IoT Based Patient Health Monitoring using ESP8266 & Arduino(Hardware)

[GROUP – 1]

INTRODUCTION

IoT is quickly revolutionising the healthcare sector thanks to a proliferation of new healthcare technology start-ups. In this project, we used the ESP8266 & Arduino platforms to develop an IoT-based patient health monitoring system. ThingSpeak is the IoT platform employed in this project. Using the HTTP protocol over the Internet or over a Local Area Network, ThingSpeak is an open-source Internet of Things (IoT) application and API that stores and retrieves data from objects. The ambient temperature and heart rate could both be read by this Internet of Things gadget. It updates them to an IoT platform while continuously checking the temperature and pulse rate.

The Arduino Sketch running over the device implements the various functionalities of the project like reading sensor data, converting them into strings, passing them to the IoT platform, and displaying measured pulse rate and temperature on character LCD.

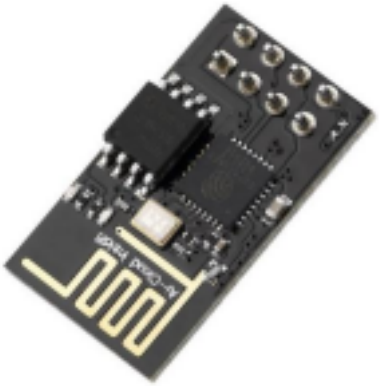
HARDWARE COMPONENTS

1. **Arduino UNO Board**

   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

2. **ESP8266-01 WiFi Module**

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.



3. **16x2 LCD Display**

4. **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.

5. **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.

6. **1K Resistor**

7. **Jumper wires**

8. **Breadboard**

SOFTWARE

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.

- Then create the API keys**.** This key is required for programming modifications and setting your data.

- 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

Arduino IDE

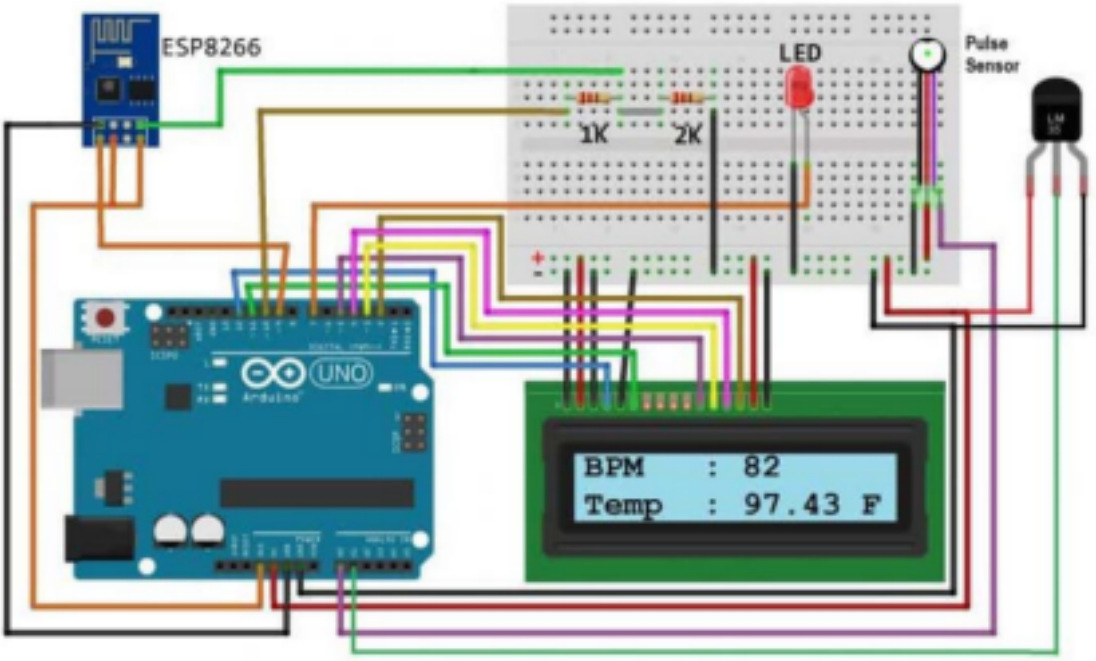
The Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards.

