will store the Arduino and Ethernet shield. The second section is a 'smartwatch' contraption which contains a breadboard, LCD, 2 LEDs (red and green), a humidity sensor and a piezo.

2. Armband: Arduino and Ethernet Shield

Logical aim: To create the logical components of the armband, there will be several steps. These being:

1. Implementation of the webserver. This will require the Ethernet shield to be configured to the network.
2. Integration of smartwatch data. The data from the smartwatch will need to be viewable in a web server
3. (If applicable) Creation of age profiles. Age profiles will be required to alert the user only when their age group BPM exceeds or beneath the average.

3. Smartwatch: Other components

Logical aim: To create the smartwatch, the logical design and implementation will be broken up into several different modules. These being:

1. Integration of sensors
 - a. Pulse sensor: This will require the initial test to see if it works, changing the serial graph data to a set variable data (BPM)
 - b. Humidity & temperature sensor: This will require an initial test to see if it works and is accurate (to a certain degree)
2. Integration of display components (LCD). The display should show the current BPM, temperature and humidity.
3. Integration of an alert system. The alert system will be triggered when an individual reaches a BPM that exceeds the average age range
 - a. If applicable, create custom profiles for each age range group

8/7/23

Step 3.1-2. a: Smartwatch with Pulse Sensor and LCD

Aim: The first test will be only the integration of pulse sensor and the LCD screen, due to the current unavailability of a humidity sensor. The code which will be used to test this will be < <https://techatronic.com/heart-beat-sensor-using-arduino-bpm-monitor/>>. This code contains a simplistic architecture of the required components for this test. If this test were to work, optimisation of the code would be required to reduce the recursive and lengthy size of the sketch.

Test 1 Results:

The results for this test were favourable after debugging sections of the sketch. The test resulted in an extended amount of data through both the serial monitor and the LCD screen. Yet, these results were not accurate. The tested BPM ranged from 0-260 at various stages of the test. In attempting to resolve the varied BPM, I found that placing pressure on the sensor and moving it in different locations changed the BPM. The pulse sensor also seemed to require a strong artery pulse which resulted in the pulse from one's finger, neck, arm to have varied accuracy, whilst the heart had a steady accuracy of 250bpm, making all arteries invalid. The BPM may need to be reprogrammed as the <PulseSensorPlayground.h> was inaccurate. Due to the macro level of the code being successful, the code will be optimised through functions, a class and pointer for an LED and speaker.

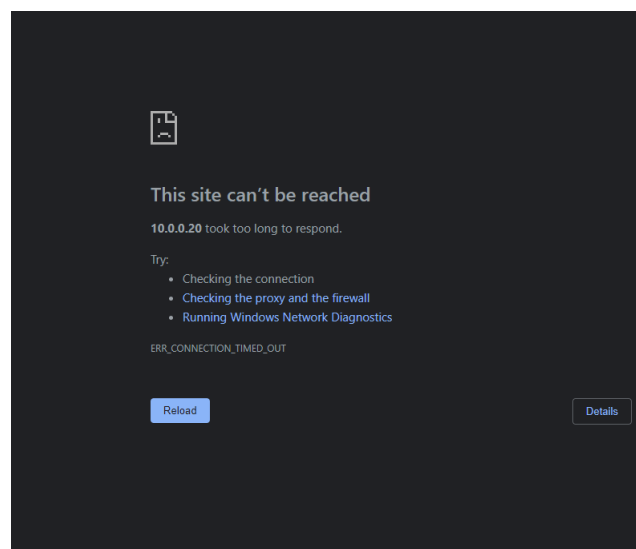
16/6/23

Step 2.1: Armband with Ethernet Shield web server

Aim: This test will be to test Ethernet Shield as well as create a functioning web server from the working Ethernet Shield. The starting code to test the Ethernet Shield and web server will be <<https://startingelectronics.com/tutorials/arduino/ethernet-shield-web-server-tutorial/basic-web-server/>>. If successful, the HTML of the web server will be edited to facilitate the output values of the sensors.

Test 1 Results:

The initial test was not favourable as the Arduino (that was correctly connected to my home router) was unable to create a web server. This test which showed flashing lights upon the shield showed that the shield was not broken but perhaps the Mac and Ip address (of the Arduino sketch) were incorrect. Based on this claim, the next test will be on changing the Mac address of the Arduino and the Ip address of the sketch to suite my home router.

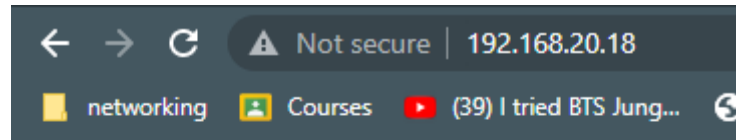


Test 2 Results:

To initially find my Ip address, the use of an Arduino example sketch named "DhcpAddressPrinter" which helps to create an available Ip address for the Arduino based on the current network. The output of this was a printed serial message with a confirmation of the connection and the Ip address to use (as seen below).

```
Initialize Ethernet with DHCP:  
My IP address: 192.168.20.18
```

Due to this successful outcome, I replaced the Mac and Ip address of the previous failed sketch with the Mac and Ip address of this exemplar sketch. This resulted in a successful output of the intended web server (as seen on the right).



Hello from Arduino!

A web page from the Arduino server

As a result, the next step will be to reiterate the current design of the web server and create a visual display for the output data of the smartwatch circuit. This screen should contain the heartbeat data, the humidity data and the temperature data.

In this image (See on right), the template for the sensors have been created. The HTML was created using basic knowledge of meta, heading, breaks etc. The style design of this web page was created from a CSS file from <Random Nerd Tutorials> (see CSS below).

Heart Monitoring System

Heart Rate:

Temperature:

Humidity:

Created by Phebe Le in Session 2 2023

```
body{ margin:60px 0px; padding:0px; text-align:center; }  
h1 { text-align: center; font-family:Arial, "Trebuchet MS", Helvetica, sans-serif; }  
h2 { text-align: center; font-family:Arial, "Trebuchet MS", Helvetica, sans-serif; }  
a { text-decoration:none; width:75px; height:50px; border-color:black; border-top:2px solid; border-bottom:2px solid; border-right:2px solid; border-left:2px solid; border-radius:10px 10px 10px 10px; -o-border-radius:10px 10px 10px 10px; -webkit-border-radius:10px 10px 10px 10px; font-family:"Trebuchet MS",Arial, Helvetica, sans-serif; -moz-border-radius:10px 10px 10px 10px; background-color:#293F5E; padding:8px; text-align:center; }  
a:link {color:white;} /* unvisited link */ a:visited {color:white;} /* visited link */ a:hover {color:white;} /* mouse over link */ a:active {color:white;} /* selected link */  
Visit http://randomnerdtutorials.com
```

This is what the current code looks like.

```
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = { 0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02 };
IPAddress ip(192,168,5,81);
EthernetServer server(80);

void setup()
{
    Ethernet.begin(mac, ip);
    server.begin();
}

void loop()
{
    EthernetClient client = server.available();

    if (client) { // got client?
        boolean currentLineIsBlank = true;
        while (client.connected()) {
            if (client.available()) {
                char c = client.read();
                if (c == '\n' && currentLineIsBlank) {
                    client.println("HTTP/1.1 200 OK");
                    client.println("Content-Type: text/html");
                    client.println();
                    client.println("<HTML>");
                    client.println("<HEAD>");
                    client.println("<meta charset='UTF-8'>");
                    client.println("<meta name='author' content='Phebe Le'>");
                    client.println("<meta http-equiv='X-UA-Compatible' content='IE-edge'>");
                    client.println("<meta name='viewport' content='width=device-width, initial-scale=1.0'>");
                    client.println("<meta name='apple-mobile-web-app-capable' content='yes' />");
                    client.println("<meta name='description' content='heartbeat, temperature, humidity'> ");
                    client.println("<meta name='apple-mobile-web-app-status-bar-style' content='black-translucent' />");
                    client.println("<link rel='stylesheet' type='text/css' href='https://randomnerdtutorials.com/ethernetcss.css' />");

                    client.println("<TITLE>Heart Monitoring System</TITLE>");
                    client.println("</HEAD>");
                    client.println("<BODY>");
                    client.println("<H1>Heart Monitoring System</H1>");
                    client.println("<hr />");
                    client.println("<hr />");
                    client.println("<H2>Heart Rate:</H2>");
                    client.println("<hr />");
                    client.println("<H2>Temperature:</H2>");
                    client.println("<hr />");
                    client.println("<H2>Humidity:</H2>");
                    client.println("<hr />");
                    client.println("<p>Created by Phebe Le in Session 2 2023</p>");
                    client.println("<hr />");
                    client.println("</BODY>");
                    client.println("</HTML>");
                    break;
                }
            }
            if (c == '\n') {
                currentLineIsBlank = true;
            }
            else if (c != '\r') {
                currentLineIsBlank = false;
            }
        }
        delay(1);
        client.stop();
    }
}
```

Currently there are two more major steps for the logical infrastructure. These being:

1. Allow the pulse sensor to read and write to an SD card
2. Humidity sensor/temperature sensor
 - a. Have a working sensor which can read data and display it on a Serial Monitor
 - b. Add the humidity sensor to the <Smartwatch> code and be able to view data from it
 - c. Allow the humidity sensor to read and write to an SD card
3. Combine the <Smartwatch> code and <Ethernet Armband> code

Currently, the physical design and infrastructure has not been completed yet.

18/6/23

Aim: This test will be to create a modular testing environment for the SD card to read and write data from the pulse sensor.

Test 1:

The implementation of the code was created based a combination of my previous knowledge of creating programs with SD cards and the heartbeat sensor code which was previously created. Due to my lacking knowledge this may contain errors and problems. On the left is the sketch.

Results of test 1:

This code was successful in creating a file and sensing the heartbeat (see results on the left). Based on presumption, the SD card should have written to the text file. Yet, this presumption will be tested before adding other functionalities such as malloc, vectors and pointers.

```
Initializing SD card...card initialized.  
We created a heartMonitor Object !  
♥ A HeartBeat Happened !  
BPM: 99  
sucess opening datalog.txt  
♥ A HeartBeat Happened !
```

Test 2:

Aim: To test if the card wrote the text file and Arduino file named "ReadWrite" will be used. The code will be edited to only allow the reading function to be accepted. Initially the test will be testing an external file before the main file

Test 2.5:

```
Initialising SD card...initialisation done.
done.
data.txt:
```

```

Read Initializing SD card...initialization done.
*/
test.txt:
testing 1, 2, 3.
testing 1, 2, 3.
testing 1, 2, 3.
testing 1, 2, 3.
testing 1, 2, 3.
testing 1, 2, 3.

void setup() {
  // Open
  Serial
  while (!Serial) ; // wait for serial port to connect. Needed for native USB port only
}

Serial.print("Initializing SD card...");

if (!SD.begin(4)) {
  Serial.println("initialization failed!");
  while (1);
}
Serial.println("initialisation done.");

// open the file. note that only one file can be open at a time,
// so you have to close this one before opening another.
myFile = SD.open("test.txt", FILE_WRITE);

// if the file opened okay, write to it:
if (myFile) {
  Serial.print("Writing to test.txt...");
  myFile.println("testing 1, 2, 3.");
  // close the file:
  myFile.close();
  Serial.println("done.");
} else {
  // if the file didn't open, print an error:
  Serial.println("error opening test.txt");
}

// re-open the file for reading:
myFile = SD.open("test.txt");
if (myFile) {
  Serial.println("test.txt:");

  // read from the file until there's nothing else in it:
  while (myFile.available()) {
    Serial.write(myFile.read());
  }
  // close the file:
  myFile.close();
} else {
  // if the file didn't open, print an error:
  Serial.println("error opening test.txt");
}

}

void loop() {
  // nothing happens after setup
}

```

19.6.23

Test 1:

In the code below, the `sd.close` function is added and the pulse sensor code with the SD card is tested once more. This results in a strong entry of data. The correctness of this file was tested with the previous Arduino example code which resulted in success. Yet this was arranged in an illogical manner due to not adding a new line. As a result, this will be tested once more to arrange it nicely.

Pulse sensor with SD card code

Serial Monitor output of pulse sensor

```
#define USE_ARDUINO_INTERRUPTS true
#include <SD.h>
#include <PulseSensorPlayground.h>
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = { 0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02 };
IPAddress ip(192, 168, 5, 81);
EthernetServer server(80);
const int chipSelect = 4;

const int PulseWire = 0;
int Threshold = 547;
PulseSensorPlayground heartMonitor;

void setup()
{
  Serial.begin(9600);
  Serial.print("Initialising SD card");
  if (!SD.begin(chipSelect)) {
    Serial.println("Card error");
    return;
  }
  Serial.println("card initialised.");
  heartMonitor.analogInput(PulseWire);
  heartMonitor.setThreshold(Threshold);

  if (heartMonitor.begin()) {
    Serial.println("We created a heartMonitor Object !"); //initialisation
  }
  delay(50);
}

void loop()
{
  int myBPM = heartMonitor.getBeatsPerMinute();
  if (heartMonitor.sawStartOfBeat()) {
    if (myBPM >= 130) {
      myBPM = myBPM / 2;
    }
    else if (myBPM <= 45) {
      myBPM = myBPM * 2;
    }

    Serial.println(myBPM);
    File sdcard_file = SD.open("data.txt", FILE_WRITE);

    if (sdcard_file) {
      sdcard_file.print("BPM: ");
      sdcard_file.print(myBPM);
      Serial.println("♥ A HeartBeat Happened ! ");
      Serial.print("BPM: ");
      Serial.println("success opening datalog.txt");
      sdcard_file.close();
      Serial.println("closed");
    }
    else {
      Serial.println("error opening datalog.txt");
    }
    delay(1000);
  }
}
```

```
Initialising SD cardcard initialised.
We created a heartMonitor Object !
107
♥ A HeartBeat Happened !
BPM: success opening datalog.txt
closed
110
♥ A HeartBeat Happened !
BPM: success opening datalog.txt
closed
112
♥ A HeartBeat Happened !
BPM: success opening datalog.txt
closed
92
♥ A HeartBeat Happened !
BPM: success opening datalog.txt
closed
83
♥ A HeartBeat Happened !
BPM: success opening datalog.txt
closed
81
```

The output of the initial test(successful), yet arranged oddly

```
BPM: success opening datalog.txt
closed
Initialising SD card...initialisation done.
data.txt:
BPM: 73BPM: 82BPM: 84BPM: 87BPM: 90BPM: 110BPM: 84BPM: 114BPM: 97BPM: 86BPM: 76BPM: 75BPM: 72BPM: 70BPM: 71BPM: 125BPM: 69BPM: 102BPM: 94BPM: 89BPM: 81BPM: 70BPM: 81BPM: 103BPM: 69BPM: 70BPM: 80BPM: 91BPM: 100BPM: 94BPM: 82BPM: 85BPM: 114B
```

Deleting the file to reset the contents of the file

```
Initialising SD card...initialisation done.
data.txt doesn't exist.
data.txt doesn't exist.
Removing data.txt...
data.txt doesn't exist.
```

In this final test regarding the SD and pulse sensor, the results created in the same method as previously.

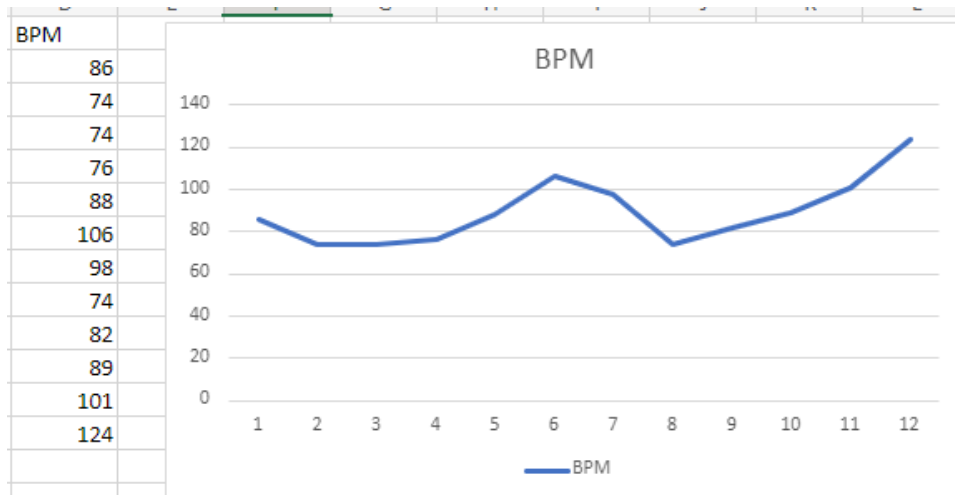
1. Upload pulse and SD code (with sd.close!)
2. Check Serial Monitor to make sure the results are valid

3. Upload the Arduino example code for reading the file
4. Check to see if both results are accurate

These results showed to be accurate which is successful. Although at this stage, the different initial number was confusing, yet this was explained by the order which the code was structures, SD being first and Serial being next.

```
Initialising SD cardcard initialised.      Initialising SD card...initialization done.
We created a heartMonitor Object !        data.txt:
74                                          BPM:
♥ A HeartBeat Happened !                  86
BPM: sucess opening datalog.txt            BPM:
closed                                     74
82                                          BPM:
♥ A HeartBeat Happened !                  74
BPM: sucess opening datalog.txt            BPM:
closed                                     76
89                                          BPM:
♥ A HeartBeat Happened !                  88
BPM: sucess opening datalog.txt            BPM:
closed                                     106
101                                         BPM:
♥ A HeartBeat Happened !                  98
BPM: sucess opening datalog.txt            BPM:
closed                                     74
124                                         BPM:
♥ A HeartBeat Happened !                  82
BPM: sucess opening datalog.txt            BPM:
closed                                     89
70                                          BPM:
♥ A HeartBeat Happened !                  101
BPM: sucess opening datalog.txt            BPM:
closed                                     124
126                                         BPM:
♥ A HeartBeat Happened !                  70
BPM: sucess opening datalog.txt            BPM:
closed                                     126
69                                          BPM:
♥ A HeartBeat Happened !                  69
BPM: sucess opening datalog.txt            BPM:
closed                                     80
80                                          BPM:
♥ A HeartBeat Happened !                  102
BPM: sucess opening datalog.txt            BPM:
closed                                     75
102                                         BPM:
♥ A HeartBeat Happened !                  78
BPM: sucess opening datalog.txt            BPM:
closed                                     87
102                                         BPM:
♥ A HeartBeat Happened !                  112
BPM: sucess opening datalog.txt            BPM:
closed                                     74
```

The next step is to input this data into a graph to test it. This was done by using excel. The previous results were copied and the BPM line was removed so there would be one line.



```

Initialising SD card...initialisation done.
data.txt:
BPM:
0
success opening data.txt
closed
BPM:
68
success opening data.txt
closed
BPM:
120
success opening data.txt
closed
BPM:
116
success opening data.txt
closed
BPM:
55
success opening data.txt
closed
BPM:
57
success opening data.txt
closed
BPM:
57
success opening data.txt
closed
BPM:
58
success opening data.txt
closed
BPM:
58
success opening data.txt
closed
BPM:
60
success opening data.txt
closed
BPM:
60
success opening data.txt
closed
BPM:
61
success opening data.txt
closed
BPM:
69
success opening data.txt
closed
BPM:
75
success opening data.txt
closed
BPM:
81
success opening data.txt
closed
BPM:
76
success opening data.txt
closed
BPM:
74
success opening data.txt
closed
BPM:
68
success opening data.txt
closed
BPM:
65
success opening data.txt
closed
BPM:
66
success opening data.txt
closed
BPM:
66
success opening data.txt
closed
BPM:
59
success opening data.txt
closed
BPM:
58
success opening data.txt
closed
BPM:
57
success opening data.txt
closed
BPM:
60
success opening data.txt
closed
BPM:
67
success opening data.txt
closed
BPM:
75

```

This is another example of the working heart rate monitoring and data reading from the Sd file.

Below is the current code.

```
#define USE_ARDUINO_INTERRUPTS true
#include <SD.h>
#include <PulseSensorPlayground.h>
#include <LiquidCrystal_I2C.h>
#include "myClass.h"
LiquidCrystal_I2C lcd(0x27,16,2);

int speaker = 9;
const int chipSelect = 4;
MyClass redLED(5);
MyClass greenLED(6);

void setup() {
  pinMode(speaker, OUTPUT);

  Serial.begin(9600);
  Serial.print("Initialising SD card");
  if (!SD.begin(chipSelect)) {
    Serial.println("Card error");
    return;
  }
  Serial.println("card initialised.");

  File sdcard_file = SD.open("data.txt", FILE_WRITE);

  if (sdcard_file) {
    sdcard_file.println("BPM: ");
    sdcard_file.println(myBPM);
    Serial.println("♥ A HeartBeat Happened ! ");
    Serial.println("BPM: ");
    Serial.println(myBPM);
    Serial.println("success opening data.txt");
    sdcard_file.close();
    Serial.println("closed");
  }
  else {
    Serial.println("error opening data.txt");
  }
}
```

Test 2:

The next major aim is the integration of all major aspects. This being the Sd card and the smartwatch, then the ethernet shield code with the full code.

Test 2 results:

The first test showed to be successful after several tests. Yet the integration of Wi-Fi was not. The major problem with this test was the lack of storage in the Arduino. As seen below the storage overwhelms the limitations significantly



```
data section exceeds available space in board
Sketch uses 27594 bytes (85%) of program storage space. Maximum is 32256 bytes.
Global variables use 2444 bytes (119%) of dynamic memory, leaving -396 bytes for local variables
Not enough memory: see https://support.arduino.cc/hc/en-us/articles/360013828179 for tips on reducing memory use
Error compiling for board Arduino Uno.
```

As a result, several unnecessary lines such as the implementation of graphics, specific html meta tags, and other external things were removed yet this continuously failed in Serial and through physical implementation. Yet the Arduino repeatedly flashed whilst

the pulse sensor was handled. Whilst this could possibly be solved by creating a library, creating more functions, decreasing the memory of strings and other variables, the lack of time is a major impediment to doing so.

Due to this, the task will now be divided into two major sections. The first section is the current smartwatch, this will contain the main code (LCD, pulse sensor, SD card, LEDs, piezo buzzer). Logically this will be the 'mobile version' of the heart-rate monitoring system. The second section will be the ethernet-based code where the pulse sensor and the ethernet shield will work together to simultaneously display the user's heart rate through the Serial Monitor and a webserver. This will be the 'stationary version', based in a hospital.

Test 3:

Aim: the test will be centred on allowing the webserver to display the inputs of the pulse sensor.

Test 3 results:

The results for this first test were successful in displaying the BPM of the pulse sensor, yet it did not update the output data consistently. As a result, the next test for this section will be based around adding a refresh rate.



Heart Monitoring System

Heart Rate:

87

Temperature:

Humidity:

Created by Phebe Le in Session 2 2023

The code below this, is not another test. It is a result of optimising the efficiency of the code by creating functions.

```

Serial.println("LED OFF");
return;
}
Serial.println("card initialized.");
int.init();
int.blinkLight();
pinMode();
delay(2000);
int.clear();
}

int.pulsing() {
//this section was copied from external web
int.createChar(1, heart1);
int.createChar(2, heart2);
int.createChar(3, heart3);
int.createChar(4, heart4);
int.createChar(5, heart5);
int.createChar(6, heart6);
int.createChar(7, heart7);
int.createChar(8, heart8);
//external section ends

int.setCursor(0,0);
int.print("Heart Monitor ");
int.setCursor(0,1);
int.print("System ");
heartMonitor.analogInput(PulseWire);
heartMonitor.setThreshold(Threshold);

if (heartMonitor.begin()) {
Serial.println("We created a heartMonitor 0");
}

int initial(int a) {
if (a < 500) {
a=0;
}
else if (a > 499) {
int.setCursor(0,0);
int.print("Put your finger ");
int.setCursor(0,1);
int.print("on the sensor ");
delay(1000);
int.clear();
delay(500);
}
}

int.LCDHeartView(int bpm) {
int.setCursor(1,1);
int.write(bpm);
int.setCursor(5,1);
int.write(bpm);
int.setCursor(0,0);
int.write(bpm);
int.setCursor(1,0);
int.write(bpm);
int.setCursor(2,0);
int.write(bpm);
int.setCursor(3,0);
int.write(bpm);
int.setCursor(4,0);
int.write(bpm);
int.setCursor(5,0);
int.write(bpm);
int.setCursor(6,0);
int.write(bpm);
int.setCursor(7,0);
int.write(bpm);
int.setCursor(8,0);
int.write(bpm);
int.setCursor(9,0);
int.print("Heart rate");
int.setCursor(5,1);
int.print(": ");
int.print(bpm);
int.print(" ");
int.print("bpm");
}

int.alertSystem(int bpm) {
if (bpm > 100 || bpm < 55) {
int.g_speaker = speaker; // create a po
greenLED.off();
redLED.on(500);
tone(g_speaker, 200);
}
}

}

void loop() {
int myBPM = heartMonitor.getBPM();
initial(Instructions_view);
if (heartMonitor.newStartOfBeat()) {
if (myBPM >= 120) {
myBPM = myBPM / 2;
}
else if (myBPM <= 50) {
myBPM = myBPM * 2;
}
File adcard_file = SD.open("data.txt", FILE_WRITE);

if (adcard_file) {
adcard_file.println("BPM: ");
adcard_file.println(myBPM);
Serial.println("♥ A HeartBeat Happened !");
Serial.println("BPM: ");
Serial.println(myBPM);
Serial.println("success opening data.txt");
adcard_file.close();
Serial.println("closed");
}
else {
Serial.println("error opening data.txt");
}
delay(1000);
LCDHeartView(myBPM);
Instructions_view = 0;
alertSystem(myBPM);
}
delay(20);
}

```

20/6/23

Aim: The aim will be heavily based on implementing the required functions for the smartwatch code. These being malloc, vectors and pointers. These will be integrated to dynamically allocate memory to store within the SD. The vectors will be created through the c++ vector class. Below is the integration of the all required aspects.

```

15 const int PulseWire = 0;
16 const int BUFFER = 20; //dynamically stores 20
17 std::vector<int> bpmBuffer;;

51 void setup() {
52     pinMode(speaker, OUTPUT);
53     Serial.begin(9600);
54     bpmBuffer.reserve(BUFFER);
55     Serial.print("Initializing SD card");
56     if (!SD.begin(chipSelect)) {

```

```
bpmBuffer.push_back(myBPM);

if (bpmBuffer.size() >= BUFFER) {
    int* dynamicBuffer = (int*)malloc(BUFFER * sizeof(int));
    if (dynamicBuffer == nullptr) {
        Serial.println("Error allocating memory");
        return;
    }
    for (int i = 0; i < BUFFER; i++) {
        dynamicBuffer[i] = bpmBuffer[i];
    }
    File sdcard_file = SD.open("data.txt", FILE_WRITE);

    if (sdcard_file) {
        for (int i = 0; i < BUFFER; i++) {
            sdcard_file.println("BPM: ");
            sdcard_file.println(dynamicBuffer[i]);
            Serial.println("♥ A HeartBeat Happened ! ");
            Serial.println("BPM: ");
            Serial.println(dynamicBuffer[i]);
            Serial.println("sucess opening data.txt");
        }
        sdcard_file.close();
        Serial.println("closed");
    }
    else {
        Serial.println("error opening data.txt");
    }
    delay(1000);
    LCDHeartView(myBPM);
    Instructions_view = 0;
    alertSystem(myBPM);
    free(dynamicBuffer);
    bpmBuffer.clear();
}
delay(20);
}
```

Below is the main code with a full integration.

```
1 #include <vector>
2 #define USE_ARDUINO_INTERRUPTS true
3 #include <SD.h>
4 #include <PulseSensorPlayground.h>
5 #include <LiquidCrystal_I2C.h>
6 #include <SPI.h>
7
8 #include "MyClass.h"
9 LiquidCrystal_I2C lcd(0x27, 16, 2);
10 PulseSensorPlayground heartMonitor;
11 int speaker = 5;
12 int Instructions_view = 500; //--- Variable for waiting time to display instructions on LCD.
13 int Threshold = 547;
14 const int chipSelect = 4;
15 const int PulseWire = 0;
16 const int BUFFER = 20; //dynamically stores 20
17 std::vector<int> bpmBuffer;
18
19 MyClass redLED(5);
20 MyClass greenLED(6);
21
22 MyClass::MyClass(int pin) {
23     pinMode(pin, OUTPUT);
24     _pin = pin;
25 }
26 void MyClass::on() {
27     digitalWrite(_pin, HIGH);
28 }
29 void MyClass::off() {
30     digitalWrite(_pin, LOW);
31 }
32
33 void MyClass::blink(int blinking) {
34     digitalWrite(_pin, HIGH);
35     delay(blinking);
36     digitalWrite(_pin, LOW);
37     delay(blinking);
38 }
39
40 //This section was copied from external website to create the LCD screen
41 byte heart[18] = {B11111, B11111, B11111, B11111, B01111, B00011, B00011, B00011,
42 byte heart[18] = {B00011, B00011, B00011, B00000, B00000, B00000, B00000, B00000,
43 byte heart[18] = {B00011, B00111, B01111, B11111, B11111, B11111, B11111, B11111,
44 byte heart[18] = {B11000, B11100, B11110, B11111, B11111, B11111, B11111, B11111,
45 byte heart[18] = {B00011, B00111, B01111, B11111, B11111, B11111, B11111, B11111,
46 byte heart[18] = {B11000, B11100, B11110, B11111, B11111, B11111, B11111, B11110,
47 byte heart[18] = {B11000, B10000, B00000, B00000, B00000, B00000, B00000, B00000,
48 byte heart[18] = {B11111, B11111, B11111, B11111, B11110, B11100, B10000, B00000,
49 //External section ends
50
51 void setup() {
52     pinMode(speaker, OUTPUT);
53     Serial.begin(5000);
54     bpmBuffer.reserve(BUFFER);
55     Serial.print("Initialising SD card");
56     if (SD.begin(chipSelect)) {
57         Serial.println("Card error");
58         return;
59     }
60     Serial.println("Card initialised.");
61     lcd.init();
62     lcd.backlight();
63     pulsing();
64     delay(2000);
65     lcd.clear();
66 }
67
68 void loop() {
69     if (bpm > 100 || bpm < 55) {
70         int* p_speaker = speaker; // create a pointer to the pin v
71         greenLED.off();
72         redLED.blink(500);
73         tone(*p_speaker, 262);
74         delay(200);
75         noTone(*p_speaker);
76         delay(200);
77     }
78     else {
79         greenLED.on();
80     }
81 }
82
83 void loop() {
84     int myBPM = heartMonitor.getBeatsPerMinute();
85
86     initial(Instructions_view);
87     if (heartMonitor.sawStartOfBeat()) {
88         if (myBPM >= 120) {
89             myBPM = myBPM / 2;
90         }
91         else if (myBPM <= 50) {
92             myBPM = myBPM * 2;
93         }
94         bpmBuffer.push_back(myBPM);
95
96         if (bpmBuffer.size() >= BUFFER) {
97             int* dynamicBuffer = (int*)malloc(BUFFER * sizeof(int));
98             if (dynamicBuffer == nullptr) {
99                 Serial.println("Error allocating memory");
100                 return;
101             }
102             for (int i = 0; i < BUFFER; i++) {
103                 dynamicBuffer[i] = bpmBuffer[i];
104             }
105             File sdcard_file = SD.open("data.txt", FILE_WRITE);
106
107             if (sdcard_file) {
108                 for (int i = 0; i < BUFFER; i++) {
109                     sdcard_file.println("BPM: ");
110                     sdcard_file.println(dynamicBuffer[i]);
111                     Serial.println("♥ A HeartBeat Happened ! ");
112                     Serial.println("BPM: ");
113                     Serial.println(dynamicBuffer[i]);
114                     Serial.println("success opening data.txt");
115                 }
116                 sdcard_file.close();
117                 Serial.println("closed");
118             }
119             else {
120                 Serial.println("error opening data.txt");
121             }
122             delay(1000);
123             LCDHeartView(myBPM);
124             Instructions_view = 0;
125             alertSystem(myBPM);
126             free(dynamicBuffer);
127             bpmBuffer.clear();
128         }
129         delay(20);
130     }
131 }
132
133 void initial(int a) {
134     if (a < 500) {
135         a++;
136     }
137     else if (a > 450) {
138         lcd.setCursor(0, 0);
139         lcd.print("Put your finger ");
140         lcd.setCursor(0, 1);
141         lcd.print("on the sensor ");
142         delay(1000);
143         lcd.clear();
144         delay(500);
145     }
146 }
147
148 void LCDHeartView(int bpm) {
149     lcd.setCursor(1, 1);
150     lcd.write(byte(1));
151     lcd.setCursor(0, 1);
152     lcd.write(byte(2));
153     lcd.setCursor(0, 0);
154     lcd.write(byte(3));
155     lcd.setCursor(1, 0);
156     lcd.write(byte(4));
157     lcd.setCursor(2, 0);
158     lcd.write(byte(5));
159     lcd.setCursor(3, 0);
160     lcd.write(byte(6));
161     lcd.setCursor(4, 1);
162     lcd.write(byte(7));
163     lcd.setCursor(2, 1);
164     lcd.write(byte(8));
165     lcd.setCursor(5, 0);
166     lcd.print("Heart Rate");
167     lcd.setCursor(5, 1);
168     lcd.print(" ");
169     lcd.print(bpm);
170     lcd.print(" ");
171     lcd.print("BPM");
172 }
173
174 void alertSystem(int bpm) {
175     if (bpm > 100 || bpm < 55) {
176         int* p_speaker = speaker; // create a pointer to the pin v
177         greenLED.off();
178         redLED.blink(500);
179         tone(*p_speaker, 262);
180         delay(200);
181         noTone(*p_speaker);
182         delay(200);
183     }
184     else {
185         greenLED.on();
186     }
187 }
188
189 void loop() {
190     int myBPM = heartMonitor.getBeatsPerMinute();
191
192     initial(Instructions_view);
193     if (heartMonitor.sawStartOfBeat()) {
194         if (myBPM >= 120) {
195             myBPM = myBPM / 2;
196         }
197         else if (myBPM <= 50) {
198             myBPM = myBPM * 2;
199         }
200         bpmBuffer.push_back(myBPM);
201
202         if (bpmBuffer.size() >= BUFFER) {
203             int* dynamicBuffer = (int*)malloc(BUFFER * sizeof(int));
204             if (dynamicBuffer == nullptr) {
205                 Serial.println("Error allocating memory");
206                 return;
207             }
208             for (int i = 0; i < BUFFER; i++) {
209                 dynamicBuffer[i] = bpmBuffer[i];
210             }
211             File sdcard_file = SD.open("data.txt", FILE_WRITE);
212
213             if (sdcard_file) {
214                 for (int i = 0; i < BUFFER; i++) {
215                     sdcard_file.println("BPM: ");
216                     sdcard_file.println(dynamicBuffer[i]);
217                     Serial.println("♥ A HeartBeat Happened ! ");
218                     Serial.println("BPM: ");
219                     Serial.println(dynamicBuffer[i]);
220                     Serial.println("success opening data.txt");
221                 }
222                 sdcard_file.close();
223                 Serial.println("closed");
224             }
225             else {
226                 Serial.println("error opening data.txt");
227             }
228             delay(1000);
229             LCDHeartView(myBPM);
230             Instructions_view = 0;
231             alertSystem(myBPM);
232             free(dynamicBuffer);
233             bpmBuffer.clear();
234         }
235         delay(20);
236     }
237 }
238
239 void initial(int a) {
240     if (a < 500) {
241         a++;
242     }
243     else if (a > 450) {
244         lcd.setCursor(0, 0);
245         lcd.print("Put your finger ");
246         lcd.setCursor(0, 1);
247         lcd.print("on the sensor ");
248         delay(1000);
249         lcd.clear();
250         delay(500);
251     }
252 }
253
254 void LCDHeartView(int bpm) {
255     lcd.setCursor(1, 1);
256     lcd.write(byte(1));
257     lcd.setCursor(0, 1);
258     lcd.write(byte(2));
259     lcd.setCursor(0, 0);
260     lcd.write(byte(3));
261     lcd.setCursor(1, 0);
262     lcd.write(byte(4));
263     lcd.setCursor(2, 0);
264     lcd.write(byte(5));
265     lcd.setCursor(3, 0);
266     lcd.write(byte(6));
267     lcd.setCursor(4, 1);
268     lcd.write(byte(7));
269     lcd.setCursor(2, 1);
270     lcd.write(byte(8));
271     lcd.setCursor(5, 0);
272     lcd.print("Heart Rate");
273     lcd.setCursor(5, 1);
274     lcd.print(" ");
275     lcd.print(bpm);
276     lcd.print(" ");
277     lcd.print("BPM");
278 }
279
280 void alertSystem(int bpm) {
281     if (bpm > 100 || bpm < 55) {
282         int* p_speaker = speaker; // create a pointer to the pin v
283         greenLED.off();
284         redLED.blink(500);
285         tone(*p_speaker, 262);
286         delay(200);
287         noTone(*p_speaker);
288         delay(200);
289     }
290     else {
291         greenLED.on();
292     }
293 }
294
295 void loop() {
296     int myBPM = heartMonitor.getBeatsPerMinute();
297
298     initial(Instructions_view);
299     if (heartMonitor.sawStartOfBeat()) {
300         if (myBPM >= 120) {
301             myBPM = myBPM / 2;
302         }
303         else if (myBPM <= 50) {
304             myBPM = myBPM * 2;
305         }
306         bpmBuffer.push_back(myBPM);
307
308         if (bpmBuffer.size() >= BUFFER) {
309             int* dynamicBuffer = (int*)malloc(BUFFER * sizeof(int));
310             if (dynamicBuffer == nullptr) {
311                 Serial.println("Error allocating memory");
312                 return;
313             }
314             for (int i = 0; i < BUFFER; i++) {
315                 dynamicBuffer[i] = bpmBuffer[i];
316             }
317             File sdcard_file = SD.open("data.txt", FILE_WRITE);
318
319             if (sdcard_file) {
320                 for (int i = 0; i < BUFFER; i++) {
321                     sdcard_file.println("BPM: ");
322                     sdcard_file.println(dynamicBuffer[i]);
323                     Serial.println("♥ A HeartBeat Happened ! ");
324                     Serial.println("BPM: ");
325                     Serial.println(dynamicBuffer[i]);
326                     Serial.println("success opening data.txt");
327                 }
328                 sdcard_file.close();
329                 Serial.println("closed");
330             }
331             else {
332                 Serial.println("error opening data.txt");
333             }
334             delay(1000);
335             LCDHeartView(myBPM);
336             Instructions_view = 0;
337             alertSystem(myBPM);
338             free(dynamicBuffer);
339             bpmBuffer.clear();
340         }
341         delay(20);
342     }
343 }
344
345 void initial(int a) {
346     if (a < 500) {
347         a++;
348     }
349     else if (a > 450) {
350         lcd.setCursor(0, 0);
351         lcd.print("Put your finger ");
352         lcd.setCursor(0, 1);
353         lcd.print("on the sensor ");
354         delay(1000);
355         lcd.clear();
356         delay(500);
357     }
358 }
359
360 void LCDHeartView(int bpm) {
361     lcd.setCursor(1, 1);
362     lcd.write(byte(1));
363     lcd.setCursor(0, 1);
364     lcd.write(byte(2));
365     lcd.setCursor(0, 0);
366     lcd.write(byte(3));
367     lcd.setCursor(1, 0);
368     lcd.write(byte(4));
369     lcd.setCursor(2, 0);
370     lcd.write(byte(5));
371     lcd.setCursor(3, 0);
372     lcd.write(byte(6));
373     lcd.setCursor(4, 1);
374     lcd.write(byte(7));
375     lcd.setCursor(2, 1);
376     lcd.write(byte(8));
377     lcd.setCursor(5, 0);
378     lcd.print("Heart Rate");
379     lcd.setCursor(5, 1);
380     lcd.print(" ");
381     lcd.print(bpm);
382     lcd.print(" ");
383     lcd.print("BPM");
384 }
385
386 void alertSystem(int bpm) {
387     if (bpm > 100 || bpm < 55) {
388         int* p_speaker = speaker; // create a pointer to the pin v
389         greenLED.off();
390         redLED.blink(500);
391         tone(*p_speaker, 262);
392         delay(200);
393         noTone(*p_speaker);
394         delay(200);
395     }
396     else {
397         greenLED.on();
398     }
399 }
400
401 void loop() {
402     int myBPM = heartMonitor.getBeatsPerMinute();
403
404     initial(Instructions_view);
405     if (heartMonitor.sawStartOfBeat()) {
406         if (myBPM >= 120) {
407             myBPM = myBPM / 2;
408         }
409         else if (myBPM <= 50) {
410             myBPM = myBPM * 2;
411         }
412         bpmBuffer.push_back(myBPM);
413
414         if (bpmBuffer.size() >= BUFFER) {
415             int* dynamicBuffer = (int*)malloc(BUFFER * sizeof(int));
416             if (dynamicBuffer == nullptr) {
417                 Serial.println("Error allocating memory");
418                 return;
419             }
420             for (int i = 0; i < BUFFER; i++) {
421                 dynamicBuffer[i] = bpmBuffer[i];
422             }
423             File sdcard_file = SD.open("data.txt", FILE_WRITE);
424
425             if (sdcard_file) {
426                 for (int i = 0; i < BUFFER; i++) {
427                     sdcard_file.println("BPM: ");
428                     sdcard_file.println(dynamicBuffer[i]);
429                     Serial.println("♥ A HeartBeat Happened ! ");
430                     Serial.println("BPM: ");
431                     Serial.println(dynamicBuffer[i]);
432                     Serial.println("success opening data.txt");
433                 }
434                 sdcard_file.close();
435                 Serial.println("closed");
436             }
437             else {
438                 Serial.println("error opening data.txt");
439             }
440             delay(1000);
441             LCDHeartView(myBPM);
442             Instructions_view = 0;
443             alertSystem(myBPM);
444             free(dynamicBuffer);
445             bpmBuffer.clear();
446         }
447         delay(20);
448     }
449 }
450
451 void initial(int a) {
452     if (a < 500) {
453         a++;
454     }
455     else if (a > 450) {
456         lcd.setCursor(0, 0);
457         lcd.print("Put your finger ");
458         lcd.setCursor(0, 1);
459         lcd.print("on the sensor ");
460         delay(1000);
461         lcd.clear();
462         delay(500);
463     }
464 }
465
466 void LCDHeartView(int bpm) {
467     lcd.setCursor(1, 1);
468     lcd.write(byte(1));
469     lcd.setCursor(0, 1);
470     lcd.write(byte(2));
471     lcd.setCursor(0, 0);
472     lcd.write(byte(3));
473     lcd.setCursor(1, 0);
474     lcd.write(byte(4));
475     lcd.setCursor(2, 0);
476     lcd.write(byte(5));
477     lcd.setCursor(3, 0);
478     lcd.write(byte(6));
479     lcd.setCursor(4, 1);
480     lcd.write(byte(7));
481     lcd.setCursor(2, 1);
482     lcd.write(byte(8));
483     lcd.setCursor(5, 0);
484     lcd.print("Heart Rate");
485     lcd.setCursor(5, 1);
486     lcd.print(" ");
487     lcd.print(bpm);
488     lcd.print(" ");
489     lcd.print("BPM");
490 }
491
492 void alertSystem(int bpm) {
493     if (bpm > 100 || bpm < 55) {
494         int* p_speaker = speaker; // create a pointer to the pin v
495         greenLED.off();
496         redLED.blink(500);
497         tone(*p_speaker, 262);
498         delay(200);
499         noTone(*p_speaker);
500         delay(200);
501     }
502     else {
503         greenLED.on();
504     }
505 }
506
507 void loop() {
508     int myBPM = heartMonitor.getBeatsPerMinute();
509
510     initial(Instructions_view);
511     if (heartMonitor.sawStartOfBeat()) {
512         if (myBPM >= 120) {
513             myBPM = myBPM / 2;
514         }
515         else if (myBPM <= 50) {
516             myBPM = myBPM * 2;
517         }
518         bpmBuffer.push_back(myBPM);
519
520         if (bpmBuffer.size() >= BUFFER) {
521             int* dynamicBuffer = (int*)malloc(BUFFER * sizeof(int));
522             if (dynamicBuffer == nullptr) {
523                 Serial.println("Error allocating memory");
524                 return;
525             }
526             for (int i = 0; i < BUFFER; i++) {
527                 dynamicBuffer[i] = bpmBuffer[i];
528             }
529             File sdcard_file = SD.open("data.txt", FILE_WRITE);
530
531             if (sdcard_file) {
532                 for (int i = 0; i < BUFFER; i++) {
533                     sdcard_file.println("BPM: ");
534                     sdcard_file.println(dynamicBuffer[i]);
535                     Serial.println("♥ A HeartBeat Happened ! ");
536                     Serial.println("BPM: ");
537                     Serial.println(dynamicBuffer[i]);
538                     Serial.println("success opening data.txt");
539                 }
540                 sdcard_file.close();
541                 Serial.println("closed");
542             }
543             else {
544                 Serial.println("error opening data.txt");
545             }
546             delay(1000);
547             LCDHeartView(myBPM);
548             Instructions_view = 0;
549             alertSystem(myBPM);
550             free(dynamicBuffer);
551             bpmBuffer.clear();
552         }
553         delay(20);
554     }
555 }
556
557 void initial(int a) {
558     if (a < 500) {
559         a++;
560     }
561     else if (a > 450) {
562         lcd.setCursor(0, 0);
563         lcd.print("Put your finger ");
564         lcd.setCursor(0, 1);
565         lcd.print("on the sensor ");
566         delay(1000);
567         lcd.clear();
568         delay(500);
569     }
570 }
571
572 void LCDHeartView(int bpm) {
573     lcd.setCursor(1, 1);
574     lcd.write(byte(1));
575     lcd.setCursor(0, 1);
576     lcd.write(byte(2));
577     lcd.setCursor(0, 0);
578     lcd.write(byte(3));
579     lcd.setCursor(1, 0);
580     lcd.write(byte(4));
581     lcd.setCursor(2, 0);
582     lcd.write(byte(5));
583     lcd.setCursor(3, 0);
584     lcd.write(byte(6));
585     lcd.setCursor(4, 1);
586     lcd.write(byte(7));
587     lcd.setCursor(2, 1);
588     lcd.write(byte(8));
589     lcd.setCursor(5, 0);
590     lcd.print("Heart Rate");
591     lcd.setCursor(5, 1);
592     lcd.print(" ");
593     lcd.print(bpm);
594     lcd.print(" ");
595     lcd.print("BPM");
596 }
597
598 void alertSystem(int bpm) {
599     if (bpm > 100 || bpm < 55) {
600         int* p_speaker = speaker; // create a pointer to the pin v
601         greenLED.off();
602         redLED.blink(500);
603         tone(*p_speaker, 262);
604         delay(200);
605         noTone(*p_speaker);
606         delay(200);
607     }
608     else {
609         greenLED.on();
610     }
611 }
612
613 void loop() {
614     int myBPM = heartMonitor.getBeatsPerMinute();
615
616     initial(Instructions_view);
617     if (heartMonitor.sawStartOfBeat()) {
618         if (myBPM >= 120) {
619             myBPM = myBPM / 2;
620         }
621         else if (myBPM <= 50) {
622             myBPM = myBPM * 2;
623         }
624         bpmBuffer.push_back(myBPM);
625
626         if (bpmBuffer.size() >= BUFFER) {
627             int* dynamicBuffer = (int*)malloc(BUFFER * sizeof(int));
628             if (dynamicBuffer == nullptr) {
629                 Serial.println("Error allocating memory");
630                 return;
631             }
632             for (int i = 0; i < BUFFER; i++) {
633                 dynamicBuffer[i] = bpmBuffer[i];
634             }
635             File sdcard_file = SD.open("data.txt", FILE_WRITE);
636
637             if (sdcard_file) {
638                 for (int i = 0; i < BUFFER; i++) {
639                     sdcard_file.println("BPM: ");
640                     sdcard_file.println(dynamicBuffer[i]);
641                     Serial.println("♥ A HeartBeat Happened ! ");
642                     Serial.println("BPM: ");
643                     Serial.println(dynamicBuffer[i]);
644                     Serial.println("success opening data.txt");
645                 }
646                 sdcard_file.close();
647                 Serial.println("closed");
648             }
649             else {
650                 Serial.println("error opening data.txt");
651             }
652             delay(1000);
653             LCDHeartView(myBPM);
654             Instructions_view = 0;
655             alertSystem(myBPM);
656             free(dynamicBuffer);
657             bpmBuffer.clear();
658         }
659         delay(20);
660     }
661 }
662
663 void initial(int a) {
664     if (a < 500) {
665         a++;
666     }
667     else if (a > 450) {
668         lcd.setCursor(0, 0);
669         lcd.print("Put your finger ");
670         lcd.setCursor(0, 1);
671         lcd.print("on the sensor ");
672         delay(1000);
673         lcd.clear();
674         delay(500);
675     }
676 }
677
678 void LCDHeartView(int bpm) {
679     lcd.setCursor(1, 1);
680     lcd.write(byte(1));
681     lcd.setCursor(0, 1);
682     lcd.write(byte(2));
683     lcd.setCursor(0, 0);
684     lcd.write(byte(3));
685     lcd.setCursor(1, 0);
686     lcd.write(byte(4));
687     lcd.setCursor(2, 0);
688     lcd.write(byte(5));
689     lcd.setCursor(3, 0);
690     lcd.write(byte(6));
691     lcd.setCursor(4, 1);
692     lcd.write(byte(7));
693     lcd.setCursor(2, 1);
694     lcd.write(byte(8));
695     lcd.setCursor(5, 0);
696     lcd.print("Heart Rate");
697     lcd.setCursor(5, 1);
698     lcd.print(" ");
699     lcd.print(bpm);
700     lcd.print(" ");
701     lcd.print("BPM");
702 }
703
704 void alertSystem(int bpm) {
705     if (bpm > 100 || bpm < 55) {
706         int* p_speaker = speaker; // create a pointer to the pin v
707         greenLED.off();
708         redLED.blink(500);
709         tone(*p_speaker, 262);
710         delay(200);
711         noTone(*p_speaker);
712         delay(200);
713     }
714     else {
715         greenLED.on();
716     }
717 }
718
719 void loop() {
720     int myBPM = heartMonitor.getBeatsPerMinute();
721
722     initial(Instructions_view);
723     if (heartMonitor.sawStartOfBeat()) {
724         if (myBPM >= 120) {
725             myBPM = myBPM / 2;
726         }
727         else if (myBPM <= 50) {
728             myBPM = myBPM * 2;
729         }
730         bpmBuffer.push_back(myBPM);
731
732         if (bpmBuffer.size() >= BUFFER) {
733             int* dynamicBuffer = (int*)malloc(BUFFER * sizeof(int));
734             if (dynamicBuffer == nullptr) {
735                 Serial.println("Error allocating memory");
736                 return;
737             }
738             for (int i = 0; i < BUFFER; i++) {
739                 dynamicBuffer[i] = bpmBuffer[i];
740             }
741             File sdcard_file = SD.open("data.txt", FILE_WRITE);
742
743             if (sdcard_file) {
744                 for (int i = 0; i < BUFFER; i++) {
745                     sdcard_file.println("BPM: ");
746                     sdcard_file.println(dynamicBuffer[i]);
747                     Serial.println("♥ A HeartBeat Happened ! ");
748                     Serial.println("BPM: ");
749                     Serial.println(dynamicBuffer[i]);
750                     Serial.println("success opening data.txt");
751                 }
752                 sdcard_file.close();
753                 Serial.println("closed");
754             }
755             else {
756                 Serial.println("error opening data.txt");
757             }
758             delay(1000);
759             LCDHeartView(myBPM);
760             Instructions_view = 0;
761             alertSystem(myBPM);
762             free(dynamicBuffer);
763             bpmBuffer.clear();
764         }
765         delay(20);
766     }
767 }
768
769 void initial(int a) {
770     if (a < 500) {
771         a++;
772     }
773     else if (a > 450) {
774         lcd.setCursor(0, 0);
775         lcd.print("Put your finger ");
776         lcd.setCursor(0, 1);
777         lcd.print("on the sensor ");
778         delay(1000);
779         lcd.clear();
780         delay(500);
781     }
782 }
783
784 void LCDHeartView(int bpm) {
785     lcd.setCursor(1, 1);
786     lcd.write(byte(1));
787     lcd.setCursor(0, 1);
788     lcd.write(byte(2));
789     lcd.setCursor(0, 0);
790     lcd.write(byte(3));
791     lcd.setCursor(1, 0);
792     lcd.write(byte(4));
793     lcd.setCursor(2, 0);
794     lcd.write(byte(5));
795     lcd.setCursor(3, 0);
796     lcd.write(byte(6));
797     lcd.setCursor(4, 1);
798     lcd.write(byte(7));
799     lcd.setCursor(2, 1);
800     lcd.write(byte(8));
801     lcd.setCursor(5, 0);
802     lcd.print("Heart Rate");
803     lcd.setCursor(5, 1);
804     lcd.print(" ");
805     lcd.print(bpm);
806     lcd.print(" ");
807     lcd.print("BPM");
808 }
809
810 void alertSystem(int bpm) {
811     if (bpm > 100 || bpm < 55) {
812         int* p_speaker = speaker; // create a pointer to the pin v
813         greenLED.off();
814         redLED.blink(500);
815         tone(*p_speaker, 262);
816         delay(200);
817         noTone(*p_speaker);
818         delay(200);
819     }
820     else {
821         greenLED.on();
822     }
823 }
824
825 void loop() {
826     int myBPM = heartMonitor.getBeatsPerMinute();
827
828     initial(Instructions_view);
829     if (heartMonitor.sawStartOfBeat()) {
830         if (myBPM >= 120) {
831             myBPM = myBPM / 2;
832         }
833         else if (myBPM <= 50) {
834             myBPM = myBPM * 2;
835         }
836         bpmBuffer.push_back(myBPM);
837
838         if (bpmBuffer.size() >= BUFFER) {
839             int* dynamicBuffer = (int*)malloc(BUFFER * sizeof(int));
840             if (dynamicBuffer == nullptr) {
841                 Serial.println("Error allocating memory");
842                 return;
843             }
844             for (int i = 0; i < BUFFER; i++) {
845                 dynamicBuffer[i] = bpmBuffer[i];
846             }
847             File sdcard_file = SD.open("data.txt", FILE_WRITE);
848
849             if (sdcard_file) {
850                 for (int i = 0; i < BUFFER; i++) {
851                     sdcard_file.println("BPM: ");
852                     sdcard_file.println(dynamicBuffer[i]);
853                     Serial.println("♥ A HeartBeat Happened ! ");
854                     Serial.println("BPM: ");
855                     Serial.println(dynamicBuffer[i]);
856                     Serial.println("success opening data.txt");
857                 }
858                 sdcard_file.close();
859                 Serial.println("closed");
860             }
861             else {
862                 Serial.println("error opening data.txt");
863             }
864             delay(1000);
865             LCDHeartView(myBPM);
866             Instructions_view = 0;
867             alertSystem(myBPM);
868             free(dynamicBuffer);
869             bpmBuffer.clear();
870         }
871         delay(20);
872     }
873 }
874
875 void initial(int a) {
876     if (a < 500) {
877         a++;
878     }
879     else if (a > 450) {
880         lcd.setCursor(0, 0);
881         lcd.print("Put your finger ");
882         lcd.setCursor(0, 1);
883         lcd.print("on the sensor ");
884         delay(1000);
885         lcd.clear();
886         delay(500);
887     }
888 }
889
890 void LCDHeartView(int bpm) {
891     lcd.setCursor(1, 1);
892     lcd.write(byte(1));
893     lcd.setCursor(0, 1);
894     lcd.write(byte(2));
895     lcd.setCursor(0, 0);
896     lcd.write(byte(3));
897     lcd.setCursor(1, 0);
898     lcd.write(byte(4));
899     lcd.setCursor(2, 0);
900     lcd.write(byte(5));
901     lcd.setCursor(3, 0);
902     lcd.write(byte(6));
903     lcd.setCursor(4, 1);
904     lcd.write(byte(7));
905     lcd.setCursor(2, 1);
906     lcd.write(byte(8));
907     lcd.setCursor(5, 0);
908     lcd.print("Heart Rate");
909     lcd.setCursor(5, 1);
910     lcd.print(" ");
911     lcd.print(bpm);
912     lcd.print(" ");
913     lcd.print("BPM");
914 }
915
916 void alertSystem(int bpm) {
917     if (bpm > 100 || bpm < 55) {
918         int* p_speaker = speaker; // create a pointer to the pin v
919         greenLED.off();
920         redLED.blink(500);
921         tone(*p_speaker, 262);
922         delay(200);
923         noTone(*p_speaker);
924         delay(200);
925     }
926     else {
927         greenLED.on();
928     }
929 }
930
931 void loop() {
932     int myBPM = heartMonitor.getBeatsPerMinute();
933
934     initial(Instructions_view);
935     if (heartMonitor.sawStartOfBeat()) {
936         if (myBPM >= 120) {
937             myBPM = myBPM / 2;
938         }
939         else if (myBPM <= 50) {
940             myBPM = myBPM * 2;
941         }
942         bpmBuffer.push_back(myBPM);
943
944         if (bpmBuffer.size() >= BUFFER) {
945             int* dynamicBuffer = (int*)malloc(BUFFER * sizeof(int));
946             if (dynamicBuffer == nullptr) {
947                 Serial.println("Error allocating memory");
948                 return;
949             }
950             for (int i = 0; i < BUFFER; i++) {
951                 dynamicBuffer[i] = bpmBuffer[i];
952             }
953             File sdcard_file = SD.open("data.txt", FILE_WRITE);
954
955             if (sdcard_file) {
956                 for (int i = 0; i < BUFFER; i++) {
957                     sdcard_file.println("BPM: ");
958                     sdcard_file.println(dynamicBuffer[i]);
959                     Serial.println("♥ A HeartBeat Happened ! ");
960                     Serial.println("BPM: ");
961                     Serial.println(dynamicBuffer[i]);
962                     Serial.println("success opening data.txt");
963                 }
964                 sdcard_file.close();
965                 Serial.println("closed");
966             }
967             else {
968                 Serial.println("error opening data.txt");
969             }
970             delay(1000);
971             LCDHeartView(myBPM);
972             Instructions_view = 0;
973             alertSystem(myBPM);
974             free(dynamicBuffer);
975             bpmBuffer.clear();
976         }
977         delay(20);
978     }
979 }
980
981 void initial(int a) {
982     if (a < 500) {
983         a++;
984     }
985     else if (a > 450) {
986         lcd.setCursor(0, 0);
987         lcd.print("Put your finger ");
988         lcd.setCursor(0, 1);
989         lcd.print("on the sensor ");
990         delay(1000);
991         lcd.clear();
992         delay(500);
993     }
994 }
995
996 void LCDHeartView(int bpm) {
997     lcd.setCursor(1, 1);
998     lcd.write(byte(1));
999     lcd.setCursor(0, 1);
1000    lcd.write(byte(2));
1001    lcd.setCursor(0, 0);
1002    lcd.write(byte(3));
1003    lcd.setCursor(1, 0);
1004    lcd.write(byte(4));
1005    lcd.setCursor(2, 0);
1006    lcd.write(byte(5));
1007    lcd.setCursor(3, 0);
1008    lcd.write(byte(6));
1009    lcd.setCursor(4, 1);
1010    lcd.write(byte(7));
1011    lcd.setCursor(2, 1);
1012    lcd.write(byte(8));
1013    lcd.setCursor(5, 0);
1014    lcd.print("Heart Rate");
1015    lcd.setCursor(5, 1);
1016    lcd.print(" ");
1017    lcd.print(bpm);
1018    lcd.print(" ");
1019    lcd.print("BPM");
1020 }
1021
1022 void alertSystem(int bpm) {
1023     if (bpm > 100 || bpm < 55) {
1024         int* p_speaker = speaker; // create a pointer to the pin v
1025         greenLED.off();
1026         redLED.blink(500);
1027         tone(*p_speaker, 262);
1028         delay(200);
1029         noTone(*p_speaker);
1030         delay(200);
1031     }
1032     else {
1033         greenLED.on();
1034     }
1035 }
1036
1037 void loop() {
1038     int myBPM = heartMonitor.getBeatsPerMinute();
1039
1040     initial(Instructions_view);
1041     if (heartMonitor.sawStartOfBeat()) {
1042         if (myBPM >= 120) {
1043             myBPM = myBPM / 2;
1044         }
1045         else if (myBPM <= 50) {
1046             myBPM = myBPM * 2;
1047         }
1048         bpmBuffer.push_back(myBPM);
1049
1050         if (bpmBuffer.size() >= BUFFER) {
1051             int* dynamicBuffer = (int*)malloc(BUFFER * sizeof(int));
1052             if (dynamicBuffer == nullptr) {
1053                 Serial.println("Error allocating memory");
1054                 return;
1055             }
1056             for (int i = 0; i < BUFFER; i++) {
1057                 dynamicBuffer[i] = bpmBuffer[i];
1058             }
1059             File sdcard_file = SD.open("data.txt", FILE_WRITE);
1060
1061             if (sdcard_file) {
1062                 for (int i = 0; i < BUFFER; i++) {
1063                     sdcard_file.println("BPM: ");
1064                     sdcard_file.println(dynamicBuffer[i]);
1065                     Serial.println("♥ A HeartBeat Happened ! ");
1066                     Serial.println("BPM: ");
1067                     Serial.println(dynamic
```


When debugging, there was an error due to the lack of proper inclusion of vector. A strange concept which was discovered was when changing from the stopwatch to the ethernet there were errors. This may be due to the previous code continuously running and not allowing the code to be completely uploaded. As a result, I found to run the ethernet code, the board had to be unplugged and replugged to successfully work. This had to be applied for the stopwatch code as well.

```
vector: No such file or directory
smartwatch:9:10: fatal error: vector: No such file or directory
#include <vector>
               ^
compilation terminated.
exit status 1
vector: No such file or directory
```

Test 1: The aim for this test is displaying the pulse sensor output into webserver.

Test 1 result: This result was recreating the previous response and this was successful.

Heart Monitoring System

Heart Rate:

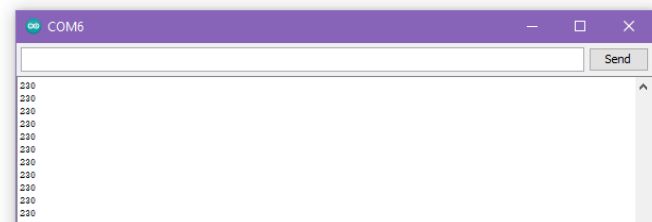
230

Temperature:

Humidity:

The code for this test is below.

Created by Phebe Le in Session 2 2023



```
wifi2
1  #define USE_ARDUINO_INTERRUPTS true
2  #include <SPI.h>
3  #include <Ethernet.h>
4  #include <PulseSensorPlayground.h>
5
6  byte mac[] = { 0x00, 0x1A, 0xBB, 0xCC, 0xDE, 0x02 };
7  IPAddress ip(192, 168, 5, 81);
8  EthernetServer server(80);
9
10 const int PulseWire = 0;
11 int Threshold = 547;
12 PulseSensorPlayground heartMonitor;
13
14 void setup()
15 {
16   Ethernet.begin(mac, ip);
17   server.begin();
18   Serial.begin(9600);
19   heartMonitor.analogInput(PulseWire);
20   heartMonitor.setThreshold(Threshold);
21   if (heartMonitor.begin()) {
22     Serial.println("We created a heartMonitor Object !"); //initialisation
23     delay(50);
24   }
25
26 void loop()
27 {
28   EthernetClient client = server.available();
29   int myBPM = heartMonitor.getBeatsPerMinute();
30   Serial.println(myBPM);
31   if (heartMonitor.sawStartOfBeat()) {
32     if (client) { // got client?
33       boolean currentLineIsBlank = true;
34       Serial.println("started");
35       while (client.connected()) {
36         if (client.available()) {
37           char c = client.read();
38           if (c == '\n' && currentLineIsBlank) {
39             client.println("HTTP/1.1 200 OK");
40             client.println("Content-Type: text/html");
41             client.println();
42             client.println("<HTML>");
43             client.println("<HEAD>");
44             client.println("<meta charset='UTF-8'>");
45             client.println("<meta name='author' content='Phebe Le'>");
46             client.println("<meta http-equiv='X-UA-Compatible' content='IE-edge'>");
47             client.println("<meta name='viewport' content='width=device-width, initial-scale=1.0'>");
48             client.println("<meta name='apple-mobile-web-app-capable' content='yes' />");
49             client.println("<meta name='description' content='heartbeat, temperature, humidity'> ");
50             client.println("<meta name='apple-mobile-web-app-status-bar-style' content='black-translucent' />");
51             client.println("<link rel='stylesheet' type='text/css' href='https://zandommerdtutorials.com/ethernetcss.css' />");
52
53             client.println("<TITLE>Heart Monitoring System</TITLE>");
54             client.println("</HEAD>");
55             client.println("<BODY>");
56             client.println("<H1>Heart Monitoring System</H1>");
57             client.println("<br />");
58             client.println("<br />");
59             client.println("<H2>Heart Rate:</H2>");
60
61             client.println(myBPM);
62
63             client.println("<br />");
64             client.println("<H2>Temperature:</H2>");
65             client.println("<br />");
66             client.println("<H2>Humidity:</H2>");
67             client.println("<br />");
68             client.println("<p>Created by Phebe Le in Session 2 2023</p>");
69             client.println("<br />");
70             client.println("</BODY>");
71             client.println("</HTML>");
72             break;
73           }
74         }
75       }
76       if (c == '\n') {
77         currentLineIsBlank = true;
78       }
79       else if (c != '\r') {
80         currentLineIsBlank = false;
81       }
82     }
83   }
84   delay(1);
85   client.stop();
86 }
87 }
88 }
```

Test 2:

Aim: The next test will be creating a refresh rate for the webserver to update the data values and adding restrictions to limit the BPM range.

Test 2 results:

These results showed to be successful yet inconsistent. As seen from the image below the data from the webserver is accurate with the current version of the Serial monitor. The started entry in the Serial Monitor is an indication of the webserver's refresh. Yet there was a delay due to the speed of the internet.

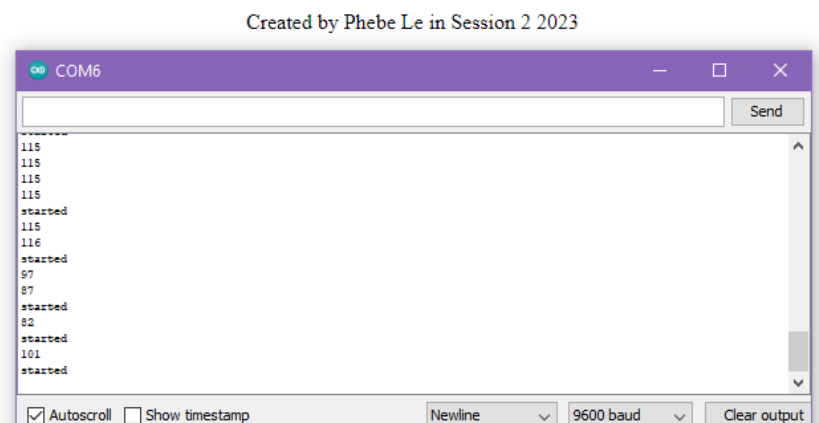
Heart Monitoring System

Heart Rate:

101

Temperature:

Humidity:



Test 3:

Aim: To add restrictions to the pulse sensor's BPM. This will be copied from the smartwatch code.

The finished ethernet and pulse sensor code is below.

```
if (heartMonitor.sawStartOfBeat()) {  
  if (myBPM >= 110) {  
    myBPM = myBPM / 2;  
  }  
  else if (myBPM <= 50) {  
    myBPM = myBPM * 2;  
  } else if (myBPM == 0) {  
    myBPM = myBPM + 50;  
  }  
  Serial.println(myBPM);  
  if (client) { // got client?  
    boolean currentLineIsBlank = true;  
    Serial.println("started");  
    while (client.connected()) {  
      if (client.available()) {  
        char c = client.read();  
        if (c == '\n' && currentLineIsBlank) {  
          client.println("HTTP/1.1 200 OK");  
          client.println("Content-Type: text/html");  
          client.println("Refresh: 1");  
          client.println();  
        }  
      }  
    }  
  }  
}
```

```
1  /*Created by Phebe Le | Session 2 | Heart Monitoring System
2  *
3  * This project is replicating a heart-rate watch system that uses output: LEDs,
4  * a buzzer, LCD display screen. Inputs: Pulse-Sensor. It also includes an SD
5  * card module which reads and writes data
6  *
7  * Due to the lack of space in the Arduino, the system is split up into two different modules
8  * a stationary system and mobile system.
9  *
10 * This section is Part 2: The stationary system consisting of a webserver with Ethernet connection
11 * and a pulse sensor
12 */
13
14
15 #define USE_ARDUINO_INTERRUPTS true
16 #include <SPI.h>
17 #include <Ethernet.h>
18 #include <PulseSensorPlayground.h>
19
20 byte mac[] = { 0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02 };
21 IPAddress ip(192, 168, 5, 81);
22 EthernetServer server(80);
23
24 const int PulseWire = 0;
25 int Threshold = 547;
26 PulseSensorPlayground heartMonitor;
27
28 void setup()
29 {
30   Ethernet.begin(mac, ip);
31   server.begin();
32   Serial.begin(9600);
33   heartMonitor.analogInput(PulseWire);
34   heartMonitor.setThreshold(Threshold);
35   if (heartMonitor.begin()) {
36     Serial.println("We created a heartMonitor Object !"); //initialisation
37     delay(50);
38   }
39
40 void loop()
41 {
42   EthernetClient client = server.available();
43   int myBPM = heartMonitor.getBeatsPerMinute();
44
45   if (heartMonitor.sawStartOfBeat()) {
46     if (myBPM >= 110) {
47       myBPM = myBPM / 2;
48     }
49     else if (myBPM <= 50) {
50       myBPM = myBPM * 2;
51     } else if (myBPM == 0) {
52       myBPM = myBPM + 50;
53     }
54     Serial.println(myBPM);
55     if (client) { // got client?
56       boolean currentLineIsBlank = true;
57       Serial.println("started");
58       while (client.connected()) {
59         if (client.available()) {
60           char c = client.read();
61           if (c == '\n' && currentLineIsBlank) {
62             client.println("HTTP/1.1 200 OK");
63             client.println("Content-Type: text/html");
64             client.println("Refresh: 1");
65             client.println();
66             Serial.println(myBPM);
67             if (client) { // got client?
68               boolean currentLineIsBlank = true;
69               Serial.println("started");
70               while (client.connected()) {
71                 if (client.available()) {
72                   char c = client.read();
73                   if (c == '\n' && currentLineIsBlank) {
74                     client.println("HTTP/1.1 200 OK");
75                     client.println("Content-Type: text/html");
76                     client.println("Refresh: 1");
77                     client.println();
78                     client.println("<HTML>");
79                     client.println("<HEAD>");
80                     client.println("<meta charset='UTF-8'>");
81                     client.println("<meta name='author' content='Phebe Le'>");
82                     client.println("<meta http-equiv='X-UA-Compatible' content='IE-edge'>");
83                     client.println("<meta name='viewport' content='width=device-width, initial-scale=1.0'>");
84                     client.println("<meta name='apple-mobile-web-app-capable' content='yes' />");
85                     client.println("<meta name='description' content='heartbeat, temperature, humidity'> ");
86                     client.println("<meta name='apple-mobile-web-app-status-bar-style' content='black-translucent' />");
87                     client.println("<link rel='stylesheet' type='text/css' href='https://randomerdutorials.com/ethernetcss.css' />");
88
89                     client.println("<TITLE>Heart Monitoring System</TITLE>");
90                     client.println("</HEAD>");
91                     client.println("<BODY>");
92                     client.println("<h1>Heart Monitoring System</h1>");
93                     client.println("<div />");
94                     client.println("<div />");
95                     client.println("<h2>Heart Rate:</h2>");
96                     client.println(myBPM);
97                     client.println("<div />");
98                     client.println("<div />");
99                     client.println("<p>Created by Phebe Le in Session 2 2023</p>");
100                    client.println("<div />");
101                    client.println("</BODY>");
102                    client.println("</HTML>");
103                    break;
104                  }
105                  if (c == '\n') {
106                    currentLineIsBlank = true;
107                  }
108                  else if (c != '\r') {
109                    currentLineIsBlank = false;
110                  }
111                }
112              }
113              delay(1);
114              client.stop();
115            }
116          }
117        }
118      }
119    }
120  }
```

21/3/23

Aim: To run the vector library and allow the smartwatch code to run

```

100 }
161 void loop() {
162     int myBPM = heartMonitor.get
163         initial(Instructions_view);
164
165     int memory[BUFFER];
166     bpmBuffer vector;
167     vector.setStorage(memory);
168     Serial.print("1");
169     if (heartMonitor.sawStartOfB
170         if (myBPM >= 120) {
171             myBPM = myBPM / 2;
172         }
173         else if (myBPM <= 50) {
174             myBPM = myBPM * 2;
175         }
176     Serial.print("1.5");
177     vector.push_back(BUFFER);
178     Serial.print("2");
179     if (vector.size() >= BUFFE
180         Serial.print("3");
181         int* dynamicBuffer = (in
182         Serial.print("4");
183         if (dynamicBuffer == nul
184             Serial.println("Error

```

Output Serial Monitor ✕

Message (Enter to send message to 'Arduino Uno')

```
Initializing SD cardcard initialized.  
We created a heartMonitor Object !  
0000000000000000000000000000000000000000000000000000000  
We created a heartMonitor Object !  
1111111111111111111111111111111111111111111111111111111  
We created a heartMonitor Object !  
1111111111111111111111111111111111111111111111111111111
```

```

}

void loop() {
    int myBPM = heartMonitor.getBeatsPerMinute();
    initial(Instructions_view);

    int memory[BUFFER];
    bpmBuffer vector;
    vector.setStorage(memory);
    Serial.println("1");
    if (heartMonitor.sawStartOfBeat()) {
        if (myBPM >= 120) {
            myBPM = myBPM / 2;
        } else if (myBPM <= 50) {
            myBPM = myBPM * 2;
        }
        Serial.println("1.5");

        Serial.println("2");
        if (vector.size() >= BUFFER) {
            Serial.println("3");
            int* dynamicBuffer = (int*)malloc(BUFFER * sizeof(int));
            Serial.println("3-4");
            Serial.println("4");
            if (dynamicBuffer == nullptr) {
                Serial.println("Error allocating memory");
                return;
            }
        }
    }
}

```

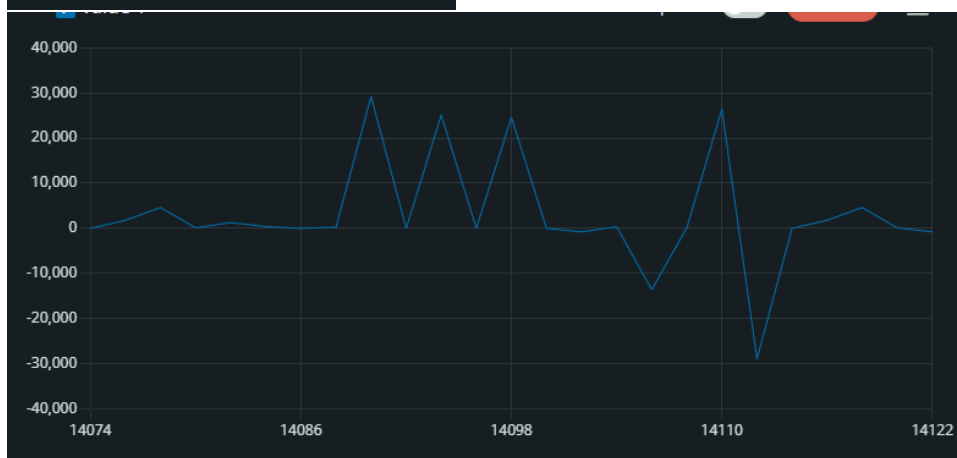
```

    }
    sdcard_file.close();
    Serial.println("closed");
} else {
    Serial.println("error opening data.txt");
}
delay(1000);
LCDHeartView(myBPM);
Instructions_view = 0;
alertSystem(myBPM);
free(memory);
// return 0;
// return;
}
delay(20);
}
}

```

```
1
Initializing SD cardcard initialized.
We created a heartMonitor Object !
```

```
7
8
▼ A HeartBeat Happened !
BPM:
25960
sucess opening data.txt
7
8
▼ A HeartBeat Happened !
BPM:
29472
sucess opening data.txt
7
8
▼ A HeartBeat Happened !
BPM:
28261
sucess opening data.txt
7
8
▼ A HeartBeat Happened !
BPM:
28531
```



Within these several pieces of codes and outputs. There has been one major focus which is to allow the vectors to dynamically store the allocated malloc of data needed. These results whilst are accurate in storing them dynamically, the BPM value is absurdly inaccurate. This may be due to the use of vectors, whilst being dynamically allocated. Alongside this the alert system and the LCD functions do not work. As a result, the first requirement will be to create stronger restrictions for this data range, and then change the location of the functions to be within the SD writing loop.

After editing and taking these steps the results show to be favourable (see below). The inclusion of serial print 7 and 8 are also included due to testing the accuracy and debugging errors. In counting this, I also found that the buffer function also works correctly as when it reaches 20 data entries the vector resets and begins the process

once again (see the 'closed' line in the first image). The restrictions also worked after changing different numbers and testing it (see second image). The third image is reading the file that had stored the data values and plotting it onto a graph.

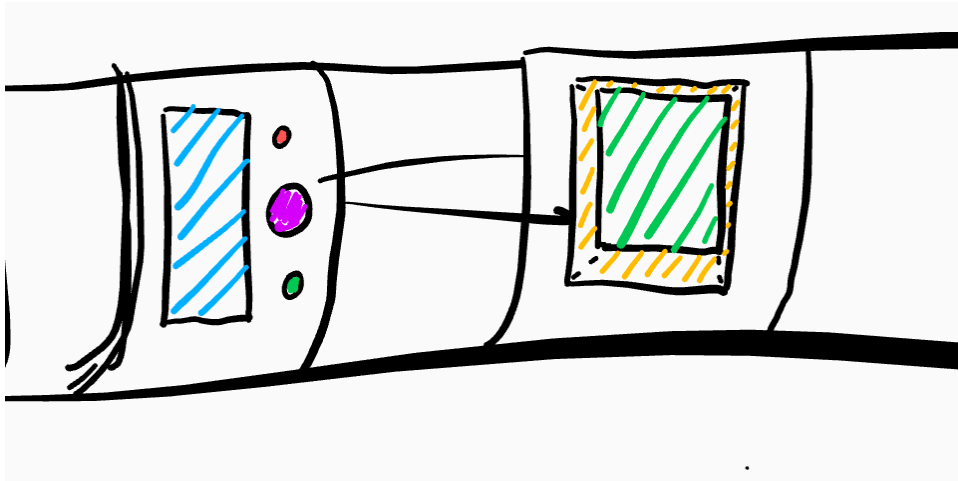
```
8
▼ A HeartBeat Happened !
BPM:
1792
sucess opening data.txt
7
8
▼ A HeartBeat Happened !
BPM:
4609
sucess opening data.txt
7
8
▼ A HeartBeat Happened !
BPM:
114
sucess opening data.txt
7
8
▼ A HeartBeat Happened !
BPM:
15616
sucess opening data.txt
closed
```

```
memory[i] = memory[i] - 80;
} else if (memory[i] <= 50 && memory[i]
//Serial.println("50/40");
memory[i] = memory[i] * 2;
} else if (memory[i] <= 39) {
//Serial.println("39");
memory[i] = memory[i] + 70;
}
// } else if (memory[i] < 10) {
// memory[i] = 50 + memory[i];
```



Housing Development

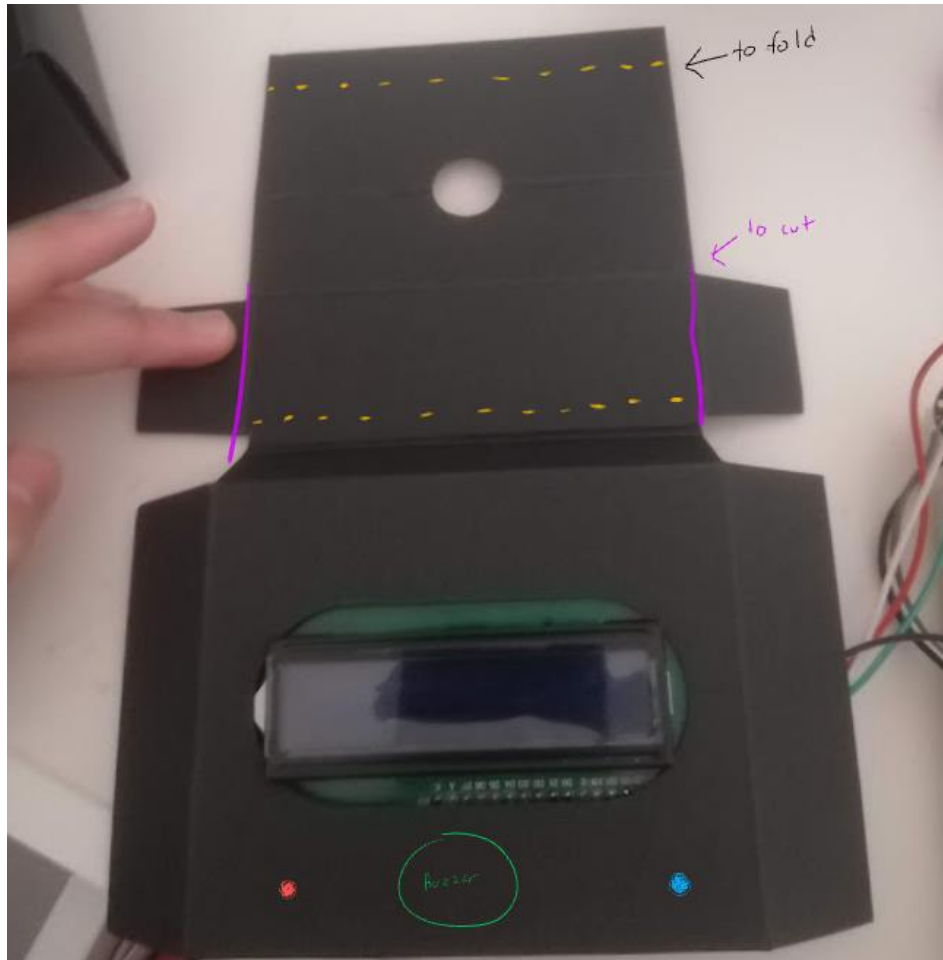
To develop the two sections, I will segregate the design into two pieces. The stopwatch and the 'arm piece'. The arm piece will contain cabling, the Arduino shield and Arduino uno. The smartwatch will contain all other components. The initial sketch of the current design is shown below.



The smartwatch

Currently the main goals for this section is to make the smart device manageable and wearable whilst containing all required elements.

Using old cardboard from smart devices, a piece that fit the LCD screen was found. This may be the main template of the design. This will be attached to elastic or a strong material which withstands heavy weights



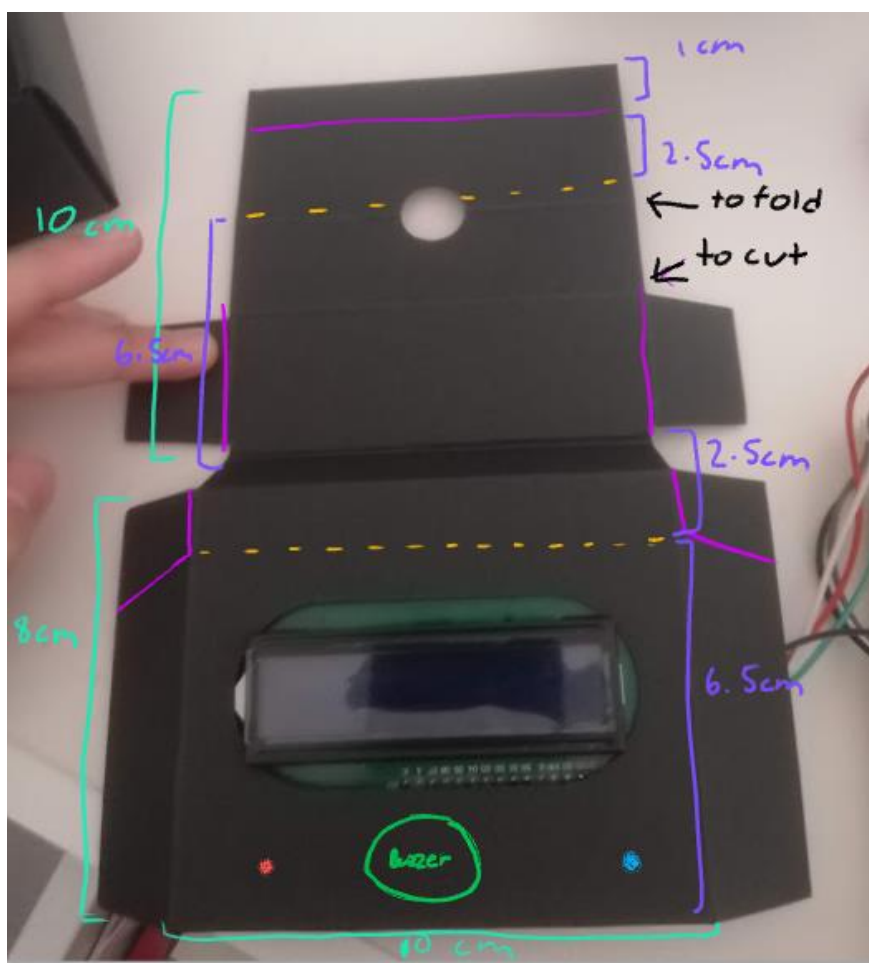
For this design the main plan is to use this box like shape to create the top of the smartwatch that is viewable to others. This will contain an LCD screen, a buzzer and two LEDs. Due to the design of this piece of cardboard a two-layer box will be applicable (see the cardboard fold on the right)

A drawing of this visualisation can be seen below





Yet despite this material, the size is significantly too large to be placed on a wrist. Due to this, I created a plan to edit the design.



The green lines with the measurements show the cardboards measurements. Whilst the purple is used to show the measurement that will be implemented onto the cardboard. The pink line will be places to cut. The yellow line illustrates the places to bend. These will be created tomorrow.

Wristband

To create the wristband another piece of cardboard will be required. The top facing part of the wristband of this will be glued to the bottom of the breadboard's cardboard. This is what it will look like when folded correctly. Whilst, this design is lacking in regards to aesthetics, due to time constraints this will be the best solution. If there was to be more

time, I would use a softer material such as an elastic band and attach a fabric-like material beneath the smartwatch to create a more comfortable design.

Wristband

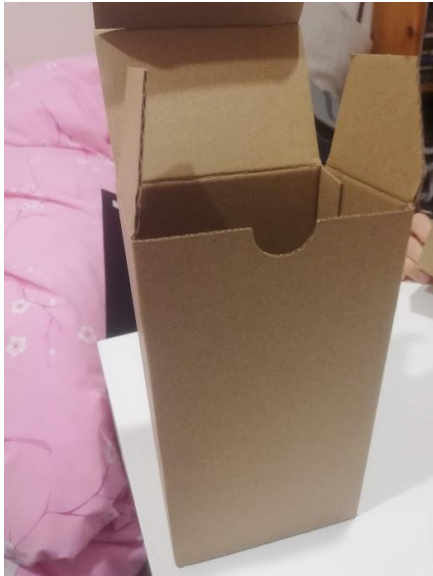
1. Cut out the cardboard
2. Make a hole and stick a metal cable tie through it
3. On the other side place a clip to slot the cable tie through
4. Connect the cable tie with the clip

End result:

- I dislike my result so I will opt for just using a cardboard box and cut off the back and front. To be able to sense heartbeat through the contraption, a hole was added.



Initial result



New concept

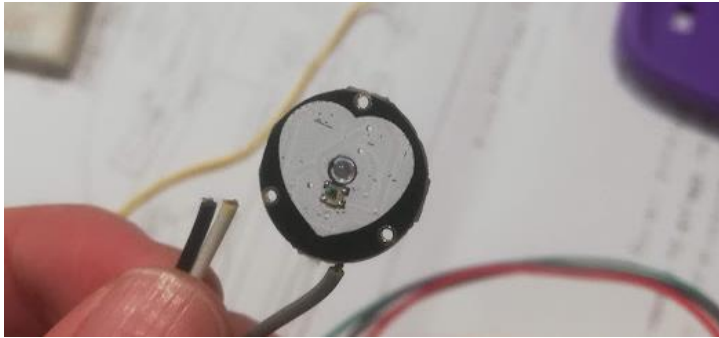
22/3/23

The initial structure of the smartwatch was adhered to as previous instructions state and this was the result. The concept the design has also changed as the Arduino, Ethernet shield and breadboard will be on one major piece instead of two separate pieces



This result was better than expected as it allowed room for the ethernet shield and the Arduino. This led me to assemble the main circuit on the smaller breadboard. Yet whilst assembling a major issue occurred: the heart sensor broke. If possible, this may be fixed

tomorrow but to temporarily (and unreliably) solve this I have sticky taped the wires to the pulse sensors which allows it to work 30% of the time.



Despite this issue, I continued to build the housing which resulted in this. Whilst I am happy with the general design of the watch, I also think that this design is significantly larger than the average watch.



The wristband-like concept was then added and aligned with the smartwatch housings hole.



