WEATHER MONITORING SYSTEM

USING BLYNK APP

Weather monitoring System Using Blynk App

Table of Contents

- Introduction

Overview of the Weather Monitoring System:

The weather monitoring system is a technological solution designed to collect, analyze, and display real-time weather data. It utilizes sensors and microcontrollers to gather information such as temperature, humidity, atmospheric pressure, wind speed, and precipitation. This data is then transmitted to a user-friendly interface, often a mobile app like the Blynk app, where users can access and visualize the weather information conveniently.

Purpose and Goals of the Project:

The primary purpose of the weather monitoring system is to provide accurate and up-to-date weather information to users. The project aims to fulfill several goals:

1. Real-time Data: The system must provide real-time weather data, allowing users to stay informed about current weather conditions in their area.

2. Accessibility: Make weather data easily accessible to users through a user-friendly and intuitive interface, such as the Blynk app.

3. Customization: Allow users to customize the data they want to monitor. They should be able to select specific parameters like temperature, humidity, or wind speed.

4. Alerts and Notifications: Implement alerting and notification mechanisms to inform users of significant weather changes or extreme conditions like storms, high winds, or temperature fluctuations.

5. Data Logging: Enable the system to log historical weather data for analysis and reference. This can be useful for tracking weather trends over time.

6. Remote Monitoring: Facilitate remote monitoring, allowing users to access weather data from anywhere, not just within their immediate vicinity.

7. User-Friendly: Ensure that the system is easy to set up and use, making it accessible to a wide range of users, including hobbyists, homeowners, and professionals.

8. Integration: Provide options for integrating the weather data with other smart home or IoT (Internet of Things) systems. For example, users may want to adjust their thermostats based on temperature trends.

9. Reliability: Ensure the system's reliability and accuracy by calibrating sensors and regularly updating software.

10. Cost-Effective: Strive to make the system cost-effective, using readily available hardware components and open-source software when possible.

11. Educational: If applicable, the project may also have an educational goal, aiming to teach users about weather monitoring technology, sensor calibration, and data analysis.

By achieving these goals, the weather monitoring system enhances users' ability to make informed decisions related to weather conditions, whether for daily planning, outdoor activities, or more specialized applications like agriculture, construction, or research.

- **Modules Used**:

1. DHT11 (Temperature and Humidity Sensor):

- The DHT11 is a low-cost sensor that measures both temperature and humidity.
- It is commonly used in weather monitoring systems to provide essential climate data.
- The DHT11 communicates with the NodeMcu (ESP8266) microcontroller to provide real-time temperature and humidity readings.

2. Rain Sensor:

- The rain sensor is used to detect the presence and intensity of rainfall.
- It typically consists of a set of conductive traces that can detect the conductivity changes caused by raindrops.
- The NodeMcu can process these changes to determine rainfall levels and trigger appropriate alerts or actions.

3. LDR Sensor (Light Dependent Resistor):

- The LDR sensor is sensitive to light levels and can be used to monitor ambient light conditions.

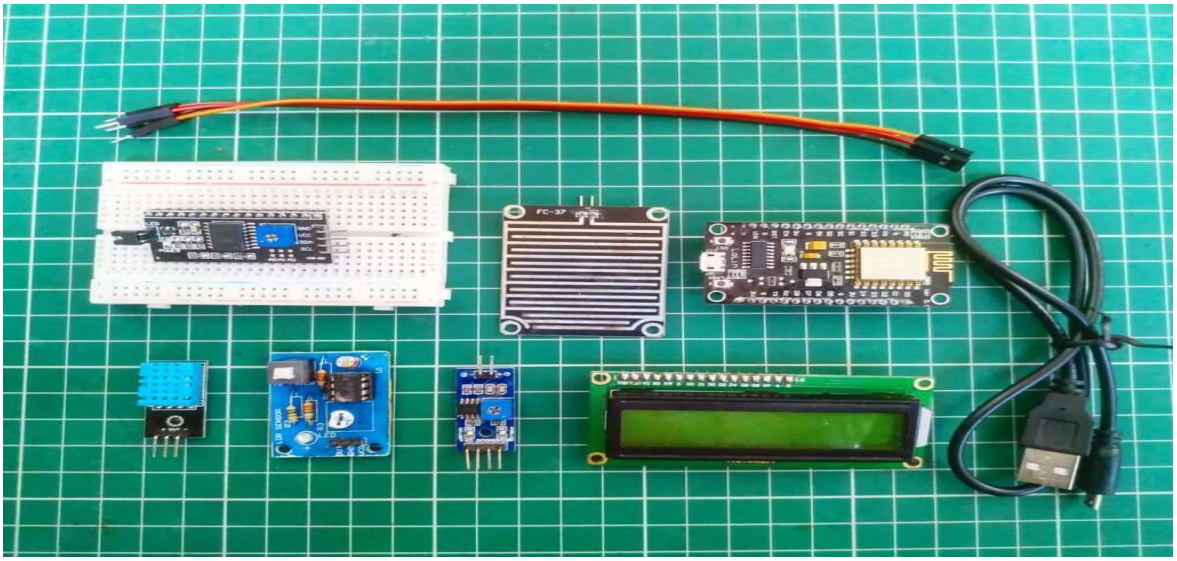
- While it may not be a direct component of weather monitoring, it can provide data on daylight hours or detect sudden changes in light (e.g., due to cloud cover or darkness).
- This data can be useful for weather analysis and as a trigger for other automation tasks.

4. NodeMcu (ESP8266):

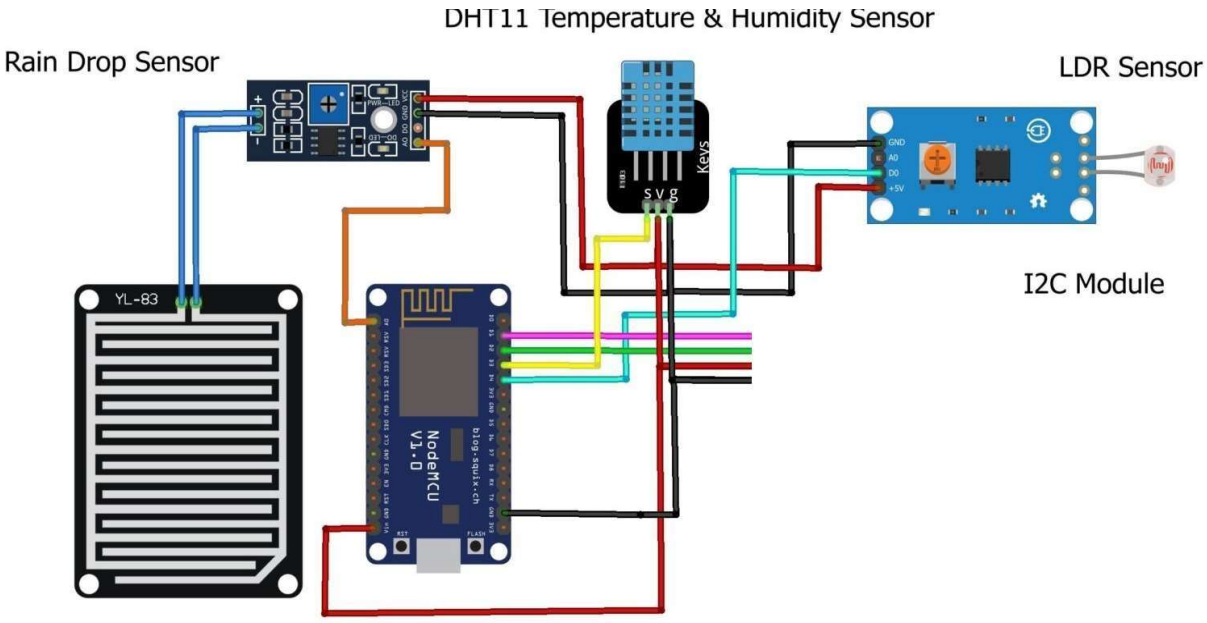
- The NodeMcu (ESP8266) is a versatile microcontroller with built-in Wi-Fi capabilities.
- It serves as the central processing unit in your weather monitoring system.
- The NodeMcu collects data from sensors, processes it, and sends it to the Blynk app or other destinations over Wi-Fi.
- Additionally, it can receive commands or alerts and trigger actions based on the weather data collected.

Material Required:

- **NodeMCU (EP8266)**
- **DHt11**
- **Rain Sensor**
- **LDR Sensor**
- **16×2 LCD with I2C module**
- **Breadboard**
- **Jumper Wires**
- **Blynk App with Wi-Fi connection**
- **Battery**



IoT Weather station Circuit Diagram



Connection table:

PINOUT

I2C LCD SCL, SDA —> D1, D2

DHT11 —> D3

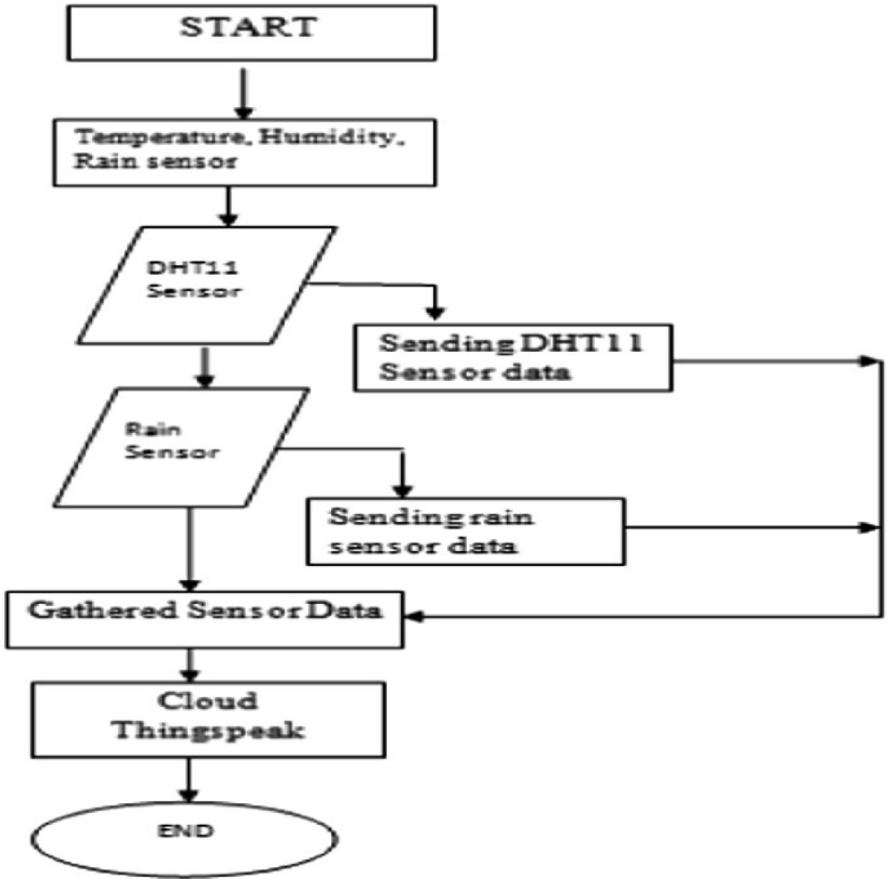
LDR —> D4

RAIN SENSOR —> A0

Working of the weather monitoring system:

- Take the above required components. Connect the components as per the circuit diagram.
- A Weather monitoring system is designed using the Nodemcu, Blynk app and mainly based on three sensors - rain sensor, DHT11 sensor, and LDR sensor.
- Through this, we can see factors such as rainfall, temperature, humidity, and amount of light. Also, we can monitor all this over the internet.

- Internet of Things (IoT) is the core root of linking all the sensors to the internet and monitoring the weather in real-time.
- The Nodemcu board connects to the Blynk app through the Blynk cloud. Also, it receives values through sensors.
- Then, the Nodemcu board sends those values to the Blynk app and the LCD display. Due to this process sensor values can be seen on the Blynk app interface and on the LCD.
- The data updated from this system can be accessed on the internet from anywhere in the world.
- The embedded system enables the user to access the various criteria and store the data in the cloud.



IoT Weather station Code

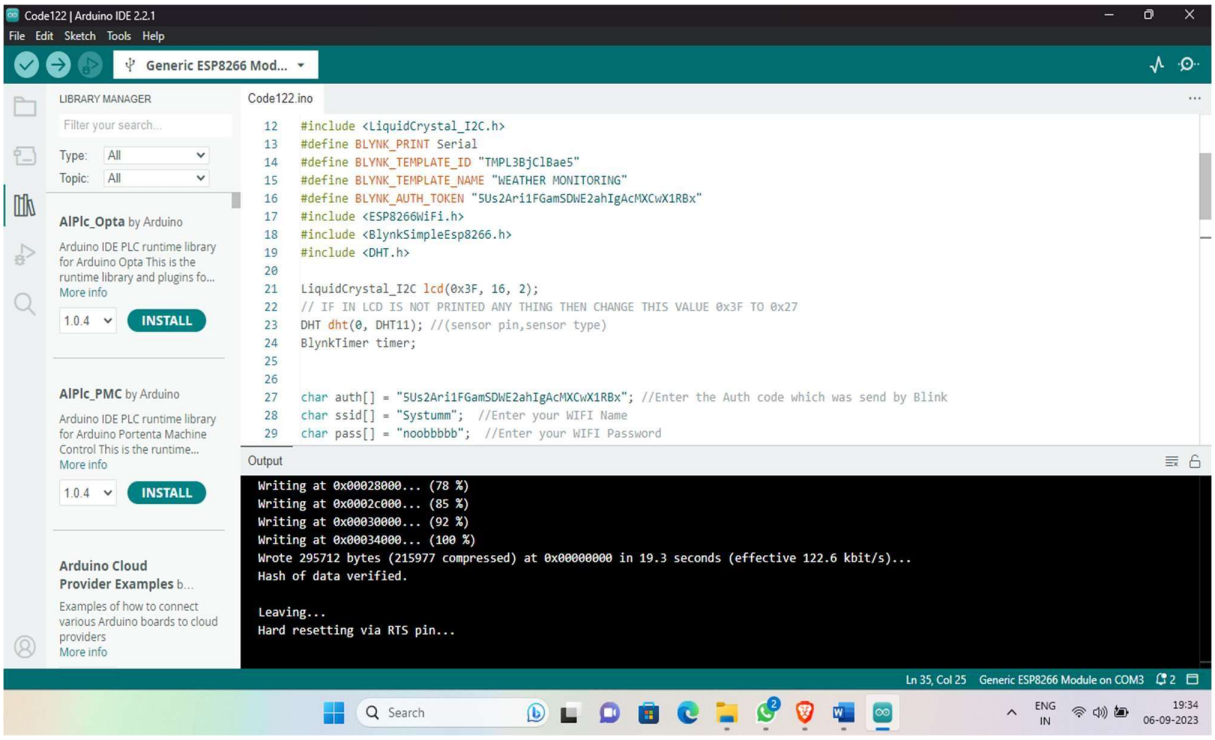
Explanation:- **First we include Some basic libraries as always for working of program. Also, we create dht11 and LCD object which stores the connecting information of both.**

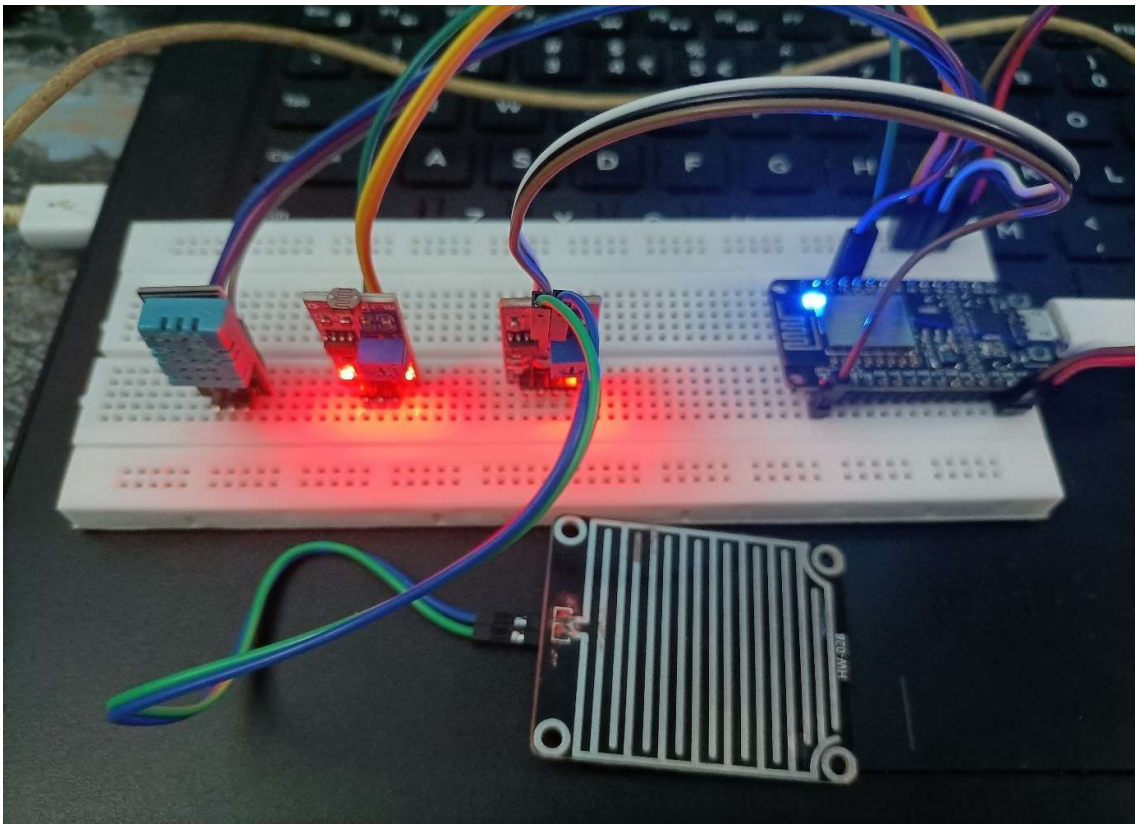
Now we create custom function weather which reads the Temperature and Humidity from DHT11 and stores it variables.

Code: Here is the main code

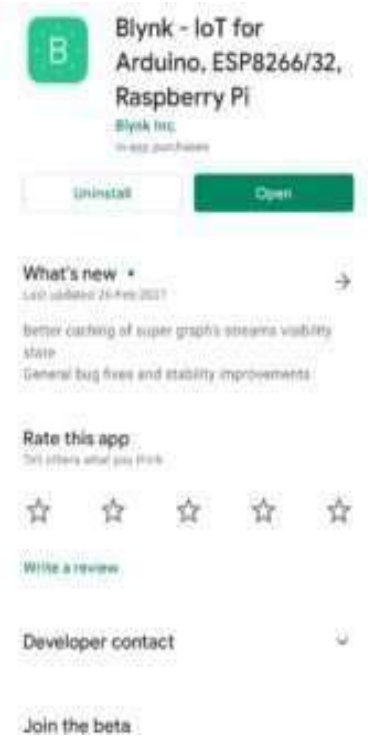
```
// I2C LIBRARY
// https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library
// BLYNK LIBRARY
// https://github.com/blynkkk/blynk-library
// ESP8266 LIBRARY
// https://github.com/ekstrand/ESP8266wifi
// Adafruit DHT sensor library:
// https://github.com/adafruit/DHT-sensor-library


#include <LiquidCrystal_I2C.h> #define BLYNK_PRINT Serial
#define BLYNK_TEMPLATE_ID "TMPL3BjClBae5" #define BLYNK_TEMPLATE_NAME "WEATHER MONITORING"
#define BLYNK_AUTH_TOKEN "5Us2Ari1FGamSDWE2ahIgAcMXCwX1RBx" #include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h> #include <DHT.h>

LiquidCrystal_I2C lcd(0x3F, 16, 2);
// IF IN LCD IS NOT PRINTED ANY THING THEN CHANGE THIS VALUE 0x3F TO 0x27
DHT dht(0, DHT11); //(sensor pin,sensor type) BlynkTimer timer;


char auth[] = "5Us2Ari1FGamSDWE2ahIgAcMXCwX1RBx"; //Enter the Auth code which was send by Blink
char ssid[] = "Systumm";        //Enter your WIFI Name
char pass[] = "noobbbbb";       //Enter your WIFI Password

void weather() {
float h = dht.readHumidity(); float t = dht.readTemperature(); int r = analogRead(A0);
bool l = digitalRead(2);

r = map(r, 0, 1023, 100, 0);
if (isnan(h) || isnan(t)) {
Serial.println("Failed to read from DHT sensor!"); return;
}
```

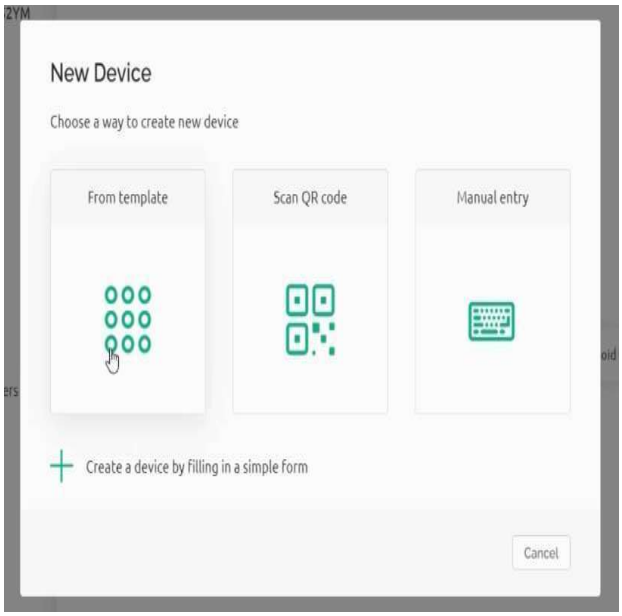
```
Blynk.virtualWrite(V0, t);          //V0 is for Temperature Blynk.virtualWrite(V1, h);       //V1 is for Humidity
Blynk.virtualWrite(V2, r);          //V2 is for Rainfall

if (l == 0) { WidgetLED led1(V3); led1.on(); lcd.setCursor(9, 1); lcd.print("L :");
lcd.print("High");
lcd.print(" ");
} else if (l == 1) { WidgetLED led1(V3); led1.off(); lcd.setCursor(9, 1); lcd.print("L :");
lcd.print("Low");
lcd.print(" ");
}

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

lcd.setCursor(0, 1); lcd.print("H :"); lcd.print(h);

lcd.setCursor(9, 0); lcd.print("R :"); lcd.print(r); lcd.print("          ");

}

void setup() {
Serial.begin(9600); // See the connection status in Serial Monitor lcd.begin();
lcd.backlight(); Blynk.begin(auth, ssid, pass); dht.begin();
// Setup a function to be called every second timer.setInterval(10L, weather);
}

void loop() {
Blynk.run(); // Initiates Blynk timer.run(); // Initiates SimpleTimer

}
```


Code122 | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Generic ESP8266 Mod...
LIBRARY MANAGER
Filter your search...
Type: All
Topic: All
AlPlc_Opta by Arduino
Arduino IDE PLC runtime library for Arduino Opta This is the runtime library and plugins fo...
More info
1.0.4
INSTALL
AlPlc_PMC by Arduino
Arduino IDE PLC runtime library for Arduino Portenta Machine Control This is the runtime...
More info
1.0.4
INSTALL
Arduino Cloud Provider Examples b...
Examples of how to connect various Arduino boards to cloud providers
More info
Code122.ino
12 #include <LiquidCrystal_I2C.h>
13 #define BLYNK_PRINT Serial
14 #define BLYNK_TEMPLATE_ID "TMPL3BjClBae5"
15 #define BLYNK_TEMPLATE_NAME "WEATHER MONITORING"
16 #define BLYNK_AUTH_TOKEN "5Us2Ari1FGamSDWE2ahIgAcMXCwX1RBx"
17 #include <ESP8266WiFi.h>
18 #include <BlynkSimpleEsp8266.h>
19 #include <DHT.h>
20
21 LiquidCrystal_I2C lcd(0x3F, 16, 2);
22 // IF IN LCD IS NOT PRINTED ANY THING THEN CHANGE THIS VALUE 0x3F TO 0x27
23 DHT dht(0, DHT11); //(sensor pin,sensor type)
24 BlynkTimer timer;
25
26
27 char auth[] = "5Us2Ari1FGamSDWE2ahIgAcMXCwX1RBx"; //Enter the Auth code which was send by Blink
28 char ssid[] = "Systumm"; //Enter your WIFI Name
29 char pass[] = "noobbbbb"; //Enter your WIFI Password
Output
Writing at 0x00028000... (78 %)
Writing at 0x0002c000... (85 %)
Writing at 0x00030000... (92 %)
Writing at 0x00034000... (100 %)
Wrote 295712 bytes (215977 compressed) at 0x00000000 in 19.3 seconds (effective 122.6 kbit/s)...
Hash of data verified.
Leaving...
Hard resetting via RTS pin...
Ln 35, Col 25 Generic ESP8266 Module on COM3
Search
ENG IN
19:34
06-09-2023



Blynk App installation:

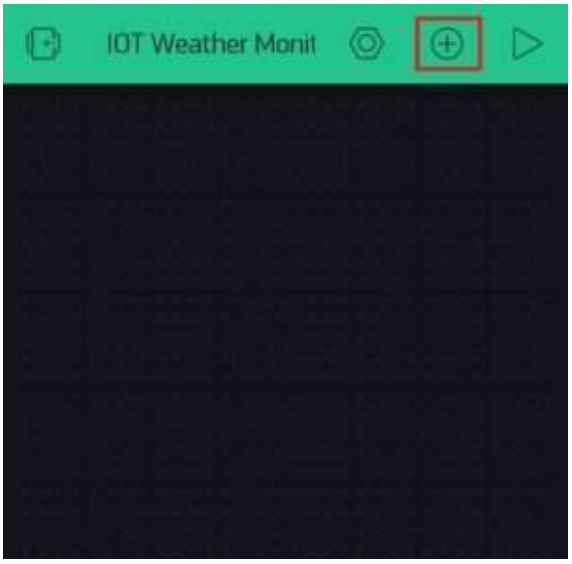




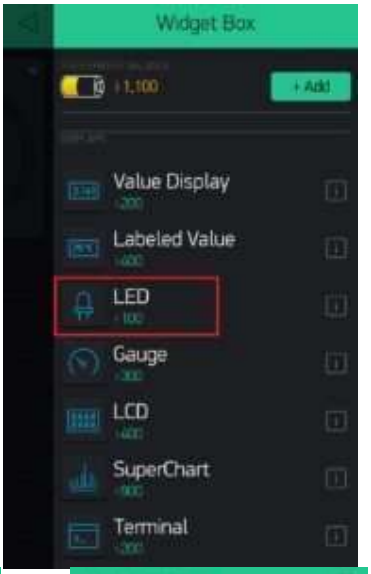
Create New Template

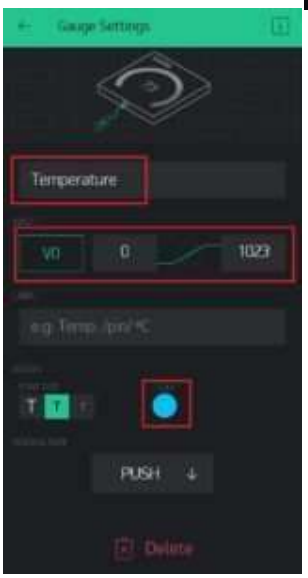
Log in to the App using your social accounts and email-id. Remember, this email-id will be used for sending Auth-Code.

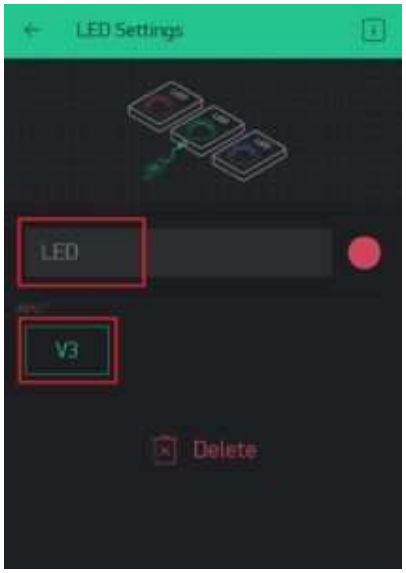
Now create a New Project with your preferred name. Here I have used 'IoT weather Monitoring'. As soon as you create, an Auth-Code will be sent to your registered email- id.



Now Add widgets to the main screen for displaying



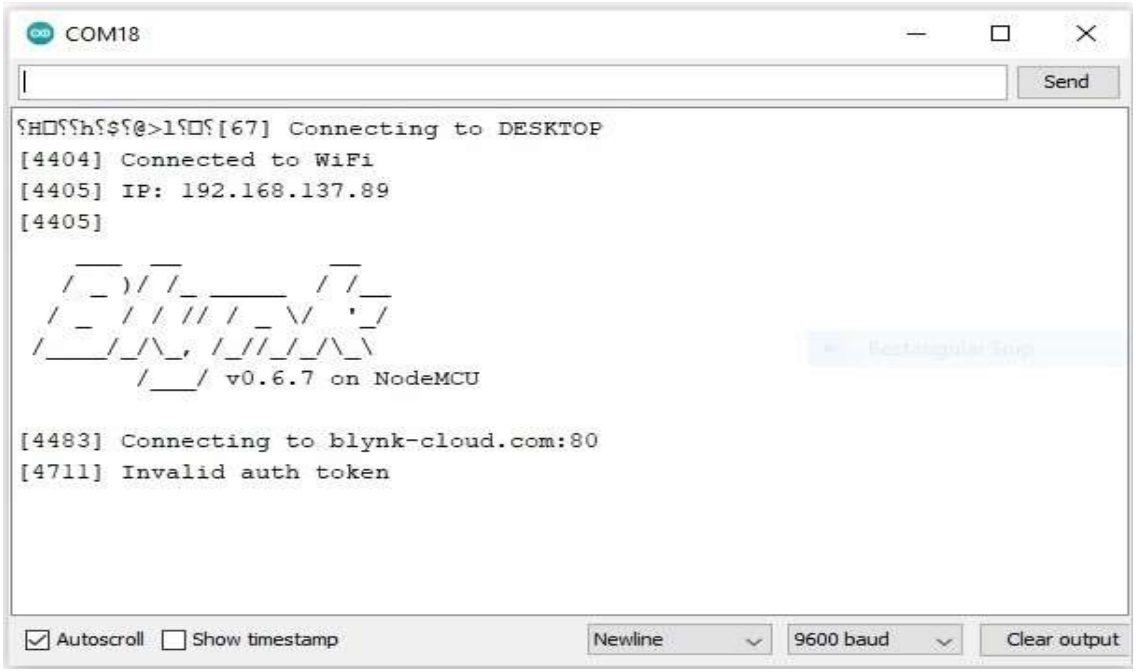




Now set things up as shown in the above images. And now you are good to start collecting information from NodeMCU.

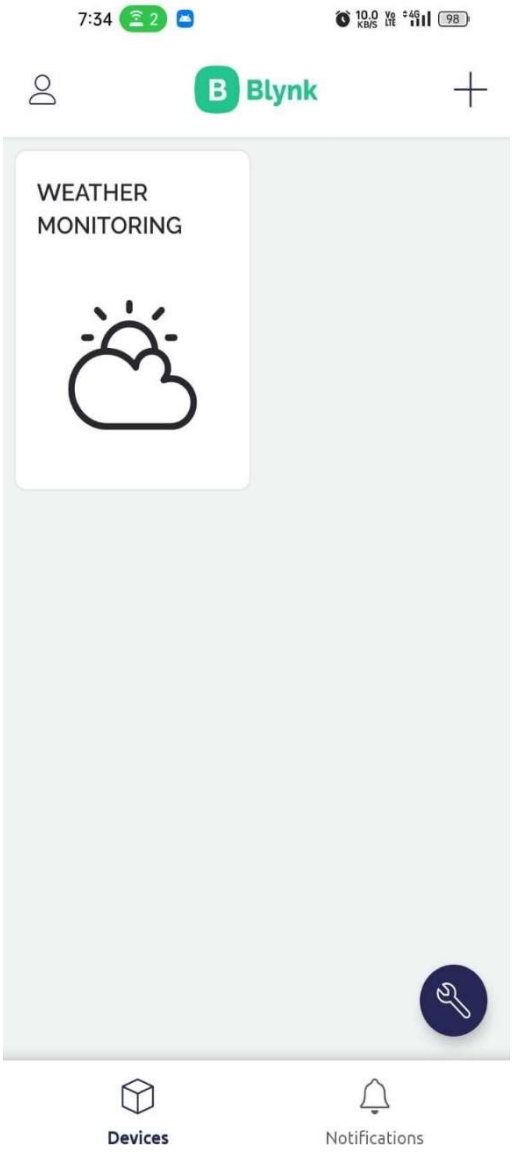
After that, you may see values are changing both on Blynk App and LCD.

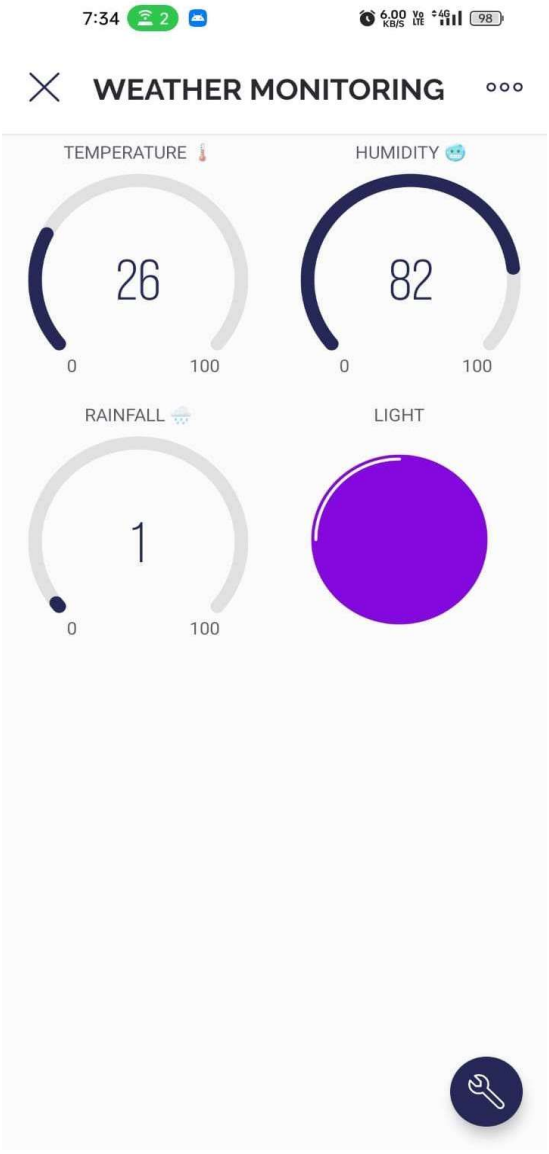
Now power up your NodeMCU and start Serial Monitor. You'll see output something like this.



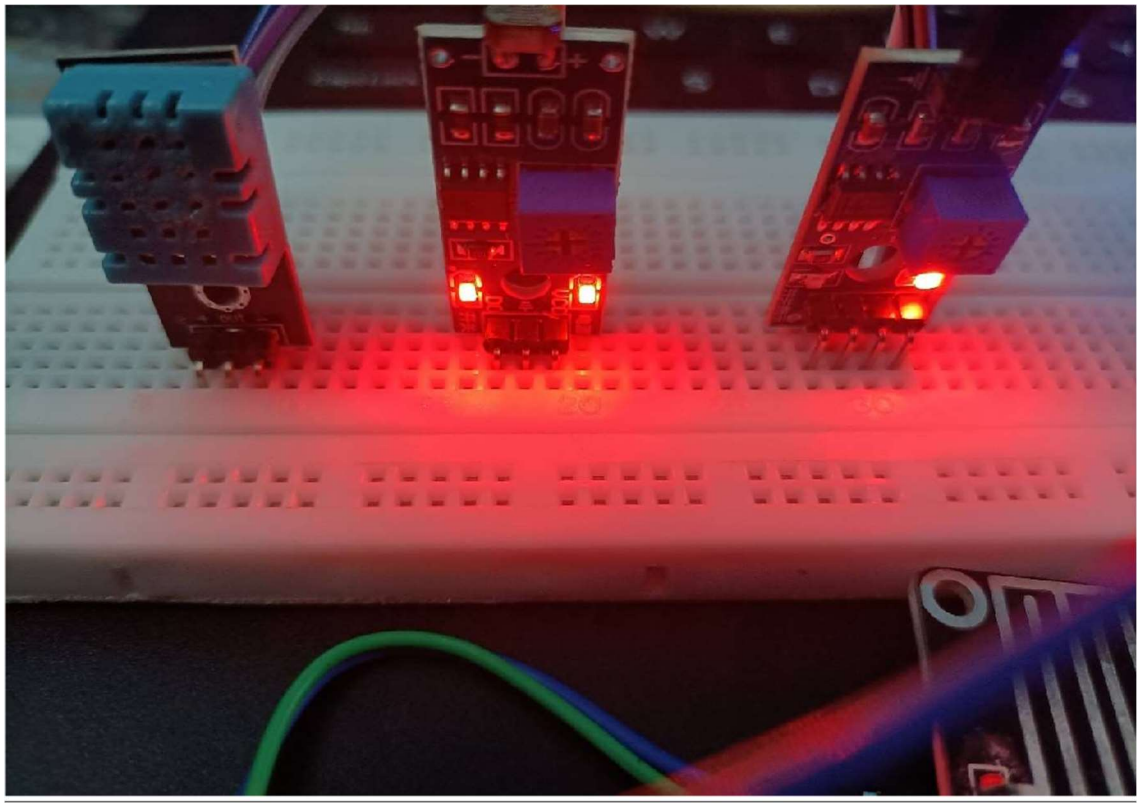
Over View OfProject

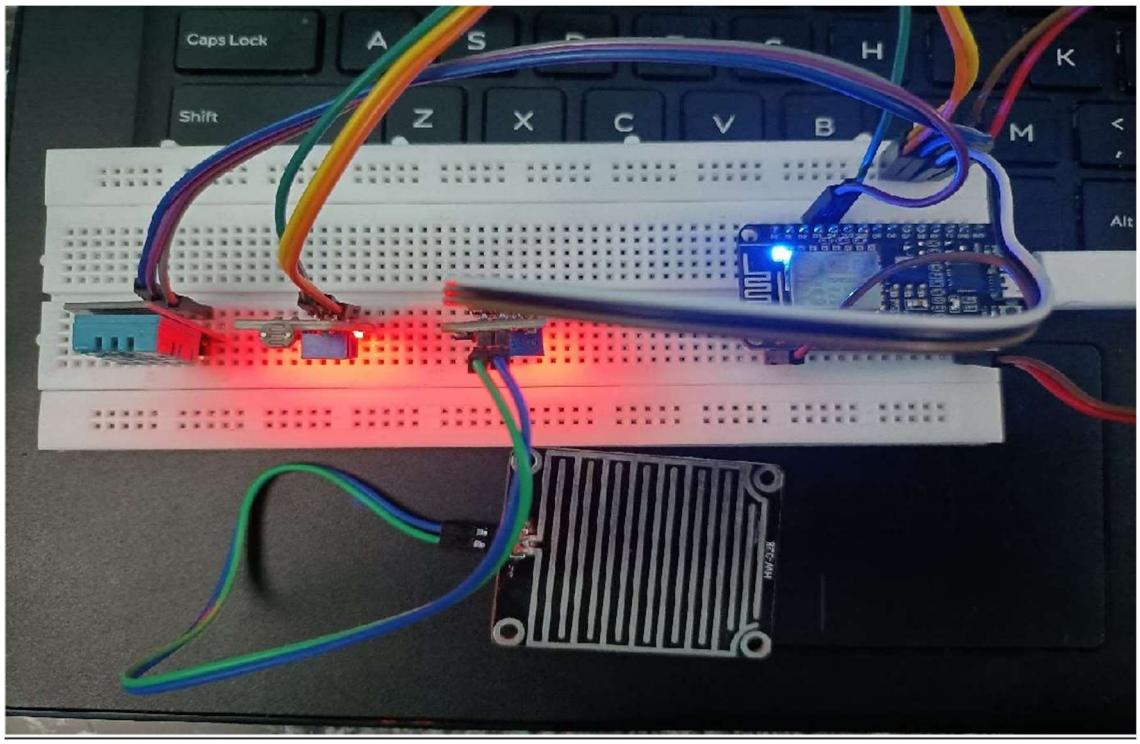
1. Blynk Dashboard





2. MODEL





APPLICATIONS:

a) The weather forecasting plays very important role in the field of agriculture.
b) It is also helpful at places like volcano and rain forests.
c) It is quite difficult for a human being to stay for longer time at such places.

ADVANTAGES:

1. IOT weather mentoring system project using Arduino Uno is fully automated.
2. It does not require any human attention.
3. We can get prior alert of weather conditions.
4. The low cost and efforts are less in this system.
5. Accuracy is high.
6. Smart way to monitor Environment.
7. Efficient.

Result:

Now the rain sensor starts to detect the amount of wetness in between 0 to 100.

Humidity sensor starts detect the temperature of the atmosphere between 0 to 100 degree(Celsius).

LDR starts detect the room light.

Weather Conditions can be monitored Using Blynk application.