Introduction to IOT
LAB ACTIVITY – 1

Institute of Engineering and Technology
JK Lakshmipat University, Jaipur

SUBMITTED TO

Mr. Divanshu Jain

SUBMITTED BY

Garv Baheti

2020BTechCSE031

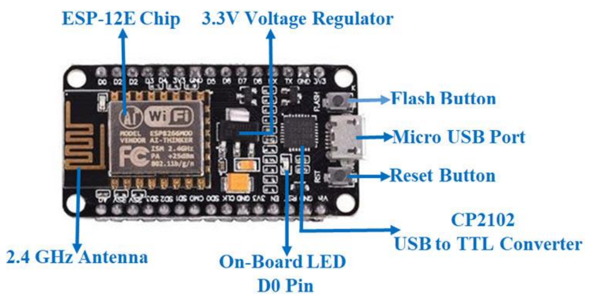
August 22, 2022

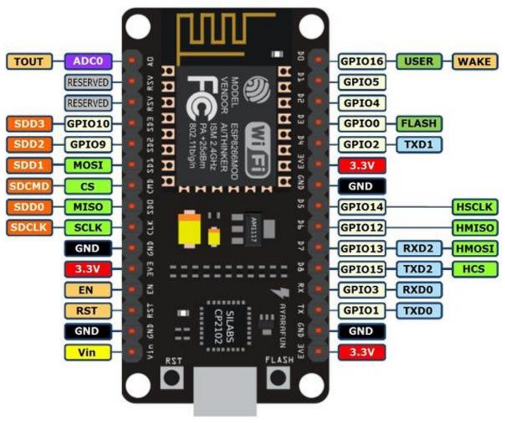
OBJECTIVE

1. To understand the pin diagram, architecture, and Specifications of NodeMCU
2. To blink the inbuilt LEDs of NodeMCU
3. To blink the External LEDs connected to pin number D5, D6, D7 of NodeMCU
4. To print the data on serial monitor of NodeMCU
5. Interface lm35 with nodemcu and print data on serial monitor

THEORY

NodeMCU is an open-source Lua based firmware and development board specially targeted for IoT based Applications. It includes firmware that runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The **NodeMCU ESP8266 development board** comes with the ESP-12E module containing the ESP8266 chip having Tensilica Xtensa 32-bit LX106 RISC microprocessor. This microprocessor supports RTOS and operates at 80MHz to 160 MHz adjustable clock frequency. NodeMCU has 128 KB RAM and 4MB of Flash memory to store data and programs. Its high processing power with in-built Wi-Fi / Bluetooth and Deep Sleep Operating features make it ideal for IoT projects. NodeMCU can be powered using a Micro USB jack and VIN pin (External Supply Pin). It supports UART, SPI, and I2C interface.





Applications

- Prototyping of IoT devices
- Low power battery operated applications
- Network projects
- Projects requiring multiple I/O interfaces with Wi-Fi and Bluetooth functionalities

NodeMCU ESP8266 Specifications & Features

- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 16
- Analog Input Pins (ADC): 1
- UARTs: 1
- SPIs: 1
- I2Cs: 1
- Flash Memory: 4 MB
- SRAM: 64 KB
- Clock Speed: 80 MHz
- USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
- PCB Antenna
- Small Sized module to fit smartly inside your IoT projects

NodeMCU Development Board Pinout Configuration

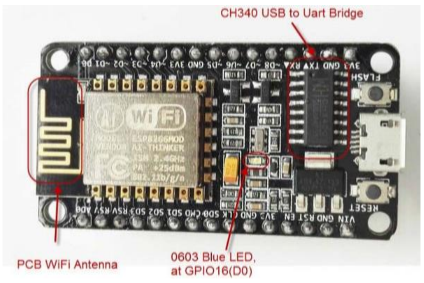
Pin Category	**Name**	**Description**
Power	Micro-USB, 3.3V, GND, Vin	**Micro-USB:** NodeMCU can be powered through the USB port **3.3V:** Regulated 3.3V can be supplied to this pin to power the board **GND:** Ground pins **Vin:** External Power Supply
Control Pins	**EN, RST**	The pin and the button resets the microcontroller
Analog Pin	A0	Used to measure analog voltage in the range of 0-3.3V
GPIO Pins	GPIO1 to GPIO16	NodeMCU has 16 general purpose input-output pins on its board
SPI Pins	SD1, CMD, SD0, CLK	NodeMCU has four pins available for SPI communication.
UART Pins	TXD0, RXD0, TXD2, RXD2	NodeMCU has two UART interfaces, UART0 (RXD0 & TXD0) and UART1 (RXD1 & TXD1). UART1 is used to upload the firmware/program.
I2C Pins		NodeMCU has I2C functionality support but due to the internal functionality of these pins, you have to find which pin is I2C.

TO BLINK THE INBUILT LEDS OF NODEMCU

REQUIREMENTS

- NodeMCU x 1
- Micro USB cable x 1
- PC x 1
- Software Arduino IDE(version 1.6.4+)

The blue led in the board was used to signal the execution of a particular procedure.Because the GPIO16 (DO) limitation to use the analogWrite().



CODE

```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

RESULT

TO BLINK THE EXTERNAL LEDS CONNECTED TO PIN NUMBER D5, D6, D7 OF NODEMCU

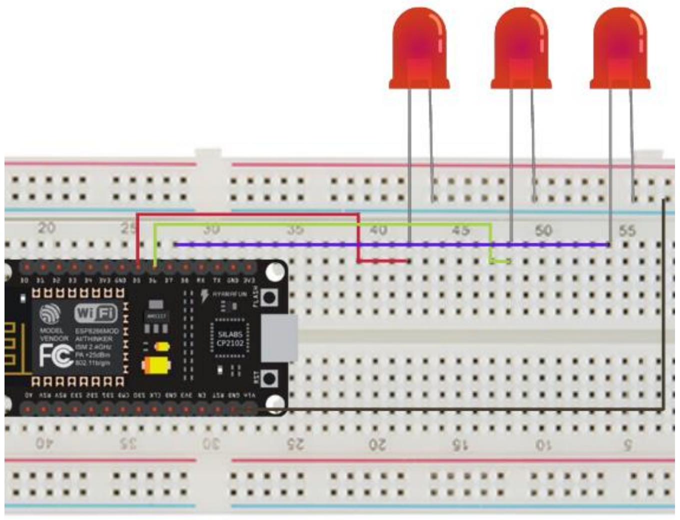
REQUIREMENTS

- NodeMCU x 1
- Micro USB cable x 1
- PC x 1
- Software Arduino IDE (version 1.6.4+)
- LED x 3

CIRCUIT

Connect the long leg of the LED1 (the positive leg, called the anode) to D5, LED2 to D6, LED3 to D7

Connect the short leg of the LEDs (the negative leg, called the cathode) to the GND Line.

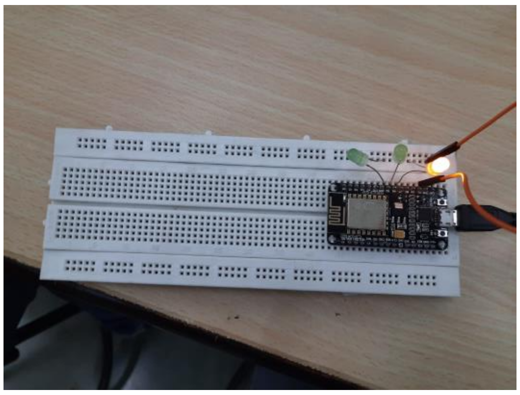


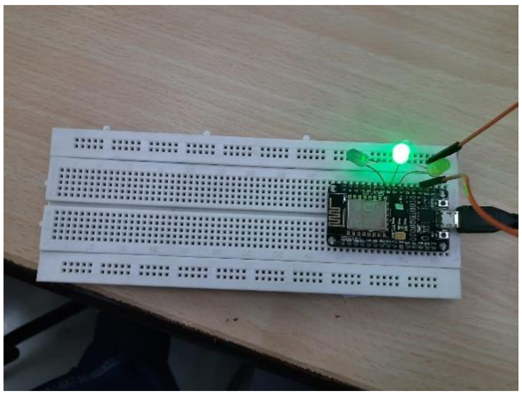
After connecting the circuit we upload the code to NodeMCU, this will blink our 3 LEDs in sequence with a delay of 600ms.

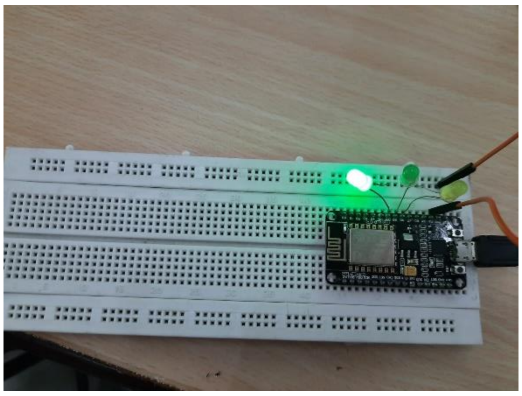
CODE

```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(D5, OUTPUT);
  pinMode(D6, OUTPUT);
  pinMode(D7, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(D5, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(600);                       // wait for a second
  digitalWrite(D5, LOW);    // turn the LED off by making the voltage LOW
  delay(600);                       // wait for a second

  digitalWrite(D6, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(600);                       // wait for a second
  digitalWrite(D6, LOW);    // turn the LED off by making the voltage LOW
  delay(600);

  digitalWrite(D7, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(600);                       // wait for a second
  digitalWrite(D7, LOW);    // turn the LED off by making the voltage LOW
  delay(600);

}
```

RESULT







TO PRINT THE DATA ON SERIAL, MONITOR OF NODEMCU

REQUIREMENTS

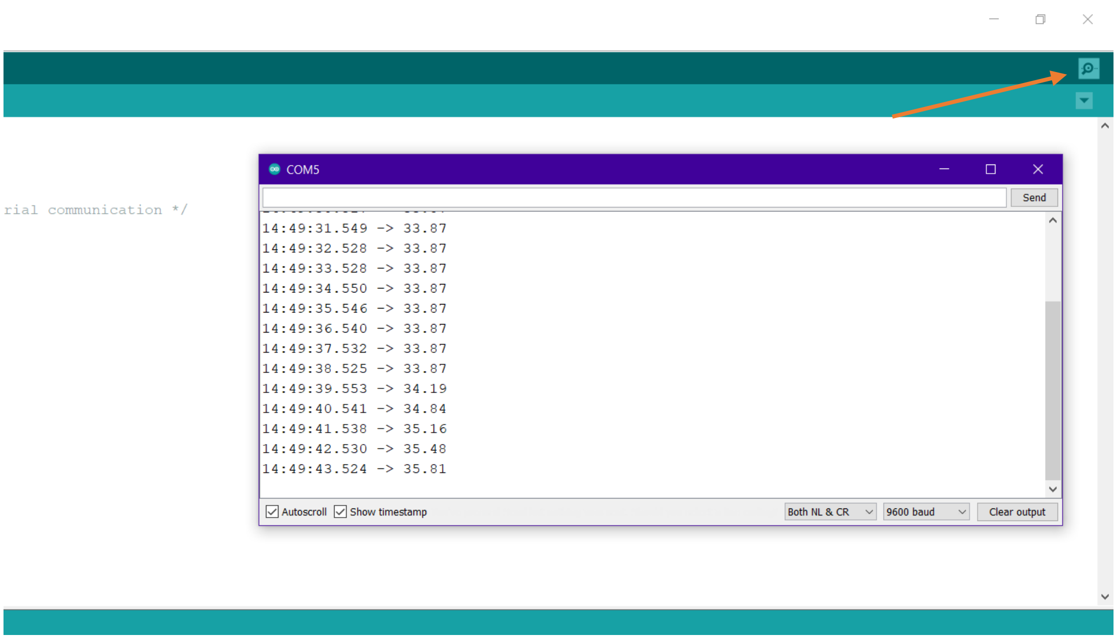
- NodeMCU x 1
- Micro USB cable x 1
- PC x 1
- Software Arduino IDE(version 1.6.4+)

CODE

```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(9600);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED off (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  Serial.println("MESSAGE");

  digitalWrite(LED_BUILTIN, LOW);    // turn the LED on by making the voltage LOW
  delay(5000);                       // wait for a second
}
```

This Code will blink the Internal LED and will Print the “MESSAGE” in Serial Monitor after every 6 seconds, which can be accessed from the top right button in the IDE.

RESULT



```
rial communication */
```

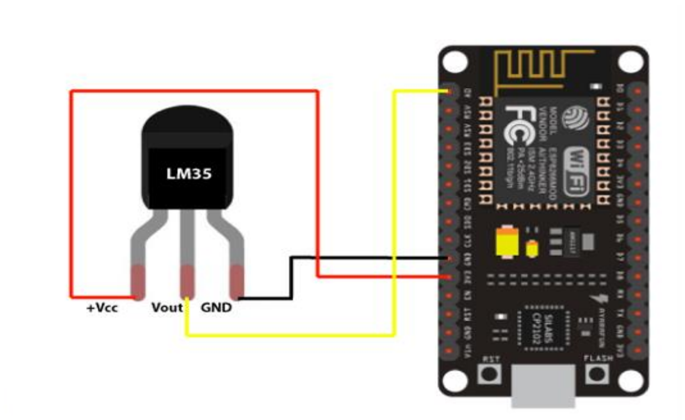
```
14:49:31.549 -> 33.87
14:49:32.528 -> 33.87
14:49:33.528 -> 33.87
14:49:34.550 -> 33.87
14:49:35.546 -> 33.87
14:49:36.540 -> 33.87
14:49:37.532 -> 33.87
14:49:38.525 -> 33.87
14:49:39.553 -> 34.19
14:49:40.541 -> 34.84
14:49:41.538 -> 35.16
14:49:42.530 -> 35.48
14:49:43.524 -> 35.81
```

INTERFACE LM35 WITH NODEMCU AND PRINT DATA ON SERIAL MONITOR

REQUIREMENTS

- NodeMCU x 1
- Micro USB cable x 1
- PC x 1
- Software Arduino IDE(version 1.6.4+)
- LM35 Sensor

CIRCUIT



The **circuit connections** are made as follows:

Pin 1 of the LM35 goes into **+3v** of the NodeMCU.

Pin 2 of the LM35 goes into Analog Pin **A0** of the NodeMCU.

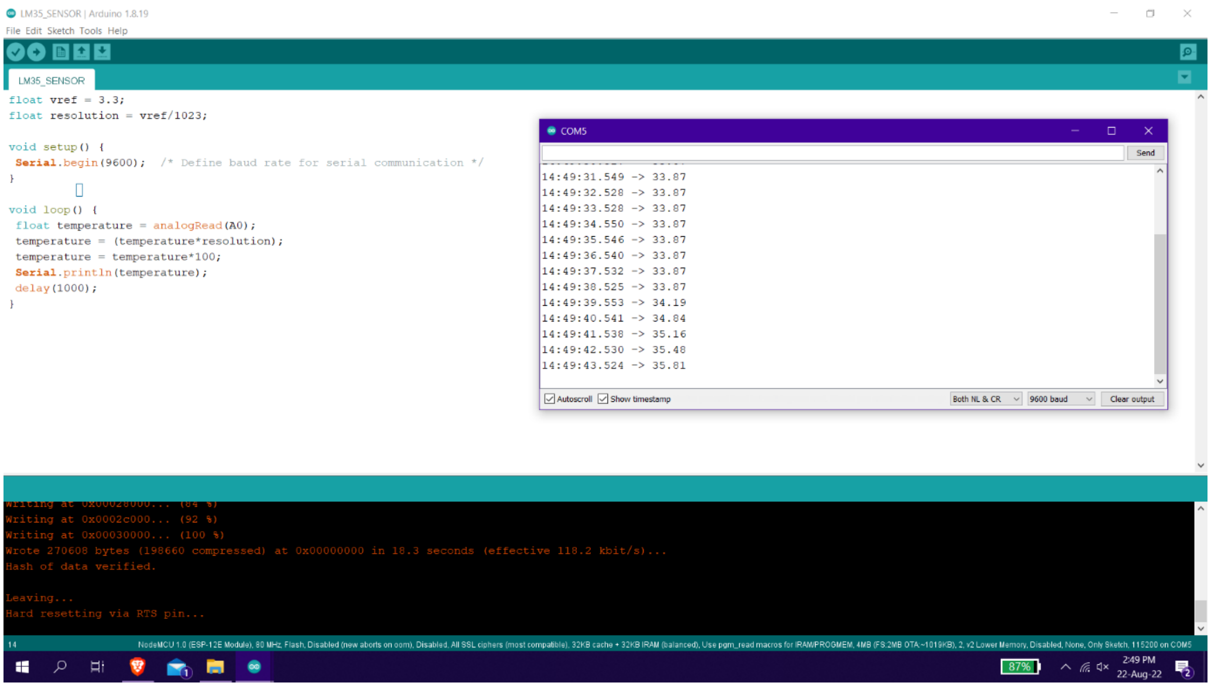
Pin 3 of the LM35 goes into Ground Pin (**GND**) of the NodeMCU.

Before getting the Celsius reading of the temperature. The analog output voltage from LM35 must first be read from the Vout pin of LM35. This will be the raw value divided by 1023 times 3300. It is divided by 1023 because a span of 1023 occupies 3.3v. Here we get the ratio of the raw value to the full span of 1024 and then multiply it by 3300 to get the millivolt value. Since the output pin can give out a maximum of 3.3 volts (1024), 1024 represents the possible range it can give out.

CODE

```
float vref = 3.3;
float resolution = vref/1023;

void setup() {
 Serial.begin(9600);  /* Define baud rate for serial communication */
}

void loop() {
 float temperature = analogRead(A0);
 temperature = (temperature*resolution);
 temperature = temperature*100;
 Serial.println(temperature);
 delay(1000);
}
```

RESULT



```
float vref = 3.3;
float resolution = vref/1023;

void setup() {
 Serial.begin(9600);  /* Define baud rate for serial communication */
}

void loop() {
 float temperature = analogRead(A0);
 temperature = (temperature*resolution);
 temperature = temperature*100;
 Serial.println(temperature);
 delay(1000);
}
```

```
14:49:31.549 -> 33.87
14:49:32.528 -> 33.87
14:49:33.528 -> 33.87
14:49:34.550 -> 33.87
14:49:35.546 -> 33.87
14:49:36.540 -> 33.87
14:49:37.532 -> 33.87
14:49:38.525 -> 33.87
14:49:39.553 -> 34.19
14:49:40.541 -> 34.84
14:49:41.538 -> 35.16
14:49:42.530 -> 35.48
14:49:43.524 -> 35.81
```

```
Writing at 0x00028000... (84 %)
Writing at 0x0002c000... (92 %)
Writing at 0x00030000... (100 %)
Wrote 270608 bytes (198660 compressed) at 0x00000000 in 18.3 seconds (effective 118.2 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

14 NodeMCU 1.0 (ESP-12E Module), 80 MHz, Flash, Disabled (new aborts on oom), Disabled, All SSL ciphers (most compatible), 32KB cache + 32KB IRAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM5