SMART HEALTH MONITORING SYSTEM

GROUP MEMBERS

MUKIL SENTHILKUMAR

20EC10048

OINDRILA MAJI

20IM30013

SAHIL DAS

20CY20025

LOHITYA WASNIK

20MI10030

CONTENTS

- Acknowledgement 1
- Weekly Analysis 2-3
- Executive Summary 4
- Introduction 5-7
- Arduino UNO 8-9
- Bread Board 10
- Jumper Wires 11
- Node MCU 12-13
- Temperature Sensor (LM35) 14

- Pulse Sensor 15-16
- BLYNK App 17-25
- Circuit Diagram 26
- Programming 27-34
- Wiring of Real Hardware 35
- User and Mobile Interface 36
- Advantages 37
- References 38-39
- Video Link 40

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our mentors **Mr. Arkopal Goswami Sir**, **Mrs. Mamoni Banerjee Ma'am** and **Mrs. Gouri Gragate Ma'am** as well as all of the TAs for their advice and guidance without which this project would not have been complete. They gave us golden opportunity to do this wonderful project which also helped us in doing a lot of research work and we came to know about so many new things about sensors. We learn how to do team work which is an essential part of a personality development of an individual. We also thankful to our parents who motivates us all the time. All our group members co-oporate with each other in difficult times. Ultimately, we would like to thank the supreme power for providing us strength to face obstacles and challenges.

WEEKLY ANALYSIS

WEEK 1

- Selection of topic for group project.
- Collecting information about temperature (LM35) and pulse rate sensor.
- Buying of all components required for the project.

WEEK 2

- Developing program for measuring temperature and pulse rate of a person.
- Linking code with Arduino UNO.

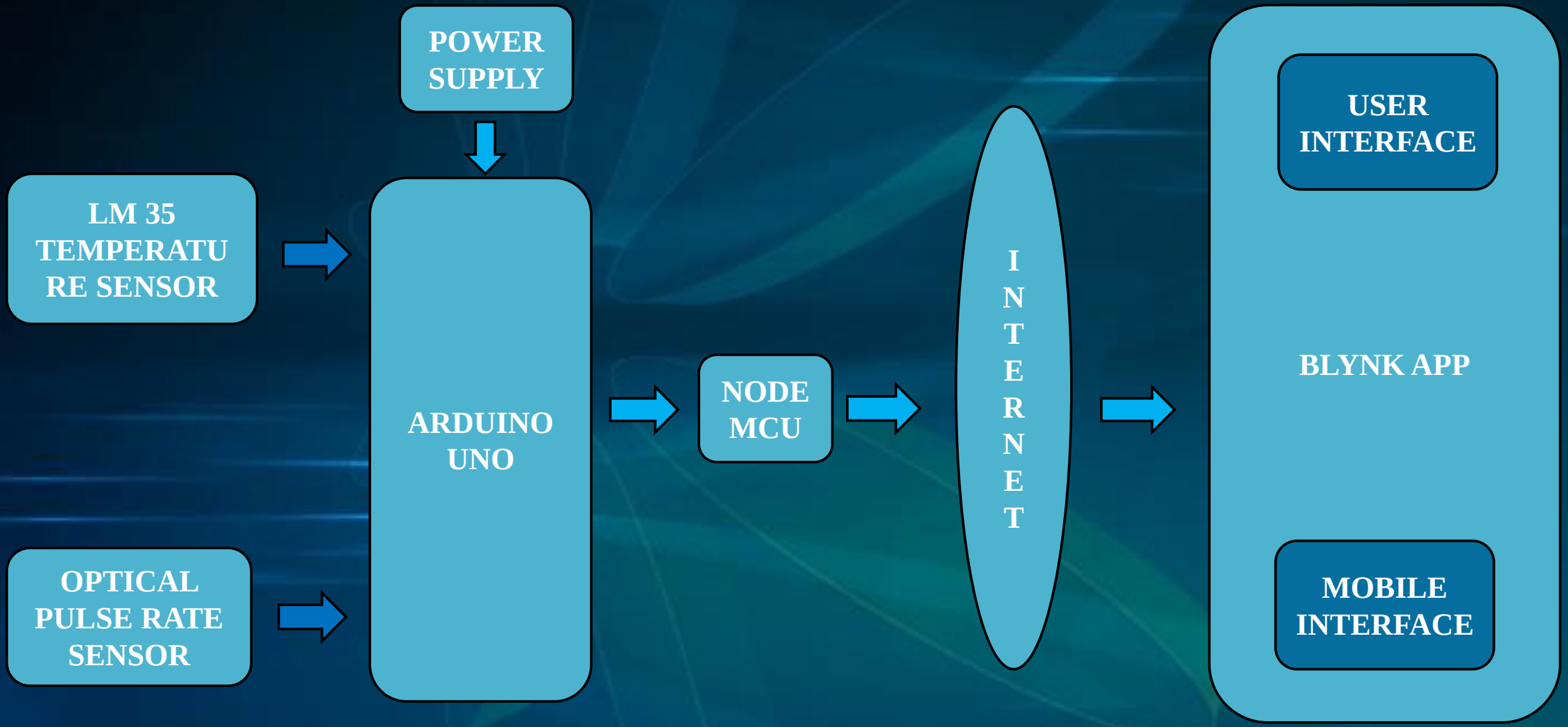
WEEK 3

- Developing program for user interface.
- Learning about Blynk App.

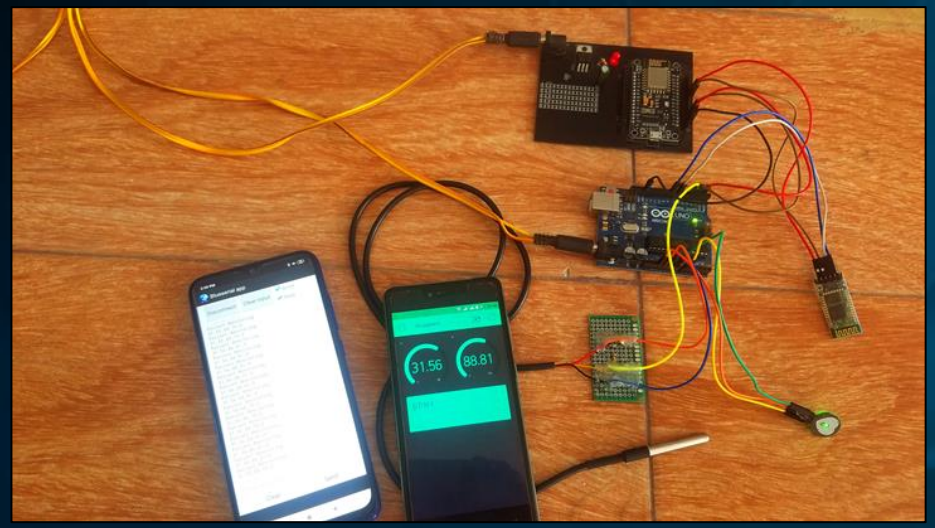
WEEK 4

- Testing the working of Temperature and Pulse rate sensor.
- Preparation of video and presentation.

EXECUTIVE SUMMARY



INTRODUCTION



Health monitoring is the major problem in today's world. Due to lack of proper health monitoring, patient suffer from serious health issues. There are lots of IoT devices now days to monitor the health of patient over internet. Health experts are also taking advantage of these smart devices to keep an eye on their patients. With tons of new healthcare technology start-ups, IoT is rapidly revolutionizing the healthcare industry.

Here in this project, we will make an IoT based Health Monitoring System which records the patient heart beat rate and body temperature and also send an email/SMS alert whenever those readings goes beyond critical values. Pulse rate and body temperature readings are recorded over ThingSpeak and Google sheets so that patient health can be monitored from anywhere in the world over internet. A panic will also be attached so that patient can press it on emergency to send email/sms to their relatives.

In this project, we are monitoring various parameters of the patient using the internet of things. In the patient monitoring system based on the Internet of Things

project, the real-time parameters of a patient's health are sent to the cloud using Internet connectivity. These parameters are sent to a remote Internet location so that users can view these details from anywhere in the world.

In the IOT based system, details of the patient's health can be seen by many users.

To operate an IOT based health monitoring system project, you need a WiFi connection. The microcontroller or the Arduino board connects to the Wi-Fi network using a Wi-Fi module. This project will not work without a working WiFi network. You can create a WiFi zone using a WiFi module or you can even create a WiFi zone using Hotspot on your smartphone.

The Arduino UNO board continuously reads input from these 3 senses. Then it sends this data to the cloud by sending this data to a particular URL/IP address. Then this action of sending data to IP is repeated after a particular interval of time. For example in this project, we have sent data after every 30 seconds.

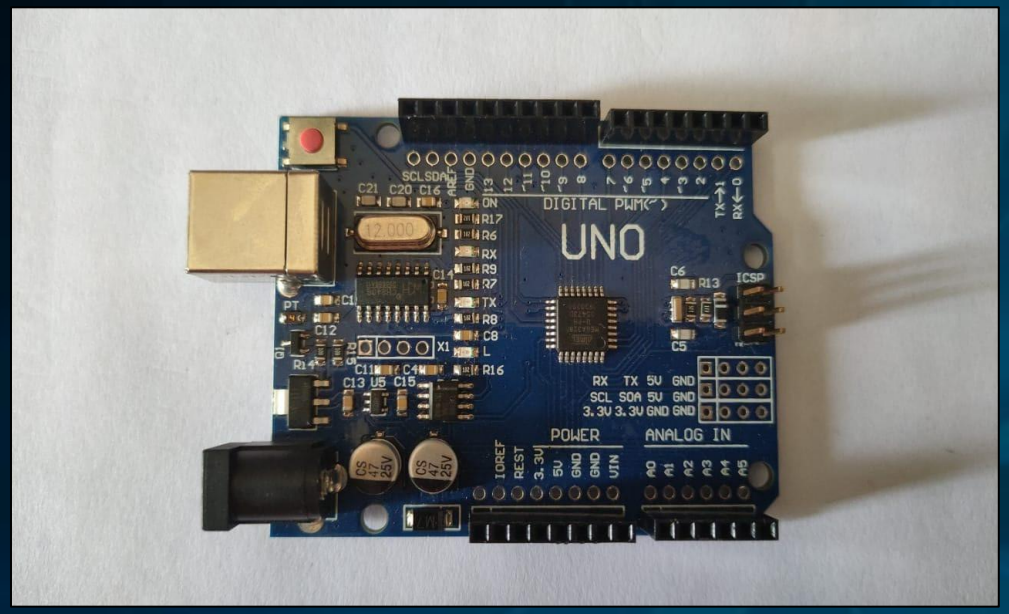
Arduino collects real time health data from pulse sensor which measures heartbeat in minutes or BPM (beats per minute). A digital temperature sensor connected to Arduino measures body temperature of the patient.

