IoT Enabled Smart Irrigation

PERIPHERATHON 1.0

Team ID: U9VRB871

Team Name: HackEye

Ishan Jawale
B.Tech. F.Y. Computer Engg. K.K.W.I.E.E.R., Nashik

Mihir Nandi
B.Tech. F.Y. Computer Engg. K.K.W.I.E.E.R., Nashik

Nikhil Patil
B.Tech. F.Y. Computer Engg. K.K.W.I.E.E.R., Nashik

Tanmay Tambat
B.Tech. F.Y. Computer Engg. K.K.W.I.E.E.R., Nashik

❖ Introduction:

- This document provides an overview of an intelligent irrigation system that channels water flow to specific areas of land using IoT technology. The system maintains a constant moisture level in the soil, and enables the user to control it wirelessly from any part of the world using their cell phone. The system includes multiple outlets that operate independently.

❖ System Design:

The intelligent irrigation system is designed to automate the irrigation process and reduce water waste. The system consists of the following components:

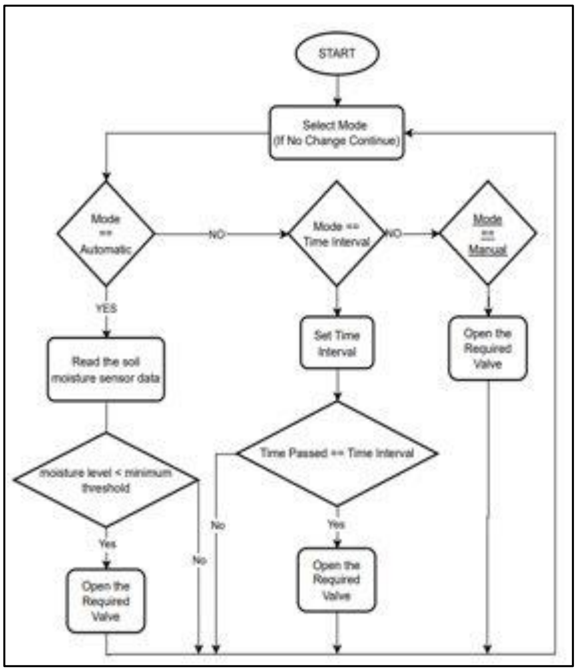
- Moisture sensors: The system includes moisture sensors that measure the moisture level in the soil.
- Water flow sensors: The system includes water flow sensors that measure the amount of water flowing through the irrigation system.
- Microcontroller: The system uses a microcontroller to process sensor data and control the irrigation system.
- Water valves: The system includes water valves that control the flow of water to specific areas of land.
- Wireless communication: The system uses wireless communication to enable the user to control the system from their cell phone.
- Power supply: The system is powered by a reliable power supply, such as a battery or solar panel.

❖ System Operation:

The system operates as follows:

- Moisture sensors measure the moisture level in the soil.
- The microcontroller processes the sensor data and determines the amount of water required.
- Water valves control the flow of water to specific areas of land.
- Water flow sensors measure the amount of water flowing through the system.
- The microcontroller adjusts the water flow rate based on the measured moisture level and the required amount of water.
- The user can control the system wirelessly from their cell phone, enabling them to adjust the irrigation schedule as required.

- Flowchart:

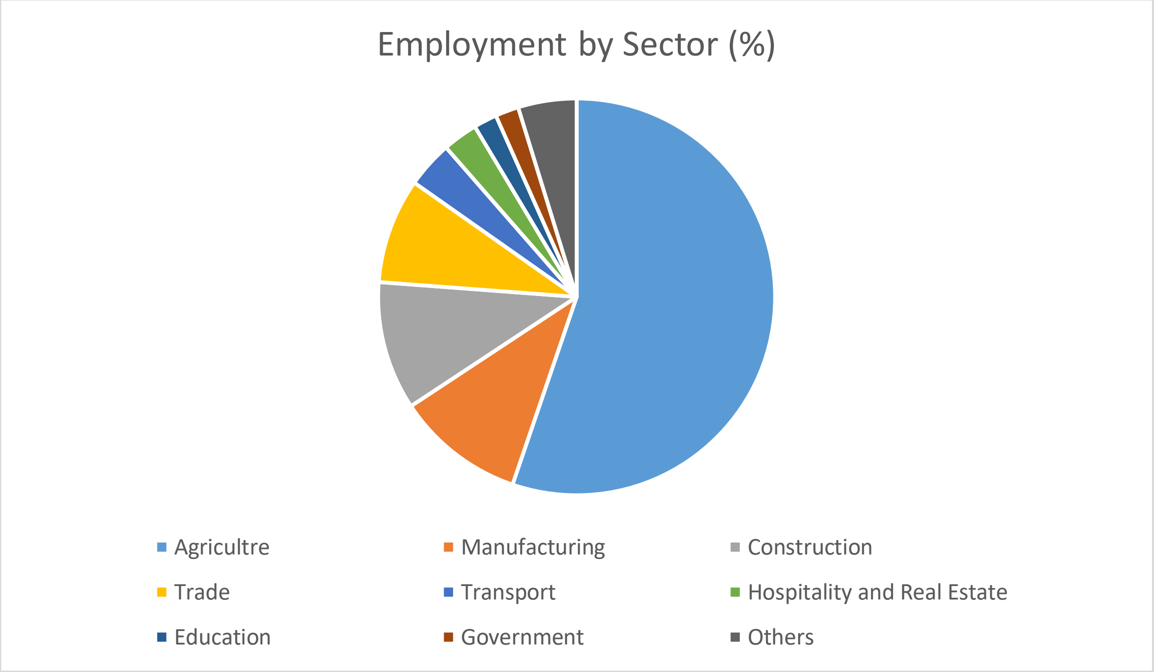


- System Features:

The intelligent irrigation system includes the following features:

- Automated irrigation: The system automates the irrigation process, reducing water waste and ensuring a constant moisture level in the soil.
- Wireless communication: The user can control the system wirelessly from their cell phone, enabling them to adjust the irrigation schedule from any part of the world.
- Multiple outlets: The system includes multiple outlets that operate independently, enabling the user to irrigate different areas of land with different water requirements.
- Energy-efficient: The system is powered by a reliable power supply, such as a battery or solar panel, making it energy-efficient and cost-effective.

- Impact on Society:

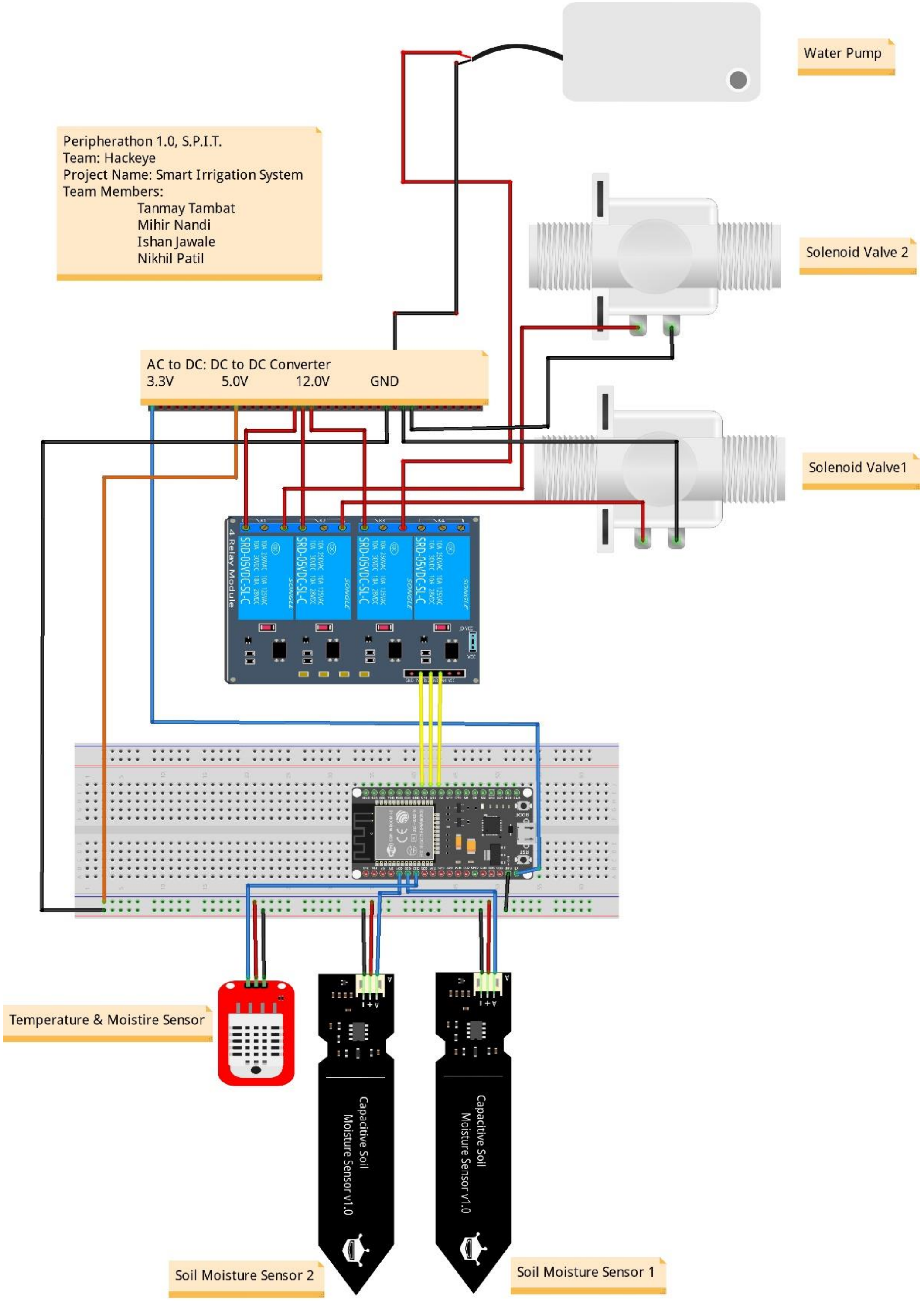


- India is known to be an agricultural country where a significant portion of the population is employed in the agricultural sector. As per the 2011 census, around 58% of India's population is dependent on agriculture and allied activities for their livelihood. Agriculture is the backbone of India's economy, accounting for around 17% of the country's GDP in 2020.
- However, despite the significant role of agriculture in India's economy, the sector is facing several challenges such as water scarcity, inefficient irrigation techniques, and soil degradation. These challenges lead to low crop yields, water wastage, and reduced farm income.
- An intelligent irrigation system, as described in the problem statement, can address some of these challenges by improving the efficiency of irrigation, reducing water waste, and maintaining a constant moisture level in the soil. With wireless communication capabilities, farmers can monitor and control the system from any part of the world, making it easier to manage their crops and improve their yields. The inclusion of multiple outlets that operate independently further enhances the system's capabilities, allowing farmers to irrigate different areas of land with different water requirements.
- In conclusion, the development of an intelligent irrigation system that channels water flow to specific areas of land using IoT technology can be a significant step towards addressing some of the challenges faced by the agricultural sector in India. With the majority of the population dependent on agriculture for their livelihoods, the success of such a system could have a far-reaching impact on the country's economy and the lives of its people.

❖ Conclusion:

- The intelligent irrigation system is an effective solution for automating the irrigation process and reducing water waste. The system maintains a constant moisture level in the soil and enables the user to control it wirelessly from any part of the world using their cell phone. The system includes multiple outlets that operate independently, making it suitable for irrigating different areas of land with different water requirements. The system is energy-efficient and cost-effective, making it a practical solution for smart agriculture.

❖ Hardware Model (Circuit Diagram)



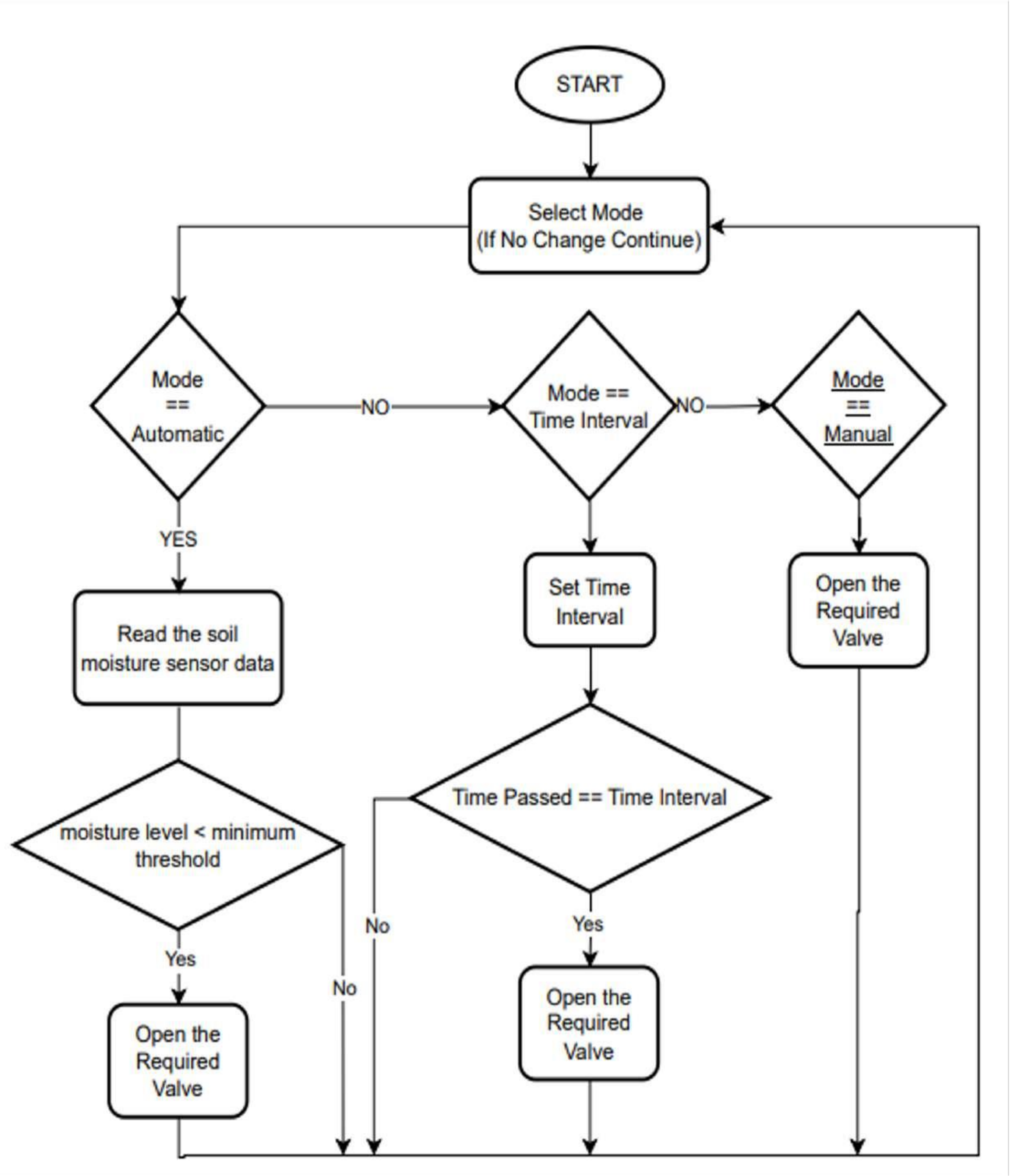
Peripherathon 1.0, S.P.I.T.
Team: Hackeye
Project Name: Smart Irrigation System
Team Members:
Tanmay Tambat
Mihir Nandi
Ishan Jawale
Nikhil Patil

❖ Software Model:

• Algorithm:

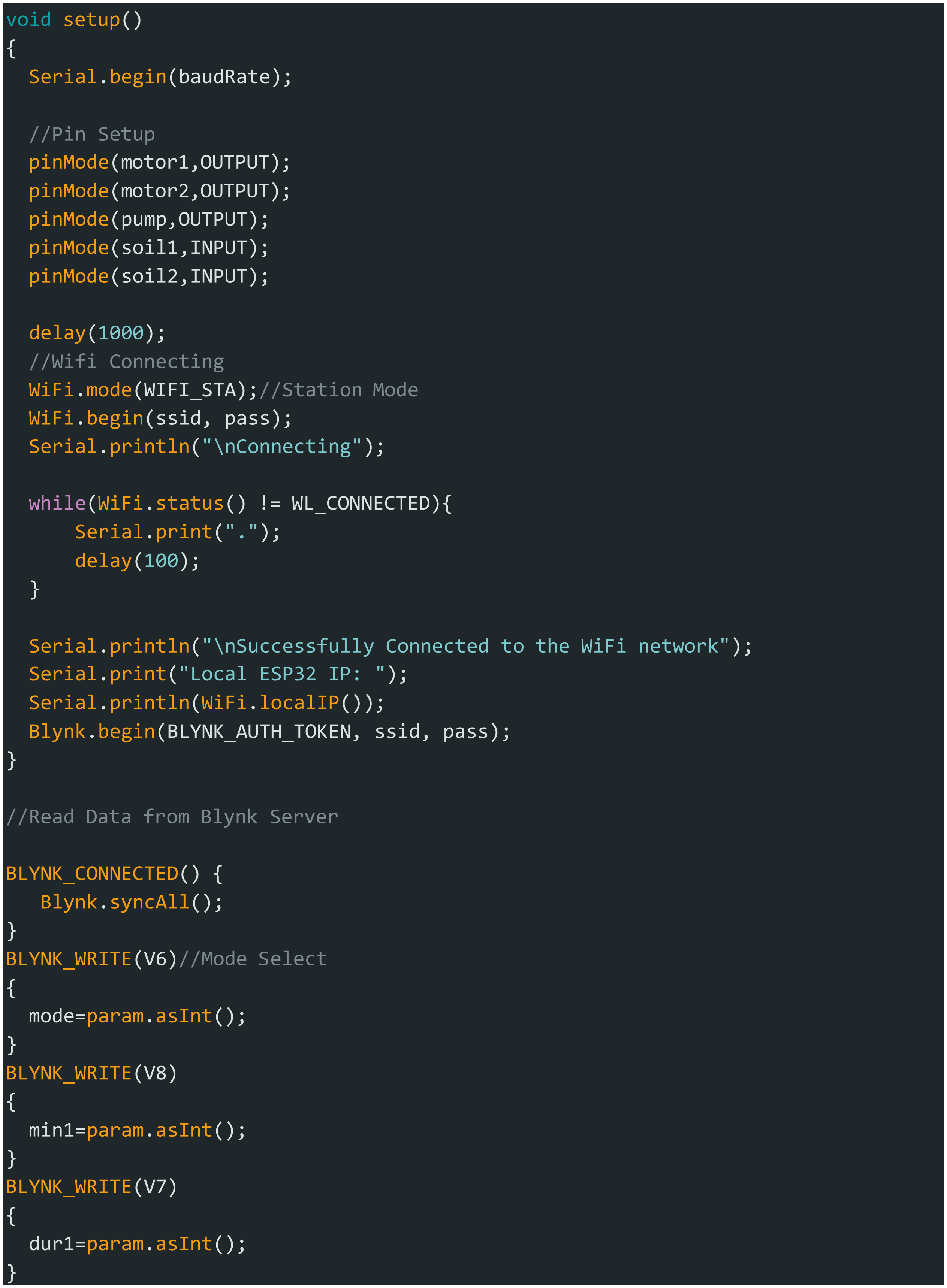
```
STEP 1: START

STEP 2: Select mode

STEP 3: If Mode == Automatic

        1) Read the soil moisture
        2) If moisture level < minimum threshold
           Then,
           Open the required valve
        3) If moisture level == minimum threshold
           Then,
            Close the valve

STEP 4: If Mode == Time Interval

        1) Set the time interval
        2) If Time passed != Time Interval
           Then,
           Open the required value
        3) If Time passed == Time Interval
           Then,
           Close the valve

STEP 5: Mode == Manual

        1) Set the choice: Yes or No
        2) If choice == Yes
           Then,
           Open the valve
        3) If choice == No
           Then,
           Close the valve

STEP 6: END
```

- **Flowchart:**

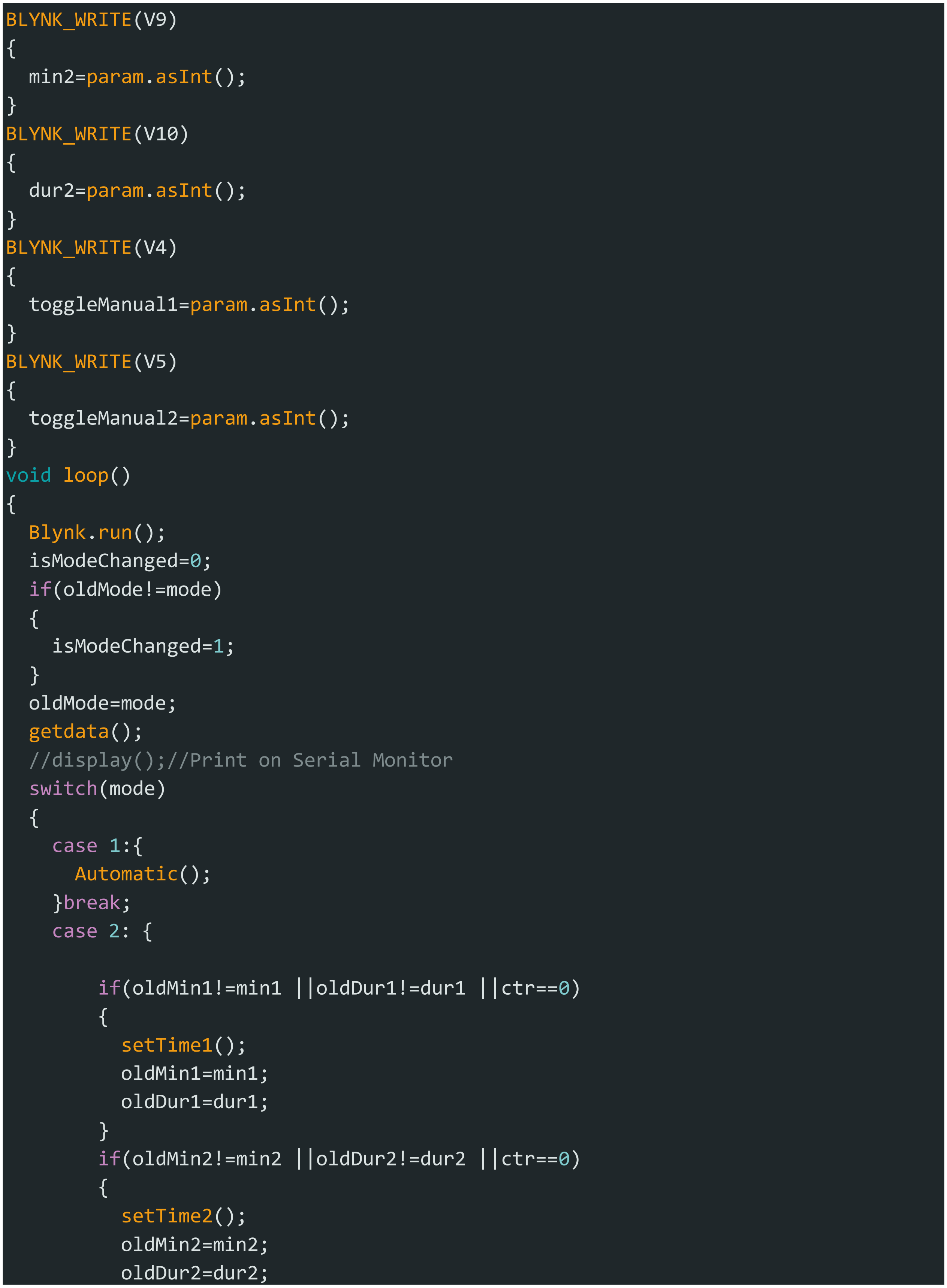

START
Select Mode
(If No Change Continue)
Mode
==
Automatic
NO
Mode ==
Time Interval
NO
Mode
==
Manual
YES
Read the soil
moisture sensor data
Set Time
Interval
Open the
Required
Valve
Time Passed == Time Interval
moisture level < minimum
threshold
No
Yes
Yes
No
Open the
Required
Valve
Open the
Required
Valve

- **Code:**

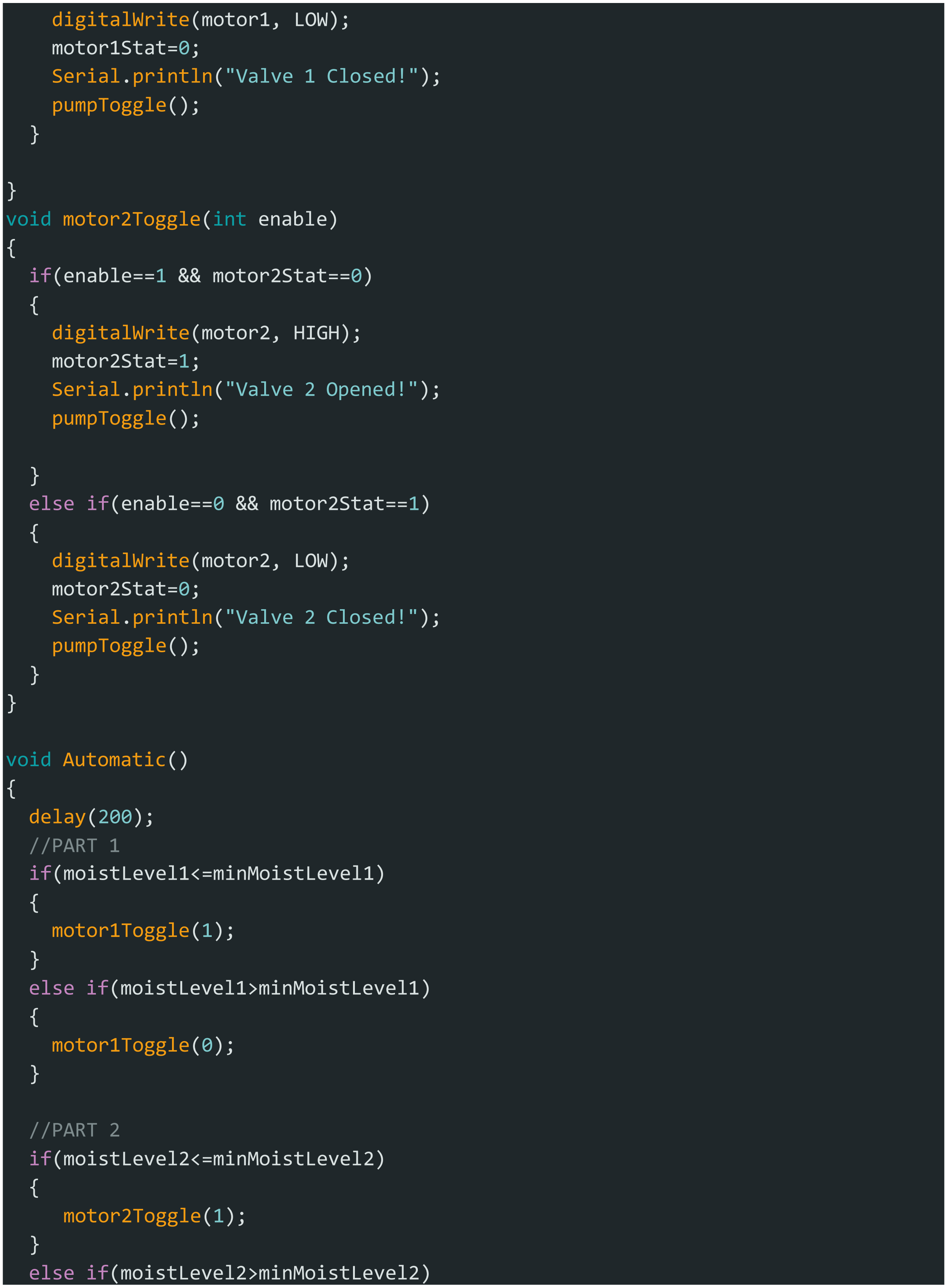
```
#define BLYNK_TEMPLATE_ID "TMPL3w5r-qi64"
#define BLYNK_TEMPLATE_NAME "Smart Irrigation System"
#define BLYNK_AUTH_TOKEN "Nk5hd2K4Xk35ZZDWGnPX-fal016MIwhb"
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
char* ssid = "Mihir";//Wi-fi Name
char* pass = "12345678";//Wi-fi Password

/*Blynk Virtual Pins:
  Mode Selection: V6
  Soil Moisture 1: V1
  Soil Moisture 2: V2
  Temperature:  V3
  Hours (Section 1): V8
  Duration (Section 2): V7
  Hours (Section 2): V9
  Duration (Section 2): V10
  Valve 1 (Manual): V4
  Valve 2 (Manual): V5
*/

//Pins Allotment
int motor1=18,motor2=5;
int pump=19;
int soil1=34;
int soil2=35;

//Default
int motor1Stat=0, motor2Stat=0;
int pumpStat=0;
int baudRate=115200;
int mode;
int minMoistLevel1=500;
int minMoistLevel2=500;
int min1=0,dur1=0,oldMin1=0,oldDur1=0; //For Time Interval of Motor 1
int min2=0,dur2=0,oldMin2=0,oldDur2=0; //For Time Interval of Motor 2
unsigned long timeWait1=0, timeIrrigate1=0, totalWait1=0, totalIrrigate1=0;
unsigned long timeWait2=0, timeIrrigate2=0, totalWait2=0, totalIrrigate2=0;
int ctr=0;

//Data Variables
int moistLevel1,moistLevel2;
int oldMode,isModeChanged;
int toggleManual1=0, toggleManual2=0;
```



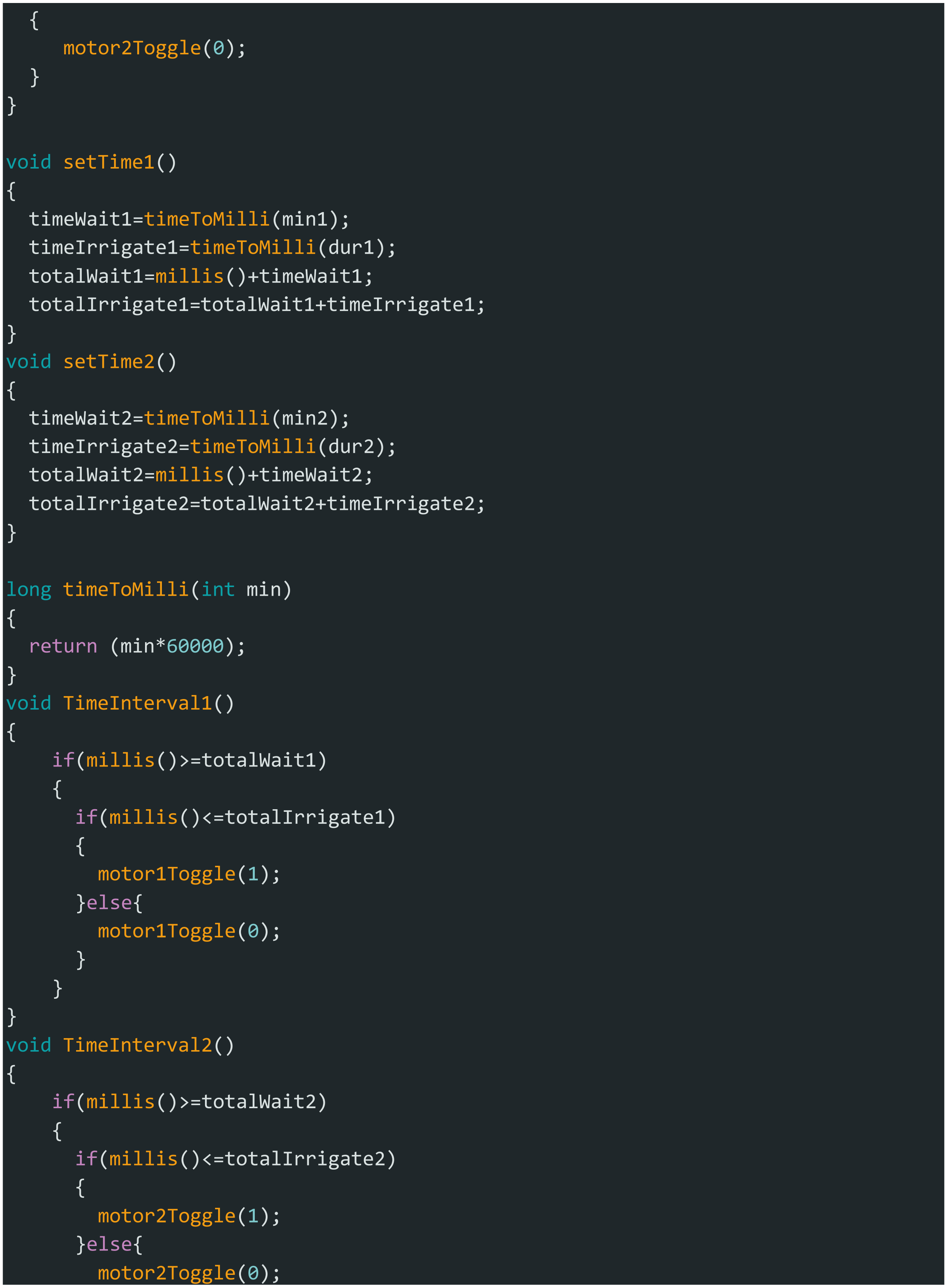
```
void setup()
{
  Serial.begin(baudRate);

  //Pin Setup
  pinMode(motor1,OUTPUT);
  pinMode(motor2,OUTPUT);
  pinMode(pump,OUTPUT);
  pinMode(soil1,INPUT);
  pinMode(soil2,INPUT);

  delay(1000);
  //Wifi Connecting
  WiFi.mode(WIFI_STA);//Station Mode
  WiFi.begin(ssid, pass);
  Serial.println("\nConnecting");

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

  Serial.println("\nSuccessfully Connected to the WiFi network");
  Serial.print("Local ESP32 IP: ");
  Serial.println(WiFi.localIP());
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
}

//Read Data from Blynk Server

BLYNK_CONNECTED() {
   Blynk.syncAll();
}
BLYNK_WRITE(V6)//Mode Select
{
  mode=param.asInt();
}
BLYNK_WRITE(V8)
{
  min1=param.asInt();
}
BLYNK_WRITE(V7)
{
  dur1=param.asInt();
}
```



```
BLYNK_WRITE(V9)
{
  min2=param.asInt();
}
BLYNK_WRITE(V10)
{
  dur2=param.asInt();
}
BLYNK_WRITE(V4)
{
  toggleManual1=param.asInt();
}
BLYNK_WRITE(V5)
{
  toggleManual2=param.asInt();
}
void loop()
{
  Blynk.run();
  isModeChanged=0;
  if(oldMode!=mode)
  {
    isModeChanged=1;
  }
  oldMode=mode;
  getdata();
  //display();//Print on Serial Monitor
  switch(mode)
  {
    case 1:{
      Automatic();
    }break;
    case 2: {

        if(oldMin1!=min1 ||oldDur1!=dur1 ||ctr==0)
        {
          setTime1();
          oldMin1=min1;
          oldDur1=dur1;
        }
        if(oldMin2!=min2 ||oldDur2!=dur2 ||ctr==0)
        {
          setTime2();
          oldMin2=min2;
          oldDur2=dur2;
        }

      TimeInterval1();
      TimeInterval2();
    }break;
    case 3: {
      Manual();
    }break;
  }
  ctr=1;
  delay(300);
}


void getdata()
{
  moistLevel1=(analogRead(soil1)*100/1023);
  Blynk.virtualWrite(V1,moistLevel1);
  moistLevel2=(analogRead(soil2)*100)/1023;
  Blynk.virtualWrite(V2,moistLevel2);
}

void pumpToggle(){
  if(motor1Stat==1 || motor2Stat==1)
  {
    digitalWrite(pump, HIGH);
    Serial.println("Pump Started!");
  }
  else{
    digitalWrite(pump,LOW);
    Serial.println("Pump Closed!");
  }
}

void motor1Toggle(int enable)
{
  if(enable==1 && motor1Stat==0)
  {
    digitalWrite(motor1, HIGH);
    motor1Stat=1;
    Serial.println("Valve 1 Opened!");
    pumpToggle();
  }
  else if(enable==0 && motor1Stat==1)
  {
```



```
    digitalWrite(motor1, LOW);
    motor1Stat=0;
    Serial.println("Valve 1 Closed!");
    pumpToggle();
  }

}
void motor2Toggle(int enable)
{
  if(enable==1 && motor2Stat==0)
  {
    digitalWrite(motor2, HIGH);
    motor2Stat=1;
    Serial.println("Valve 2 Opened!");
    pumpToggle();

  }
  else if(enable==0 && motor2Stat==1)
  {
    digitalWrite(motor2, LOW);
    motor2Stat=0;
    Serial.println("Valve 2 Closed!");
    pumpToggle();
  }
}

void Automatic()
{
  delay(200);
  //PART 1
  if(moistLevel1<=minMoistLevel1)
  {
    motor1Toggle(1);
  }
  else if(moistLevel1>minMoistLevel1)
  {
    motor1Toggle(0);
  }

  //PART 2
  if(moistLevel2<=minMoistLevel2)
  {
     motor2Toggle(1);
  }
  else if(moistLevel2>minMoistLevel2)
```



```
  {
     motor2Toggle(0);
  }
}

void setTime1()
{
  timeWait1=timeToMilli(min1);
  timeIrrigate1=timeToMilli(dur1);
  totalWait1=millis()+timeWait1;
  totalIrrigate1=totalWait1+timeIrrigate1;
}
void setTime2()
{
  timeWait2=timeToMilli(min2);
  timeIrrigate2=timeToMilli(dur2);
  totalWait2=millis()+timeWait2;
  totalIrrigate2=totalWait2+timeIrrigate2;
}

long timeToMilli(int min)
{
  return (min*60000);
}
void TimeInterval1()
{
    if(millis()>=totalWait1)
    {
      if(millis()<=totalIrrigate1)
      {
        motor1Toggle(1);
      }else{
        motor1Toggle(0);
      }
    }
}
void TimeInterval2()
{
    if(millis()>=totalWait2)
    {
      if(millis()<=totalIrrigate2)
      {
        motor2Toggle(1);
      }else{
        motor2Toggle(0);
      }
    }
}
void Manual(){
    motor1Toggle(toggleManual1);
    motor2Toggle(toggleManual2);
}
void display()
{
  Serial.println("Moisture Levels:");
  Serial.println(moistLevel1);
  Serial.println(moistLevel2);
}
void reset()
{
  motor1Toggle(0);
  motor2Toggle(0);
}
```