PRACTICAL: 1

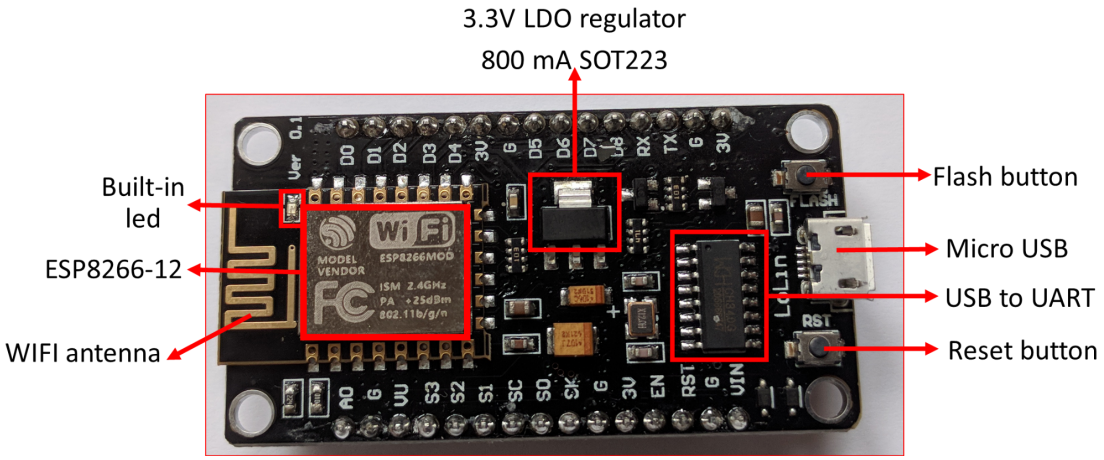
Aim : Getting started with NodeMCU, Arduino with ESP8266 and ESP32 in the Arduino IDE

Theory :

What is NodeMCU

NodeMCU is a low-cost open source IoT platform. It initially included firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which was based on the ESP-12 module. Later, support for the ESP32 32-bit MCU was added.

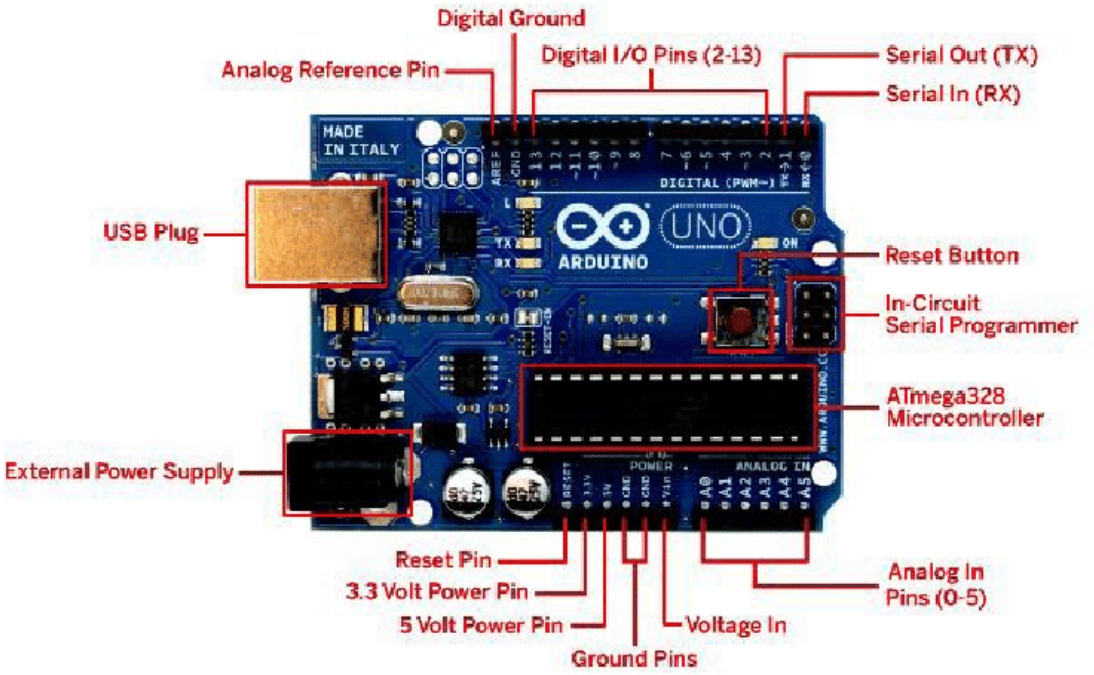
NodeMCU Architecture



What is Arduino

Arduino is an open-source hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices.

Arduino Architecture



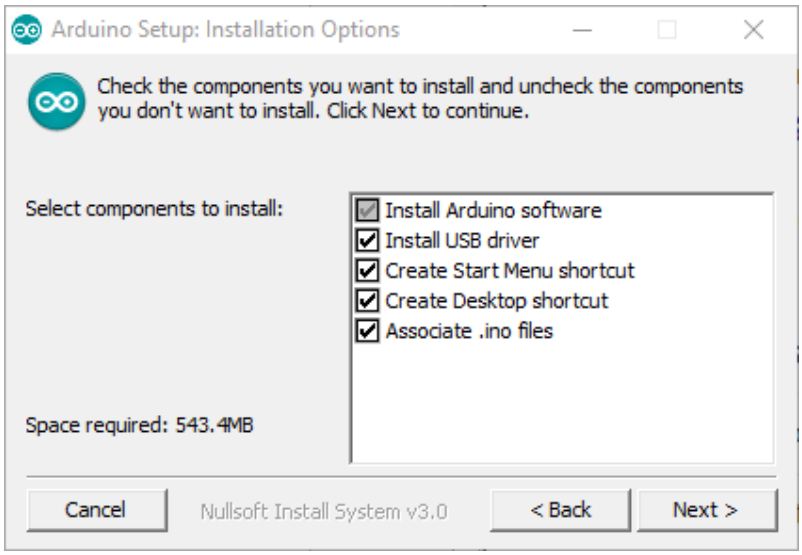
Installation of Arduino IDE

Step 1 : Visit https://www.arduino.cc/en/software

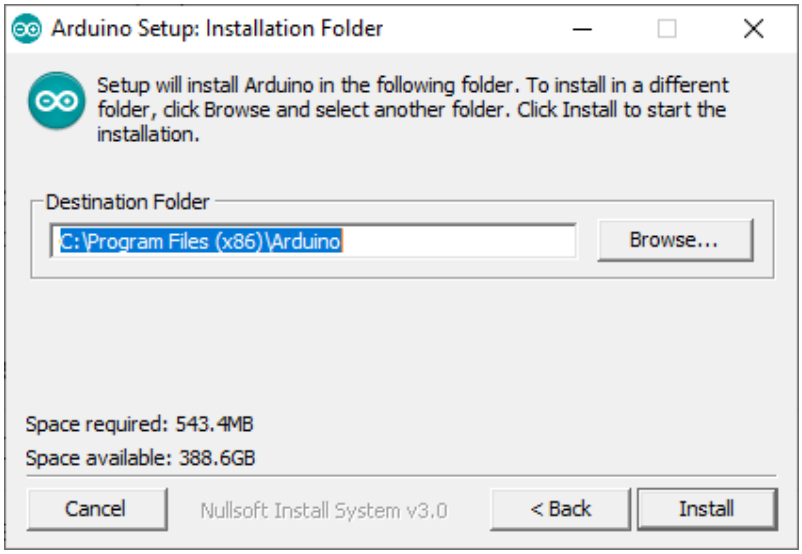
Step 2 : Click on Windows Installer, for Windows 7 and up, just download.

Step 3 : Run and install step

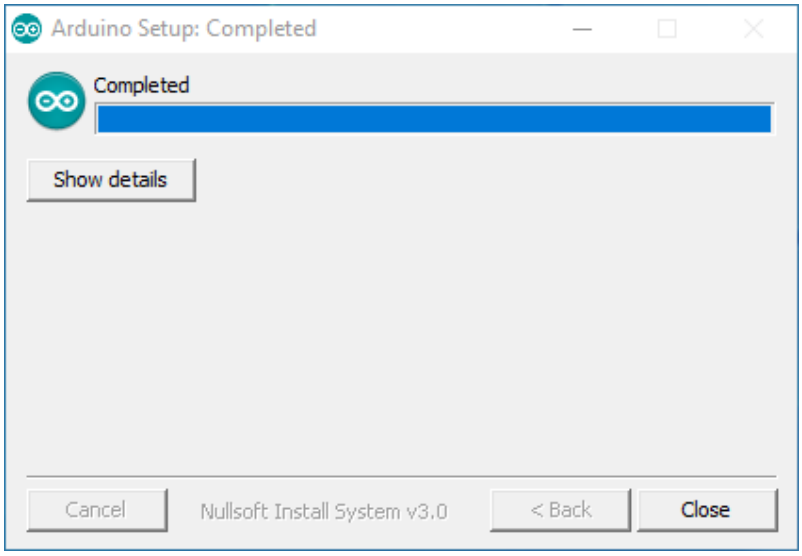
Step 4: Click Next



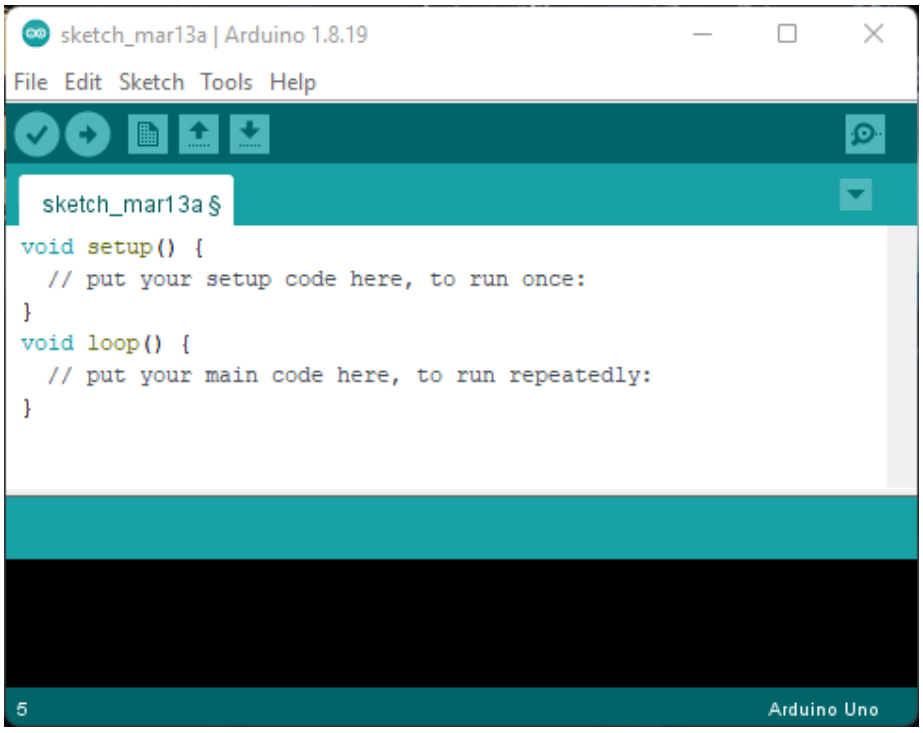
Step 5: Click Install



Step 6: Click Close

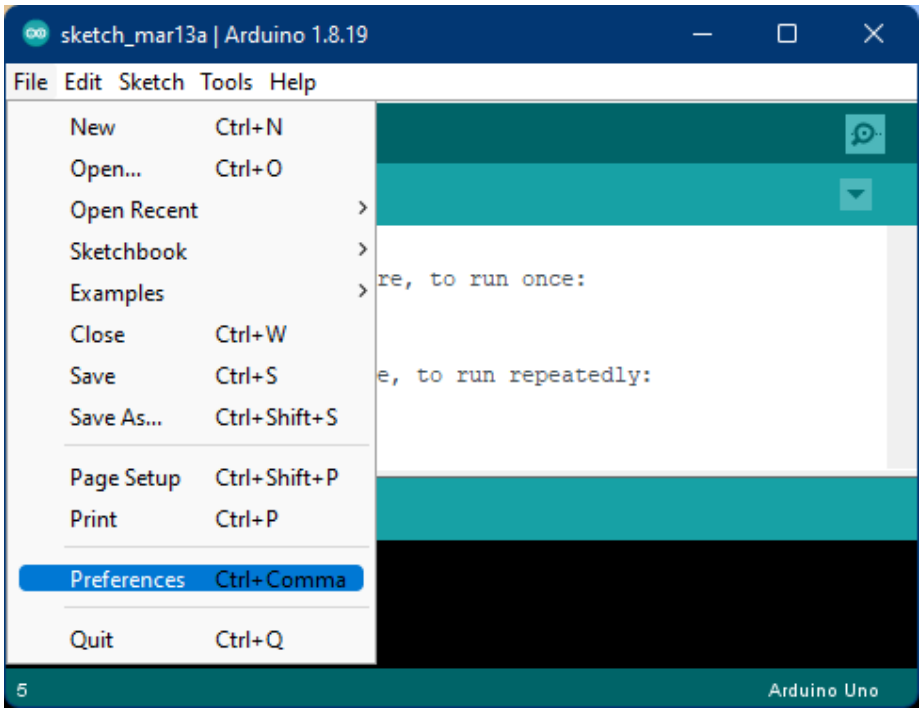


Step 7 : Run Arduino IDE

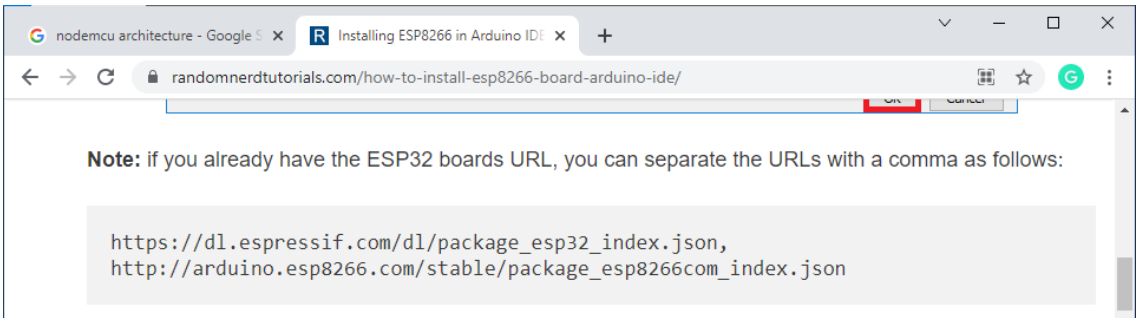


Installation of ESP8266 NodeMCU board manager in Arduino IDE

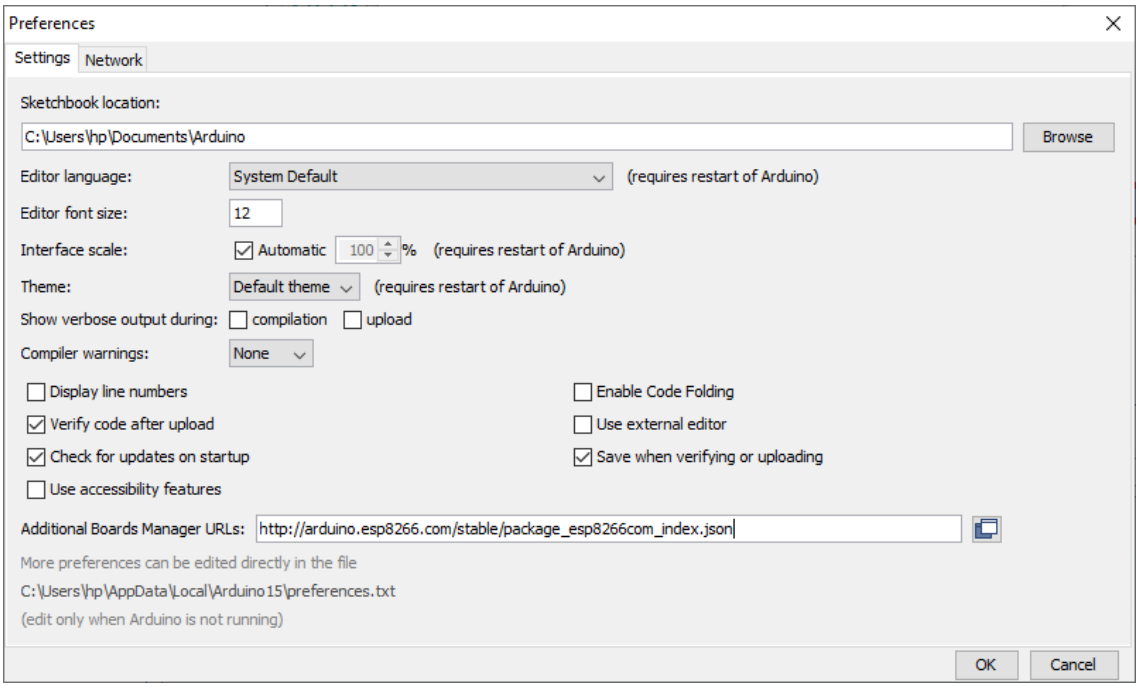
Step 1 : Goto File -> Preferences



Step 2 : Visit https://randomnerdtutorials.com/how-to-install-esp8266-board-arduino-ide/

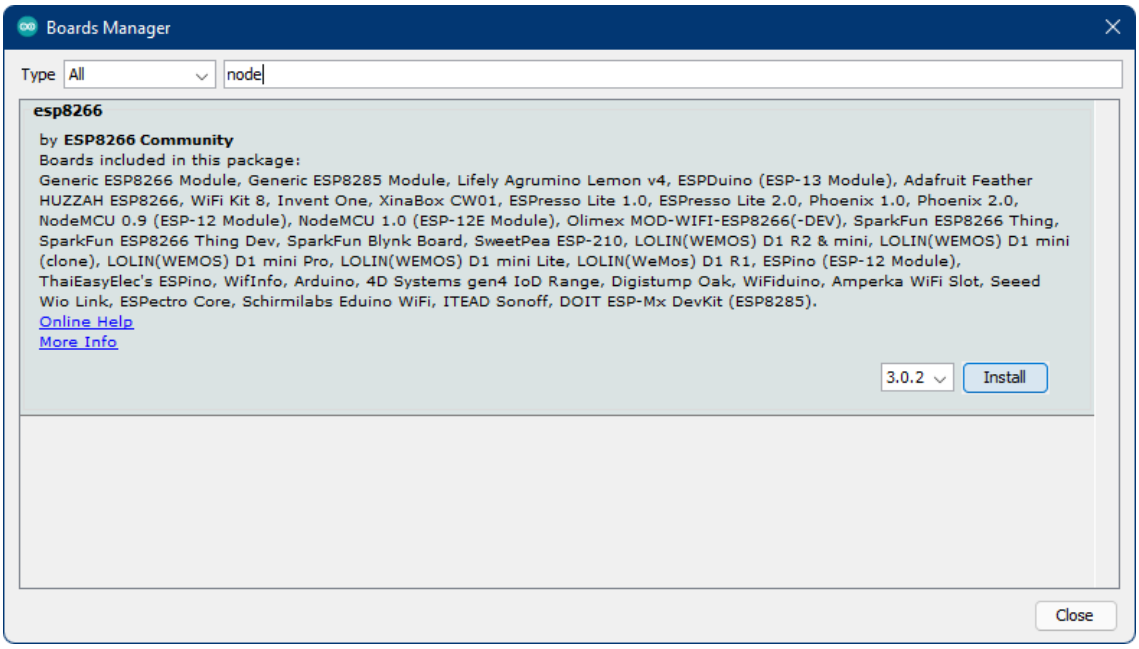


Step 3 : Copy "https://dl.espressif.com/dl/package_esp32_index.json ,http://arduino.esp8266.com/stable/package_esp8266com_index.json" and paste in 'Additional Board Manager URls Box'

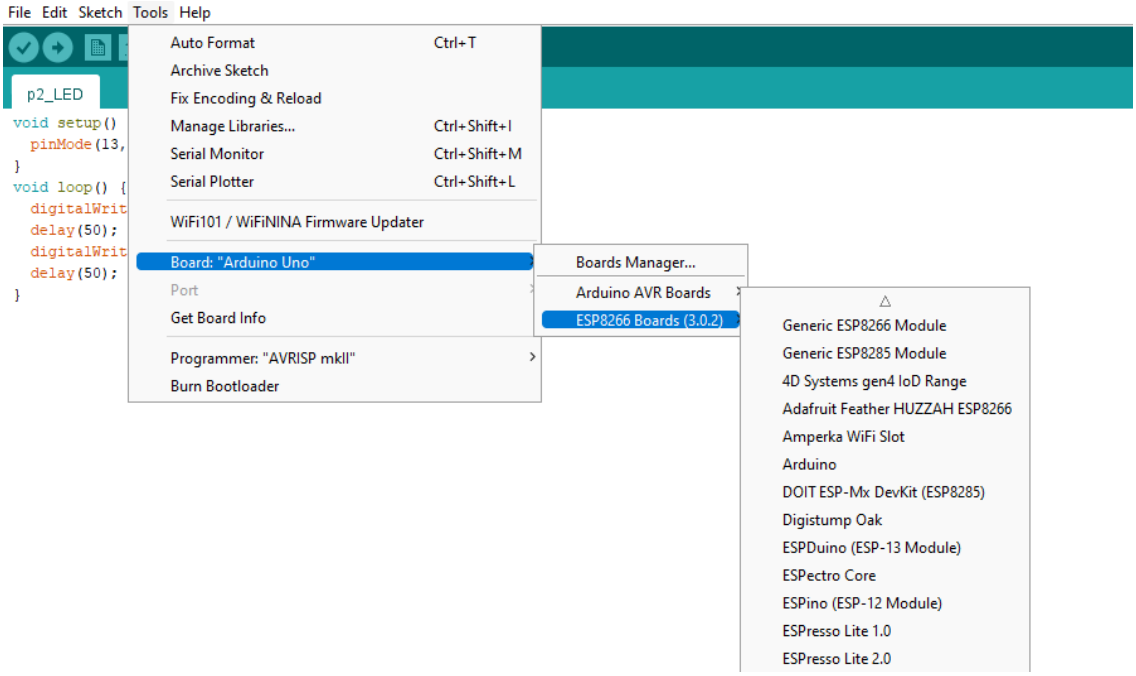


And click ok.

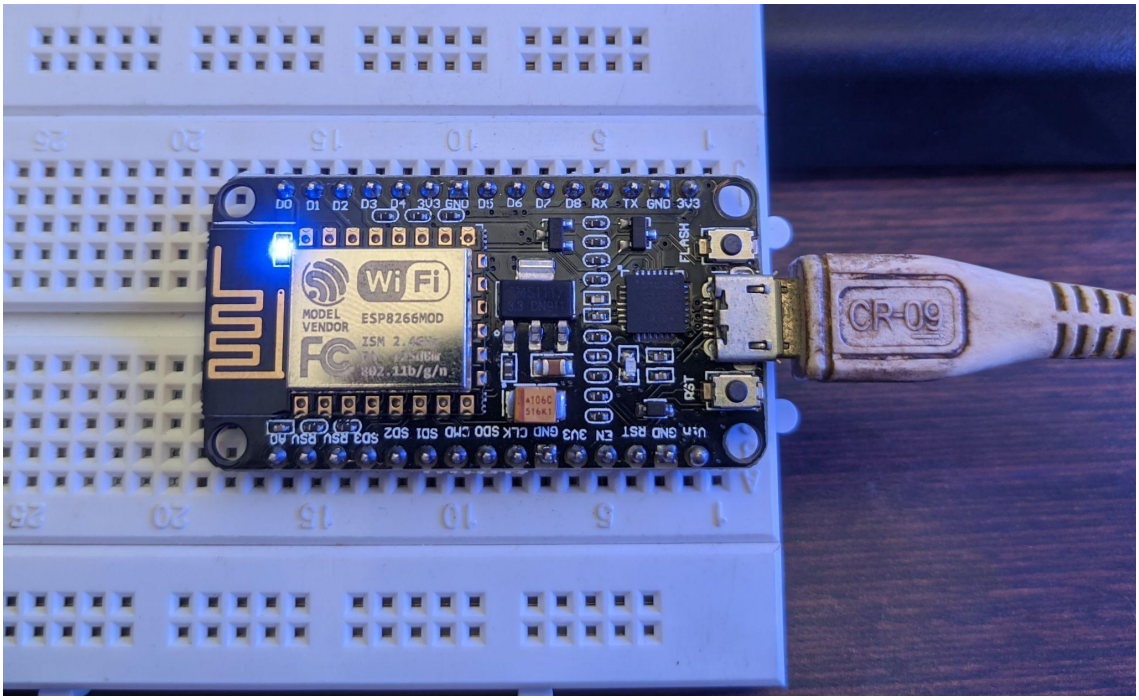
Step 4: Search the word ESP8266 in Tools>boards>boards manager from the Tools menu. Then install ESP8266 boards. After complete installation, you will see the INSTALLED label on ESP8266 Boards.



Step 5: Go to Tools then go to Board: “Generic ESP8266 Module” ESP8266 boards(3.0.2) Generic ESP8266 Module.



Step 6: Successfully Installed.



PRACTICAL: 2

Aim : GPIO Interfacing and programming

Theory :

GPIO

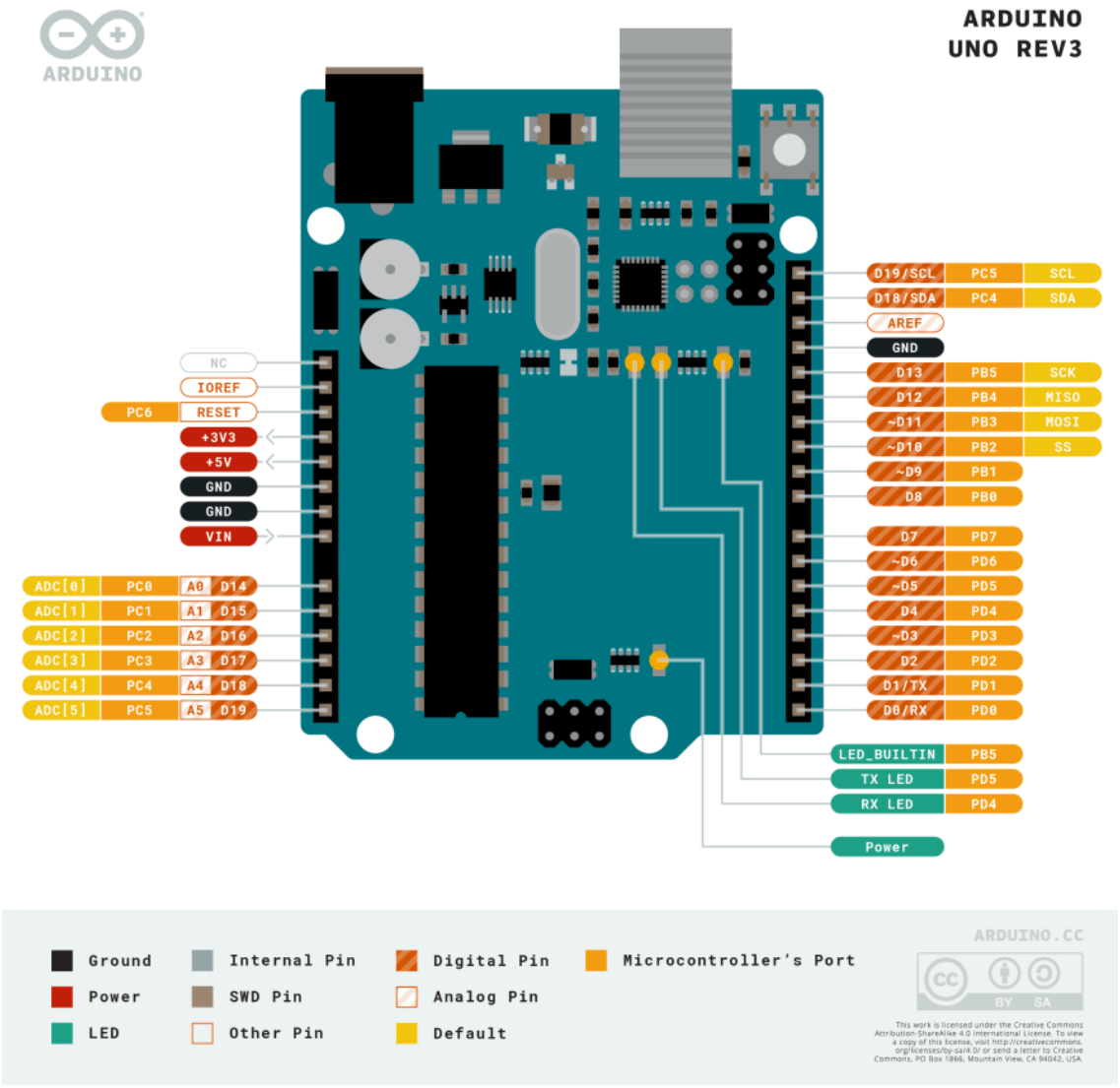
- ➔ A general-purpose input/output (GPIO) is an uncommitted digital signal pin on an integrated circuit or electronic circuit board which may be used as an input or output, or both, and is controllable by the user at runtime.
- ➔ As an input port, it can be used to communicate to the CPU the ON/OFF signals received from switches, or the digital readings received from sensors.

Arduino GPIO

- ➔ General-Purpose Input Output (GPIO) is a digital pin of an IC. It can be used as input or output for interfacing devices.
- ➔ If we want to read the switch's state, sensor data, etc then we need to configure it as input. And if we want to control the LED brightness, motor rotation, show text on display, etc then we need to configure it as output.
- ➔ Arduino analog pins can also be used as digital input/output pins. Let's see digital input, output of Arduino (ATmega).

Digital Output

- ➔ Arduino (ATmega) digital pins can be configured as output to drive output devices. We have to configure these pins to use as output.
- ➔ To configure these pins, pinMode () function is used which sets the direction of the pin as input or output.
- ➔ pinMode(pin no, Mode)
- ➔ This function is used to configure a GPIO pin as input or output,pin no number of pins whose mode we want to set.
- ➔ Mode INPUT, OUTPUT or INPUT_PULLUP
- ➔ E.g. pinMode (3, OUTPUT) //set pin 3 as output
- ➔ These Arduino (ATmega) pins can source or sink current up to 40 mA which is sufficient to drive led, LCD display but not sufficient for motors, relays, etc.
- ➔ These pins produce output in terms of HIGH (5 V or 3.3 V) or LOW (0 V). We can set output on these pins using the digitalWrite() function.
- ➔ digitalWrite(pin no, Output value)
- ➔ This function is used to set output as HIGH (5 V) or LOW (0 V)
- ➔ pin no number of a pin whose mode we want to set. Output value HIGH or LOW
- ➔ E.g. digitalWrite (3, HIGH)



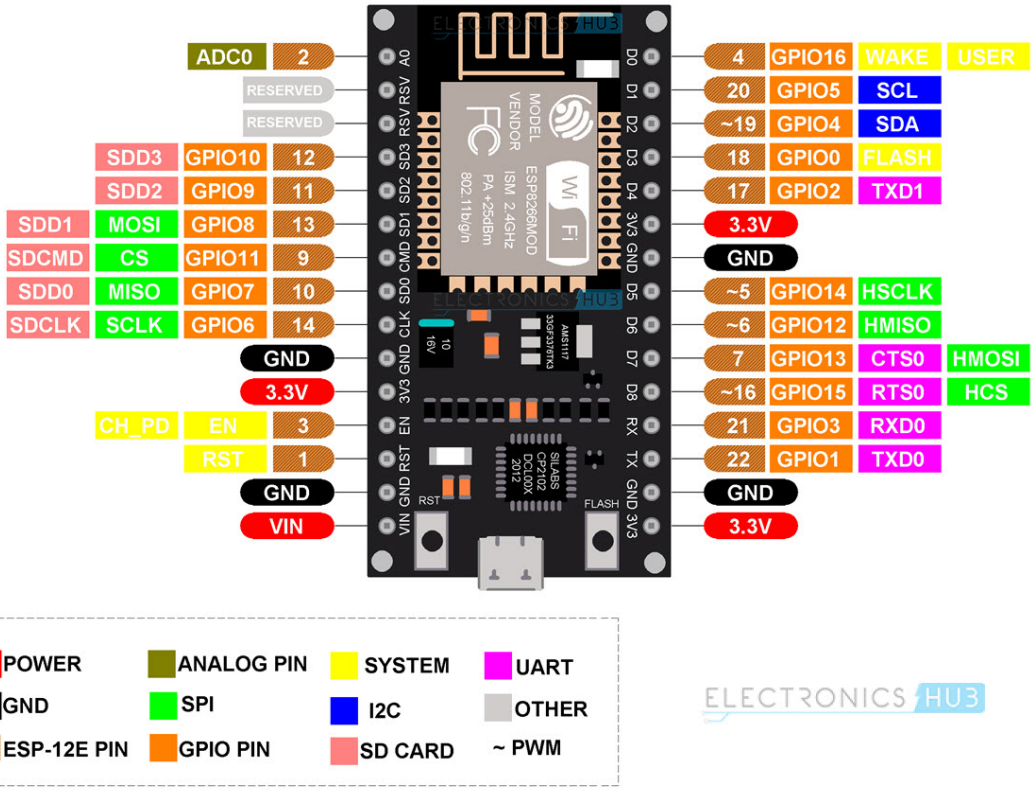
Digital Input

- ➔ To read data from senor or from any device/circuit, we need to configure a digital pin as input. Arduino pins are set as digital input (default). So, there is no need to configure the pin as input.
- ➔ To configure the pin as digital input, pinMode() function is used. We can read data from the GPIO pin using the digitalRead() function.
- ➔ digitalRead(pin)
- ➔ It is used to read data from a specified GPIO pin.

NodeMCU Pinout Configuration

- ➔ The following image shows the pinout for the NodeMCU board. A typical NodeMCU board (if it is based on the original NodeMCU Devkit design) has 30 pins. In this, 8

pins are related to power and 2 are reserved. The remaining 20 pins are associated with pins of ESP-12E Module.



- ➔ For practical purposes **ESP8266 NodeMCU** V2 and V3 boards present identical pinouts. While working on the NodeMCU based projects we are interested in the following pins.
- ➔ Power pins (3.3 V).
- ➔ Ground pins (GND).
- ➔ Analog pins (A0).
- ➔ Digital pins (D0 – D8, SD2, SD3, RX, and TX – GPIO XX)
- ➔ Most ESP8266 NodeMCU boards have one input voltage pin (Vin), three power pins (3.3v), four ground pins (GND), one analog pin (A0), and several digital pins (GPIO XX).

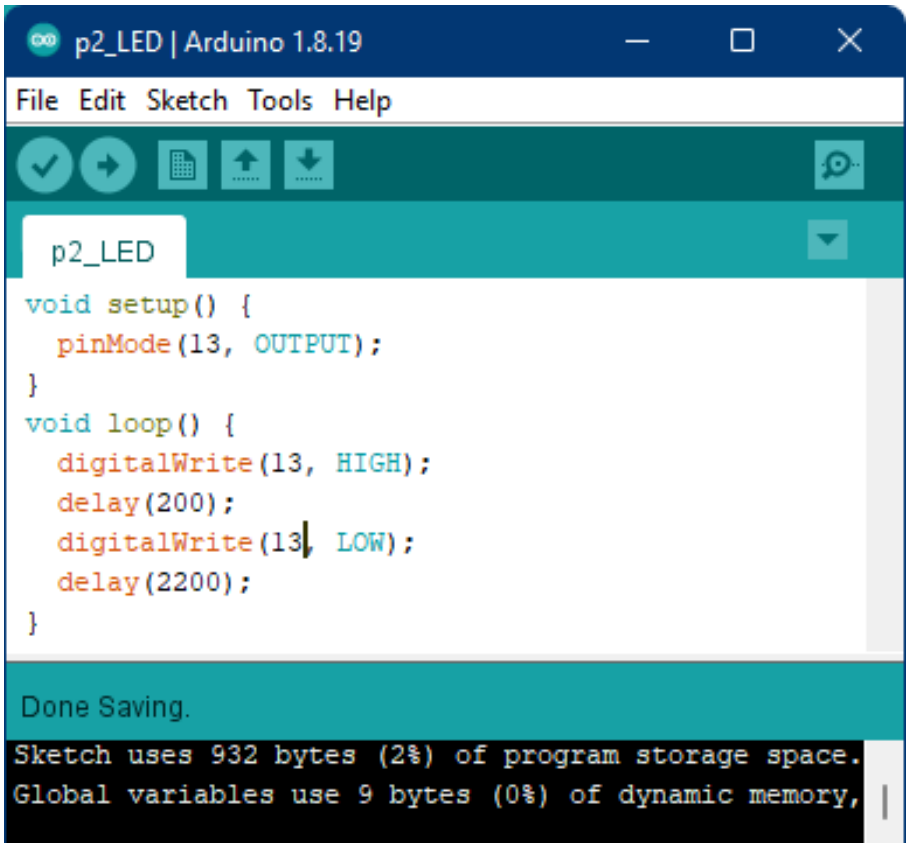
LED Blinking using NodeMCU :

- ➔ Components require :
  - 1 x NodeMCU
  - 1 x Breadboard
  - 1 x Ohm Resistor

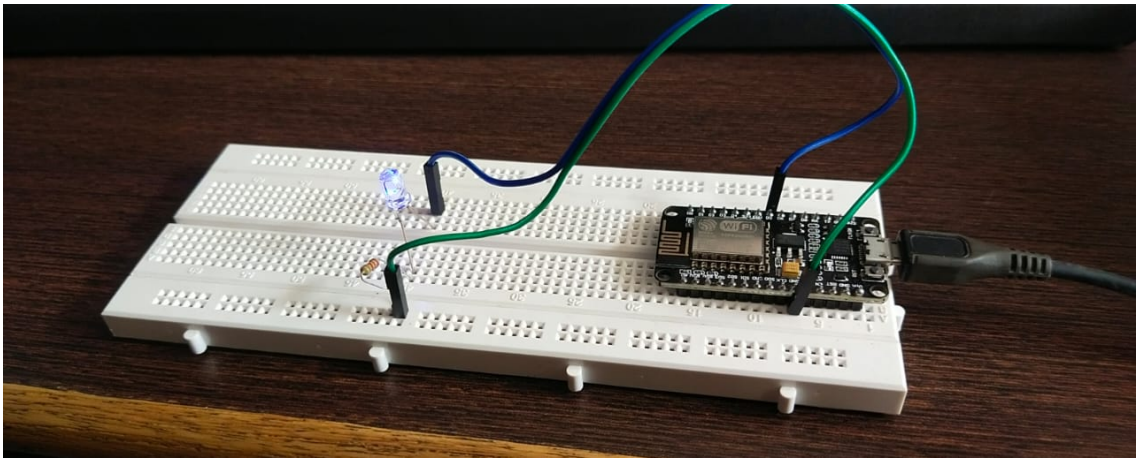
- 1 x Through hole LED
- 2 x Female to male jumper wires

➔ Sketch for LED Blinking

```
void setup( ) {
        pinMode(13, OUTPUT);
}
void loop( ) {
        digitalWrite(13, HIGH);
        delay(200);
        digitalWrite(13, LOW);
        delay(2200);
}
```



LED Blinking Output



PRACTICAL: 3

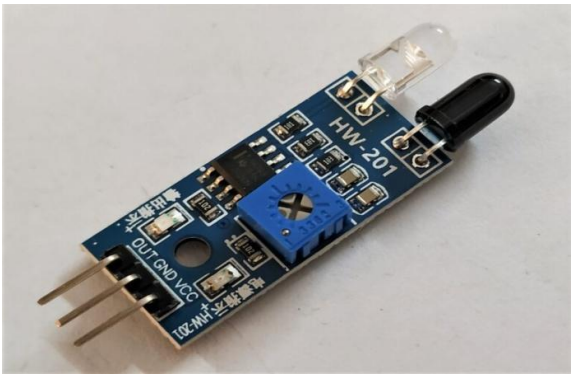
Aim: Digital ON/OFF sensor (PIR and IR) interfacing programming.

What is PIR and IR sensor?

IR sensor:-

An infrared sensor is an electronic device that emits in order to sense some aspects of the surroundings. An IR sensor can measure the heat of an object as well as detect the motion. These types of sensors measure only infrared radiation, rather than emitting it that is called a passive IR sensor. Usually, in the infrared spectrum, all the objects radiate some form of thermal radiation.

There are two types of infrared sensors: active and passive. Active infrared sensors both emit and detect infrared radiation. Active IR sensors have two parts: a light emitting diode (LED) and a receiver. When an object comes close to the sensor, the infrared light from the LED reflects off of the object and is detected by the receiver. Active IR sensors act as proximity sensors, and they are commonly used in obstacle detection systems. A passive infrared sensor is an electronic sensor that measures infrared light radiating from objects in its field of view. They are most often used in PIR-based motion detectors. PIR sensors are commonly used in security alarms and automatic lighting applications.



IR sensor

Hardware Requirements:-

- 1 x NodeMCUESP8266
- 1 x Breadboard
- 1 x IR sensor
- Jumper Wires

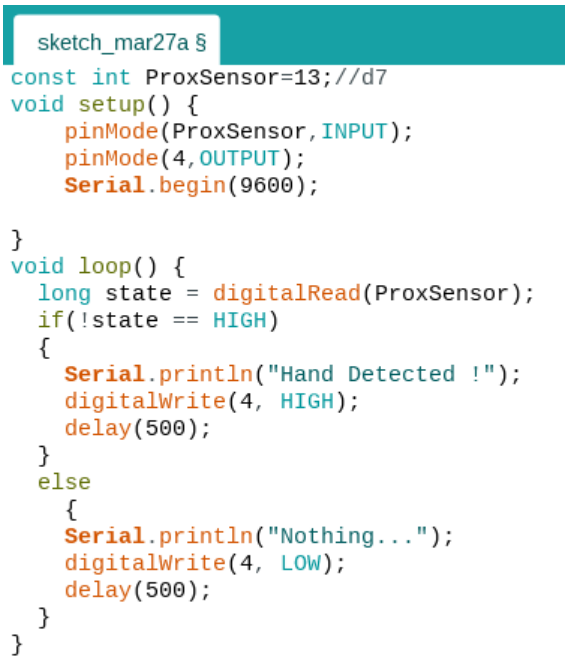
Connection of IR sensor with NodeMCU the circuit connections are made as follows:

- The Vcc pin of the IR module is connected to +3v of the NodeMCU.

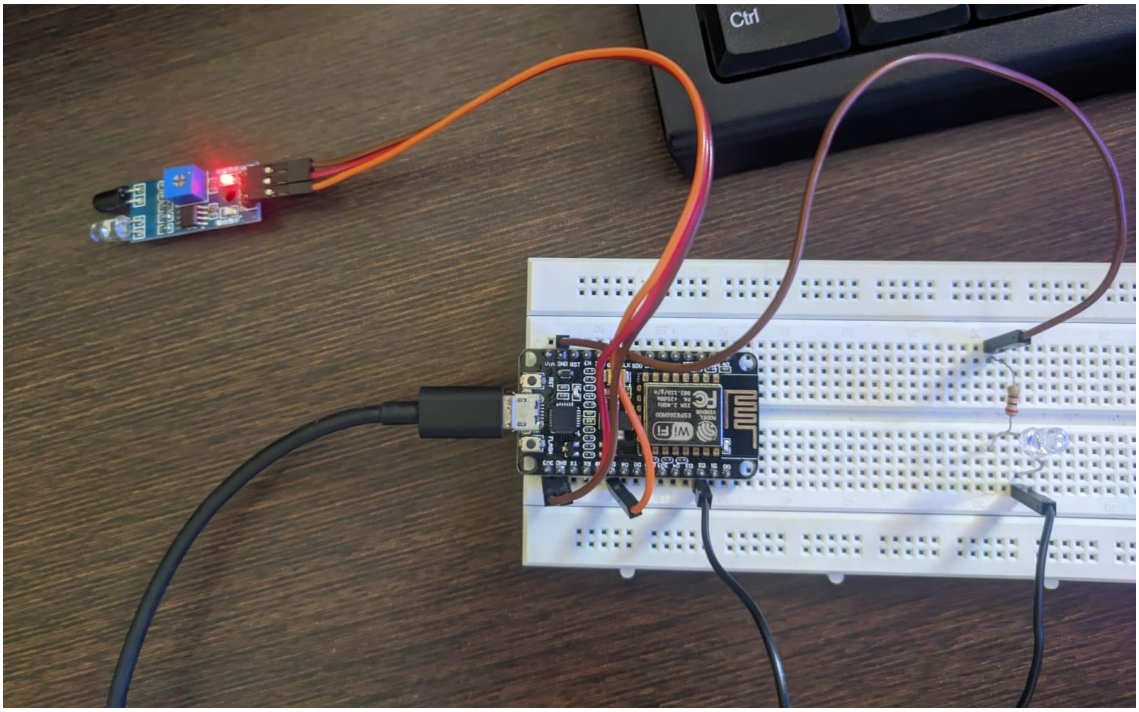
- Output pin of the IR module is connected to Digital pin D1 of the NodeMCU.
- The GND pin of the IR module is connected to the Ground pin (GND) of the NodeMCU.

Sketch for IR sensor using Arduino:-

```
const int ProxSensor=13;//d7
void setup() {
   pinMode(ProxSensor,INPUT);
   pinMode(4,OUTPUT);
   Serial.begin(9600);
}
void loop() {
  long state = digitalRead(ProxSensor);

  if(!state == HIGH)
  {
    Serial.println("Hand Detected !");
    digitalWrite(4, HIGH);
    delay(500);
  }
  else
    {
    Serial.println("Nothing...");
    digitalWrite(4, LOW);
    delay(500);
  }
}
```



IR sensor Output:-



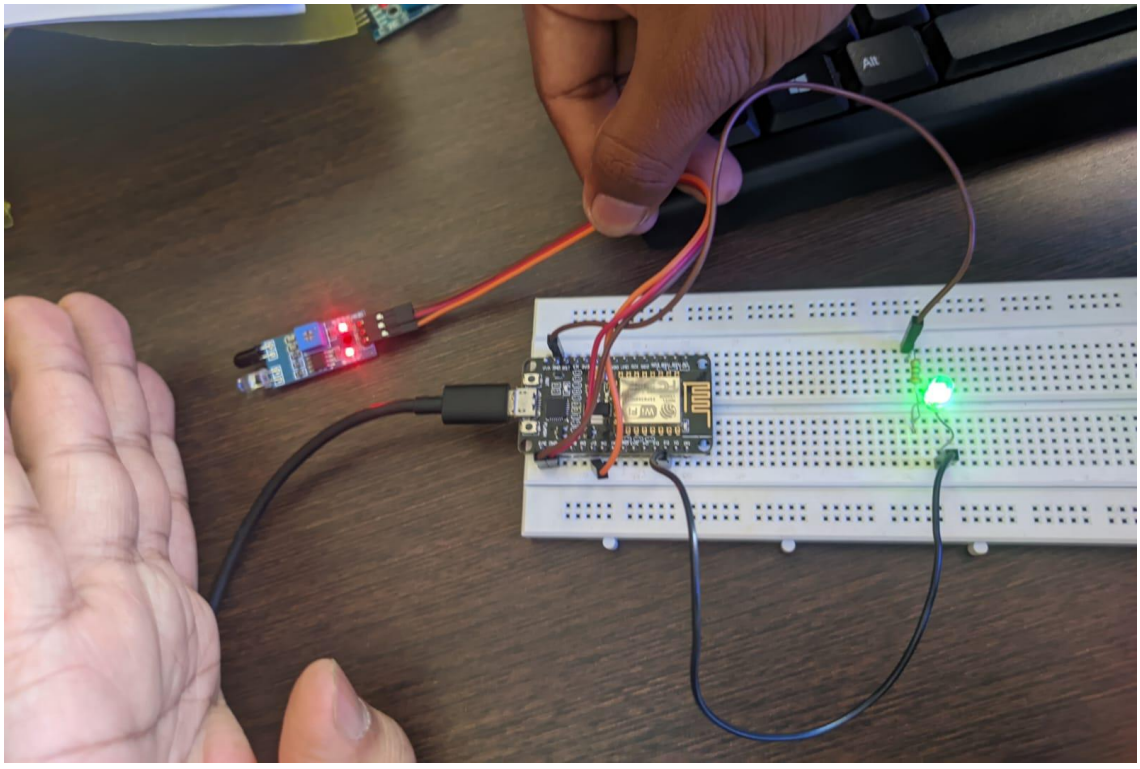


Figure 1. Digital interfacing building circuit (IR Sensor)

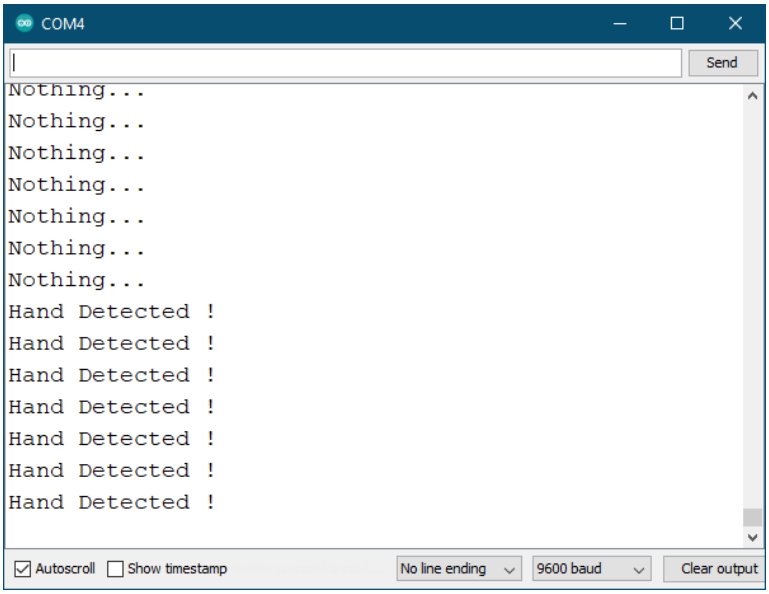


Figure 2 . Software simulation digital output

PIR sensor:-

A passive infrared sensor (PIR sensor) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view. They are most often used in PIR-based motion detectors. PIR sensors are commonly used in security alarms and automatic lighting applications.PIR sensors allow you to sense motion, almost always used to detect whether a human has moved in or out of the sensors range. They are small, inexpensive, low-power, easy to use and don't wear out. For that reason they are commonly found in appliances and gadgets used in homes or businesses. They are often referred to as PIR, "Passive Infrared", "Pyroelectric", or "IR motion" sensors.

PIR sensor

Hardware Requirements:-

- 1 x NodeMCU ESP8266
- 1 X PIR Sensor
- 1 X Breadboard
- Jumper Wires
- 1 X LED
- 1 x Ohm Resistor

Sketch for PIR sensor using Arduino:-

```
int ledPin = 12; // LED
int pirpin = 13; // PIR Out pin
int pirStat = 0; // PIR status
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(pirPin, INPUT);
  Serial.begin(9600);
}
void loop() {
  pirstatdigitalRead(pirpin);
  if (pirStat == HIGH) {
    digitalwrite(ledPin, HIGH);
    Serial.println("Motion detect");
  } else {
    digitalwrite(ledPin, LOW);
    Serial.println("Motion absent");
  }
}
```

PIR sensor Output:-

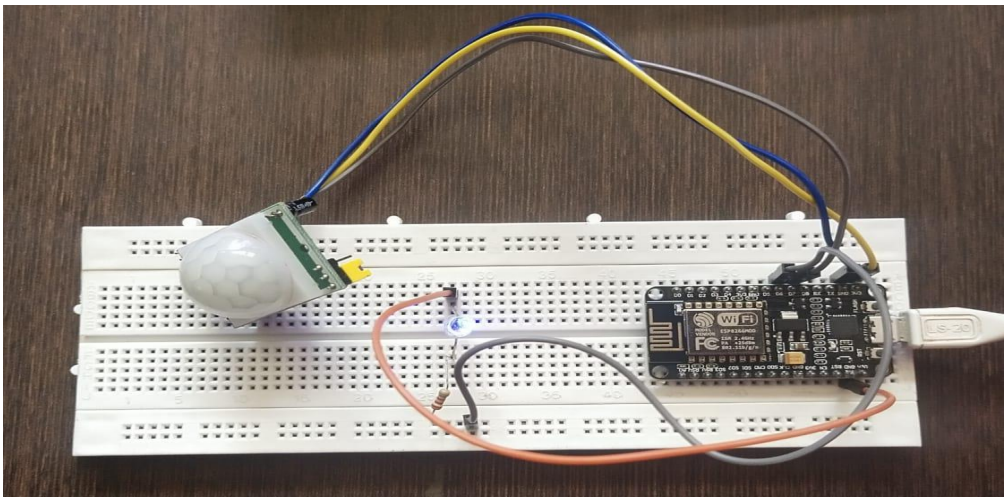


Figure 3. Digital interfacing building circuit (PIR Sensor)

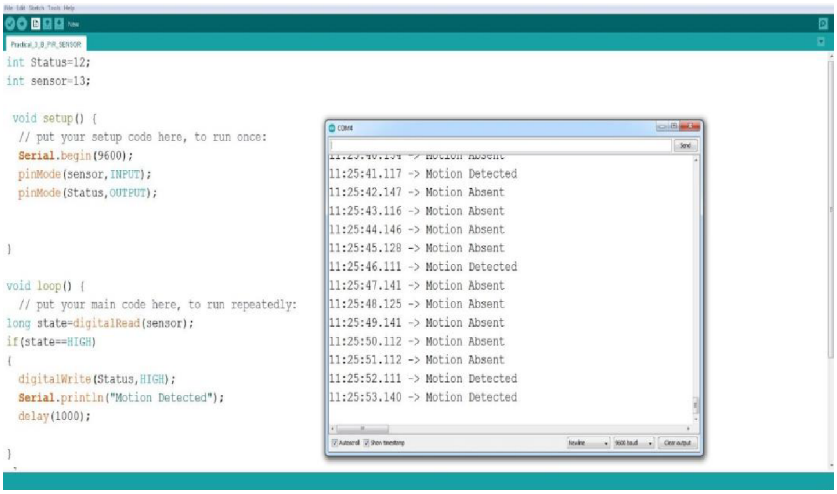


Figure 4. Code and software simulation digital output

PRACTICAL: 4

AIM: - Analog Sensor programming and uploading sensor data on cloud and Write a program to upload data of temperature and Humidity using Arduino/Node MCU.

What is a DHT11 Sensor?

The DHT11 is a basic, ultra-low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed).

Getting API Key

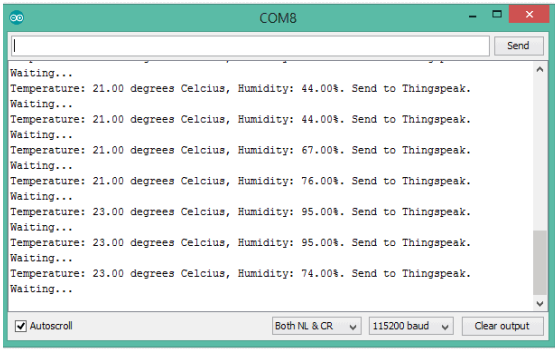
1. Go to **https://thingspeak.com/** and create an account if you do not have one. Login to your account.
2. Create a new channel by clicking on the button. **Enter basic details of the channel.** Than **Scroll down and save the channel.**
3. Channel Id is the identity of your channel. Note down this. Then go to API keys copy and paste this key to a separate notepad file will need it later.

CODE :

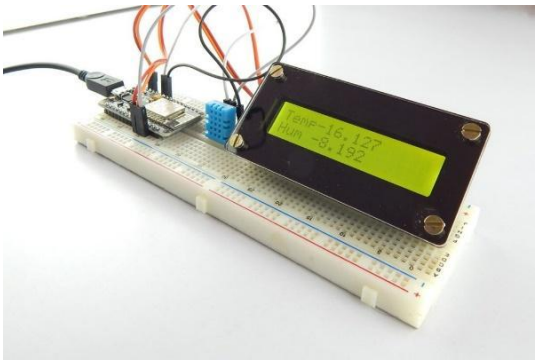
```
#include<DHT.h>  // Including library for dht
#include<ESP8266WiFi.h>
String apiKey = "7W5FGV2017QNJ8HI";
const char ssid = "RNP7";
const char pass = "123";
const char server = "api.thingspeak.com";

#define DHTPIN 0
DHT dht(DHTPIN, DHT11);
WiFiClient client;

void setup() {
  Serial.begin(115200);
  delay(10);
  dht.begin();
  Serial.println("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
```

```
  }
  Serial.println("");
  Serial.println("WiFi connected");
}
void loop() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  if (isnan(h) || isnan(t)) {
    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(t);
    postStr += "&field2=";
    postStr += String(h);
    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.print("Temperature: ");
    Serial.print(t);
    Serial.print(" degrees Celcius, Humidity: ");
    Serial.print(h);
    Serial.println("%. Send to Thingspeak.");
  }
  client.stop();
  Serial.println("Waiting...");
  delay(10000);
}
```

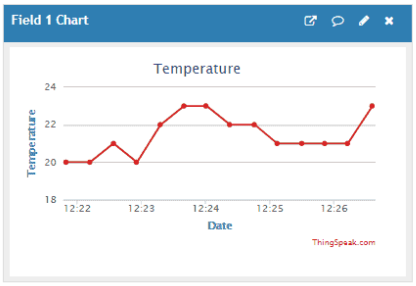
OUTPUT OF DHT11 SENSOR DIAGRAM

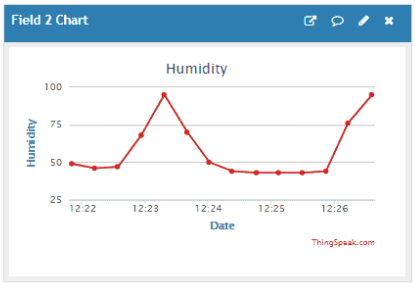


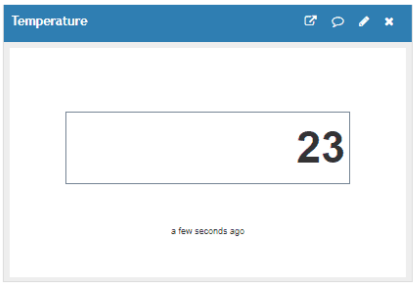
CONNECTION

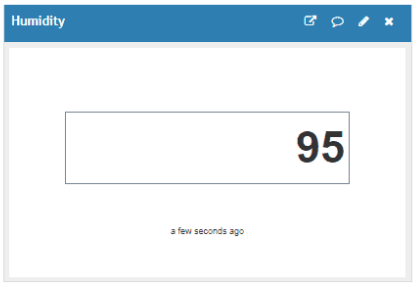


CLOUD OUTPUT:









PRACTICAL: 5

AIM: - Controlling devices remotely using Bluetooth link, Wi-Fi link

Introduction:

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.

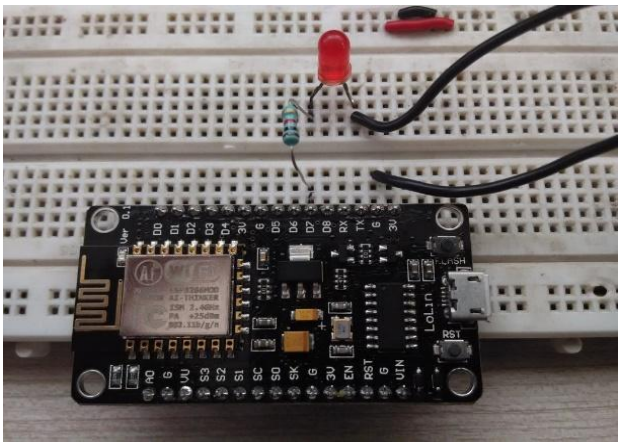
How It Works

A. **Connect** in seconds to approved computers, machines, devices, or even unattended servers or other equipment.
B. **Access** system, data, files, and applications.
C. **Control** the equipment, system, machine, computer, phone, or IoT device, as though you were the primary user handling it in person.

CODE to Control LED

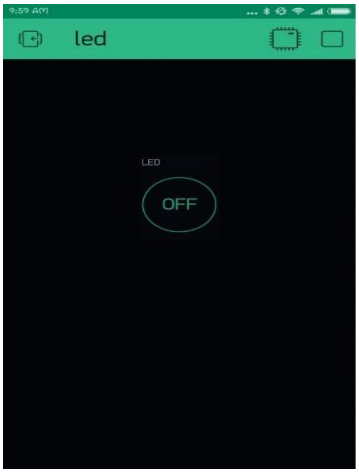
```
home_auto
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "Xn-YmlSEfNVIBVRCw51QLxsDED_FqZOr";

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "GalaxyA30";
char pass[] = "riteeka13";


void setup()
{

  Serial.begin(115200);
 Blynk.begin(auth, ssid, pass);
}

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

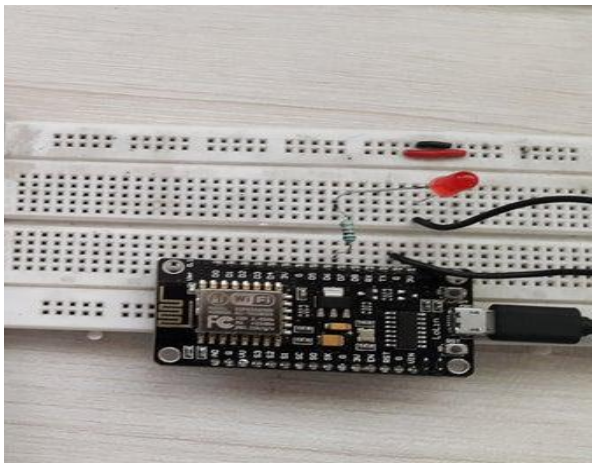
CONNECTION DIAGRAM



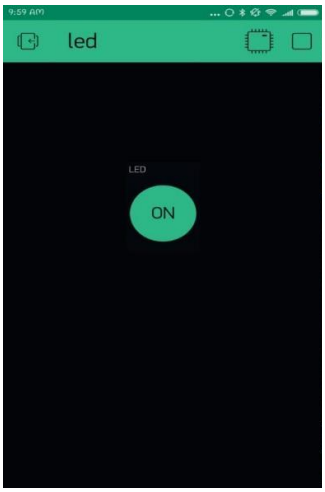
OUTPUT :

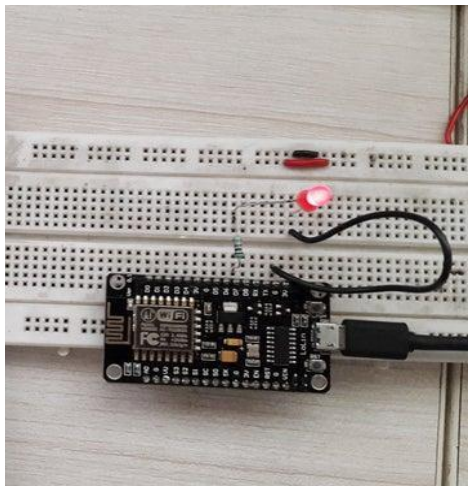
OFF in Blynk





ON in Blynk





PRACTICAL : 6

AIM: -Write a program to monitor Heartbeat in Arduino UNO.

What is a Heartbeat Sensor?

Heartbeat Sensor is an electronic device that is used to measure the heart rate i.e. speed of the heartbeat. Monitoring body temperature, heart rate and blood pressure are the basic things that we do in order to keep us healthy.
In order to measure the body temperature, we use thermometers and a sphygmomanometer to monitor the Arterial Pressure or Blood Pressure.

Working of the Circuit

Upload the code to Arduino UNO and Power on the system. The Arduino asks us to place our finger in the sensor and press the switch.

Place any finger (except the Thumb) in the sensor clip and push the switch (button). Based on the data from the sensor, Arduino calculates the heart rate and displays the heartbeat in bpm.

While the sensor is collecting the data, sit down and relax and do not shake the wire as it might result in faulty values.

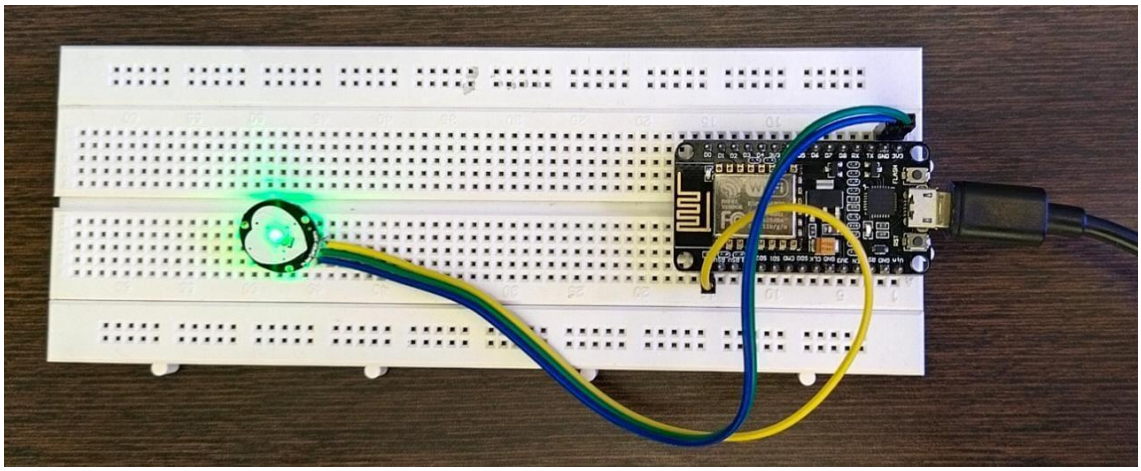
After the result is displayed on the LCD, if you want to perform another test, just push the reset button on the Arduino and start the procedure once again.

CODE:

```
int PulseSensorPurplePin = 0; // Pulse Sensor PURPLE WIRE connected to
ANALOG PIN 0
int LED13 = 13; //  The on-board Arduion LED
int Signal; // holds the incoming raw data. Signal value can range from 0-1024
int Threshold = 550; // Determine which Signal to "count as a beat", and which to
ignore.
// The SetUp Function:
void setup() {
  pinMode(LED13, OUTPUT); // pin that will blink to your heartbeat!
  Serial.begin(9600); // Set up Serial Communication at certain speed.
}
// The Main Loop Function
void loop() {
  Signal = analogRead(PulseSensorPurplePin); // Read the Pulse Sensors value.
  // Assign this value to the "Signal" variable.
  Serial.println(Signal); // Send the Signal value to Serial Plotter.
  if (Signal > Threshold) { // If the signal is above "550", then "turn-on" Arduino's
on-Board LED.
```

```
    digitalWrite(LED13, HIGH);
  } else {
    digitalWrite(LED13, LOW); //  Else, the signal must be below "550", so "turn-off"
this LED.
  }
  delay(10);
}
```

CONNECTION DIAGRAM

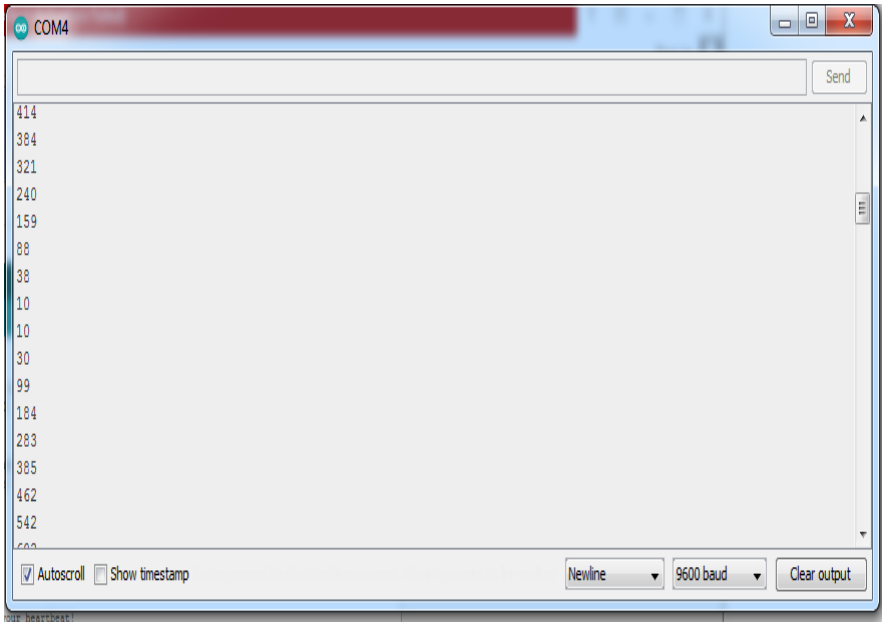


CODE:

pract_5_hbs

```
int PulseSensorPurplePin = 0;        // Pulse Sensor PURPLE WIRE connected to ANALOG PIN 0
int LED13 = 13;                      //  The on-board Arduion LED


int Signal;                          // holds the incoming raw data. Signal value can range from 0-1024
int Threshold = 550;                 // Determine which Signal to "count as a beat", and which to ingore.


// The SetUp Function:

void setup() {

  pinMode(LED13,OUTPUT);             // pin that will blink to your heartbeat!
   Serial.begin(9600);               // Set's up Serial Communication at certain speed.

}

// The Main Loop Function
void loop() {

  Signal = analogRead(PulseSensorPurplePin);     // Read the PulseSensor's value.
                                                 // Assign this value to the "Signal" variable.

   Serial.println(Signal);                       // Send the Signal value to Serial Plotter.


   if(Signal > Threshold){                       // If the signal is above "550", then "turn-on" Arduino's on-Board LED.
     digitalWrite(LED13,HIGH);
   }

   else {
     digitalWrite(LED13,LOW);                    //  Else, the sigal must be below "550", so "turn-off" this LED.
   }


delay(10);

}
```

OUTPUT:



PRACTICAL: 7

AIM: -Experiments on Agriculture lot (Soil moisture, PH monitor)

In agriculture IoT applications include farm vehicle tracking, livestock monitoring, storage monitoring and other farm operations. The diagram on the right provides a visual of this application.
In this IoT model, sensors can be deployed in the farm – to the ground, in water, in vehicles etc. to collect data. The collected data is stored in the cloud system or server and accessed by the farmer via the internet or their mobile phones.

What is Soil moisture Sensor?

Soil moisture sensors measure the volumetric water content in soil. Since the direct gravimetric measurement of free soil moisture requires removing, drying, and weighing of a sample, soil moisture sensors measure the volumetric water content indirectly by using some other property of the soil, such as electrical resistance, dielectric constant, or interaction with neutrons, as a proxy for the moisture content.

What is PH monitor Sensor?

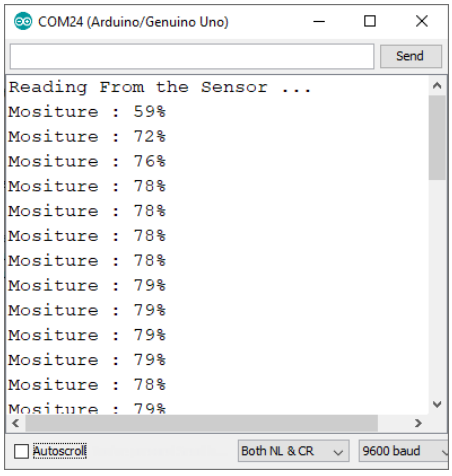
A pH sensor is one of the most essential tools that's typically used for water measurements. This type of sensor is able to measure the amount of alkalinity and acidity in water and other solutions.

CODE:

soil_moisture_anu

```
const int Pin=0;
const int limit=480;


void setup() {
  Serial.begin(9600);
  pinMode(13,OUTPUT);

  // put your setup code here, to run once:

}

void loop() {
  int value;
  value=analogRead(Pin);
  Serial.println("analog value:");
  Serial.println(value);
  delay(1000);
  if (value<limit)
  {
    digitalWrite(13,HIGH);
  }
  else
  {
    digitalWrite(13,LOW);
  }
  // put your main code here, to run repeatedly:

}
```

OUTPUT OF Soil moisture Sensor



CODE:

Sensor

pH_monitor_anu

```
int pH_Value;
float Voltage;

void setup()
{
  Serial.begin(9600);
  pinMode(pH_Value, INPUT);
}

void loop()
{
  pH_Value = analogRead(A0);
  Voltage = pH_Value * (5.0 / 1023.0);
  Serial.println(Voltage);
  delay(500);
}
```

OUTPUT OF PH monitor

PRACTICAL: 8

AIM:-IoT based home automation.

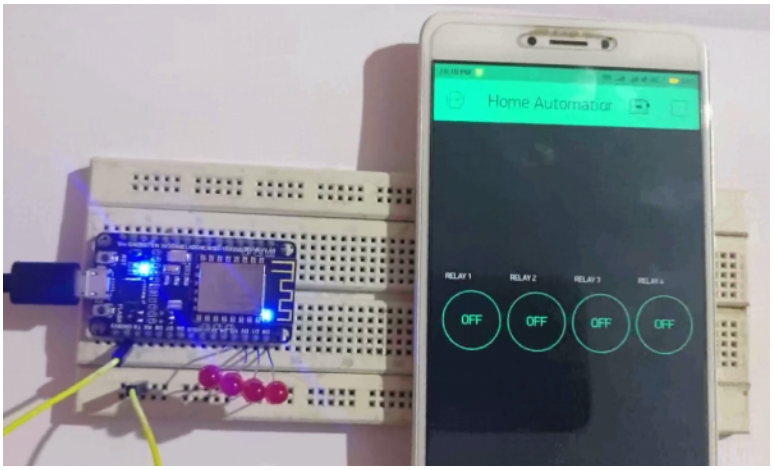
The Internet of Things (or commonly referred to as IoT) based Home Automation system, as the name suggests aims to control all the devices of your smart home through internet protocols or cloud based computing.
The IoT based Home Automation system offers a lot of flexibility over the wired systems as it comes with various advantages like ease-of-use, ease-of-installation, avoid complexity of running through wires or loose electrical connections, easy fault detection and triggering and above and all it even offers easy mobility.

CODE:

home_automation_anu

```
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

// You should get Auth Token in the Blynk App.
char auth[] = "yf8s_RKlgijkZZmylrVenzPSLhVnk9Fp";

// Your WiFi .
// Set password for open networks.
char ssid[] = "ANU";
char pass[] = "12345678";

void setup()
{
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);
}

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

OUTPUT of Blynk:



PRACTICAL: 9

AIM: Familiarization with Raspberry Pi and its architecture. Perform necessary software installation.

Architecture

The architecture of a RPI is based on a Broadcom SOC BCM2835 which belongs to ARM Cortex family with ARMv6 architecture. It is a 32 bit RISC with a clock speed of 700 MHz and has eight pipelines. This architecture is different from the Arduino processor which is X86 based and works on CISC which is a better option as it provides the pipelining facilities. BCM2835 also provides branch prediction, improving the flow in the instruction pipeline. The advantages that are obtained because of having a RISC architecture are low transistor count, low power consumption and therefore less heat generation. This architecture also features three pipelines, viz, ALU, MAC and load /store.

Introduction:

Ultra-low-cost credit-card-sized single-board Linux computer Raspberry Pi developed by Raspberry foundation (UK), is a small, powerful and lightweight ARM11 (Broadcom 2835) based development board or a computer which is able to perform operations similar to a PC. The powerful graphics capabilities and HDMI video output make it ideal for multimedia applications such as media centers and narrowcasting solutions. With access to the internet, through Ethernet or Wi-Fi (with a USB dongle), and a high definition output, the Raspberry Pi is an incredibly versatile piece of computing kit. The Raspberry Pi is based on a Broadcom BCM2835 chip that is based on ARMv6. It does not feature a built-in hard disk or solid-state drive, instead relying on an SD card for booting and long-term storage.

This single board computer is inexpensive yet comes packed with Ethernet, USB, a high powered graphics engine, digital I/O ports and enough CPU power to accomplish your projects.

The Foundation provides Debian and Arch Linux ARM distributions for download. Tools are available for Python as the main programming language.

The Raspberry Pi Model B features:

The Broadcom BCM2835 ARM11 (used in most smart-phones) 700Mhz ‚System on Chip‟ Processor (Similar performance to a 300MHz Pentium 2 Processor).

Integrated Video core 4 Graphics Processing Unit (GPU) capable of playing Full 1080p High Definition Blu-Ray Quality Video (Roughly equivalent graphical processing power of an Xbox 1)

512Mb SDRAM

The free, versatile, and highly developer friendly Debian GNU/Linux Operating System

2 x USB Ports

HDMI Video Output

RCA Video Output

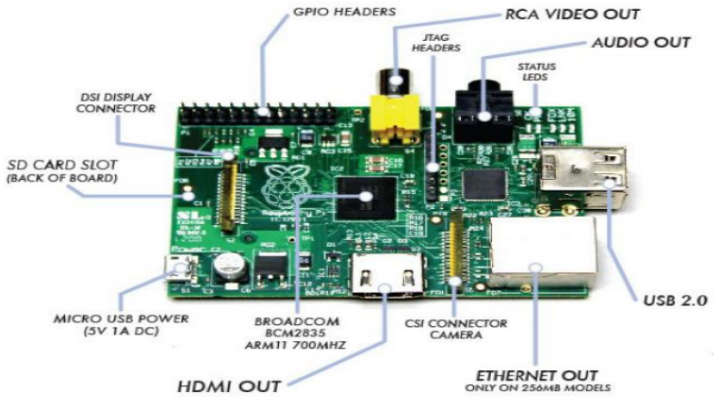
3.5mm Audio Output Jack

10/100Mb Ethernet Port for Internet Access

5V Micro USB Power Input Jack

SD, MMC, SDIO Flash Memory Card Slot

26-pin 2.54mm Header Expansion Slot (Which allow for peripherals and expansion boards)

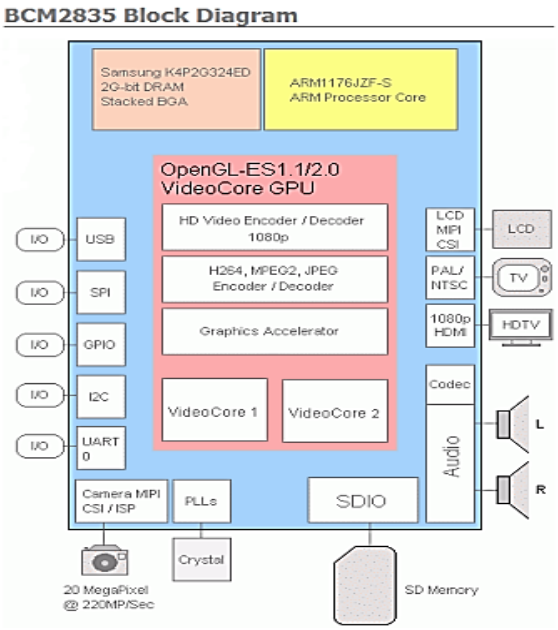


Scratch view of raspberry pi Hardware Specification & Description:

Name	Description
The Processor	This chip is a 32 bit, 700 MHz System on Chip, which is built on the ARM11 architecture (ARMv6 instruction set). SoC-Broadcom BCM2835
The Secure Digital Card slot	SD / MMC / SDIO card slot (3.3V card power support only)
The USB port	Model B has 2 Nos. USB 2.0 ports. Model A has only one current limited USB port.
Ethernet port	Model B has standard RJ45 connector and Model A does not.
HDMI connector	The HDMI port provides digital video and audio output hence TV can be connected. 14 different video resolutions are supported.
Status LEDs	ACT Green Light when the SD card is accessed PWR Red Hooked up to 3.3V power FDX Green ON if the network adapter is full duplex. LNX Green Network activity light. 100 Yellow On if the network connection is 100Mbps
Analog Audio out	This is a standard 3.5mm mini analog audio jack,intended to drive high impedance load(like amplified speakers).
Composite video out	This is a standard RCA-type jack that provides NTSC or PAL video signals.

Memory (SDRAM)	512 MB in Model B and 256 MB in Model A
Supportable Operating systems	Arch Linux ARM, Debian GNU/Linux, Gentoo, Fedora, FreeBSD, NetBSD, Plan 9, Raspbian OS, RISC OS, Slackware Linux

BLOCK DIAGRAM:



DETAILED DESCRIPTION OF COMPONENTS:

1) System Timer: The System Timer peripheral provides four 32-bit timer channels and a single 64-bit free running counter. Each channel has an output compare register, which is compared against the 32 least significant bits of the free running counter values.

2) The Processor: At the heart of the Raspberry Pi is the same processor you would have found in the iPhone 3G and the Kindle 2, so you can think of the capabilities of the Raspberry Pi as comparable to those powerful little devices. This chip is a 32 bit, 700 MHz System on a Chip, which is built on the ARM11 architecture. ARM chips come in a variety of architectures with different cores configured to provide different capabilities at different price points

3) Interrupt controller: The interrupt controller can be programmed to interrupt the processor when any of the status bits are set. The GPIO peripheral has three dedicated interrupt lines. Each GPIO bank can generate an independent interrupt. The third line generates a single interrupt whenever any bit is set.

4) General Purpose Input/output (GPIO): 3.3 volt logic via 26 pin header (NOT 5 volt or short tolerant) Pins can be configured to be input/output. General Purpose Input/output (GPIO) is a generic pin on a chip whose behavior can be controlled by the

user at run time. True GPIO (General Purpose Input Output) pins that you can use to turn LEDs on and off etc.

5) PCM / I2S Audio: The PCM audio interface is an APB peripheral providing input and output of telephony or high quality serial audio streams. It supports many classic PCM formats including I2S. The PCM audio interface has 4 interface signals; PCM_CLK - bit clock. PCM_FS - frame sync signal. PCM_DIN - serial data input. PCM_DOUT - serial data output. PCM is a serial format with a single bit data_in and out.

6) DMA Controller: The BCM2835 DMA Controller provides a total of 16 DMA channels. Each channel operates independently from the others and is internally arbitrated onto one of the 3 system buses.

7) UART: The BCM2835 device has two UARTS. On mini UART and PL011 UART. The PL011 UART is a Universal Asynchronous Receiver/Transmitter. This is the ARM UART (PL011) implementation. The UART performs serial-to-parallel conversion on data characters received from an external peripheral device or modem, and parallel-to-serial conversion on data characters received from the Advanced Peripheral Bus.

8) Pulse Width Modulator: PWM controller incorporates the following features: • Two independent output bit-streams, clocked at a fixed frequency. • Bit-streams configured individually to output either PWM or a serialized version of a 32-bit word. • PWM outputs have variable input and output resolutions. • Serialize mode is configured to load data to and/or read data from a FIFO storage block that can store up to eight 32-bit words. • Both modes are clocked by clk_pwm which is nominally 100MHz, but can be varied by clock manager.

9) CPU

ARM 1176JZF-S (armv6k) 700MHz
RISC Architecture and low power draw
Not compatible with traditional PC software

10) MEMORY

RAM:- 512MB (Model B rev.2), 256 MB (Model A, Model B rev.1)

SD Card:- At least 4GB SD card is needed, and it should be a Class 4 card. Class 4 cards are capable of transferring at least 4MB/sec. Some of the earlier Raspberry Pi boards had problems with Class 6 or higher cards, which are capable of faster speeds but are less stable. One can also use a micro SD card using an adapter. As there is no hard drive on the Pi; everything is stored on an SD Card. A protective case is needed as the solder joints on the SD socket may fail if the SD card is accidentally bent.

11) Two USB 2.0 ports in RPi: Dual USB sockets on RPi model B, single on model A.It can be expandable via regular or powered hubs. On the Model B there are two USB 2.0 ports, but only one on the Model A. Some of the early Raspberry Pi boards were limited

in the amount of current that they could provide. Some USB devices can draw up 500mA.

12) Ethernet port: The model B has a standard RJ45 Ethernet port. The Model A does not, but can be connected to a wired network by a USB Ethernet adapter (the port on the Model B is actually an onboard USB to Ethernet adapter). WiFi connectivity via a USB dongle is another option.

13) HDMI connector: The HDMI port provides digital video and audio output. 14 different video resolutions are supported, and the HDMI signal can be converted to DVI (used by many monitors), composite (analog video signal usually carried over a yellow RCA connector), or SCART (a European standard for connecting audio-visual equipment) with external adapters.

14) Video:

HDMI or (digital) DVI via cheap adaptor/cable
Composite NTSC/PAL via RCA
Wide range of resolutions
NO VGA without an add-on, nontrivial converter (Adafruit)

15) Audio:

Via HDMI or from stereo jack
Output only
Support Maturity appears to be lagging

16) Networking

10/100mbps via RJ45 on model B
Wireless via USB add-on supported

17) Power: There is no power switch on the Pi. A Micro-USB connector is used to supply power (this isn‟t an additional USB port; it‟s only for power). Micro-USB was selected because cheap USB power supplies are easy to find.

Primary power via microUSB plug: a 1Amp cell charger works well, but to use a USB hard drive, 2 Amp power is needed.

Model A about a quarter amp less PC USB port does not work

PRACTICAL: 10

AIM: To interface LED with Raspberry Pi and write Python programming to turn ON/ OFF LED using Raspberry Pi.

The features of the Raspberry Pi's GPIO Pins before proceeding with the further step of how to Blink an LED using Raspberry Pin and its GPIO Pins.

Components Required

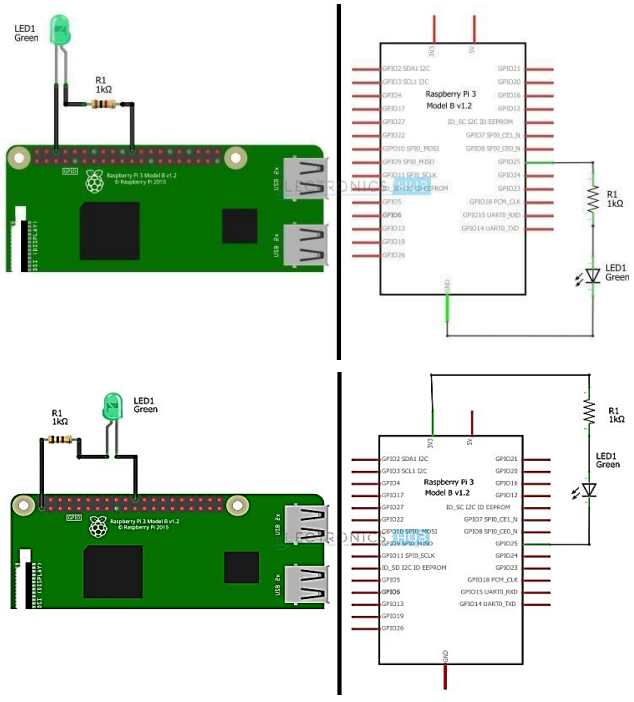
- Raspberry Pi 3 Model B (any Raspberry Pi would do fine)
- 5mm LED x 1
- 1KΩ Resistor (1/4 Watt) x 1
- Mini Breadboard x 1
- Connecting wires
- Miscellaneous (Computer, Ethernet cable, Power Supply for Raspberry Pi etc.)

Circuit Diagram of Blinking LED with Raspberry Pi

In order to Blink an LED using Raspberry Pi, we need to first connect the LED to the Raspberry Pi. There are two ways you can connect your LED to the Raspberry Pi. I'll show both ways of connecting the LED.

Circuit 1

In the first circuit, the anode of the LED is connected to GPIO25 (Physical Pin 22) through a 1KΩ current limiting resistor. The cathode of the LED is connected to the GND Pin.



In this circuit, the anode of the LED is connected to the 3.3V supply pin of the Raspberry Pi through the 1KΩ resistor. The cathode of the LED is connected to GPIO25 (Physical Pin 22).

In this circuit, the GPIO pin acts as the sink (GND).

NOTE: I'll be concentrating on the first circuit, where the GPIO pin GPIO25 acts as the source. The code explained in the further sections will be specific to this circuit. The code can also be used for second circuit with slight or no modifications.

CODE:

```
import RPi.GPIO as GPIO  # Import Raspberry Pi GPIO library
from time import sleep  # Import the sleep function from the time module

GPIO.setwarnings(False)  # Ignore warning for now
GPIO.setmode(GPIO.BOARD)  # Use physical pin numbering
GPIO.setup(
   8, GPIO.OUT, initial=GPIO.LOW
)  # Set pin 8 to be an output pin ,set initial value to low (off)
while True:  # Run forever
   GPIO.output(8, GPIO.HIGH)  # Turn on
   sleep(1)  # Sleep for 1 second
   GPIO.output(8, GPIO.LOW)  # Turn off
   sleep(1)  # Sleep for 1 second
```

Code for Blinking an LED with Raspberry Pi:

```
#!/usr/bin/env python
import RPi.GPIO as GPIO  # RPi.GPIO can be referred as GPIO from now
import time
ledPin = 22  # pin22
def setup():
      GPIO.setmode(GPIO.BOARD)  # GPIO Numbering of Pins
      GPIO.setup(ledPin, GPIO.OUT)  # Set ledPin as output
     GPIO.output(ledPin, GPIO.LOW)  # Set ledPin to LOW to turn Off the
LED
def loop():
   while True:
      print "LED on"
      GPIO.output(ledPin, GPIO.HIGH)  # LED On
      time.sleep(1.0)  # wait 1 sec
      print "LED off"
      GPIO.output(ledPin, GPIO.LOW)  # LED Off
      time.sleep(1.0)  # wait 1 sec

def endprogram():

   GPIO.output(ledPin, GPIO.LOW)  # LED Off
   GPIO.cleanup()  # Release resources
if __name__ == "__main__":  # Program starts from here
   setup()
   try:
      loop()
    except KeyboardInterrupt:  # When 'Ctrl C' is pressed, the destroy() will be
executed.
      endprogram()
```

OUTPUT OF LED