## Project No 6:-IOT based Health Monitoring Sysytem

A microcontroller-based device with appropriate bio-medical sensors will be attached to patient to provide constant cloud-based monitoring. The vital signs i.e. temperature and pulse rate of human body which are major clues to detect any health problem will be sensed by respective sensors supported by NodeMCU in a Wi-Fi environment and the data will be sent to ThingSpeak cloud where the data will be analysed to look for any irregularity. In case of any irregularity a notification will be sent to doctors and nurses.

By this system, patients can be kept under proper constant monitoring without being dependent on any human’s responsibility at a very low cost. This will also reduce any possible errors and help the doctor to quickly respond to the situation.

# Components Required

* ESP8266
* Heart Rate Sensor
* DHT11 Temparature Sensor or DS18B20 Sensor
* USB cable
* Connecting wires
* Thingspeak account

## Circuit Diagram

## IMG_256

****Connections:****

**Connect the Heart rate sensor to node MCU as shown in Figure**

**Heart Rate Sensor--------------Node MCU**

**(-) pin -------------------------GND**

**(+) pin -------------------------3V3**

**OUT pin ---------------------------A0**

****Connect the DHT 11 or DS18B20 to node MCU****

**Temparature Sensor---------Node MCU**

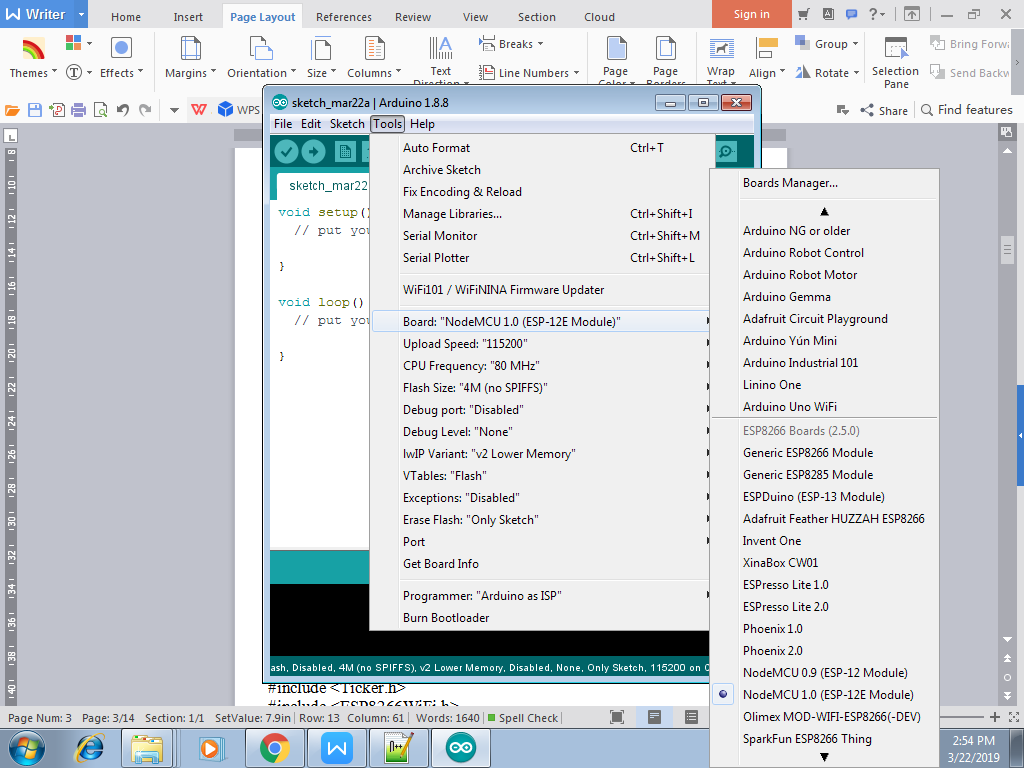
**(-)pin ---------------------------GND**

**(+) pin --------------------------3V3**

**Out pin --------------------------D4**

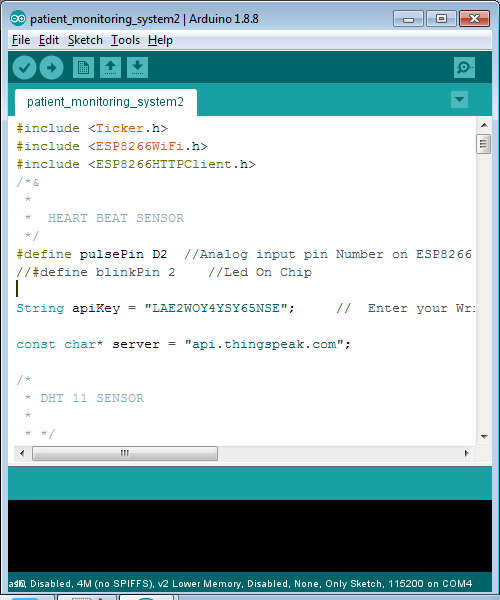
****Coding AND Thingspeak****

Upload the code and setup your thingspeak channel to receive the data (you can easily find a lot of tutorials regarding this on internet. Make sure the field 1 is for BPM and field 2 is for temperature on your thingspeak channel and then, select NodeMCU as your board.Now, upload the code and make sure to edit the WiFi credentials and the thingspeak API key accordingly in the code before uploading.



****Node Write the following code and upload it in Node MCU****

****Note:- Change WiFi and thingspeak channel credential in the code.****



# Software

1. **Arduino IDE code**

#include <Ticker.h>

#include <ESP8266WiFi.h>

#include <ESP8266HTTPClient.h>

/\*&

\*

\* HEART BEAT SENSOR

\*/

#define pulsePin D2 //Analog input pin Number on ESP8266

//#define blinkPin 2 //Led On Chip

String apiKey = "Your APi Key";

const char\* server = "api.thingspeak.com";

/\*

\* DHT 11 SENSOR

\*

\* \*/

#include <SimpleDHT.h>

int pinDHT11 = D0;

SimpleDHT11 dht11(pinDHT11);

byte temperature = 0;

byte humidity = 0;

int temperature1;

int humidity1;

////////////////

//LOOP VARIABLES

////////////////

volatile int BPM;

volatile int Signal;

volatile int IBI = 600;

volatile boolean Pulse = false;

volatile boolean QS = false;

//////////////////////

//INTERRRUPTS VARIABLE

//////////////////////

Ticker flipper;

volatile int rate[10];

volatile unsigned long sampleCounter = 0;

volatile unsigned long lastBeatTime = 0;

volatile unsigned long current;

volatile int P = 512;

volatile int T = 512;

volatile int thresh = 560; // used to find instant moment of heart beat, seeded 530du

volatile int amp = 0; // 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

volatile unsigned long lastMillis = 0; // used to determine pulse timing

volatile float tempSignal=0;

////////////////

//WiFi VARIABLES

////////////////

const char\* ssid = "Your SSID";

const char\* password = "Your Password";

const char\* host = "Web Service Hostname";

const int port = 5192; //Web Service Port Number

// We now create a URI for the request

String url = "POST /MyService/GetValue?BPM="; //Web Service Post request URL

volatile int msTime = 0;

WiFiClient client;

#include <ESP8266WebServer.h>

//#include "index.h"

//WiFiServer server(80);

//ESP8266WebServer serverAP(8000);

//ESP8266WebServer server(80); //Server on port 80

//const char\* myssid = "NodeMcu"; //SSID AP MODE

//const char\* mypassword = "admin123";//Password AP MODE

//String posing\_data = ""<!DOCTYPE html><html><head><meta http-equiv="refresh" content="2"></head><h1><td>heart Beat </td></h1>"";

//String posting\_data = "<h1><td>heart Beat </td></h1>";

/\*void handleRoot()

{

String s1 = MAIN\_page;

String s2 = MAIN\_page2;

String s3 = MAIN\_page3;

String s4 = MAIN\_page4;

String s5 = MAIN\_page5;

String s = s1+s2+"="+BPM+s4+"="+celsius+s5+"="+"0""<br>"+s3;

server.send(200, "text/html",s);

Serial.println("data sent on request");

}\*/

void setup() {

//pinMode(blinkPin,OUTPUT); // pin that will blink to your heartbeat!

Serial.begin(115200);

delay(10);

Serial.println();

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

//server.on("/", handleRoot); //Which routine to handle at root location

//server.begin();

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

delay(2000);

interruptSetup(); // sets up to read Pulse Sensor signal every 2mS

lastMillis=millis();

}

void loop() {

//tempdata();

//server.handleClient(); //Handle client requests

//check\_temp();

\_check\_temp();

if (QS == true)

{ // A Heartbeat Was Found

// BPM and IBI have been Determined

// Quantified Self "QS" true when arduino finds a heartbeat

serialOutputWhenBeatHappens(); // A Beat Happened, Output that to serial.

QS = false; // reset the Quantified Self flag for next time

}

if(BPM > 60 && BPM < 120)

{

Serial.print("BPM : ");

Serial.println(BPM);

}

else

{

BPM = random(60,100);

Serial.print("BPM : .....");

Serial.println(BPM);

}

if (isnan(BPM) || isnan(temperature1))

{

Serial.println("Failed to read from DHT sensor!");

return;

}

if (client.connect(server,80)) // "184.106.153.149" or api.thingspeak.com

{

String postStr = apiKey;

postStr +="&field1=";

postStr += String(temperature1);

postStr +="&field2=";

postStr += String(BPM);

postStr += "\r\n\r\n";

client.print("POST /update HTTP/1.1\n");

client.print("Host: api.thingspeak.com\n");

client.print("Connection: close\n");

client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");

client.print("Content-Type: application/x-www-form-urlencoded\n");

client.print("Content-Length: ");

client.print(postStr.length());

client.print("\n\n");

client.print(postStr);

Serial.println("----------------------");

Serial.println("--send to thinkspeak--");

Serial.println("-------=--------------");

}

else

{

Serial.println("server didn't connect....");

}

client.stop();

delay(15000);

}

// CHECK BODY TEMPEEATURE

void \_check\_temp()//dht temp check....

{

int err = SimpleDHTErrSuccess;

if ((err = dht11.read(&temperature, &humidity, NULL)) != SimpleDHTErrSuccess) {

Serial.print("Read DHT11 failed, err="); Serial.println(err);delay(1000);

return;

}

Serial.print("temperature:");

Serial.print((int)temperature); Serial.print(" \*C \t");

Serial.print("Humidity:");

Serial.print((int)humidity); Serial.println(" H \t");

temperature1=(int)temperature;

humidity1=(int)humidity;

delay(1500);

float h = (int)humidity;

float t = (int)temperature;

}

// Decides How To OutPut BPM and IBI Data

void serialOutputWhenBeatHappens() {

sendDataToSerial('B',BPM); // send heart rate with a 'B' prefix

sendDataToSerial('Q',IBI); // send time between beats with a 'Q' prefix

}

void sendDataToSerial(char symbol, int data ) {

Serial.print(symbol);

Serial.println(data);

}

void interruptSetup() {

// Initializes Ticker to have flipper run the ISR to sample every 2mS as per original Sketch.

flipper.attach\_ms(2, ISRTr);

}

void ISRTr() {

noInterrupts(); // disable interrupts while we do this

Signal = analogRead(pulsePin); // read the Pulse Sensor

current=millis();

int difference = current-lastMillis;

lastMillis=current;

sampleCounter += difference; // keep track of the time in mS with this variable

msTime+=difference;

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,LOW); // 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

interrupts(); // 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

}

}

if (Signal < thresh && Pulse == true) { // when the values are going down, the beat is over

//digitalWrite(blinkPin,HIGH); // 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 = 530; // 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

BPM=0;

}

interrupts(); // enable interrupts when youre done!

}

**OUTPUT:-**

Afer uploading the code in Node MCU. Place your finger on Heart rate sensor you can see your heartbeat rate on thingspeak account as follow.