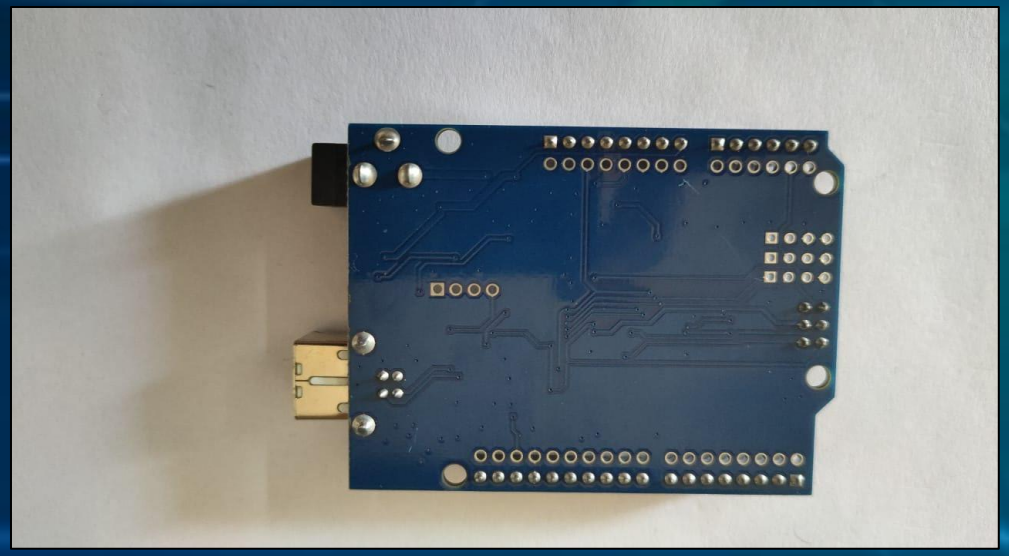
A buzzer produces auditory beeps when the patient's heartbeat occurs / detected. This gives a brief insight to a healthcare professional how a patient's heart is performing in a particular health condition. Abnormal heartbeats can be detected by just listening to the beeps.

A generic ESP8266 IoT module is connect to Arduino via UART, it is responsible for connecting the machine to internet and also for sending health data to a IoT server (Thing speak) for storing and monitoring.

This circuit is not only capable of sending patient's health data to a server but also can show real time data on a 16×2 LCD display. This is useful for a healthcare professional who is actively monitoring a patient on site.

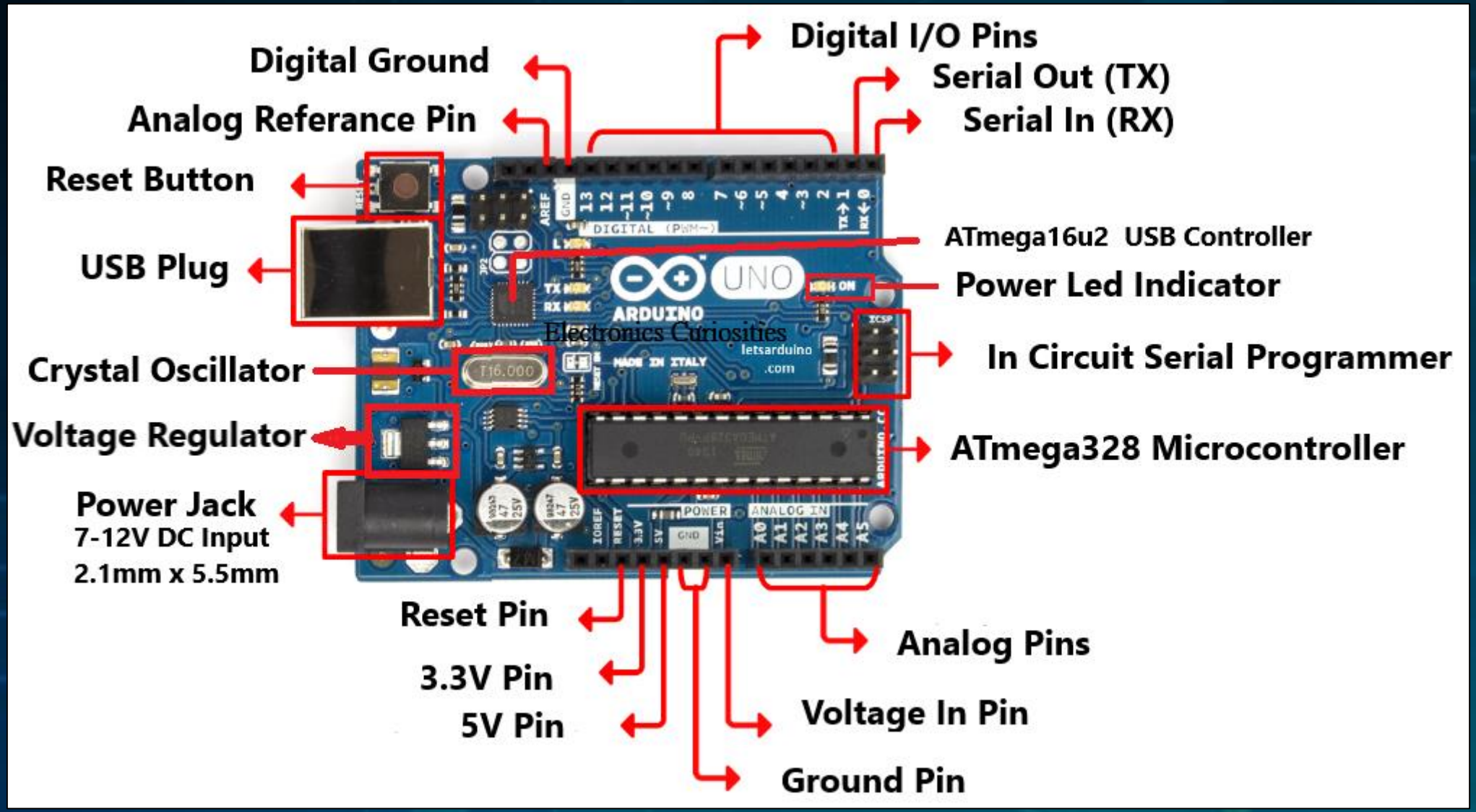
ARDUINO UNO



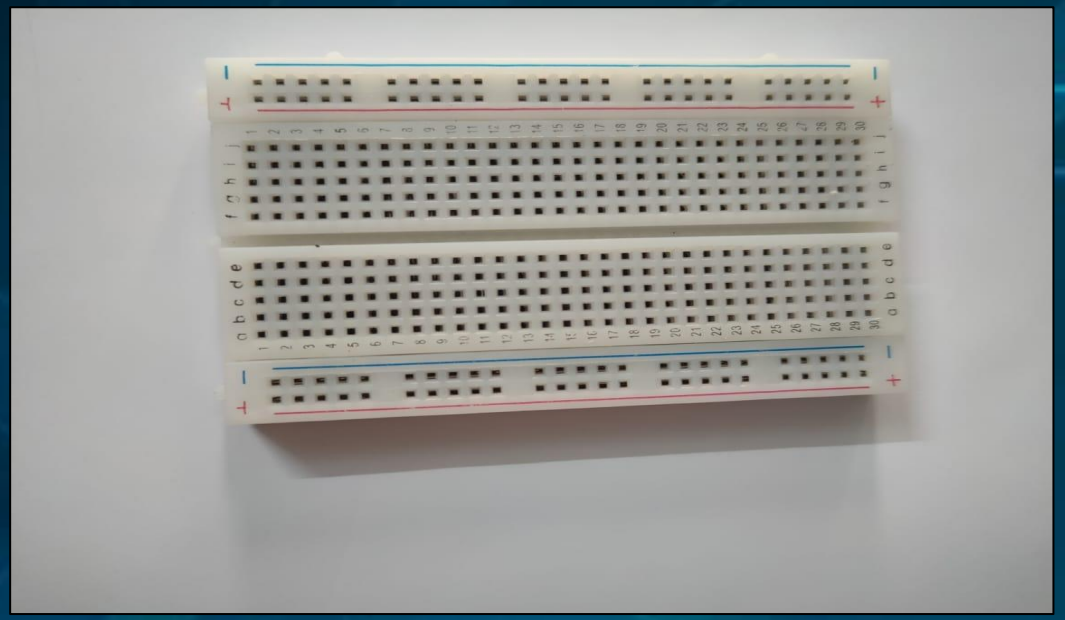


Arduino UNO is microcontroller based on ATmega328P. this board contain total 20 pins in which 14 are digital pins and 6 analogy pins. It is programmable with the Arduino IDE and the program is transfer in the Arduino by using the cable (type B). It also powered by the USB cable (type B) and it can also power by external power source.

It helps us to connect and control the all the components to sync with each other like it connect with sessor and get the data from it through I2C pin and show us the BPM and SpO2 (saturation value) as an output on the LCD. During connecting the oximeter with Arduino, we connect GND is connected to GND, VIN with 3V3, SCL to SCL and SDL to SDL.

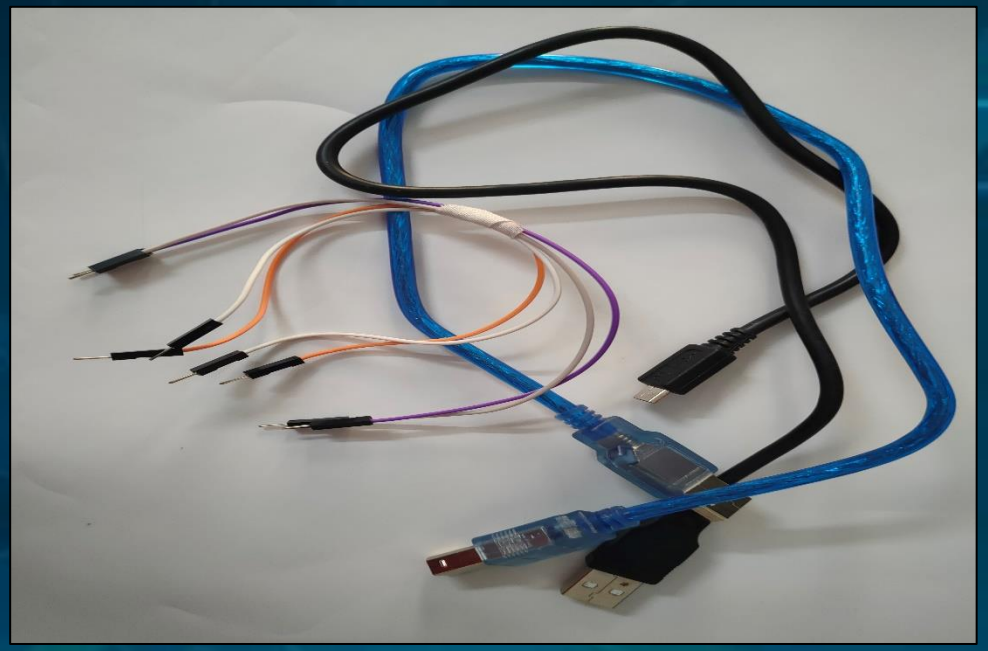

Digital Ground
Analog Referance Pin
Reset Button
USB Plug
Crystal Oscillator
Voltage Regulator
Power Jack
7-12V DC Input
2.1mm x 5.5mm
Digital I/O Pins
Serial Out (TX)
Serial In (RX)
ATmega16u2 USB Controller
Power Led Indicator
In Circuit Serial Programmer
ATmega328 Microcontroller
Reset Pin
3.3V Pin
5V Pin
Analog Pins
Voltage In Pin
Ground Pin
Electronics Curiosities
UNO
ARDUINO
DIGITAL (PWM~)
POWER
ANALOG IN

BREADBOARD



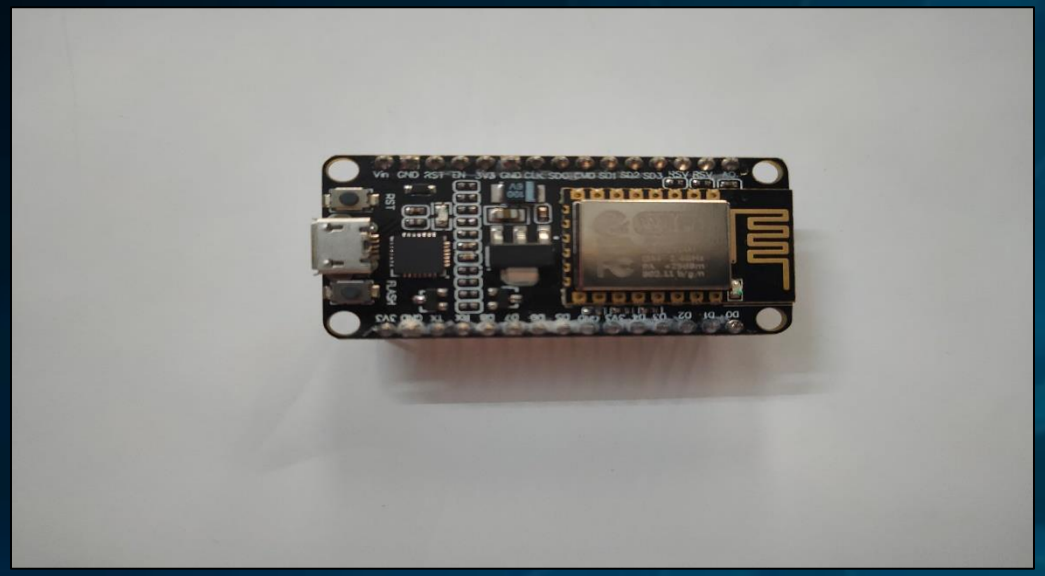
Breadboard is solderless device on which we make a temporary prototype device with electronic and test circuit design. Since the it is solderless device i.e., it does not require any soldering to connect the components, and this is one of the main reasons to use breadboard to make a prototype.

JUMPER WIRES



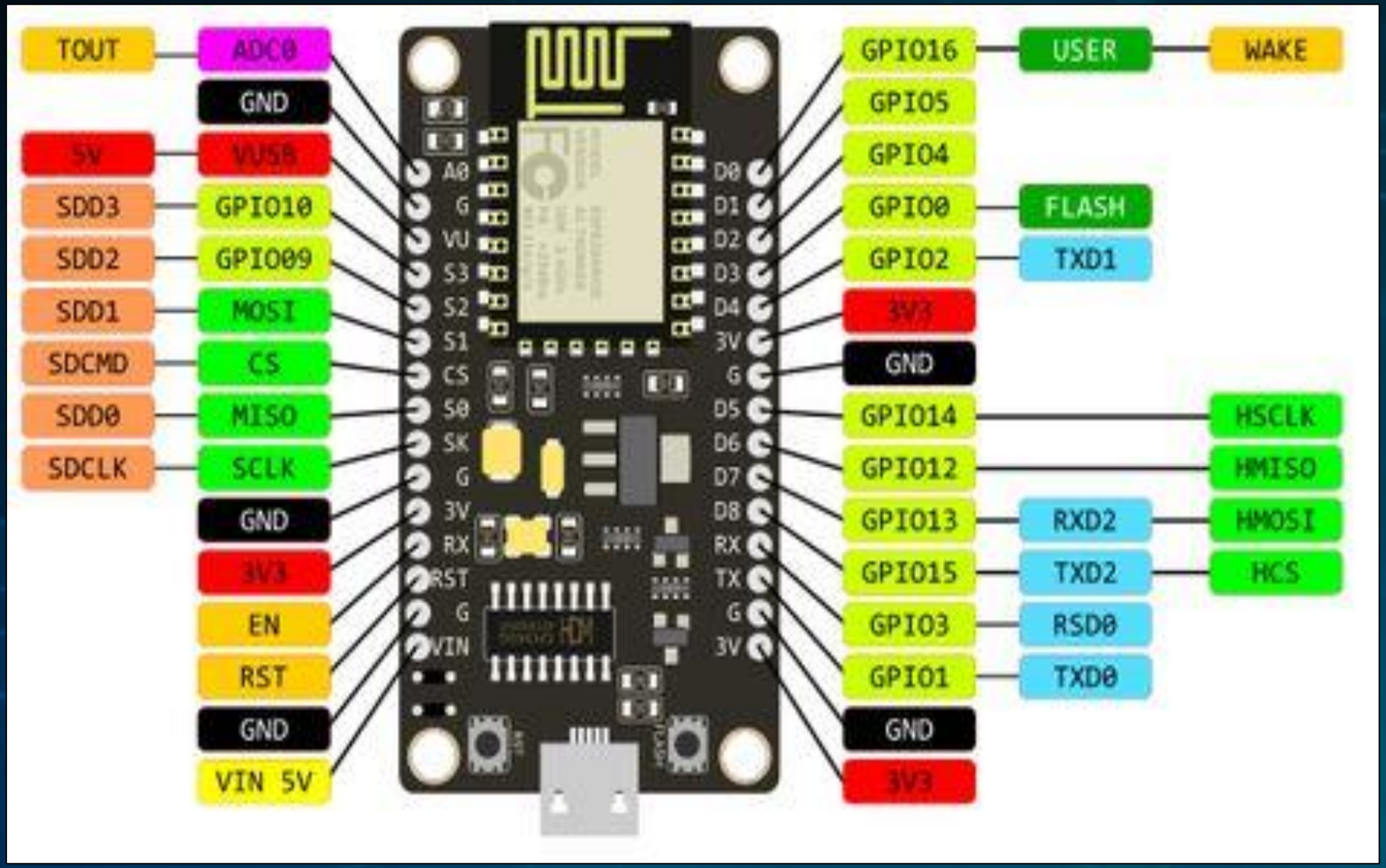
Jumper wire are the group of wire which are join in a form of a cable and each wire contain pin or connector at both the end. This type of wire is used to create prototype of any device because this wire does not require any type of soldering.

NODE MCU

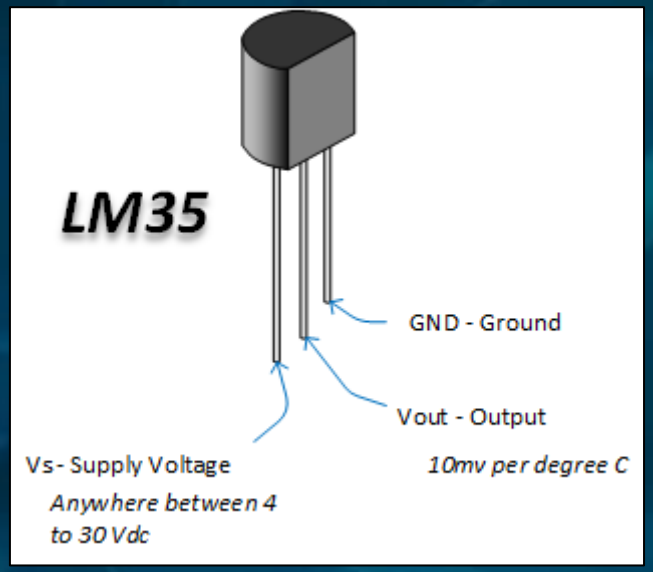


The NodeMCU (Node Micro Controller Unit) is an open-source software and hardware development environment built around an inexpensive System-on-a-Chip (SoC) called the ESP8266. The ESP8266, designed and manufactured by Espress if Systems, contains the crucial elements of a computer: CPU, RAM, networking (WiFi), and even a modern operating system and SDK. That makes it an excellent choice for the Internet of Things (IoT) projects of all kinds.

However, as a chip, the ESP8266 is also hard to access and use. You must solder wires, with the appropriate analog voltage, to its pins for the simplest tasks such as powering it on or sending a keystroke to the “computer” on the chip. You also have to program it in low-level machine instructions that can be interpreted by the chip hardware. This level of integration is not a problem using the ESP8266 as an embedded controller chip in mass-produced electronics. It is a huge burden for hobbyists, hackers, or students who want to experiment with it in their own IoT projects.

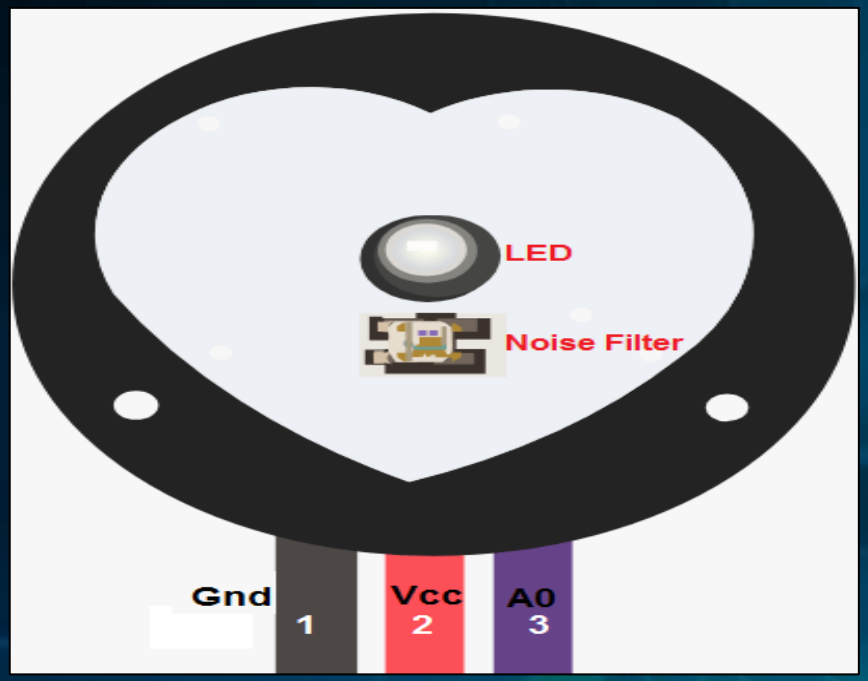

TOUT
ADC0
GND
5V
VUSB
SDD3
GPIO10
SDD2
GPIO09
SDD1
MOSI
SDCMD
CS
SDD0
MISO
SDCLK
SCLK
GND
3V3
EN
RST
GND
VIN 5V
A0
G
VU
S3
S2
S1
CS
S0
SK
G
3V
RX
RST
G
VIN
D0
D1
D2
D3
D4
3V
G
D5
D6
D7
D8
RX
TX
G
3V
GPIO16
USER
WAKE
GPIO5
GPIO4
GPIO0
FLASH
GPIO2
TXD1
3V3
GND
GPIO14
HSCLK
GPIO12
HMISO
GPIO13
RXD2
HMOSI
GPIO15
TXD2
HCS
GPIO3
RSD0
GPIO1
TXD0
GND
3V3

TEMPERATURE SENSOR (LM35)



LM35 is a analog linear temperature sensor. Its output is proportional to the temperature (in degree Celsius). The operating temperature range is from -55°C to 150°C. The output voltage varies by 10mV in response to every °C rise or fall in temperature. It can be operated from a 5V as well as 3.3 V supply and the stand by current is less than 60uA.

PULSE SENSOR



Pulse Sensor is a well-designed plug-and-play heart-rate sensor for Arduino. The sensor clips onto a fingertip or earlobe and plugs right into Arduino. It also includes an open-source monitoring app that graphs your pulse in real time.

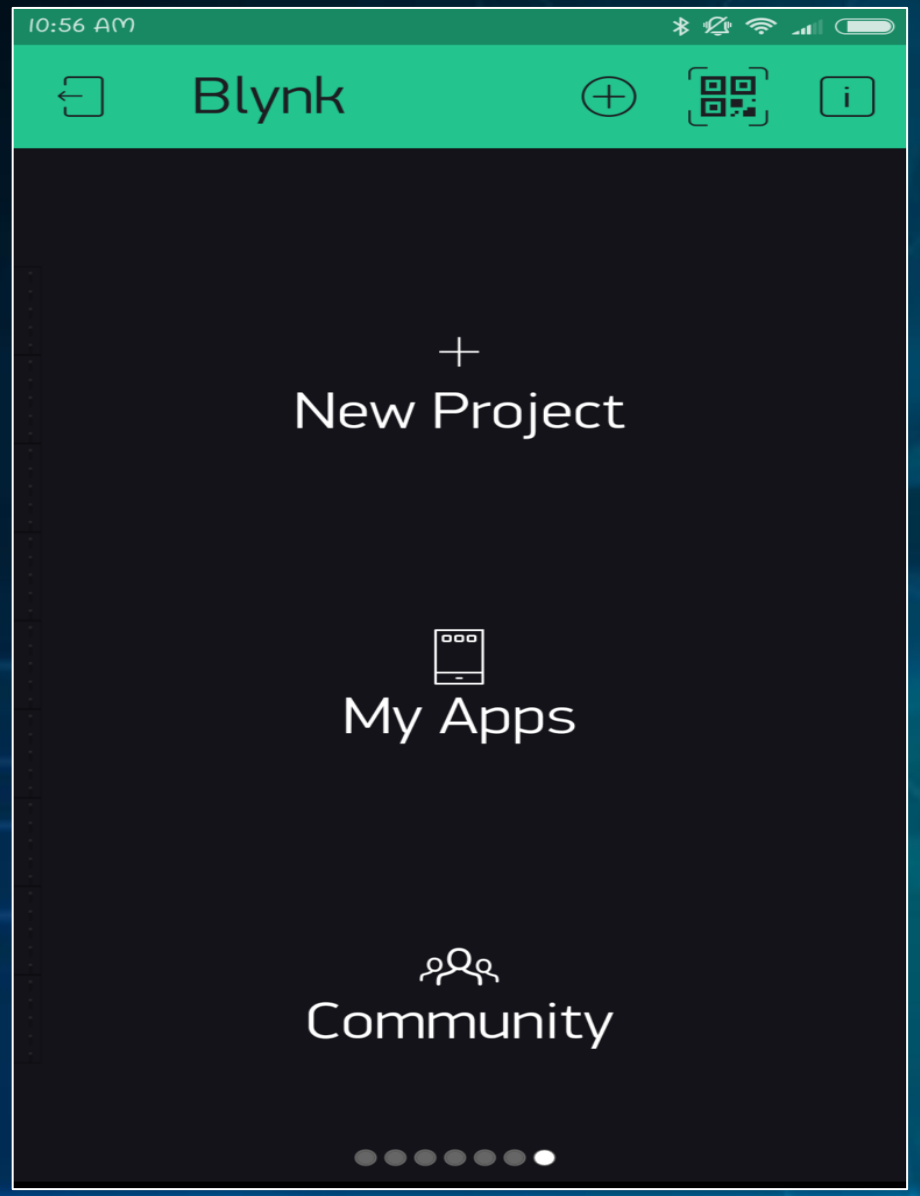
The front of the sensor is the covered with the Heart shape logo. This is the side that makes contact with the skin. On the front you see a small round hole, which is where the LED shines throug -h from the back, and there is also a little square is an ambient light sensor square is an ambient light sensor, exactly like the one used in cellphones, tablets, and laptops, to adjust the screen brightness in different light conditions. The LED shines light into the fingertip or earlobe, or other capillary tissue, and sensor reads the amount of light that bounces back. That's how it calculates the heart rate. The other side of the sensor is where the rest of the parts are mounted.

Before we use this sensor, we need to protect the exposed side of the sensor so that we can get accurate readings and avoid the short circuit due to sweat.

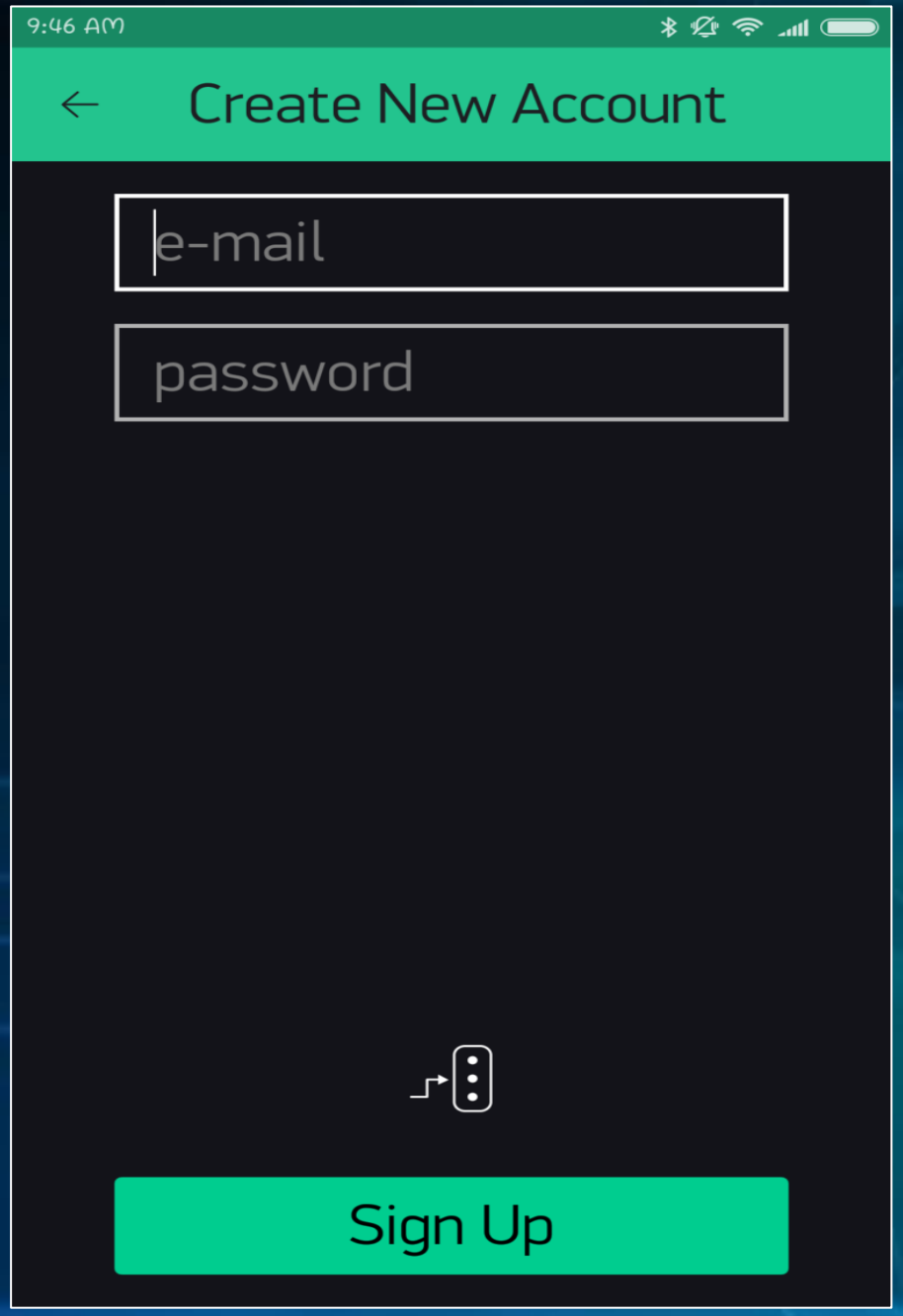
There are three wires coming out of the sensor, Signal(S), Vcc(3 - 5 V) and GND.

