Experiment No: 1

Installation of Node MCU Using Arduino IDE and Basic LED Blinking

Name of the Student: Soham Kolhatkar **Roll No**. 55 **BRANCH**: CSE(AI)

Aim: To install Node MCU using Arduino IDE and to Blink an On-Board LED on Node MCU

Components Required:

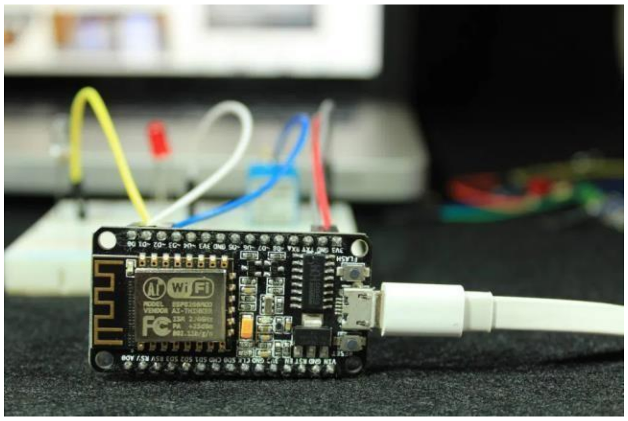
1. Node MCU – 1
2. Micro USB Cable – 1
3. PC/Laptop – 1
4. Connecting Wires
5. Bread Board – 1

Software Required:

Arduino IDE Theory:

Today, IOT applications are on the rise, and connecting objects are getting more and more important. There are several ways to connect objects such as Wi-Fi protocol.

Node MCU is an open source platform based on ESP8266 which can connect objects and let data transfer using the Wi-Fi protocol. In addition, by providing some of the most important features of microcontrollers such as GPIO, PWM, ADC, and etc, it can solve many of the project's needs alone.



The general features of this board are as follows:

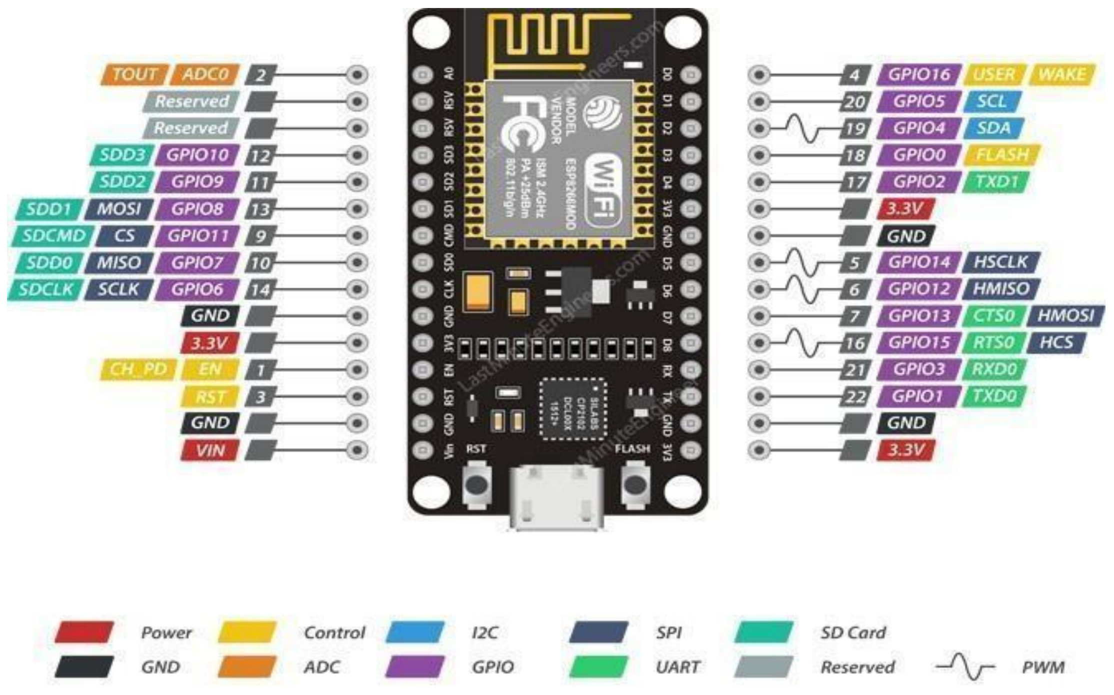
- Easy to use

- Programmability with Arduino IDE or IUA languages
- Available as an access point or station
- Practicable in Event-driven API applications
- Having an internal antenna
- Containing 13 GPIO pins, 10 PWM channels, I2C, SPI, ADC, UART, and 1-Wire

