Chapter 1: Introduction

1.1 Project Summary

Many serious diseases and disorders e.g. heart failures need close and continuous monitoring procedure after the diagnosis; so that it can prevent further damage to the mentioned diseases or disorders. Monitoring these types of heart patients takes place at hospitals or healthcare centers. For an instance, Heart arrhythmias needs continuous monitoring. However; the patients are often given early discharge, owing to need of hospital bed for another patient on the waiting list, who needs to be treated as soon as possible. So, whenever there is a need such that patient needs long term monitoring, this heartbeat monitoring system can be of great help.

1.1.1 Purpose

The main purpose of this application is to continuously monitor heartbeat of a patient when they are travelling and they need to be continuous monitored. Also, patients who need to be hospitalized, but are not able to do so, because of unavailability of hospital bed, can be monitored using this system. So, the doctor can get an idea about how severe the disease is.

1.2 Scope

Number of hours waiting in hospitals or ambulatory patient monitoring/treatment, are other known issues for both the healthcare institutions and the patients. This project provides healthcare institutions to maximize the quality of healthcare services by controlling costs. As the population is increasing continuously, demand for services is also increasing gradually; the ability to maintain the quality and availability of care, while effectively managing financial and human resources, can be easily achieved by this project. Making use of nowadays modern technology in this context is the most important factor that makes such communication system successful.

1.3 Objective

My objective is to learn how arduino works, more about embedded technology, how the heartbeat measuring through sensor works. My main aim behind creating this application is giving a helping hand to healthcare industry of our globe.

Chapter 2: System Requirements Study

2.1 User Characteristics

- This application can be used by anyone who need to monitor his/her heartbeats at regular interval.
- It’s foremost focus is to help people who continuously need to measure their heartbeat and it also focuses to help in healthcare industry.

2.2 Tools & Technology Used

▪ Platform Used	: Arduino 1.8.10
▪ Programming language	: Arduino

Chapter 3: System Design

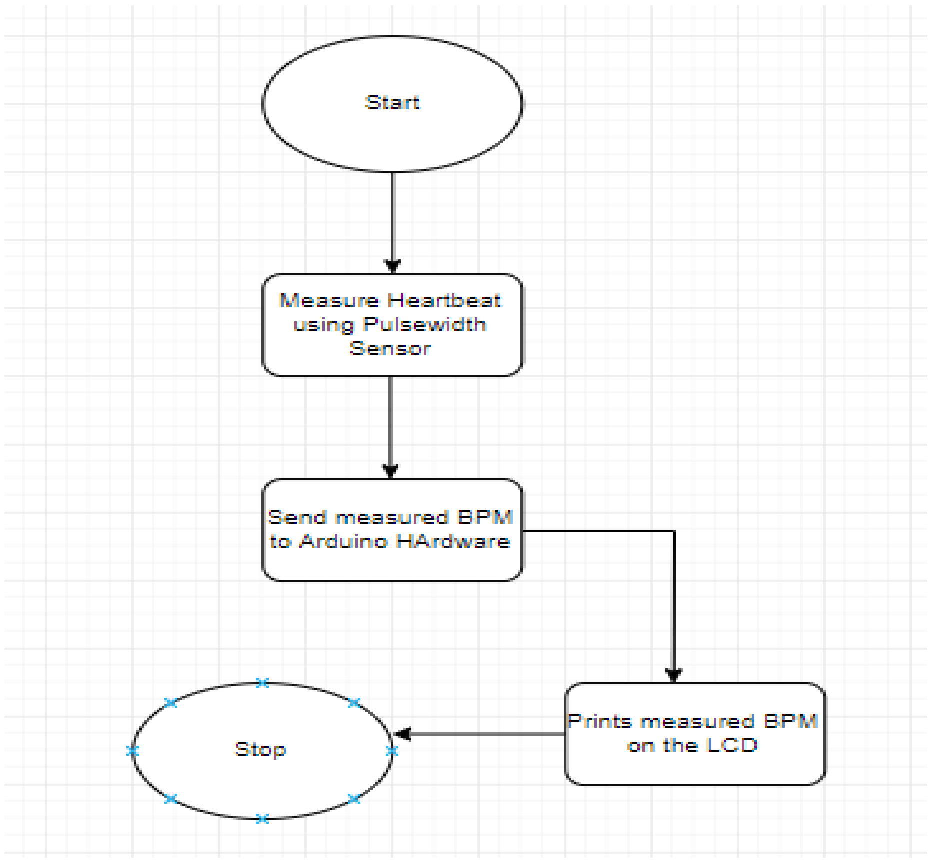

Start
Measure Heartbeat using Pulsewidth Sensor
Send measured BPM to Arduino HArdware
Prints measured BPM on the LCD
Stop

Fig 3.1.1 Project Flow

Here, as soon as we place our finger on the pulse-width sensor, the sensor senses our heartbeat in bpm and sends the data on Arduino hardware. This Arduino hardware further passes this data and prints it on 16 X 2 LCD Display.
So, in this way, it continuously senses the data and prints it on LCD Display.

Chapter 4 Implementation Planning

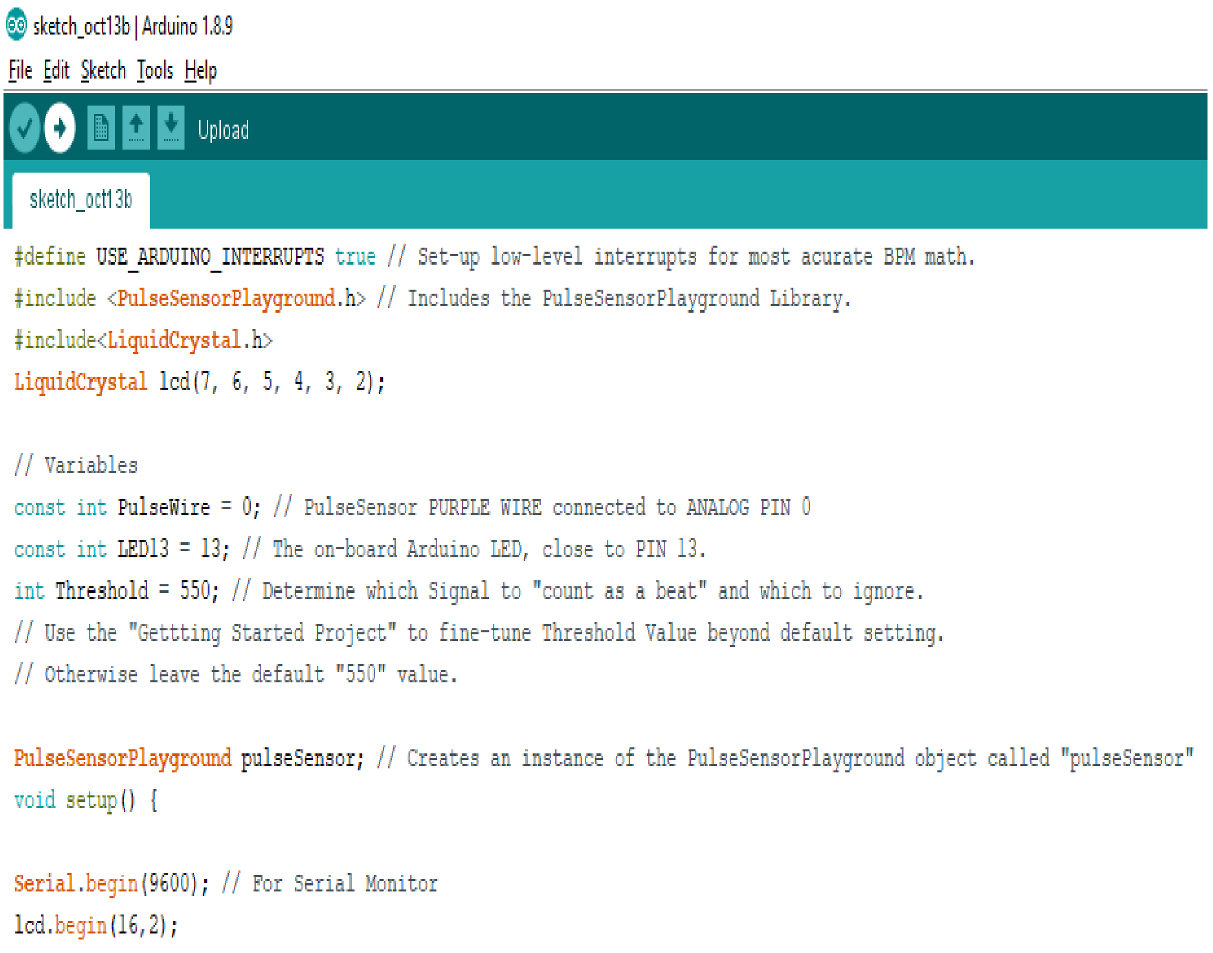
4.1 Implementation Environment

It is an Embedded based application. It is carried out in an Arduino IDE. Arduino Software is one of the many of them.
Arduino IDE is an open-source software program. This software allows users to write and upload code within a real-time work environment.

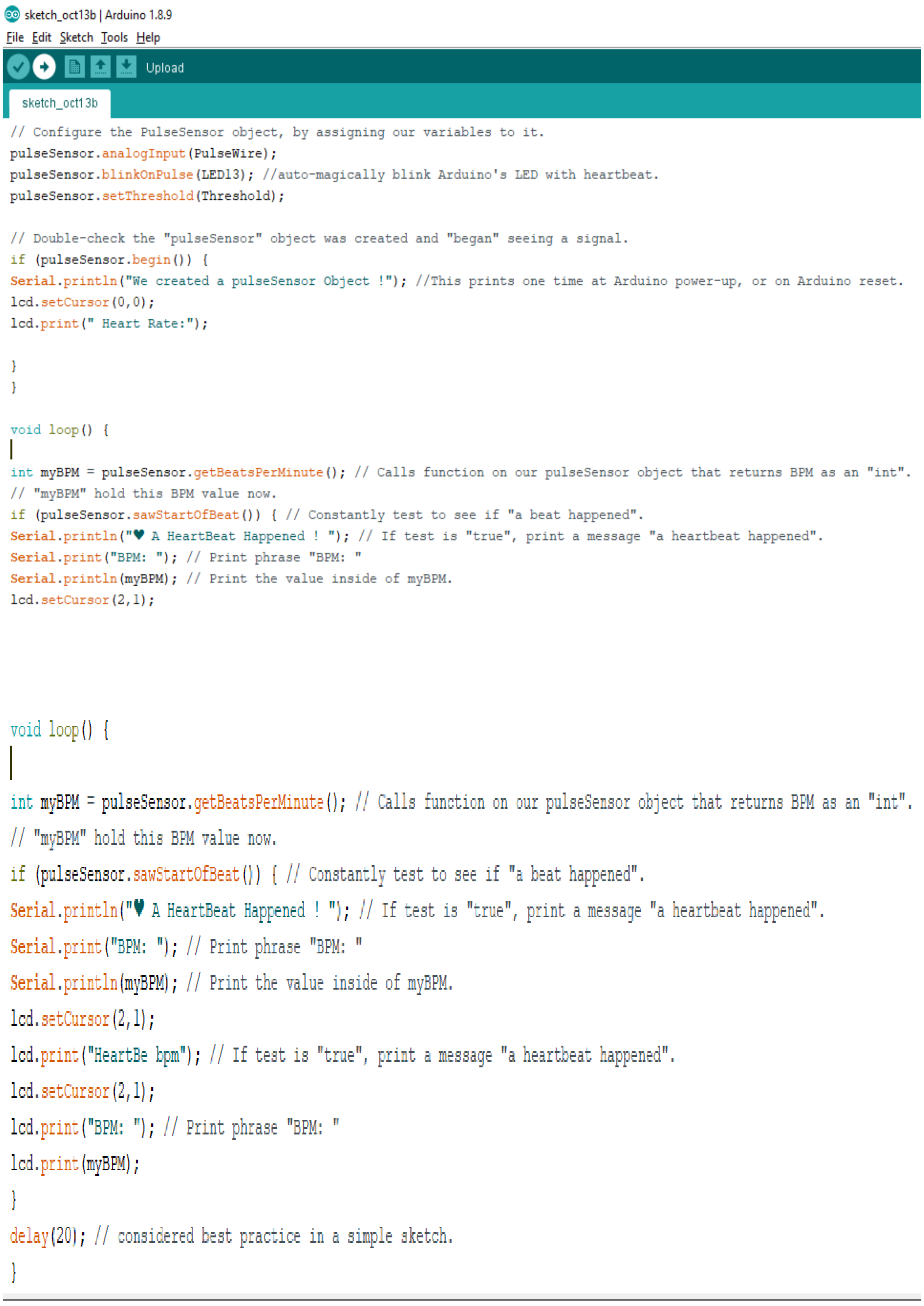
4.2 Coding Standards

Sample code of the program (Code Snippets)

GUI CODE: INO File



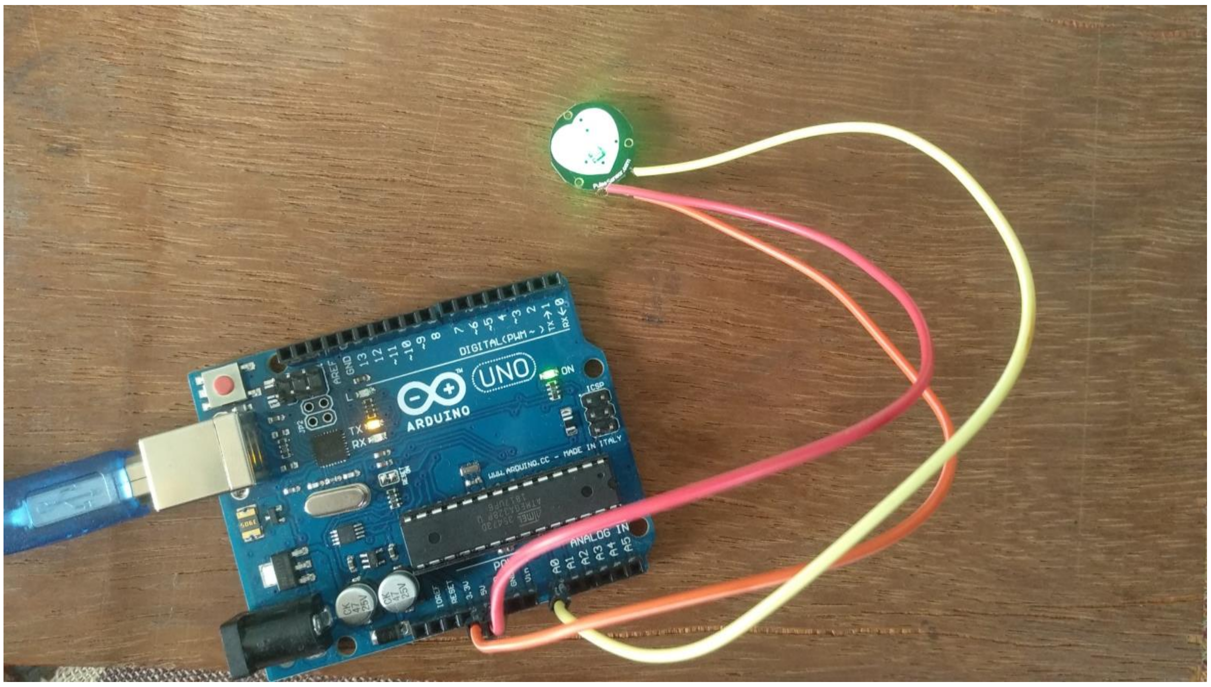
```
#define USE_ARDUINO_INTERRUPTS true // Set-up low-level interrupts for most acurate BPM math.
#include <PulseSensorPlayground.h> // Includes the PulseSensorPlayground Library.
#include<LiquidCrystal.h>
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);

// Variables
const int PulseWire = 0; // PulseSensor PURPLE WIRE connected to ANALOG PIN 0
const int LED13 = 13; // The on-board Arduino LED, close to PIN 13.
int Threshold = 550; // Determine which Signal to "count as a beat" and which to ignore.
// Use the "Gettting Started Project" to fine-tune Threshold Value beyond default setting.
// Otherwise leave the default "550" value.

PulseSensorPlayground pulseSensor; // Creates an instance of the PulseSensorPlayground object called "pulseSensor"
void setup() {

Serial.begin(9600); // For Serial Monitor
lcd.begin(16,2);
```



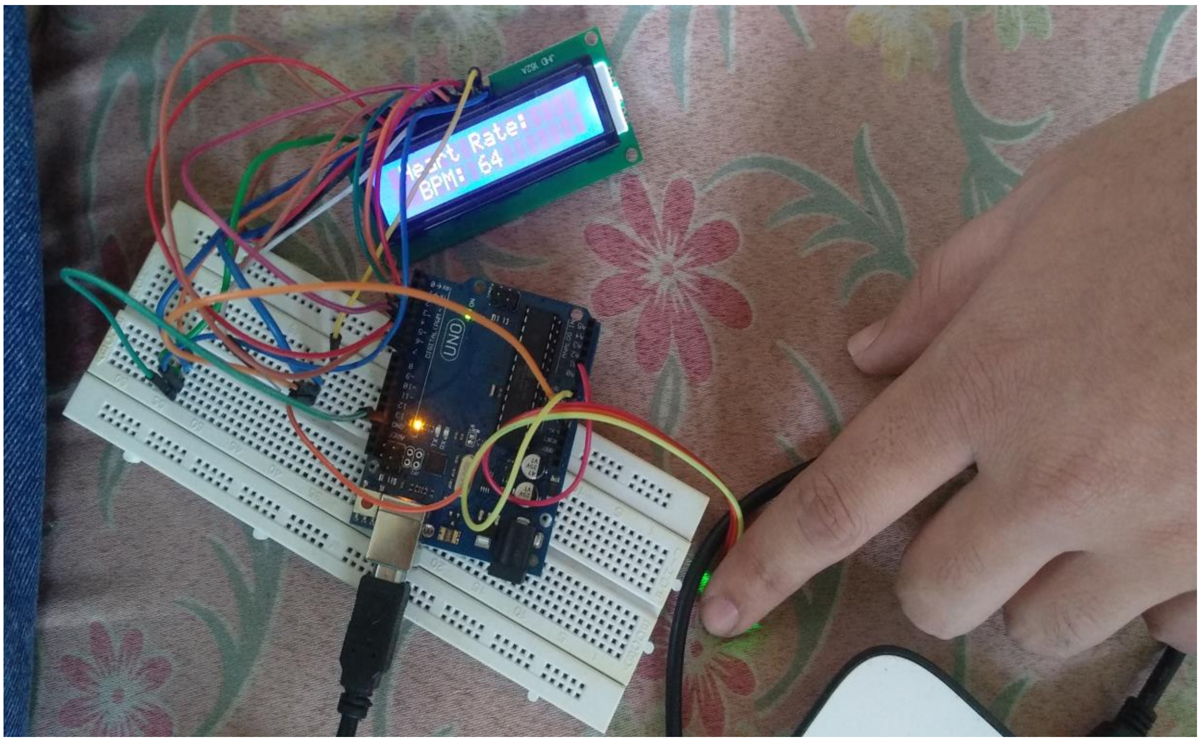
```
// Configure the PulseSensor object, by assigning our variables to it.
pulseSensor.analogInput(PulseWire);
pulseSensor.blinkOnPulse(LED13); //auto-magically blink Arduino's LED with heartbeat.
pulseSensor.setThreshold(Threshold);

// Double-check the "pulseSensor" object was created and "began" seeing a signal.
if (pulseSensor.begin()) {
Serial.println("We created a pulseSensor Object !"); //This prints one time at Arduino power-up, or on Arduino reset.
lcd.setCursor(0,0);
lcd.print(" Heart Rate:");

}
}

void loop() {

int myBPM = pulseSensor.getBeatsPerMinute(); // Calls function on our pulseSensor object that returns BPM as an "int".
// "myBPM" hold this BPM value now.
if (pulseSensor.sawStartOfBeat()) { // Constantly test to see if "a beat happened".
Serial.println("♥ A HeartBeat Happened ! "); // If test is "true", print a message "a heartbeat happened".
Serial.print("BPM: "); // Print phrase "BPM: "
Serial.println(myBPM); // Print the value inside of myBPM.
lcd.setCursor(2,1);
```

```
void loop() {

int myBPM = pulseSensor.getBeatsPerMinute(); // Calls function on our pulseSensor object that returns BPM as an "int".
// "myBPM" hold this BPM value now.
if (pulseSensor.sawStartOfBeat()) { // Constantly test to see if "a beat happened".
Serial.println("♥ A HeartBeat Happened ! "); // If test is "true", print a message "a heartbeat happened".
Serial.print("BPM: "); // Print phrase "BPM: "
Serial.println(myBPM); // Print the value inside of myBPM.
lcd.setCursor(2,1);
lcd.print("HeartBe bpm"); // If test is "true", print a message "a heartbeat happened".
lcd.setCursor(2,1);
lcd.print("BPM: "); // Print phrase "BPM: "
lcd.print(myBPM);
}
delay(20); // considered best practice in a simple sketch.
}
```

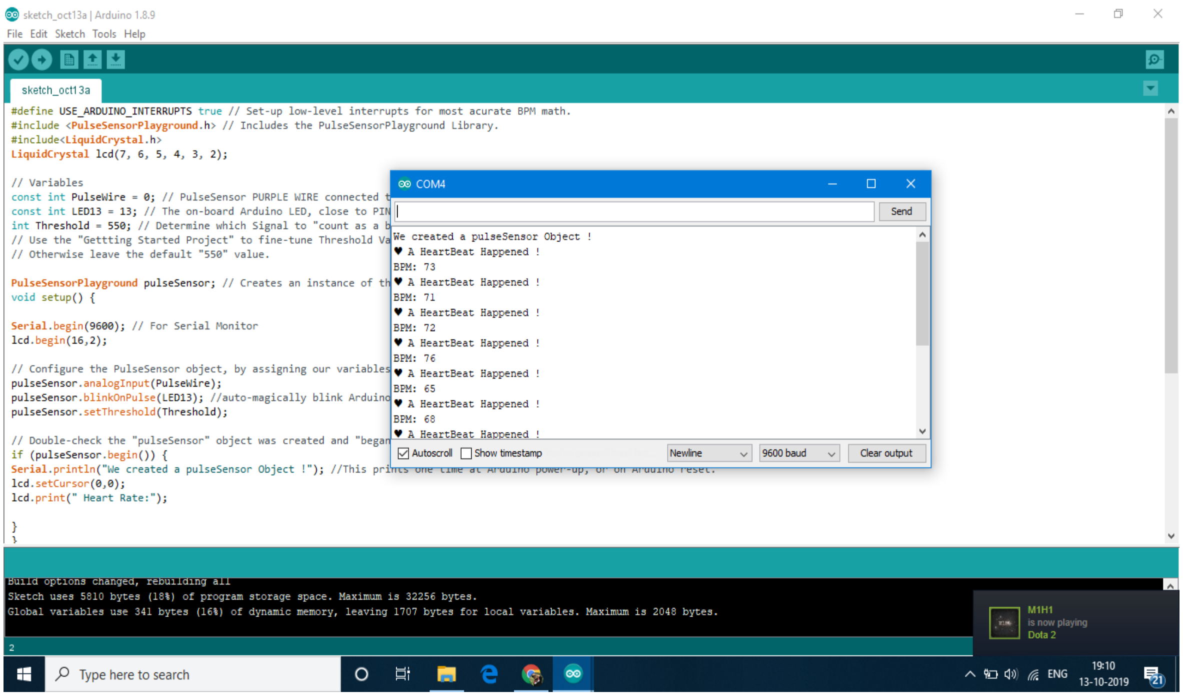
4.3 Snapshots of project



4.3.1- Arduino UNO and pulse-width Sensor



4.3.2- Heartbeat Monitoring Image-I



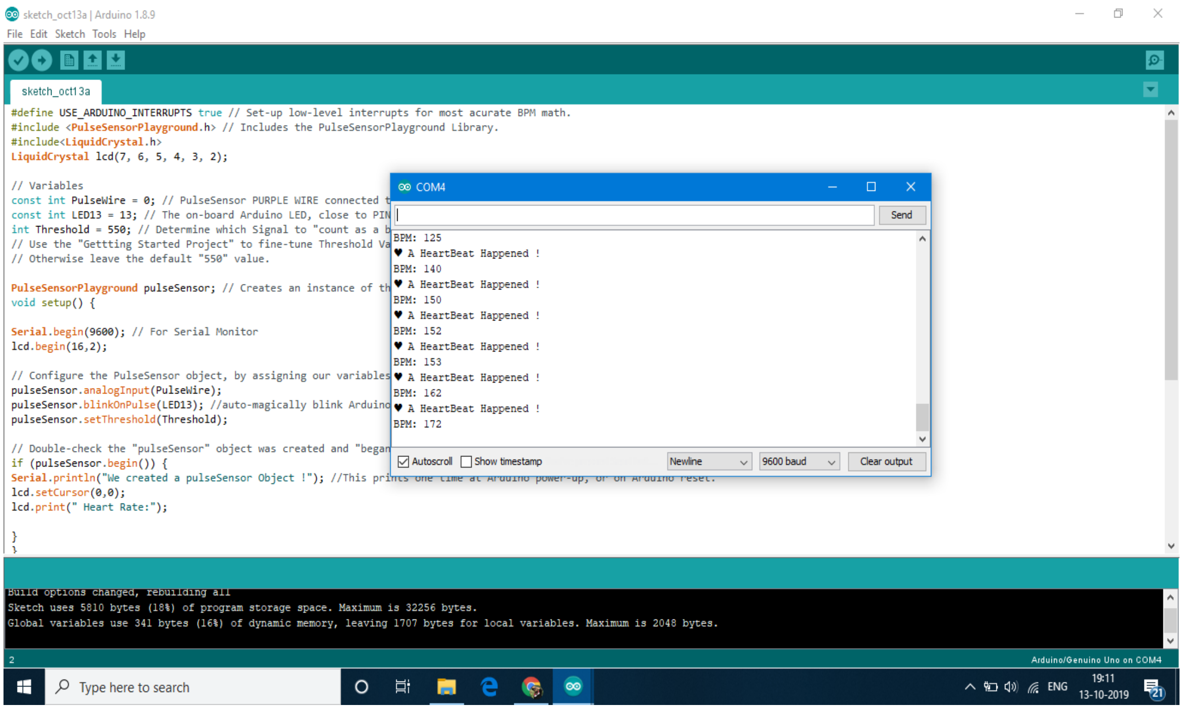
```
#define USE_ARDUINO_INTERRUPTS true // Set-up low-level interrupts for most acurate BPM math.
#include <PulseSensorPlayground.h> // Includes the PulseSensorPlayground Library.
#include<LiquidCrystal.h>
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);

// Variables
const int PulseWire = 0; // PulseSensor PURPLE WIRE connected t
const int LED13 = 13; // The on-board Arduino LED, close to PIN
int Threshold = 550; // Determine which Signal to "count as a b
// Use the "Gettting Started Project" to fine-tune Threshold Va
// Otherwise leave the default "550" value.

PulseSensorPlayground pulseSensor; // Creates an instance of th
void setup() {

Serial.begin(9600); // For Serial Monitor
lcd.begin(16,2);

// Configure the PulseSensor object, by assigning our variables
pulseSensor.analogInput(PulseWire);
pulseSensor.blinkOnPulse(LED13); //auto-magically blink Arduino
pulseSensor.setThreshold(Threshold);

// Double-check the "pulseSensor" object was created and "began
if (pulseSensor.begin()) {
Serial.println("We created a pulseSensor Object !"); //This prints one time at Arduino power-up, or on Arduino reset.
lcd.setCursor(0,0);
lcd.print(" Heart Rate:");

}
}
```

```
We created a pulseSensor Object !
♥ A HeartBeat Happened !
BPM: 73
♥ A HeartBeat Happened !
BPM: 71
♥ A HeartBeat Happened !
BPM: 72
♥ A HeartBeat Happened !
BPM: 76
♥ A HeartBeat Happened !
BPM: 65
♥ A HeartBeat Happened !
BPM: 68
♥ A HeartBeat Happened !
```

[✓] Autoscroll [ ] Show timestamp — Newline — 9600 baud

```
Build options changed, rebuilding all
Sketch uses 5810 bytes (18%) of program storage space. Maximum is 32256 bytes.
Global variables use 341 bytes (16%) of dynamic memory, leaving 1707 bytes for local variables. Maximum is 2048 bytes.
```

4.3.3- Serial Monitor readings when finger is placed



```
#define USE_ARDUINO_INTERRUPTS true // Set-up low-level interrupts for most acurate BPM math.
#include <PulseSensorPlayground.h> // Includes the PulseSensorPlayground Library.
#include<LiquidCrystal.h>
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);

// Variables
const int PulseWire = 0; // PulseSensor PURPLE WIRE connected t
const int LED13 = 13; // The on-board Arduino LED, close to PIN
int Threshold = 550; // Determine which Signal to "count as a b
// Use the "Gettting Started Project" to fine-tune Threshold Va
// Otherwise leave the default "550" value.

PulseSensorPlayground pulseSensor; // Creates an instance of th
void setup() {

Serial.begin(9600); // For Serial Monitor
lcd.begin(16,2);

// Configure the PulseSensor object, by assigning our variables
pulseSensor.analogInput(PulseWire);
pulseSensor.blinkOnPulse(LED13); //auto-magically blink Arduino
pulseSensor.setThreshold(Threshold);

// Double-check the "pulseSensor" object was created and "began
if (pulseSensor.begin()) {
Serial.println("We created a pulseSensor Object !"); //This prints one time at Arduino power-up, or on Arduino reset.
lcd.setCursor(0,0);
lcd.print(" Heart Rate:");

}
}
```

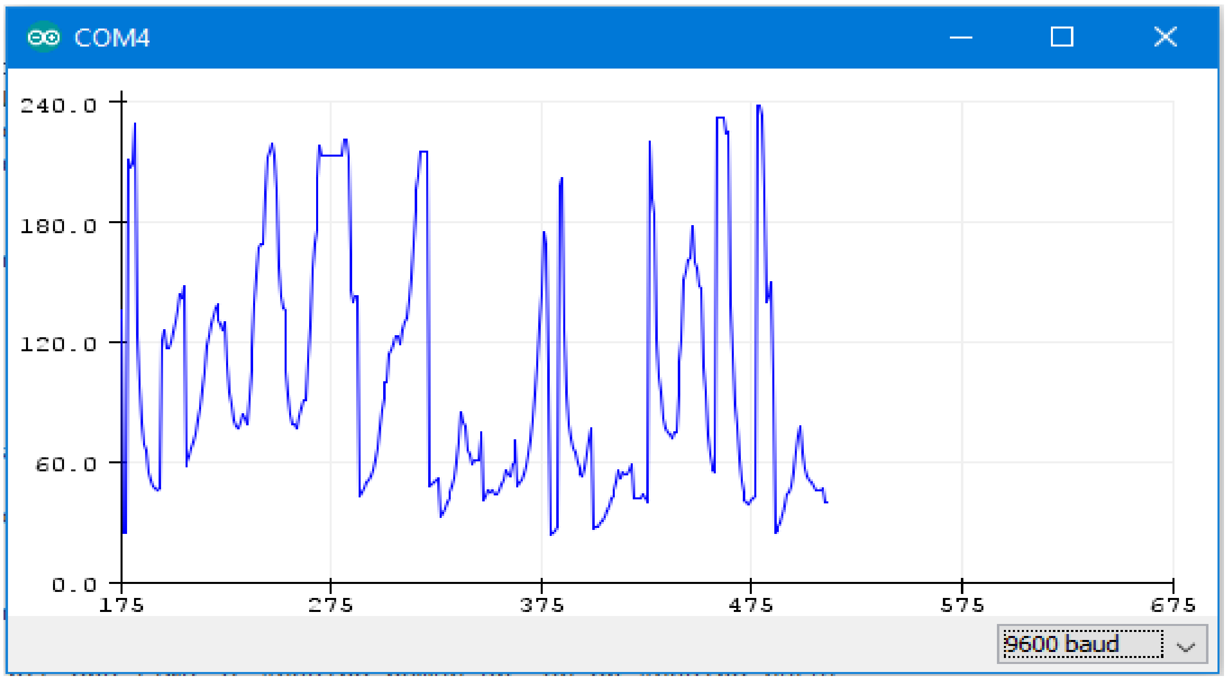
```
BPM: 125
♥ A HeartBeat Happened !
BPM: 140
♥ A HeartBeat Happened !
BPM: 150
♥ A HeartBeat Happened !
BPM: 152
♥ A HeartBeat Happened !
BPM: 153
♥ A HeartBeat Happened !
BPM: 162
♥ A HeartBeat Happened !
BPM: 172
```

[✓] Autoscroll [ ] Show timestamp — Newline — 9600 baud

```
Build options changed, rebuilding all
Sketch uses 5810 bytes (18%) of program storage space. Maximum is 32256 bytes.
Global variables use 341 bytes (16%) of dynamic memory, leaving 1707 bytes for local variables. Maximum is 2048 bytes.
```

sketch_oct13a | Arduino 1.8.9
File Edit Sketch Tools Help
sketch_oct13a
COM4
Send
Clear output
Arduino/Genuino Uno on COM4
Type here to search
ENG
19:11
13-10-2019

4.3.4- Serial Monitor readings when finger is not placed



4.3.5- Heartbeat graph generated

Chapter 5: Limitations and Future Enhancement

5.1 Limitations

- The only limitation of this project is that it will not display the heart-rate bpm with 100% accuracy, as other factors which are in the environment will slightly affect it.
- For Eg; If we are measuring the heartbeat using pulse sensor in a very noisy area, it will not give 100% accuracy. Though noise cancellation algorithms are present in sensor, it will be able to resist only limited amount of noise.

5.2 Future Enhancement

- In future, we are planning to build a software through which this reports generated can be sent directly to the doctor with the help of GSM technology.
- For instance, we are planning to connect this application with GSM technology, So by doing so, as soon as, heartbeat goes out of a given range, the text message can be sent directly on the doctors phone.

Chapter 6 :Conclusion

Heart related disease is one of the indispensable causes of untimely deaths in world, and heart beat values are the only possible diagnostic tool that could promote early detection of cardiac events. So, modern wireless technology and sensors are the only solutions which can help to prevent this types of diseases. This will help to decrease a lot of burden to healthcare domain. In this project, the heart beat rate of the patient is sensed. When the sensor detects a heart beat rate, it will display it on LCD which in turn will give an idea to patient that the patient needs medical attentions, thus greatly improving his or her chances of survival.

References:-

1. https://components101.com/sensors/pulse-sensor
2. https://www.researchgate.net/publication/317117507_Heartbeat_and_Temperature_Monitoring_System_for_Remote_Patients_using_Arduino
3. https://www.irjet.net/archives/V5/i3/IRJET-V5I3254.pdf
4. https://medium.com/@chawlamahima76/heartbeat-and-body-temperature-monitoring-using-arduino-cf0a339b50f