Signal wire is connected to Arduino Analog pin.

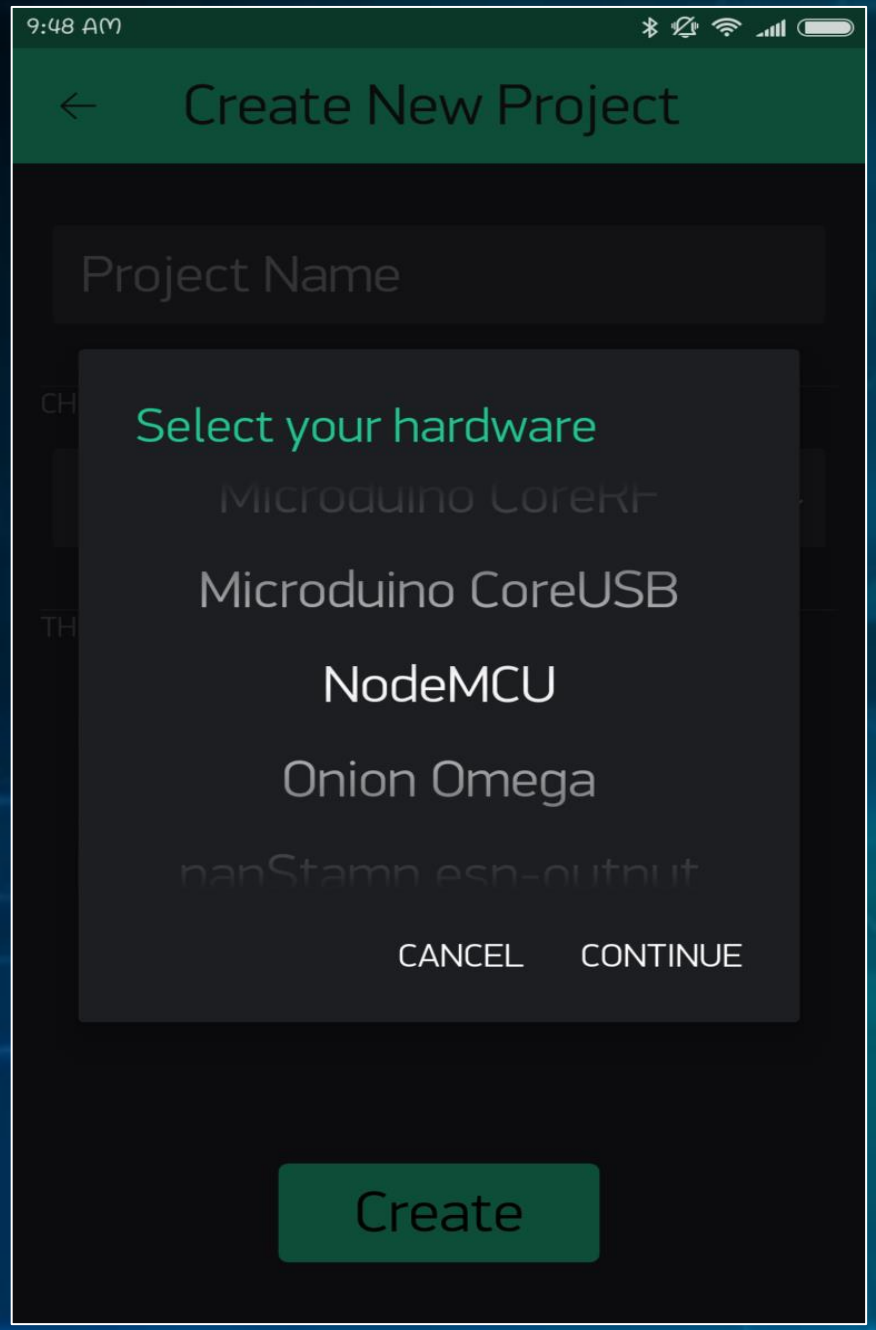
BLYNK APP



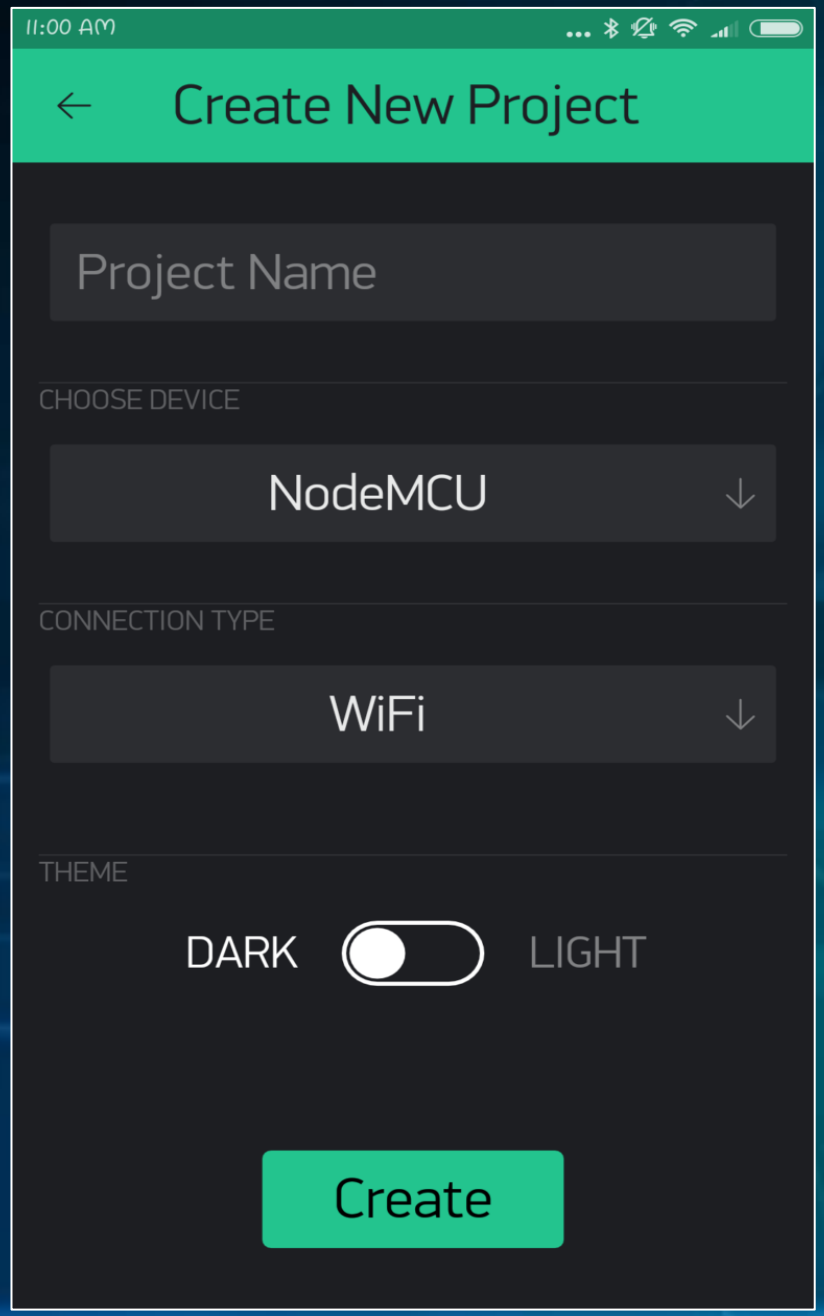
Blynk is a Platform with IOS and Android apps to control Arduino, Raspberry Pi and the likes over the Internet. It's a digital dashboard where you can build a graphic interface for your project by simply dragging and dropping widgets.



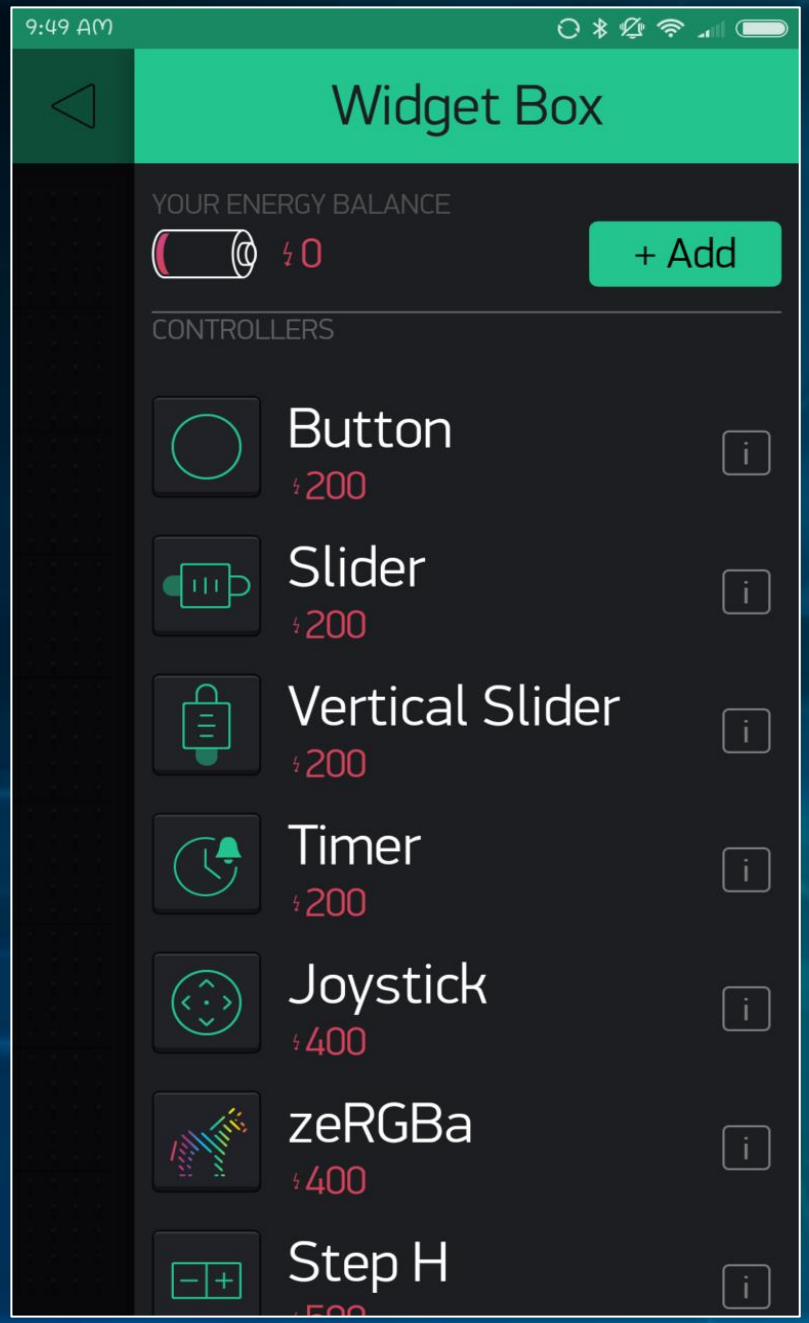
- Blynk application can be found from the following links:
  - Android Blynk App
  - IOS Blynk App
- After downloading the app, create an account and log in. (If possible than log in with your real mail id for better connectivity later.)
- You'll also need to install the **Blynk Arduino Library**, which helps generate the firmware running on your ESP8266. Download the latest release from https://github.com/blynkkk/blynk-library/releases , and follow along with the directions there to install the required libraries.



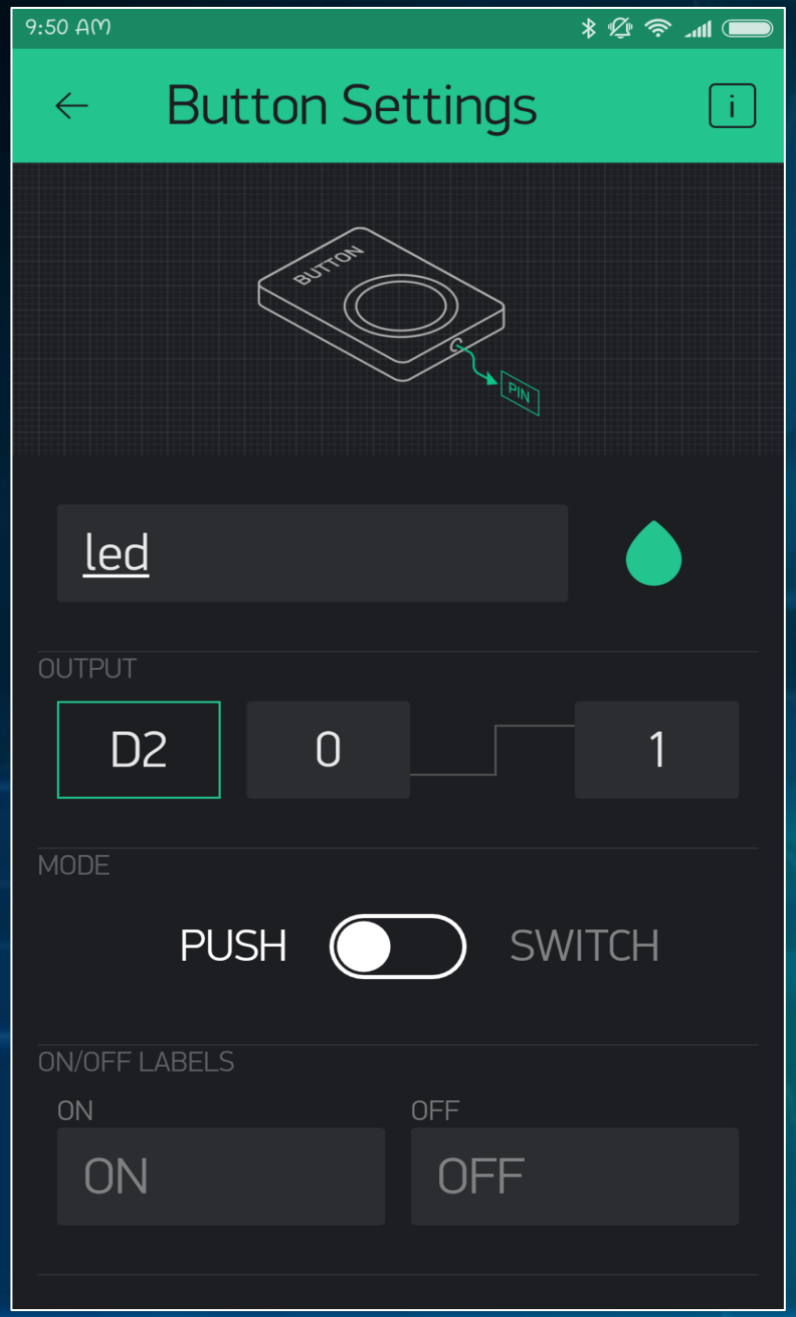
- Click the "Create New Project" in the app to create a new Blynk app. Give it any name.
- Blynk works with hundreds of hardware models and connection types. Select the Hardware type. After this, select connection type. In this project we have select WiFi connectivity.



- The ***Auth Token*** is very important – you'll need to stick it into your ESP8266's firmware. For now, copy it down or use the "E-mail" button to send it to yourself.

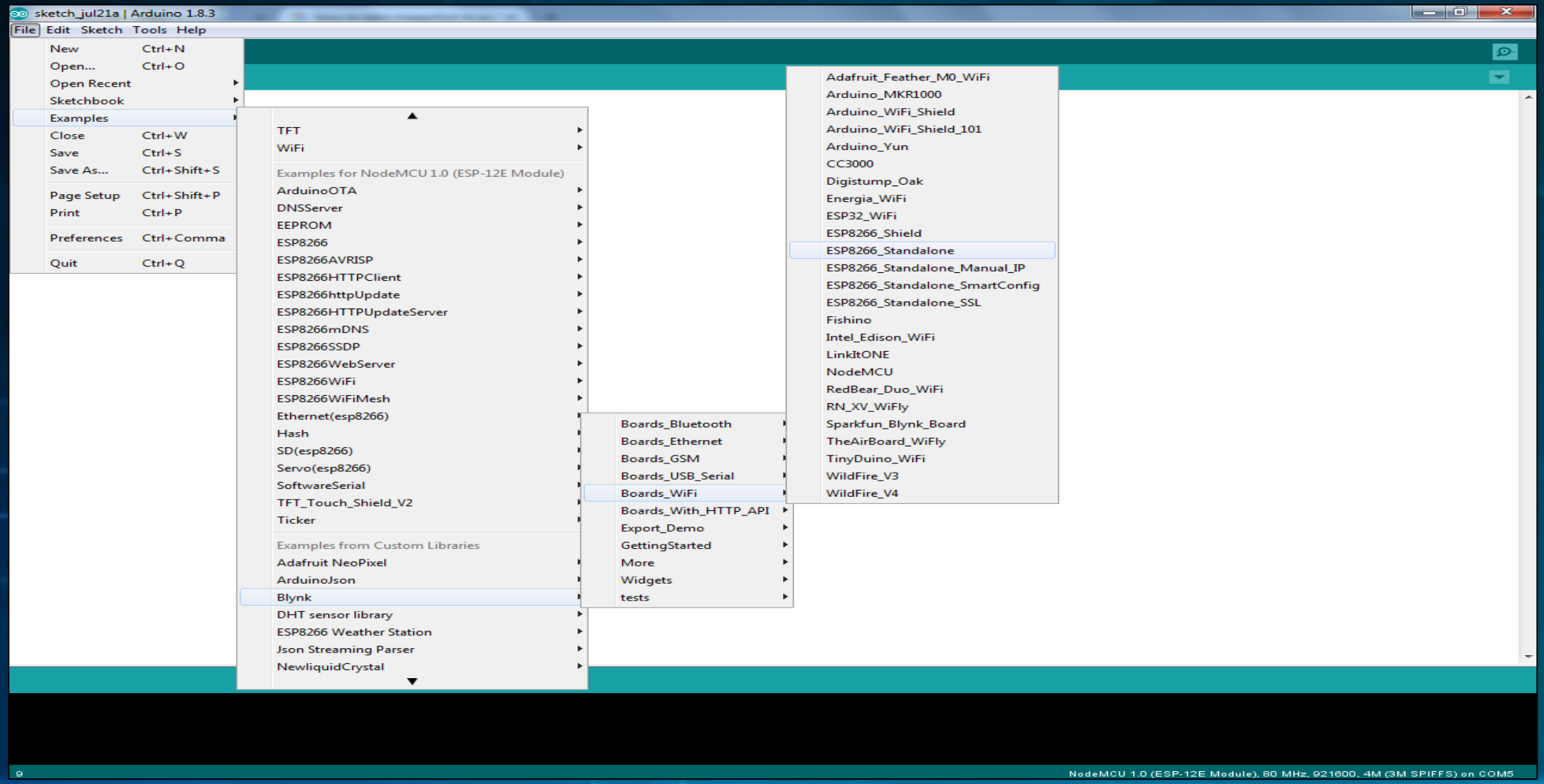


- Then you'll be presented with a blank new project. To open the widget box, click in the project window to open.
- We are selecting a button to control Led connected with NodeMCU.
  - Click on Button.
  - Give name to Button say led.
  - Under OUTPUT tab- Click pin and select the pin to which led is connected to NodeMCU, here it is digital pin 2, hence select digital and under pin D2. And Click continue.



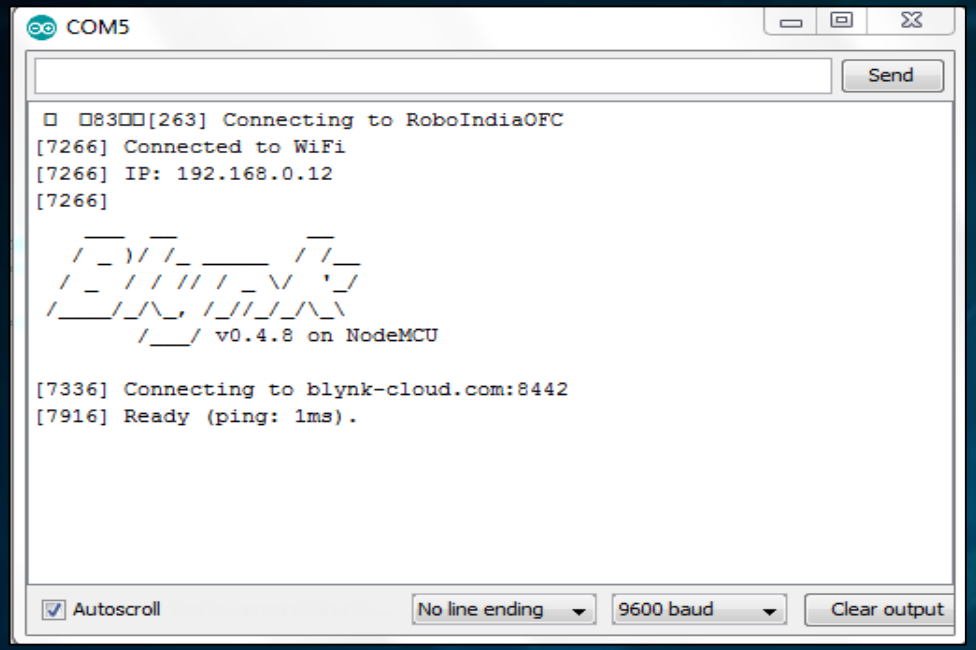
- Under MODE tab- Select whether you want this button as “push button” or “Switch”.
- You have successfully created a GUI for Arduino.

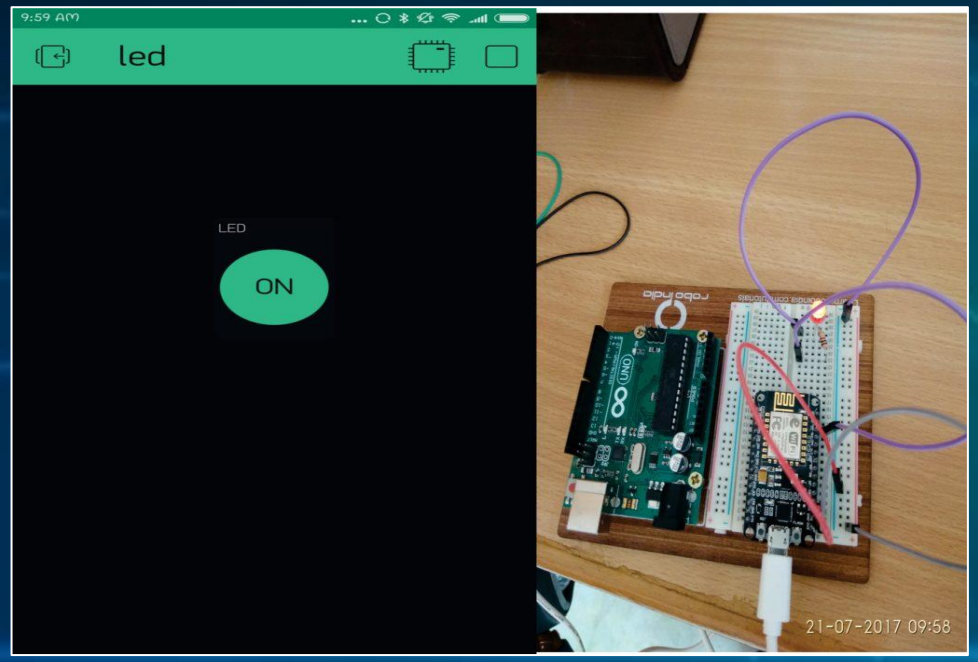
- Now that your Blynk project is set-up, open Arduino and navigate to the ESP8266_Standalone example in the File > Examples > Blynk > Boards_WiFi> ESP8266_Standalone menu.



```
#define BLYNK_PRINT Serial


#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "YourAuthToken";

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "YourNetworkName";
char pass[] = "YourPassword";

void setup()
{
  // Debug console
  Serial.begin(9600);

  Blynk.begin(auth, ssid, pass);
}

void loop()
{
  Blynk.run();
}
```

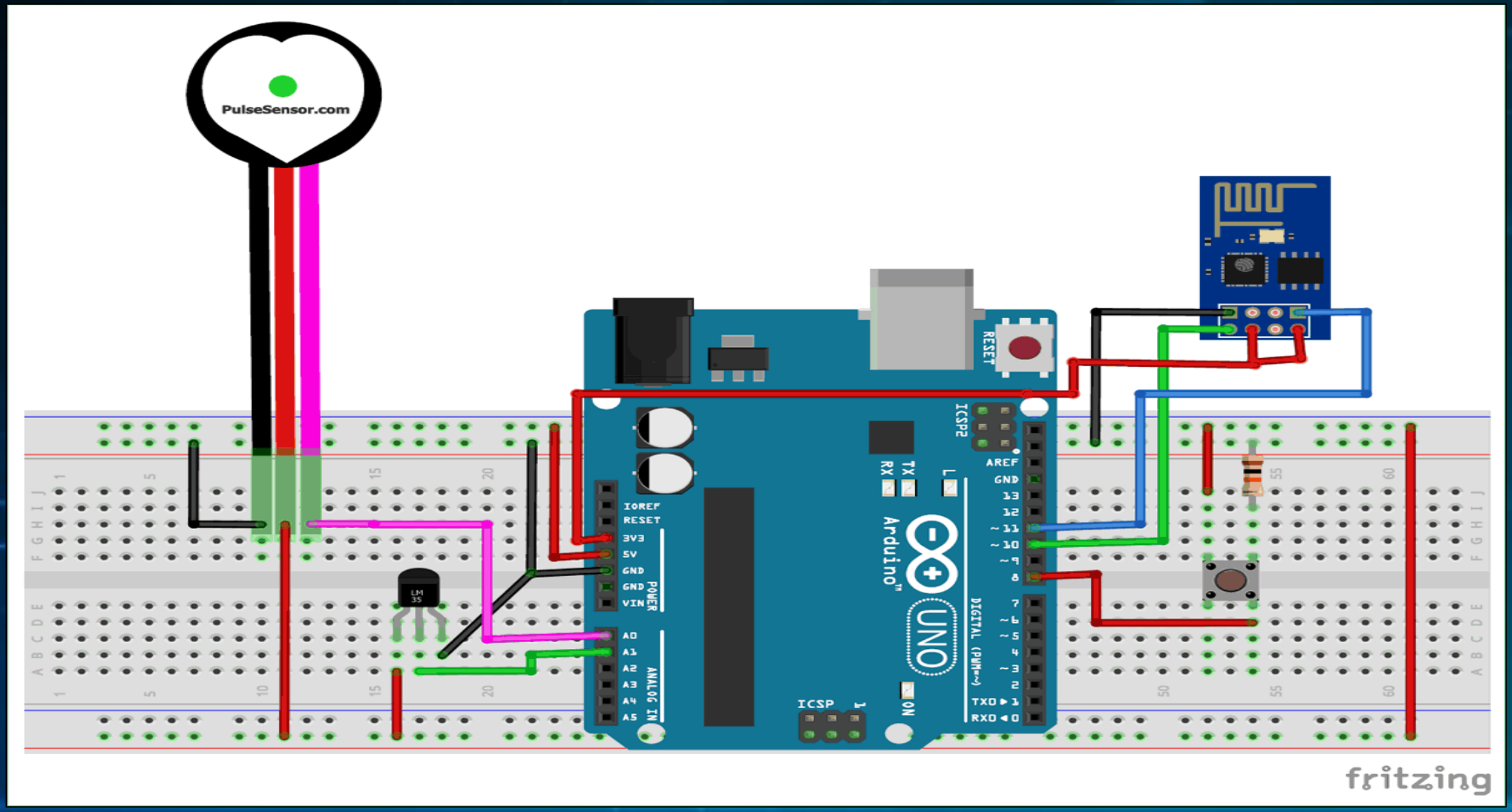
- Before uploading, make sure to paste your authorization token into the auth [] variable. Also make sure to load your Wifi network settings into the Blynk.begin(auth, “ssid”, “pass”) function.





- After the app has uploaded, open the serial monitor, setting the baud rate to 9600. Wait for the "Ready" message.
- Then click the "Run" button in the top right corner of the Blynk app. Press the button and watch the LED.
- Then add more widgets to the project. They should immediately work on the ESP8266 without uploading any new firmware.

CIRCUIT DIAGRAM



PROGRAMMING

ARDUINO CODE:

```
#include <SoftwareSerial.h>
#define USE_ARDUINO_INTERRUPTS true
#include <PulseSensorPlayground.h>
float temp;
int tempPin = 1;
int sampleTime = 1000;
String cdata;
const int PulseWire = 0;
int Threshold = 550;
PulseSensorPlayground pulseSensor;
SoftwareSerial nodemcu(2,3);
void setup() {
    Serial.begin(9600);
    nodemcu.begin(9600);
```

```
pulseSensor.analogInput(PulseWire);
  pulseSensor.setThreshold(Threshold);


   if (pulseSensor.begin()) {
    Serial.println("We created a smart health monitoring system !");
  }
}


void loop() {
                                    //
************************************************************ PULSE RATE
*****************************************
 int myBPM = pulseSensor.getBeatsPerMinute();
```

```
if (pulseSensor.sawStartOfBeat()) {

}
                    //
**********************************************
TEMPERATURE ***********************************
  temp = analogRead(tempPin);
  Serial.print("RAW DATA: ");
  Serial.print (temp);
  Serial.println(" ");
  float tempc = temp * 0.48828125;
  Serial.print("CELSIUS: ");
  Serial.print(tempc);
  Serial.println("*C ");
  float temps = tempc *9 / 5;
float tempf = temps + 32;
Serial.print("FAHRENHEIT: ");
```

```
Serial.print(tempf);
Serial.println("*F");
Serial.print("BPM: ");
Serial.println(myBPM);
delay(sampleTime);
                    //**********************************************************
****DATA
TRANSFER**********************************************************

cdata=cdata+tempc+','+tempf+','+myBPM+':';
nodemcu.print(cdata);
Serial.println(cdata);
cdata="";
delay(sampleTime);
}
```

NODEMCU CODE:

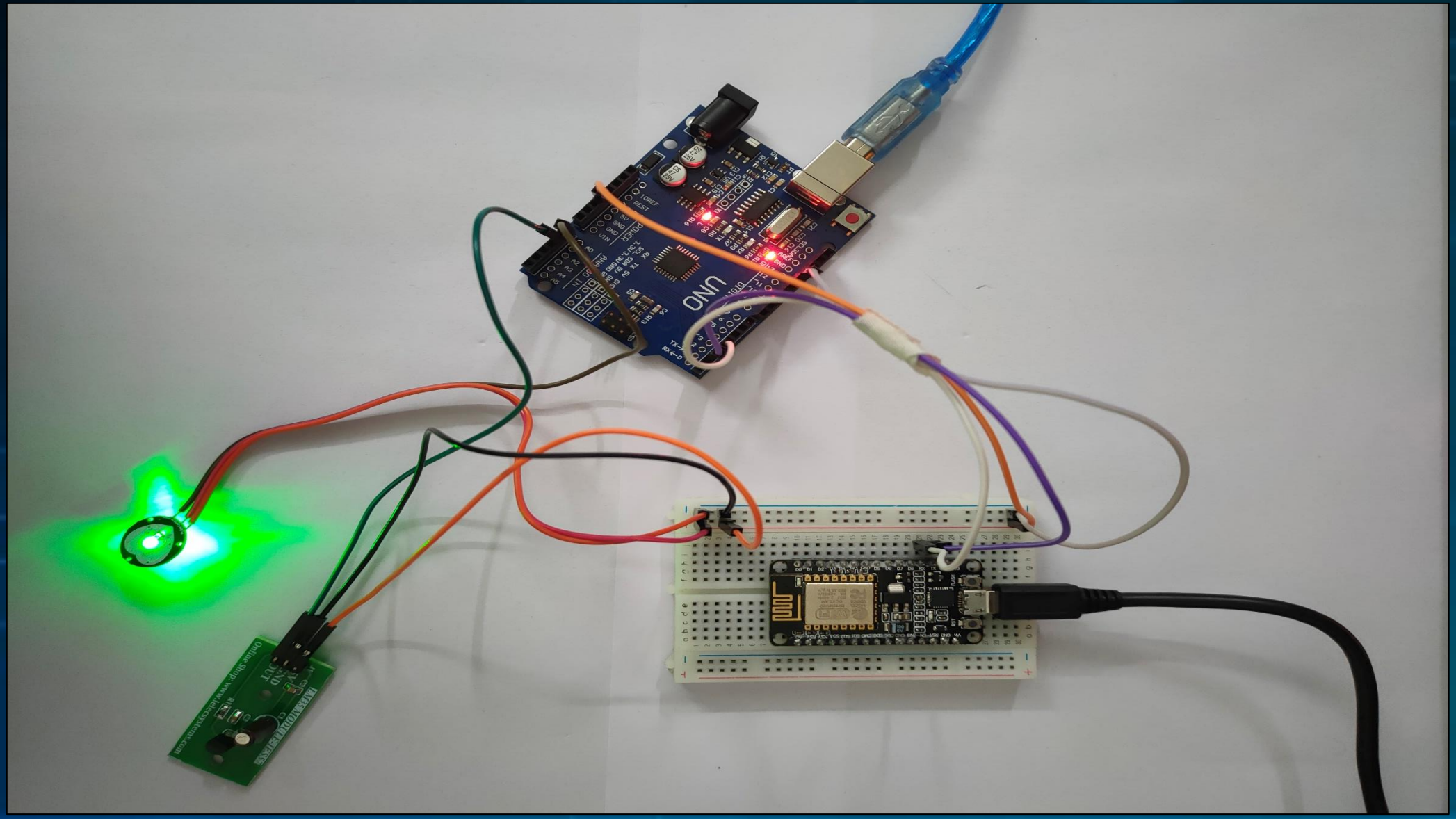
```
#include <SimpleTimer.h>
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <SoftwareSerial.h>
char auth[] = "g2t-Vz_OQXiqrYz8giWopSCO_1hDIP1R";
char ssid[] = "zzz";
char pass[] = "9245545900";
String myString,data1,data2,data3;
WidgetLCD lcd(V5);
SimpleTimer timer;
void myTimerEvent()
{
  Blynk.virtualWrite(V1, millis() / 1000);
}
String getValue(String data, char separator, int index)
```

```
{
    int found = 0;
    int strIndex[] = { 0, -1 };
    int maxIndex = data.length() - 1;

    for (int i = 0; i <= maxIndex && found <= index; i++) {
        if (data.charAt(i) == separator || i == maxIndex) {
            found++;
            strIndex[0] = strIndex[1] + 1;
            strIndex[1] = (i == maxIndex) ? i+1 : i;
        }
    }
    return found > index ? data.substring(strIndex[0], strIndex[1]) : "";
}
void setup()
{
  Serial.begin(9600);
```

```
 Blynk.begin(auth, ssid, pass);
  while(!Serial){
    ;
  }
}
void loop()
{

  if (Serial.available()) {
    char r = Serial.read();
    myString = myString + r;
    if (r==':'){
      Serial.print("Temperature in Celcius: ");
      Serial.println(data1);
      Serial.print("Temperature in Fahrenheit: ");
      Serial.println(data2);
      Serial.print("BPM: ");
```

```
Serial.println(data3);
      myString="";
      Serial.println(data1);
      Serial.println(data2);
      Serial.println(data3);
    }
    }

    Blynk.virtualWrite(V3,data1);
    Blynk.virtualWrite(V4,data2);
    lcd.clear();
    lcd.print(0,0,"BPM: ");
    lcd.print(11,0,data3);
  }
```

WIRING OF REAL HARDWARE



USER AND MOBILE INTERFACE

```
[1065] Connected to WiFi
[1065] IP: 192.168.1.8
[1066]

    ____  __            __
   / _ )/ /_ _____  / /__
  / _  / / // / _ \/  '_/
 /____/_/\_, /_//_/_/\_\
        /___/ v0.6.1 on NodeMCU

[1140] Connecting to blynk-cloud.com:80
[1306] Ready (ping: 51ms).
Temperature in Celcius: 32.23
Temperature in Fahrenheit: 90.01
BPM: 101:
Temperature in Celcius: 31.25
Temperature in Fahrenheit: 88.25
BPM: 0:
Temperature in Celcius: 31.25
Temperature in Fahrenheit: 88.25
BPM: 0:
Temperature in Celcius: 30.76
Temperature in Fahrenheit: 87.37
BPM: 0:
Temperature in Celcius: 32.23
Temperature in Fahrenheit: 90.01
BPM: 39:
Temperature in Celcius: 31.25
Temperature in Fahrenheit: 88.25
BPM: 36:
Temperature in Celcius: 33.20
Temperature in Fahrenheit: 91.77
BPM: 37:
Temperature in Celcius: 31.25
```

ADVANTAGES

- IOT Monitoring proves really helpful when we need to monitor & record and keep track of changes in the health parameters of the patient over a period of time. So with the IOT health monitoring, we can have the database of these changes in the health parameters. Doctors can take the reference of these changes or the history of the patient while suggesting the treatment or the medicines to the patient.
- Hospital stays are minimized due to Remote Patient Monitoring.
- Hospital visits for normal routine checkups are Minimized.
- Patient health parameter data is stored over the cloud. So it is more beneficial than maintaining the records on printed papers kept in the files. Or even the digital records which are kept in a particular computer or laptop or memory device like a pen- drive. Because there are chances that these devices can get corrupt and data might be lost. Whereas, in the case of IOT, cloud storage is more reliable and does have minimal chances of data loss.

REFERENCES

- https://www.vervetronics.com/solutions/iot-smart-healthcaremedical/patient-monitoring/
- https://circuitdigest.com/microcontroller-projects/iot-based-patient-monitoring-system-using-esp8266-and-arduino
- Arduino Temperature Sensor (LM35) : 4 Steps – Instructables
- https://www.sparkfun.com/products/11574
- https://elearn.ellak.gr/mod/book/view.php?id=2326#:~:text=The%20NodeMCU%20(Node%20MicroController%20Unit,(SoC)%20called%20the%20ESP8266
- https://roboindia.com/tutorials/blynk-introduction-nodemcu/
- https://electronics-project-hub.com/iot-based-health-monitoring-system-arduino/

- https://www.ijert.org/smart-health-monitoring-system-using-iot#:~:text=IOT%20Monitoring
- http://wiring.org.co/learning/tutorials/breadboard/#:~:text=A%20breadboard%
- https://www.arduino.cc/en/pmwiki.php?n=Main/arduinoBoardUno

VIDEO LINK

- https://youtu.be/8S8IRNxngEM

THANK YOU