PROGRESS REPORT ON PATIENT HEALTH MONITORING SYSTEM

-BY ISHA LALCHANDANI

ABSTRACT:

Remote patient monitoring has become an important part of the healthcare industry since travelling to the doctor or their availability 24 X 7 is not always necessary, this project reduces the need to visit the doctor by monitoring temperature and pulse rate and storing this local data into the centralised repository connected to the server. The microcontroller Arduino in coded in C language and connected through the WIFI module ESP8266, this is an integrated TCP/IP protocol Stack that gives access to WIFI network. Sensors used are:

- BODY TEMPERATURE MODULE(DS18B20)
- HEARTBEAT MODULE

ThingSpeak is used to store and retrieve data using HTTP protocol over internet which enables creation of sensor login applications, it allows users to analyse and visualise uploaded data, perform statistical analysis on data, write data to ThingSpeak channel.

KEYWORDS:

Remote Monitoring, Arduino, WIFI module, Sensors, ThingSpeak

TABLE OF CONTENTS

Contents

LIST OF FIGURES

Contents

1 Introduction:

In the world of exponentially growing technologies, IOT is continuing to make its mark in almost every sector that exists. IOT and its applications have completely revolutionaries the way the world operates now. It's crucial to support the correctness and completeness of our daily needs. Nowadays, it has become a very time absorbing task for both doctor and patient to monitor his/her health and progress report. To overcome this major problem IOT plays a major role by introducing RPM system. With the help of RPM, the doctor can monitor patient's health without the patient visiting him. It increases access to health services with minimal cost and time while increasing the efficiency and number of patients that doctor can look after. This project has given us the opportunity to make an efficient remote patient monitoring system which measures heartbeat and temperature of patient and records it on a centralised server, thereby increasing quality of life.

2 Objective:

To monitor and analyse the pulse rate and temperature of a patient with the help of IOT based system.

Sub Objectives:

1. Setup IOT network along with sensors.

2. Setup of ThingSpeak server

3. Analysis and visualisation.

3 Design:

This project is designed to make a working device for monitoring the patient's temperature and heartbeat with the help of sensors.
The circuit is designed such that the positive pin of the LED is attached to the pin 13, because it has an inbuilt resistor and the negative pin is attached to ground. The LED is turned ON when the Voltage level is on HIGH and is turned off when the Voltage level is on LOW. This fluctuation of voltage from LOW to HIGH and from HIGH to LOW with a gap of 50 ms, results in flickering of the LED light.
The Temperature sensor DS18B20 consists of 3 wires-

BLACK WIRE: It is Negatively charged and is connected to the ground
YELLOW WIRE: It is used for I/O
RED WIRE: It is Positively charged and is connected to power source

The OneWire library is used to take I/O from one wire and can also be used to power the device (known as parasitic way). The instance of OneWire (object) is referred as a parameter for the DallasTemperature object.
The DallasTemperature contains all the calculations regarding the temperature sensor DS18B20 and is freely available under the GNU license. It is very accurate and inexpensive.

The BLACK wire of the temperature sensor is connected to the breadboard with the help of a Male to Male jumper wire, placed in parallel to it, is then attached to the ground of Arduino.
The RED wire is connected to the breadboard (working like a system bus) with the help of a Male to Male jumper wire. Another jumper wire is the connected in series to the digital pin of 5 Volt, providing power.
The YELLOW wire is connected to the digital pin 2 as initialised in the code with the help of a male to male jumper wire. Then connect the YELLOW wire to the RED wire with the help of a jumper wire, this uses the logic of OneWire, that provides energy to the system. We also connect a resistor with the RED wire.The Heartbeat Sensor consists of 3 wires :-
WHITE WIRE: It is negatively charged and connected to the ground
BLACK WIRE: It is positively charged and connected to the power source
GREY WIRE: It will be used for input and will be connected to A0 i.e. analog input 0.
Currently, all the wires of the temperature sensor is directly connected to the Arduino board. The white wire is pinned to the ground, black wire is pinned to the 5V power pin while the grey pin is connected to the analog input pin A0. This is the initial setup of the heartbeat sensor with Arduino, to see the output reading of BPM we can access it through the serial monitor where we can select the baud rate we have initialised and see the results once the heartbeat sensor comes into the contact with the body.

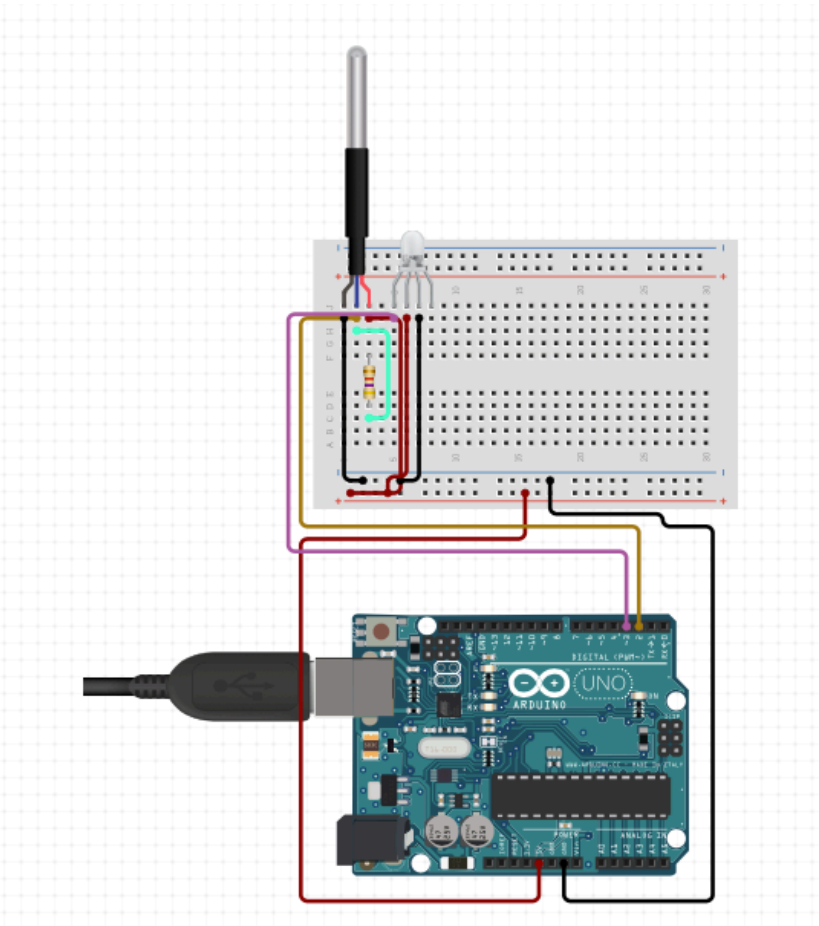


FIGURE 1:TEMPERATURE SENSOR MODULE

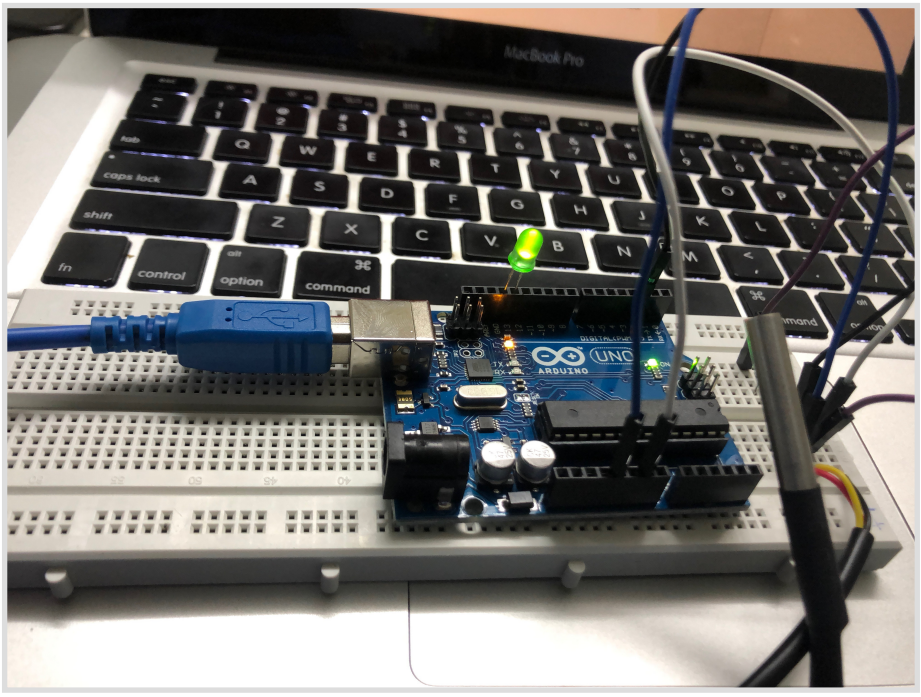


FIGURE2:TEMPERATURE SENSOR WITH ARDUINO WORKING MODULE

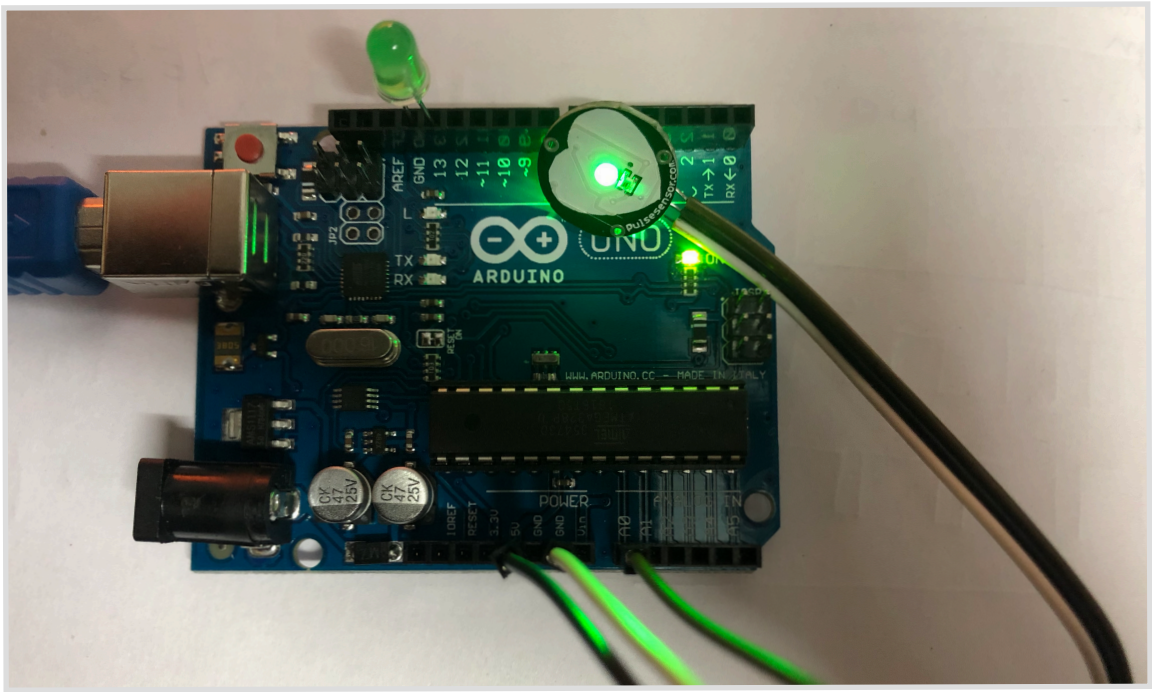


FIGURE3:HEARTBEAT SENSOR WITH ARDUINO WORKING MODULE

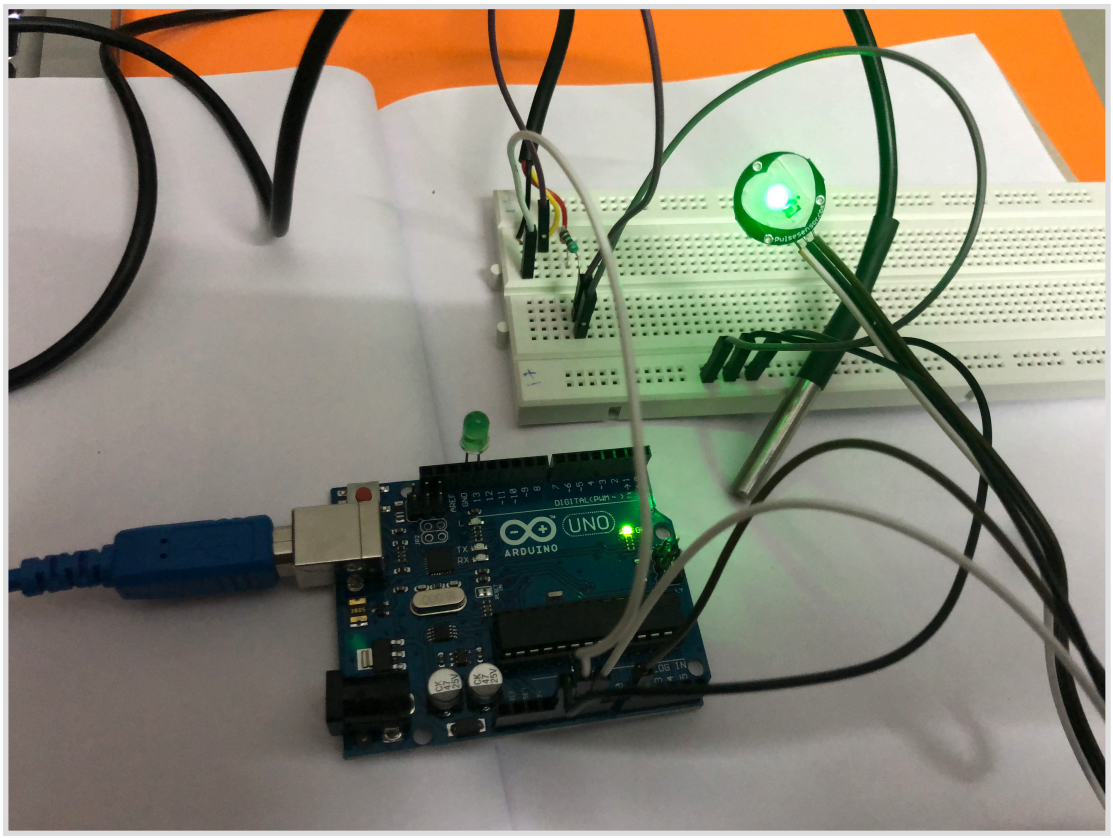


FIGURE 4: HEARTBEAT AND TEMPERATURE SENSOR WITH ARDUINO WORKING MODULE

4 Implementation:

The system is programmed such that they detect the temperature and heartbeat of a patient. This system is achieved by using sensors- Temperature sensor DS18B20 and heartbeat sensor working on Arduino Uno R3.

4.1 Pseudo Code:

Pseudo Code A:

1. Initialise digital pin LED_BUILTIN as an output.
2. Turn the LED on, at HIGH (voltage level)
3. Wait for 50ms
4. Turn the e LED off, at LOW(voltage level)
5. Wait for 500 ms
6. Run the loop function over and over again for flickering of LED light.

CODE:

```
#define LED 13
void setup()
{
  pinMode(LED, OUTPUT);
}
void loop()
{
  digitalWrite(LED, HIGH);
  delay(50);
  digitalWrite(LED , LOW);
  delay(500);
}
```

Pseudo Code B:

1. Import libraries- OneWire and DallasTemperature
2. Connect the sensor to the pin number 2
3. An object of TYPE OneWire is made, to communicate with the OneWire device
4. Reference of OneWire instance is passed to DallasTemperature
5. Serial port, over which the output is displayed
6. Start the sensor DS18B20
7. Globally request to seek temperatures from all the devices connected
8. Display the results

CODE:

```
#include <OneWire.h>
#include <DallasTemperature.h>
#define sensor_con 2

   OneWire x(sensor_con);
   DallasTemperature sensor(&x);
void setup()
{
  Serial.begin(9600);
  Serial.println("Temperature Demo ");
  sensor.begin();
 }
void loop()
{
 Serial.print("Requesting Temperature:");
 sensor.requestTemperatures();
 Serial.println("Input successful");
 Serial.print("Temperature Readings are:");
 Serial.println(sensor.getTempFByIndex(0));
 delay(1000);
}
```

Pseudo Code C:

1. All the Variables are initialised.
2. To achieve a reliable measurement of the timing between each beat, we set the Timer2, which throws an interrupt after every 2 milliseconds.
3. PWM output is disabled on pin 3 and pin 11.
4. Interrupt setup function is initialised which indicates Timer2 approaching clear time on compare mode (CTC).
5. Sei() function is called to make sure that the global interrupts are enabled.
6. The maximum count is set to 124, exceeding which the flag is incremented after which Interrupt Service routine function is called.
7. First input of signal is taken from the pulse pin.

8. A sample counter is incremented by 2.
9. A variable N is defined by sample_product – last_beat_time, to avoid Noise.
10. BPM is derived by the average of previously recorded 10 Inter Beat Interval's(IBI).
11. The Arduino is powered up, it reads the sensor value and looks for the heartbeat.
12. 3/5 of IBI is set so as to avoid noise.
13. Finding the highest and the lowest values of the pulse wave.
14. Now, checking if we have a pulse input and then finding realistic BPM values. This is achieved by discarding the first beat, since it's lousy and accepting the beats after it.
15. The falling pulse is monitored and the pulse is set to false so that the next beat can be detected.
16. If no beat is found after 2.5 seconds, variables used are reinitialised to initial values.

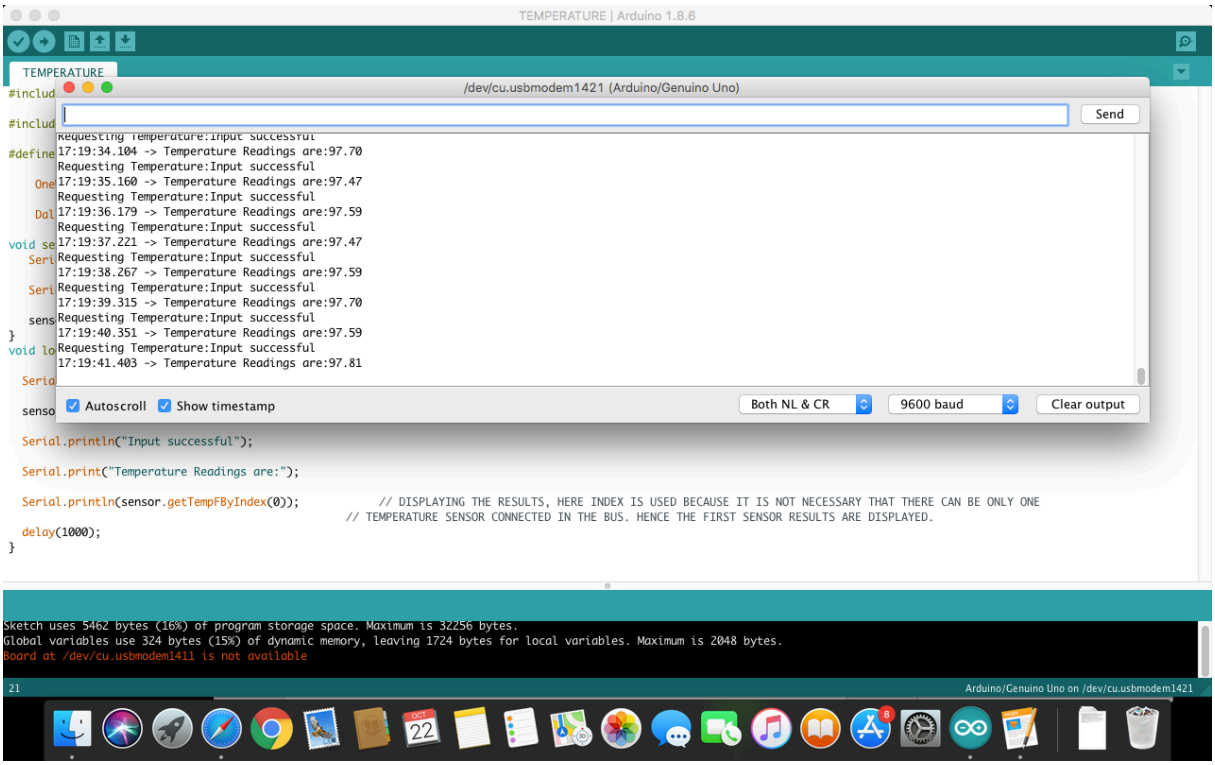
CODE:

```
int pulsePin = 0;
int blinkPin = 13;
volatile int BPM;
volatile int Signal;
volatile int IBI = 600;
volatile boolean Pulse = false;
volatile int rate[10];
volatile int P = 512;
volatile int T = 512;
volatile int thresh = 512;
volatile int amp = 100;
volatile boolean firstBeat = true;
volatile boolean secondBeat = false;
volatile unsigned long samplecounter = 0;
volatile unsigned long lastBeatTime = 0;
void setup()
{
 pinMode(blinkPin,OUTPUT);
 Serial.begin(115200);
 interruptSetup();
}

void loop()
{
    Serial.print("BPM: ");
    Serial.println(BPM);
    delay(200);
}

void interruptSetup()
{
 TCCR2A = 0x02;
 OCR2A = 0X7C;
 TCCR2B = 0x06;
 TIMSK2 = 0x02;
 sei();
}

ISR(TIMER2_COMPA_vect)
{
  cli();
  Signal = analogRead(pulsePin);
  samplecounter += 2;
  int N = samplecounter - lastBeatTime;
  if(Signal < thresh && N > (IBI/5)*3)
  {
     if (Signal < T)
     {
```

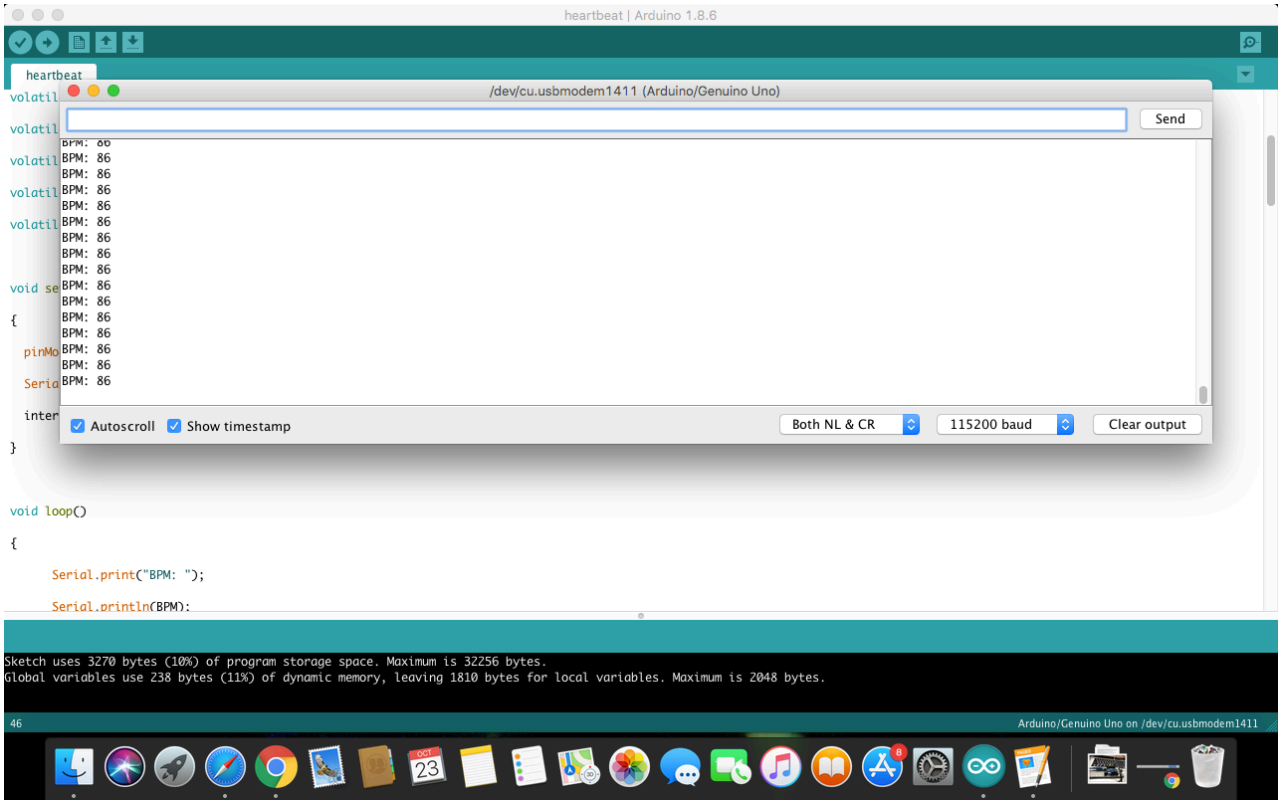
```
        T = Signal;
      }
    }
if(Signal > thresh && Signal > P)
   {
    P = Signal;
   }
 if (N > 250)
 {
  if ( (Signal > thresh) && (Pulse == false) && (N > (IBI/5)*3) )
     {
      Pulse = true;
      digitalWrite(blinkPin,HIGH);
      IBI = samplecounter - lastBeatTime;
      lastBeatTime = samplecounter;
     if(secondBeat)
      {
       secondBeat = false;
       for(int i=0; i<=9; i++)
       {
         rate[i] = IBI; //Filling the array with the heart rate values
       }
      }
      if(firstBeat)
      {
       firstBeat = false;
       secondBeat = true;
       sei();
       return;
      }
    word runningTotal = 0;
    for(int i=0; i<=8; i++)
      {
       rate[i] = rate[i+1];
       runningTotal += rate[i];
      }
    rate[9] = IBI;
    runningTotal += rate[9];
    runningTotal /= 10;
    BPM = 60000/runningTotal;
  }
 }
 if (Signal < thresh && Pulse == true)
  {
    digitalWrite(blinkPin,LOW);
    Pulse = false;
    amp = P - T;
    thresh = amp/2 + T;
    P = thresh;
    T = thresh;
```

```
    }
  if (N > 2500)
    {
      thresh = 512;
      P = 512;
      T = 512;
      lastBeatTime = samplecounter;
      firstBeat = true;
      secondBeat = false;
    }
 sei();
}
```

4.2 Output Screen:



OUTPUT:1


heartbeat | Arduino 1.8.6
heartbeat
/dev/cu.usbmodem1411 (Arduino/Genuino Uno)
Send
BPM: 86
Autoscroll
Show timestamp
Both NL & CR
115200 baud
Clear output
void loop()
{
Serial.print("BPM: ");
Sketch uses 3270 bytes (10%) of program storage space. Maximum is 32256 bytes.
Global variables use 238 bytes (11%) of dynamic memory, leaving 1810 bytes for local variables. Maximum is 2048 bytes.
Arduino/Genuino Uno on /dev/cu.usbmodem1411

OUTPUT:2

References:

1. Authors: Jigar Chauhan and Sachin Bojewar, “Sensor networks based healthcare monitoring system” in International Conference on Inventive Computation Technologies(ICICT), 2016

DOI:10.1109/INVENTIVE.2016.7824814

2. Authors: Aminian and Naji, “Journal of Health & Medical Informatics” in Health Med Inform, 2013

DOI: 10.4172/2157-7420.1000121

3. Authors: Amna Abdullah, Asma Ismael, Aisha Rashid, Ali Abou-ElNour, and Mohammed Tarique, “REAL TIME WIRELESS HEALTH MONITORING APPLICATION USING MOBILE DEVICES” International Journal of Computer Networks Communications (IJCNC) Vol.7, No.3, May 2015

DOI:10.1109/INVENTIVE.2016.7824814

4. Authors: Ahmed Abdulkadir Ibrahim, Yasin Muhammad, Wang Zhuopeng, “IOT Patient Health Monitoring System” in International Journal of Engineering Research and Applications (IJERA) ,vol. 7

DOI: 10.9790/9622-0710070103

A APPENDIX I PROJECT CODE:

```
#include <OneWire.h>
#include <DallasTemperature.h>
#define LED 13
#define sensor_con 2

   OneWire x(sensor_con);
   DallasTemperature sensor(&x);

int pulsePin = 0;
int blinkPin = 13;
volatile int BPM;
volatile int Signal;
volatile int IBI = 600;
volatile boolean Pulse = false;
volatile int rate[10];
volatile int P = 512;
volatile int T = 512;
volatile int thresh = 512;
volatile int amp = 100;
volatile boolean firstBeat = true;
volatile boolean secondBeat = false;
volatile unsigned long samplecounter = 0;
volatile unsigned long lastBeatTime = 0;

void setup()
{
 pinMode(blinkPin,OUTPUT);
  Serial.begin(9600);
  Serial.println("Temperature Demo ");
  sensor.begin();
  pinMode(LED, OUTPUT);
  Serial.begin(115200);
  interruptSetup();
}

void loop()
{

 digitalWrite(LED, HIGH);
 delay(50);
 digitalWrite(LED , LOW);
 delay(500);
 Serial.print("Requesting Temperature:");
 sensor.requestTemperatures();
 Serial.println("Input successful");
 Serial.print("Temperature Readings are:");
 Serial.println(sensor.getTempFByIndex(0));

   Serial.print("BPM: “);
   Serial.println(BPM);
   delay(200); //  take a break
}

void interruptSetup()
{

 TCCR2A = 0x02;
 OCR2A = 0X7C;
 TCCR2B = 0x06;
 TIMSK2 = 0x02;
 sei();
}

ISR(TIMER2_COMPA_vect)
{
 cli();
 Signal = analogRead(pulsePin);
 samplecounter += 2;
 int N = samplecounter - lastBeatTime;
```

```
 if(Signal < thresh && N > (IBI/5)*3)
   {
```



```
     if (Signal < T)
     {
       T = Signal;
     }
   }
 if(Signal > thresh && Signal > P)
   {
     P = Signal;
   }

  if (N > 250)
 {
   if ( (Signal > thresh) && (Pulse == false) && (N > (IBI/5)*3) )
      {
        Pulse = true;
        digitalWrite(blinkPin,HIGH);
        IBI = samplecounter - lastBeatTime;
        lastBeatTime = samplecounter;
       if(secondBeat)
        {
          secondBeat = false;
          for(int i=0; i<=9; i++)
          {
            rate[i] = IBI; //Filling the array with the heart rate values
          }
        }

     if(firstBeat)
      {
        firstBeat = false;
        secondBeat = true;
        sei();
        return;
      }

     word runningTotal = 0;
     for(int i=0; i<=8; i++)
      {
        rate[i] = rate[i+1];
        runningTotal += rate[i];
      }

      rate[9] = IBI;
      runningTotal += rate[9];
      runningTotal /= 10;
      BPM = 60000/runningTotal;
     }
}

  if (Signal < thresh && Pulse == true)
    {
     digitalWrite(blinkPin,LOW);
     Pulse = false;
     amp = P - T;
     thresh = amp/2 + T;
     P = thresh;
     T = thresh;
    }
 if (N > 2500)
    {
     thresh = 512;
     P = 512;
     T = 512;
     lastBeatTime = samplecounter;
     firstBeat = true;
     secondBeat = false;
    }
 sei();
}
```