ESP8266 Specifications:

- 11 b/g/n protocol
- Wi-Fi Direct (P2P), soft-AP
- Integrated TCP/IP protocol stack
- Built-in low-power 32-bit CPU
- SDIO 2.0, SPI, UART ESP 8266 Pin Out:

Pin Description:



Power Pin: There are four power pins viz. one VIN pin & three 3.3V pins. The VIN pin can be used to directly supply the ESP8266 and its peripherals, if you have a regulated 5V voltage source. The 3.3V pins are the output of an on-board voltage regulator. These pins can be used to supply power to external components

GND: GND is a ground pin of ESP8266 Node MCU development board.

I2C Pins: I2C Pins are used to hook up all sorts of I2C sensors and peripherals in your project. Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized programmatically, and the clock frequency is 100 kHz at a maximum. It should be noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device

GPIO Pins: ESP8266 Node MCU has 17 GPIO pins which can be assigned to various functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically. Each digital enabled GPIO can be configured to internal pull-up or pull-down, or set to high impedance. When configured as an input, it can also be set to edge-trigger or level-trigger to generate CPU interrupts.

ADC Channel: The Node MCU is embedded with a 10-bit precision SAR ADC. The two functions can be implemented using ADC viz. Testing power supply voltage of VDD3P3 pin and testing input voltage of TOUT pin. However, they cannot be implemented at the same time.

UART Pins: ESP8266 Node MCU has 2 UART interfaces, i.e. UART0 and UART1, which provide asynchronous communication (RS232 and RS485), and can communicate at up to 4.5 Mbps. UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication. It supports fluid control. However, UART1 (TXD1 pin) features only data transmit signal so, it is usually used for printing log.

SPI Pins: ESP8266 features two SPIs (SPI and HSPI) in slave and master modes. These SPIs also support the following general-purpose SPI features:

- 4 timing modes of the SPI format transfer.
- Up to 80 MHz and the divided clocks of 80 MHz.
- Up to 64-Byte FIFO.

SDIO Pins: ESP8266 features Secure Digital Input/output Interface (SDIO) which is used to directly interface SD cards. 4-bit 25 MHz SDIO v1.1 and 4-bit 50 MHz SDIO v2.0 are supported

PWM Pins: The board has 4 channels of Pulse Width Modulation (PWM). The PWM output can be implemented programmatically and used for driving digital motors and LEDs. PWM frequency range is adjustable from 1000 μs to 10000 μs, i.e., between 100 Hz and 1 kHz.

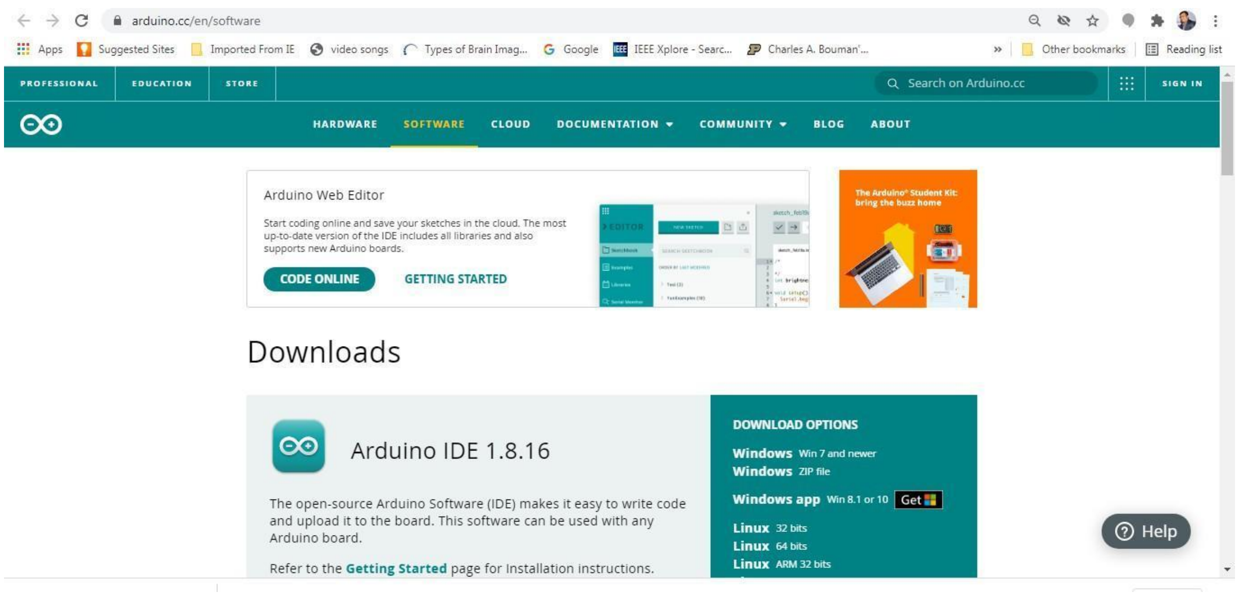
Control Pins: Control Pins are used to control ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin. EN pin – The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power. RST pin – RST pin is used to reset the ESP8266 chip. WAKE pin – Wake pin is used to wake the chip from deep-sleep.

Procedure:

1. **Installing ESP8266 Board in Arduino IDE:**

The ESP8266 community created an add-on for the Arduino IDE that allows you to program the ESP8266 using the Arduino IDE and its programming language.

Step 1: Visit arduino.cc/en/Main/Software for downloading the latest version of Arduino IDE software [Arduino IDE 1.8.16]



Start coding online and save your sketches in the cloud. The most up-to-date version of the IDE includes all libraries and also supports new Arduino boards.

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the **Getting Started** page for Installation instructions.

Step 2: Click on appropriate download link as per the operating system installed on your PC / laptop

Step 3: Download and install it on your PC / LaptopStep 4: Install ESP 8266 Add on in Arduino IDE. Open Arduino IDE ----> File   > Preferences
Step 5: Enter **http://arduino.esp8266.com/stable/package_esp8266com_index.json** into the "Additional button"

Step 6: Open the Boards Manager. Go to Tools---- > Board > Boards Manager…

the “OK”

Step 7: Search for **ESP8266** and press install button for the **ESP8266 by ESP8266 Community**

Step 8: Go to Tools ----> Board ----> ESP 8266 Boards (3.0.2) > Select Node MCU 0.9(ESP-

12 module)

Step 9: Go to Tools ----> Port > Select Serial Port COM3

1. **Testing the Installation [Basic LED Blinking]:**

On-board LED

Most of the ESP8266 development boards have two built-in LEDs. These LEDs are usually connected to GPIO D0 and D4.

Step 1: Connect Node MCU to PC / Laptop with the help of micro USB cable Step 2: Open new Sketch, Go to file > New

Step 3: Write following code in new sketch

```
#define LED_BUILTIN D4


// the setup function runs once when you press reset or power the board void setup() {

pinMode(LED_BUILTIN, OUTPUT); // initialize digital pin LED_BUILTIN as an 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

}
```

Step 4: Save the new sketch by appropriate name in a folder on your PC / Laptop Step 5: Upload the sketch on Node MCU. Go to Sketch   > Upload

Step 6: Observe the output [Blinking LED]

Practice:

1. Change the on and off time of LED
2. Upload a sketch for dancing on board LEDs [one on and other off continuously]

CODE:

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

Conclusion

We can turn the LED on and off at precise intervals, creating a blinking effect. The combination of digital output control, current limiting, and timing functions allows us to implement this simple yet fundamental electronics project using ardiuno and its ide.

PHOTO: