Patient Health Monitoring System

Project Report
FALL SEM 2022-23

ECS
(Engineering Clinics)

Submitted by

U Vamsi Krishna (*20BCE7233)*
G Dileep Chandu (*20BCI7313)*
B Bhargav (*20BCD7273)*

Under the Guidance of

Prof. M.Sucharitha

VIT-AP UNIVERSITY

Inavolu, Beside AP Secretariat

Amaravati ,Andhra Pradesh 522237

ABSTRACT:

Patients are facing a problematic situation of unforeseen demise due to the specific reason of heart problems and attacks which is because of the nonexistence of good medical maintenance to patients at the needed time. And also there are so many people who are suffering from different types of health issues mainly heart diseases, due to their busy daily life they cannot even go to the hospital for a check-up of their health condition and the people who are physically ILL or bed-ridden and some people can’t be taken to hospital. For those people who have problems like these, this multifunctional medical device is very useful by this device we can do our health checks up’s in home itself.

So we are proposing an innovative project to dodge such sudden death rates by using Patient Health Monitoring that uses sensor technology. This system uses a Temperature module, Heartbeat sensor, Electrocardiogram sensor, and Pulse oximeters sensor for tracking the patient’s health. This sensor are connected to the Arduino-Uno board. To track the patient’s health microcontroller is in turn interfaced to an LCD display and wi-fi connection to send the data to the web server (wireless sensing node). Thus Patient health monitoring system helps to monitor patient health.

Our project incorporates sensors to measure parameters like body temperature, heart beat rate, SpO2, and ECG. An Arduino Uno board is used for analyzing the inputs from the patient and gives the reading as an output on the LCD screen. This is very useful for future analysis and review of a patient’s health condition.

For more versatile medical applications, this project can be improvised, by incorporating blood pressure monitoring systems, dental sensors, and annunciation systems, thereby making it useful in hospitals as a very efficient and dedicated patient care system.

Table of Content:

1 INTRODUCTION:

Heart diseases are becoming a big issue for the last few decades and many people die because of certain health problems. Therefore, heart disease cannot be taken lightly. By analyzing or monitoring the ECG signal, SpO2, BPM, and Temp at the initial stage this disease can be prevented. So we present this project, i.e. ECG Monitoring with AD8232 ECG Sensor & Arduino with ECG Graph and BPM & SpO2 with Pulse Oximeter sensor and BPM with temp with Pulse sensor and LM35 Temperature Sensor.

This project is significant in various ways because, in today's world, everyday many lives are affected because the patients are not timely and properly operated, and also real-time parameter values are not efficiently measured in the clinic as well as in hospitals. Sometimes it becomes difficult for hospitals to frequently check patients' health conditions. Also, continuous monitoring of ICU patients is very difficult. To deal with these types of situations, our system is beneficial. Our system is designed to be used in hospitals and homes and also for measuring and monitoring various parameters like temperature, ECG, heart rate, and blood pressure. The results can be recorded using an Arduino Uno board. The system will also generate an alert notification which will be sent to the doctor. Our system is useful for monitoring the health system of every person by easily attaching the device and recording it.

The AD8232 is a neat little chip used to measure the electrical activity of the heart. This electrical activity can be charted as an ECG or Electrocardiogram. Electrocardiography is used to help diagnose various heart conditions. So in this project, we will interface AD8232 ECG Sensor with Arduino and observe the ECG signal on a serial plotter or Processing IDE and by the Pulse Oximeter sensor we monitoring the BPM and SpO2 of the patients.

And also in this project, we are going to make IoT BPM Monitoring on ThingSpeak using Pulse Sensor, ESP8266 & Arduino. The device will detect the pulse rate using the Pulse Sensor and will show the readings in BPM (Beats Per Minute) on the LCD display. It will send the readings to the ThingSpeak server via the Wi-Fi module ESP8266. This will help us to monitor the heartbeat via the Internet from any part of the world. ThingSpeak is an open-source Internet of Things (IoT) application and API to store and retrieve data from things using the HTTP protocol over the Internet or via a Local Area Network.

The main uses are that people in the military can find an easier way to frequent check up of their health , when they go to cold ad hilly areas, and to check up on their heart condition during the illness or sickness. And for covid patients too it will helpful a lot.

2 BACKGROUND:

COVID-19 is a contagious complaint that affects the case's lungs vigorously which results in the reduction of oxygen situations in the blood. An unforeseen drop in oxygen position in the blood will lead to Hypoxemia. So frequent monitoring of the case's Saturation of Peripheral Oxygen (SPO2)[1] and heart rate are needed. By recording and monitoring these parameters immediate treatment can be handed to the case in case of emergency.

An IoT-based health monitoring system for Temperature and BPM[2]. Patient Health Monitoring is beyond the abstruse explanations in order to improve healthcare accouterment and address that accompaniment absolute casework by assembling the abeyant of IoT.

3 PROBLEM DEFINITION :

In today's Social Health Insurance structure where patients stay at home after Operations, they are monitored by a medical caretaker or a family member. Nowadays many people who work full time are facing the problem of monitoring their loved ones, especially old age patients. So to overcome this problem we are using this patient health monitoring system using IoT. This uses sensor technology with micro-controller and wifi module to help the user monitor their loved ones.

One of the increasing popular public concerns is human health. Anything else becomes meaningless if one gets sick or dead. For this reason, people spend a lot of money to keep sound health. Unfortunately, people always find that it is too late to receive serious medical care when things are non-invertible.

If early actions can be taken in time then lots of patients can be cured. However, access to many medical equipment is inconvenient and expensive. Heart rate and body temperature are the most vital ones among the most notable indexes of the human health, and they have the advantage of easy access. Moreover, unlike the X-ray, the measurement of heart rate and body temperature has no effect on human health itself.

There are some devices in the current market which can provide raw medical measurement data to patients and doctors, but the patients may not interpret the medical measurement into meaningful diagnosis since they have little medical background. On the other hand, if raw medical data is delivered to the doctor, it kills much time and may cause trouble, but in emergencies time can never be wasted.

It is tough to share data over a large area within a short period. Most of the products available in the current market have these major drawbacks with limitation in flexibility and portability

4 OBJECTIVE:

The main objective of our project is to design a compact health monitoring system for Monitoring BPM, SpO2, Temperature, and ECG signals using Arduino Uno and the embedded c program. People in the military, covid patients, and survivors can find an easier way to frequent check-ups of their health when they are in different climatic conditions and hilly areas and to check up on their heart body temperature SpO2 condition during the illness or sickness.

Generally in hospitals to measure the temperature of a patient thermometer is used,
And to measure spo2 and bpm pulse oximeter is used and to take ECG (the particular patient will be taken to a room) there will be a use of an ECG Holter Monitor.

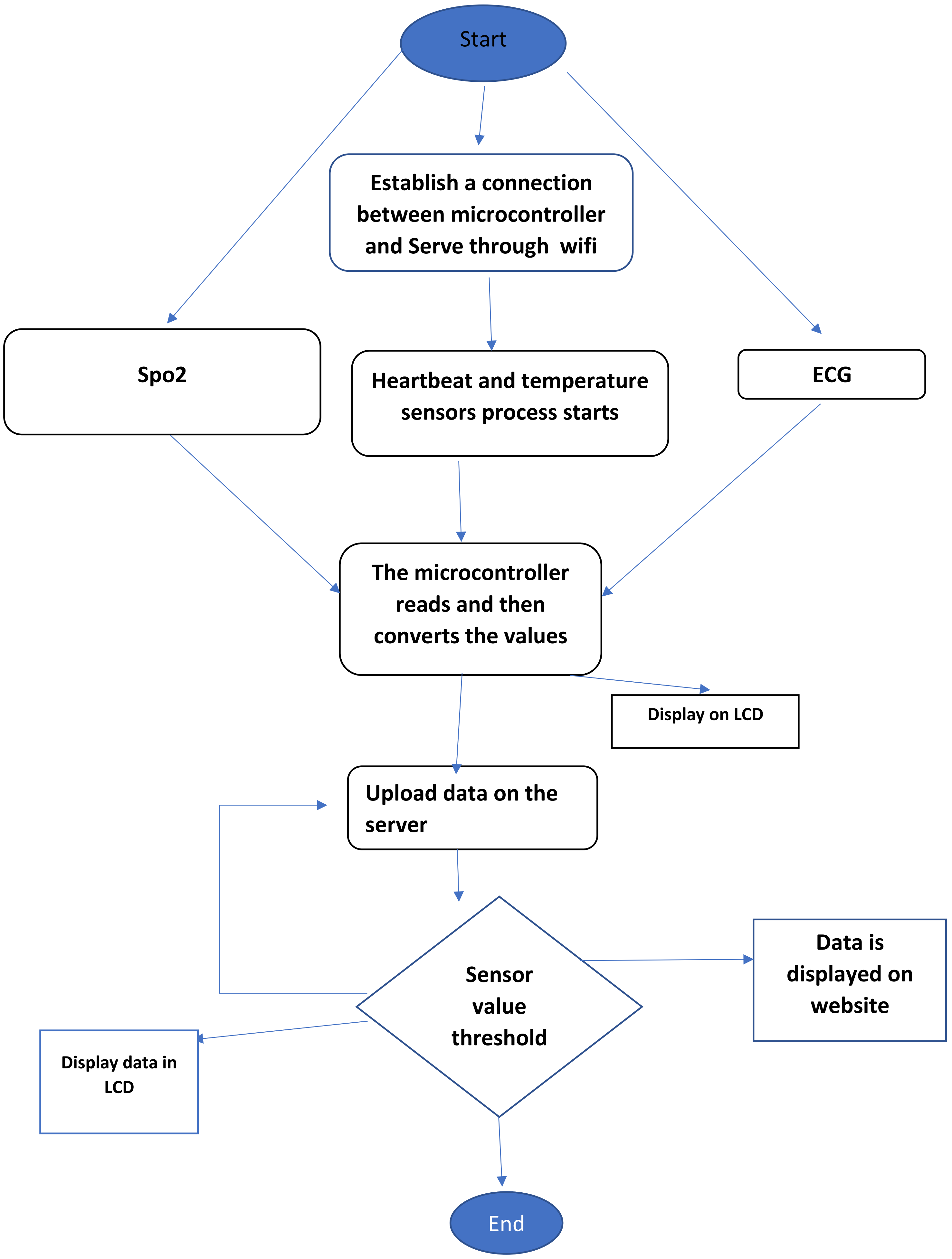
Through the implementation of this project, a compact form of the thermometer, pulse oximeter, and ECG Holter monitor is developed, which works as a multifunctional medical kit. Detection and Monitoring of BPM, SpO2, Temperature, and ECG signals using Arduino UNO and embedded c programming.

5 METHODOLOGY :

The IoT-based health surveillance method uses the concept of controlling the body temperature and heart rate of a patient. The unit is connected to the fingertips of the patient and the sensor is also attached to the body of the patient. This is a temperature sensor dependent on resistance whose resistance is measured by adjusting the body temperature of the patient as well as the heart rate sensor, pulse or flow-based sensor with an electrical signal transmitting value. Arduino-UNO, the primary or smart controller of this device, has two sensor calculation

5.1 Software Requirements:

Wire - Standard Library
Arduino-Liquid Crystal-I2C
MAX30100lib


Start
Establish a connection between microcontroller and Serve through wifi
Spo2
Heartbeat and temperature sensors process starts
ECG
The microcontroller reads and then converts the values
Display on LCD
Upload data on the server
Sensor value threshold
Data is displayed on website
Display data in LCD
End

5.2 System Requirements

5.2.1 Electrocardiogram(ECG)

An ECG is a paper or digital recording of the electrical signals in the heart. It is also called an electrocardiogram or an EKG. The ECG is used to determine heart rate, heart rhythm, and other information regarding the heart's condition. ECGs are used to help diagnose heart arrhythmias, heart attacks, pacemaker function, and heart failure.

ECG can be analyzed by studying the components of the waveform. These waveform components indicate cardiac electrical activity. The first upward of the ECG tracing is the P wave. It indicates atrial contraction.

The QRS complex begins with Q, a small downward deflection, followed by a larger upwards deflection, a peak (R); and then a downwards S wave. This QRS complex indicates ventricular depolarization and contraction.

Finally, the T wave, which is normally a smaller upwards waveform, representing ventricular re-polarization.

Medical uses of ECG:

An electrocardiogram can be a useful way to find out whether your high blood pressure has caused any damage to your heart or blood vessels. Because of this, you may be asked to have an ECG when you are first diagnosed with high blood pressure.

Some of the things an ECG reading can detect are: cholesterol clogging up your heart's blood supply

1. a heart attack in the past

2. enlargement of one side of the heart

3. abnormal heart rhythms

4 AD8232 ECG Sensor

This sensor is a cost-effective board used to measure the electrical activity of the heart. This electrical activity can be charted as an ECG or Electrocardiogram and output as an analog reading. ECGs can be extremely noisy, the AD8232 Single Lead Heart Rate Monitor acts as an op-amp to help obtain a clear signal from the PR and QT Intervals easily.

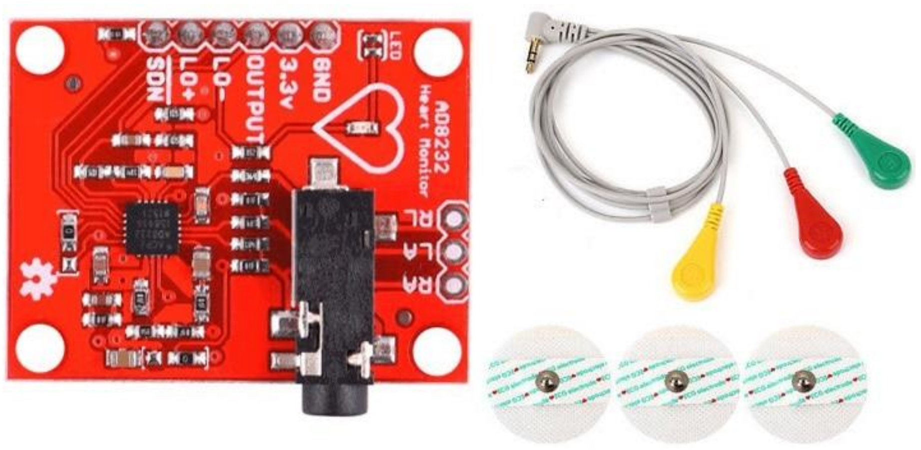


Fig 2:AD8232 Sensor

The AD8232 is an integrated signal conditioning block for ECG and other biopotential measurement applications. It is designed to extract, amplify, and filter small biopotential signals in the presence of noisy conditions, such as those created by motion or remote electrode placement.

The AD8232 module breaks out nine connections from the IC that you can solder pins, wires, or other connectors to. SDN, LO+, LO-, OUTPUT, 3.3V, GND provide essential pins for operating this monitor with an Arduino or other development board. Also provided on this board are RA (Right Arm), LA (Left Arm), and RL (Right Leg) pins to attach and use your own custom sensors. Additionally, there is an LED indicator light that will pulsate to the rhythm of a heartbeat.

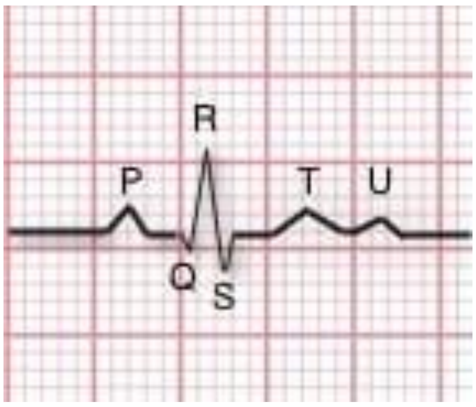


Fig 3: Graphical Reprenstation of Ecg curve

Note: *This product is NOT a medical device and is not intended to be used as such or as an accessory to such nor diagnose or treat any conditions.*

AD8232 ECG Sensor Placement on Body:

It is recommended to snap the sensor pads on the leads before application to the body. The closer to the heart the pads are, the better the measurement. The cables are color-coded to help identify proper placement.

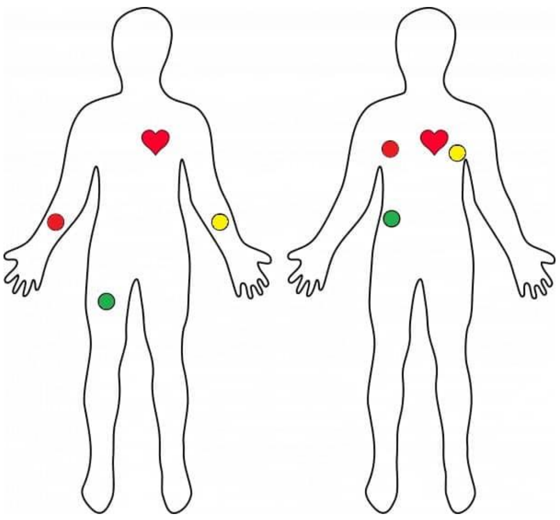


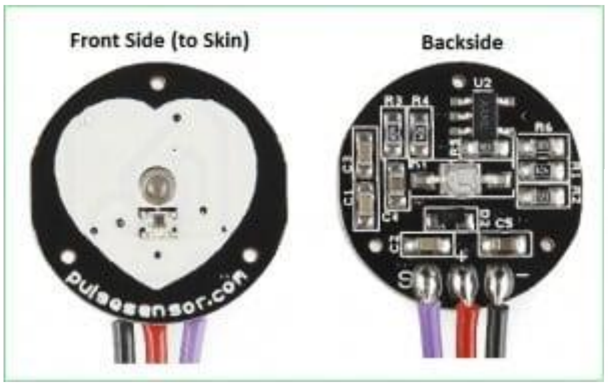
Fig :Places were to have ECG electrodes

Red: RA (Right Arm)

Yellow: LA (Left Arm)
Green: RL (Right Leg).

5.2.2 Pulse Sensor:

The Pulse Sensor is a plug-and-play heart-rate sensor for Arduino. It can be used by students, artists, athletes, makers, and game & mobile developers who want to easily incorporate live heart-rate data into their projects. The essence is an integrated optical amplifying circuit and noise-eliminating circuit sensor. Clip the Pulse Sensor to your earlobe or fingertip and plug it into your Arduino, you can ready to read heart rate. Also, it has an Arduino demo code that makes it easy to use.



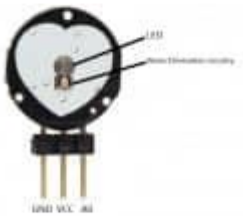


Fig 4,5:Pulse Sensor Front and back side

The pulse sensor has three pins: VCC, GND & Analog Pin.

There is also a LED in the center of this sensor module which helps in detecting the heartbeat. Below the LED, there is a noise elimination circuitry that is supposed to keep away the noise from affecting the readings.

5.2.3 LM35 Temperature Sensor:

The LM35 series are precision integrated-circuit temperature devices with an output voltage linearly proportional to the Centigrade temperature. The LM35 device has an advantage over linear temperature sensors calibrated in Kelvin, as the user is not required to subtract a large constant voltage from the output to obtain convenient Centigrade scaling. The LM35 device does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55°C to 150°C temperature range.

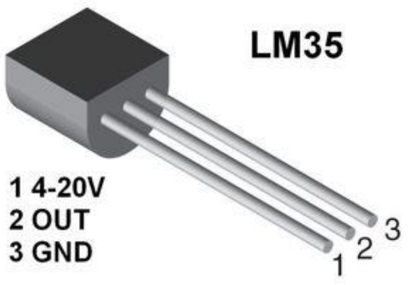


Fig 6: LM35 Mod

5.2.4 ESP8266:

The ESP8266 is a very user-friendly and low-cost device to provide internet connectivity to your projects. The module can work both as an Access point (can create hotspot) and as a station (can connect to Wi-Fi), hence it can easily fetch data and upload it to the internet making the Internet of Things as easy as possible. It can also fetch data from the internet using API’s hence your project could access any information that is available on the internet, thus making it smarter. Another exciting feature of this module is that it can be programmed using the Arduino IDE which makes it a lot more user-friendly.

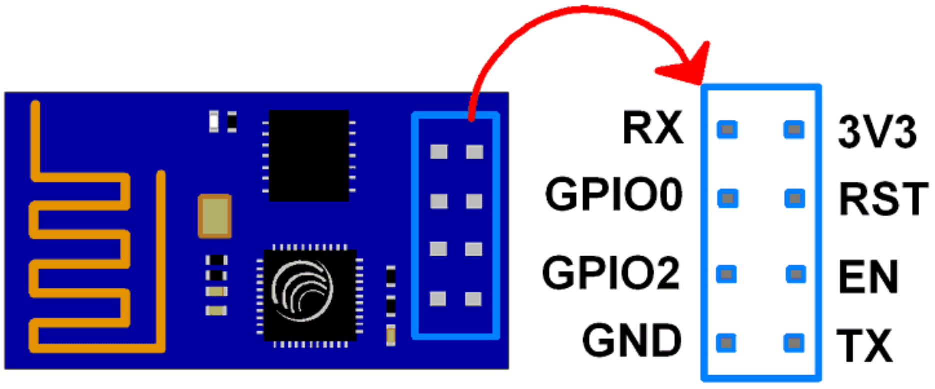


Fig 7:Esp8266 Wifi module

The ESP8266 module works with 3.3V only, anything more than 3.7V would kill the module hence be cautious with your circuits. Here is its pins description.

Pin 1: Ground: Connected to the ground of the circuit
Pin 2: Tx/GPIO – 1: Connected to Rx pin of programmer/uC to upload program
Pin 3: GPIO – 2: General purpose Input/output pin
Pin 4 : CH_EN: Chip Enable/Active high
Pin 5: Flash/GPIO – 0: General purpose Input/output pin
Pin 6 : Reset: Resets the module
Pin 7: RX/GPIO – 3: General purpose Input/output pin
Pin 8: Vcc: Connect to +3.3V only

5.2.5 Arduino UNO

Arduino UNO is a microcontroller board based on the **ATmega328P**. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.

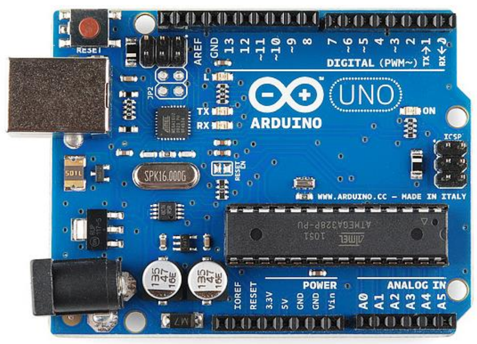


Fig 8 Arduino UNO

5.2.6 Bread Board

Breadboards are temporary work boards for electronic circuit. The horizontal rows are connected throughout the row and may make a complete row with the addition of a simple jumper at the center point. Similarly, the vertical columns are connected vertically down half the width of the board. A jumper may be inserted at the middle location to connect the full width of the breadboard.

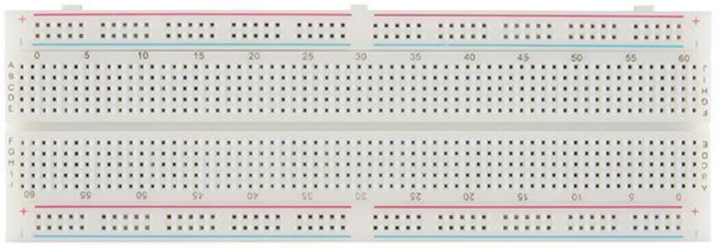


Fig 9: Bread Board

5.2.7 MAX30100 Module Pinout:

The MAX30100 module brings out the following connections.

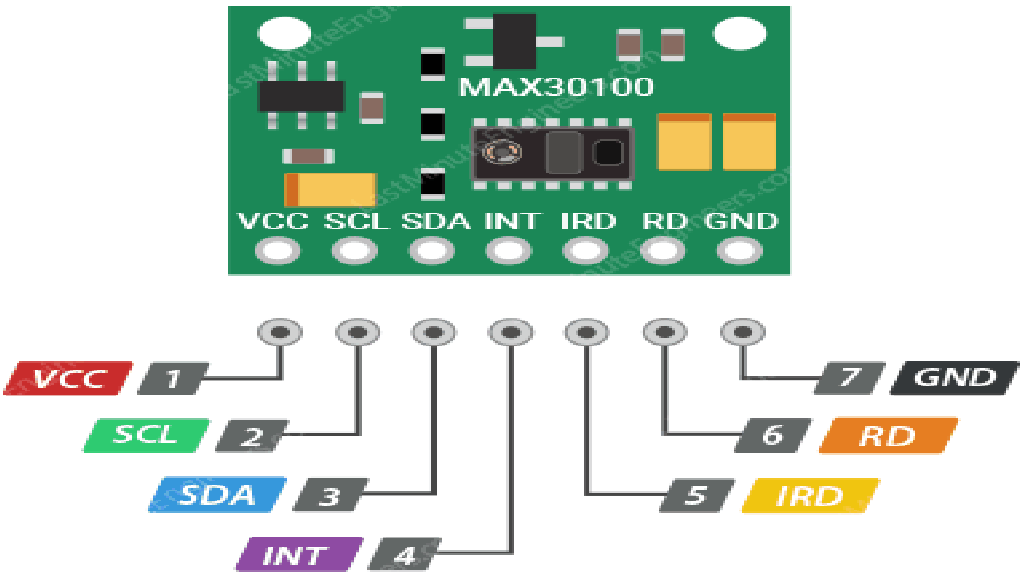


Fig 9

VIN is the power pin. You can connect it to 3.3V or 5V output from your Arduino.

SCL is the I2C clock pin, connect to your Arduino's I2C clock line.

SDA is the I2C data pin, connect to your Arduino's I2C data line.

INT The MAX30100 can be programmed to generate an interrupt for each pulse. This line is open-drain, so it is pulled HIGH by the onboard resistor. When an interrupt occurs the INT pin goes LOW and stays LOW until the interrupt is cleared.

IRD The MAX30100 integrates an LED driver to drive LED pulses for SpO2 and HR measurements. Use this if you want to drive the IR LED yourself, otherwise leave it unconnected.

RD pin is similar to the IRD pin, but is used to drive the Red LED. If you don't want to drive the red LED yourself, leave it unconnected.

GND is the ground.

5.3. Circuit Diagram & Connections:

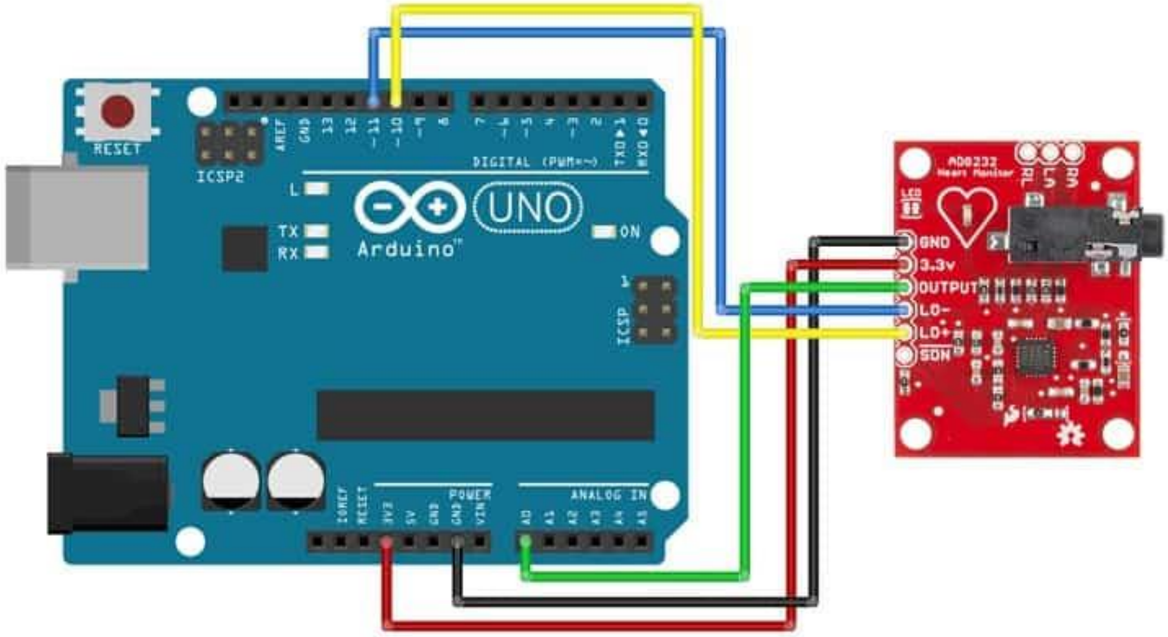


Fig 9:Circuit Diagram of ECG Sensore

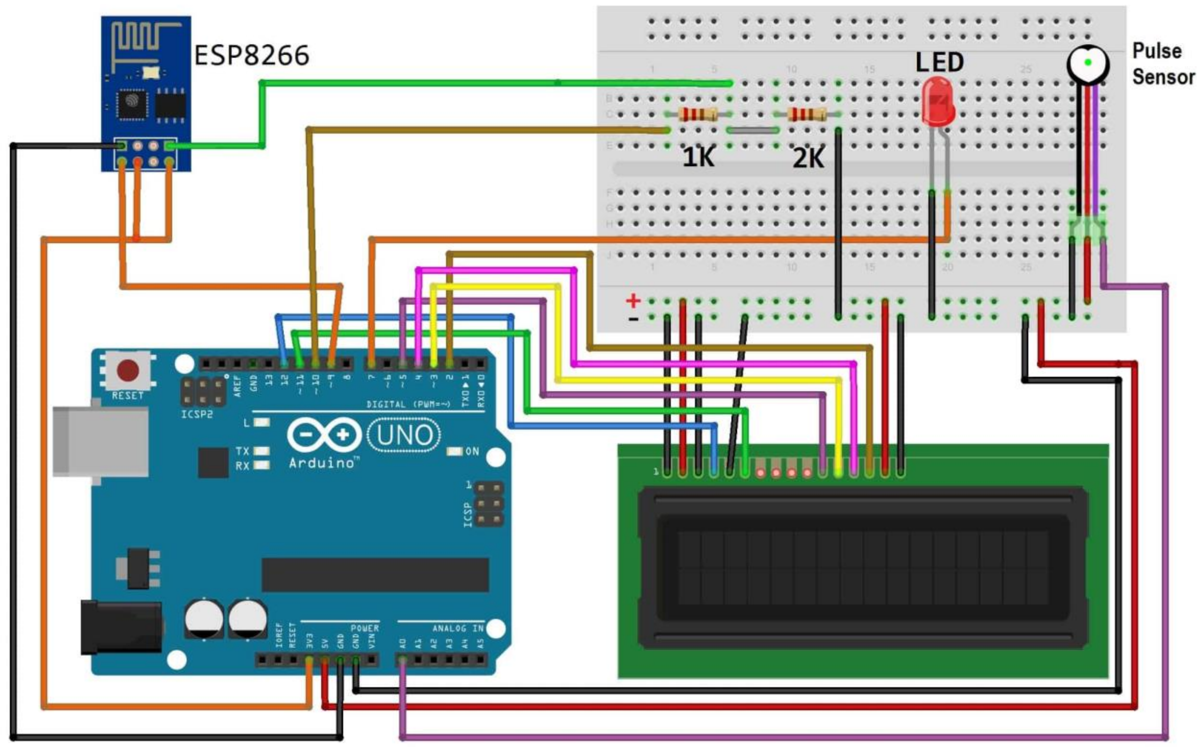


Fig 10:Circuit diagram of pulse and temp

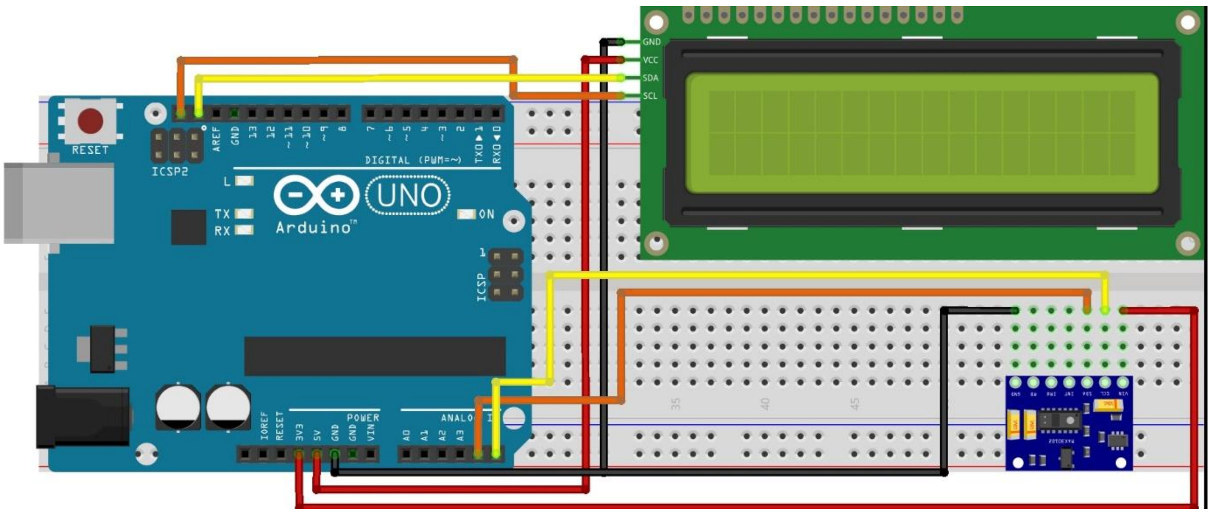
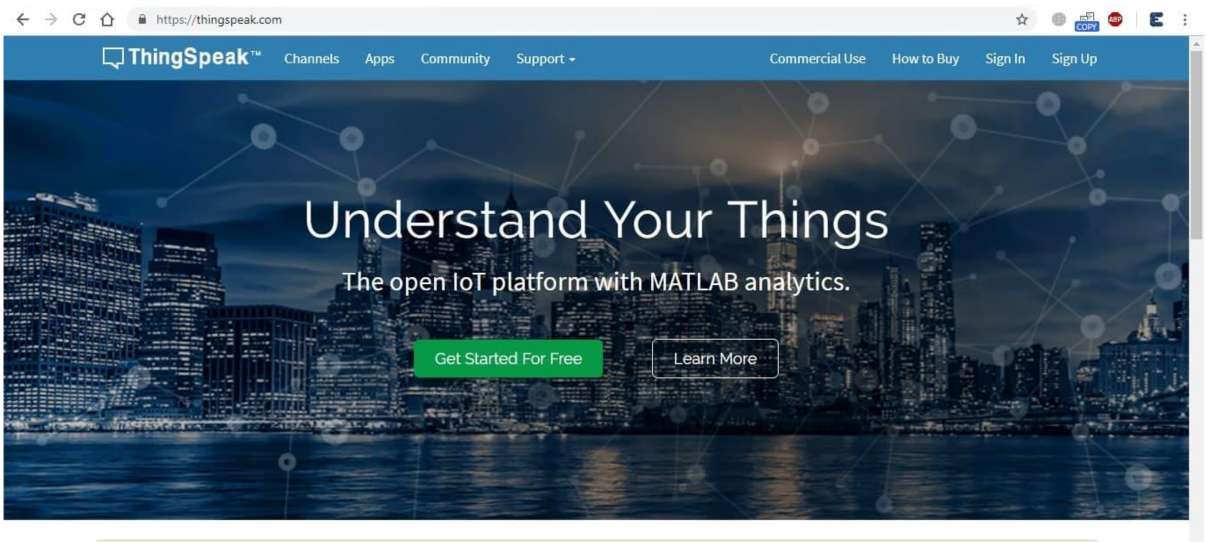


Fig 11:Circuit diagram of spo2 using pulse oximeter sensore

Fig 12:Overal Circuit

5.4.Setting the ThingSpeak:

ThingSpeak provides a very good tool for IoT-based projects. By using the ThingSpeak site, we can monitor our data and control our system over the Internet, using the Channels and web pages provided by ThingSpeak. So first you need to sign up for ThingSpeak. So visit https://thingspeak.com and create an account.

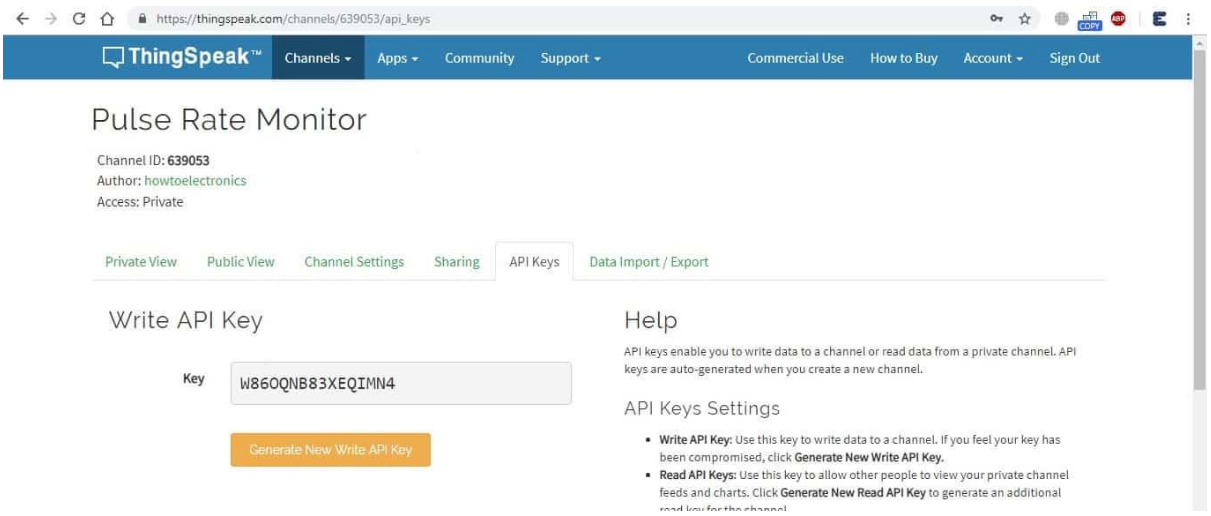


Understand Your Things

The open IoT platform with MATLAB analytics.

Fig 13:Think Speak website

Then create a new channel and set up what you want.Then create the API keys. This key is required for programming modifications and setting your data.



https://thingspeak.com/channels/639053/api_keys

Pulse Rate Monitor

Channel ID: **639053**
Author: howtoelectronics
Access: Private

Write API Key

Key W86OQNB83XEQIMN4

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

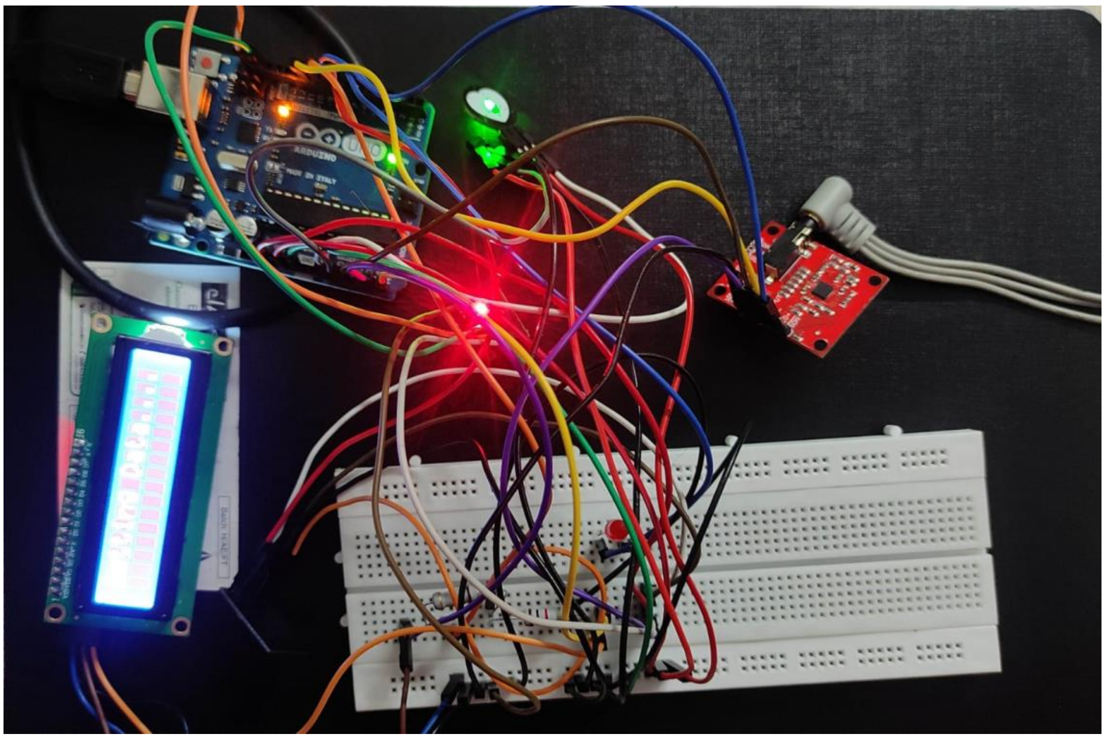
API Keys Settings

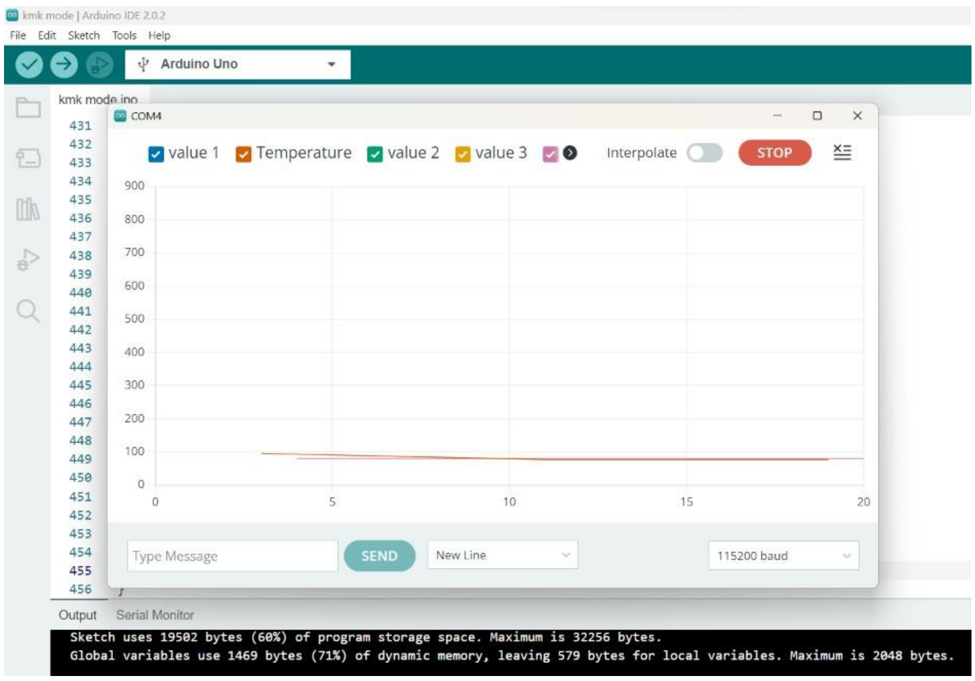
- **Write API Key:** Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key.**
- **Read API Keys:** Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional

Fig 14:api key from think speak

Then upload the code to the Arduino UNO by assembling the circuit shown above. Open the serial monitor and it will automatically connect to Wi-Fi and set up everything.Now click on channels so that you can see the online data streaming, i.e IoT Based Patient Health Monitoring System using ESP8266 & Arduino as shown in the figure here.

6 Results and Discussion:

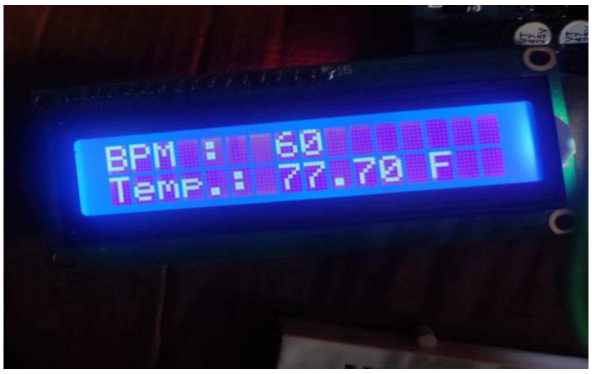




```
Sketch uses 19502 bytes (60%) of program storage space. Maximum is 32256 bytes.
Global variables use 1469 bytes (71%) of dynamic memory, leaving 579 bytes for local variables. Maximum is 2048 bytes.
```

Fig 17:ECG curve display in the serial monitor

When we connect an ECG sensor to the heart along with some other sensors to know the heart rate they record the electrical signals and hardware signals. A computer records information and displays it as waves on the monitor if the test is normal it should show that your heart is beating at an even rate of 60 to 100 beats per minute. Many different heart conditions can show up on an ECG, including a fast, slow, or abnormal heart rhythm, a heart defect, coronary artery disease, heart valve disease, or an enlarged heart. Normal ECG values for waves and intervals are as follows: RR interval: 0.6-1.2 seconds. P wave: 80 milliseconds. PR interval: 120-200 milliseconds.



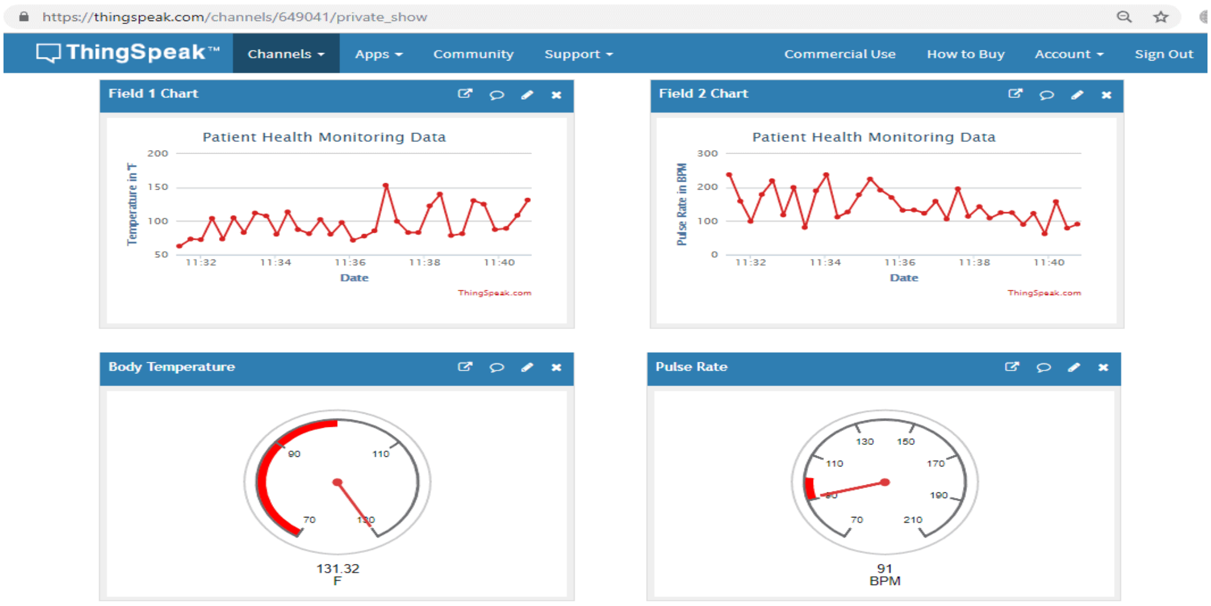


Fig 18:BPM and Temperature display in 16*2 LCD display

When we place the finger on the pulse sensor, the light reflected will change based on the volume of blood inside the capillary blood vessels. This variation in light transmission and reflection can be obtained as a pulse from the output of the pulse sensor. Here we connected the pulse sensor to the Arduino Uno and place the pulse sensor on the figure tip, you should see Arduino Uno built in a led blink with your heartbeat. The range of output from the Pulse Sensor running at 3.3V is only 0 to 675.

The temperature sensor in Arduino Uno board converts the human body temperature to voltage, it further converts the voltage to Celsius, Celsius to Fahrenheit, and prints the Fahrenheit temperature on the LCD screen. We are using a temperature sensor (LM35). When we enter the code and upload it to your Arduino. Once the code is running, open the serial monitor and make sure your baud rate is set to 9600. You will see the temperature displayed in both Fahrenheit and Celsius. The temperature may look incorrect to you.

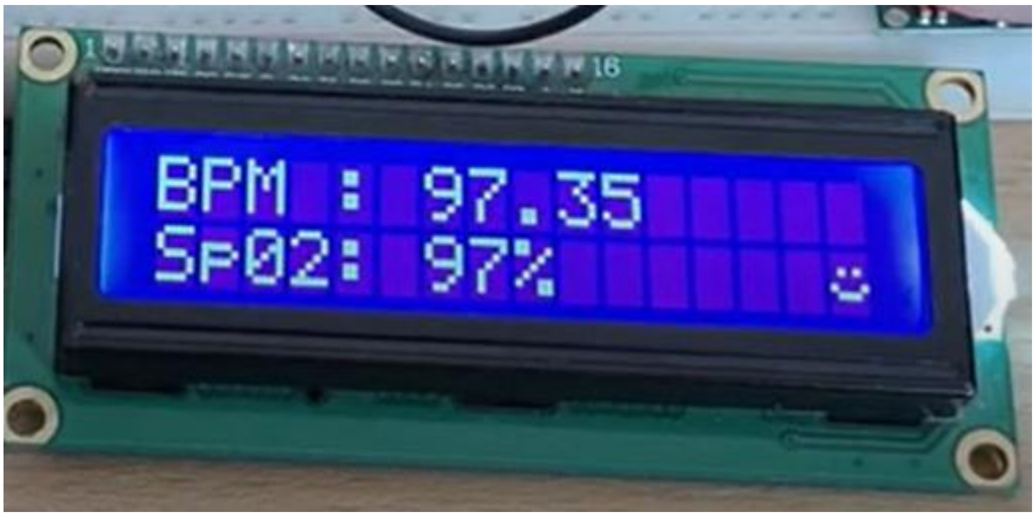


Fig 19:BPM and SpO2 display in 16*2 LCD display

The pulse oximeter uses a cold light source that shines a light through the fingertip, making the tip appear to be red. By analysing the light from the light source that passes through the finger, the device is able to determine the percentage of oxygen in the red blood cell. By seeing the face at the corner of the LCD Screen u can able to determine your SpO2 condition also.

7 Conclusion and Future Scope:

In this project, the process of creating a feasible, low-cost patient health monitoring system that can lower the time for a check-up. This project of a patient health monitoring system has been developed based on the Arduino UNO board. The microcontroller circuit has been developed with less number of components and is highly reliable. After verifying the data that was shown in got in LCD screen, assured about the success of the project. The presence of each module has been systematically out and placed carefully, thus contributing to the best working of every unit. With the use of this system, we can frequently monitor our health. Thus, the functionality of the entire system has been tested thoroughly and it is said to function successfully.

Based on the discussion and the results of the research, it was concluded that the ECG, Temperature, Spo2%, and heartbeat beat detection system development has successfully been built by using a microcontroller Arduino UNO. This tool works in accordance with the instructions carried out by the program and displays the results on the LCD Display. One after the other in a sequential manner.

The human body scanning system could be made more sophisticated by incorporating blood pressure and EEG sensors. The entire medical data acquisition could be made wireless and wearable. Such a package would contain the circuiting for inputs from ECG sensors, EEG sensors, pressure measurement, and pulse rate transducers. This wearable module can transmit the data continuously over a fiber optic link or through an internet digital radio. The received data can be stored in a separate memory and processed by a microcontroller. This enhancement will enable the monitoring of patients to be more flexible and strain-free.

Hence an attempt is made to design a device that not only acts as an alarm system but also can measure the parameters of the body and the attempt made is successful.

8 REFERNECE:

1. N Mahesh, G S Deepak Prasath, E Divyadharshini, V Gokul, "Implementation of Covid-19 Health Monitoring System", *2022 International Conference on Electronics and Renewable Systems (ICEARS)Google scholar*

2. Chirakanphaisarn, Thongkanluang, Chiwpreechar, "Heart rate measurement and electrical pulse signal analysis for subject span of 20–80 years," 2016 Sixth International Conference on Digital Information Processingand Communications (ICDIPC),Beirut, 2016. Goole Scholar

3. Z. Zhiao, Chnaowei and z. Nakdahira, "Healthcare application based on Internet of Things", *Proc. IEET Int. ConfE. on. Technolgy. Application.*, pp. 661-662, Nov. 2013.
Google Scholar

5. Z. Achmed and G. Miguel, "Assimilating Wirel-ess Sensing Nets with Cloud& Computing", *2010 Sixtth Innt. Confe. Mobi. Ad-hocc Sense. Netks*, pp. 263-266, Oct. 2010.
Google Scholar

https://www.hackster.io/iasonas-christoulakis/measure-spo2-heart-rate-and-bpt-using-arduino-68724d

https://forum.arduino.cc/t/oxygen-percentage-saturation-code-spo2/263567

https://www.engineersgarage.com/arduino-based-ecg-cardiac-monitor-ad8232/

https://create.arduino.cc/projecthub/iasonas-christoulakis/measure-spo2-heart-rate-and-bpt-using-arduino-68724d

9 Appendix:

Arduino Source Code:

```
#include <LiquidCrystal_I2C.h>
//--------------------------------------
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"
#define REPORTING_PERIOD_MS     1000
///---------------------------------------------------
LiquidCrystal_I2C lcd(0x26, 16, 2);

#include <SoftwareSerial.h>
float pulse = 0;
float temp = 0;
```

```
SoftwareSerial ser(9,10);
String apiKey = "OO707TGA1BLUNN12";
//--------------------------------------
 byte smile[] = {
  B00000,
  B00000,
  B01010,
  B00000,
  B10001,
  B01110,
  B00000,
  B00000
};
byte mod[] = {
  B00000,
  B00000,
  B01010,
  B00000,
  B11111,
  B00000,
  B00000,
  B00000
};
byte sad[] = {
  B00000,
  B00000,
  B01010,
  B00000,
  B01110,
  B10001,
  B00000,
  B00000
};
PulseOximeter pox;
uint32_t tsLastReport = 0;

void onBeatDetected()
{

  Serial.println("Beat!!!");

}
//----------------------
// Variables
int pulsePin = A0; // Pulse Sensor purple wire connected to analog pin 0
int blinkPin = 7 ; // pin to blink led at each beat
int fadePin = 13; // pin to do fancy classy fading blink at each beat
int fadeRate = 0; // used to fade LED on with PWM on fadePin
```

```
// Volatile Variables, used in the interrupt service routine!

volatile int BPM; // int that holds raw Analog in 0. updated every 2mS
volatile int Signal; // holds the incoming raw data
volatile int IBI = 600; // int that holds the time interval between beats! Must
be seeded!
volatile boolean Pulse = false; // "True" when User's live heartbeat is detected.
"False" when nota "live beat".
volatile boolean QS = false; // becomes true when Arduoino finds a beat.

// Regards Serial OutPut -- Set This Up to your needs
static boolean serialVisual = true; // Set to 'false' by Default. Re-set to
'true' to see Arduino Serial Monitor ASCII Visual Pulse
volatile int rate[10]; // array to hold last ten IBI values
volatile unsigned long sampleCounter = 0; // used to determine pulse timing
volatile unsigned long lastBeatTime = 0; // used to find IBI
volatile int P = 512; // used to find peak in pulse wave, seeded
volatile int T = 512; // used to find trough in pulse wave, seeded
volatile int thresh = 525; // used to find instant moment of heart beat, seeded
volatile int amp = 100; // used to hold amplitude of pulse waveform, seeded
volatile boolean firstBeat = true; // used to seed rate array so we startup
with reasonable BPM
volatile boolean secondBeat = false; // used to seed rate array so we startup
with reasonable BPM

void setup()
{
//-----------------------------------

// initialize the serial communication:
//Serial.begin(9600);
pinMode(9, INPUT); // Setup for leads off detection LO +
pinMode(11, INPUT); // Setup for leads off detection LO -


  //1
  Serial.begin(115200);
  //lcd.begin();
lcd.init();
  lcd.clear();
  lcd.backlight();


  lcd.createChar(1 , smile);
  lcd.createChar(2 , mod);
  lcd.createChar(3 , sad);
  if (!pox.begin()) {
    Serial.println("FAILED");
    for (;;);
  } else {
    Serial.println("SUCCESS");
  }
  pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);

  pox.setOnBeatDetectedCallback(onBeatDetected);




// ----------------------------------------------------
   lcd.init();
  lcd.clear();
  lcd.backlight();
//lcd.begin();
pinMode(blinkPin,OUTPUT); // pin that will blink to your heartbeat!
pinMode(fadePin,OUTPUT); // pin that will fade to your heartbeat!
Serial.begin(115200); // we agree to talk fast!
interruptSetup(); // sets up to read Pulse Sensor signal every 2mS

// IF YOU ARE POWERING The Pulse Sensor AT VOLTAGE LESS THAN THE BOARD VOLTAGE,

// UN-COMMENT THE NEXT LINE AND APPLY THAT VOLTAGE TO THE A-REF PIN

// analogReference(EXTERNAL);

lcd.clear();
lcd.setCursor(0,0);
lcd.print(" Patient Health");
lcd.setCursor(0,1);
lcd.print(" Monitoring ");
delay(4000);
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Initializing....");
delay(5000);
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Getting Data....");
ser.begin(9600);
ser.println("AT");
delay(1000);
ser.println("AT+GMR");
delay(1000);
```

```
ser.println("AT+CWMODE=3");
delay(1000);
ser.println("AT+RST");
delay(5000);
ser.println("AT+CIPMUX=1");
delay(1000);

String cmd="AT+CWJAP=\"Alexahome\",\"98765432\"";
ser.println(cmd);
delay(1000);
ser.println("AT+CIFSR");
delay(1000);
}
// Where the Magic Happens
void loop()
{
 /// ----------------------------------------------------

if((digitalRead(9) == 1)||(digitalRead(11) == 1)){
Serial.println('!');
}
else{
// send the value of analog input 0:
Serial.println(analogRead(A2));
}
//Wait for a bit to keep serial data from saturating
delay(1);


  //1
  pox.update();
  if (millis() - tsLastReport > REPORTING_PERIOD_MS) {

  }
//----------------------------------------------------------------
serialOutput();
if (QS == true) // A Heartbeat Was Found
{

// BPM and IBI have been Determined
// Quantified Self "QS" true when arduino finds a heartbeat
fadeRate = 255; // Makes the LED Fade Effect Happen, Set 'fadeRate' Variable to
255 to fade LED with pulse
serialOutputWhenBeatHappens(); // A Beat Happened, Output that to serial.
QS = false; // reset the Quantified Self flag for next time
}
ledFadeToBeat(); // Makes the LED Fade Effect Happen
delay(20); // take a break
read_temp();
```

```
esp_8266();
}
void ledFadeToBeat()
{
fadeRate -= 15; // set LED fade value
fadeRate = constrain(fadeRate,0,255); // keep LED fade value from going into
negative numbers!
analogWrite(fadePin,fadeRate); // fade LED
}
void interruptSetup()
{
// Initializes Timer2 to throw an interrupt every 2mS.
TCCR2A = 0x02; // DISABLE PWM ON DIGITAL PINS 3 AND 11, AND GO INTO CTC MODE
TCCR2B = 0x06; // DON'T FORCE COMPARE, 256 PRESCALER
OCR2A = 0X7C; // SET THE TOP OF THE COUNT TO 124 FOR 500Hz SAMPLE RATE
TIMSK2 = 0x02; // ENABLE INTERRUPT ON MATCH BETWEEN TIMER2 AND OCR2A
sei(); // MAKE SURE GLOBAL INTERRUPTS ARE ENABLED
}
void serialOutput()
{ // Decide How To Output Serial.
if (serialVisual == true)
{
arduinoSerialMonitorVisual('-', Signal); // goes to function that makes Serial
Monitor Visualizer
}
else
{
sendDataToSerial('S', Signal); // goes to sendDataToSerial function
}
}
void serialOutputWhenBeatHappens()
{
if (serialVisual == true) // Code to Make the Serial Monitor Visualizer Work
{
Serial.print("* Heart-Beat Happened * "); //ASCII Art Madness
Serial.print("BPM: ");
Serial.println(BPM);
}
else
{
sendDataToSerial('B',BPM); // send heart rate with a 'B' prefix
sendDataToSerial('Q',IBI); // send time between beats with a 'Q' prefix
}
}
void arduinoSerialMonitorVisual(char symbol, int data )
{
const int sensorMin = 0; // sensor minimum, discovered through experiment
const int sensorMax = 1024; // sensor maximum, discovered through experiment
```

```
int sensorReading = data; // map the sensor range to a range of 12 options:
int range = map(sensorReading, sensorMin, sensorMax, 0, 11);
// do something different depending on the
// range value:
switch (range)
{
case 0:
Serial.println(""); /////ASCII Art Madness
break;
case 1:
Serial.println("---");
break;
case 2:
Serial.println("------");
break;
case 3:
Serial.println("---------");
break;
case 4:
Serial.println("------------");
break;
case 5:
Serial.println("--------------|-");
break;
case 6:
Serial.println("--------------|---");
break;
case 7:
Serial.println("--------------|-------");
break;
case 8:
Serial.println("--------------|----------");
break;
case 9:
Serial.println("--------------|----------------");
break;
case 10:
Serial.println("--------------|-------------------");
break;
case 11:
Serial.println("--------------|-----------------------");
break;
}
}

void sendDataToSerial(char symbol, int data )
{
Serial.print(symbol);
Serial.println(data);
}
ISR(TIMER2_COMPA_vect) //triggered when Timer2 counts to 124
{
cli(); // disable interrupts while we do this
Signal = analogRead(pulsePin); // read the Pulse Sensor
sampleCounter += 2; // keep track of the time in mS with this variable
int N = sampleCounter - lastBeatTime; // monitor the time since the last beat
to avoid noise
// find the peak and trough of the pulse wave

if(Signal < thresh && N > (IBI/5)*3) // avoid dichrotic noise by waiting 3/5 of
last IBI
{
if (Signal < T) // T is the trough
{
T = Signal; // keep track of lowest point in pulse wave
}
}
if(Signal > thresh && Signal > P)
{ // thresh condition helps avoid noise
P = Signal; // P is the peak
} // keep track of highest point in pulse wave
// NOW IT'S TIME TO LOOK FOR THE HEART BEAT
// signal surges up in value every time there is a pulse
if (N > 250)
{ // avoid high frequency noise
if ( (Signal > thresh) && (Pulse == false) && (N > (IBI/5)*3) )
{
Pulse = true; // set the Pulse flag when we think there is a pulse
digitalWrite(blinkPin,HIGH); // turn on pin 13 LED
IBI = sampleCounter - lastBeatTime; // measure time between beats in mS
lastBeatTime = sampleCounter; // keep track of time for next pulse

if(secondBeat)
{ // if this is the second beat, if secondBeat == TRUE
secondBeat = false; // clear secondBeat flag
for(int i=0; i<=9; i++) // seed the running total to get a realisitic BPM at
startup
{
rate[i] = IBI;
}
}
if(firstBeat) // if it's the first time we found a beat, if firstBeat == TRUE
{
firstBeat = false; // clear firstBeat flag
secondBeat = true; // set the second beat flag
sei(); // enable interrupts again
return; // IBI value is unreliable so discard it
}
// keep a running total of the last 10 IBI values
word runningTotal = 0; // clear the runningTotal variable
for(int i=0; i<=8; i++)
{ // shift data in the rate array
rate[i] = rate[i+1]; // and drop the oldest IBI value
runningTotal += rate[i]; // add up the 9 oldest IBI values
}
rate[9] = IBI; // add the latest IBI to the rate array
runningTotal += rate[9]; // add the latest IBI to runningTotal
runningTotal /= 10; // average the last 10 IBI values
BPM = 60000/runningTotal; // how many beats can fit into a minute? that's BPM!
QS = true; // set Quantified Self flag
// QS FLAG IS NOT CLEARED INSIDE THIS ISR
pulse = BPM;
}
}
if (Signal < thresh && Pulse == true)
{ // when the values are going down, the beat is over
digitalWrite(blinkPin,LOW); // turn off pin 13 LED
Pulse = false; // reset the Pulse flag so we can do it again
amp = P - T; // get amplitude of the pulse wave
thresh = amp/2 + T; // set thresh at 50% of the amplitude
P = thresh; // reset these for next time
T = thresh;
}
if (N > 2500)
{ // if 2.5 seconds go by without a beat
thresh = 512; // set thresh default
P = 512; // set P default
T = 512; // set T default
lastBeatTime = sampleCounter; // bring the lastBeatTime up to date
firstBeat = true; // set these to avoid noise
secondBeat = false; // when we get the heartbeat back
}
sei(); // enable interrupts when youre done!
}// end isr
void esp_8266()
{
// TCP connection AT+CIPSTART=4,"TCP","184.106.153.149",80
String cmd = "AT+CIPSTART=4,\"TCP\",\"";
cmd += "184.106.153.149"; // api.thingspeak.com
cmd += "\",80";
ser.println(cmd);
Serial.println(cmd);
if(ser.find("Error"))
{
```

```
Serial.println("AT+CIPSTART error");
return;
}
String getStr = "GET /update?api_key=";
getStr += apiKey;
getStr +="&field1=";
getStr +=String(temp);
getStr +="&field2=";
getStr +=String(pulse);
getStr += "\r\n\r\n";
// send data length
cmd = "AT+CIPSEND=4,";
cmd += String(getStr.length());
ser.println(cmd);
Serial.println(cmd);
delay(1000);
ser.print(getStr);
Serial.println(getStr); //thingspeak needs 15 sec delay between updates
delay(3000);
}
void read_temp()
{
int temp_val = analogRead(A1);
float mv = (temp_val/1024.0)*5000;
float cel = mv/10;
temp = (cel*9)/5 + 32;
Serial.print("Temperature:");
Serial.println(temp);

//--------------------------------------
   delay(800);
    lcd.clear();
    lcd.setCursor(0 , 0);
    lcd.print("BPM : ");
    lcd.print(pox.getHeartRate());
    lcd.setCursor(0 , 1);
    lcd.print("Sp02: ");
    lcd.print(pox.getSpO2());
    lcd.print("%");
    tsLastReport = millis();

    if (pox.getSpO2() >= 96) {
      lcd.setCursor(15 , 1);
      lcd.write(1);
        delay(500);
    }
    else if (pox.getSpO2() <= 95 && pox.getSpO2() >= 91) {
      lcd.setCursor(15 , 1);
      lcd.write(2);
          delay(500);
    }
    else if (pox.getSpO2() <= 90) {
      lcd.setCursor(15 , 1);
      lcd.write(3);
      delay(500);
    }
    delay(1000);
//------------------------------------
lcd.clear();
lcd.setCursor(0,0);
lcd.print("BPM :");
lcd.setCursor(7,0);
lcd.print(BPM);
lcd.setCursor(0,1);
lcd.print("Temp.:");
lcd.setCursor(7,1);
lcd.print(temp);
lcd.setCursor(13,1);
lcd.print("F");
}
```