Self Watering System

- Automation for effortless hydration

INTRODUCTION :

In a world where technology is transforming every aspect of our lives, the realm of agriculture and plant care is no exception. The IoT-Based Self-Watering System Project represents a groundbreaking innovation in the field of horticulture and agriculture. This project introduces a smart, automated solution that seamlessly integrates the Internet of Things (IoT) with plant care, revolutionizing the way we nurture and sustain greenery.

The core objective of this project is to develop an intelligent, interconnected system that can effectively manage and optimize the watering needs of plants. By leveraging IoT technology, the project offers a dynamic and data-driven approach to plant hydration. It allows real-time monitoring, precise control, and remote management of watering systems, ensuring that plants receive the right amount of water at the right time.

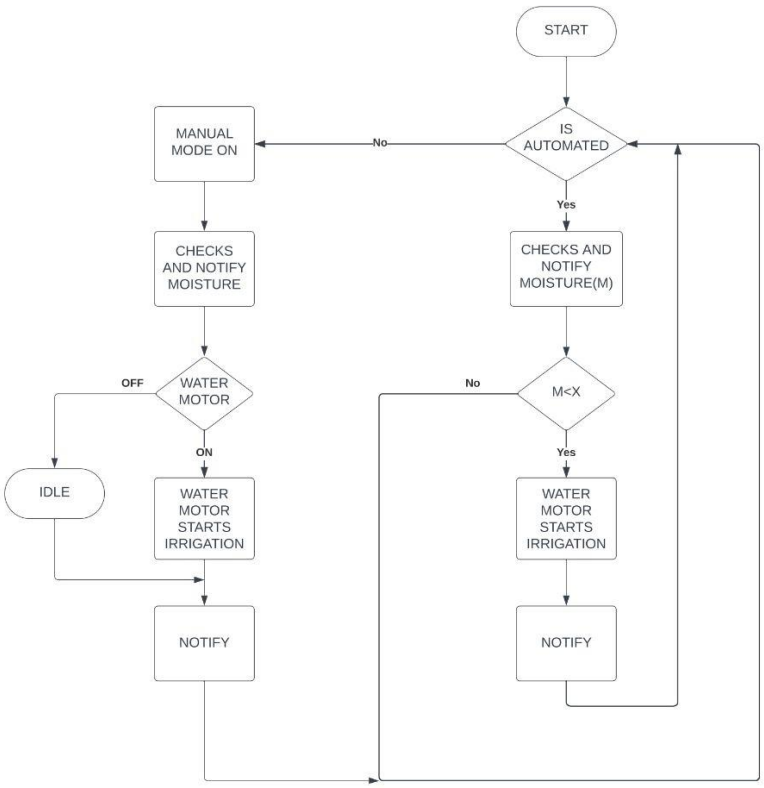
Key components of the project include sensor nodes that gather environmental data, a central IoT hub that processes this data, and automated actuators responsible for dispensing water as needed. Additionally, the project often involves a user-friendly interface, typically in the form of a mobile app or web portal, that enables users to monitor and adjust the watering schedule from anywhere in the world.

The benefits of such a system are multifaceted. It enhances plant health and growth by preventing overwatering or underwatering, conserves water resources by eliminating waste, reduces the manual labor required for plant care, and offers greater convenience and peace of mind to gardeners and farmers.

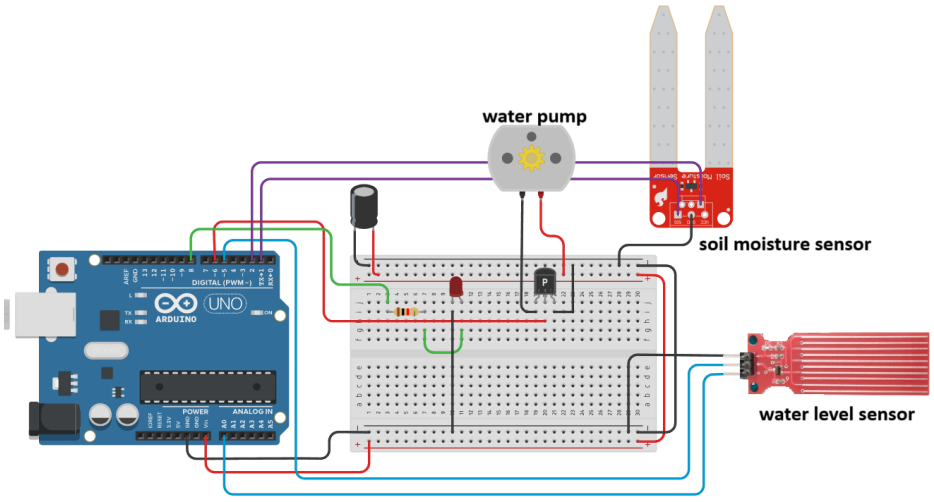
This IoT-based self-watering system project holds immense promise not only for individuals looking to maintain vibrant gardens but also for the agriculture industry aiming to increase crop yield and efficiency. Its potential to contribute to sustainable agriculture and urban green spaces makes it a significant innovation in the era of smart and connected solutions. This project showcases the power of technology to transform traditional practices and promote efficient, data-driven, and environmentally conscious plant care.

Self Watering System

FLOW CHART :



Circuit Diagram:



Self Watering System

Code:

```
#define BLYNK_PRINT Serial
#define BLYNK_TEMPLATE_ID "TMPL3-txE0Pzz"
#define BLYNK_TEMPLATE_NAME "Self Watering System"
#define BLYNK_AUTH_TOKEN "YOUR_AUTH_TOKEN"

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>


char auth[] = "YOUR_AUTH_TOKEN";
char ssid[] = "YOUR_SSID";
char pass[] = "YOUR_PASS";

#define LED_PIN     D8

// Sensor pins
#define WATER_PWR D5
#define WATER_PIN A0
#define MOISTURE_PWR D2
#define MOISTURE_PIN D1
#define WATER_PUMP D6

/* Define how long the pump will run (3 seconds) */
#define PUMP_TIME 3000

/* Change these values based on your calibration values */
#define LVL_HIGH 520  // Define max value water level
#define LVL_LOW 100   // Define min value of water level

int sw;

BLYNK_WRITE(V4){
   int pinValue = param.asInt();
   if (pinValue == 1){
     sw=HIGH;
     int WATER_LEVEL = readWaterLevel();
     float percentLevel = map(WATER_LEVEL, 0, 520, 0, 100);
```

```
    Serial.print("Water Level: ");
    Serial.print(percentLevel);
    Serial.println(" %");
    if (WATER_LEVEL >= LVL_LOW) {
      Serial.println("Water level is perfect");
      int MOISTURE_LEVEL = readMoisture();
      if (MOISTURE_LEVEL == LOW) {
        Serial.println("Moisture is perfect");
        Blynk.virtualWrite(V1, "Moisture is perfect");
        Blynk.virtualWrite(V3, "The water pump is on standby!");
      }
      else {
        Serial.println("Low moisture! Time to water!");
        Blynk.virtualWrite(V1, "Low moisture! Time to water!");
        Blynk.virtualWrite(V3, "Water pump started!");
        digitalWrite(LED_PIN,HIGH);
        digitalWrite(WATER_PUMP, HIGH);
        Serial.print("Water pump started!");
        delay(PUMP_TIME);
        digitalWrite(WATER_PUMP, LOW);
        digitalWrite(LED_PIN,LOW);
        Serial.println("The water pump is on standby!");
      }
    }
     else {
      Serial.println("Water level is low! Time to add water!");
    }
    Blynk.virtualWrite(V2, percentLevel);
    Serial.println();
  }
  else if (pinValue == 0) {
    sw=LOW;
    Serial.println("System is in Manual mode!!");
   // do something when button is released;
  }
  Serial.print("V4 button value is: "); // printing value to serial monitor
  Serial.println(pinValue);
}
```

Self Watering System

```
BLYNK_WRITE(V5){
      int pinValue = param.asInt(); // assigning incoming value from pin V1 to a variable

      if (pinValue == 1 && sw == LOW){
         Blynk.virtualWrite(V3, "Water pump started!");
         digitalWrite(LED_PIN,HIGH);
         digitalWrite(WATER_PUMP, HIGH);
         Serial.println("Water pump started!");
      }
       else if (pinValue == 0) {

         digitalWrite(WATER_PUMP, LOW);
         digitalWrite(LED_PIN,LOW);
         Serial.println("The water pump is on standby!");
         Blynk.virtualWrite(V3, "Water pump is off!");
         Serial.println("Water pump is off!");
      }
     Serial.print("V5 button value is: "); // printing value to serial monitor
      Serial.println(pinValue);
}

/* Define the time range for sensor measurements */
const int measurementInterval = 10000;

/* Time variable to keep track of the time range */
unsigned long previousMillis = 0;

void setup() {
  Serial.begin(115200);
  sw = HIGH;
  Blynk.begin(auth,ssid,pass);
  pinMode(WATER_PWR, OUTPUT);
  pinMode(WATER_PIN, INPUT);
  pinMode(MOISTURE_PWR, OUTPUT);
  pinMode(MOISTURE_PIN, INPUT);
  pinMode(WATER_PUMP, OUTPUT);
  pinMode(LED_PIN, OUTPUT);
  /* Initially keep the sensors and motor OFF */
  digitalWrite(LED_PIN,HIGH);
```

```
  delay(3000);
  digitalWrite(LED_PIN,LOW);
  digitalWrite(WATER_PWR, LOW);
  digitalWrite(MOISTURE_PWR, LOW);
  digitalWrite(WATER_PUMP, LOW);
}

void loop() {
  // Run the Blynk app.
  Blynk.run();
  // Run the LED function
  /* Get current time. If the defined time range
  has not passed, terminate the loop */
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis < measurementInterval) {
    return;
  }
  previousMillis = currentMillis;
  if(sw==HIGH)
    without_wifi();
}

void without_wifi(){
  int WATER_LEVEL = readWaterLevel();
  float percentLevel = map(WATER_LEVEL, 0, 520, 0, 100);
  Serial.print("Water Level: ");
  Serial.print(percentLevel);
  Serial.println(" %");
  if (WATER_LEVEL >= LVL_LOW) {
    Serial.println("Water level is perfect");
    int MOISTURE_LEVEL = readMoisture();
    if (MOISTURE_LEVEL == LOW) {
      Serial.println("Moisture is perfect");
      //Blynk.virtualWrite(V1, "Moisture is perfect");
      //Blynk.virtualWrite(V3, "The water pump is on standby!");
    }
    else {
      Serial.println("Low moisture! Time to water!");
      Blynk.virtualWrite(V1, "Low moisture! Time to water!");
      Blynk.virtualWrite(V3, "Water pump started!");
```

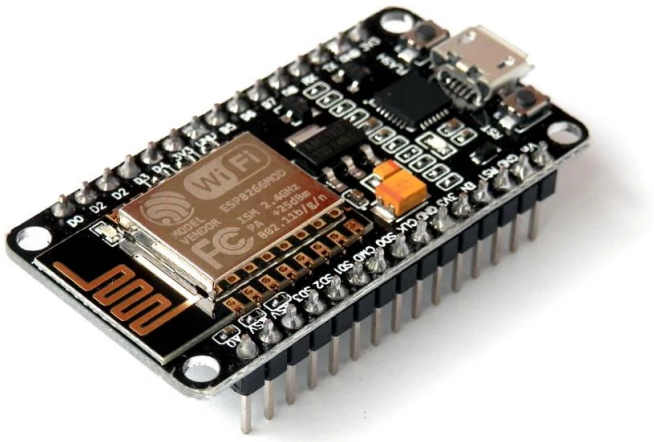
Self Watering System

```
      digitalWrite(LED_PIN,HIGH);
      digitalWrite(WATER_PUMP, HIGH);
      Serial.print("Water pump started!");
      delay(PUMP_TIME);
      digitalWrite(WATER_PUMP, LOW);
      digitalWrite(LED_PIN,LOW);
      Serial.print("The water pump is on standby!");
    }
  }
  else {
    Serial.println("Water level is low! Time to add water!");
  }
  Blynk.virtualWrite(V2, percentLevel);
  Serial.println();
}

/* This function returns the analog water level measurement */
int readWaterLevel(){
  digitalWrite(WATER_PWR, HIGH);           // Turn the sensor ON
  delay(10);                               // Allow power to settle
  int sensorValue = analogRead(WATER_PIN);  // Read the analog value from sensor
  digitalWrite(WATER_PWR, LOW);            // Turn the sensor OFF
  return sensorValue;                      // Return analog water value
}

/* This function returns the digital soil moisture measurement */
int readMoisture(){
  digitalWrite(MOISTURE_PWR, HIGH);            // Turn the sensor ON
  delay(10);                                   // Allow power to settle
  int sensorValue = digitalRead(MOISTURE_PIN);  // Read the digital value from sensor
  digitalWrite(MOISTURE_PWR, LOW);             // Turn the sensor OFF
  return sensorValue;                          // Return digital moisture value
}
```

Self Watering System

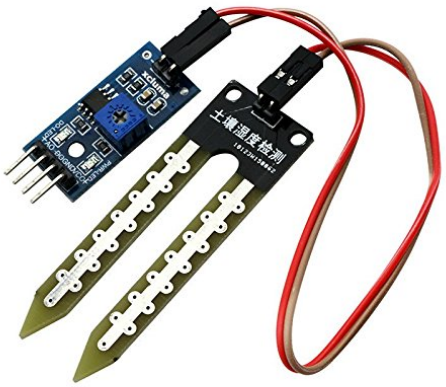
Sensors and Components:

NODEMCU ESP8266:



The NodeMCU ESP8266 is an affordable and popular microcontroller board with built-in WiFi capabilities. It is commonly used for Internet of Things (IoT) projects, home automation, and remote control applications. The board can be programmed using the Arduino IDE or Lua scripting, making it accessible to a wide range of developers. It features multiple GPIO pins for connecting sensors and peripherals and has a supportive community. Overall, it's a versatile and cost-effective choice for wireless IoT projects.

Moisture Sensor :



Self Watering System

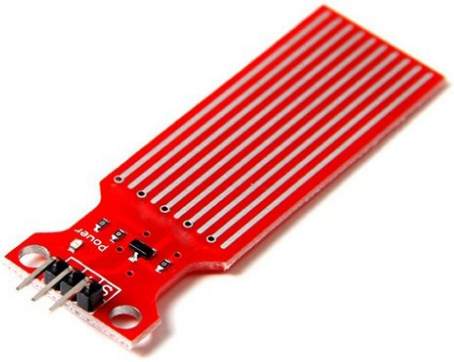
A moisture sensor is a device that measures the moisture content of soil. It typically works on the principle of electrical resistance or capacitance:

Resistive Moisture Sensor: Measures the soil's electrical resistance, which decreases as soil moisture increases. It provides analog or digital readings based on resistance.

Capacitive Moisture Sensor: Measures the soil's dielectric constant, which changes with moisture levels. It provides digital or analog outputs.

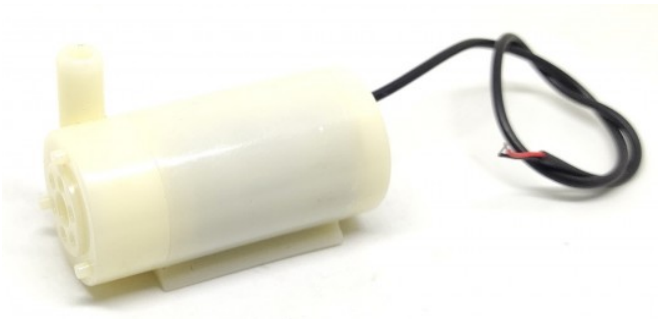
The sensor is inserted into the soil, and changes in electrical properties are used to determine the soil's moisture content. These sensors are commonly used in agriculture, gardening, and environmental monitoring to optimize irrigation and ensure proper plant hydration.

Water level Sensor :



This water level sensor module has a series of parallel exposed traces to measure droplets/water volume in order to determine the water level. Very Easy to monitor water level as the output to analog signal is directly proportional to the water level. This output analog values can be directly read via ADC and can also be connected directly to Arduino's analog input pins.

Water Pump :

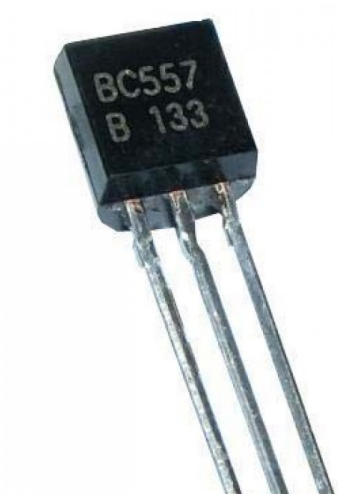


Self Watering System

A DC water pump motor, typically powered by a voltage range of 3.3V to 5V, is a compact and efficient device used to circulate or transfer liquid. It operates on the principle of converting electrical energy into mechanical energy to move water or other liquids. When powered, the motor's rotor turns, creating a centrifugal or diaphragm action that pushes the liquid through an outlet. The flow rate and pressure generated depend on the motor's specifications.

These small water pump motors are commonly used in applications like cooling systems for electronic devices, miniature fountains, and hydroponic systems. They are valued for their low voltage requirements, compact size, and ease of integration into various projects. The pump's performance characteristics, such as flow rate and head pressure, are crucial considerations when selecting the right pump for a specific application.

PNP Transistor :



A PNP transistor is a type of bipolar junction transistor (BJT) commonly used in electronic circuits. It operates as a current-controlled switch. The "PNP" stands for "Positive-Negative-Positive," referring to the arrangement of its semiconductor layers.

Working:

Emitter-Base Junction: When a small current (known as the base current) flows from the base terminal to the emitter terminal, it causes the PNP transistor to conduct. The emitter-base junction is forward-biased, allowing current to flow.

Collector-Base Junction: Simultaneously, the collector-base junction remains reverse-biased, preventing the flow of current from collector to base. This region is

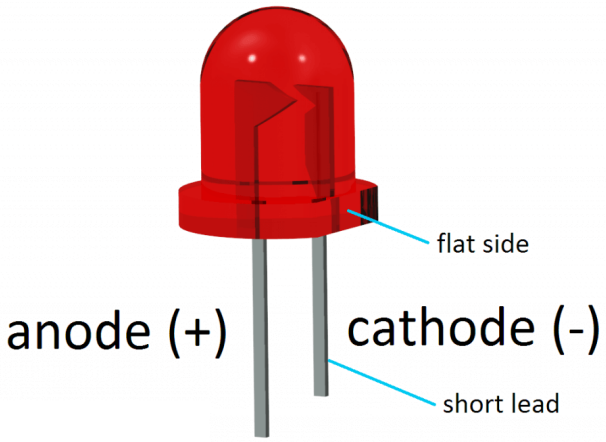
responsible for the transistor's ability to control a larger current (known as the collector current) between the collector and emitter terminals.

In summary, by controlling a small current at the base terminal, a PNP transistor can switch a larger current between its collector and emitter terminals. It can be used in various electronic applications, including amplification and switching.

Resistor :

A resistor is an electronic component that limits the flow of electrical current. It works by obeying Ohm's Law, where current (I) is directly proportional to voltage (V) and inversely proportional to resistance (R), as I = V/R. Resistors are used to control current, divide voltage, and provide specific resistance values in electronic circuits.

LED :



Self Watering System

An LED, or Light-Emitting Diode, is a semiconductor device that emits light when an electric current flows through it. The working of an LED is based on the principle of electroluminescence, where the movement of electrons within the semiconductor material produces photons of light. When a voltage is applied to the LED, electrons and holes recombine in the semiconductor, releasing energy in the form of visible light. LEDs are commonly used for illumination, displays, indicators, and many other lighting applications due to their efficiency, durability, and the ability to emit light of various colors.

Capacitor :

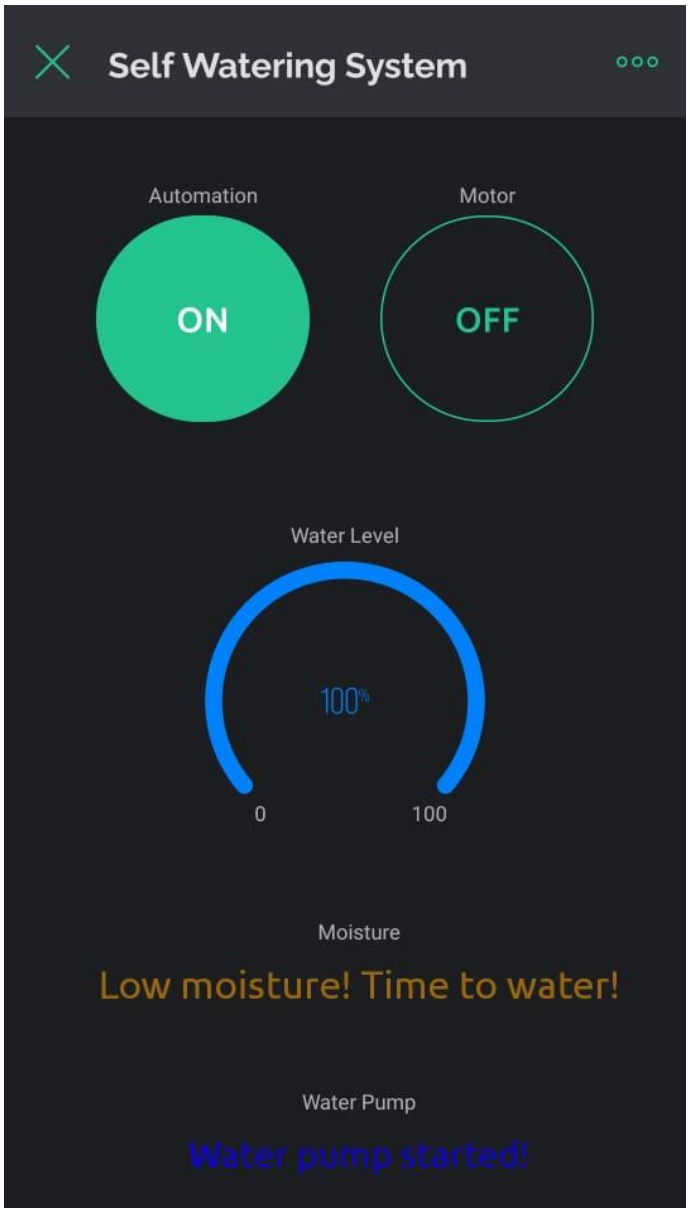
A capacitor is an electronic component that stores and releases electrical energy. It consists of two conductive plates separated by an insulating material (dielectric). When voltage is applied across the plates, they store electric charge. Capacitors are used in electronics for various purposes, including energy storage, filtering, and timing. They can quickly discharge their stored energy when needed and are commonly found in circuits to stabilize voltage, filter noise, and perform various timing functions.

Self Watering System

Blynk :

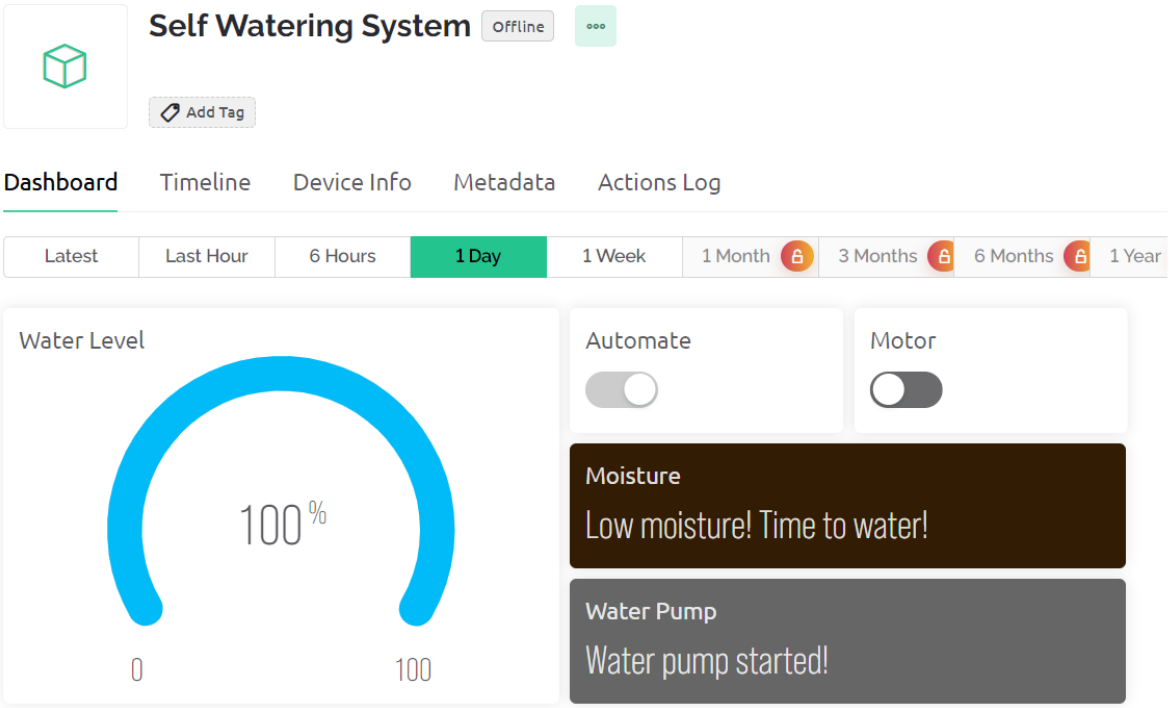
Blynk is a mobile app and IoT (Internet of Things) platform that allows users to create custom applications and control hardware devices using their smartphones. It provides a user-friendly interface for connecting and controlling a wide range of IoT devices, making it easier for individuals and developers to build IoT projects without extensive programming knowledge. Blynk supports a variety of hardware platforms and is widely used for home automation, remote monitoring, and other IoT applications.

Mobile Dashboard :



Self Watering System

Computer Dashboard :



Summary :

A self-watering system IoT project is a smart and innovative solution that uses Internet of Things (IoT) technology to automate and optimize the watering of plants. This project typically involves sensors to monitor soil moisture levels and actuators to control the water supply. The key components include moisture sensors, a microcontroller (e.g., Arduino or Raspberry Pi), a water pump or valve, and an internet-connected module (e.g., Wi-Fi or GSM).

The project works by continuously monitoring the soil moisture level using the sensors. When the moisture level falls below a set threshold, the microcontroller triggers the water supply to irrigate the plants. The system can be remotely monitored and controlled through a mobile app or a web interface, allowing users to adjust watering schedules and receive real-time data on soil conditions. This automation ensures that plants receive the right amount of water, reducing the risk of overwatering or underwatering.

Overall, a self-watering system IoT project offers the advantages of efficient plant care, water conservation, and the convenience of remote monitoring and control, making it a valuable tool for both home gardeners and commercial agriculture.