IOT SMART WATERING PLANT MONITOR SYSTEM

A.LOHITH EE23BTECH11004
D.KANISHK EE23BTECH11029
K.JASWANTH EE23BTECH11033
Y.PREM SAGAR EE23BTECH11065

1 Introduction

NEED FOR SMART WATERING PLANT MONITOR SYSTEM :
→ Actually, it is difficult task to monitor a plant or a group of plants regularly . Due to the absence of monitoring or irregular monitoring, it may affect the growth of a plant.
→ Basically monitoring means checking the habitual external factors for a plant to grow like temperature around it, humidity in its atmosphere , moisture content in the soil etc.. and making sure that the values of these factors are in the correct range.
→ So, to make this difficult task into a easy task our team had made a SMART PLANT SYSTEM in which can one check the above discussed factors at a plantation place and can take measures if the values of these factors are not in the correct range.
→ we can directly see the values of the factors in our mobile phone while anywhere in the globe.
→ if we get any unmiserable values then we can simply turn the on button in our mobile phone and can water the plant by the motor water pump from anywhere on the globe without any hand help.But the plants should be present in a place where there is internet to take the commands from the mobile.
→ Actually this is a basic model mainly used in our garden plants .But we can develop it and also can make into a big system for large scale plantations.

So, let us see a process to make a simple basic smart watering plant monitor system and also lets find the internal structure of the system

2 PROCESS

AIM: To make a SMART WATERING PLANT MONITOR SYSTEM

APPARATUS REQUIRED : DHT11 Temperature and humidity sensor, Soil mois-

ture sensor , NODE MCU, 1-channel 5V relay module for arduino, submerisable mini water Pump, few male to female jumper wires, some normal wires, LED bulb, DC voltage source of 5-10V.

NOW LETS FIND THE WORKING OF EACH APPARATUS MENTIONED ABOVE

1. Submerisable mini water Pump: We have to keep this mini pump inside water .Basically there is a motor inside this pump and also a tube connected to the pump which is for the passage of water after switching on . This motor will rotate when we apply the voltage and pumps the water through the tube .This has speciafication volatge of 5-10V.

Figure 1: Submerisable mini water Pump

MICROCONTROLLER: → microcontroller is a compact integrated circuit (IC) designed to execute specific tasks within embedded systems.
→ Memory: Microcontrollers typically contain various types of memory to store program instructions, data, and configuration settings.
→ Input/Output (I/O) Ports: Microcontrollers have pins or ports that allow them to interact with the external environment. These ports can be configured as digital inputs or outputs, analog inputs, or specialized interfaces (e.g., serial communication ports, PWM outputs).
→ Communication Interfaces: Microcontrollers can communicate with other devices or systems using various communication protocols such as UART, SPI, I2C, USB, Ethernet, or WiFi.

2. DHT11 Temperature and humidity sensor : The DHT11 is a low-cost digital temperature and humidity sensor that operates using a digital signal protocol.
Here's how it generally works:
→Sensing Elements: The DHT11 sensor contains a humidity sensing component and a thermistor (temperature sensor). These components change their electrical resistance based on the temperature and humidity of the environment.
→ Signal Processing: Inside the DHT11, there's a small microcontroller that processes

the signals from the sensing elements and converts them into digital data.
→ Digital Output: The DHT11 provides digital output via a single data pin. It communicates with external devices, like microcontrollers (e.g., Arduino), using a simple serial protocol.
→ Data Transmission: When the DHT11 receives a signal from an external device requesting data, it starts the data transmission process. It sends a start signal followed by the actual data.
→ Data Encoding: The data transmitted by the DHT11 consists of a fixed-length data stream. This data stream includes information about temperature, humidity, and a checksum for error checking.
→ Communication Protocol: The DHT11 uses a proprietary communication protocol to transmit data. It sends bits of data one by one, with each bit represented by the duration of a high or low signal on the data pin.
→ Interpreting Data: The receiving device (e.g., Arduino) interprets the data stream received from the DHT11. It decodes the binary data into meaningful temperature and humidity readings.
Specification:
Temperature range: 0 to 50° C error of + 2° C.
Humidity: 20-90% RH + 5% RH
Interface: digital

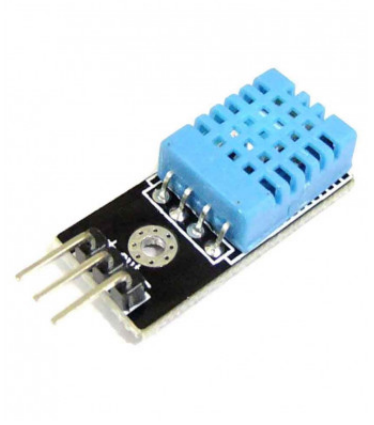

Figure 2: DHT11 Temperature and humidity sensor

3. Soil moisture sensor:Soil moisture sensors work by measuring the volumetric water content in the soil. There are several types of soil moisture sensors, but one com-

mon type is the capacitive soil moisture sensor. Here's how it generally works:
→ Capacitive Principle: Capacitive soil moisture sensors operate based on the principle of capacitance. Capacitance is the ability of a system to store an electrical charge. In this case, the soil acts as a dielectric between two electrodes, forming a capacitor.
→ Probe Construction: The sensor typically consists of two or more electrodes (probes) inserted into the soil. One of the electrodes acts as a reference, while the other(s) measure the capacitance between them, which is influenced by the soil moisture content.
→ Dielectric Constant of Water: The dielectric constant of water is much higher than that of air or soil minerals. When the soil is dry, there is less water content, and the dielectric constant is lower, resulting in lower capacitance. Conversely, when the soil is moist, the water content increases, leading to a higher dielectric constant and higher capacitance.
→ Measurement Circuitry: The sensor is connected to measurement circuitry, which typically includes an oscillator and a microcontroller. The oscillator generates an alternating electric field between the electrodes, and the microcontroller measures the capacitance.
→ Calibration: Before use, soil moisture sensors often require calibration to establish a relationship between the measured capacitance and the actual soil moisture content. This calibration is typically done by taking readings at different known moisture levels and creating a calibration curve.
→ Output: The output of the soil moisture sensor is usually an analog or digital signal proportional to the soil moisture content. This signal can be further processed or displayed by a microcontroller, Arduino, or other data logging devices.
→ Placement: Proper placement of the soil moisture sensor is essential for accurate measurements. The sensors should be inserted into the root zone of the plants and positioned at different depths to capture moisture variations throughout the soil profile.
Specifications: →Working voltage: 5V
→ Working current: <20 mA
→ Interface: Analog
→Working Temperature: 10°C-30°C

4. NODE MCU : → NodeMCU is a popular open-source firmware and development kit that helps you to prototype IoT (Internet of Things) devices quickly and easily. It's based on the ESP8266 WiFi module, which integrates a microcontroller unit (MCU) with built-in WiFi capabilities
→ Hardware: The core of NodeMCU is the ESP8266 microcontroller unit (MCU), which contains a powerful 32-bit processor along with integrated WiFi connectivity. This chip is capable of running code and communicating over WiFi networks.
→ Firmware: NodeMCU comes preloaded with Lua scripting language firmware. Lua is a lightweight scripting language well-suited for embedded systems. This firmware provides an easy-to-use interface for programming the ESP8266 without the need for complicated toolchains or programming languages.
→ Development Environment: To program a NodeMCU device, you typically use the Arduino IDE (Integrated Development Environment) or other compatible IDEs such as

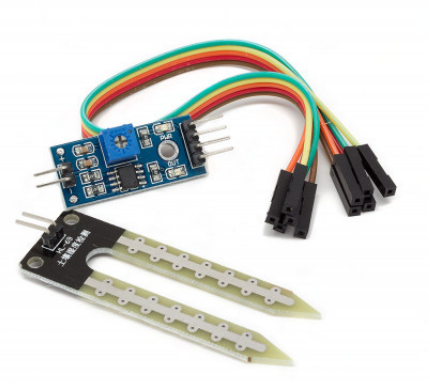

Figure 3: Soil moisture sensor

PlatformIO. You write your code in either Lua or C/C++, upload it to the NodeMCU board via USB, and then the board executes the code.
→ Connectivity: NodeMCU enables you to connect your devices to the internet via WiFi. It can act as a client, connecting to existing WiFi networks, or as a server, hosting its own network. This connectivity allows your NodeMCU-powered devices to communicate with other devices or servers over the internet.
→ GPIO Pins: The ESP8266 chip has several General Purpose Input/Output (GPIO) pins that you can use to interact with external components such as sensors, LEDs, motors, and more. These pins can be configured and controlled programmatically to read inputs or output signals.

Figure 4: NODE MCU

5. 1-channel 5V relay module for arduino : A 1-channel 5V relay module for Arduino is a simple electronic device that allows you to control high-voltage devices (such as lights, motors, or appliances) with a low-voltage microcontroller like the Arduino. Here's how it typically works:
→ Components: The relay module consists of several key components: A relay: This is an electro mechanical switch that can be controlled electronically. It has two states: normally open (NO) and normally closed (NC). When the relay is energized, it switches from one state to the other.
→ Driver circuitry: This circuitry is responsible for controlling the relay based on signals received from the Arduino. It typically includes a transistor (such as a bipolar junction transistor or MOSFET) to amplify the current from the Arduino's digital output pin to drive the relay coil.
→Protection diode: This diode is connected in parallel with the relay coil to protect the driver circuitry from voltage spikes that may occur when the relay is turned off.
→ Connection to Arduino: The relay module is connected to the Arduino using three connections:
→→ VCC (or JD-VCC): This connection is typically connected to the Arduino's 5V pin to provide power to the relay module.
→→ GND: This connection is connected to one of the Arduino's ground pins to complete the circuit.
→→ Signal input: This connection is connected to one of the Arduino's digital output pins (such as pin 2, 3, 4, etc.) to control the relay.
→ Control Logic: To control the relay, you program the Arduino to set the digital output pin connected to the relay module to either HIGH or LOW:
→→ When the digital output pin is set to HIGH, it energizes the relay coil, causing the relay switch to change state (either from normally open to normally closed, or vice versa), allowing current to flow through the load connected to the relay.
→→ When the digital output pin is set to LOW, the relay is de-energized, and the switch returns to its original state, cutting off the current flow to the load.
→ Load Connection: The load (such as a light bulb, motor, or other high-voltage device) is connected to the relay's common (COM) terminal and one of its normally open (NO) or normally closed (NC) terminals, depending on whether you want the load to be powered when the relay is activated or deactivated.

NOW WE WILL DISCUSS ABOUT THE PROCESS HAPPENING IN THE SYSTEM :
PROCEDURE:
HERE IS THE CIRCUIT DIAGRAM

CONNECTIONS TO BE MADE:

CONNECTIONS BETWEEN NODE MCU AND SOIL MOISTURE SENSOR

Figure 5: 1-channel 5V relay module for arduino

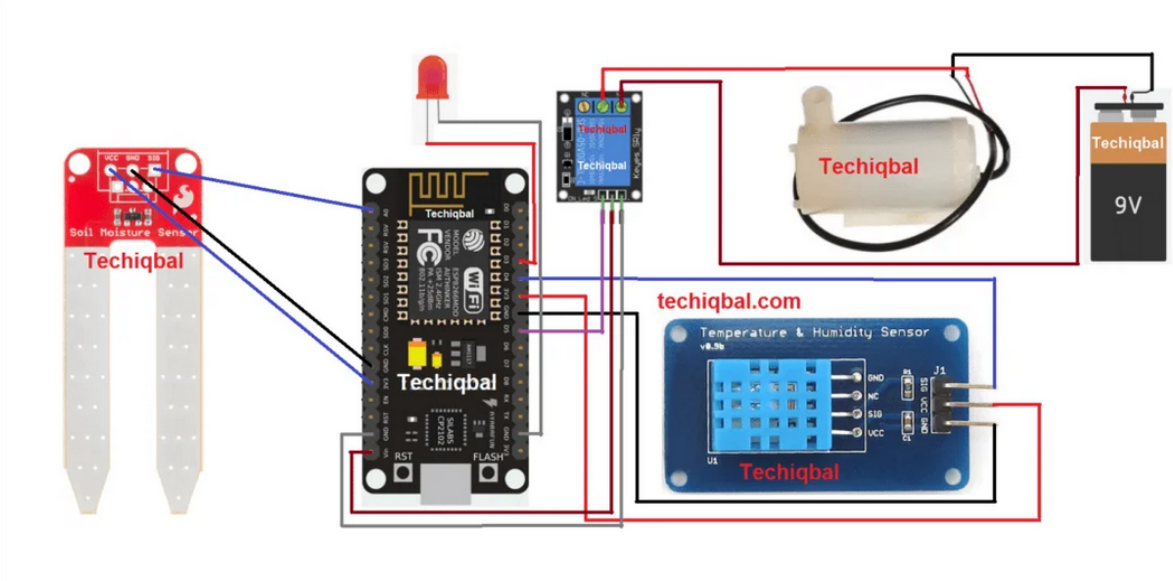

Figure 6: 1-channel 5V relay module for arduino

→ Connect the UCC,GND and SIG ports of moisture sensor to 3V3,GND and AO ports of the NODE MCU correspondingly (1 to 1) with the help of normal wires or jumper wires as shown in the above figure 6.

→ **FUNCTION**:
→ : The UCC (VCC) and GND ports of the moisture sensor are connected to the power

supply and ground of the NodeMCU, respectively. This provides the necessary power (usually 3.3V to 5V) to operate the moisture sensor.
→ The SIG (signal) port of the moisture sensor carries the analog voltage signal that corresponds to the soil moisture level. This signal is transmitted to the NodeMCU's analog input (AO) port for processing.
→ The AO (analog output) port of the NodeMCU is configured to receive the analog voltage signal from the moisture sensor. This allows the NodeMCU to read the voltage level, which represents the moisture level of the soil.
→ By connecting these ports in this configuration, the NodeMCU can read the analog output from the moisture sensor and process it using Arduino code to calculate the soil moisture content. The NodeMCU can then perform actions based on this moisture level.

CONNECTIONS BETWEEN NODE MCU AND TEMPERATURE AND HUMIDITY SENSOR:
→ Connect SIG,VCC and GND ports of the temperature and humidity sensor to D4,3V3 and GND ports of the NODE MCU correspondingly(1 to 1) as shown in the above figure 6 with the help of the wires.

→ **FUNCTION**:
→ The SIG port of the temperature and humidity sensor carries the digital or analog signal that corresponds to the temperature and humidity readings. This signal is transmitted to a digital input/output (D4) port of the NodeMCU for processing.
→ The VCC and GND ports of the temperature and humidity sensor are connected to the power supply and ground of the NodeMCU, respectively. This provides the necessary power (usually 3.3V to 5V) to operate the sensor.
→ By connecting the SIG port to a digital input/output (D4) port of the NodeMCU, the NodeMCU can read the digital or analog signal from the temperature and humidity sensor. This allows the NodeMCU to retrieve temperature and humidity readings from the sensor.
→ By integrating temperature and humidity readings alongside soil moisture data, the device can provide more comprehensive information about the environmental conditions affecting plant growth

CONNECTION BETWEEN LED BULB AND NODE MCU:
→ connect the positive and negative terminals of the bulb to the D3 and GND ports of the NODE MCU as shown in the above figure 6 with the help of the wires.

→**FUNCTION**
→ The LED bulb serves as an indicator or actuator in the system. By connecting it to a digital output (D3) port of the NodeMCU, the microcontroller can control the LED bulb. For example, it can turn the LED on or off based on certain conditions or thresholds determined by the Arduino code.

CONNECTIONS BETWEEN WATER PUMP AND NODE MCU AND RELAY MODULE

→ connect the negative terminal of the water pump to the negative terminal of the battery and connect the normally open and common contact ports of the relay module to the positive terminals of the water pump and battery correspondingly (1 to 1) as shown in the figure 6.

FUNCTION

→ By connecting the negative terminal of the water pump to the negative terminal of the battery, you complete the circuit for the pump, allowing it to operate when needed. This ensures that the pump has a reliable power source to function properly.

→ The relay module is used to control the power supply to the water pump. By connecting the normally open (NO) and common (COM) contact ports of the relay module to the positive terminals of the water pump and battery, respectively, you set up a switch that can be controlled by the Arduino code. When the relay is activated (closed), it allows current to flow from the battery to the water pump, activating the pump. When the relay is deactivated (open), it interrupts the current flow, turning off the pump.

the Arduino code can send a signal to the relay module to activate the water pump,

CONNECTIONS BETWEEN NODE MCU AND RELAY MODULE:

→ Connect the D5, Vin and GND of the NODE MCU to the Vcc , ground and input ports of the rely module correspondingly(1 to 1) with the help of jumper wires as shown in figure 6 .

FUNCTION

→ Control of Relay Module: The relay module acts as a switch to control the water pump or other external devices. By connecting the input port of the relay module to D5 of the NodeMCU, you allow the NodeMCU to control the activation and deactivation of the relay. When D5 is set to HIGH, it triggers the relay to close the switch, allowing current to flow from the Vin pin to the Vcc pin of the relay module. This, in turn, activates the relay and any connected device, such as a water pump.

→ Power Supply for Relay Module: The Vin pin of the NodeMCU provides a voltage supply that can be used to power external devices. By connecting Vin to Vcc of the relay module, you provide power to the relay module itself, ensuring that it can function properly.

→ Ground Connection: It's important to connect the ground (GND) of the NodeMCU to the ground (GND) of the relay module to ensure a common reference point for the electrical signals. This helps prevent electrical interference and ensures proper operation of the relay module. → Integration with Arduino Code: By controlling the relay module through the NodeMCU, you can integrate the activation of external devices (such as a water pump) into your Arduino code.

SO THIS IS ABOUT THE PARTS OF THE SYSTEM AND THE REASON BE-

HIND ITS SPECIFIED CONNECTIONS AND THE WORKING PROCESS

NOW LETS DISCUSS ABOUT THE AURDUINO CODE

```
    #define BLYNK_TEMPLATE_ID "TMPL3L31HI6GW"
#define BLYNK_TEMPLATE_NAME "plant water monitering system"
#define BLYNK_PRINT Serial

#include <OneWire.h>
#include <SPI.h>
#include <BlynkSimpleEsp8266.h>
#include <DHT.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS D2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
char auth[] ="tjdkwI67ELQFLYwd6kmOuzhXSFNGJoMn";
char ssid[] = "TP-Link_4EBE";
char pass[] = "54498280";
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
const int relayPin = D2;
SimpleTimer timer;
void loop(){
Blynk.run();
timer.run();
}
void sendSensor()
{
float h = dht.readHumidity();
float t = dht.readTemperature();

if (isnan(h) || isnan(t)) {
Serial.println("Failed to read from DHT sensor!");
return;
}
Blynk.virtualWrite(V5, h); //V5 is for Humidity
Blynk.virtualWrite(V6, t); //V6 is for Temperature
}
void setup()
{
Serial.begin(9600);
dht.begin();
timer.setInterval(1000L, sendSensor);
```

```
Blynk.begin(auth, ssid, pass);
sensors.begin();
pinMode(D3, OUTPUT);
pinMode(relayPin, OUTPUT);
}

BLYNK_WRITE(V0) // Executes when the value of virtual pin 0 changes
{
  if(param.asInt() == 1)
  {
    // execute this code if the switch widget is now ON
    digitalWrite(D3,HIGH);  // Set digital pin 2 HIGH
  }
  else
  {
    // execute this code if the switch widget is now OFF
    digitalWrite(D3,LOW);  // Set digital pin 2 LOW
  }
}
BLYNK_WRITE(V3) // Executes when the value of virtual pin 4 changes
{
  if(param.asInt() == 1)
  {
    // execute this code if the switch widget is now ON
    digitalWrite(relayPin, HIGH);  // Set relay pin HIGH to turn on the relay
  }
  else
  {
    // execute this code if the switch widget is now OFF
    digitalWrite(relayPin, LOW);  // Set relay pin LOW to turn off the relay
  }
}

int sensor=0;
int output=0;
void sendTemps()
{
sensor=analogRead(A0);
output=(145-map(sensor,0,1023,0,100)); //in place 145 there is 100(it change with the change
delay(1000);
sensors.requestTemperatures();
float temp = sensors.getTempCByIndex(0);
Serial.println(temp);
Serial.print("moisture = ");
Serial.print(output);
Serial.println("%");
```

```
Blynk.virtualWrite(V1, temp);
Blynk.virtualWrite(V2,output);
delay(1000);
}
```

CODE ANALYSING

→The first 3 lines of the code are define identifiers for your Blynk project template. BLYNK_TEMPLATE_ID and BLYNK_TEMPLATE_NAME are set to specific values that identify your project in the Blynk platform. BLYNK_PRINT is set to Serial to enable debugging messages to be printed to the Serial monitor.

#include <OneWire.h>
#include <SPI.h> $ >
#include <BlynkSimpleEsp8266.h>
#include <DHT.h>
#include <DallasTemperature.h>

→ These lines in the code include necessary libraries for the project:
→ OneWire: this Library for communication with devices using the OneWire protocol.
→SPI: this Library is for SPI communication.
→BlynkSimpleEsp8266: This Library is for interfacing the Arduino with the Blynk server using ESP8266 WiFi module.
→DHT: This Library is for interfacing with DHT temperature and humidity sensors.
→DallasTemperature: this Library is for interfacing with Dallas Temperature sensors

→ #define ONE_WIRE_BUS D2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
→ These lines define the pin (D2) to which the OneWire bus is connected and create instances of OneWire and DallasTemperature objects to communicate with temperature sensors connected to the OneWire bus.

char auth[] = ”tjdkwI67ELQFLYwd6kmOuzhXSFNGJoMn”;
char ssid[] = ”IITH-Guest-PWD-IITH@2023”;
char pass[] = ”IITH@2023”;

→These lines define authentication credentials (auth) for connecting to the Blynk server and WiFi network credentials (ssid and pass) for connecting the Arduino to the local WiFi network.

#define DHTPIN 2
#define DHTTYPE DHT11
DHT dh (DHTPIN, DHTTYPE);

→These lines define the pin (D2) to which the DHT11 temperature and humidity sensor is connected and create an instance of the DHT object to interact with the sensor.

→ const int relayPin = D2;

→Pin D2 is defined as the pin to which the relay is connected.

→SimpleTimer timer;

→ An instance of the SimpleTimer object named timer is created. This object will be used to schedule tasks at specific intervals.

→BlynkTimer timer;
→This line creates a BlynkTimer object named timer that will be used to schedule tasks.

```
void sendSensor ({)
{float h = dht.readHumidity();
float t = dht.readTemperature();
if (isnan(h) —— isnan(t))
Serial.println(”Failed to read from DHT sensor!”);
return;}
Blynk.virtualWrite(V5, h); //V5 is for Humidity
Blynk.virtualWrite(V6, t); //V6 is for Temperature
```

→The function sendSensor() reads the temperature and humidity values from the DHT sensor and sends them to the Blynk server using virtual pins V5 and V6.

```
void setup()
{Serial.begin(9600);
dht.begin();
timer.setInterval(1000L, sendSensor);
Blynk.begin(auth, ssid, pass);
sensors.begin();
pinMode(D3, OUTPUT);
pinMode(relayPin, OUTPUT); }
```

→ The setup() function initializes Serial communication, starts the DHT sensor, sets

the timer interval for sending sensor data to 1000 milliseconds (1 second), connects to the Blynk server using the provided authentication and WiFi credentials, and initializes the Dallas Temperature sensor.

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

The loop() function continuously runs the Blynk library's main function (Blynk.run()) and executes any scheduled tasks managed by the timer object (timer.run()).

```
BLYNK_WRITE(V0)
{if(param.asInt() == 1)
{digitalWrite(D3,HIGH);
}else
{digitalWrite(D3,LOW);
}}
```

→This function is executed whenever the value of virtual pin V0 changes. If the value is 1, it sets pin D3 high; otherwise, it sets it low.

```
BLYNK_WRITE(V3)
{if(param.asInt() == 1)
{digitalWrite(relayPin, HIGH);
}else
{digitalWrite(relayPin, LOW);
}
}
```

→Similar to the previous function, this one is executed when the value of virtual pin V3 changes. If the value is 1, it sets the relay pin high; otherwise, it sets it low.

```
int sensor=0;
int output=0;
void sendTemps()
{sensor=analogRead(A0);
output=(145-map(sensor,0,1023,0,100)); //in place 145 there is 100(it change with the change in sensor)
delay(1000);
sensors.requestTemperatures();
float temp = sensors.getTempCByIndex(0);
Serial.println(temp);
Serial.print(”moisture = ”);
```

```
Serial.print(output);
Serial.println("
Blynk.virtualWrite(V1, temp);
Blynk.virtualWrite(V2,output);
delay(1000);
}
```

This function sendTemps() reads the analog value from a moisture sensor connected to pin A0, maps it to a percentage value, reads the temperature from the Dallas Temperature sensor, prints the temperature and moisture level to the Serial monitor, and sends them to the Blynk server using virtual pins V1 and V2.

BLINK APP PURPOSE:
The purpose of the Blynk app is to provide a user-friendly interface for controlling and monitoring Internet of Things (IoT) devices remotely. It allows users to create custom dashboards and control panels to interact with their connected hardware devices, such as Arduino or ESP8266 microcontrollers, Raspberry Pi, and various sensors and actuators

THINGS TO BE DONE IN BLINK APP:
→ First install the blink app from play store and login your emailID
→ Click on create a new project and give a name to new project.
→ Then select NODE MCU in the choose device option. Then click wifi in the connection type option.
→ Then a token would be send to the registered gmail. We had to upload that token in the arduino code for uploading into NODE MCU.
→ Then click on plus button, then click on guage option for 3 times as we need 3 types of readings.Also add 2 buttons .
→ give names to the 3 guages. We had given temperature, soil moisture and humidity as 3 names .
→ for soil moisture select range as 100 and select the virtual pin V2
→ for temperature select range as 50 and select virtual pin V6
→ for humidity select range as 100 and select virtual pin V5.
→ for one button give name as light and select digital pin D5 and make push to switch button
→ foe another button give name as water pump ans select digital pin D5. Make the push button to switch .
→ When we upload the arduino program code into the NODE MCU then this will automatically connected to the plant system.

UPLOADING ARDUINO CODE INTO THE NODE MCU:
→ The NodeMCU has a built-in USB-to-Serial converter, allowing it to communicate with a laptop or Arduino via a USB cable

→On the laptop or Arduino side, we can also open a serial port with the same baud rate to communicate with the NodeMCU.
→ In the Arduino code running on the NodeMCU, you typically initialize Serial communication using Serial.begin(baud_rate). This sets up the serial port with a specific baud rate.
→ We can send data from the laptop or Arduino to the NodeMCU by writing to the serial port. For example, using Serial.write() or Serial.println() functions in Arduino IDE.
→ In this way the arduino receives data from the external source and implements in way according to the code
→ Once the NodeMCU receives the data, we can interpret it based on the communication protocol and application requirements.
→ After processing the received data, you may need to provide feedback or send a response back to the laptop or Arduino to acknowledge the received data or provide status updates.
→ But in this project we had used Blink app for getting the output from the arduino .This can be done by writing data back to the serial port using functions like Serial.write() or Serial.println().

WORKING OF THE PROJECT:
→ First place the soil sensor in the soil near the roor of the plant for better monitoring.
→ place the temperature and moist sensor beside the plant
→ now turn on the battery of 9V or DC voltage of 9V that was connected to the circuit
→ Now the sensors will record the data and sends to the NODE MCU
→ As the arduino code was written in such a way that these output values will be displayed in the blink app.
→ we can turn on the light button in the blink app for checking weather the code is correct or not or is there any error in the circuit if the light does not glow.
→ If the water level is low we can easily on the motor button in the blink app so the motor will be on automatically without any hand help.

ADVANTAGES:
→We can monitor the plants external living conditions like soil moisture, humidity, temaparture .
→ If these are not in the miserable level then we can take measures .
→ we can water the plants directly from our phone without any external hand help.
→ we can operate it anywhere from the globe. But the plants should be placed in a place where there is availability of internet

PRACTICAL APPLICATIONS:
We can also make advancements in this project and can make it useful for large scales like AGRICULTURE, NURSERIES,HOME GARDEN ETC... for monitoring the plants.

OBSERVATION:
→ Our all team members observed that when we turn on the battery then we are getting the reading of temperature, humidity and moisture content of the plant in the blink app.
→ when we are turning on the light button , the LED light was blowing which shows that our project was working.
→ when we are turning on the motor pump button , the motor was switching on and the water was pumping out from the pump. Which is our final task.

CONCLUSION:
By knowing about the each and every component and by analysing the above mentioned steps one can made a smart watering plant monitor system.