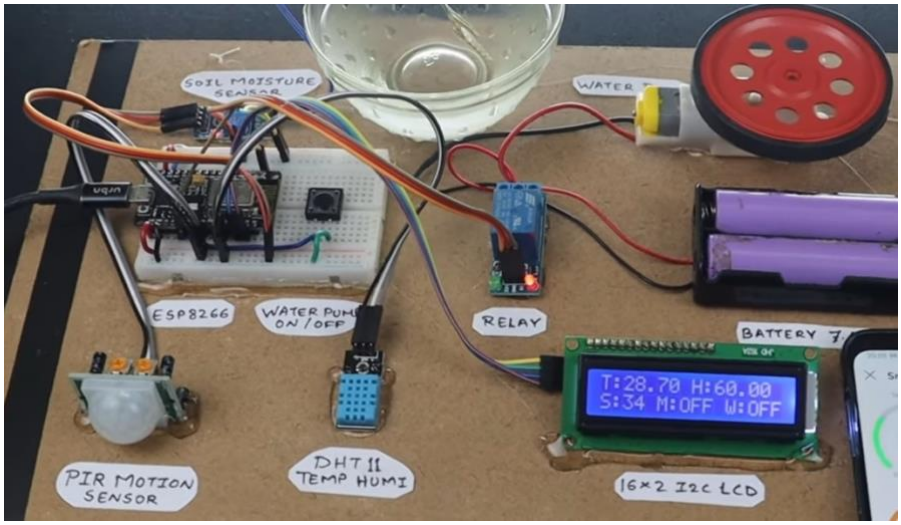
Smart Agriculture Monitoring System using NodeMCU (ESP8266)

By- Mayank Patel



The Smart Agriculture Monitoring System leverages the NodeMCU (ESP8266) microcontroller to enhance farming efficiency by automating environmental monitoring and enabling remote access to vital data. This system integrates sensors to collect real-time data on parameters such as soil moisture, temperature, humidity, and light intensity, which are crucial for optimizing crop growth.

The NodeMCU (ESP8266), with its built-in Wi-Fi capabilities, serves as the central processing unit. It collects data from connected sensors and uploads it to an IoT platform, such as ThingSpeak or Firebase, allowing farmers to monitor their fields remotely through a smartphone or computer. Alerts can be configured to notify users about critical changes, such as low soil moisture or extreme weather conditions, enabling timely interventions.

In addition to monitoring, the system can be expanded to include automation features, such as controlling irrigation systems or switching on lights, based on sensor readings. For instance, when soil moisture drops below a predefined threshold, the system can activate a water pump automatically.

This solution is cost-effective and scalable, making it suitable for small-scale and large-scale farming operations. By reducing water and energy wastage and improving crop yields, it contributes to sustainable agriculture. The use of NodeMCU (ESP8266) ensures seamless connectivity and reliable performance while keeping the overall cost of implementation low.

The Smart Agriculture Monitoring System is an excellent example of how IoT technologies can revolutionize traditional farming practices, ensuring better resource utilization, reducing manual effort, and fostering precision agriculture for a more productive future.

Material Required

1. Relay Board

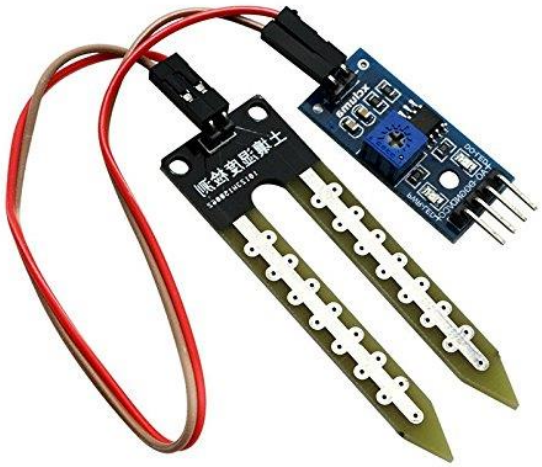
A relay board is an electronic module that uses relays to control high-voltage or high-current devices through low-voltage signals, such as those from microcontrollers like Arduino or NodeMCU. It acts as a bridge, isolating low-power circuits from high-power loads, enabling safe and efficient control of appliances like lights, fans, or motors.

2. Node MCU

The NodeMCU ESP8266 is a low-cost, open-source IoT development board featuring the ESP8266 Wi-Fi module.

It combines a powerful 32-bit microcontroller with built-in Wi-Fi capabilities, making it ideal for IoT and smart home projects. The board supports programming in Lua or Arduino IDE, providing flexibility for beginners and experts. With its GPIO pins, it can interface with various sensors, relays, and actuators. The NodeMCU supports protocols like MQTT, HTTP, and WebSocket for seamless communication with cloud platforms and mobile apps. Its compact design, affordability, and versatility make it popular for building connected devices and prototyping innovative IoT solutions.

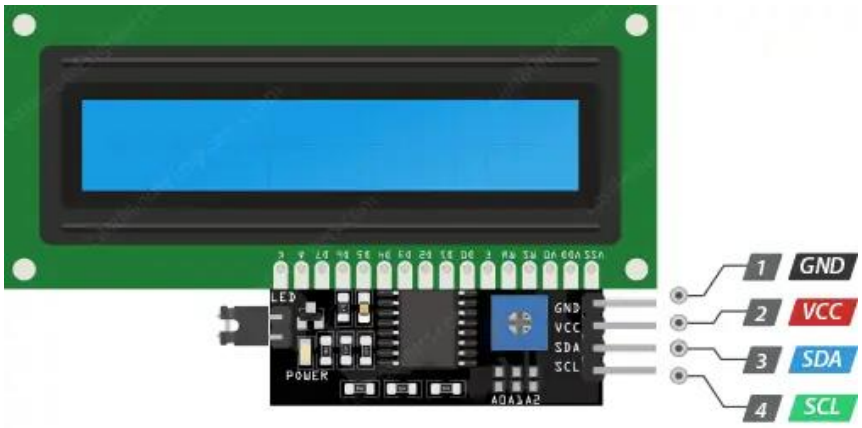
3. Soil Moisture Sensor



A **Soil Moisture Sensor** is a device designed to measure the water content in the soil. It plays a crucial role in modern agriculture, enabling precise irrigation management to optimize water use and promote healthy plant growth. The sensor typically consists of probes or electrodes that are inserted into the soil to detect moisture levels.

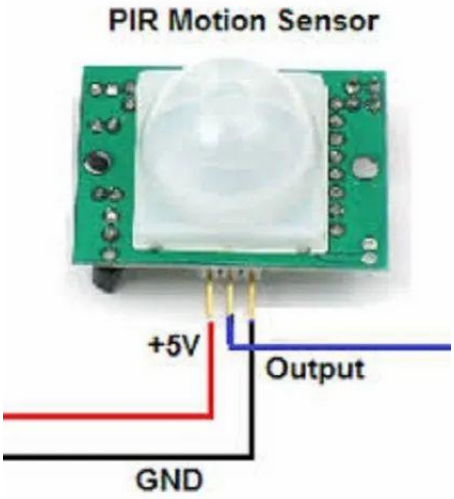
4. 16x2 I2C LCD Display

The **16x2 I2C LCD Display** is a compact, user-friendly module designed to display information in a variety of embedded systems and IoT applications. It features two rows with the capacity to show up to 16 characters per row, making it ideal for concise data representation. The inclusion of I2C (Inter-Integrated Circuit) communication simplifies its connection to microcontrollers like NodeMCU (ESP8266) or Arduino.



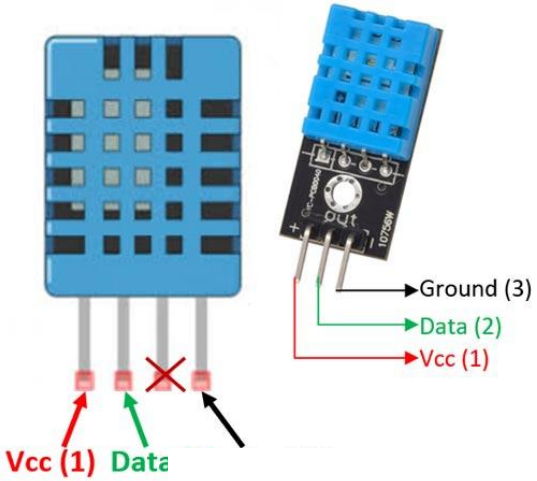
5. PIR Motion

A PIR (Passive Infrared) Motion Sensor is a device used to detect motion by measuring infrared radiation (heat) emitted by objects in its field of view. It is widely used in security systems, home automation, and other applications requiring motion detection. The sensor is “passive” because it does not emit any signals but instead senses the infrared radiation naturally emitted by objects.



6. DHT11 SENSOR

The **DHT11 Sensor** is a compact and affordable sensor designed to measure temperature and humidity. It is widely used in IoT, weather monitoring, and environmental sensing projects due to its ease of use, reliability, and integration capabilities.



7. Motor ON/OFF Button

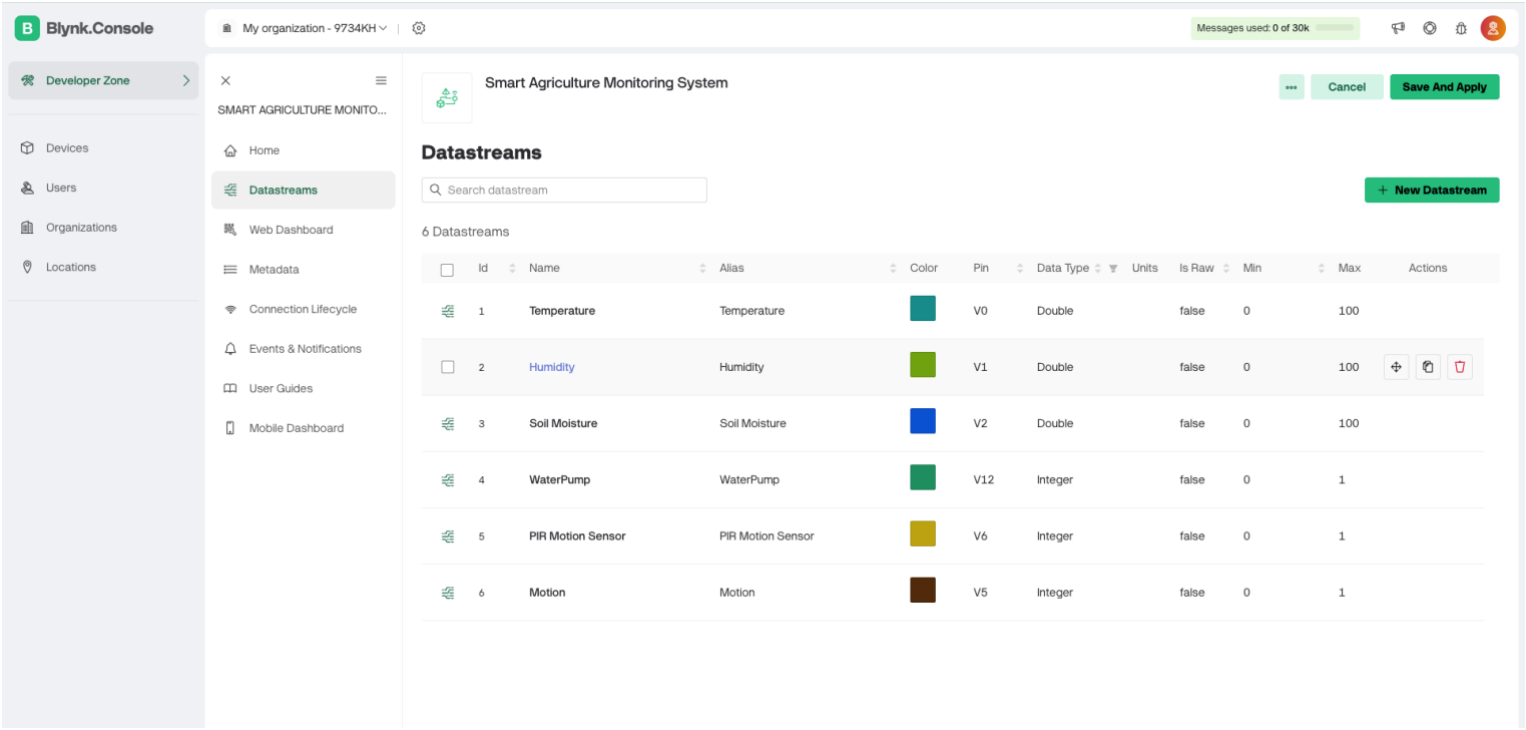
The Motor ON/OFF Button is a simple yet essential component used to manually control the operation of motors in various applications, including irrigation systems, home appliances, and industrial machines. It provides a direct and user-friendly interface for switching motors on and off.

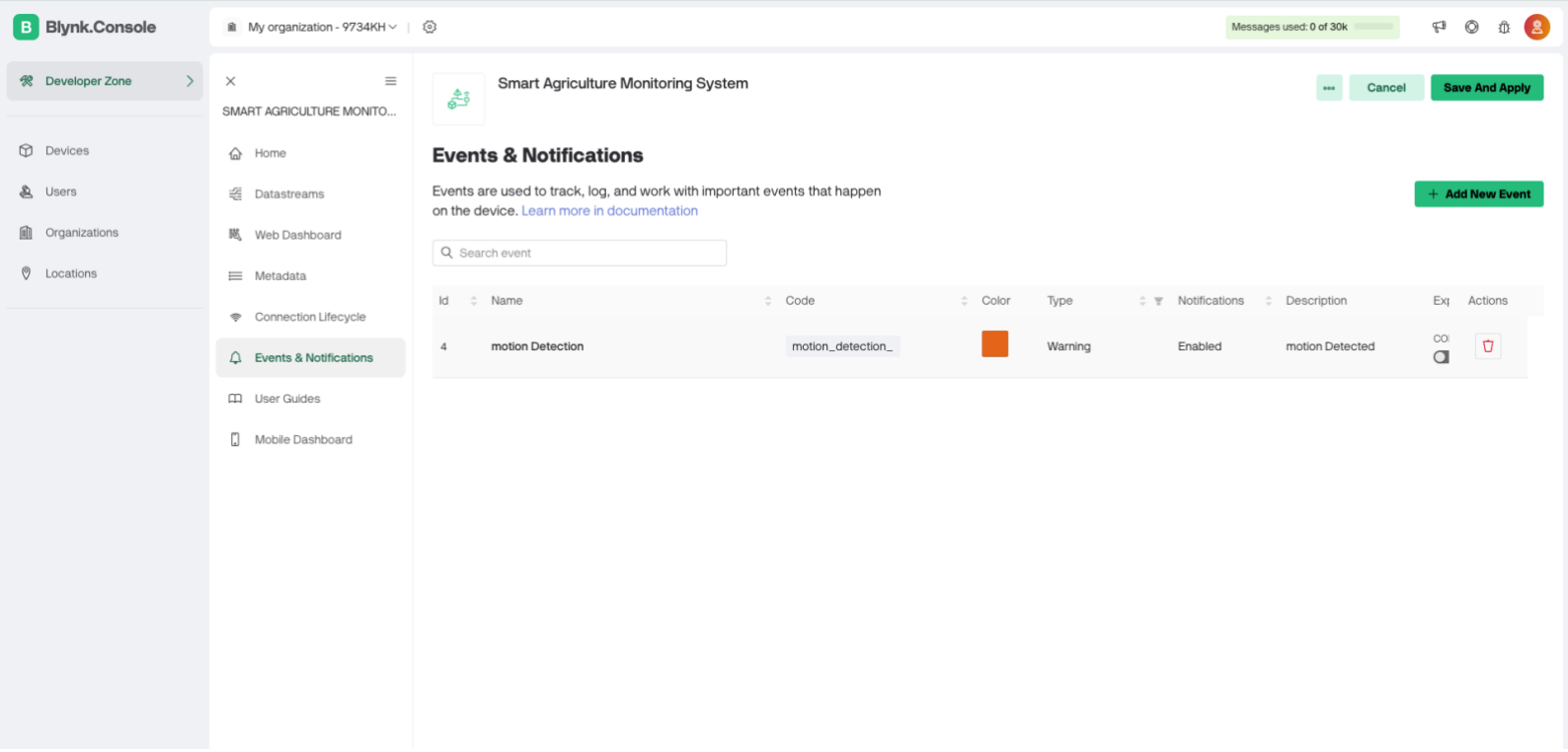
8. Water Pump

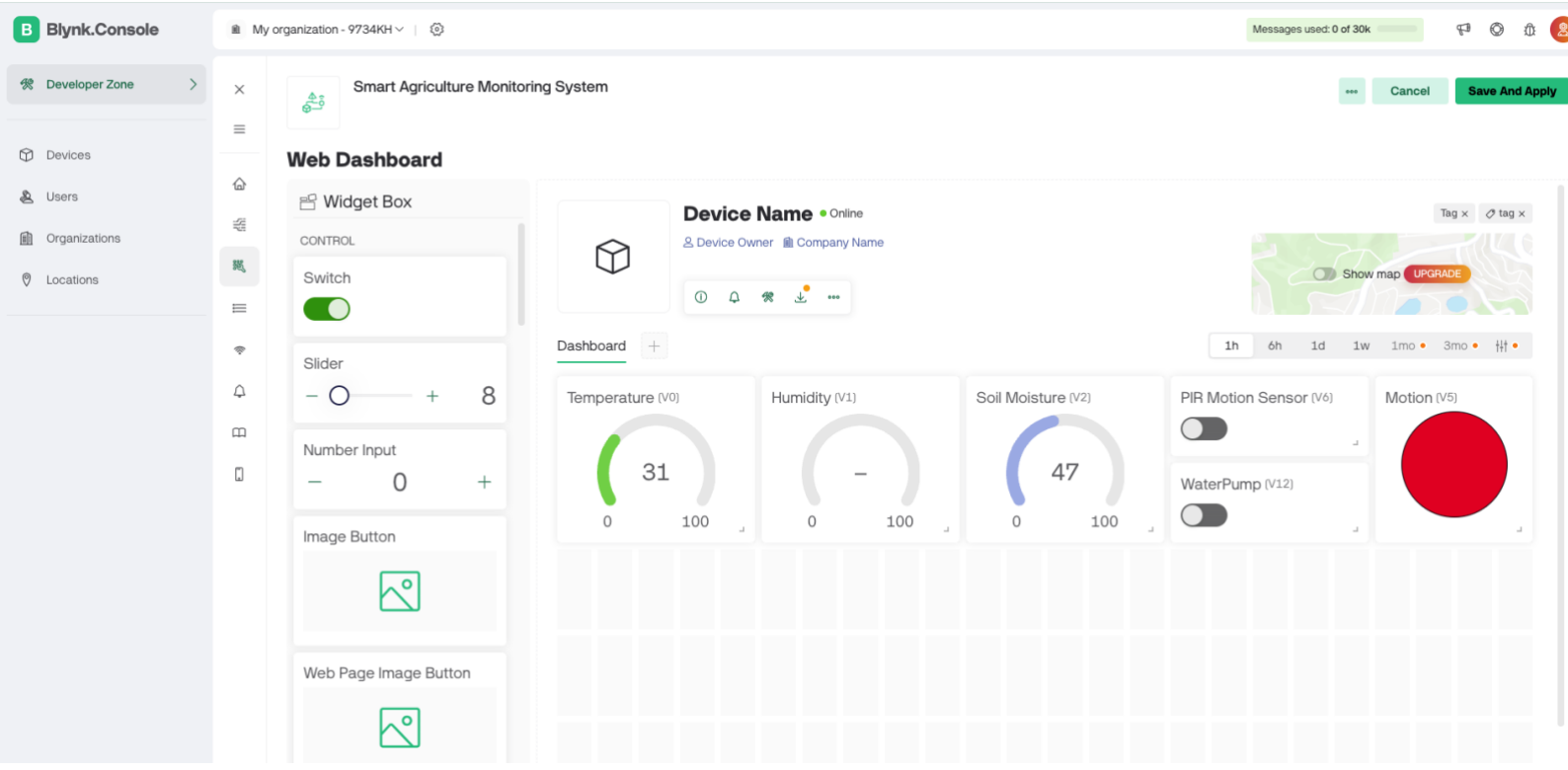
A **Water Pump** is an essential component in various systems where water transfer or circulation is required. In smart agriculture and IoT-based irrigation systems, water pumps play a crucial role in delivering water to crops efficiently, either manually or automatically.

9. Blynk

Blynk is a user-friendly IoT platform that allows developers to build and control smart devices through a mobile app. It supports a wide range of hardware, including Arduino, NodeMCU, ESP8266, and Raspberry Pi, enabling seamless integration with sensors and actuators. The platform features a drag-and-drop app builder, allowing users to create custom dashboards for real-time monitoring and control of connected devices. Blynk supports cloud, local server options, and communication protocols like Wi-Fi, Bluetooth, and GSM. Its features include virtual pins, automation, and notifications, making it ideal for IoT projects such as home automation, smart gardening, or industrial monitoring systems.

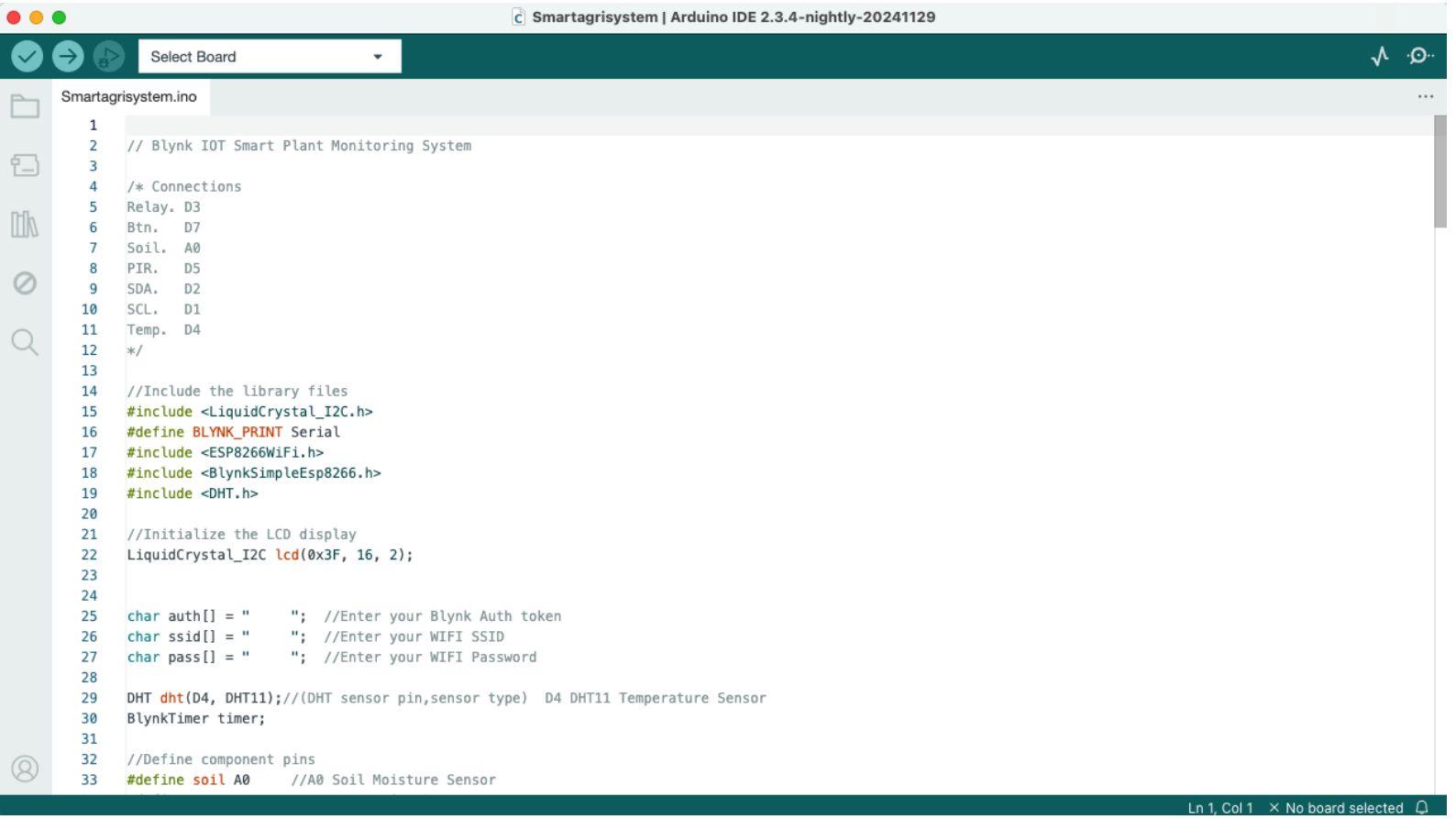






5. Arduino IDE

Arduino IDE is an open-source, cross-platform software application used to write, compile, and upload code to Arduino boards and compatible microcontrollers like NodeMCU or ESP32. It provides a simple interface with a code editor, a serial monitor for debugging, and built-in libraries for various hardware components. The IDE supports C and C++ programming, making it beginner-friendly while offering advanced capabilities for experienced developers. With a vast community and extensive library support, Arduino IDE simplifies creating projects like robotics, home automation, and IoT devices. Its versatility and ease of use make it a cornerstone of DIY electronics and prototyping.



Setup the IDE For NondeMCU

Important Link - https://github.com/esp8266/Arduino

Installing with Boards Manager

Starting with 1.6.4, Arduino allows the installation of third-party platform packages using Boards Manager. We have packages available for Windows, Mac OS, and Linux (32 and 64-bit).

- Download and install Arduino IDE 1.x or 2.x
- Start Arduino and open the Preferences window
- Enter **https://arduino.esp8266.com/stable/package_esp8266com_index.json** into the *File>Preferences>Additional Boards Manager URLs* field of the Arduino IDE. You can add multiple URLs, separating them with commas.
- Open Boards Manager from Tools > Board menu and install *esp8266* platform (and don't forget to select your ESP8266 board from Tools > Board menu after installation).

CODE Uses

```
// SSmart Agriculture Monitoring System using NodeMCU (ESP8266)

/* Connections
Relay. D3
Btn.  D7
Soil.  A0
PIR.  D5
SDA.  D2
SCL.  D1
Temp.  D4
*/

//Include the library files
#include <LiquidCrystal_I2C.h>
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <DHT.h>

//Initialize the LCD display
LiquidCrystal_I2C lcd(0x3F, 16, 2);


char auth[] = "C6Y-PjdTczW6kc_VN0nfFO_LE26Ip-dF";  //Enter your Blynk Auth token
char ssid[] = "Nalla_Manus";  //Enter your WIFI SSID
char pass[] = "nininimaaab";  //Enter your WIFI Password

DHT dht(D4, DHT11);//(DHT sensor pin,sensor type)  D4 DHT11 Temperature Sensor
BlynkTimer timer;

//Define component pins
#define soil A0    //A0 Soil Moisture Sensor
```

```
#define PIR D5     //D5 PIR Motion Sensor
int PIR_ToggleValue;

void checkPhysicalButton();
int relay1State = LOW;
int pushButton1State = HIGH;
#define RELAY_PIN_1      D3   //D3 Relay
#define PUSH_BUTTON_1    D7   //D7 Button
#define VPIN_BUTTON_1    V12

//Create three variables for pressure
double T, P;
char status;



void setup() {
 Serial.begin(9600);
 lcd.begin();
 lcd.backlight();
 pinMode(PIR, INPUT);

 pinMode(RELAY_PIN_1, OUTPUT);
 digitalWrite(RELAY_PIN_1, LOW);
 pinMode(PUSH_BUTTON_1, INPUT_PULLUP);
 digitalWrite(RELAY_PIN_1, relay1State);


 Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
 dht.begin();

 lcd.setCursor(0, 0);
 lcd.print("  Initializing  ");
 for (int a = 5; a <= 10; a++) {
   lcd.setCursor(a, 1);
   lcd.print(".");
   delay(500);
 }
 lcd.clear();
 lcd.setCursor(11, 1);
 lcd.print("W:OFF");
 //Call the function
 timer.setInterval(100L, soilMoistureSensor);
 timer.setInterval(100L, DHT11sensor);
 timer.setInterval(500L, checkPhysicalButton);
}


//Get the DHT11 sensor values
void DHT11sensor() {
 float h = dht.readHumidity();
 float t = dht.readTemperature();

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

```
 Blynk.virtualWrite(V0, t);
 Blynk.virtualWrite(V1, h);

 lcd.setCursor(0, 0);
 lcd.print("T:");
 lcd.print(t);

 lcd.setCursor(8, 0);
 lcd.print("H:");
 lcd.print(h);

}


//Get the soil moisture values
void soilMoistureSensor() {
 int value = analogRead(soil);
 value = map(value, 0, 1024, 0, 100);
 value = (value - 100) * -1;

 Blynk.virtualWrite(V3, value);
 lcd.setCursor(0, 1);
 lcd.print("S:");
 lcd.print(value);
 lcd.print(" ");

}

//Get the PIR sensor values
void PIRsensor() {
 bool value = digitalRead(PIR);
 if (value) {
   Blynk.logEvent("motion_detection","WARNNG! Motion Detected!"); //Enter your Event Name
   WidgetLED LED(V5);
   LED.on();
 } else {
   WidgetLED LED(V5);
   LED.off();
 }
 }

BLYNK_WRITE(V6)
{
 PIR_ToggleValue = param.asInt();
}


BLYNK_CONNECTED() {
 // Request the latest state from the server
 Blynk.syncVirtual(VPIN_BUTTON_1);
}

BLYNK_WRITE(VPIN_BUTTON_1) {
 relay1State = param.asInt();
 digitalWrite(RELAY_PIN_1, relay1State);
}
```

```
void checkPhysicalButton()
{
 if (digitalRead(PUSH_BUTTON_1) == LOW) {
   // pushButton1State is used to avoid sequential toggles
   if (pushButton1State != LOW) {

     // Toggle Relay state
     relay1State = !relay1State;
     digitalWrite(RELAY_PIN_1, relay1State);

     // Update Button Widget
     Blynk.virtualWrite(VPIN_BUTTON_1, relay1State);
   }
   pushButton1State = LOW;
 } else {
   pushButton1State = HIGH;
 }
}


void loop() {
   if (PIR_ToggleValue == 1)
   {
   lcd.setCursor(5, 1);
   lcd.print("M:ON ");
     PIRsensor();
     }
    else
    {
   lcd.setCursor(5, 1);
   lcd.print("M:OFF");
   WidgetLED LED(V5);
   LED.off();
    }

if (relay1State == HIGH)
{
 lcd.setCursor(11, 1);
 lcd.print("W:ON ");
 }
 else if (relay1State == LOW)
 {
  lcd.setCursor(11, 1);
  lcd.print("W:OFF");
  }


 Blynk.run();//Run the Blynk library
 timer.run();//Run the Blynk timer

 }
```

CIRCUIT DIAGRAM No.1