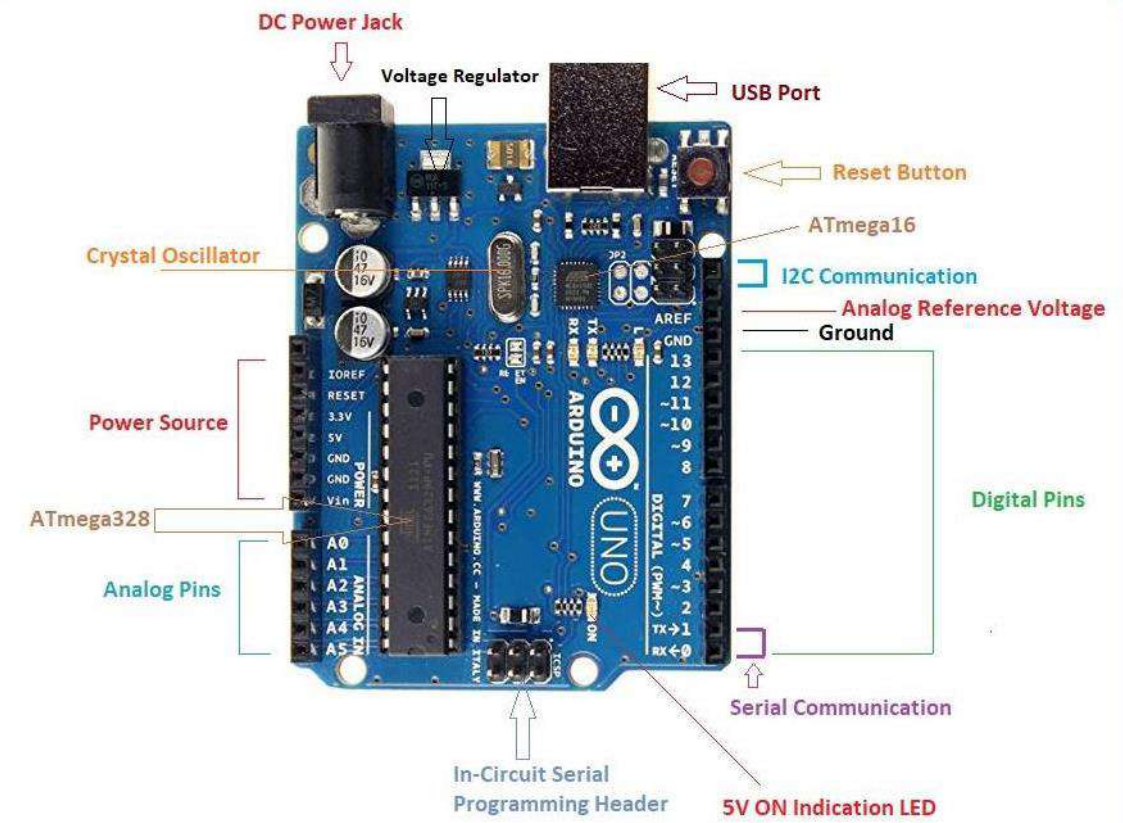
Practical No:-01

Aim:- Study and Install IDE of Arduino

Introduction to Arduino

Arduino is an open-source hardware and software platform designed to make it easy for anyone to create interactive electronic projects. It is widely used by hobbyists, students, educators, and professionals for a variety of applications, from simple DIY projects to complex systems. Arduino has revolutionized the field of electronics and programming by bridging the gap between software and hardware. Its simplicity, affordability, and versatility make it a preferred platform for creating interactive devices that can sense and control the physical world.



History and Evolution of Arduino

1. **Origin**:
    - Arduino was developed in 2005 at the Interaction Design Institute Ivrea in Italy.

- It was created as an affordable tool for students to create digital projects without needing expensive equipment.

2. **Evolution**:
   - Over the years, Arduino has grown from a simple prototyping platform to a global phenomenon, offering a wide range of boards and software tools.
   - It has a thriving community of makers, developers, and educators contributing to its ecosystem.

Types of Arduino Boards

1. **Entry-Level Boards**:
   - **Arduino Uno**: The most popular and beginner-friendly board, featuring the ATmega328P microcontroller.
   - **Arduino Nano**: Compact and suitable for smaller projects.
2. **Advanced Boards**:
   - **Arduino Mega**: Designed for projects requiring more I/O pins and memory.
   - **Arduino Due**: A 32-bit board for advanced applications.
3. **IoT Boards**:
   - **Arduino Nano 33 IoT**: Built for Internet of Things (IoT) applications with Wi-Fi and Bluetooth.
   - **Arduino MKR1000**: Integrates Wi-Fi for connected devices.
4. **Wearable Boards**:
   - **Arduino LilyPad**: Designed for e-textiles and wearable projects.
5. **Specialized Boards**:
   - **Arduino Portenta**: Geared toward industrial and AI applications.
   - **Arduino Leonardo**: Offers built-in USB communication for HID devices like keyboards and mice.

Core Concepts of Arduino

1. **Digital and Analog Pins**:
   - Digital pins can be used for input or output, such as controlling LEDs or reading button presses.

    - Analog pins allow the board to read analog signals (e.g., from temperature sensors) and convert them to digital values.
2. **Microcontroller**:
    - The "brain" of the Arduino board, responsible for processing data and executing the uploaded code.
3. **Power Supply**:
    - Powered via USB or an external power source (e.g., a battery or adapter).
4. **Programming**:
    - Arduino uses a simplified version of C/C++ for programming.
    - Key functions include:
        - **setup()**: Runs once when the board is powered on or reset, used to initialize settings.
        - **loop()**: Repeats continuously, containing the core logic of your program.

Common Sensors and Modules Used with Arduino

1. **Input Sensors**:
    - **Temperature Sensors**: Measure ambient temperature.
    - **Ultrasonic Sensors**: Measure distance.
    - **Light Sensors (LDRs)**: Detect the intensity of light.
2. **Output Actuators**:
    - **LEDs**: Emit light as visual indicators.
    - **Motors**: Drive wheels, fans, or robotic arms.
    - **Speakers**: Produce sound.
3. **Communication Modules**:
    - **Wi-Fi Modules (ESP8266)**: Enable wireless connectivity.
    - **Bluetooth Modules (HC-05)**: Facilitate short-range wireless communication.
    - **GSM/GPS Modules**: Provide cellular connectivity and location tracking.

Advantages of Arduino

1. **Low Cost**: Affordable for beginners and professionals.
2. **Versatility**: Compatible with a vast array of sensors and modules.

3. **Community Support**: A large, active community provides tutorials, forums, and libraries.
4. **Open-Source Ecosystem**: Encourages innovation and customization.
5. **Cross-Disciplinary Applications**: Used in art, science, engineering, and education.

Examples of Arduino Projects

1. **Beginner**:
    - **Blinking LED**: A simple program to make an LED blink.
    - **Temperature Display**: Use a temperature sensor and display data on an LCD.
2. **Intermediate**:
    - **Obstacle-Avoiding Robot**: Build a robot that detects and avoids obstacles.
    - **Home Automation**: Control lights and appliances via a smartphone.
3. **Advanced**:
    - **Weather Station**: Monitor temperature, humidity, and air pressure.
    - **Smart Irrigation System**: Automatically water plants based on soil moisture.

Features of Arduino

Open Source: Arduino's hardware designs and software are open source, encouraging community collaboration.

Ease of Use: Simple to program and set up, making it beginner-friendly.

Variety of Modules: Compatible with various sensors, actuators, and shields for expansion.

Cross-Platform: Works on Windows, macOS, and Linux.

How Arduino Works

- Connect the board to your computer using a USB cable.
- Write code in the Arduino IDE.
- Upload the code to the board.
- Use connected hardware (e.g., sensors, LEDs) to interact with the physical environment.

Getting Started with Arduino

- **Choose a Board:** Select an Arduino board that fits your project needs.

- **Download the IDE:** Install the Arduino IDE from the official website (arduino.cc).
- **Learn the Basics:** Start with simple projects, such as blinking an LED.
- **Expand:** Explore advanced projects using sensors, motors, and communication modules.

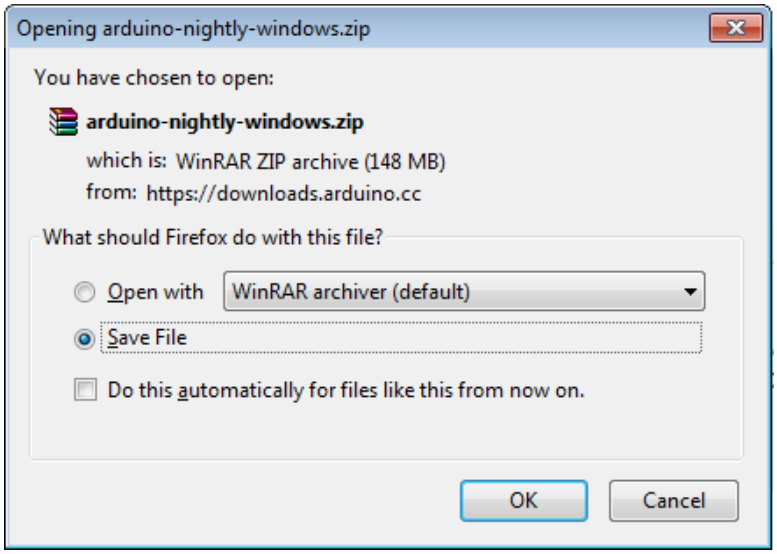
Steps to Get Started with Arduino

Step 1: Gather Your Materials

- Get your Arduino board (e.g., Uno, Mega, Nano) and a USB cable.
    - For **Arduino UNO/Mega**: Use a standard USB cable (A to B plug).
    - For **Arduino Nano**: Use an A to Mini-B cable.

Step 2: Download Arduino IDE

- Visit the Arduino Download Page.
- Choose the IDE version compatible with your operating system (Windows, macOS, or Linux).
- Download and unzip the file.

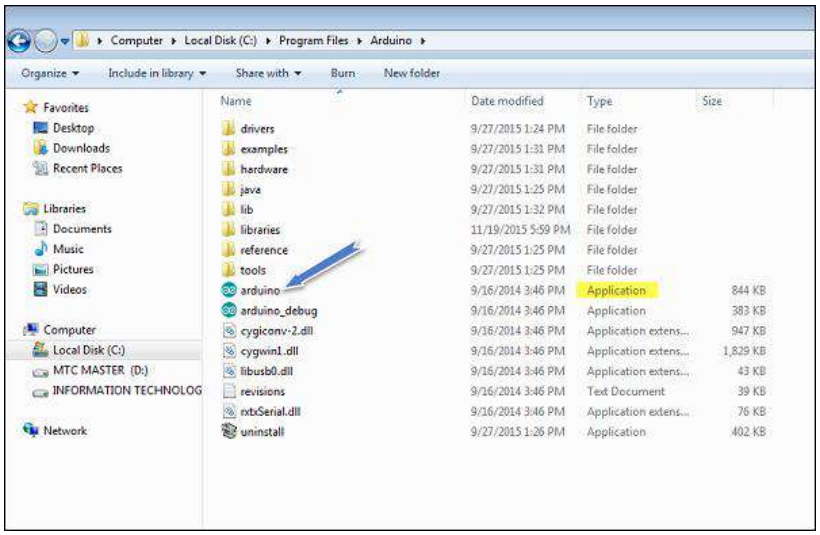


Step 3: Power Up Your Board

- Connect your Arduino to the computer using the USB cable.
- Ensure the green power LED (labeled PWR) is glowing.
- For Arduino Diecimila, set the jumper to USB power mode.

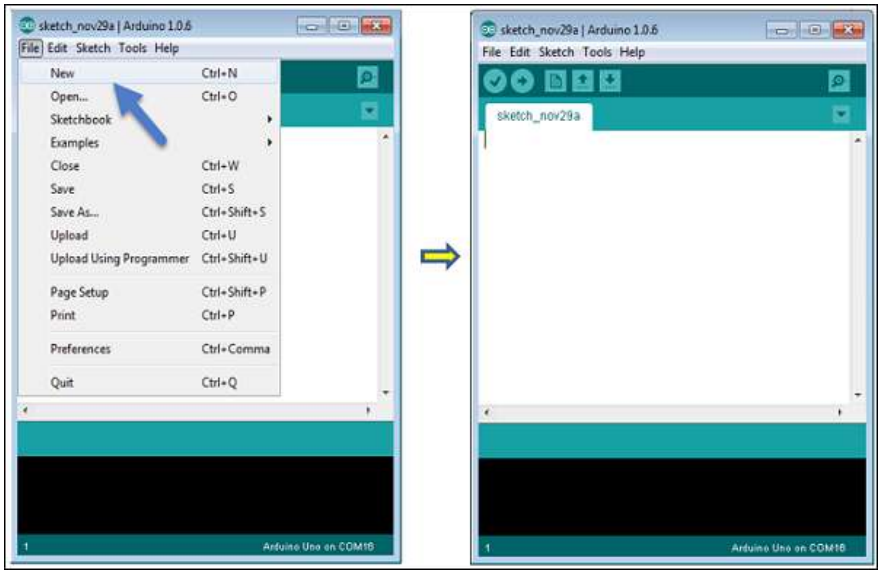
Step 4: Launch Arduino IDE

- Open the IDE by double-clicking the application icon (infinity label).

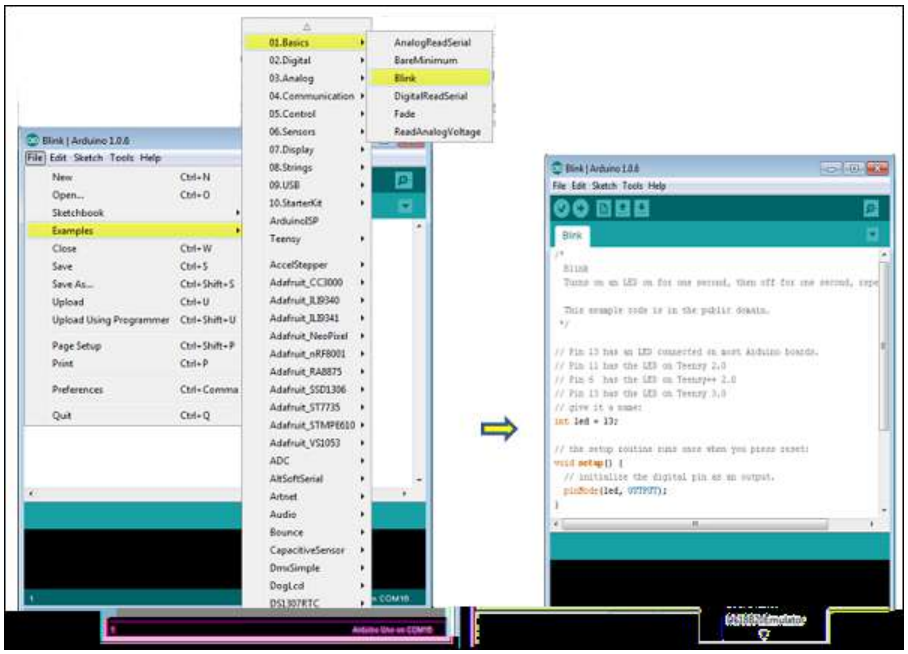


Step 5: Open a Project

- Create a new project: **File → New**, or

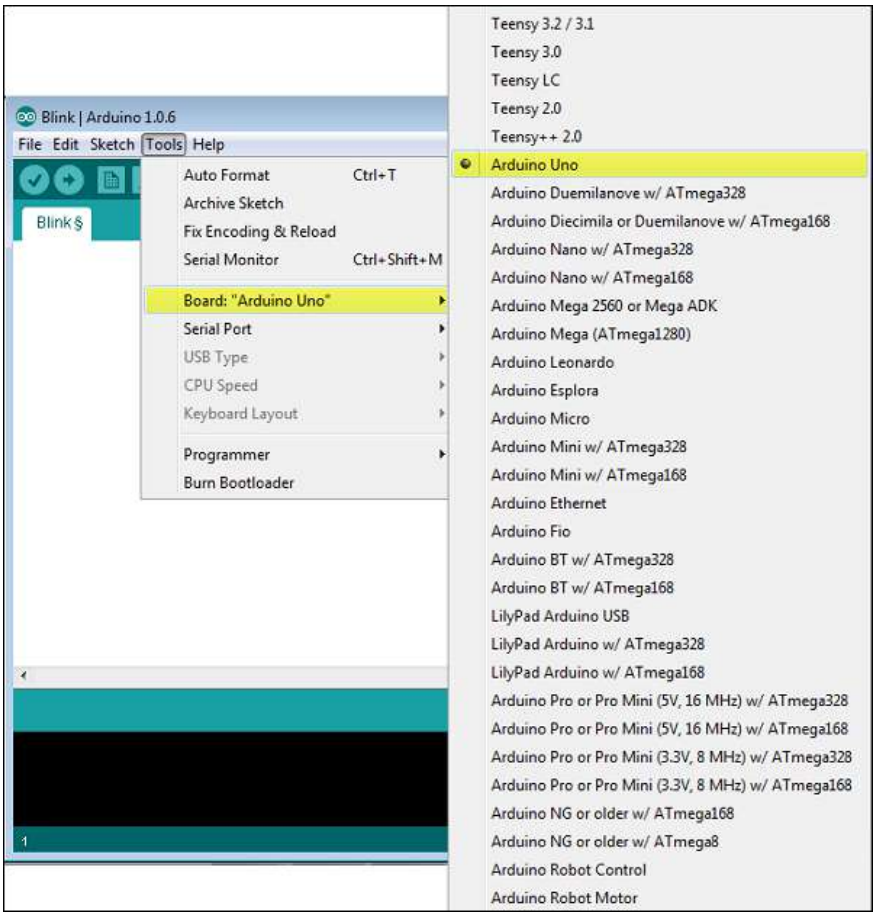


- Open an example project: **File → Example → Basics → Blink**.



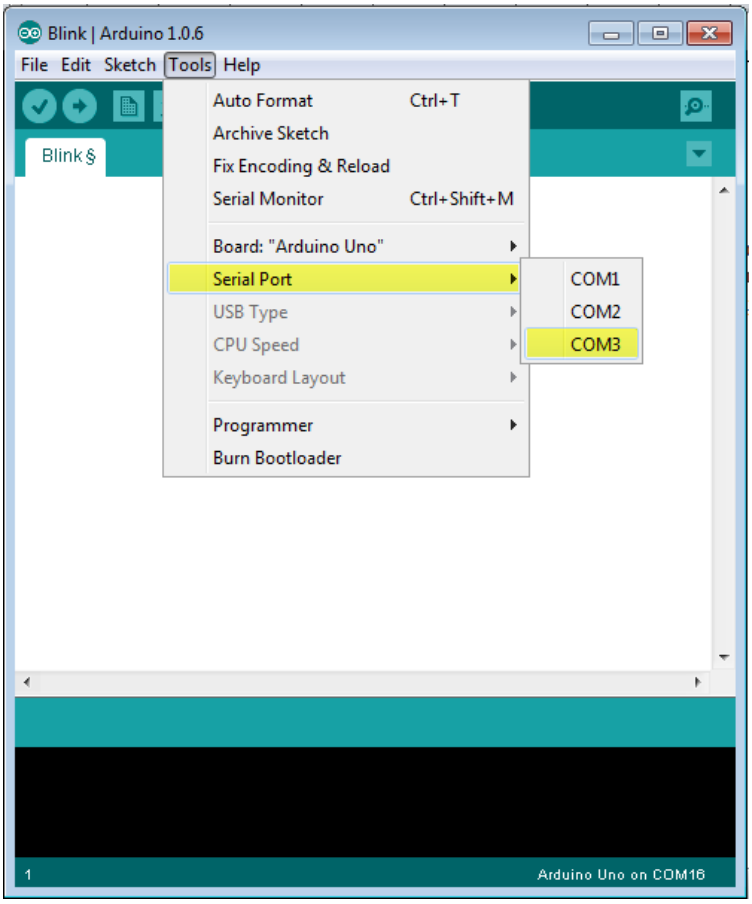
Step 6: Select Your Board

- Go to **Tools → Board**, and choose the board type you are using (e.g., Arduino Uno).



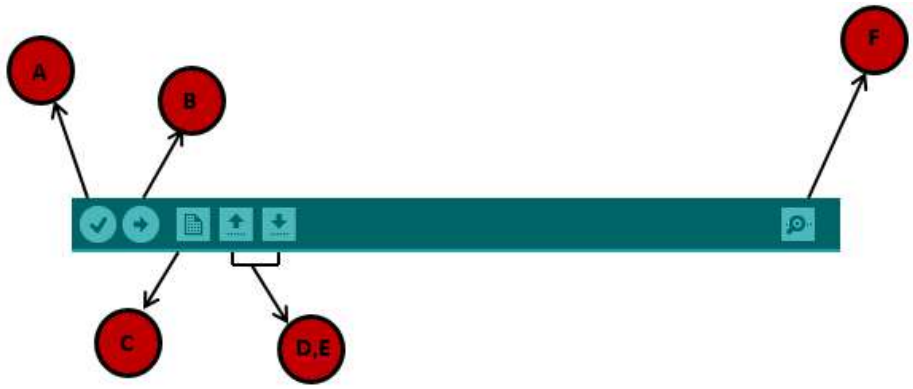
Step 7: Select Your Serial Port

- Go to **Tools → Serial Port**, and select the COM port associated with your Arduino (usually COM3 or higher).



Step 8: Upload the Program

- Write or edit the code in the IDE.
- Click the **Upload** button (arrow icon) to upload the program.
- Look for "Done uploading" in the status bar after a successful upload.



A − Used to check if there is any compilation error.

B − Used to upload a program to the Arduino board.

C − Shortcut used to create a new sketch.

D − Used to directly open one of the example sketch.

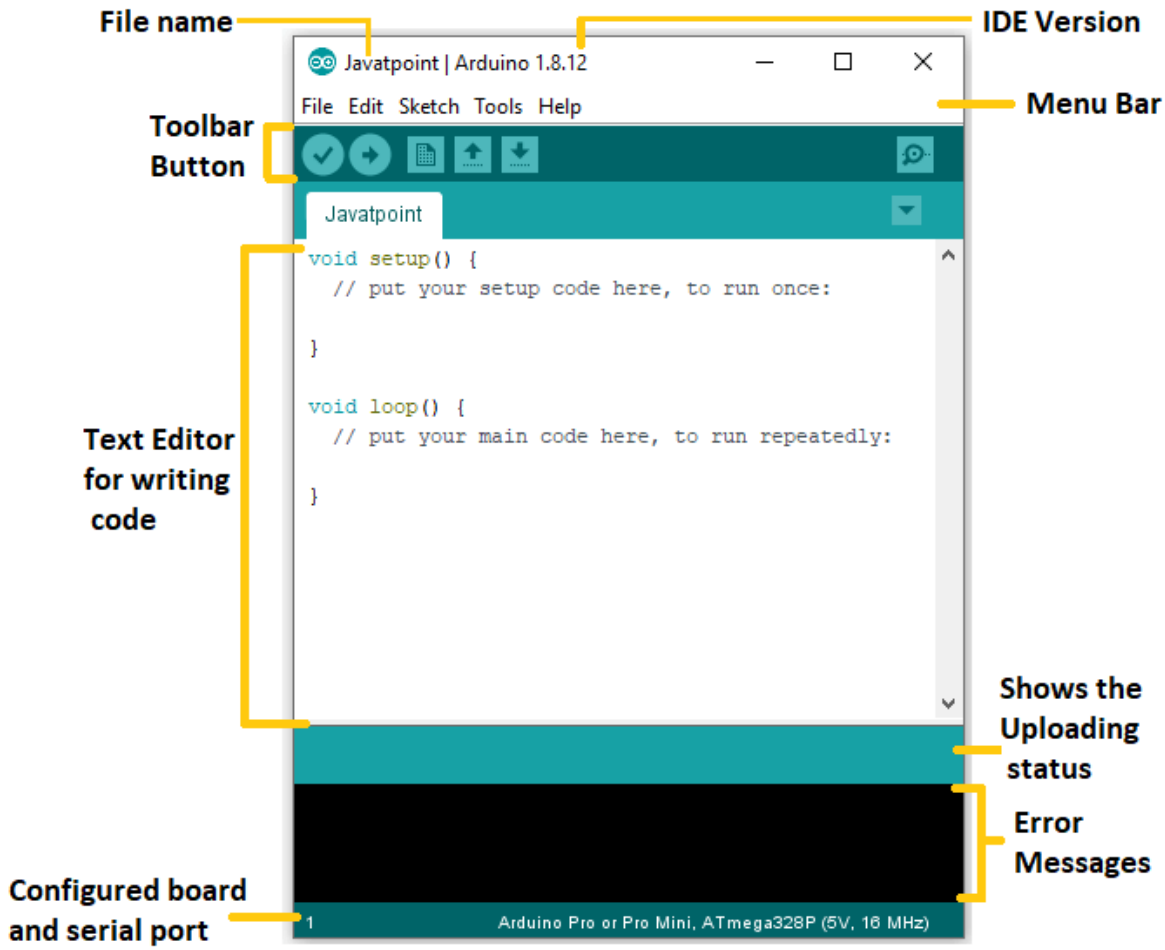
E − Used to save your sketch.

F − Serial monitor used to receive serial data from the board and send the serial data to the board.

We are ready to start coding and experimenting with our Arduino projects!

Introduction to Arduino IDE

The **Arduino IDE** (Integrated Development Environment) is open-source software used to write, compile, and upload code (called sketches, saved with .ino extension) to Arduino boards. It supports **C/C++** and is compatible with Windows, macOS, and Linux.



Key Features

- **Toolbar**:
    - **New**: Create a new sketch.
    - **Open**: Open an existing sketch.
    - **Save**: Save the current sketch.
    - **Upload**: Compile and upload code to the board.
    - **Verify**: Check for code errors.
- **Serial Monitor**: Exchanges data between the board and computer.

Menu Bar Options

- **File**: Open, save, or print sketches; manage examples.
- **Edit**: Modify text (Undo, Redo, Cut, Copy, Paste).
- **Sketch**: Verify, compile, or include libraries.
- **Tools**: Select board/port, manage libraries, use Serial Monitor, or Burn Bootloader.
- **Help**: Access documentation and troubleshooting resources.

RESULT:

In this practical, we studied and installed the Arduino IDE, which allows programming Arduino boards using a simplified version of C/C++. We explored key features of the IDE, including creating and uploading sketches, using libraries, and interacting with hardware through the Serial Monitor.

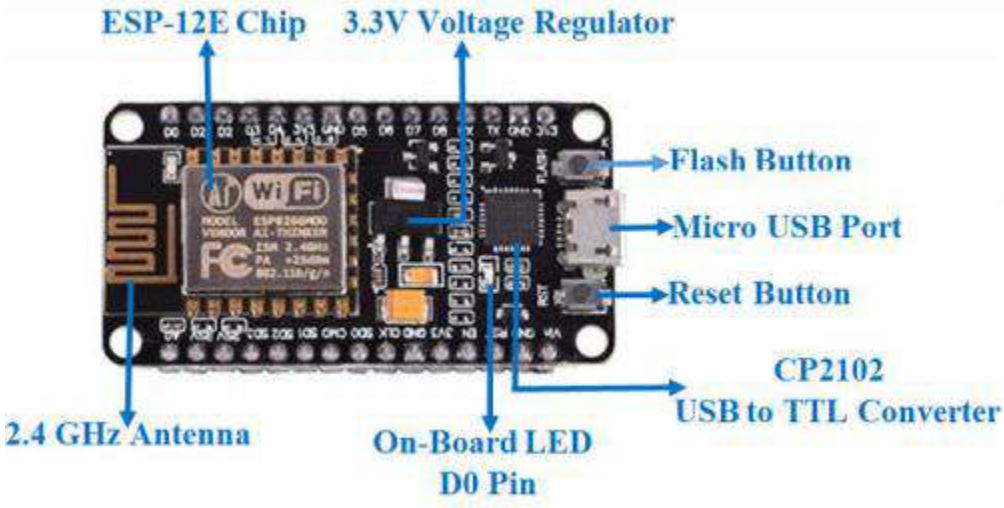
Practical No:-02

Aim:- Study and Install IDE of NodeMCU

NodeMCU is an open-source IoT platform built on the **ESP8266 Wi-Fi SoC (System on Chip)**. Designed for **Internet of Things (IoT)** applications, it integrates hardware and software features to simplify the process of building connected devices.

1. What is NodeMCU?

NodeMCU stands for **Node Microcontroller Unit**. It is an open-source development board and firmware based on the popular **ESP8266** module, making it a cost-effective and powerful choice for IoT applications. It supports both **Lua scripting language** and **Arduino IDE**, allowing developers to program it with ease.



2. Key Features of NodeMCU

- **Built-in Wi-Fi**: The ESP8266 chip provides robust Wi-Fi connectivity, enabling the device to connect to a network or act as an access point.
- **Open-Source**: NodeMCU is fully open-source, allowing the community to contribute to and enhance its capabilities.
- **Small Form Factor**: The compact size makes it ideal for embedding in a variety of projects.

- **High Compatibility**: Supports programming through both Lua scripting language and the Arduino IDE.
- **Integrated GPIO Pins**: Provides access to GPIO (General Purpose Input/Output) pins for interfacing with sensors, LEDs, and other peripherals.
- **Additional Communication Protocols**:
    - **PWM (Pulse Width Modulation)** for controlling devices like motors.
    - **I2C (Inter-Integrated Circuit)** for interfacing with sensors and other modules.
    - **SPI (Serial Peripheral Interface)** for high-speed communication.
    - **ADC (Analog-to-Digital Converter)** for reading analog signals.

3. NodeMCU Firmware

The NodeMCU firmware runs on the ESP8266 Wi-Fi SoC and is based on the **Lua programming language**, offering:

- Easy scripting for IoT projects.
- Ready-to-use libraries for common functions like HTTP requests, MQTT communication, and GPIO control.
- Open-source development, allowing users to modify or extend its functionality.

4. Applications of NodeMCU

NodeMCU is widely used in IoT projects due to its affordability, compact design, and robust features. Common applications include:

- **Home Automation**: Controlling lights, fans, and other appliances remotely.
- **Weather Monitoring**: Connecting to sensors to measure temperature, humidity, and pressure.
- **Smart Agriculture**: Monitoring soil moisture and automating irrigation systems.
- **Industrial IoT**: Monitoring machinery and automating processes.
- **Wearables**: Integration into fitness trackers and other smart devices.

5. Advantages of NodeMCU

- **Affordable**: Low cost compared to other IoT development boards.
- **Wi-Fi Connectivity**: Simplifies IoT projects with built-in Wi-Fi.

- **Versatility**: Suitable for a wide range of applications, from simple to advanced.
- **Community Support**: Large community of developers offering tutorials, libraries, and forums for support.
- **Power Efficiency**: Optimized for low-power applications.

6. Programming NodeMCU

NodeMCU can be programmed using:

1. **Lua Language**: A lightweight scripting language, easy to learn for beginners.
2. **Arduino IDE**: Allows the use of C/C++ programming and Arduino libraries, making it accessible to Arduino enthusiasts.

Steps to Program NodeMCU:

1. Install the required software (e.g., Arduino IDE or Lua Loader).
2. Select the NodeMCU board and configure the COM port in the IDE.
3. Write the program (sketch or script) for your project.
4. Upload the program to NodeMCU via a USB connection.

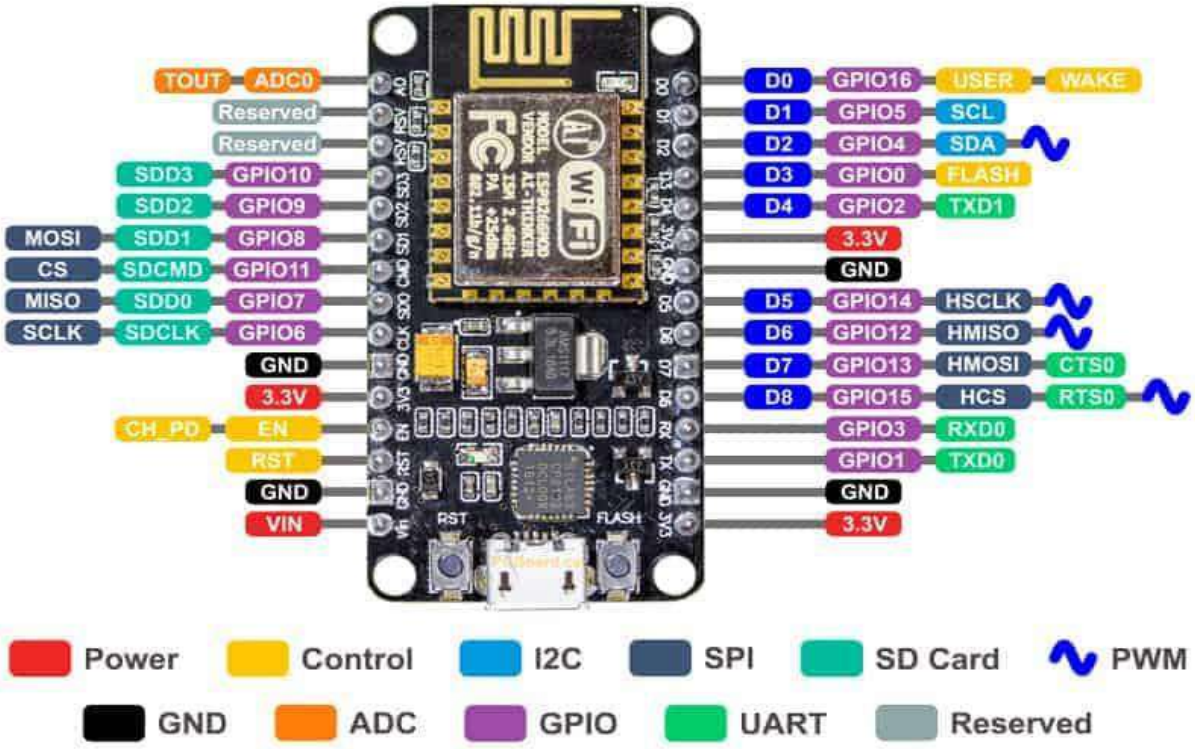
7. Specifications of NodeMCU

- **Microcontroller**: ESP8266
- **Operating Voltage**: 3.3V
- **Wi-Fi Standards**: 802.11 b/g/n
- **Flash Memory**: 4MB
- **GPIO Pins**: 16 (available for interfacing)
- **Analog Input Pins**: 1 (max input voltage 3.3V)
- **Clock Speed**: 80 MHz (can be overclocked to 160 MHz)
- **Power Supply**: USB or external (via Vin and GND pins)

8. Limitations of NodeMCU

- Limited GPIO pins compared to larger microcontrollers.
- Single ADC pin restricts the ability to read multiple analog sensors.
- Requires careful handling due to its 3.3V logic level (incompatible with 5V peripherals without level shifting).

NodeMCU ESP8266 Pinout Diagram



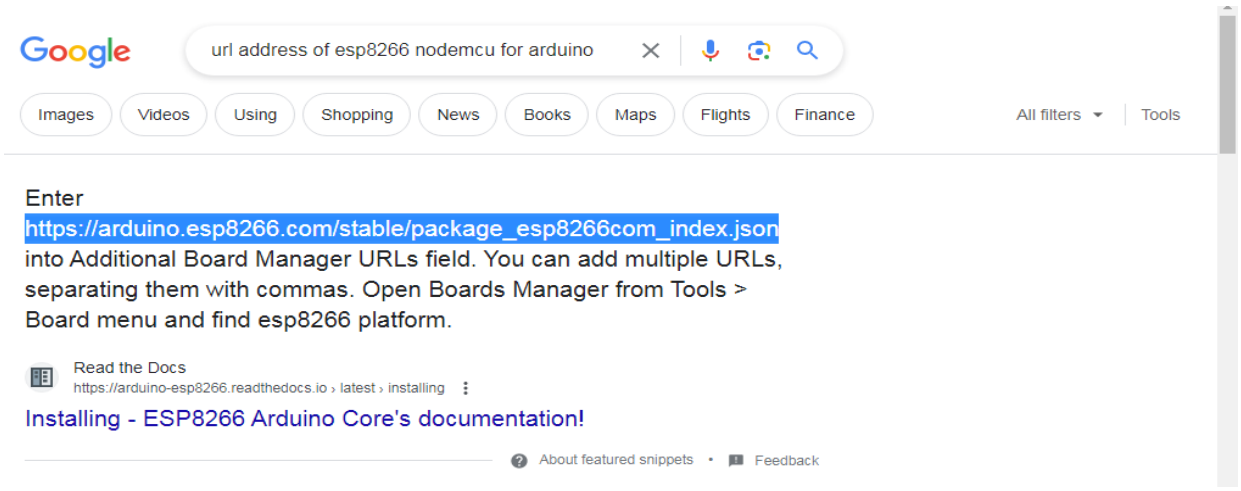
Pin Name	Function/Description
Vin	Input voltage pin (external power supply, 5V input).
3V3	3.3V output from the voltage regulator.
GND	Ground pins (there are multiple GND pins on the board).
D0 (GPIO16)	General-purpose digital I/O. Can also be used as WAKE pin.
D1 (GPIO5)	General-purpose digital I/O, supports I2C (SCL).
D2 (GPIO4)	General-purpose digital I/O, supports I2C (SDA).
D3 (GPIO0)	General-purpose digital I/O. Used for flash mode selection.
D4 (GPIO2)	General-purpose digital I/O. Connected to the built-in LED.
D5 (GPIO14)	General-purpose digital I/O, supports SPI (SCK).
D6 (GPIO12)	General-purpose digital I/O, supports SPI (MISO).
D7 (GPIO13)	General-purpose digital I/O, supports SPI (MOSI).
D8 (GPIO15)	General-purpose digital I/O, supports SPI (CS).
A0 (ADC0)	Analog input pin. Can measure voltages from 0–3.3V.
RX (GPIO3)	UART RX pin for serial communication.
TX (GPIO1)	UART TX pin for serial communication.
RST	Reset pin for restarting the board.

Important Points

1. **Power Supply**:
    - Use the **Vin pin** to power NodeMCU externally (5V).
    - Use the **3V3 pin** for powering peripherals that require 3.3V.
2. **Built-In LED**:
    - The built-in LED is connected to **D4 (GPIO2)** and can be controlled programmatically.
3. **I2C Communication**:
    - Default I2C pins: **SDA (D2, GPIO4)** and **SCL (D1, GPIO5)**.
4. **SPI Communication**:
    - SPI pins include **MOSI (D7)**, **MISO (D6)**, **SCK (D5)**, and **CS (D8)**.
5. **UART Communication**:
    - Serial communication pins: **TX (GPIO1)** and **RX (GPIO3)**.
6. **Analog Input**:
    - The **A0 pin** is used for reading analog signals. Voltage must be limited to 0–3.3V (use a voltage divider for higher voltages).

Steps to Install NodeMCU with CP2102 Driver

1. Search for **"URL address for ESP8266 NodeMCU for Arduino IDE"** on google.



Copy the link from here.

2. Open arduino IDE, click on file

3. Select preferences then select additional board manager and paste the URL here and click on OK.
4. Now select Tools -> Board ->Board Manager -> Search for ESP8266 by ESP8266 community and install it.

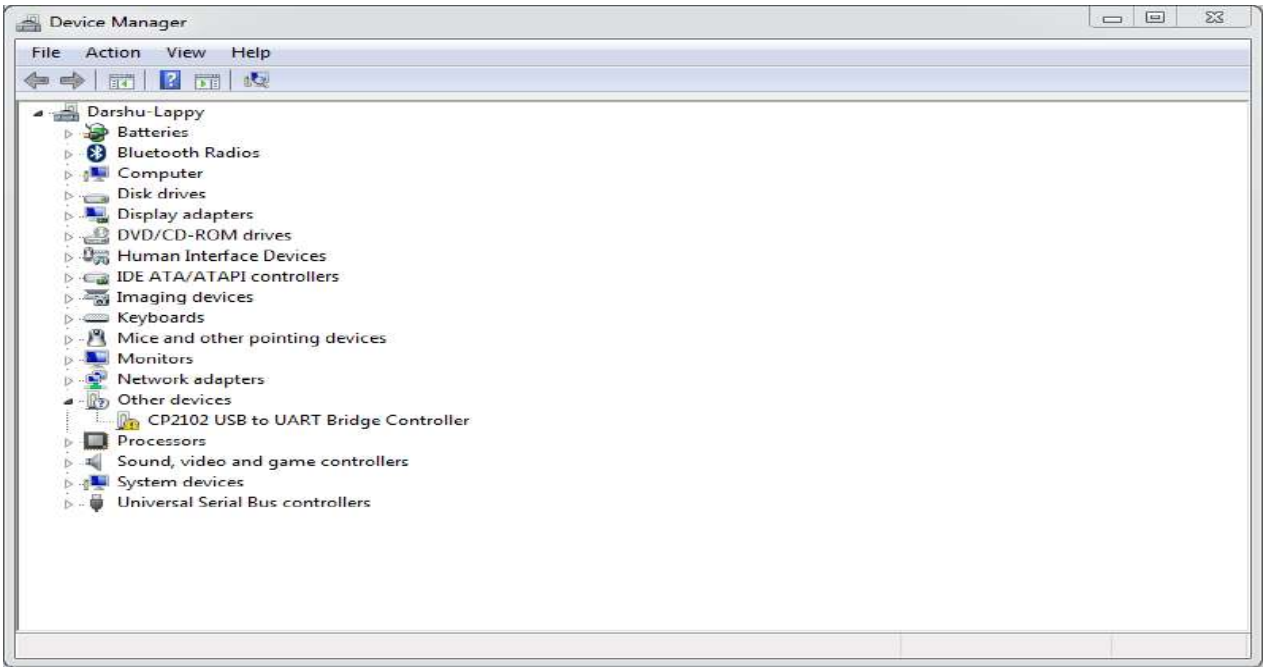
Steps to install CP2102 driver

1. Search for **CP2102 driver download** on google. Or click on the link given below. https://www.dnatechindia.com/cp-2102-driver-download-installation.html

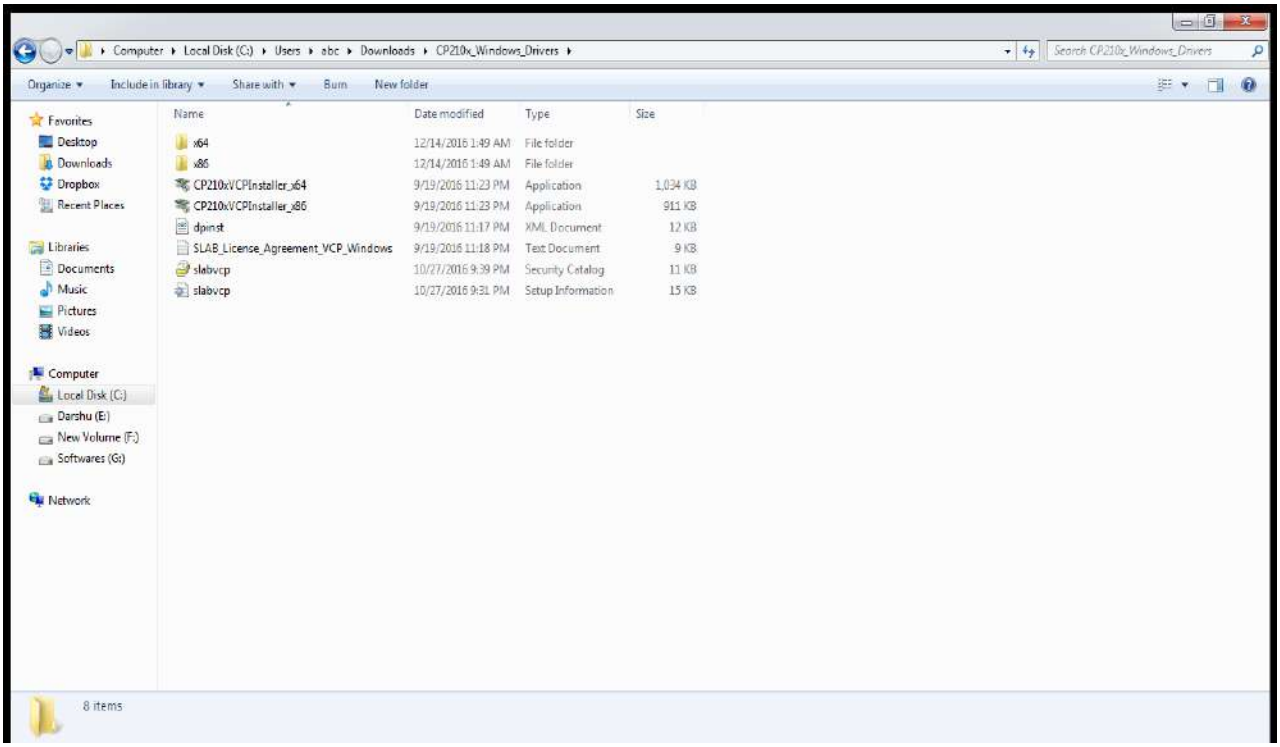
   https://drive.google.com/file/d/0B0rzq9C03J8ecUZIdE1KVUFvSVk/view?usp=sharing&resourcekey=0-7nouXDGHyIX2Kgcxv1uYng

Step by Step Installation of CP2102 Drivers

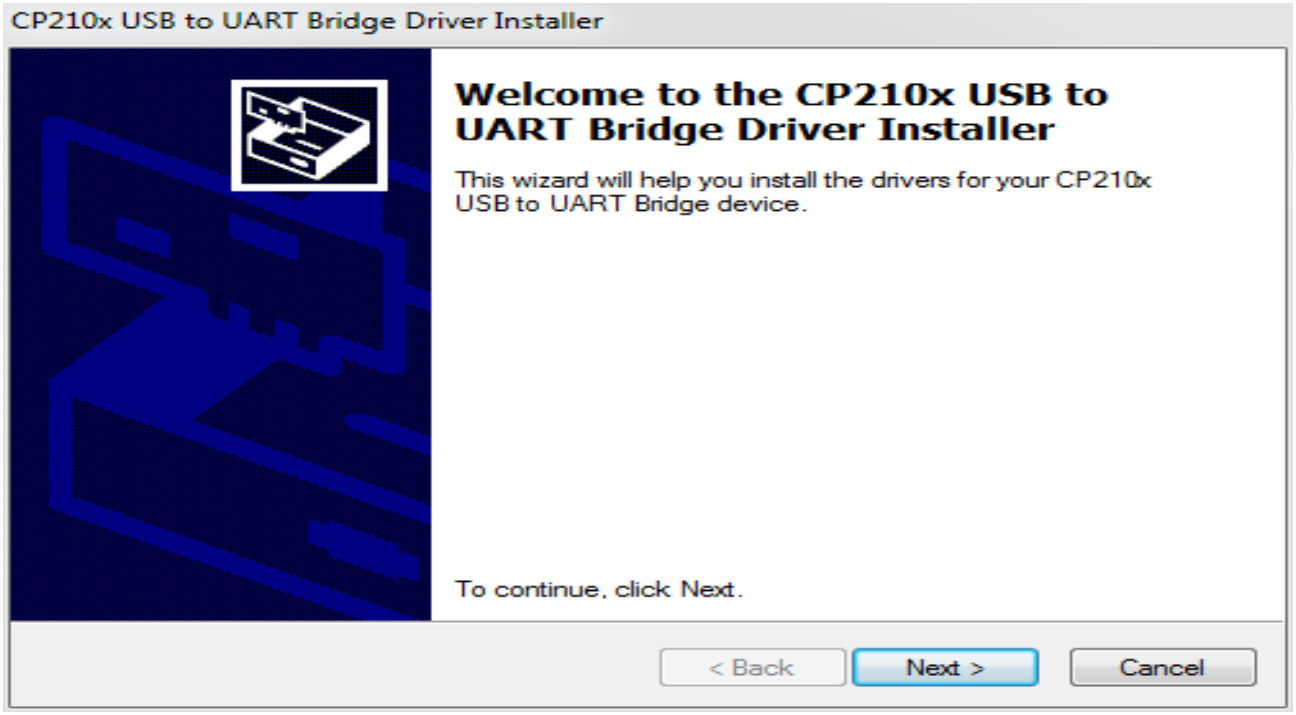
Initially connect your hardware having CP2102 USB to Serial IC to your PC. In the device manager it will show "CP2102 USB to UART Bridge Controller" (as shown in below figure) which means that your PC has detected the drivers but please note the "!" sign which indicated cp2102 drivers have not been installed.

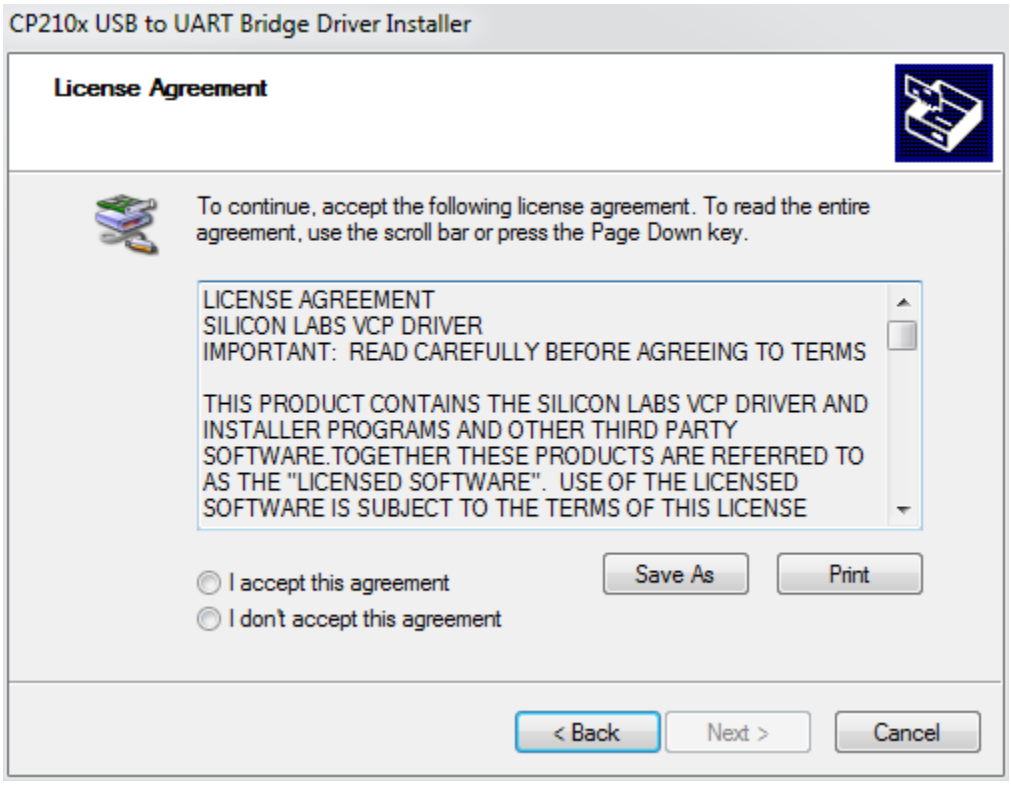


Now extract the “CP210x_Windows_Drivers” in a folder and in that you will find applications named “CP210xVCPInstaller_x64” & “CP210xVCPInstaller_x86”. If your system is 64 bit then run “CP210xVCPInstaller_x64.exe” application and if it is 32 bit then run “CP210xVCPInstaller_x86” application.

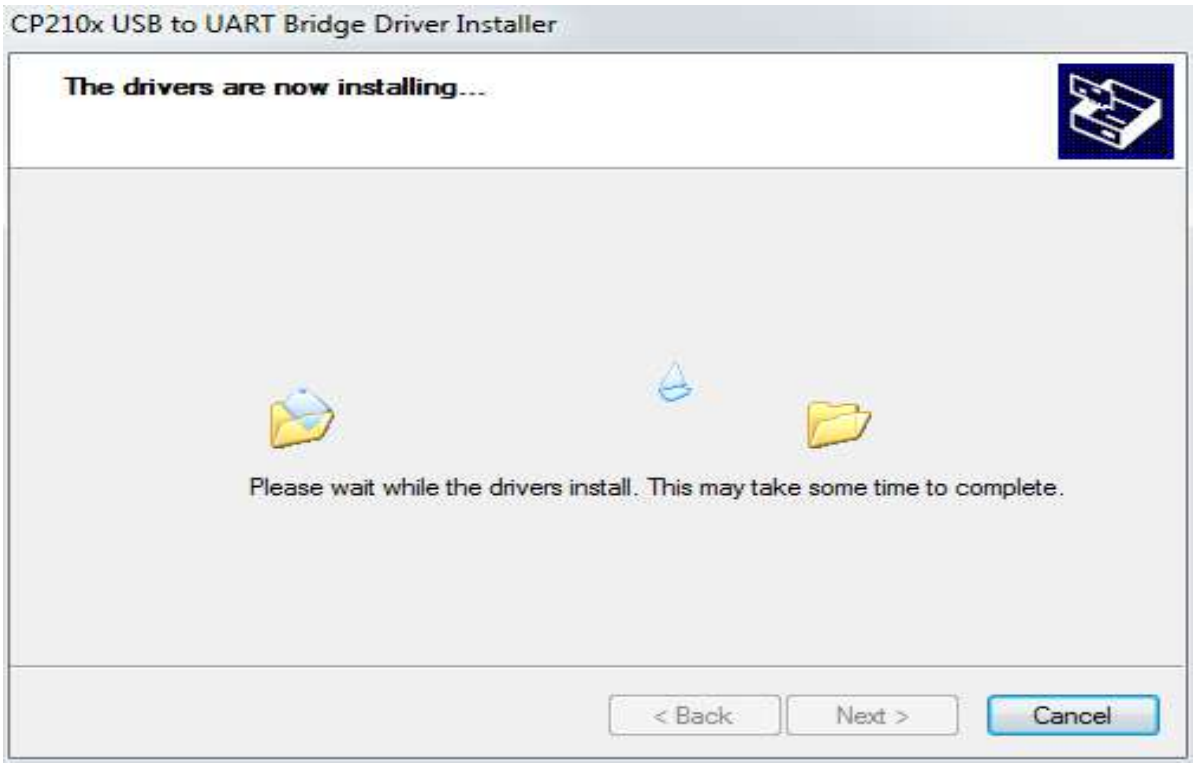


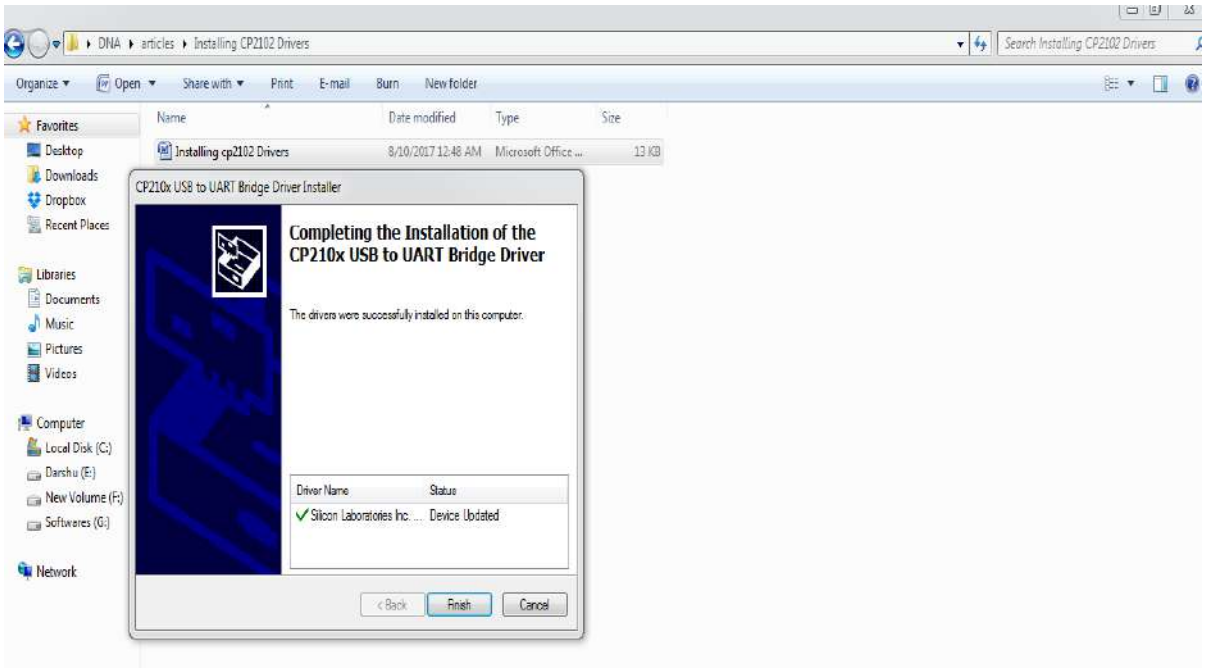
Once you run your application it will show you the welcome screen click “Next” on the screen. Then the license Agreement screen is shown select “I agree” and then on “Next”.



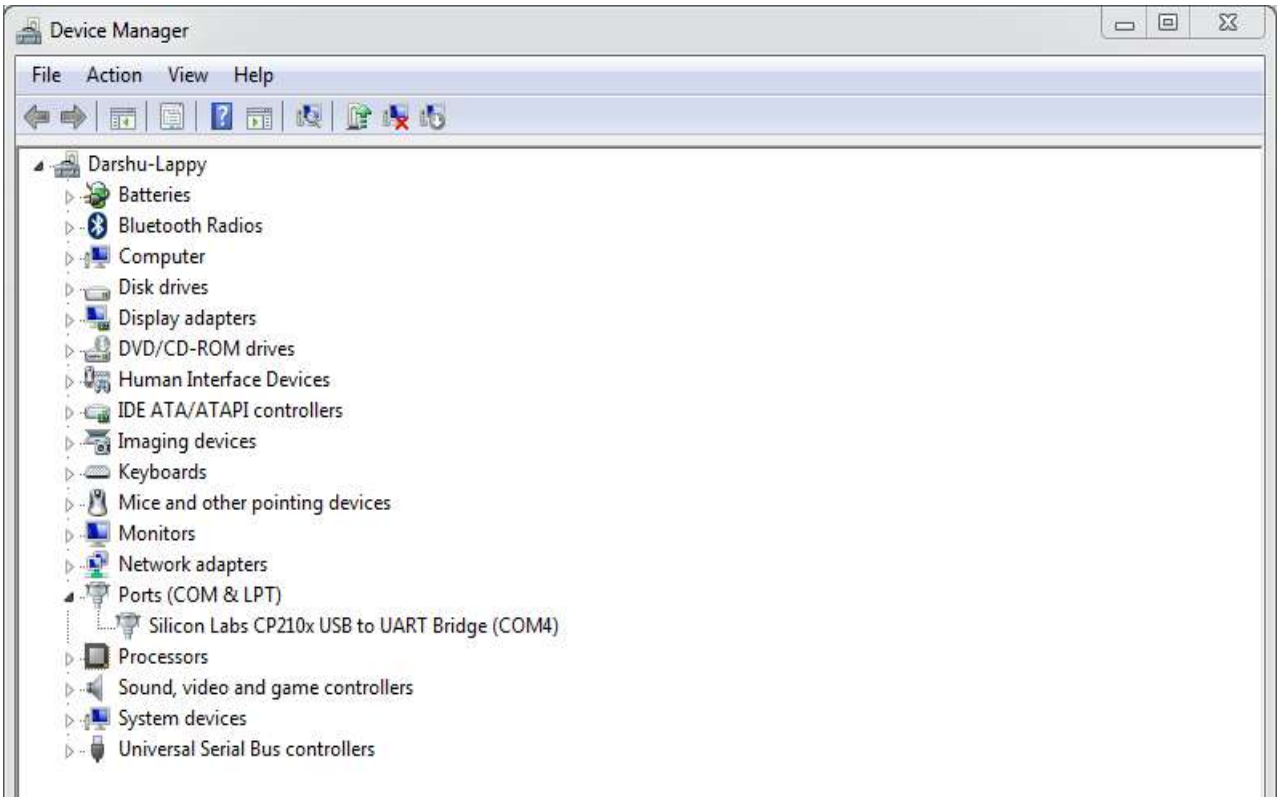


Just relax now the CP2102 drivers will start installing…..and within few seconds completed screen will be displayed.





Once installed it will show driver successfully installed. Now go again back to device manager and there you will see that the driver has been successfully been installed and a com port has been allotted. In below image you can see that “com4” has been allotted for cp2102 IC in my laptop.



Drivers for the CP2102 have now been installed successfully. You can see in above image that Port Number 4 has been allocated to this IC. If you want you can even change the virtual com port number from Device Manager.

RESULT:

In this practical, we studied and installed the NodeMCU IDE for IoT projects, exploring its key features like built-in Wi-Fi, GPIO pins, and communication protocols. We also installed the necessary CP2102 drivers, enabling successful communication between NodeMCU and the computer for programming and testing IoT applications.

Practical No:-03

Aim:- Write the steps to add libraries in Arduino and setup of Arduino IDE for programming.

Introduction

Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc. For example, the Liquid Crystal library makes it easy to talk to character LCD displays.

Library	Purpose	Usage
Servo	Servo Motor Control	Control servo motors with precise movement.
Wire	I2C Communication	Communicate with I2C devices like sensors.
SPI	SPI Communication	Communicate with devices using SPI.
LiquidCrystal	LCD Display Control	Control LCD displays (HD44780 compatible).
Adafruit GFX	Graphics Library	Draw on graphical displays (TFT, OLED).
Adafruit NeoPixel	RGB LED Control	Control addressable RGB LEDs (WS2812).
SoftwareSerial	Software-based Serial Communication	Use serial communication on other pins.
Ethernet	Ethernet Networking	Connect to networks via Ethernet shield.
SD	SD Card Interface	Read/write data from SD cards.
Time	Timekeeping	Keep track of time and delays.
DHT	Temperature & Humidity Sensor	Interface with DHT11/DHT22 sensors.
Blynk	IoT App	Control Arduino via a smartphone app.
OneWire	One-Wire Devices	Communicate with One-Wire devices (e.g., DS18B20).
WiFi	WiFi Communication	Connect to WiFi (for ESP8266/ESP32).
TFT	TFT Display Control	Interface with TFT screens.
NewPing	Ultrasonic Sensor	Interface with ultrasonic distance sensors.
IRremote	Infrared Communication	Decode signals from IR remotes.
DallasTemperature	Temperature Sensors	Interface with Dallas temperature sensors.
LiquidCrystal_I2C	I2C LCD Display Control	Control I2C LCD displays.

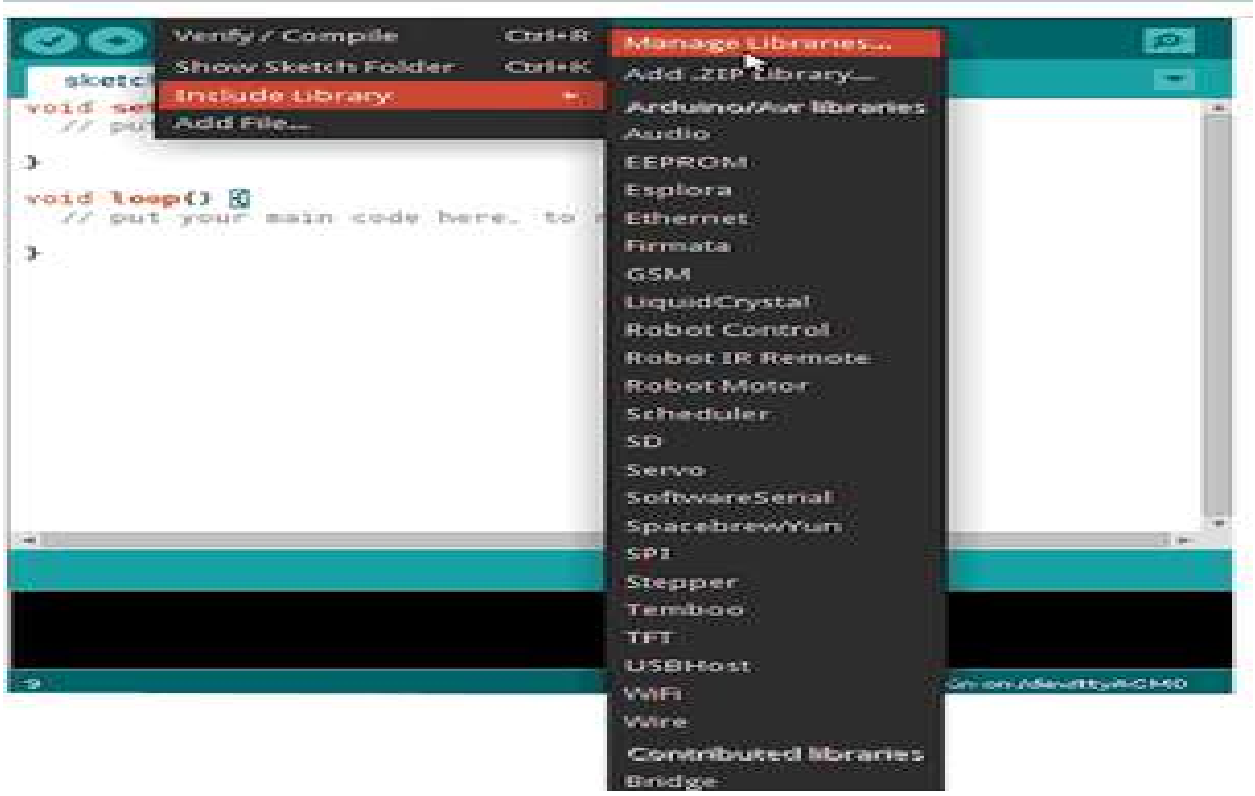
There are thousands of libraries available for download directly through the Arduino IDE, and you can find all of them listed at the **Arduino Library Reference.**

Procedure:

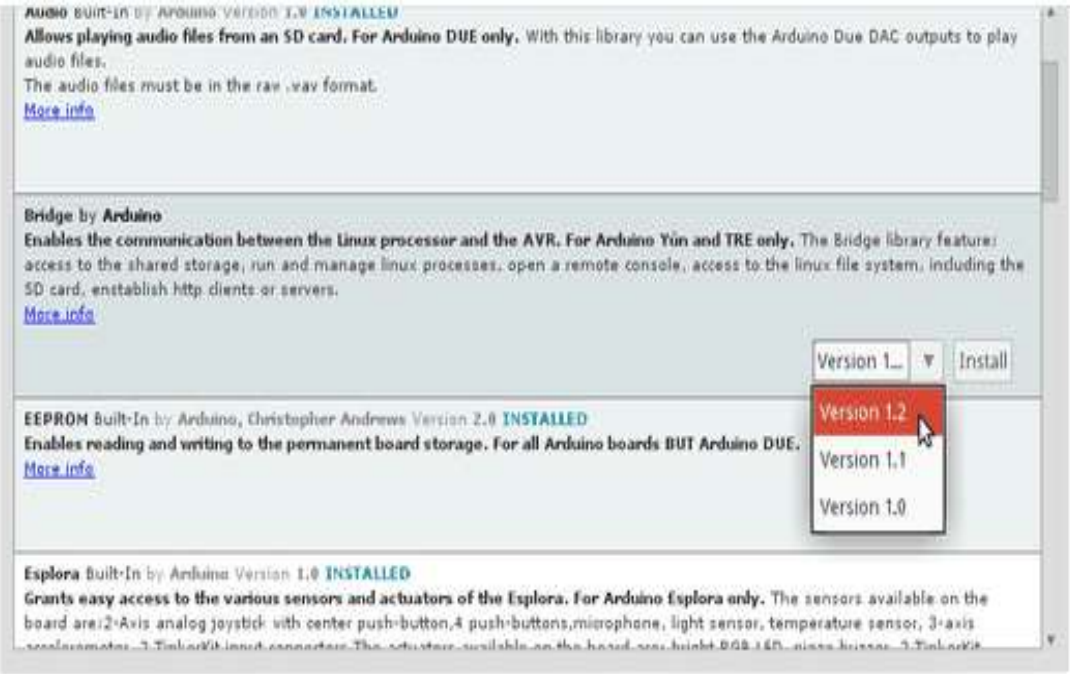
(a) Using the Library Manager

To install a new library into your Arduino IDE you can use the Library Manager (available from IDE version 1.6.2). Open the IDE and click to the "**Sketch**" menu and then **Include Library** > **Manage Libraries**.

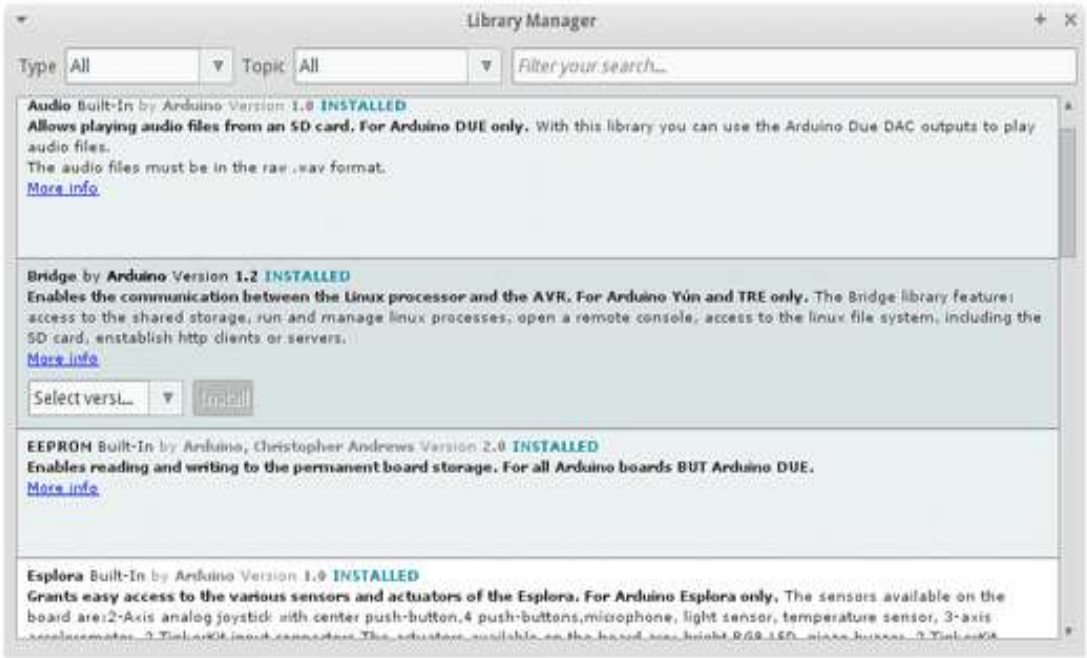
Then the Library Manager will open and you will find a list of libraries that are already installed or ready for installation.



In this example we will install the Bridge library. Scroll the list to find it, click on it, then select the version of the library you want to install. Sometimes only one version of the library is available. If the version selection menu does not appear, don't worry: it is normal.



Finally click on install and wait for the IDE to install the new library. Downloading may take time depending on your connection speed. Once it has finished, an Installed tag should appear next to the Bridge library. You can close the library manager.



You can now find the new library available in the Sketch > Include Library menu.

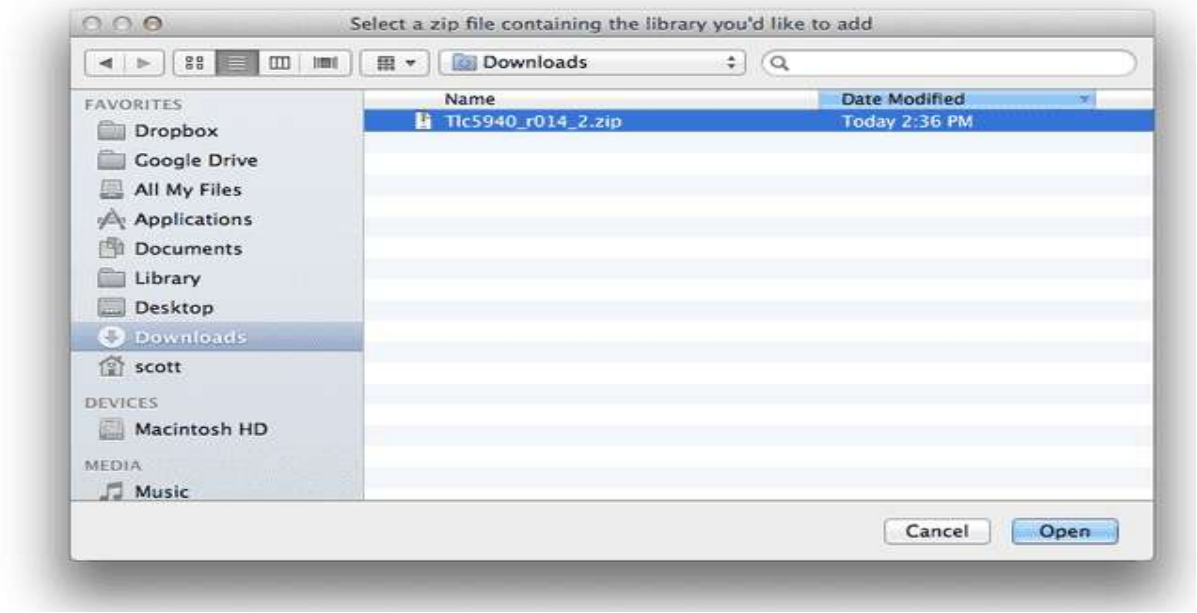
(b) If you want to add your own library to Library Manager

Importing a .zip Library

Libraries are often distributed as a ZIP file or folder. The name of the folder is the name of the library. Inside the folder will be a .cpp file, a .h file and often a keywords.txt file, examples folder, and other files required by the library. Starting with version 1.0.5, you can install 3rd party libraries in the IDE. Do not unzip the downloaded library, leave it as is.

In the Arduino IDE, navigate to **Sketch > Include Library > Add .ZIP Library**. At the top of the drop down list, select the option to "Add .ZIP Library".

You will be prompted to select the library you would like to add. Navigate to the .zip file's location and open it.



Return to the **Sketch > Include Library menu**. . You should now see the library at the bottom of the drop-down menu. It is ready to be used in your sketch. The zip file will have been expanded in the libraries folder in your Arduino sketches directory.

RESULT:

In this practical, we learned how to add libraries to the Arduino IDE, both through the Library Manager and by importing a .zip library. Using these methods, we can easily extend the functionality of the Arduino platform to interface with various sensors, displays, and other modules in our projects.

Practical No:-04

Aim:- Introduction to different types of IoT Sensors

Introduction

In IoT (Internet of Things), a **sensor** is a device that detects and measures physical or environmental conditions and converts this information into an electrical signal that can be read, processed, and transmitted to other devices for analysis and action.

Role of Sensors in IoT

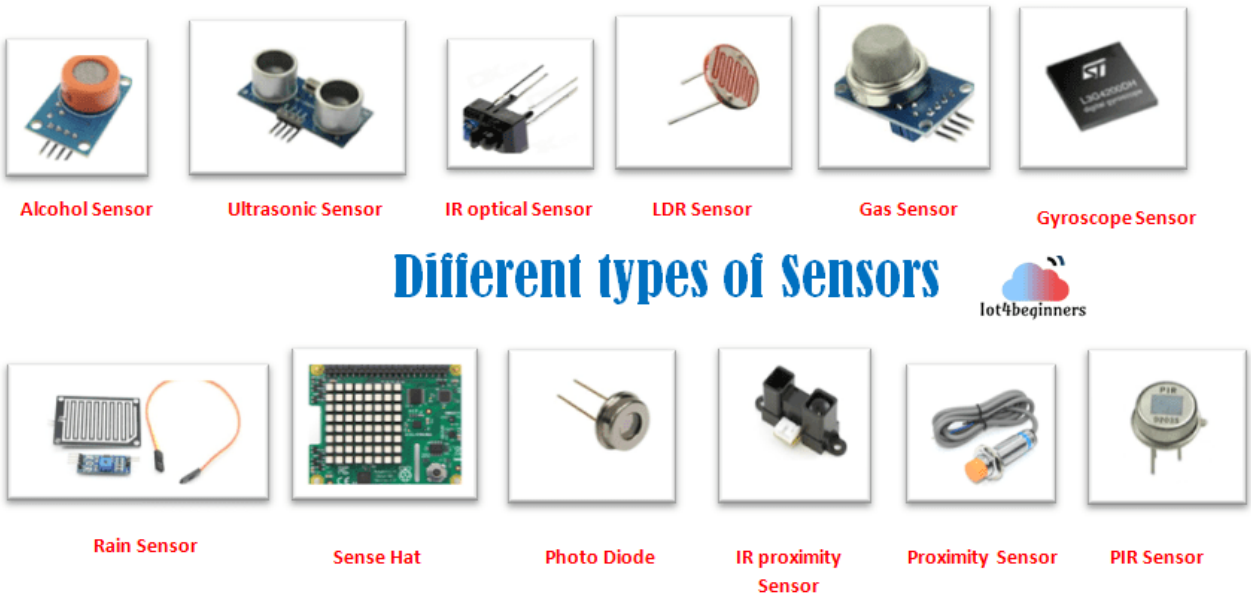
Sensors act as the **"eyes"** or **"ears"** of IoT systems. They collect data from the environment, which is then processed and used to trigger actions or provide insights. The IoT system relies on sensors to continuously monitor conditions such as temperature, humidity, light levels, pressure, motion, sound, and many other variables.

Types of Sensors in IoT

1. **Temperature Sensors**: Measure temperature and send data to IoT systems for weather monitoring, HVAC control, and industrial applications.
2. **Humidity Sensors**: Monitor moisture levels in the air or soil for applications like agriculture, home automation, and industrial processes.
3. **Motion Sensors**: Detect movement, often used in security systems, smart homes, and industrial automation.
4. **Proximity Sensors**: Detect the presence of objects without physical contact, commonly used in automotive, robotics, and retail systems.
5. **Gas Sensors**: Detect the presence of gases such as CO2, methane, or carbon monoxide, useful in environmental monitoring and safety applications.
6. **Pressure Sensors**: Measure atmospheric or liquid pressure, used in weather stations, automotive systems, and industrial equipment.
7. **Light Sensors**: Measure the intensity of light in the environment, used for controlling lighting in smart homes or street lighting systems.

8. **Sound Sensors**: Detect sound levels and convert them into an electrical signal for applications like noise monitoring or audio-based IoT systems.
9. **Gas and Smoke Sensors**: Detect hazardous gases or smoke, essential for fire alarms and safety systems.
10. **Accelerometers**: Measure changes in velocity and movement, commonly used in wearables, fitness trackers, and automotive systems.

Sensor Type	Function	Main Uses
Alcohol Sensor	Detects alcohol gases	Breath analyzers, Safety monitoring
Ultrasonic Sensor	Measures distance	Parking systems, Robot navigation
IR Optical Sensor	Detects objects	Line following, Object detection
LDR Sensor	Measures light	Automatic lighting, Photography
Gas Sensor	Detects gases	Air quality, Safety systems
Gyroscope Sensor	Measures orientation	Phone orientation, Gaming
Rain Sensor	Detects water	Auto wipers, Irrigation
Sense Hat	Multiple measurements	Weather stations, Education
Photo Diode	Light to electrical signal	Light detection, Communications
IR Proximity	Detects close objects	Touchless switches, Robotics
Proximity Sensor	Detects objects	Manufacturing, Automation
PIR Sensor	Detects motion	Security, Auto lighting



How Sensors Work in IoT

- **Data Collection**: Sensors capture data from the environment (e.g., temperature, humidity, motion).
- **Signal Conversion**: The physical data is converted into electrical signals (analog or digital).
- **Data Transmission**: The signal is then sent to a microcontroller or other devices for processing via wired or wireless communication (e.g., Wi-Fi, Bluetooth, Zigbee).
- **Data Processing & Action**: The data is processed to make decisions or trigger actions, such as turning on a fan if the temperature rises above a set threshold.

Applications of Sensors in IoT

1. **Smart Homes**: Sensors are used in IoT applications to control lighting, temperature, security systems, and appliances remotely.
2. **Healthcare**: Wearable sensors monitor vital signs like heart rate and glucose levels, providing real-time health data for patients and doctors.
3. **Agriculture**: Soil moisture and environmental sensors help farmers optimize irrigation and monitor crops for better yield.
4. **Industrial Automation**: Sensors monitor machinery, detect wear and tear, and automate production processes.
5. **Smart Cities**: Sensors in traffic systems, streetlights, and waste management contribute to improved urban living conditions.
6. **Environmental Monitoring**: Sensors track air quality, water quality, and pollution levels for environmental protection and public health.

RESULT:

The practical introduces various types of IoT sensors, explaining their functions and applications across different fields. Sensors in IoT play a crucial role in data collection, processing, and triggering actions for smart systems, such as smart homes, healthcare, agriculture, and industrial automation.

Practical No:-05

Aim:- Write a Program using NodeMCU for Blink LED

Introduction

This project demonstrates how to blink an LED using a NodeMCU. The LED is controlled by the GPIO pins of the NodeMCU, and it will blink at a set interval. This is a beginner-friendly project to understand the basic working of NodeMCU.

Structure

- **Objective:** To blink an LED on the NodeMCU board at regular intervals using a simple program.
- **Key Features:**
    - Uses NodeMCU (ESP8266) for control.
    - LED blinks at regular intervals (on/off).
    - Output visible through the physical LED blinking.

Hardware Required

- NodeMCU Kit (includes NodeMCU ESP8266) – 1 unit
- External LED (optional, for external setup) – 1 unit
- 220-ohm Resistor (optional, for external setup) – 1 unit
- Breadboard and connecting wires (optional, for external setup)

Connections

- **Onboard LED**:
    - The onboard LED is typically connected to **GPIO 2 (D4)** or as per your kit's documentation.
- **External LED Setup**:
    - If using an external LED:
        - Connect the anode (longer leg) of the LED to GPIO 2 (D4).

- Connect the cathode (shorter leg) to GND via a 220-ohm resistor.

Steps of Working

- **Hardware Setup:**

Use the onboard LED or connect the external LED as described above.

- **Software Setup:**

Install the Arduino IDE and ensure the ESP8266 board package is installed.

Connect the NodeMCU to your computer using a USB cable.

- **Code Upload:**

Open the Arduino IDE.

Select the board and the correct COM port.

Copy and upload the program given below.

- **Monitor the Output:**

After the program is uploaded successfully, the onboard or external LED will blink at regular intervals.

Program

```
// Define the pin where the onboard or external LED is connected
#define LED_PIN 2 // Onboard LED or external LED connected to GPIO2 (D4)
void setup() {
  // Initialize the LED pin as an output
  pinMode(LED_PIN, OUTPUT);
}
void loop() {
  // Turn the LED on
  digitalWrite(LED_PIN, LOW); // LOW for onboard LED, HIGH for external LED
  delay(1000);              // Wait for 1 second
```

```
  // Turn the LED off
  digitalWrite(LED_PIN, HIGH); // HIGH for onboard LED, LOW for external LED
  delay(1000);              // Wait for 1 second
}
```

Precautions

- **Power Supply:** Ensure the NodeMCU kit is connected to a stable power source via USB.
- **Pin Configuration:** Refer to the IoT Academy kit manual for exact pin labeling and functionality.
- **Connections:** Double-check wiring for external LEDs to avoid short circuits.
- **Resistor Usage:** Use a 220-ohm resistor to protect an external LED from excessive current.
- **Onboard LED Logic:** Note that onboard LEDs typically operate with inverted logic (LOW = ON, HIGH = OFF).

Expected Output

After successfully uploading and running the program:

- The onboard or external LED will blink on and off every second.
- The pattern will be:
    - LED ON for 1 second.
    - LED OFF for 1 second.

The blinking will continue as long as the NodeMCU is powered.

RESULT

The LED successfully blinks on and off at 1-second intervals, demonstrating the correct setup of the NodeMCU hardware and successful program execution. This validates the working of GPIO pins and basic control logic for LED blinking.

Original Output

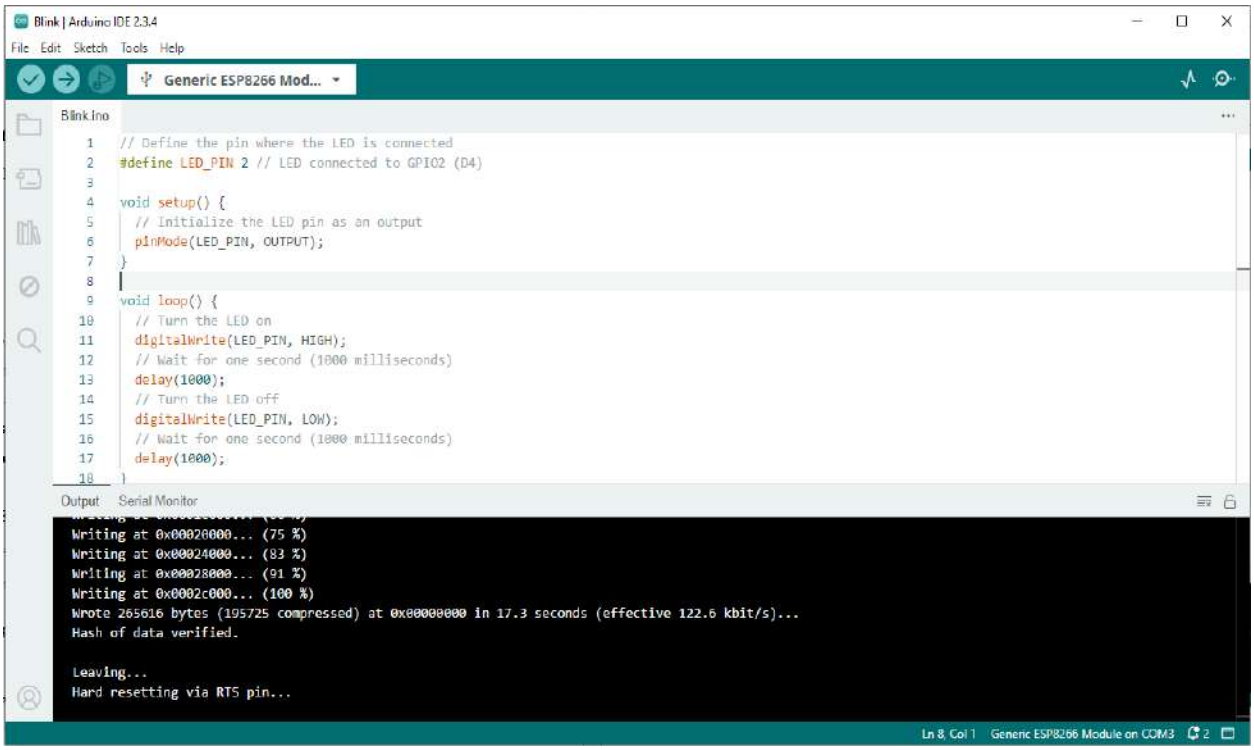


Figure:- Running Program

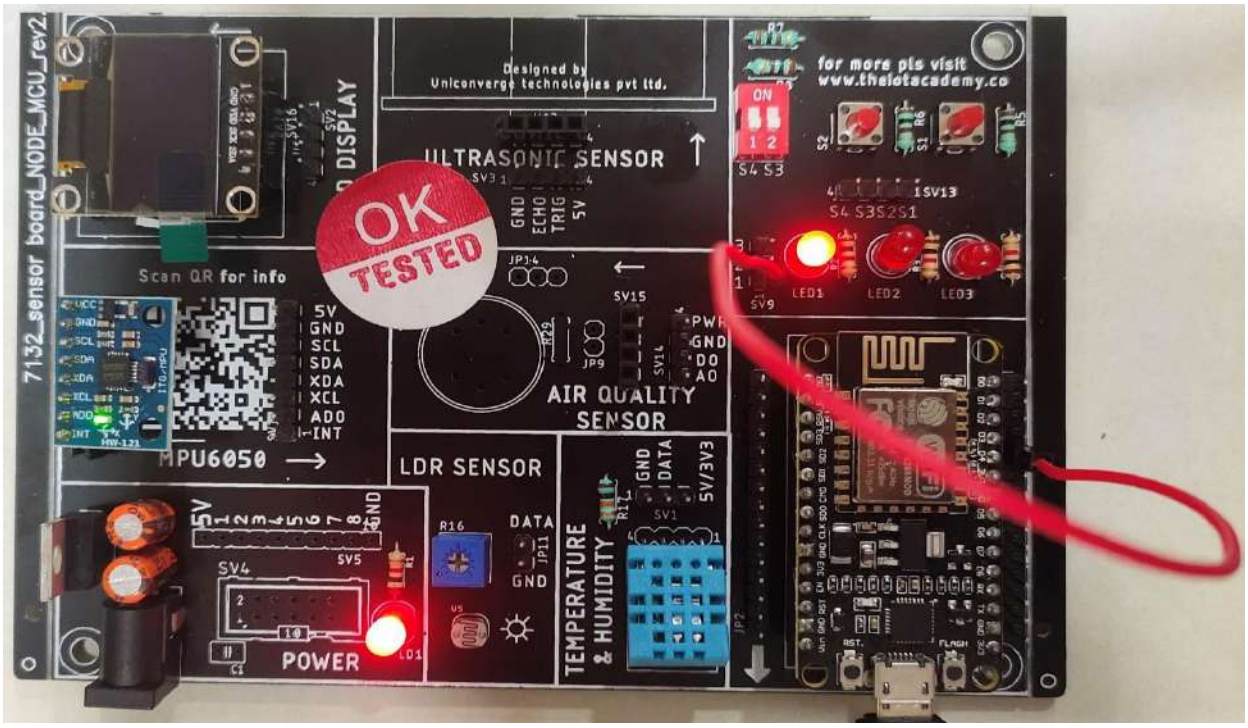


Figure:- Board Connection

Practical No:-06

Aim:- Write a Program using Monitoring Temperature and Humidity Using DHT11 and NodeMCU

1. Introduction

The project involves monitoring temperature and humidity using the **DHT11 sensor** and the **NodeMCU ESP8266 microcontroller**. The DHT11 sensor provides digital output for temperature and humidity, which the NodeMCU reads and displays on the Serial Monitor. This project is ideal for IoT applications, including weather monitoring systems, smart homes, and industrial environments.

2. Structure

- **Objective**: To measure and display temperature (in Celsius and Fahrenheit) and humidity on the Serial Monitor using NodeMCU and DHT11 sensor.
- **Key Features**:
    - Uses NodeMCU for Wi-Fi compatibility.
    - Accurate measurement of temperature and humidity.
    - Output visible on the Serial Monitor.

3. Hardware Required

1. **NodeMCU ESP8266** (1 unit)
2. **DHT11 Sensor** (1 unit)
3. **Connecting Wires**
4. **USB Cable** (to connect NodeMCU to the computer)
5. **Breadboard** (optional, for easier wiring)

4. Connections

Here is the connection of the DHT11 sensor with NodeMCU:

DHT11 Pin	NodeMCU Pin
VCC	3.3V
GND	GND
DATA	D4 (GPIO2)

5. Steps of Working

1. **Hardware Setup**:
    - Connect the DHT11 sensor to the NodeMCU as per the connection diagram.
    - Ensure secure connections to avoid loose wiring.
2. **Software Setup**:
    - Install the **Arduino IDE** on your computer.
    - Add the **ESP8266 Board Package** to the Arduino IDE (via Preferences → Additional Board URLs).
    - Install the **DHT Sensor Library** from the Library Manager (search for "DHT Sensor Library").
3. **Code Upload**:
    - Select the correct board (**NodeMCU 1.0 (ESP-12E)**) and port in the Arduino IDE.
    - Upload the program (given below) to the NodeMCU.
4. **Monitor the Output**:
    - Open the **Serial Monitor** in the Arduino IDE.
    - Set the baud rate to **115200** to view the temperature and humidity readings.

6. Program

```
#include <DHT.h> // Include the DHT sensor library

// Define the type of DHT sensor
#define DHTTYPE DHT11 // Using DHT11 sensor

// Define the GPIO pin where the DHT sensor is connected
#define DHTPIN 2 // GPIO2 corresponds to D4 on NodeMCU

// Initialize the DHT sensor
DHT dht(DHTPIN, DHTTYPE);
```

```
void setup() {
  // Begin communication with Serial Monitor
  Serial.begin(115200);

  // Initialize the DHT sensor
  dht.begin();

  // Display startup message
  Serial.println("DHT11 Temperature and Humidity Monitoring");
}

void loop() {
  // Wait 2 seconds between each measurement
  delay(2000);

  // Read temperature in Celsius
  float temperatureC = dht.readTemperature();

  // Read temperature in Fahrenheit
  float temperatureF = dht.readTemperature(true);

  // Read humidity percentage
  float humidity = dht.readHumidity();

  // Check for errors in sensor readings
  if (isnan(temperatureC) || isnan(temperatureF) || isnan(humidity)) {
    Serial.println("Error: Failed to read from DHT sensor!");
    return;
  }

  // Display the readings on the Serial Monitor
```

```
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.print(" %\t");
  Serial.print("Temperature in Celsius: ");
  Serial.print(temperatureC);
  Serial.print(" °C\t");
  Serial.print("Temperature in Fahrenheit: ");
  Serial.print(temperatureF);
  Serial.println(" °F");
}
```

7. Precautions

1. **Power Supply**: Ensure that the NodeMCU is connected to a stable power source via the USB cable.
2. **Connections**: Double-check all connections to avoid short circuits or loose connections.
3. **Library Installation**: Make sure the **DHT Sensor Library** and **ESP8266 Board Package** are properly installed in the Arduino IDE.
4. **Sensor Placement**: Place the DHT11 sensor in an environment free from external interference, such as direct heat or moisture, for accurate readings.
5. **Baud Rate**: Set the Serial Monitor's baud rate to **115200** for proper data display.

8. Expected Output

After successful code upload and execution:

- Humidity and temperature readings will be displayed on the Serial Monitor in the following format:
  Humidity: 45.00 % Temperature in Celsius: 25.50 °C Temperature in Fahrenheit: 77.90 °F
- If there’s an error in reading the sensor, the message:
  Error: Failed to read from DHT sensor!
  will be displayed.

RESULT:

The NodeMCU successfully reads and displays temperature and humidity values from the DHT11 sensor on the Serial Monitor. Accurate readings are shown in both Celsius and Fahrenheit, along with the humidity percentage, validating the setup and functionality.

Original Output

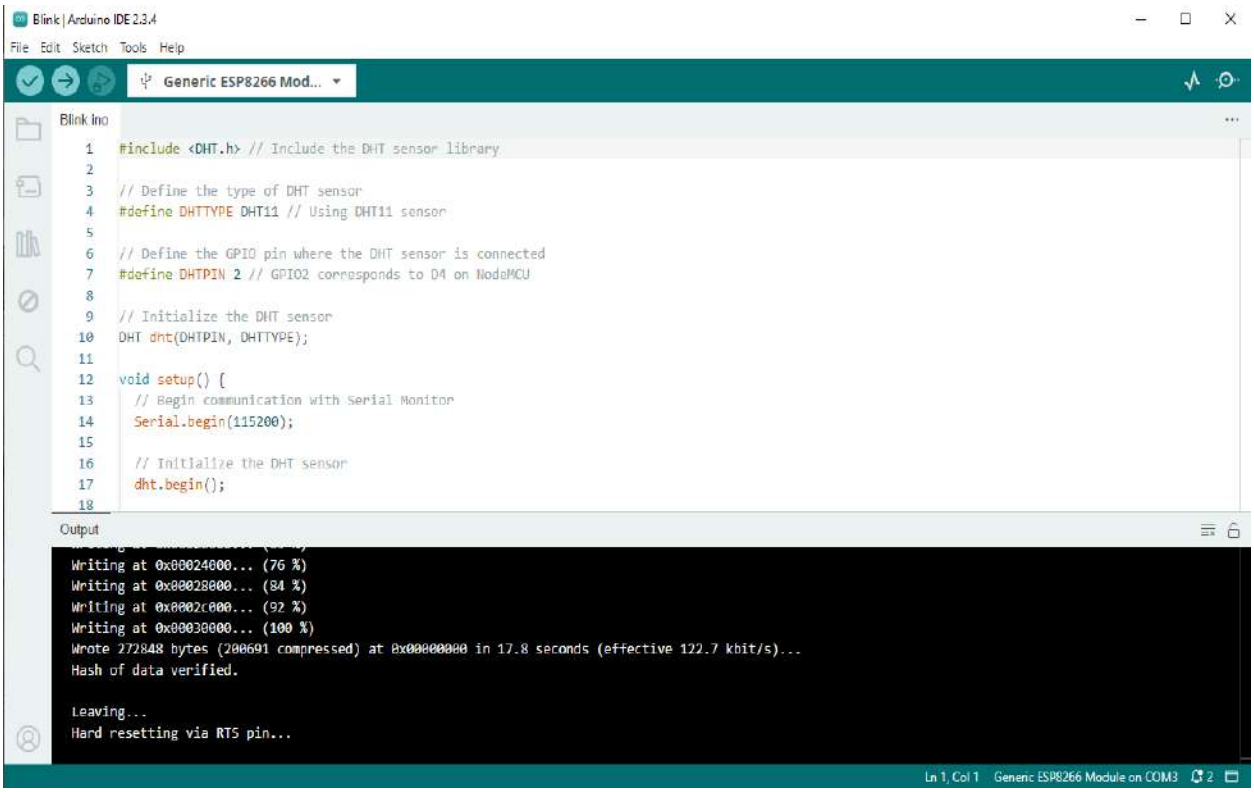


Figure:- Running Program

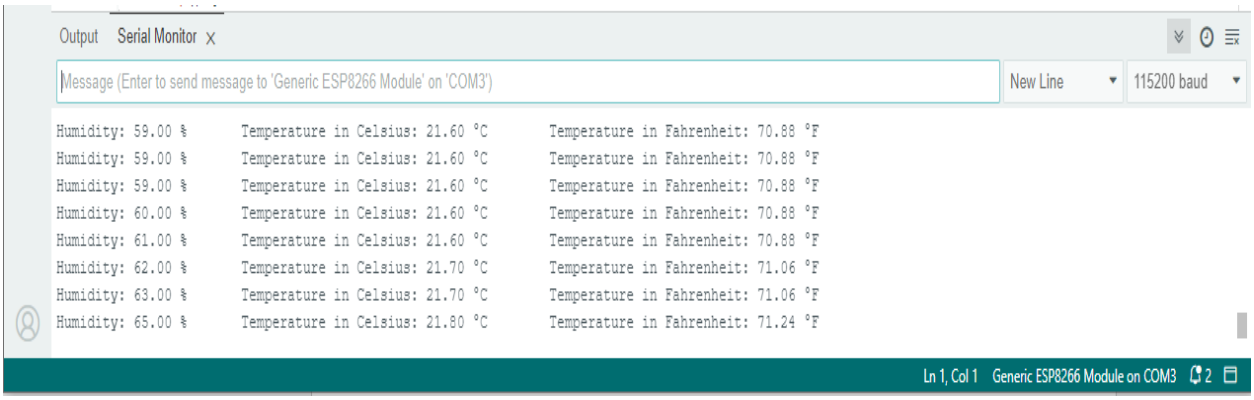


Figure:- Serial Monitor

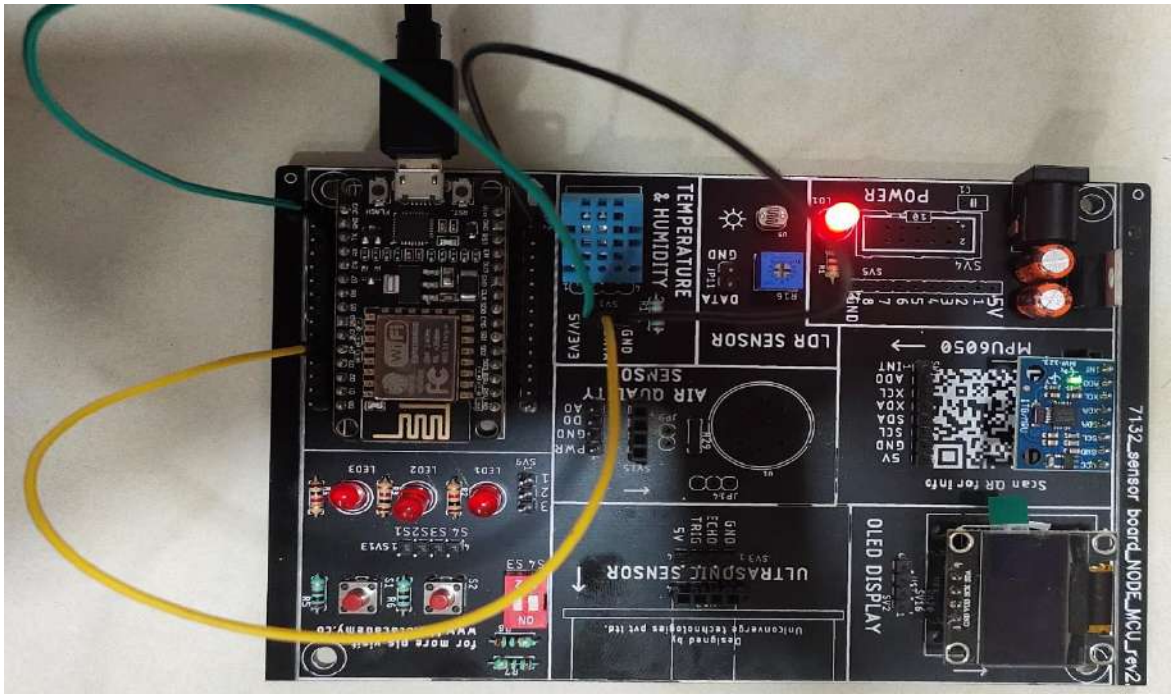


Figure:- Board Connection

Practical No:-07

Aim:- Write a Program to Fade LED on NodeMCU

1. Introduction

This project demonstrates how to fade an LED connected to a NodeMCU board (ESP8266) using the analogWrite() function. The LED's brightness will gradually increase and decrease to create a fading effect. This project is a basic example of how to use PWM-like functionality on the NodeMCU board.

2. Structure

- **Objective:** To fade an LED connected to a NodeMCU using the analogWrite() function, creating a smooth fading effect.
- **Key Features:**
    - Uses NodeMCU (ESP8266) for control.
    - Gradual change in LED brightness (fade in and fade out).
    - Simulates analog brightness control using PWM.

3. Hardware Required

- **NodeMCU ESP8266** – 1 unit
- **LED** – 1 unit
- **220-ohm Resistor** – 1 unit (optional, for protection)
- **Breadboard and Connecting Wires** – as required
- **USB Cable** – to connect NodeMCU to the computer

4. Connections

- Connect the **anode (longer leg)** of the LED to **GPIO4 (D2)** on NodeMCU.
- Connect the **cathode (shorter leg)** to **GND** through a **220-ohm resistor** (optional).

5. Steps of Working

1. **Hardware Setup:**

- Connect the LED to the NodeMCU as shown in the connection diagram.
- Ensure all connections are secure to avoid short circuits.

2. **Software Setup:**
   - Install the Arduino IDE and the necessary ESP8266 board package.
   - Select the board under **Tools → Board**.
3. **Code Upload:**
   - Open the Arduino IDE and paste the provided code.
   - Select the correct COM port and upload the program to the NodeMCU.
4. **Monitor the Output:**
   - Once the program is uploaded successfully, the LED will begin fading in and out.

6. Program

```
// Define the pin where the LED is connected
#define LED_PIN 4 // GPIO4 corresponds to D2 on NodeMCU

// Initialize brightness variable
int brightness = 0;    // Initial brightness level
int fadeAmount = 1;    // Amount to change brightness each step

void setup() {
  // Set the LED pin as an output
  pinMode(LED_PIN, OUTPUT);
}

void loop() {
  // Set the LED brightness using PWM
  analogWrite(LED_PIN, brightness);

  // Adjust the brightness for the next loop
  brightness += fadeAmount;
```

```
  // Reverse the fading direction when reaching limits
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }

  // Pause for a short duration to see the fade effect
  delay(30);
}
```

7. Precautions

1. **Power Supply:** Ensure that the NodeMCU is connected to a stable power source via USB.
2. **Pin Configuration:** GPIO4 (D2) on NodeMCU is used for PWM control. Ensure the pin supports PWM functionality.
3. **Connections:** Double-check all wiring connections to avoid short circuits and ensure proper LED orientation.
4. **Resistor Usage:** Always use a resistor (220-ohm or similar) to protect the LED from excessive current.
5. **Smoothness:** This program creates a fade effect by changing brightness in steps. You may need to adjust delay() for a smoother fade.

8. Expected Output

- After uploading and running the program:
    - The LED will gradually fade in (increase brightness) and fade out (decrease brightness).
    - The fading will happen continuously, reversing direction when the brightness reaches the minimum (0) or maximum (255).

RESULT:

The LED connected to the NodeMCU will gradually fade in and out, increasing and decreasing its brightness in smooth steps. The fading effect will continuously reverse direction between the minimum and maximum brightness levels.

Original Output

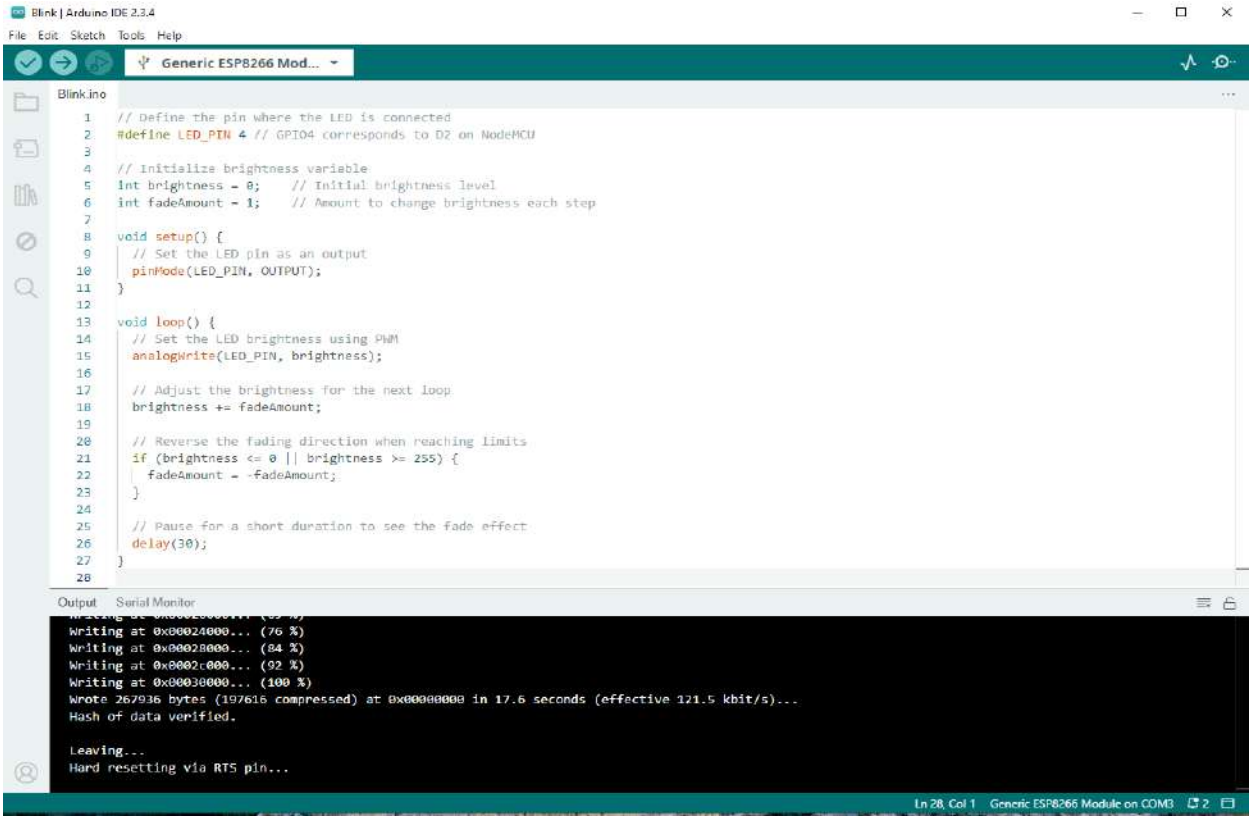


Figure:- Running Program

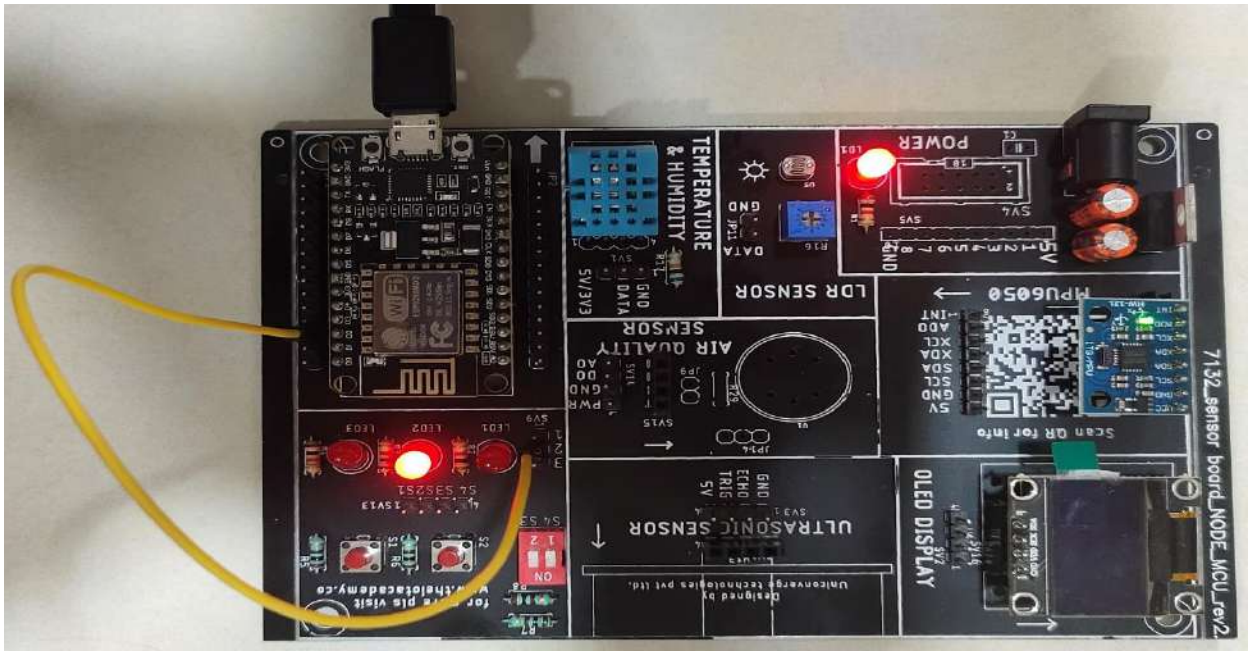


Figure:- Board Connection

Practical No:-08

Aim:- Write a program to print " Hello World" on LCD Display using NodeMCU

1. Introduction

To display a "Hello, World!" message on an LED screen using a NodeMCU, you'll typically use an **OLED** display, such as the popular **SSD1306** OLED display. Below is a basic example of how to display "Hello, World!" on a 128x64 OLED screen using NodeMCU and the Arduino IDE.

2. Hardware Required

- **NodeMCU ESP8266** – 1 unit
- **OLED Display (SSD1306)** – 1 unit (128x64)
- **Connecting Wires**
- **Breadboard** (optional)

3. Connections

- **VCC/VDD** of the OLED to **3V** on NodeMCU
- **GND** of the OLED to **GND** on NodeMCU
- **SCL/SCK** of the OLED to **D1 (GPIO5)** on NodeMCU
- **SDA** of the OLED to **D2 (GPIO4)** on NodeMCU

4. Steps to Set Up the Software

1. **Install the Required Libraries:**
    - Open the Arduino IDE.
    - Go to **Sketch → Include Library → Manage Libraries**.
    - Search for Adafruit SSD1306 and install it.
    - Search for Adafruit GFX and install it.
2. **Select the NodeMCU Board:**

- Go to **Tools → Board → Select Board**

3. **Upload the Program**

5. Program

```
#include <Wire.h>                 // Include the Wire library for I2C communication

#include <Adafruit_GFX.h>           // Include the Adafruit graphics library

#include <Adafruit_SSD1306.h>       // Include the Adafruit SSD1306 library



#define SCREEN_WIDTH 128            // OLED display width, in pixels

#define SCREEN_HEIGHT 64            // OLED display height, in pixels



#define OLED_RESET    -1            // Reset pin # (or -1 if sharing Arduino reset pin)



// Define I2C address for the OLED (commonly 0x3C)

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);



void setup() {

  // Start serial communication

  Serial.begin(115200);
```

```
// Initialize OLED display with I2C address 0x3C
if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
  Serial.println(F("SSD1306 allocation failed"));
  for(;;);
}

// Clear the display buffer
display.clearDisplay();

// Set text color to white
display.setTextColor(SSD1306_WHITE);

// Set text size and cursor position
display.setTextSize(1);
display.setCursor(0, 0);

// Display "Hello, World !"
display.println(F("Hello, World !"));
```

```
  // Render the display
  display.display();
}

void loop() {
  // No continuous code needed for this example
}
```

6. Precautions

- Ensure proper wiring connections to avoid display malfunctions.
- Make sure the OLED display is compatible with the I2C protocol (SSD1306).
- Check the address of the SSD1306 OLED (usually 0x3C or 0x3D).

7. Expected Output

The OLED display will show the message **"Hello, World !"** centered on the screen, once the program is uploaded and executed on the NodeMCU.

RESULTS

The OLED display successfully shows the message "Hello, World!". The output is clear and stable, indicating proper communication between the NodeMCU and the SSD1306 OLED display.

Original Output

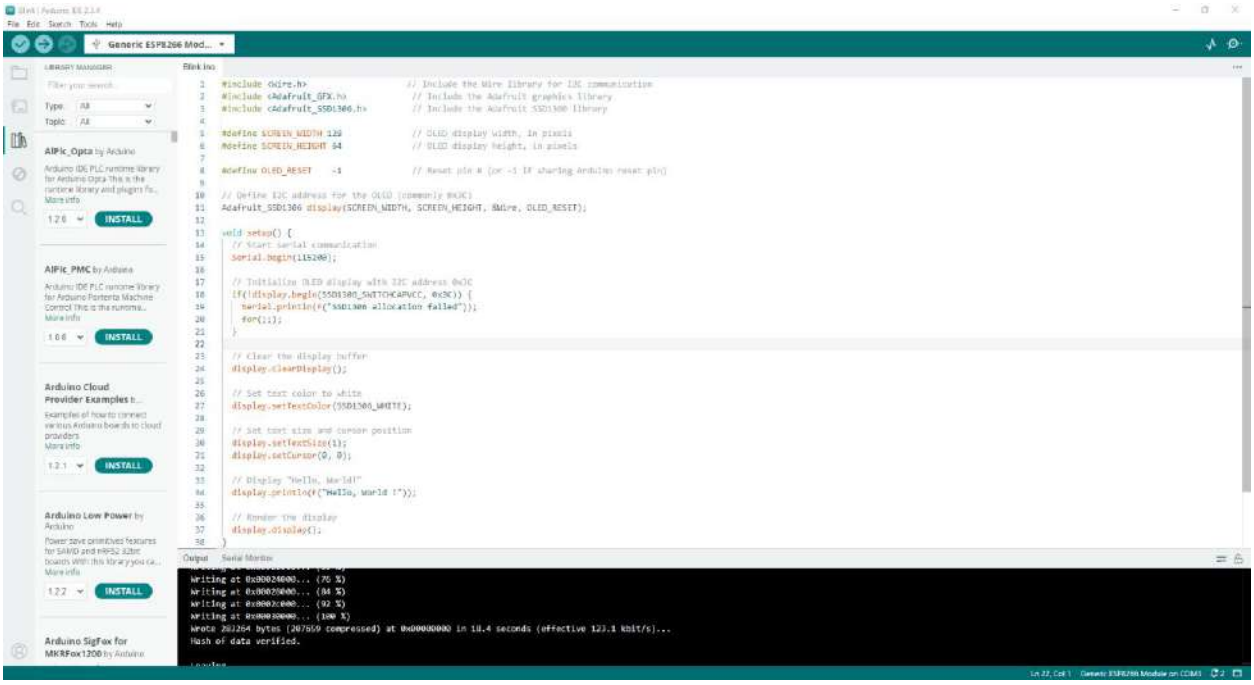


Figure:- Running Program

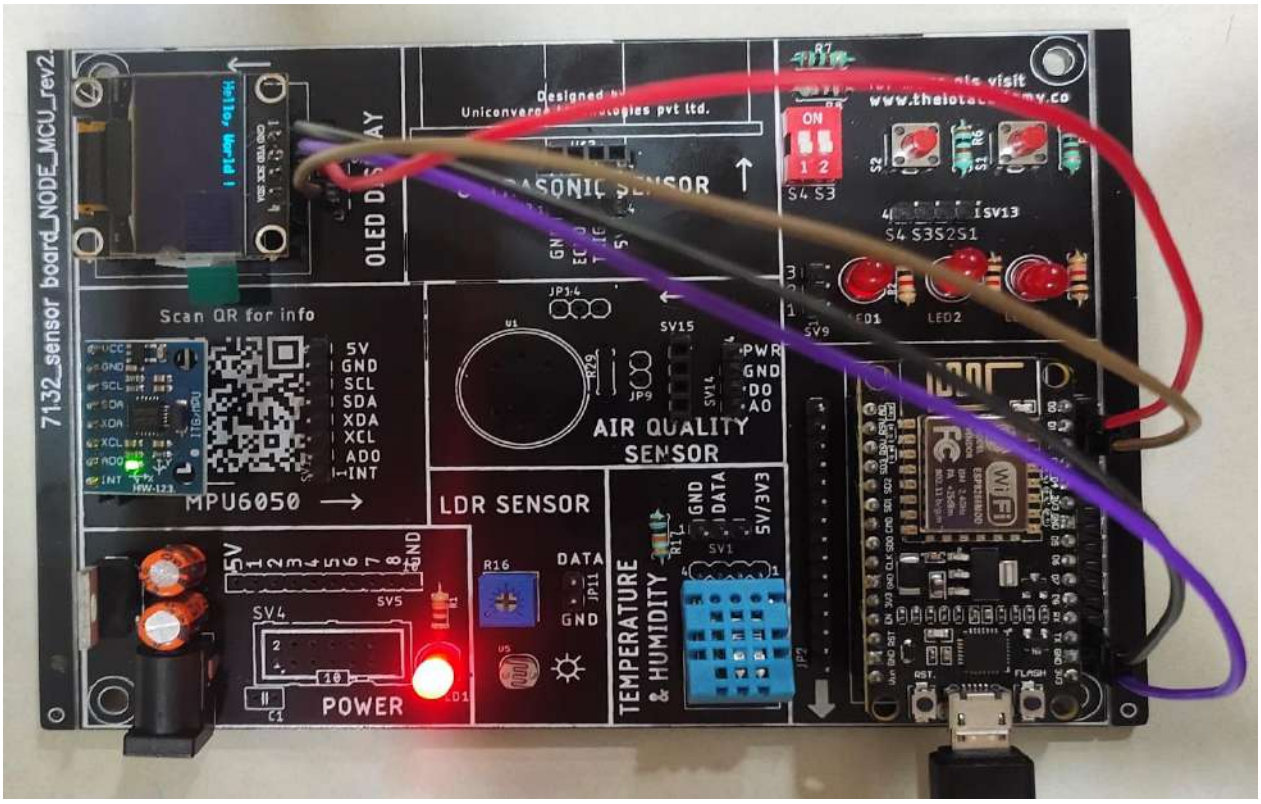


Figure:- Board Connection

Practical No:-09

Aim:- Write a program to Monitoring Air Quality Using NodeMCU and MQ135 Sensor

1. Introduction

The MQ135 sensor is widely used for air quality monitoring. It detects gases like CO2, NH3, benzene, and smoke. This program uses NodeMCU to read sensor values and categorize air quality into qualitative levels like "Good," "Moderate," and "Poor."

2. Hardware Required

1. **NodeMCU ESP8266** – 1 unit
2. **MQ135 Air Quality Sensor** – 1 unit
3. **Connecting Wires**
4. **Breadboard** (optional)

3. Connections

MQ135 Pin	**NodeMCU Pin**
VCC	3.3V
GND	GND
AOUT	A0
DOUT	D2 (GPIO4)

4. Steps of Working

1. **Hardware Setup:**
    - Connect the sensor's pins to the NodeMCU as per the wiring diagram.
    - Ensure secure connections.
2. **Software Setup:**
    - Install and open the Arduino IDE.

- Configure the board as **NodeMCU 1.0 (ESP-12E Module)**.
- Select the correct COM port.

5. Program

```
#define MQ135_AOUT A0  // Analog pin for air quality readings
#define MQ135_DOUT 4   // Digital pin (DOUT connected to GPIO4)

void setup() {
  // Start serial communication
  Serial.begin(115200);

  // Set DOUT pin as input
  pinMode(MQ135_DOUT, INPUT);

  // Welcome message
  Serial.println("MQ135 Real-World Air Quality Monitoring");
}

void loop() {
  // Read analog value (proportional to gas concentration)
  int airQualityAnalog = analogRead(MQ135_AOUT);

  // Read digital value (threshold-based detection)
  int airQualityDigital = digitalRead(MQ135_DOUT);

  // Categorize air quality based on analog values
  String airQualityCategory;
  if (airQualityAnalog < 200) {
    airQualityCategory = "Good";
  } else if (airQualityAnalog < 400) {
```

```
    airQualityCategory = "Moderate";
  } else if (airQualityAnalog < 600) {
    airQualityCategory = "Unhealthy";
  } else {
    airQualityCategory = "Hazardous";
  }

  // Print analog air quality reading
  Serial.print("Analog Air Quality Value: ");
  Serial.println(airQualityAnalog);

  // Print qualitative air quality
  Serial.print("Air Quality Category: ");
  Serial.println(airQualityCategory);

  // Print digital status
  if (airQualityDigital == HIGH) {
    Serial.println("Digital Status: Air Quality is Good");
  } else {
    Serial.println("Digital Status: Poor Air Quality Detected!");
  }

  // Add delay for better readability
  delay(500);  // 0.5 seconds delay between readings
}
```

6. Precautions

- Ensure the MQ135 sensor is powered by a stable 3.3V.
- Place the sensor in a well-ventilated environment for accurate readings.
- Warm up the MQ135 sensor for at least 5 minutes before taking measurements.

7. Expected Output

1. **Serial Monitor Output:**

   Analog Air Quality Value: 150

   Air Quality Category: Good

   Digital Status: Air Quality is Good
2. **Real-Time Air Quality:**
   - The analog value will vary based on gas concentration.
   - The category will update based on predefined thresholds.

RESULTS

The NodeMCU successfully monitors air quality using the MQ135 sensor. It categorizes air quality as "Good," "Moderate," "Unhealthy," or "Hazardous" based on analog readings and displays the results in real time on the Serial Monitor.

Original Output

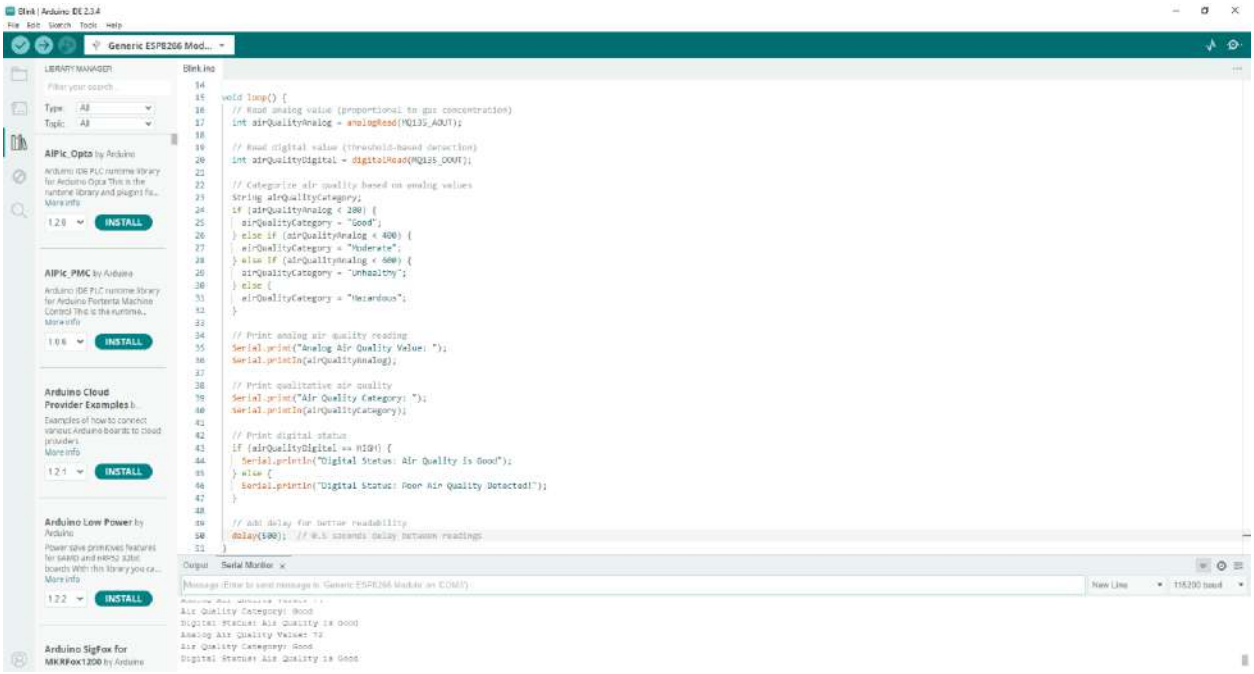


Figure:- Running Program

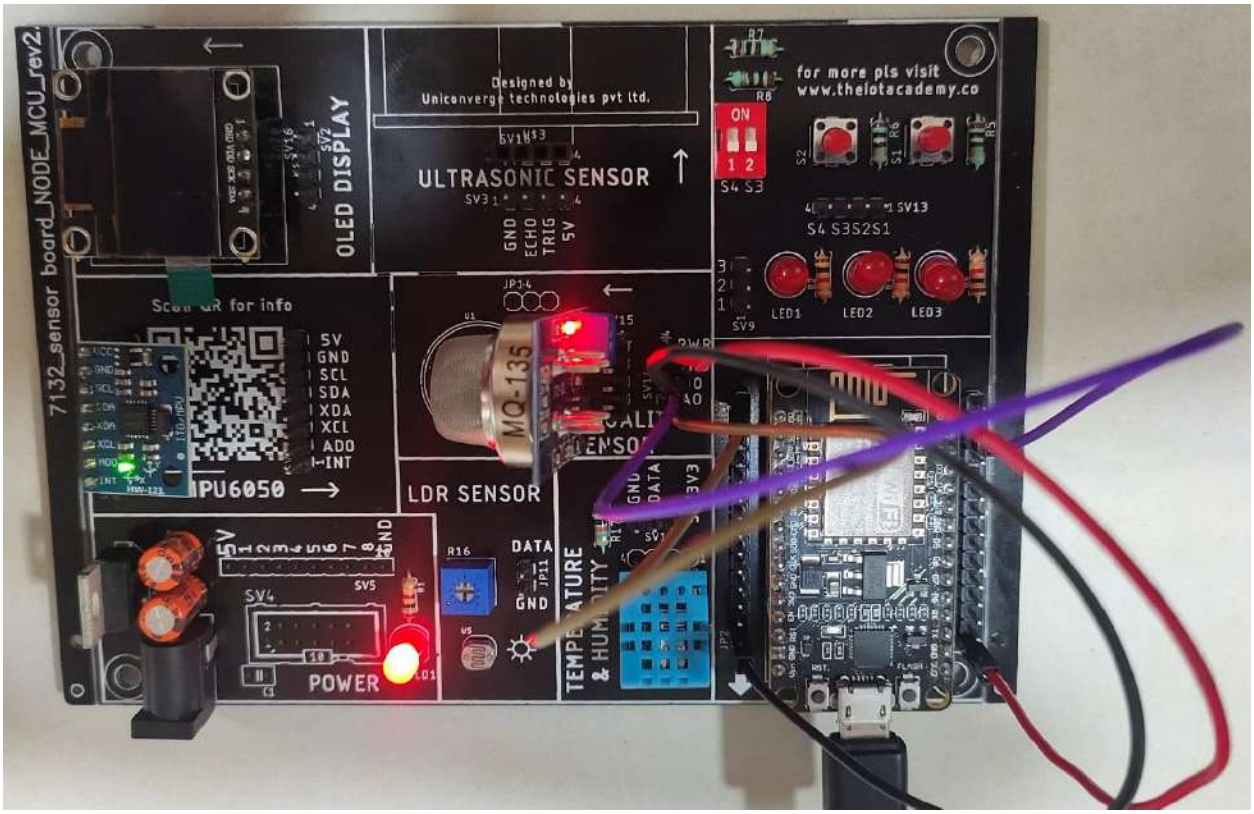


Figure:- Board Connection

Practical No:-10

Aim:- Write a program for Arduino by using Ultrasonic sensors and servo motor (HC- SR04), and make a smart dustbin.

1. Introduction

This project demonstrates an automated dustbin lid mechanism controlled by an ultrasonic sensor and a servo motor. The system opens the lid when an object is detected within a specified range and closes it after a short delay. This automation promotes hygienic waste disposal practices.

2. Structure

The program consists of:

1. Setting up the hardware components (ultrasonic sensor and servo motor).
2. Measuring the distance to detect the presence of an object.
3. Controlling the servo motor to open and close the lid based on distance measurements.

3. Hardware Required

- Arduino board (e.g., Uno)
- HC-SR04 Ultrasonic Sensor
- Servo motor (e.g., SG90)
- Jumper wires
- Breadboard
- Power source (USB or battery)

4. Connections

Component	Pin Name	Connection
Ultrasonic Sensor	VCC	5V
	GND	GND
	Trig	Pin 9
	Echo	Pin 8
Servo Motor	Signal	Pin 3
	VCC	5V
	GND	GND

5. Steps of Working

1. The ultrasonic sensor measures the distance to a nearby object.
2. If an object is detected within 15 cm, the program sends a signal to the servo motor to open the lid.
3. After a 2-second delay, the servo motor closes the lid.
4. If no object is detected, the lid remains closed.

6. Program

```
#include <Servo.h>

// Define pins for Ultrasonic Sensor
const int trigPin = 9;
const int echoPin = 8;

// Define servo motor
Servo dustbinLid;
const int servoPin = 3;

long previousDistance = 0;  // Store previous distance value
bool lidOpened = false;     // Flag to track lid status

void setup() {
  // Initialize Serial Monitor
  Serial.begin(9600);

  // Set pin modes
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  // Attach servo motor
  dustbinLid.attach(servoPin);
```

```
  // Start with the lid closed
  dustbinLid.write(0);
}

void loop() {
  // Measure distance using ultrasonic sensor
  long duration, distance;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Calculate distance in cm
  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2;

  // Print distance to Serial Monitor
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  // Check if the distance is within the threshold for opening the lid
  if (distance <= 15 && !lidOpened) {
    dustbinLid.write(90);  // Open lid
    delay(2000);           // Keep lid open for 2 seconds
    dustbinLid.write(0);   // Close lid
    lidOpened = true;      // Set lidOpened flag to true
  }
```

```
  // If object is no longer within range, reset lidOpened flag
  if (distance > 15 && lidOpened) {
    lidOpened = false;     // Reset lidOpened flag
  }

  delay(500);  // Short delay before next measurement
}
```

7. Precautions

1. Ensure proper connections to avoid short circuits.
2. The servo motor should not be overpowered; use a separate power source if needed.
3. Avoid placing the ultrasonic sensor in direct sunlight or near reflective surfaces, as this may interfere with accurate distance measurements.

8. Expected Output

- When an object is detected within 15 cm, the dustbin lid opens, remains open for 2 seconds, and then closes automatically.
- The measured distance is displayed on the Serial Monitor.

RESULTS

The automated dustbin lid system functions as expected, providing hands-free operation for hygienic waste disposal.

Original Output

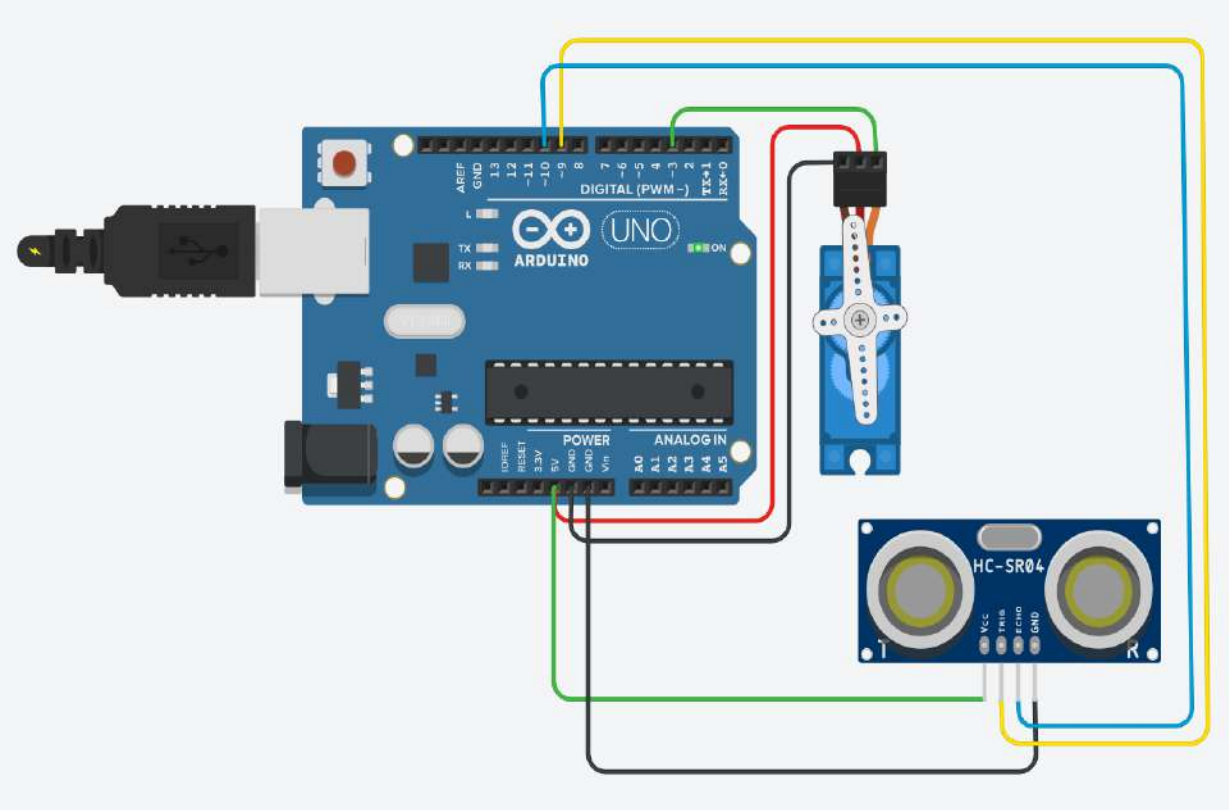


Figure:- Lid Close

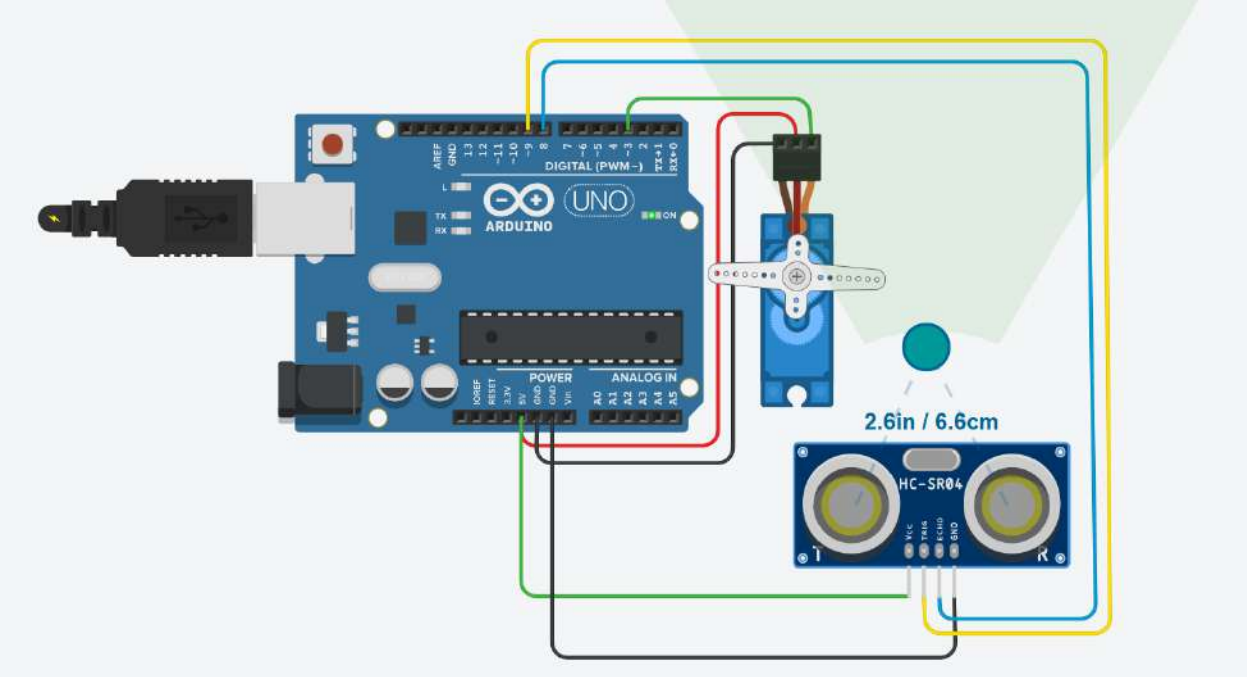


Figure:- Lid Open

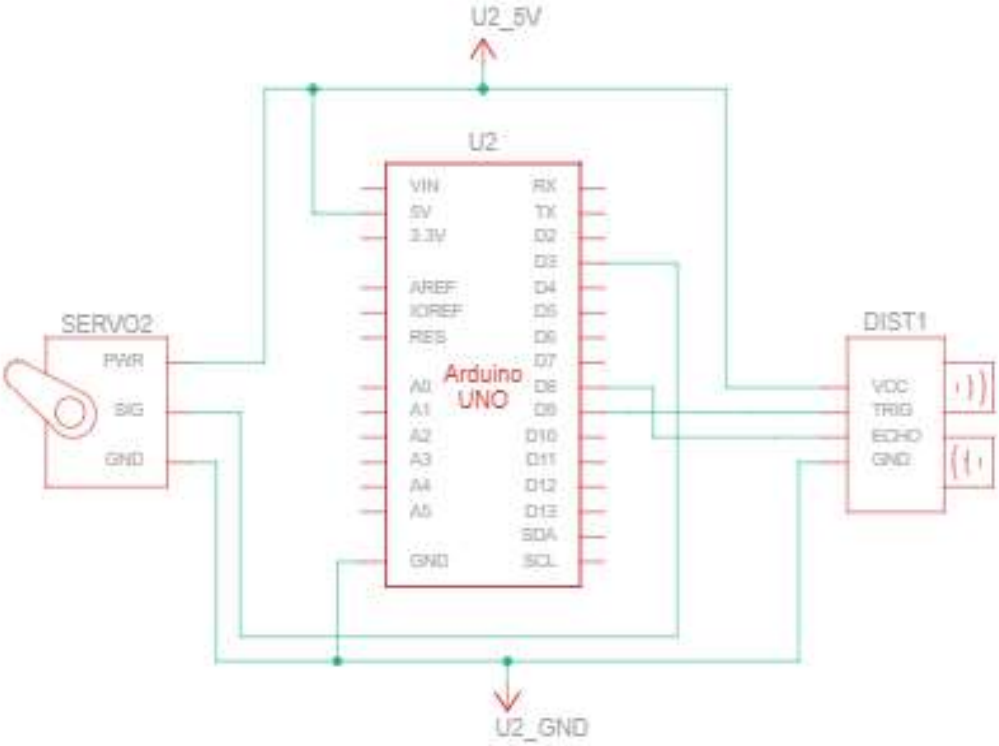


Figure:- Connection Diagram

Practical No:-11

Aim:- Write a program for Blink LED using NodeMCU (ESP8266) and Blynk app

1. Introduction

This project demonstrates how to control an LED remotely using the Blynk IoT platform and an ESP8266 microcontroller. Users can toggle the LED on/off using either the Blynk mobile app or web dashboard.

2. Structure

The system consists of three main components:

- ESP8266 microcontroller with an LED
- Blynk cloud server
- Blynk app/web interface for control

3. Hardware Required

- ESP8266 development board (NodeMCU or similar)
- LED
- 220Ω resistor
- Breadboard
- Jumper wires
- Micro USB cable for programming
- Smartphone with internet connection
- Computer with Arduino IDE installed

4. Connection Diagram

1. Connect LED anode (longer leg) to GPIO16 (D0) through a 220Ω resistor
2. Connect LED cathode (shorter leg) to GND
3. Power ESP8266 via USB or external power supply

5. Steps of Working

1. ESP8266 connects to WiFi network
2. Device establishes connection with Blynk cloud
3. Blynk app/web interface communicates with cloud
4. Cloud sends commands to ESP8266
5. ESP8266 controls LED based on received commands

6. Setting up Blynk Web Dashboard

1. Visit blynk.cloud in your browser
2. Login with your Blynk account
3. Select your device
4. Click "New Dashboard"
5. Add button widget:
    - Click "Add Widget"
    - Choose Button
    - Configure:
        - Name: LED Control
        - Datastream: V0
        - Switch mode: On/Off

7. Setting up Blynk Mobile App

1. Download Blynk IoT app from App Store/Play Store
2. Create new account or login
3. Create new device:
    - Click + icon
    - Choose ESP8266
    - Name your device
    - Copy the auth token
4. Add a button widget:
    - Click + to add widget

- Select Button
- Set output pin to V0
- Set mode to Switch

8. Program

```
// Include the library files
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

#define BLYNK_AUTH_TOKEN "j6Z0lztLmzt9Y30gNRvTx4L0vm7cJhFj" // Enter your Blynk auth token

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Saharsh Gera"; // Enter your WiFi name
char pass[] = "Saharsh123";   // Enter your WiFi password

#define LED_PIN 16 // GPIO16 corresponds to D0 on ESP8266

// Get the button value
BLYNK_WRITE(V0) {
  digitalWrite(LED_PIN, param.asInt());
}

void setup() {
  // Set the LED pin as an output pin
  pinMode(LED_PIN, OUTPUT);
  // Initialize the Blynk library
  Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
}

void loop() {
```

```
  // Run the Blynk library
  Blynk.run();
}
```

9. Precautions

1. Double-check WiFi credentials before uploading
2. Ensure correct GPIO pin connection
3. Use appropriate resistor value for LED
4. Don't share auth token publicly
5. Keep ESP8266 in range of WiFi router
6. Check power supply polarity

10. Expected Output

- LED should turn ON when button is pressed in app/web
- LED should turn OFF when button is released
- Changes should be near-instantaneous
- Status should sync between mobile and web interface

Troubleshooting

1. If LED doesn't respond:
    - Check physical connections
    - Verify WiFi connection
    - Confirm auth token
2. If connection fails:
    - Check WiFi signal strength
    - Verify network credentials
    - Restart ESP8266

RESULTS:

When you press the button in the Blynk mobile app or web dashboard, the LED connected to GPIO16 (D0) of ESP8266 will turn ON, and when you release the button, the LED will turn OFF. The changes are instantaneous with real-time synchronization between the LED state and the Blynk interface.

Original Output

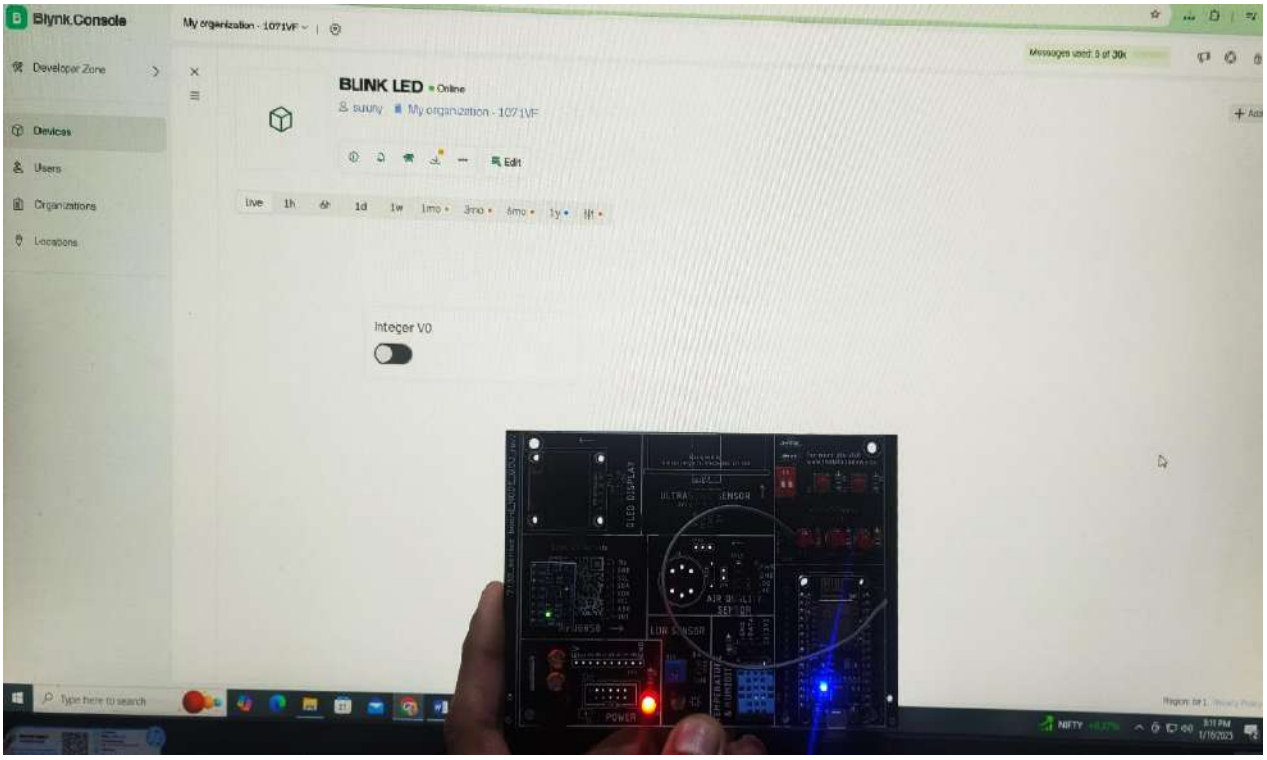


Figure:- LED OFF

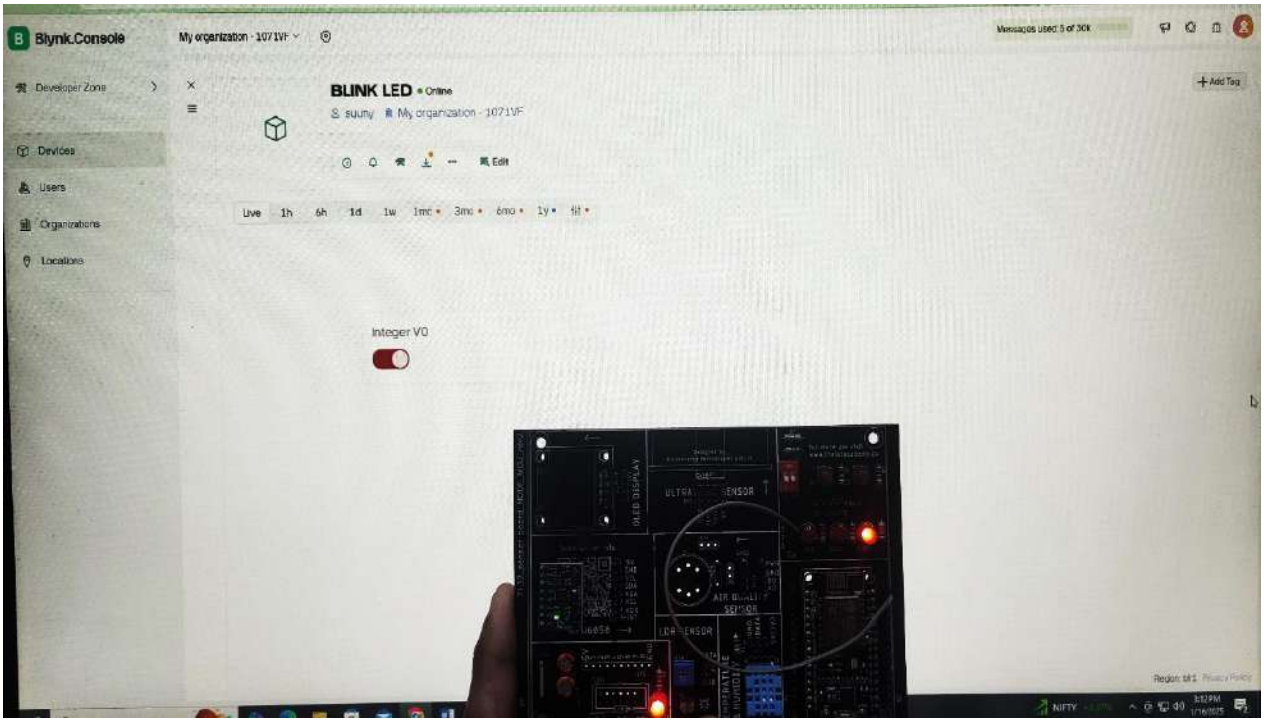


Figure:- LED ON

Write a program for controlling bulb on/off by using Blynk app

```
#define BLYNK_TEMPLATE_ID "TMPLZdgdgOwummO6"
#define BLYNK_DEVICE_NAME "Control Electric Bulb"
#define BLYNK_AUTH_TOKEN "5_S4mjFLbdfdggg7jZhdLDjgZxpKTWh_23liU4"

#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>



char auth[] = BLYNK_AUTH_TOKEN;

char ssid[] = "";  // type your wifi name
char pass[] = "";  // type your wifi password

int relaypin = D4;
void setup()
{
  Serial.begin(115200);
  Blynk.begin(auth, ssid, pass);
  pinMode(relaypin,OUTPUT);


  }

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

```
}
```

1. Introduction

2. Structure

3. Hardware Required

4. Connection Diagram

5. Steps of Working

6. Program

7. Precautions

8. Expected Output

RESULT: