PATIENT PULSE RATE AND TEMPERATURE MONITORING IOT BASED SYSTEM

A
Project Report

submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY in

COMPUTER SCIENCE Specialisation in

Business Analytics and Optimisation

By :

Name	Roll No	Branch
Abhishek Sethi	R103216004	CSE BAO
Chirag Gupta	R103216028	CSE BAO
Isha Lalchandani	R103216043	CSE BAO

Under the guidance of

Dr. Rajeev Tiwari
Associate Professor, SCS, UPES

School of Computer Science
UNIVERSITY OF PETROLEUM AND ENERGY STUDIES
Dehradun-248007

Approved By

Project Guide Department head

CANDIDATES DECLARATION

I/We hereby certify that the project work entitled Patient Pulse Rate and Temperature Monitoring IOT Based System in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science And Engineering with Specialisation in Business Analytics and Optimisation and submitted to the Department of Analytics, at School of Computer Science, University of Petroleum And Energy Studies, Dehradun, is an authentic record of our work carried out during a period from August, 2018 to December, 2018 under the supervision of **, Dr. Rajeev Tiwari, Associate Professor and Department of Virtualisation, School of Computer Science.**

The matter presented in this project has not been submitted by me/ us for the award of any other degree of this or any other University.

Name:	**Abhishek Sethi**	**Chirag Gupta**	**Isha Lalchandani**
Roll No:	**R103216004**	**R103216028**	**R103216043**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

(Date: 26 Nov 2018) **(Dr. Rajeev Tiwari)**

Project Guide

Dr. T.P. Singh

Head
Department of Analytics
School of Computer Science
University of Petroleum And Energy Studies
Dehradun - 248001 (Uttarakhand)

ACKNOWLEDGEMENT

We wish to express our deep gratitude to our guide **Dr. Rajeev Tiwari**, for all advice, encouragement and constant support he has given us through out our project work. This work would not have been possible without his support and valuable suggestions.

We sincerely thank to our Head of the Department, **Dr. T. P. Singh**, for his great support in do-ing our **Patient Pulse Rate and Temperature Monitoring IOT Based System** at **SoCS**.

We would like to thank all our **friends** for their help and constructive criticism during our project work. Finally we have no words to express our sincere gratitude to our parents who have shown us this world and for every support they have given us.

Name	**Abhishek Sethi**	**Chirag Gupta**	**Isha Lalchandani**
Roll No.	**R103216004**	**R103216028**	**R103216043**

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

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.

Literature Review:

We've gone through the various research work of health monitoring system, they have used Zigbee technology, Raspberrypi, IOT and various other technologies. For the proposed system keywords are: pulse rate sensing unit, temperature sensing unit, WI-FI module, ThingSpeak, Embedded C, Remote Patient Monitoring (RPM). In this project, basic parameters like body temperature & heart beat is monitored and is transferred on ThingSpeak to make it locally accessible for users. The system is design to read the body temperature and heartbeat of patient at run time. The system mainly focuses on collecting the physical parameter and thereby making that information available on the server. One can refer to this data to determine the health condition of patients. In case of any abnormality, the system keeps a track of the health recorded.[1]To realize distributed body temperature monitoring system is designed using temperature sensor. Body temperature data is collected with the help of DS18B20 temperature sensor. When the heart beat detector is working at that time the LED flashes in unison with each Heart beat and shows the status of device. It works on the principle of light modulation by blood flow through finger, at each pulse. This digital output is connected to Arduino. The detected values should be available for every doctor who is appointed for that patient, for this the detected values should be made local by uploading them on a centralised repository, that stores data on ThingSpeak which visualizes the uploaded data. This API shows detected values at runtime. It will contain the basic information of the patient and the determined values of body temperature and pulse rate.[3] "Raspberry Pi Based Patient Monitoring System uses Wireless Sensor Nodes", by Mendrela Biswas, presented at International Research Journal of Engineering and Technology (IRJET)

in April-2016.In this project, the monitoring of the patient is done by the doctor continuously without actually visiting the patient. Here, we are using various sensors to sense the physiological parameters like temperature, blood pressure, ECG and the level of saline. These sensed signals are transmitted to the Raspberry pi to update the data continuously via ADC which will convert these analog signals into digital signals. Through RF transmitter, the data is sent wirelessly to the monitor screen of the doctor. So, the doctor can visualize the patient's data, when a critical condition occurs, the visual indications will be sent onto the screen.[4] "IOT Patient Health Monitoring System", by Ahmed Abdulkadir Ibrahim, Yasin Muhammad, Wang Zhuopeng presented at International Journal of Engineering Research and Applications (IJERA), vol. 7. This system is designed to be used in hospitals for measuring and monitoring various parameters like temperature, ECG, heartbeat etc. The results can be recorded using Arduino displayed on a LCD display. Also the results can be sent to server using WI-FI module over ThingSpeak. Doctors can login to a ThingSpeak channel and view those results. In our system we are measuring patient's parameters (temperature, heart rate,) with the help of different sensors available.

Problem Statement:

This Project will mainly be the bridge between specialists and people living in remote areas, through this one can easily monitor heartbeat and body temperature without paying a visit to a clinic or a hospital, hence decreasing the possibilities of any abnormalities.

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.

Setup of ThingSpeak server

Analysis and visualisation.

Design Methodology:

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.

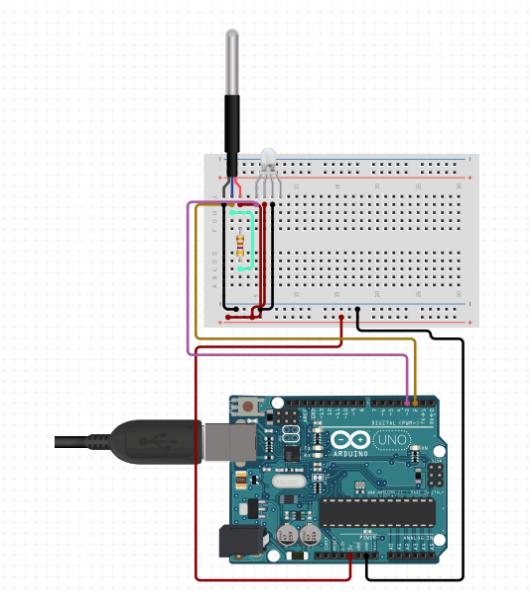


FIGURE 1:TEMPERATURE SENSOR MODULE

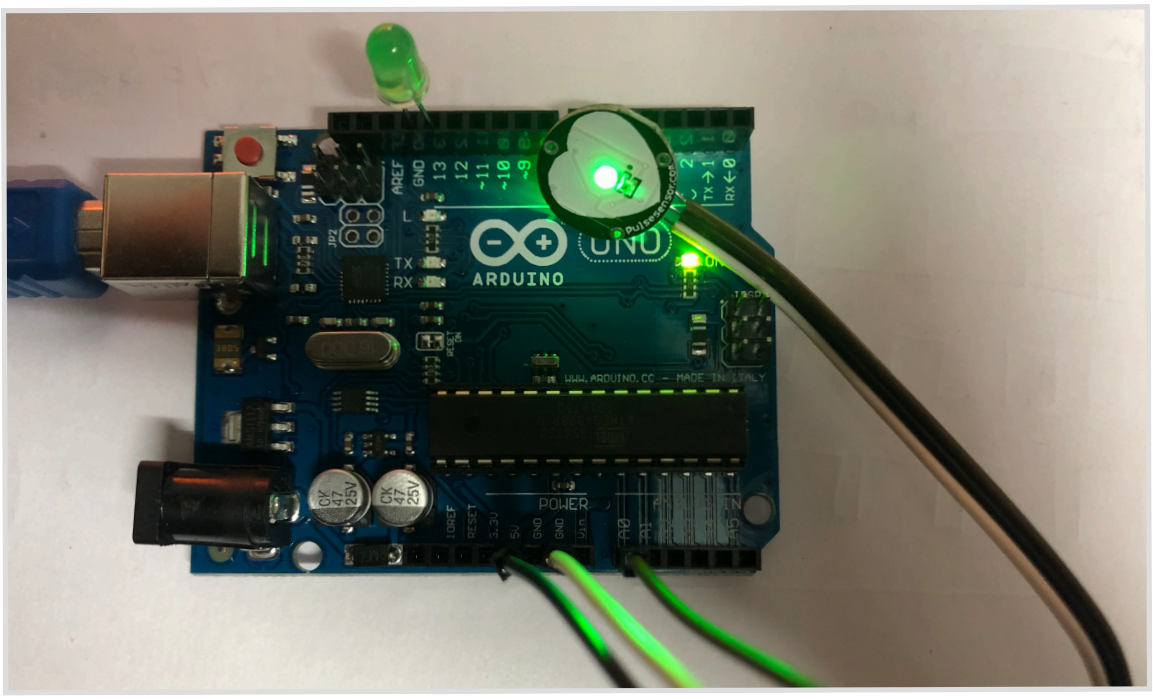


FIGURE2:TEMPERATURE SENSOR WITH ARDUINO WORKING MODULE

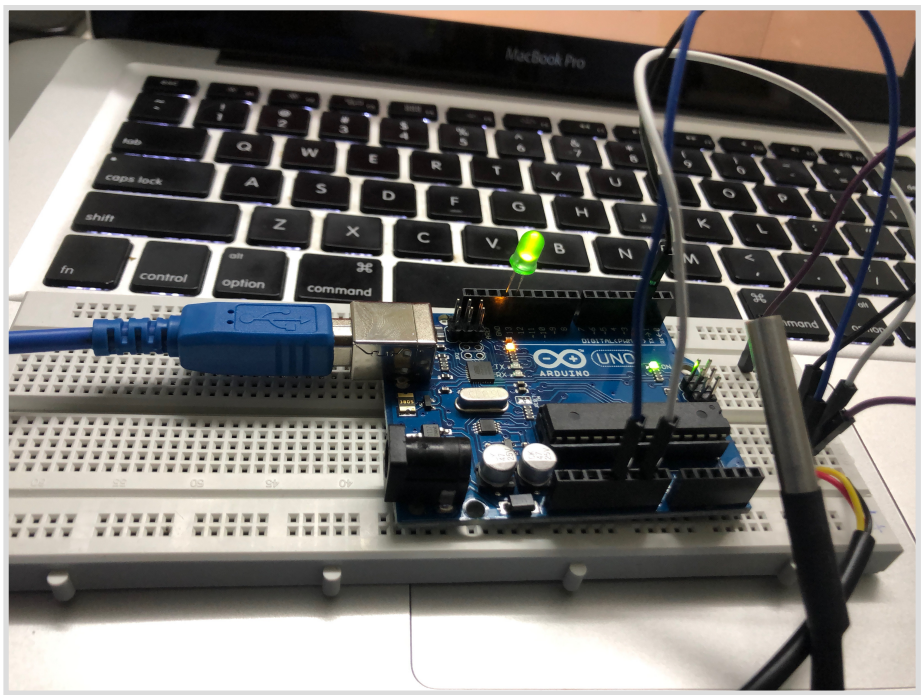


FIGURE3:HEARTBEAT SENSOR WITH ARDUINO WORKING MODULE

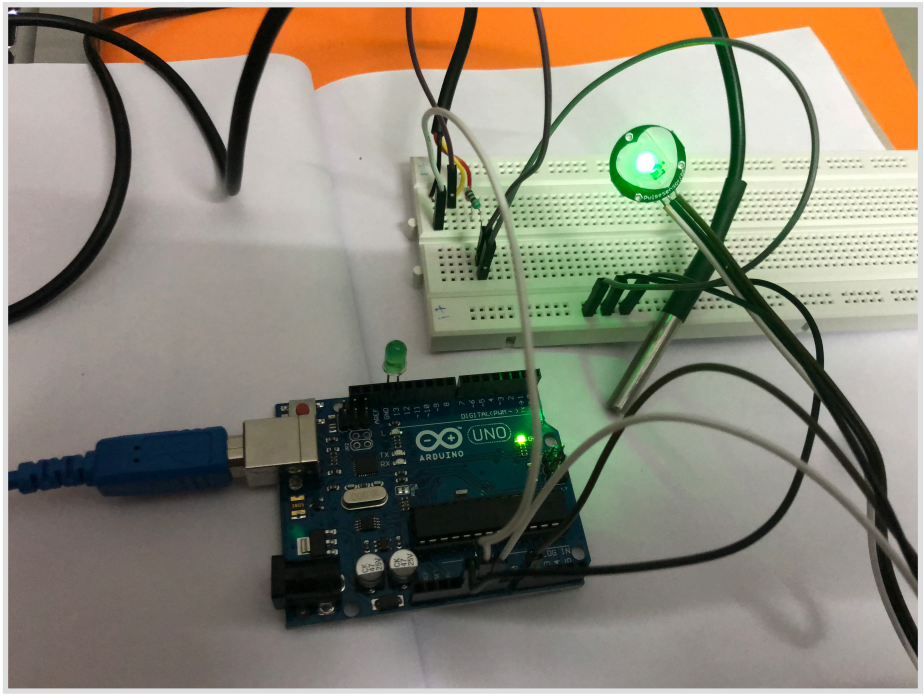


FIGURE 4: HEARTBEAT AND TEMPERATURE SENSOR WITH ARDUINO WORKING MODULE

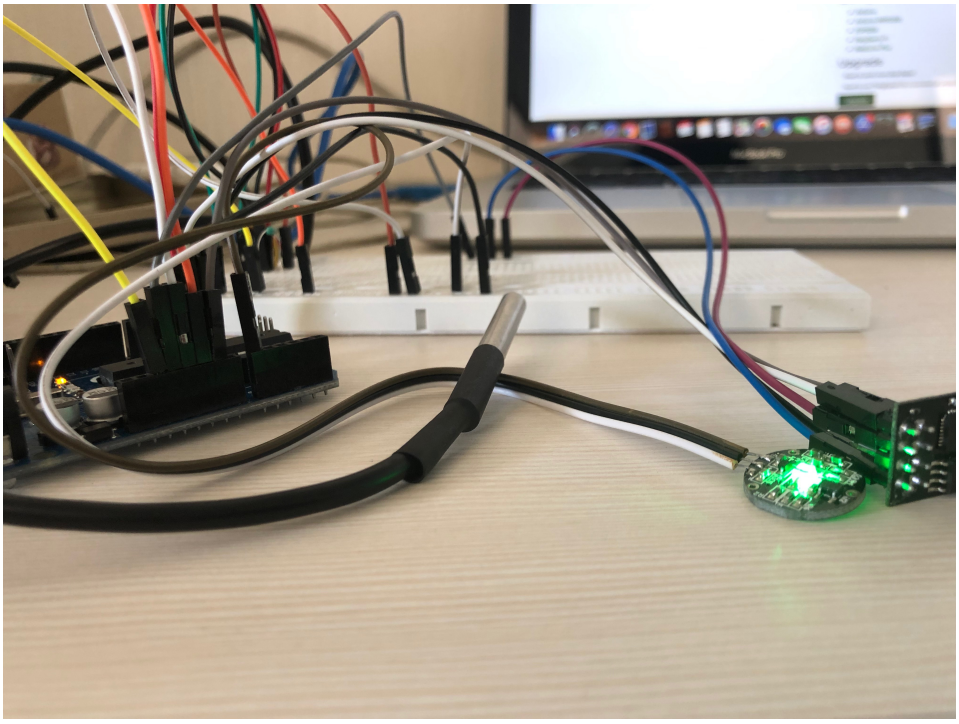


FIGURE 5:Working Of Wi-Fi Module

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)
     {
```

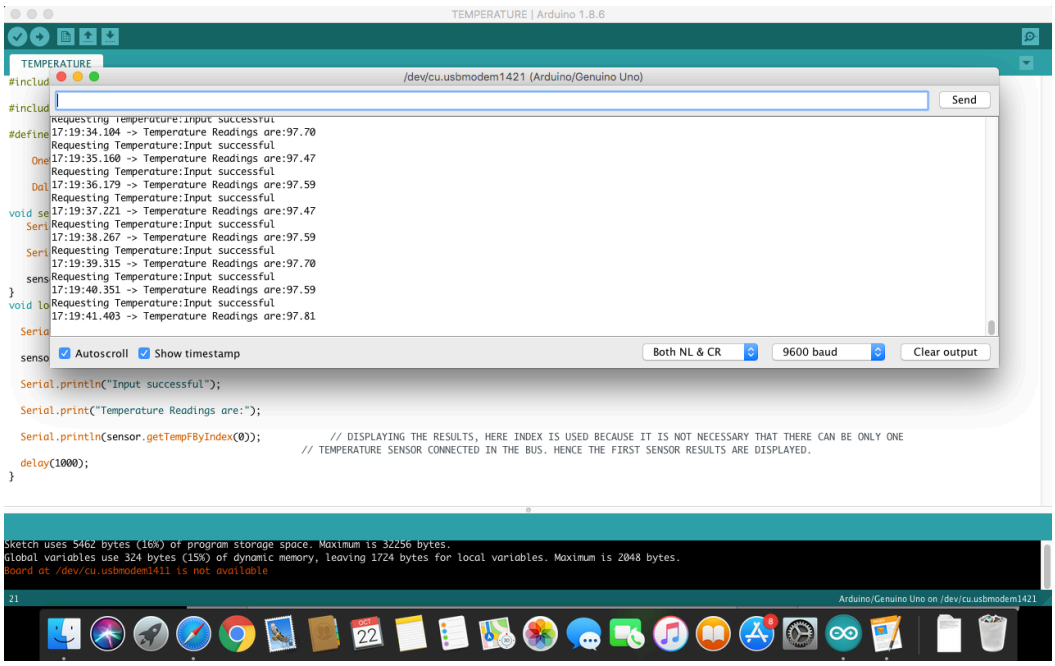
```
        T = Signal;
      }
    }
if(Signal > thresh && Signal > P)
   {
     P = Signal;
   }
 if (N > 250)
 {
  if ( (Signal > thresh) && (Pulse == false) && (N > (IBI/5)*3) )
     {
      Pulse = true;
      digitalWrite(blinkPin,HIGH);
      IBI = samplecounter - lastBeatTime;
      lastBeatTime = 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();
}
```

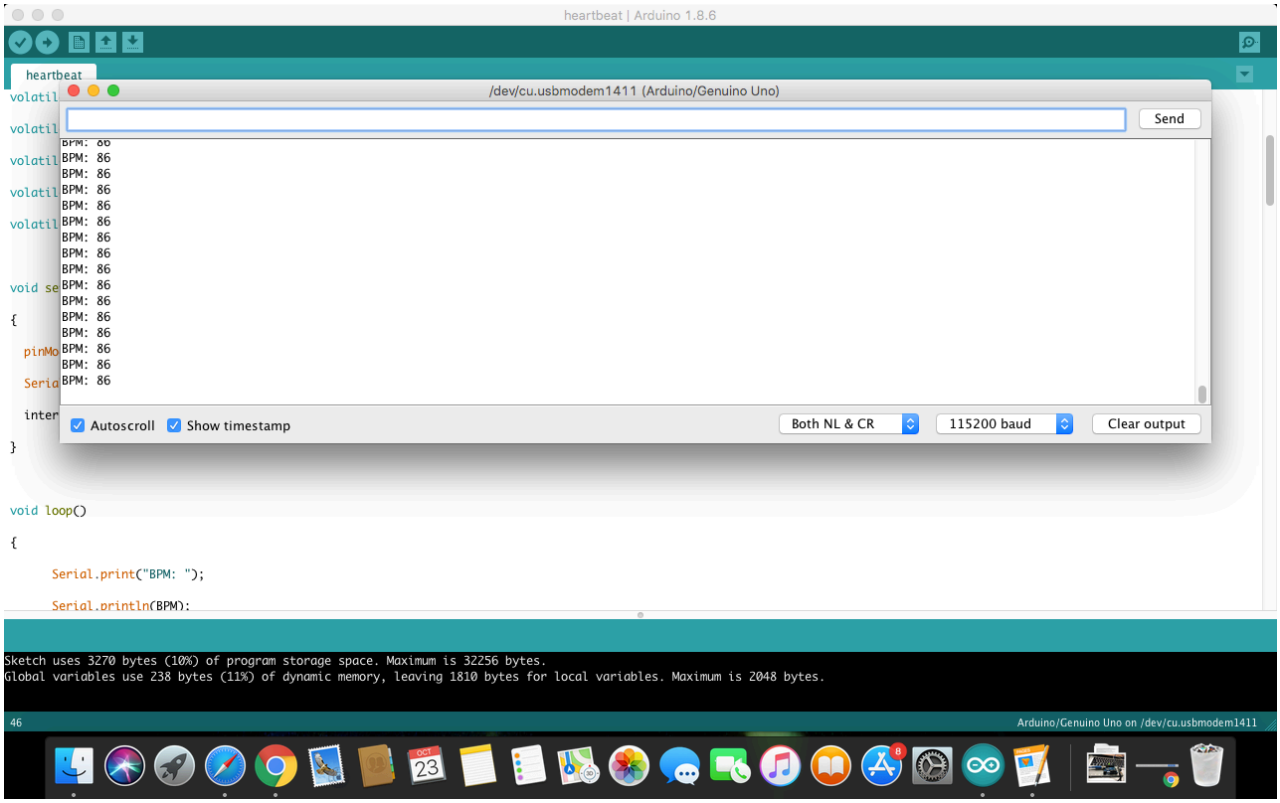
Pseudo Code D:

1. Include the libraries <ESP8266webserver.h> , <ESP8266wifi.h> ,<wificlient.h> in sketch.
2. Input the desired credentials of the wifi i.e. wifi ssid and password.
3. Enter the web address to read from the host that is api.thingspeak.com
4. Take the input of the API key.
5. Enable both the hotspot and client.
6. Connect to wifi router.
7. Wait for the connection.
8. If connection is successful show the IP address in serial monitor and run the AT command, if reply is OK, then the connection is successful. ELSE keep trying until we are connected to the network.
9. Make GET request as per HTTP GET protocol format, wait for server to respond with timeout of 5 seconds.
10. If data is available before timeout, then read it and print the desired responses from Thingspeak, ELSE print request timeout.

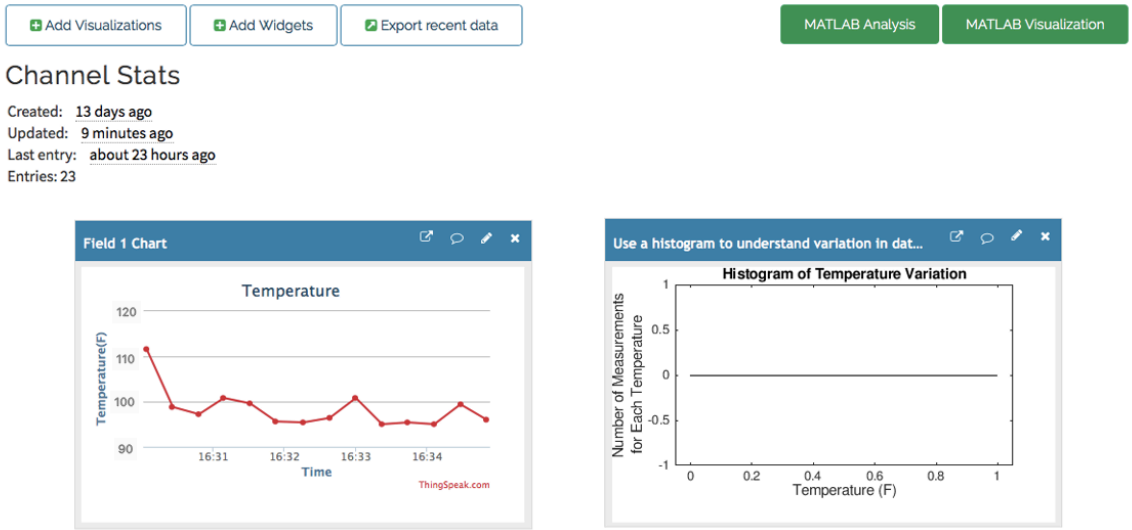
4.2 Output Screen:



OUTPUT:1



OUTPUT:2



OUTPUT:3

4.3 Result Analysis

The project is designed to measure the temperature and heartbeat of patients without having to visit the Doctor personally. The WifiID and Password are taken as input and WIfiModule connects over this network to the ThingSpeak Channel. The results are displayed in the form of graphs which are formulated through coding on the channel. The graphs show the temperatures recorded for a patient at different times and a variation between their temperatures recorded. It also shows the pulse rate of a Patient recorded at different times and graph plotted fro the same. The result displayed is individual for every Patient and makes remote health monitoring easy.

Conclusion and Future Scope

Future Scope of this project includes the use of Machine learning Algorithms using MATLAB to help medical organizations improve customer service, extract value from large data amounts, efficiently analyze medical records, and enhance patient treatment. The project can be further be improvised using Raspberry Pi and adding biosensors that can enable users to measure glucose levels, arterial pressure, heart rate, oxygen level, blood alcohol level, and alert users and doctors if some health problems are detected. With the completion of this project we can accurately calculate the heartbeat and temperature of a patient with the help of the required sensors and modules. We can also visualise the outputs of heartbeat and temperature graphically with the help of Thingspeak.

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

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

DOI: 10.4172/2157-7420.1000121

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

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

APPENDIX I PROJECT CODE:

```
#include <ESP8266WebServer.h>

#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#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;


//#include <ESP8266WiFi.h>
//#include <WiFiClient.h>
//#include <ESP8266WebServer.h>

/* Set these to your desired credentials. */
const char *ssid = "OnePlus 3T";  //ENTER YOUR WIFI SETTINGS <<<<<<<<<
const char *password = "Parth@123";



//Web address to read from
const char *host = "api.thingspeak.com";
String apiKey = "2L56F80DVJYTPUYZ";  //ENTER YOUR API KEY <<<<<<<<<<<
//=====================================================================
//                Power on setup
//=====================================================================

void setup() {
  delay(1000);
  Serial.begin(115200);

  WiFi.mode(WIFI_STA);
```

```
  WiFi.begin(ssid, password);
  Serial.println("");

  Serial.print("Connecting");
  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  //If connection successful show IP address in serial monitor
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());  //IP address assigned to your ESP
  //----------------------------------
  pinMode(blinkPin,OUTPUT);
  //  Serial.begin(9600);
  //  Serial.println("Temperature Demo ");

   sensor.begin();
     pinMode(LED, OUTPUT);

  Serial.begin(115200);
  interruptSetup();
}

//=============================================================================
//                  Main Program Loop
//=============================================================================
void loop() {
  WiFiClient client;
  const int httpPort = 80; //Port 80 is commonly used for www
 //----------------------------------------------------------------------
 //Connect to host, host(web site) is define at top
 if(!client.connect(host, httpPort)){
   Serial.println("Connection Failed");
   delay(300);
   return; //Keep retrying until we get connected
 }

//----------------------------------------------------------------------
  //Make GET request as pet HTTP GET Protocol format
  String ADCData;
  int adcvalue=analogRead(A0);  //Read Analog value of LDR
  ADCData = String(adcvalue);   //String to interger conversion
  String Link="GET /update?api_key="+apiKey+"&field1=";  //Requeste webpage
  Link = Link + ADCData;
  Link = Link + " HTTP/1.1\r\n" + "Host: " + host + "\r\n" + "Connection: close\r\n\r\n";


  client.print(Link);
  delay(100);

//----------------------------------------------------------------------
 //Wait for server to respond with timeout of 5 Seconds
 int timeout=0;
 while((!client.available()) && (timeout < 1000))     //Wait 5 seconds for data
 {
   delay(10);  //Use this with time out
   timeout++;
 }

//----------------------------------------------------------------------
 //If data is available before time out read it.
```

```
 if(timeout < 500)
 {
    while(client.available()){
      Serial.println(client.readString()); //Response from ThingSpeak
    }
 }
 else
 {
    Serial.println("Request timeout..");
 }

 delay(5000);  //Read Web Page every 5 seconds
 //------------------------------
 digitalWrite(LED, HIGH);

  delay(50);                                          //DELAY OF 50ms

  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);
}



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();
}
```