The Arduino IDE (Integrated Development Environment) is a software application used to write and upload code to Arduino microcontroller boards. It provides an easy-to-use interface for programming and developing projects using the Arduino platform.

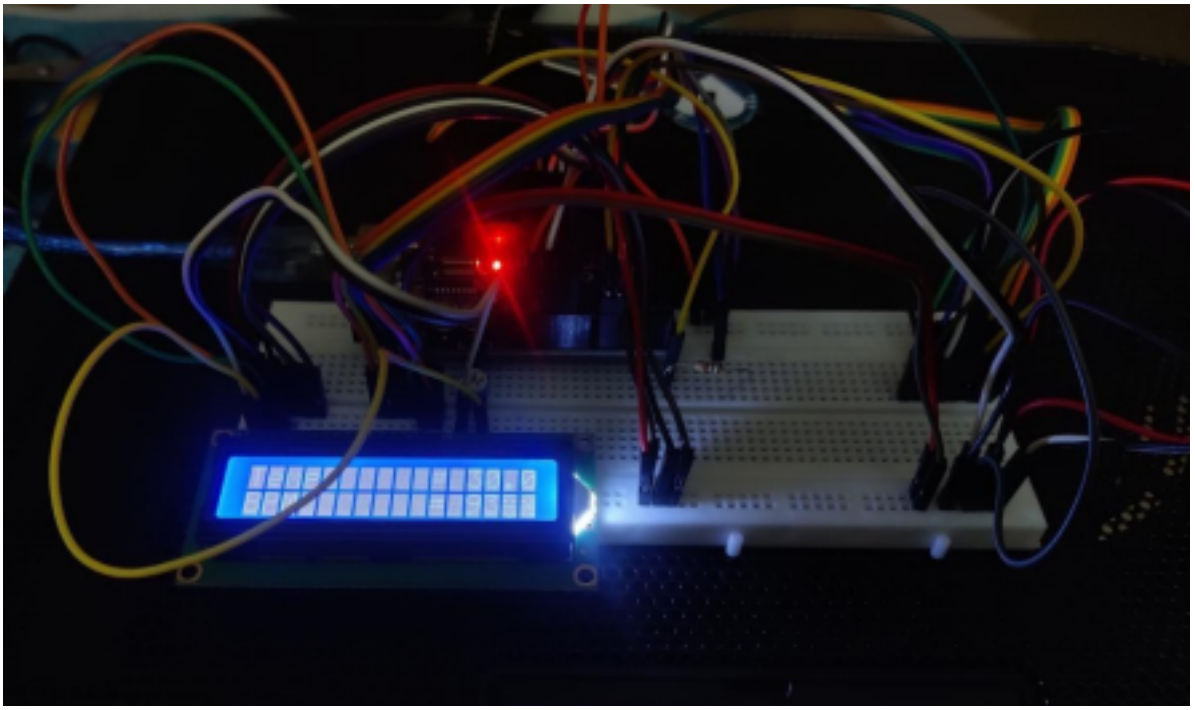
PROCEDURE :

1. Connect Pulse Sensor output pin to A0 of Arduino and other two pins to VCC & GND.
2. Connect LM35 Temperature Sensor output pin to A1 of Arduino and other two pins to VCC & GND.
3. Connect Pin 1,3,5,16 of LCD to GND.
4. Connect Pin 2,15 of LCD to VCC.
5. Connect Pin 4,6,11,12,13,14 of LCD to Digital Pin12,11,5,4,3,2 of Arduino. 6. The RX pin of ESP8266 works on 3.3V and it will not communicate with the Arduino when we will connect it directly to the Arduino. So, we will have to make a voltage divider for it which will convert the 5V into 3.3V. This can be done by connecting the 2.2K & 1K resistor. Thus the RX pin of the ESP8266 is connected to pin 10 of Arduino through the resistors.
7. Connect the TX pin of the ESP8266 to pin 9 of the Arduino.

