Experiment No. 12

Aim: Upload data from a single sensor to ThingSpeak using ESP8266 (Node MCU)

Components Required:

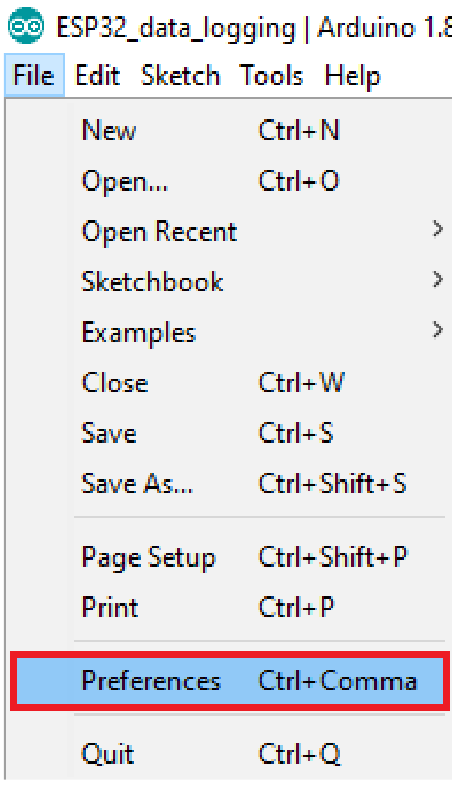
- ESP8266 (NodeMCU) development board
- DHT11 temperature and humidity sensor
- Breadboard and jumper wires
- USB cable for powering the NodeMCU
- A computer with Arduino IDE installed

Theory of DHT 11

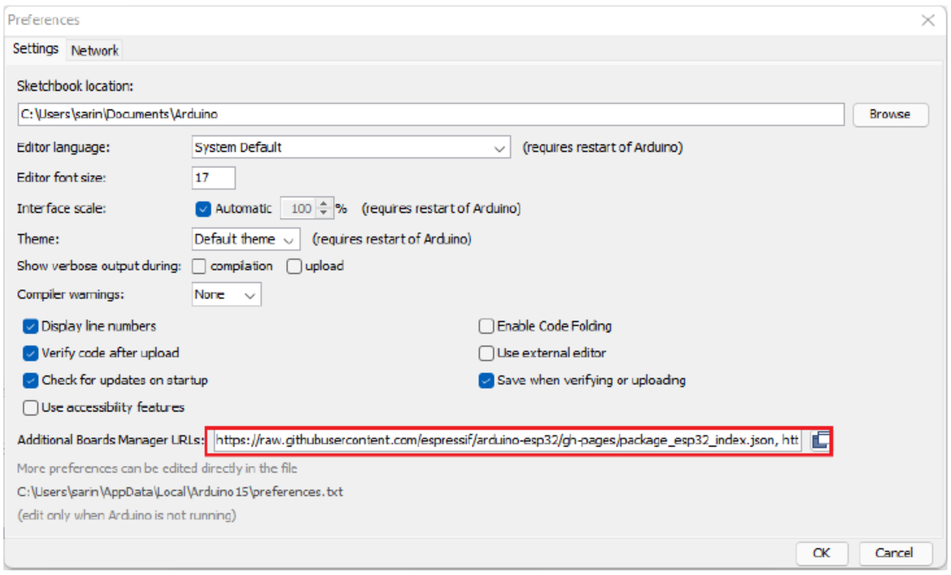
The DHT11 is a low-cost digital humidity and temperature sensor that uses a capacitive humidity sensor and a thermistor to measure the surrounding air's conditions. Its low price and compact size make it an attractive option for various applications, such as climate control systems, weather stations, and home automation. The sensor features a single-wire digital interface, making it easy to add to microcontroller-based projects. The data provided by the DHT11 sensor includes relative humidity and temperature readings, and it has a wide operating voltage range of 3.3V-5V.

To install the ESP32 board in your Arduino IDE, follow the following instructions:

i. In your Arduino IDE, go to File> Preferences



ii. Enter the following into the "Additional Board Manager URLs" field: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json



Preferences

Settings | Network

Sketchbook location: C:\Users\sarin\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 17

Interface scale: ☑ Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: ☐ compilation ☐ upload

Compiler warnings: None

☑ Display line numbers ☐ Enable Code Folding

☑ Verify code after upload ☐ Use external editor

☑ Check for updates on startup ☑ Save when verifying or uploading

☐ Use accessibility features

Additional Boards Manager URLs: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json, htt

More preferences can be edited directly in the file

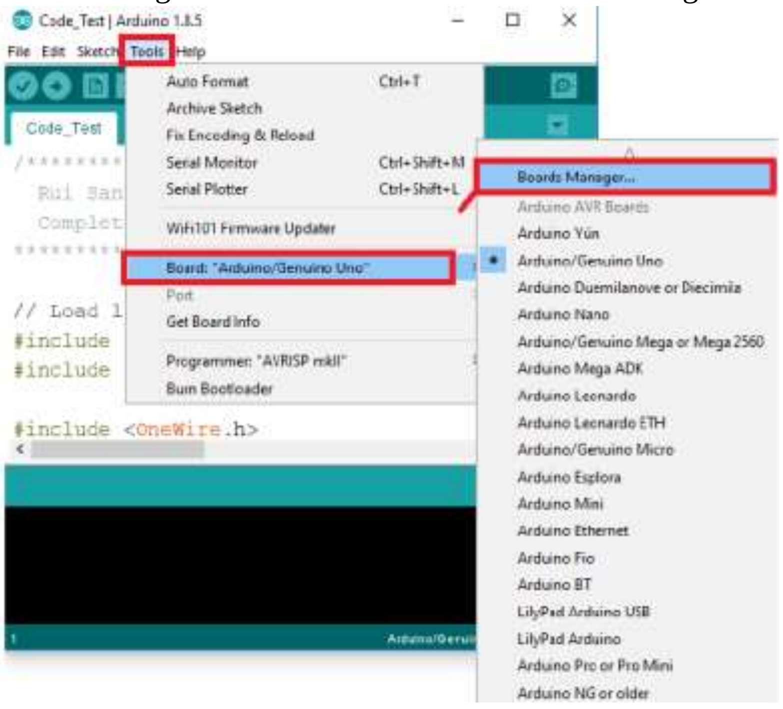
C:\Users\sarin\AppData\Local\Arduino15\preferences.txt

(edit only when Arduino is not running)

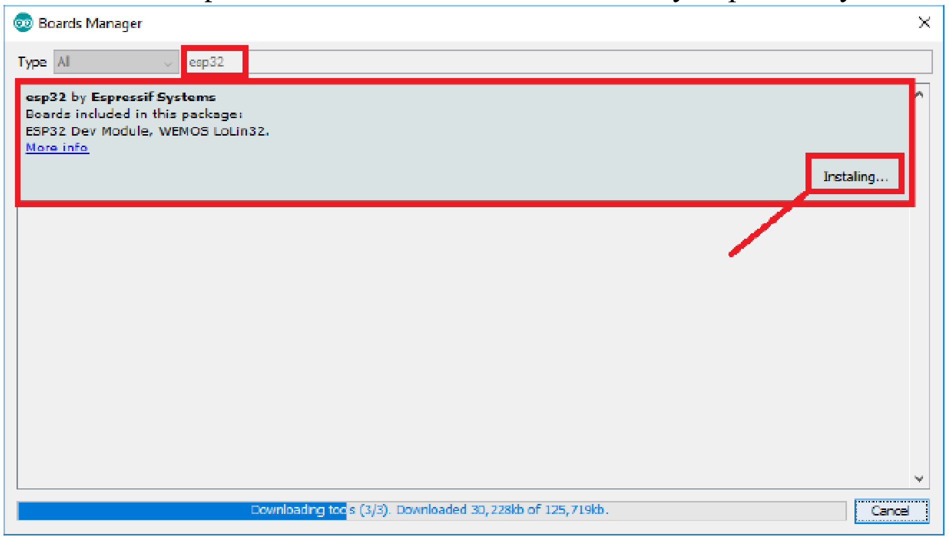
OK | Cancel

Then, click the “OK” button

iii.Open the Boards Manager. Go to Tools > Board > Boards Manager



iv. Search for ESP32 and press install button for the “ESP32 by Espressif Systems”



Boards Manager

Type All esp32

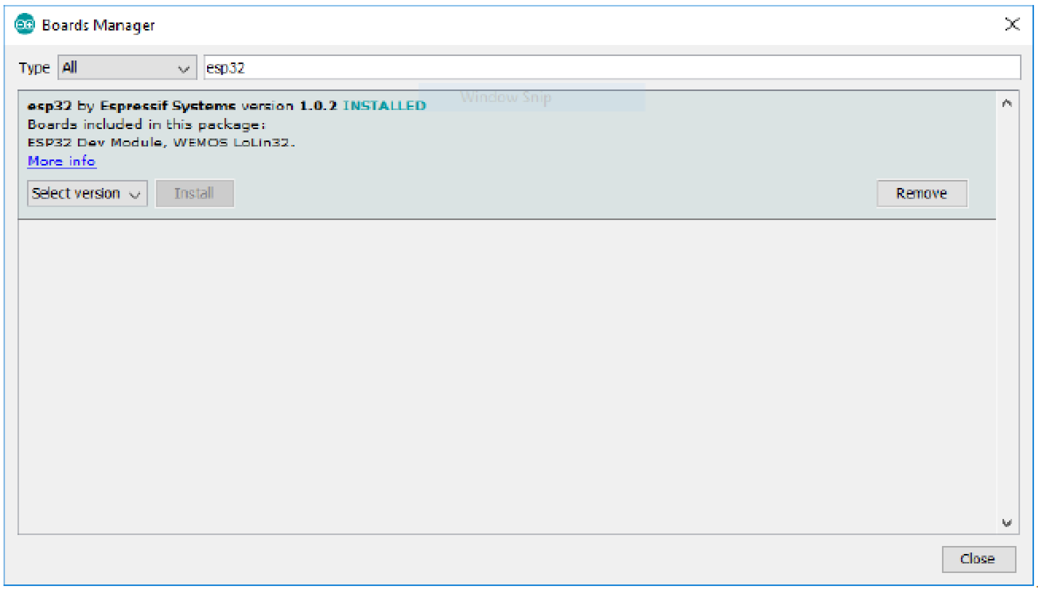
esp32 by **Espressif Systems**
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32.
More info

Installing...

Downloading tools (3/3). Downloaded 30,228kb of 125,719kb.

Cancel

v. That’s it. It should be installed after a few seconds



Boards Manager

Type All esp32

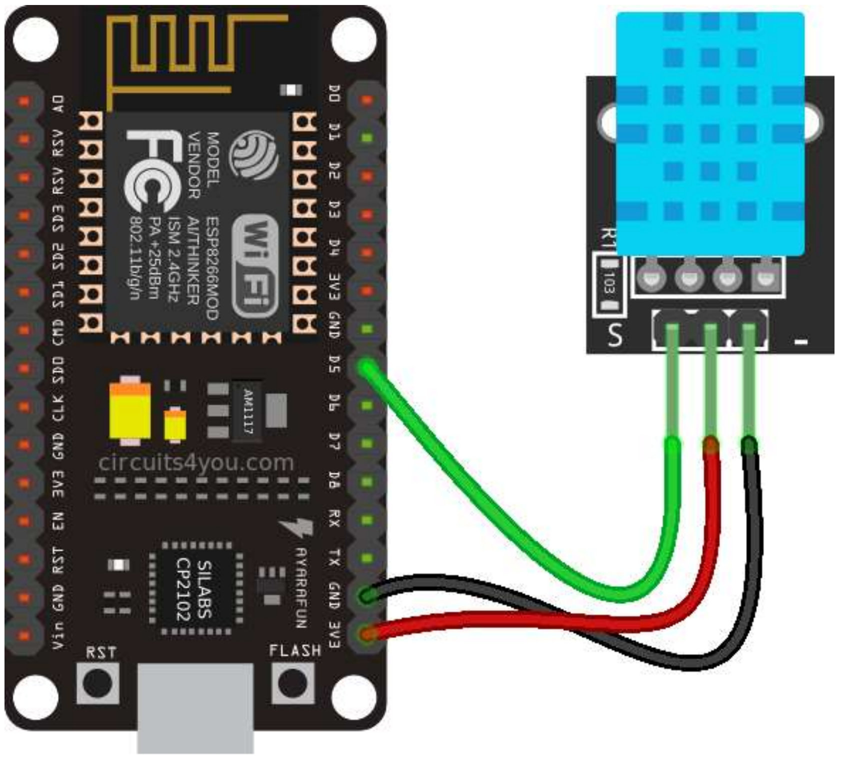
esp32 by Espressif Systems version 1.0.2 INSTALLED
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32.
More info

vi. Create an Account on ThingSpeak: Visit the ThingSpeak website (https://thingspeak.com) and create an account. Note down your ThingSpeak channel ID and API key, which will be used later to send data.

vii. Wire the components as shown in figure.

viii. Install Required Libraries Launch the Arduino IDE and install the necessary libraries by going to "Sketch" -> "Include Library" -> "Manage Libraries." Search for and install the following libraries: "DHT sensor library" by Adafruit, "ESP8266WiFi" by ESP8266 Community

Connection Diagram



Code Snippet

```
#include <ESP8266WiFi.h>
#include <DHT.h>

#define DHTPIN D4
#define DHTTYPE DHT11

const char* ssid = "YOUR_WIFI_SSID";
const char* password = "YOUR_WIFI_PASSWORD";
const char* server = "api.thingspeak.com";
const String apiKey = "YOUR_API_KEY";

DHT dht(DHTPIN, DHTTYPE);

WiFiClient client;

void setup() {
  Serial.begin(115200);
  delay(10);
  dht.begin();
  connectToWiFi();
}

void loop() {
  delay(2000);
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  String url = "/update?api_key=" + apiKey + "&field1=" +
String(temperature) + "&field2=" + String(humidity);

  if (client.connect(server, 80)) {
    client.print(String("GET ") + url + " HTTP/1.1\r\n" +
                 "Host: " + server + "\r\n" +
                 "Connection: close\r\n\r\n");
    delay(10);
    client.stop();
    Serial.println("Data sent to ThingSpeak successfully!");
  } else {
    Serial.println("Connection to ThingSpeak failed!");
  }
}

void connectToWiFi() {
  Serial.print("Connecting to WiFi...");
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(100);
  }

  Serial.println("Connected to WiFi!");
}
```

ix. Modify Wi-Fi and ThingSpeak Details: Replace "YOUR_WIFI_SSID" with the name of your Wi-Fi network, "YOUR_WIFI_PASSWORD" with your Wi-Fi password, and "YOUR_API_KEY" with your ThingSpeak API key.
x. Upload the Sketch
xi. Monitor the Serial Output: After the upload is complete, open the serial monitor by clicking "Tools" -> "Serial Monitor" in the Arduino IDE. Make sure the baud rate is set to 115200. You should see the NodeMCU connecting to the Wi-Fi network and sending data to ThingSpeak. Check the serial monitor for any error messages.
xii. Verify Data on ThingSpeak: Go to your ThingSpeak channel and open the corresponding channel view. You should see the temperature and humidity data being updated in real-time.

Experiment No. 13

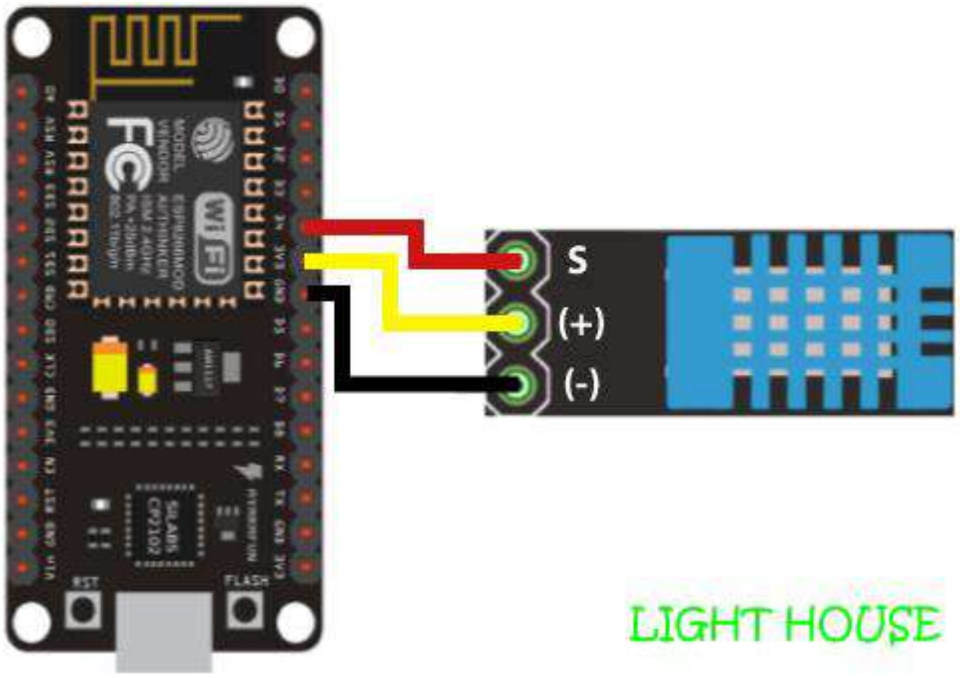
Aim: Interface DHT11 (Humidity Sensor) Using NodeMCU

Components Required

- Breadboard
- Micro USB Cable
- ESP8266 NodeMCU
- DHT11 Humidity and Temperature sensor
- Jumper Wires
- Arduino IDE

Wiring: Wire the components as follows:

- Connect the VCC pin of the DHT11 sensor to the 3.3V pin of the NodeMCU.
- Connect the GND pin of the DHT11 sensor to the GND pin of the NodeMCU.
- Connect the data pin of the DHT11 sensor to any GPIO pin of the NodeMCU.
- Connect the NodeMCU to your computer using a USB cable.



Code

```
#include <DHT.h>
#define DHTPIN 2
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  dht.begin();
}
void loop() {
  delay(2000);
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print(" °C\t");
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.println(" %");
}
```

Experiment No. 14

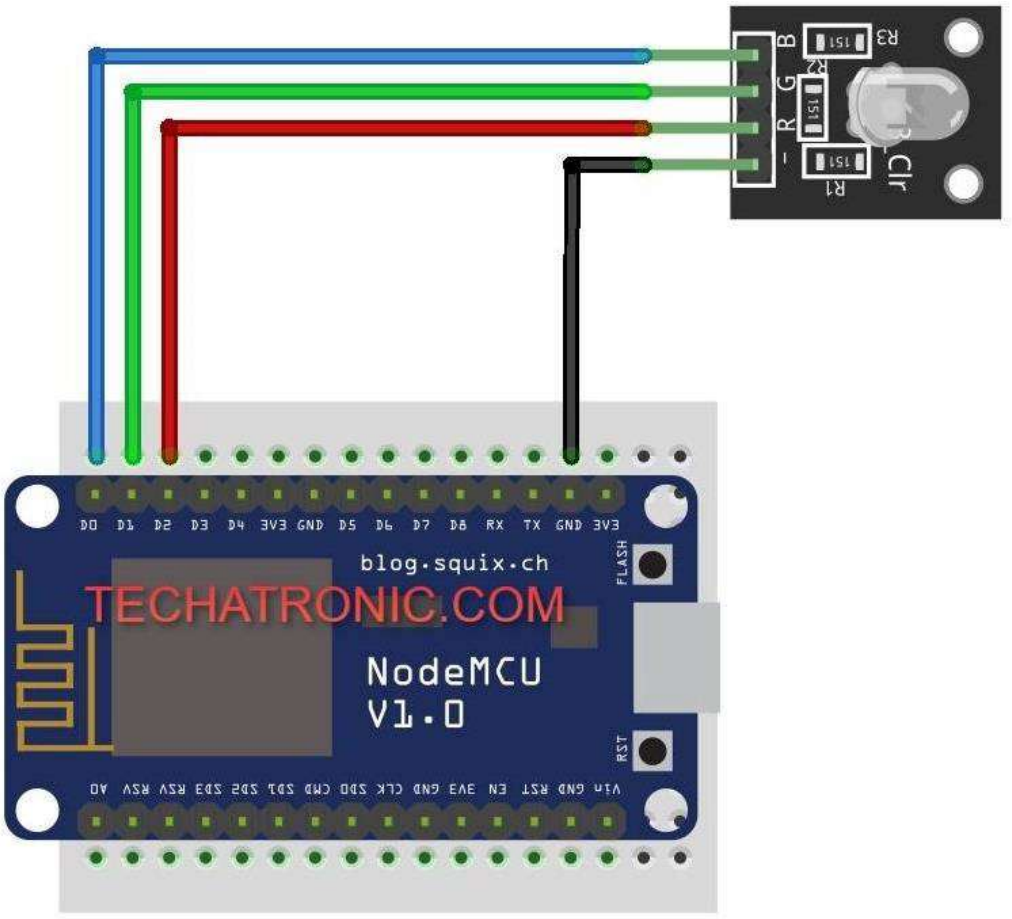
Aim: Interface RGB Led Using NodeMCU

Components Required

- Breadboard
- Micro USB Cable
- ESP8266 NodeMCU
- RGB Led sensor
- Jumper Wires
- Arduino IDE

Wiring: Wire the components as follows:

- GND pin of module – GND pin of NodeMCU.
- R pin (red light) of module – digital-2 pin of NodeMCU.
- G pin (green light) of module – digital-1 pin of NodeMCU.
- B pin (blue color) of module – digital-0 pin of NodeMCU.
- Connect the NodeMCU to your computer using a USB cable.


B
G
R
-
R1
R2
R3
Clr
D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
blog.squix.ch
TECHATRONIC.COM
NodeMCU V1.0
FLASH
RST
A0 RSV RSV SD3 SD2 SD1 CMD SDO CLK GND 3V3 EN RST GND Vin

Code

```
#define RED_PIN     12      // Digital pin 2 of NodeMCU
#define GREEN_PIN   13      // Digital pin 1 of NodeMCU
#define BLUE_PIN    14      // Digital pin 0 of NodeMCU
```

```
void setup() {
  pinMode(RED_PIN, OUTPUT);
  pinMode(GREEN_PIN, OUTPUT);
  pinMode(BLUE_PIN, OUTPUT);
}

void loop() {
  setColor(255, 0, 0);   // Red
  delay(1000);
  setColor(0, 255, 0);   // Green
  delay(1000);
  setColor(0, 0, 255);   // Blue
  delay(1000);
  setColor(255, 255, 255);   // White
  delay(1000);
}

void setColor(int red, int green, int blue) {
  analogWrite(RED_PIN, red);
  analogWrite(GREEN_PIN, green);
  analogWrite(BLUE_PIN, blue);
}
```

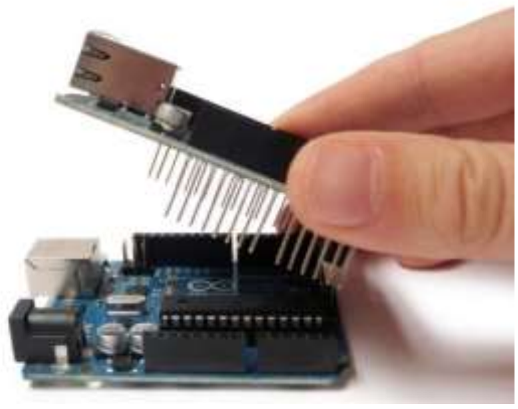
Experiment No. 15

Aim: Create an echo server with the Ethernet Shield over Arduino.

Components Required
- Arduino Uno Board
- Ethernet Shield (W5100)
- Ethernet Cable
- Computer with Arduino IDE installed
- USB Cable for Arduino

Wiring: Wire the components as follows:
1. Connect the Ethernet Shield to the Arduino board.
2. Connect the Arduino board to your computer using the USB cable.

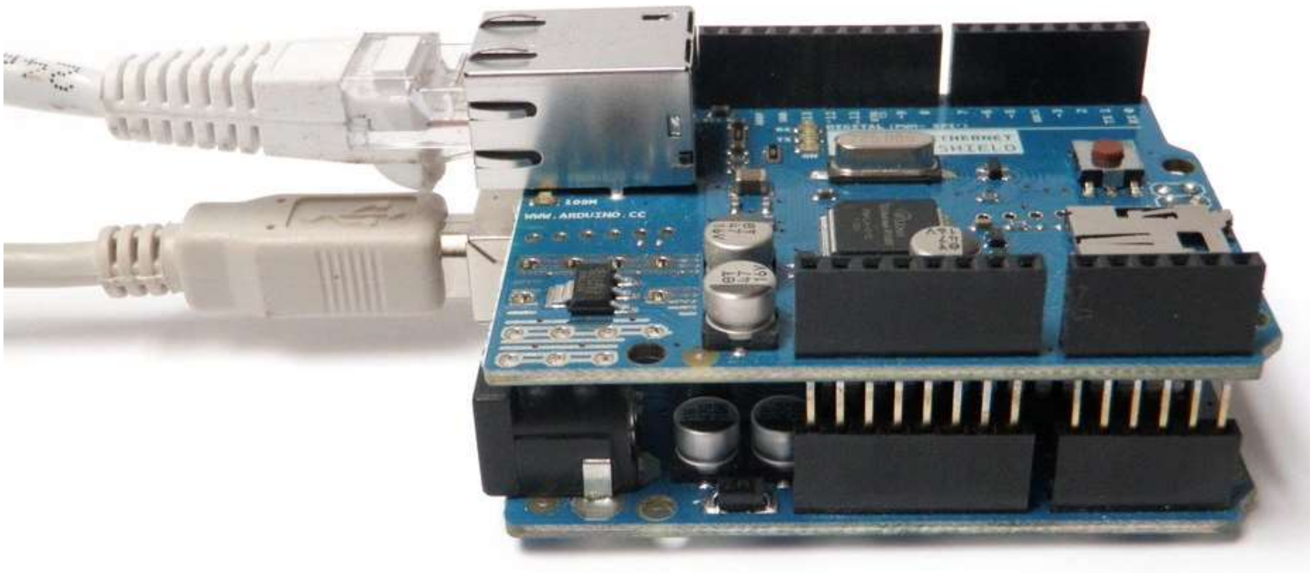
Installing Necessary Libraries
1. Open the Arduino IDE on your computer.
2. Go to "Sketch" -> "Include Library" -> "Manage Libraries..."
3. Search for "Ethernet" and install the latest version.



Code:

```
#include <SPI.h>
#include <Ethernet.h>

// Enter a MAC address for your controller below.
byte mac [] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };

// Set the IP address of the server
IPAddress ip(192, 168, 1, 177);

// Create a server object
EthernetServer server(23);

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  // start the Ethernet connection and the server:
```

```
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.print("Server is at ");
  Serial.println(Ethernet.localIP());
}

void loop() {
  // listen for incoming clients
  EthernetClient client = server.available();
  if (client) {
    Serial.println("new client");
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        // if you've gotten to the end of the line (received a newline
        // character) and the line is blank, the http request has ended,
        // so you can send a reply
        if (c == '\n' && currentLineIsBlank) {
          // send a standard http response header
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close");  // the connection will be closed after completion of the
response
          client.println("Refresh: 5");  // refresh the page automatically every 5 seconds
          client.println();
          client.println("<!DOCTYPE HTML>");
          client.println("<html>");

          // output the value of each analog input pin
          for (int analogChannel = 0; analogChannel < 6; analogChannel++) {
            int sensorReading = analogRead(analogChannel);
            client.print("analog input ");
            client.print(analogChannel);
            client.print(" is ");
            client.print(sensorReading);
            client.println("<br />");
          }
          client.println("</html>");
          break;
        }
        if (c == '\n') {
          // you're starting a new line
          currentLineIsBlank = true;
        } else if (c != '\r') {
          // you've gotten a character on the current line
          currentLineIsBlank = false;
        }
      }
    }
    // give the web browser time to receive the data
    delay(1);
    // close the connection:
    client.stop();
    Serial.println("client disconnected");
  }
}
```



Procedure:

1. Connect the Arduino to your computer using the USB cable.
2. Select the appropriate board and port from the "Tools" menu.
3. Click the upload button to upload the code to the Arduino.
4. Once the code is uploaded, open the serial monitor in the Arduino IDE.
5. Note the IP address assigned to the server (displayed in the serial monitor).
6. Open a web browser and enter the IP address in the address bar.
7. The browser should display analog readings from the Arduino.

Result:

This lab practical demonstrates the implementation of an Echo Server using Arduino and Ethernet Shield. Students should understand the code, the role of the server, and how to test the communication between the server and a client.