Lab Activity – 1

PREPARED BY

Niharika Sharma (2022BTech064)

SUBMITTED TO

Mr. Divyanshu Jain

Department of Computer Science Engineering
Institute of Engineering and Technology
JK Lakshmipat University, Jaipur

12th August, 2024

CONTENTS

About NodeMCU:

NodeMCU is a highly versatile open-source development board and firmware primarily used for IoT (Internet of Things) projects. It is built around the ESP8266 Wi-Fi module, which is renowned for its integration of a full TCP/IP stack and microcontroller capabilities. This combination allows NodeMCU to perform as a standalone device that can connect to the internet and execute various applications, ranging from simple sensor monitoring to complex automation systems.

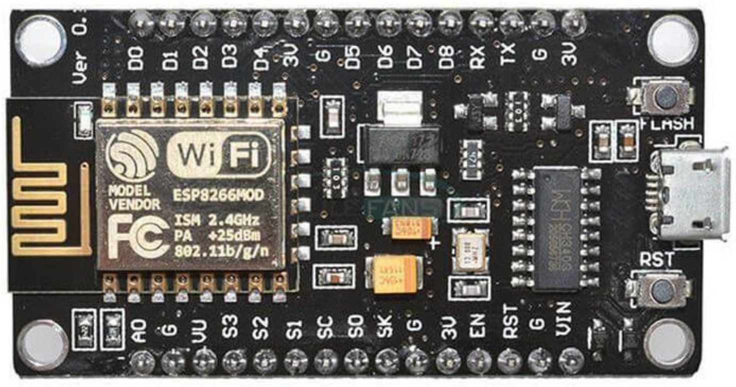


Fig 1.1 Node MCU

Hardware Specifications

The NodeMCU board is compact yet powerful, featuring a 32-bit microcontroller based on the Tensilica Xtensa LX106 core. This microcontroller operates at a clock frequency of 80 MHz, with the capability to reach up to 160 MHz for more demanding tasks. The board includes 4 MB of flash memory, which provides ample storage for firmware and user programs.

Pin Configuration

- **GPIO Pins:** NodeMCU offers multiple General-Purpose Input/Output (GPIO) pins that can be used to interface with external devices such as LEDs, sensors, motors, and more.
- **Analog Input:** It includes one analog input pin, which is useful for connecting analog sensors, such as temperature or light sensors.

- **PWM Outputs:** Pulse Width Modulation (PWM) outputs are available for controlling devices like servos or dimming LEDs.

Applications

- **Home Application**
  NodeMCU can be used to control home appliances remotely via a smartphone or a web application. For example, it can switch lights on or off, control a thermostat, or manage a security system.

- **Remote Monitoring**
  By interfacing with sensors, NodeMCU can monitor environmental conditions such as temperature, humidity, or air quality. The data can be sent to a cloud server for real- time monitoring or historical analysis.

- **Data Logging**
  NodeMCU can log data from sensors and store it locally or send it to a remote server for further processing and analysis. This is particularly useful in applications like weather stations or industrial monitoring.

Cost and Availability

NodeMCU is highly affordable, which makes it accessible to a wide range of users, from hobbyists to professional developers. Its low cost, combined with its powerful features, has contributed to its widespread adoption in the IoT space.

Activity 1: WAP for internal, external led blinking on NodeMCU.

i. Aim

The aim of this activity is to develop a simple program to control the blinking of both internal and external LEDs using a NodeMCU

ii. Component Used

- **NodeMCU**: A microcontroller with built-in Wi-Fi and GPIO pins
- **External LED**: A light-emitting diode to be connected externally.
- **Breadboard**: For making circuit connections.
- **Resistor:** To limit the current through the external LED and protect it.
- **Jumper Wires**: For connecting components on the breadboard.

iii. Circuit Diagram

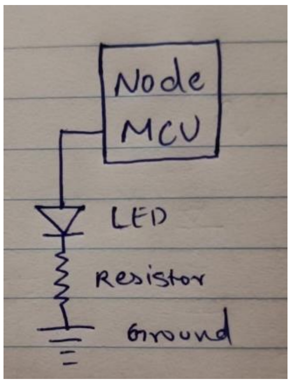
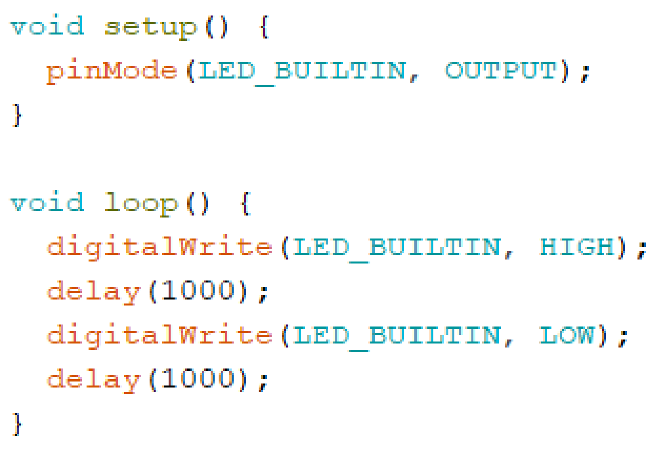


Fig. Circuit Diagram

iv. Theory

The NodeMCU is a versatile microcontroller with built-in Wi-Fi capabilities, making it ideal for IoT projects. It has several GPIO pins that can be configured as either inputs or outputs to interact with external components like LEDs, sensors, and more. In this activity, we use the GPIO pins to control both an internal LED (at GPIO02) and an external LED (at GPIO05). The internal LED, being in an active-low configuration, turns on when the pin is set to ‘LOW’ and off when set to ‘HIGH’. The external LED is connected through a current-limiting resistor to protect it from excessive current. Controlling LEDs using GPIO pins demonstrates fundamental microcontroller operations, such as toggling digital outputs to create visual indicators. This basic LED blinking task is often a starting point for learning how to interact with hardware through programming. The knowledge gained here can be applied to more complex tasks, such as controlling multiple devices, creating timing-based sequences, or integrating sensors for smart systems. Understanding how to program and manage GPIO pins is essential for developing a wide range of embedded systems and IoT applications.

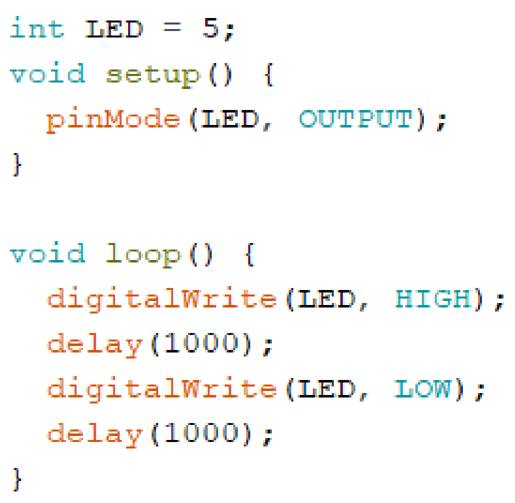
v. Code



```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

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

Fig. Code for blinking Built-in LED



```
int LED = 5;
void setup() {
  pinMode(LED, OUTPUT);
}

void loop() {
  digitalWrite(LED, HIGH);
  delay(1000);
  digitalWrite(LED, LOW);
  delay(1000);
}
```

Fig. Code for blinking external LED connected to D0

vi.

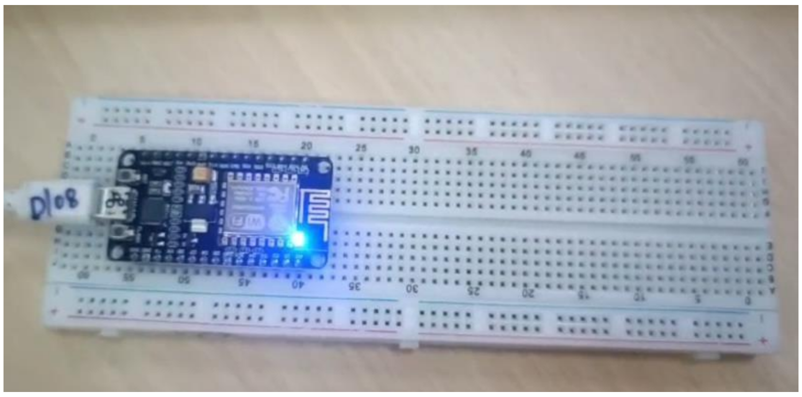


Fig. Built-in LED Blinking

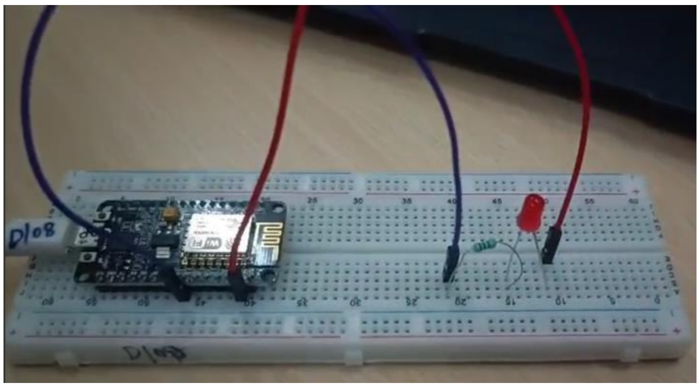


Fig. External-LED set to LOW

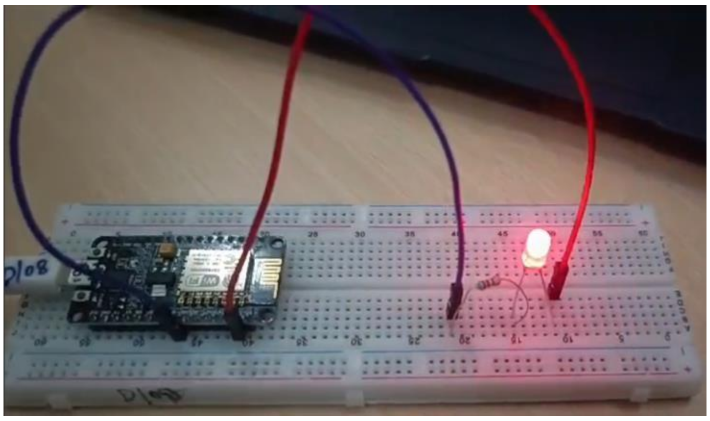


Fig. External- LED set to High

vii. Learning

Through this activity, you will have learned:

- How to configure GPIO pins on the NodeMCU as output.
- How to control an internal LED connected to a specific GPIO pin.
- How to set up and control an external LED using a GPIO pin.

Activity 2: WAP for LED Fading on NodeMCU

i. Aim

The aim of this experiment is to create a program for the NodeMCU microcontroller that allows an LED to gradually fade in and out.

ii. Components Used

- NodeMCU (ESP8266)
- LED (Light Emitting Diode)
- Resistor (220Ω)
- Breadboard
- Jumper wires

iii. Circuit Diagram

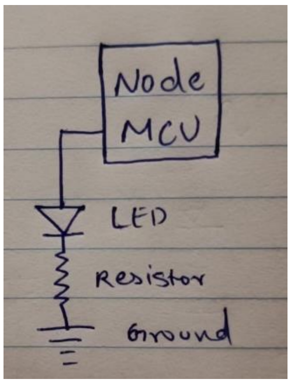
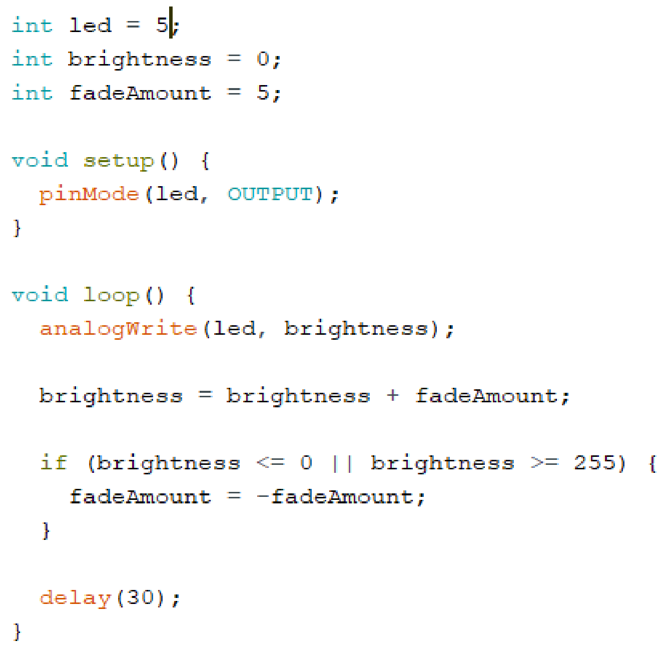


Fig. Circuit Diagram

iv. Theory

Pulse Width Modulation (PWM) is a powerful technique used to simulate analog output by manipulating digital signals. In essence, PWM involves switching a digital signal between high (on) and low (off) states at a very fast rate. By adjusting the proportion of time that the signal remains high within each cycle, the effective power delivered to a device, such as an LED, can be controlled. This technique allows digital microcontrollers, like the NodeMCU, which typically operate in binary (on/off) modes, to create effects that mimic analog behavior, such as varying the brightness of an LED. The key parameter in PWM is the duty cycle, which represents the percentage of one cycle during which the signal is high. A higher duty cycle means the signal is on for a greater portion of the cycle, resulting in a brighter LED. Conversely, a lower duty cycle means the signal is off more often, dimming the LED. By continuously adjusting the duty cycle in a smooth manner, the NodeMCU can create the appearance of the LED fading in and out, providing a simple yet effective way to control brightness without the need for actual analog output capabilities.

v. Code



```
int led = 5;
int brightness = 0;
int fadeAmount = 5;

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {
  analogWrite(led, brightness);

  brightness = brightness + fadeAmount;

  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }

  delay(30);
}
```

Fig. Code for fading LED

vi. Result

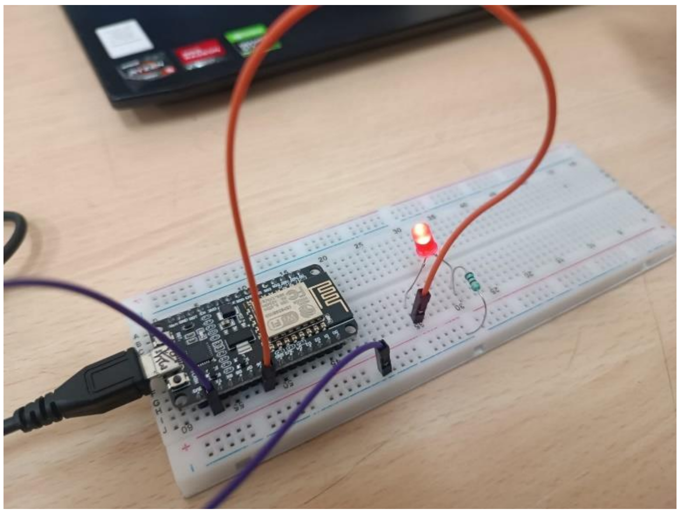


Fig. LED Fading

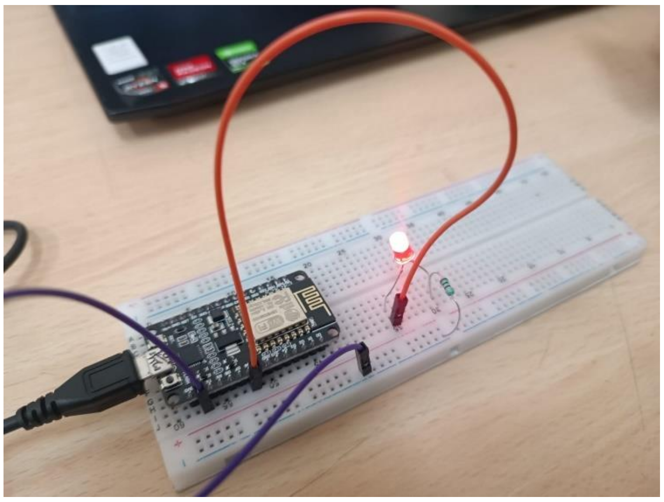


Fig. LED Glowing 1

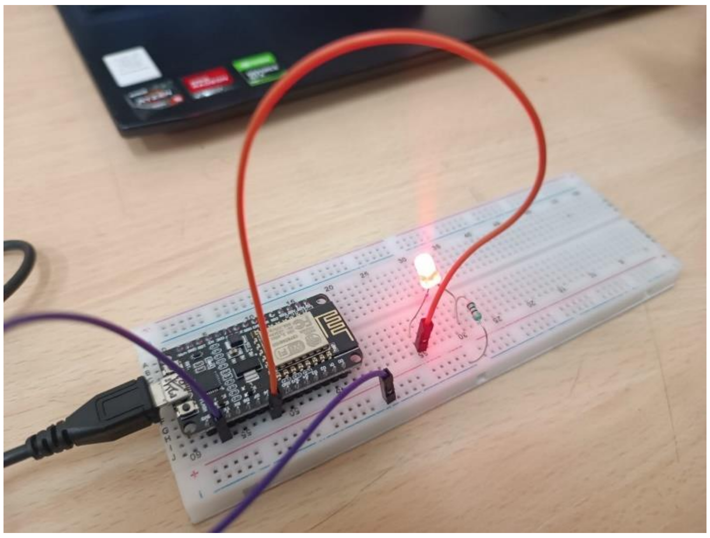


Fig. LED Glowing 2

vii. Learning

Through this experiment, you have learned:

- How to set up a simple circuit with an LED using the NodeMCU.
- The basics of Pulse Width Modulation (PWM) and how it can be used to control the brightness of an LED.
- How to write and upload a basic Arduino sketch to the NodeMCU for controlling hardware components.
- The importance of controlling the delay between LED brightness changes to achieve a smooth fading effect.

Activity 3: WAP for LED control using Switch in pull down and pull up configuration on NodeMCU

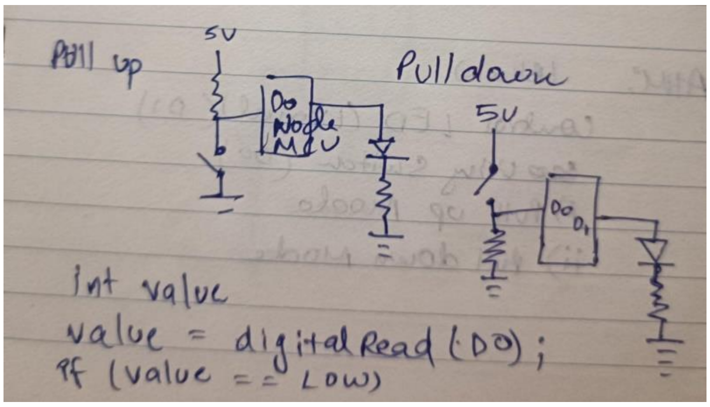
i. Aim

To control an LED using a switch connected to a NodeMCU, demonstrating both pull-up and pull-down resistor configurations.

ii. Components Used

- NodeMCU (ESP8266)
- LED (5mm)
- Resistor
- Push Button Switch
- Breadboard and Jumper Wires

iii. Circuit Diagram



```
int value
value = digitalRead (D0);
if (value == LOW)
```

Fig. Pull up Circuit Diagram

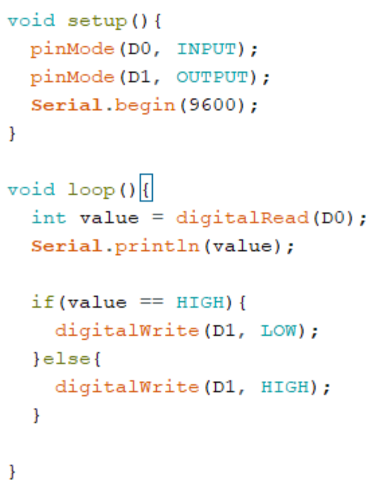
iv. Theory

Pull-Up Configuration: In this setup, the pull-up resistor is connected between the GPIO pin and VCC (3.3V). The resistor ensures that the GPIO pin is normally at a high voltage level (3.3V) when the switch is open, as there is no direct path to ground. When the switch is closed, it creates a direct connection between the GPIO pin and

GND, causing the pin to read a low voltage level (0V). This configuration is useful for ensuring that the GPIO pin has a known high state when the switch is not pressed, preventing it from floating and picking up random noise.

Pull-Down Configuration: In this setup, the pull-down resistor is connected between the GPIO pin and GND. The resistor ensures that the GPIO pin is normally at a low voltage level (0V) when the switch is open, as there is no direct path to VCC. When the switch is closed, it creates a direct connection between the GPIO pin and VCC (3.3V), causing the pin to read a high voltage level (3.3V). This configuration helps to keep the GPIO pin in a stable low state when the switch is not pressed, avoiding floating states and potential interference.

v. Code



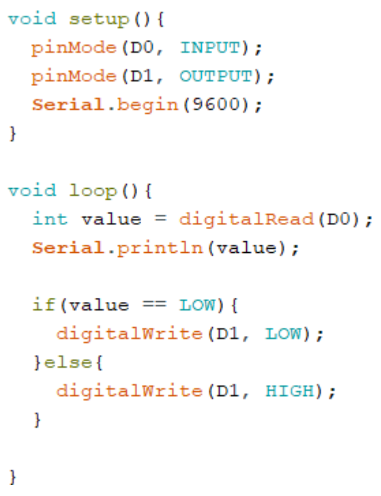
```
void setup(){
  pinMode(D0, INPUT);
  pinMode(D1, OUTPUT);
  Serial.begin(9600);
}

void loop(){
  int value = digitalRead(D0);
  Serial.println(value);

  if(value == HIGH){
    digitalWrite(D1, LOW);
  }else{
    digitalWrite(D1, HIGH);
  }

}
```

Fig. Pull-up Code



```
void setup(){
  pinMode(D0, INPUT);
  pinMode(D1, OUTPUT);
  Serial.begin(9600);
}

void loop(){
  int value = digitalRead(D0);
  Serial.println(value);

  if(value == LOW){
    digitalWrite(D1, LOW);
  }else{
    digitalWrite(D1, HIGH);
  }

}
```

Fig. Pull-down Code

vi. Result

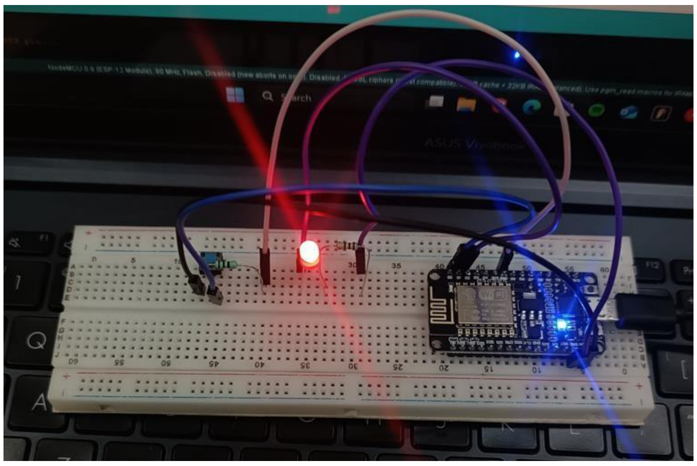
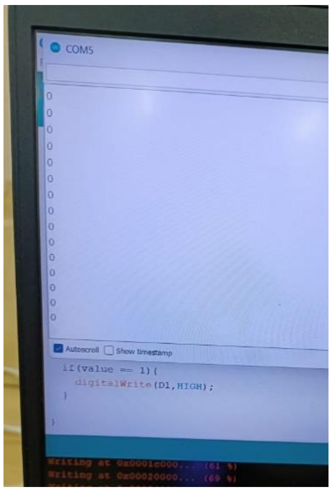


Fig 4.5 Pull-up Circuit



```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
  if(value == 1){
    digitalWrite(D1,HIGH);
  }
}
```

```
Writing at 0x0001c000... (61 %)
Writing at 0x00020000... (69 %)
```

Fig 4.6 Pull-up serial monitor when switch is pressed

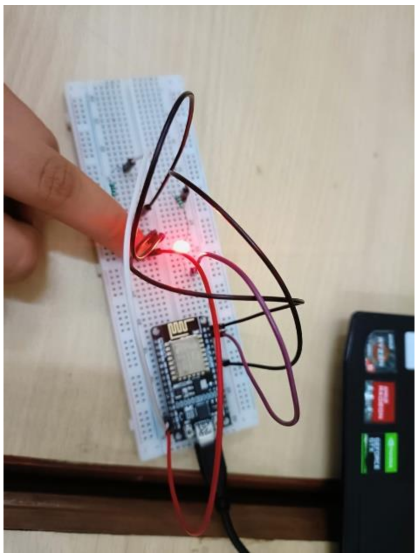


Fig 4.7 Pull-down Circuit

Fig 4.8 Pull-down Serial monitor when switch is pressed

vii. Learning

- Understanding the role of pull-up and pull-down resistors in digital circuits.
- Learning how to configure GPIO pins on the NodeMCU for reading different switch states.

Activity 4: WAP for LED control/fading using Potentiometer

i. Aim

To create a circuit that controls the brightness of an LED using a potentiometer.

ii. Components Used

- NodeMCU
- LED
- Potentiometer
- Resistor
- Breadboard and jumper wires

iii. Circuit Diagram

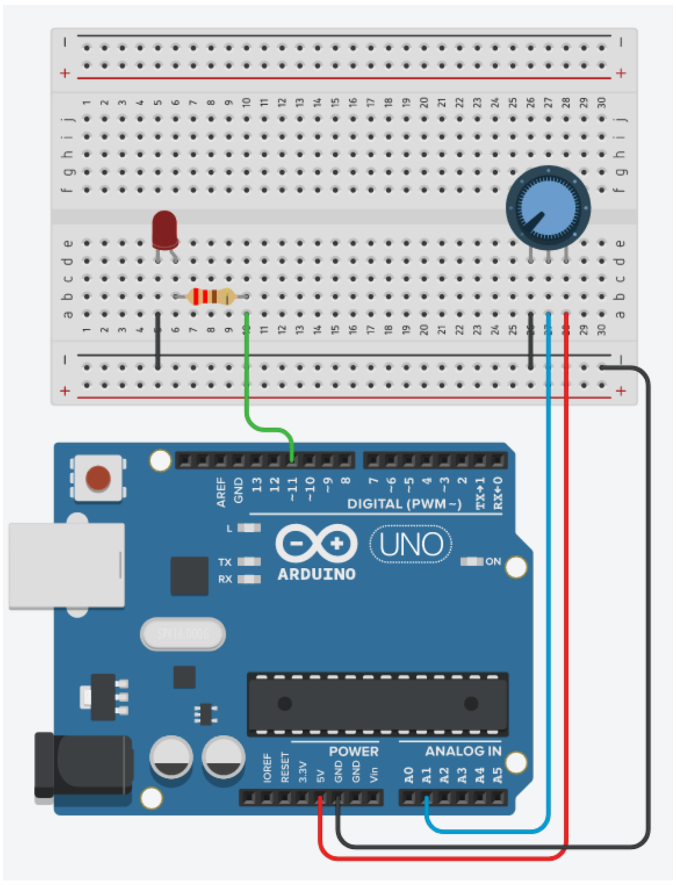


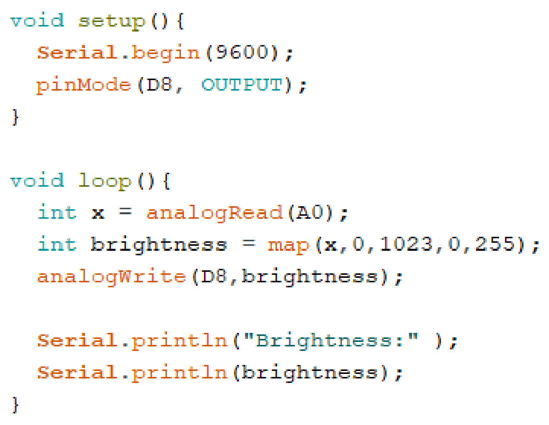
Fig 5.1 Circuit Diagram for controlling LED brightness using potentiometer

iv. Theory

The potentiometer is used as a variable resistor to adjust the voltage output on its wiper terminal, which the NodeMCU reads as an analog input. The NodeMCU's analog-to-digital converter (ADC) reads this value, which ranges from 0 to 1023.

The NodeMCU can generate PWM signals on its digital pins, which can be used to control the brightness of the LED. The brightness is adjusted by varying the duty cycle of the PWM signal. A higher duty cycle results in a brighter LED, while a lower duty cycle dims the LED.

v. Code



```
void setup(){
  Serial.begin(9600);
  pinMode(D8, OUTPUT);
}

void loop(){
  int x = analogRead(A0);
  int brightness = map(x,0,1023,0,255);
  analogWrite(D8,brightness);

  Serial.println("Brightness:" );
  Serial.println(brightness);
}
```

Fig 5.2 Code for changing brightness using potentiometer

vi. Result

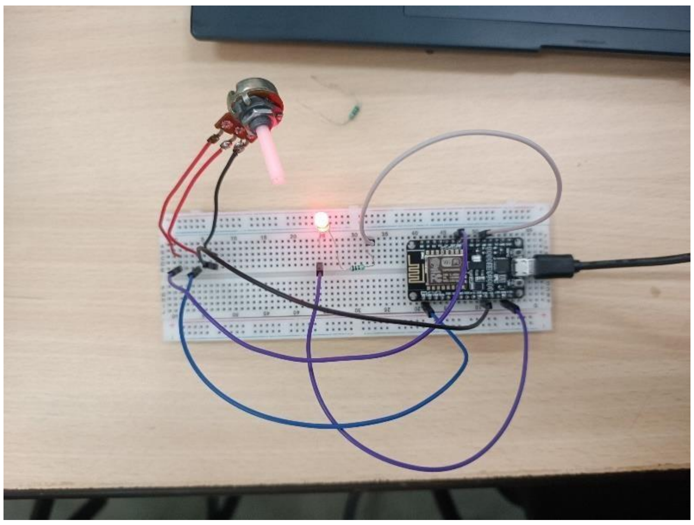


Fig 5.3 Circuit for controlling brightness using potentiometer

vii. Learning

- **Working with NodeMCU:** This project helps you understand the NodeMCU, a powerful IoT microcontroller based on the ESP8266.
- **PWM Control:** You learn how to generate and use PWM signals with NodeMCU to control devices like LEDs.

Analog Input Handling: The project reinforces how to handle analog inputs and co