Circuit Diagram :



Hardware setup :



Simulation - Arduino IDE

```
#include <SoftwareSerial.h>

#define RX 9

#define TX 10

#define sensorPin A1
```

```
#define USE_ARDUINO_INTERRUPTS true
#include <PulseSensorPlayground.h>
#include <LiquidCrystal.h>
String AP = "GNXS-2.4G-BB93A0"; // AP NAME
String PASS = "BC62D2BB93A0"; // AP PASSWORD
String API = "FU2YZYEKG3XZMTG2"; // Write API
KEY String HOST = "api.thingspeak.com";
String PORT = "80";
String field = "field1";
String field2 = "field2";
int countTrueCommand;
int countTimeCommand;
boolean found = false;
int valSensor = 1;
int valSensor2 = 1;
int myBPM;
const int PulseWire = 0;
const int LED = LED_BUILTIN;
int Threshold = 550;
PulseSensorPlayground pulseSensor;
SoftwareSerial esp8266(RX,TX);


const int rs = 8, en = 11, d4 = 3, d5 = 4, d6 = 5, d7 = 6;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);


void setup() {
 Serial.begin(9600);
 esp8266.begin(115200);
```

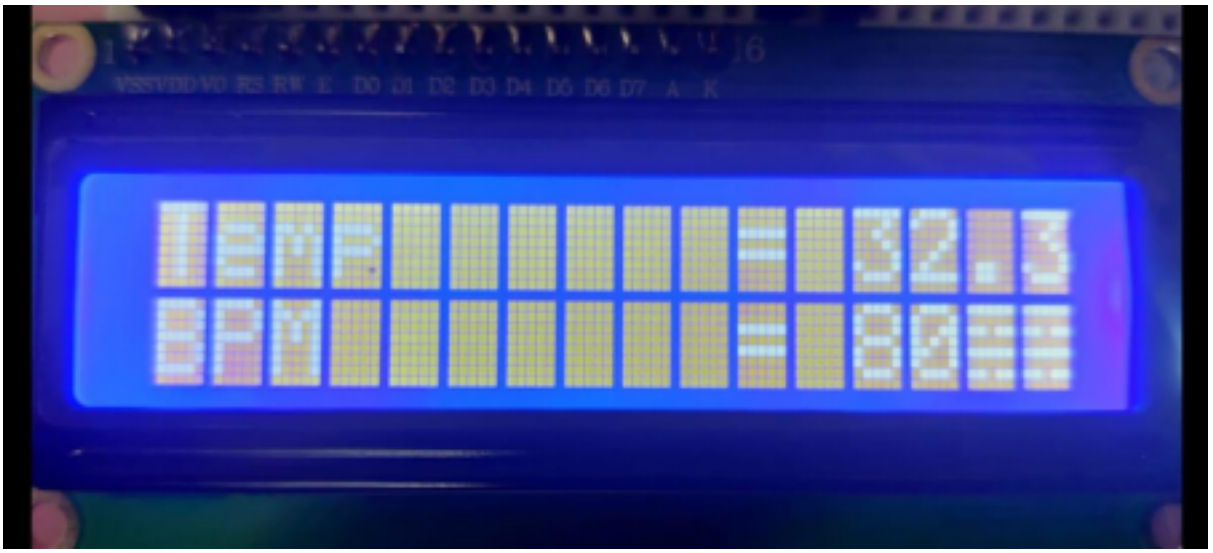
```
 sendCommand("AT",5,"OK");
 sendCommand("AT+CWMODE=1",5,"OK");
 sendCommand("AT+CWJAP=\""+ AP +"\",\""+ PASS +"\"",20,"OK");
lcd.begin(16, 2);
 pulseSensor.analogInput(PulseWire);
 pulseSensor.blinkOnPulse(LED);
 pulseSensor.setThreshold(Threshold);
 if (pulseSensor.begin()) {
 Serial.println("Pulse Sensor Success!");
}
}

void loop() {

 valSensor = getSensorData();
 String getData = "GET /update?api_key="+ API +"&"+ field
+"="+String(valSensor); sendCommand("AT+CIPMUX=1",5,"OK");
 sendCommand("AT+CIPSTART=0,\"TCP\",\""+ HOST +"\","+
 PORT,15,"OK"); sendCommand("AT+CIPSEND=0,"
 +String(getData.length()+4),4,">");
 esp8266.println(getData);delay(500);countTrueCommand++;
 sendCommand("AT+CIPCLOSE=0",5,"OK");

 valSensor2 = getSensorData2();
 String getData2 = "GET /update?api_key="+ API +"&"+ field2 +"="+String(valSensor2);
sendCommand("AT+CIPMUX=1",5,"OK");
 sendCommand("AT+CIPSTART=0,\"TCP\",\""+ HOST +"\","+
 PORT,15,"OK"); sendCommand("AT+CIPSEND=0,"
```

```
 +String(getData2.length()+4),4,">");
 esp8266.println(getData2);delay(500);countTrueCommand++;
 sendCommand("AT+CIPCLOSE=0",5,"OK");
}
int getSensorData(){
 int reading = analogRead(sensorPin);
 float voltage = reading * (5.0 / 1024.0)/12;
 float temperatureC = voltage * 100;
 Serial.print("Temperature: ");
 Serial.println(temperatureC);

 lcd.setCursor(0,0);
 lcd.print("Temp = ");
 lcd.println(temperatureC);



 return temperatureC;
}

int getSensorData2(){
if (pulseSensor.sawStartOfBeat()) {
int myBPM = pulseSensor.getBeatsPerMinute();
if(myBPM<120){
 Serial.print("BPM: ");
 Serial.println(myBPM);
 lcd.setCursor(0,1);
 lcd.print("BPM = ");
 lcd.println(myBPM);
 lcd.print(".00");
```

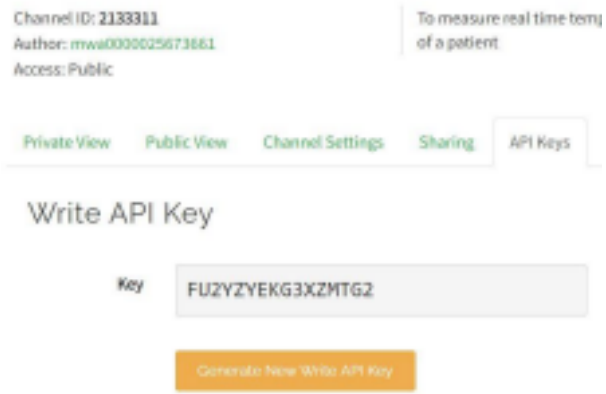
```
 return myBPM;
}
}
}
void sendCommand(String command, int maxTime, char readReplay[])
{  Serial.print(countTrueCommand);
 Serial.print(". at command => ");
 Serial.print(command);
 Serial.print(" ");
 while(countTimeCommand < (maxTime*1))
 {
 esp8266.println(command);//at+cipsend
 if(esp8266.find(readReplay))//ok
 {
 found = true;
 break;
 }

 countTimeCommand++;
 }

 if(found == true)
 {
 Serial.println("OYI");
 countTrueCommand++;
 countTimeCommand = 0;
 }
```

```
if(found == false)
{
Serial.println("Fail");
countTrueCommand = 0;
countTimeCommand = 0;
}
found = false;
```

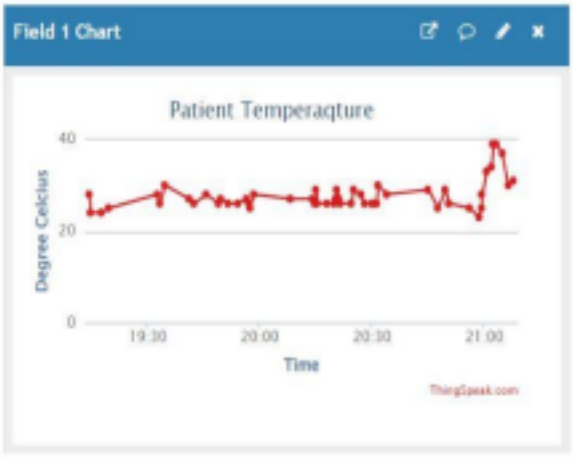
Outputs



ThingSpeak



Outputs



Conclusion:

"IoT Based Patient Health Monitoring using ESP8266 & Arduino" project is an innovative and practical solution for remote patient monitoring. It provides a reliable, accurate, and real-time monitoring system for patients' health parameters, enabling healthcare professionals to provide timely and effective care.

References:

https://how2electronics.com/iot-patient-health-monitoring-system-esp8266/

https://www.arduino.cc/

https://www.espressif.com/en/products/socs/esp8266