Smart Irrigation System

A Project Report on Automated Irrigation Using IoT

2024.08.05

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

In most agricultural areas, rational use of water in irrigation is very important for the preservation of water and for the successful production of crops. Conventional irrigation practices are known to inefficient in that they rarely give real-time information on the status of the soil and thus over irrigation or under irrigation may occur. Smart irrigation systems, which have come with IoT, presents a solution through the use of sensors and automation in the usage of water.

The system that is incorporated in this project features a technique known as surface irrigation, this is widely acknowledged as one of the oldest known techniques for irrigation. This method spreads water all over the soil surface depending on gravity impact to soil particles. It is still relatively straightforward, but there is fairly high risk of waste that can potentially be addressed by implementing valuable control systems based on IoT devices.

1.2 Objectives

Thus, smart irrigation systems are used very widely these days in the processes of modern agriculture, especially when speaking about the Cabbage production – it is one of the most water-consuming crops. Cabbage needs water at the right amount and the right time but applying massive water it develop root diseases and loss of nutrients and too little causes stress on the plant and thus low yields. A smart irrigation system focusses mostly on the efficient use of water by incorporating technology in the system which works according to the real time data that is input and which makes sure that crops need the right amount of water at the right time. Conserving water resources; Optimising crop production and quality.

A system that allows the use of computer to control the supply of water depending on the moisture found on the soil. From the same Thingsboard Cloud, the system offers notifications to its users whenever the level of the water in the tank is low and has control switches on a dashboard. Thus, this approach guarantees the purely automatic control combined with the opportunity for the user supervision, developing the strategies of water usage in agricultural systems.

2 Project Overview

2.1 System Description in Sri Lanka

2.1.1 Crop Selection

FIGURE 1 : CABBAGE PLANTATIOIN

The selected crop used is the crop cabbage. Cabbage is a vegetable widely used in Sri Lankan cultivation. It is highly taken in culinary due to its availability in a varied climatic condition and more nutritional values. It is grown well in Central highlands and other suitable areas in Sri Lanka. Detailed information about water, soil conditions, temperature, pest management and harvesting are discussed here.

- Climatic Requirements

Cabbage grows well in cool areas. The central highlands is good for cabbage production, preferably Nuwara-Eliya, Badulla, and Kandy districts, with an appropriate mean temperature of 15 °C and 20 °C. A mean temperature above 25 °C may result in inconsistent heading and soft heads, yielding low-quality heads. Suitable districts for the production of cabbages are the types with moderate temperatures. They should coincide with the cooler planting months.

- Soil Requirements

Cabbage prefers well-drained, fertile soils with a pH range of 6.0 to 6.8. The soil should be rich in organic matter, which can be enhanced through the incorporation of compost or well-rotted manure. Loamy soils are ideal for cabbage, providing adequate drainage and moisture retention. Proper soil preparation, including plowing and harrowing, is crucial to create a fine tilth for transplanting seedlings.

- Water Requirements

Water management is a critical aspect of cabbage cultivation. Cabbage requires consistent moisture throughout its growing period, with an average water requirement of 25-30 mm per week. This can be supplied through rainfall or irrigation. During the dry season, supplemental irrigation is necessary, with a preference for drip or sprinkler systems to ensure even water distribution. Overwatering should be avoided as it can lead to root diseases and reduced plant vigor.

- Fertilization

Cabbage is a heavy feeder and requires balanced fertilization for optimal growth. The application of nitrogen (N), phosphorus (P), and potassium (K) is essential. A typical recommendation is to apply 150-200 kg/ha of nitrogen, 50-60 kg/ha of phosphorus, and 150-200 kg/ha of potassium. In addition, micronutrients like boron and calcium are crucial for preventing common disorders like hollow stem and tip burn.

- Pest and Disease Management

Common pests affecting cabbage in Sri Lanka include aphids, caterpillars, and cabbage loopers. Integrated Pest Management (IPM) practices are recommended, combining biological control, cultural practices, and the judicious use of chemical pesticides. Diseases such as black rot, clubroot, and downy mildew can be managed through crop rotation, resistant varieties, and proper field sanitation.

- Harvesting

Cabbage is typically ready for harvest 75-90 days after transplanting, depending on the variety and growing conditions. Harvesting should be done when the heads are firm and reach the desired size. The heads should be cut at the base, leaving a few outer leaves to protect them during transport. Proper post-harvest handling, including cooling and storage, is essential to maintain the quality and shelf life of cabbage.

- Conclusion

Here, the project basically about the requirement of the soil.

Soil requirement

- Deep, well drained loamy soil rich in humus (organic matter). Loamy sand is also good provided that there is adequate irrigation.
- Head development stage is most crucial period for water. Just after transplanting, irrigate twice a day. After that irrigate once a day.
- Moisture level: not less than 25 cm deficit on the root zone. (About 1-2 inches of water are required per week.) Use drip irrigation, if possible, to conserve water.

  - **Field Capacity (FC)**: Loamy soil, especially when rich in humus, has a good balance of sand, silt, and clay, giving it a high water-holding capacity. Humus increases this capacity by enhancing soil structure and water retention.
  - **Permanent Wilting Point (PWP)**: Loamy soils typically have a PWP of around 10-15%.
  - **Available Water Capacity (AWC)**: For loamy soil, the AWC might range from 15% to 35% of the soil volume.
  - **Cabbage Water Requirements:** Cabbage requires consistent moisture, with a preference for soil moisture around 60-80% of the AWC for optimal growth. This means that loamy soil should be kept close to field capacity.

- **Estimating the Deficit:** A "25 cm deficit" in a loamy soil rich in humus indicates that the soil moisture is significantly below the field capacity, likely approaching the lower end of the available water capacity.
- **Soil Moisture Percentage Calculation:**

  Field Capacity (FC) for loamy soil rich in humus might be around 30-35%.
  Permanent Wilting Point (PWP) might be around 10-15%.

  Let's assume,
  FC = 35%
  PWP = 15%
  AWC = FC - PWP = 35% - 15% = 20%

- pH: 6.0 - 7.0 (optimum). Cabbage does not very well suit in a highly acidic soil. A pH range of 5.6 to 7.3 gives a good yield. In Sri Lanka, pH 6.0 - 6.50 is preferred.)
- Mean relative humidity should be in the range of 60% to 90%.

2.2 System Design Procedure

2.2.1 Land Assessment

- Field Size: Measure the total area to be irrigated. Used one acre.
- Soil Type: Determine the soil type to understand its water retention properties.
- Topography: Map the land’s elevation to design an efficient irrigation layout.

2.2.2 Water Source and Supply

- Water Source: Identify a reliable water source (well, river, reservoir).
- Pumping System: Select pumps based on the water source and field size. Calculate water requirements for the cabbage crops. Design and select appropriate pumps.
- Water Storage: Consider a reservoir or storage tank if necessary.

2.2.3 Irrigation Layout

- Main Canal: Design a main canal to distribute water to different sections of the field.
- Sub-Canals: Layout sub-canals branching from the main canal to ensure uniform water distribution.
- Field Channels: Create field channels to direct water flow to the cabbage plants.
- Plan for the placement of cannels

  - Row Spacing: 60-75 cm (24-30 inches)
  - Plant Spacing: 30-45 cm (12-18 inches) within the row

Drip Irrigation Setup:

Single Drip Line Per Row: Typically, one drip line is placed along each row of cabbage plants.

Calculation Example:

- Area of One Acre: 1 acre = 43,560 square feet (or approximately 4,047 square meters)
- Row Spacing: Let's assume 70 cm (0.7 meters).

Number of Rows per Acre:

The width of one acre = 43,560 square feet.

Convert square feet to square meters: $43{,}560 \times 0.092903 \approx 4{,}047$ square meters

Let's assume your field is roughly square-shaped, so each side of the acre would be approximately:

$$\text{Side of the acre} = \sqrt{4{,}047} \approx \underline{\underline{63.6\text{meters}}}$$

Now, the number of rows,

$$\text{Number of Rows} = \frac{\text{Field Width (Side)}}{\text{Row Spacing}} = \frac{63.6\text{meters}}{0.7\text{meters}} \approx \underline{\underline{91\text{rows}}}$$

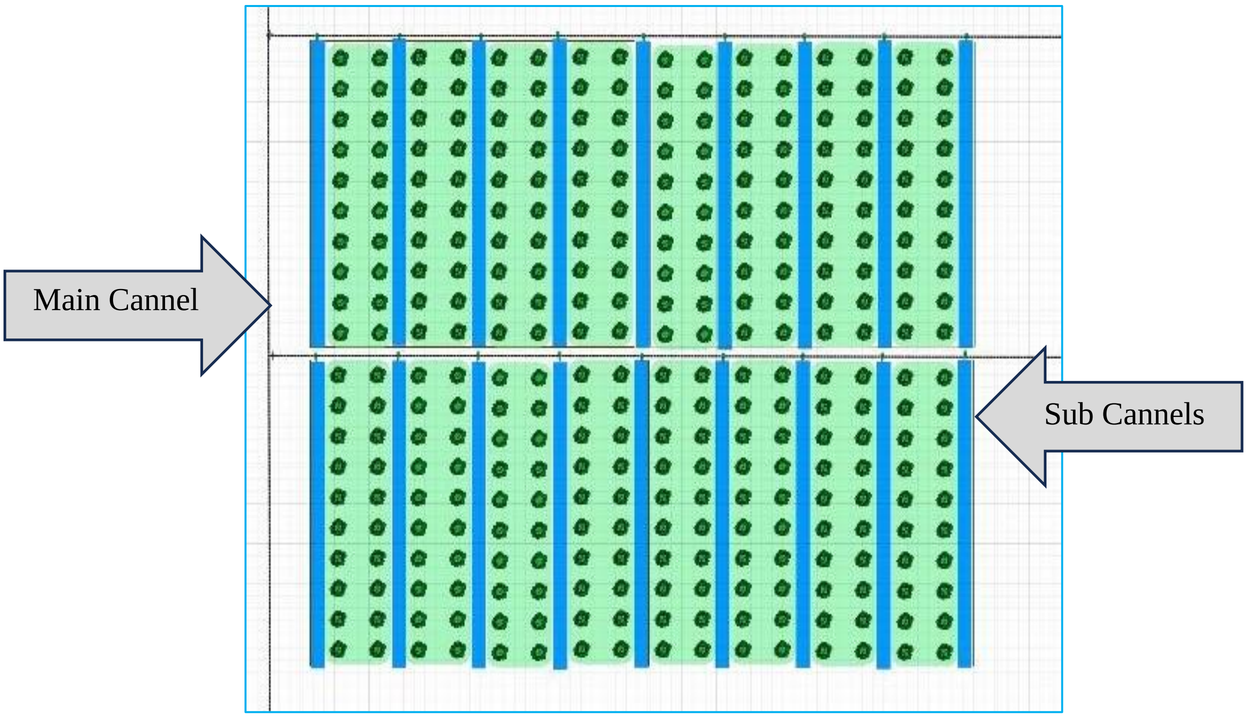


FIGURE 2 : SAMPLE LAYOUT OF THE IRRIGATION SYSTEM

2.3 Tools and Equipment Requirements

2.3.1 Irrigation Components

- Canals and Channels: Concrete or lined ditches for main and sub-canals.
- Gates and Valves: Control water flow in the canals and channels.
- Field Layout: Raised beds or furrows for planting cabbage.

2.3.2 Sensors and Monitoring

- Soil Moisture Sensors: Measure soil moisture levels to optimize irrigation schedules.
- Flow Meters: Monitor water flow rates.
- Weather Sensors: Track weather conditions to adjust irrigation plans.

2.3.3 System Integration and Automation

- Hardware Setup: Set up the hardware for the irrigation system
- Install canals, gates, valves, and pumps as per the design.
- Set up power supply and distribution.
- Install sensors (soil moisture, flow meters, weather sensors).

2.4 Control System Development

- Select a microcontroller or PLC (e.g., Arduino, Raspberry Pi).
- Develop control algorithms for automated irrigation based on sensor data.
- Integrate communication modules (Wi-Fi, GSM, LoRa) for remote monitoring.

2.5 Software Development

2.5.1 Application Development

- Design user interfaces for real-time monitoring and control.
- Develop the app using frameworks like React Native (mobile) and React.js (web).
- Implement features for data visualization and remote control.

2.5.2 Create a Dashboard Using ThingsBoard

- Install and configure ThingsBoard on a local server or cloud.
- Integrate sensor data using MQTT or HTTP protocols.
- Design custom dashboards to display key metrics (soil moisture, weather conditions, system status).

2.6 Circuit Diagram and Wiring

2.6.1 Circuit Design

- Create circuit diagrams showing connections between sensors, microcontroller, and actuators.
- Plan for a stable power supply setup.

2.6.2 Wiring and Installation

- Connect sensors to the microcontroller’s inputs.
- Set up relays or motor drivers for valve control.
- Ensure secure and weatherproof wiring.

3 Procedure

3.1 Development

3.1.1 Microcontroller Programming

- Write code to read data from soil moisture and weather sensors.
- Implement control logic to open/close valves based on sensor data.
- Enable communication with ThingsBoard server.

3.1.2 Application Programming

- Create APIs for data retrieval and control actions.
- Develop frontend features for data visualization and control.
- Integrate with ThingsBoard for real-time updates.

Here, ESP32 microcontroller and soil moisture sensor and flow sensors are used. Developed with Thingsboard Cloud as user interface.

4 Technical Overview for Prototype

4.1 System Architecture

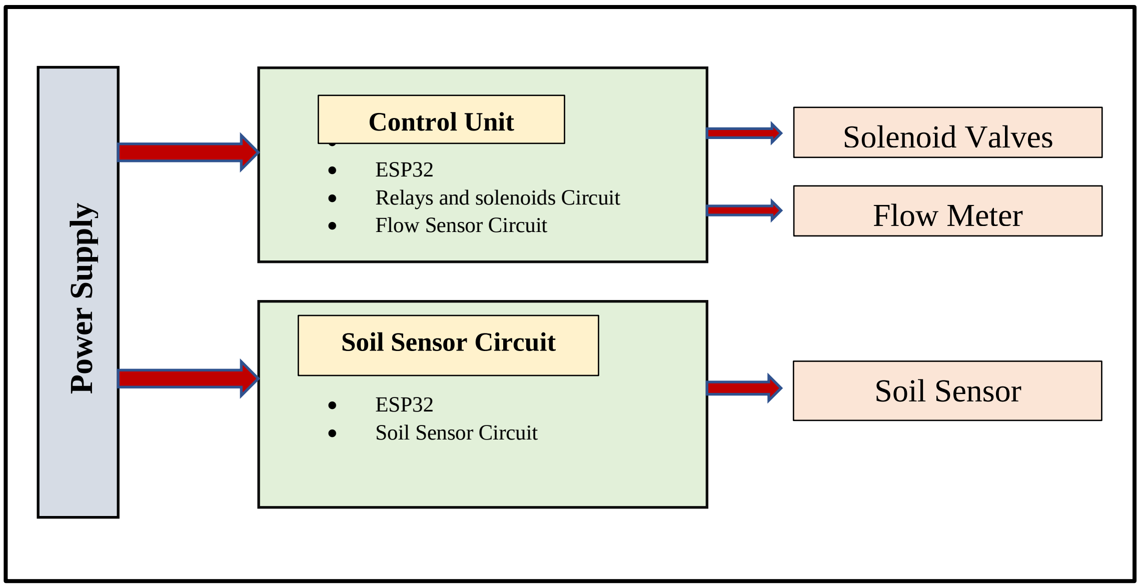


FIGURE 3 : SYSTEM ARCHITECTURE OF PROTOTYPE

4.1.1 Control System

This is named as device_02.

Component list for control system

- ESP32, 38 pin
- 4_Relay Module
- Solenoids × 4
- Flow Sensor
- LM 2596 Adjustable buck converter × 2
- AMS 1117 12V to 5V converter
- Logic level converter
- Power last modules × 3

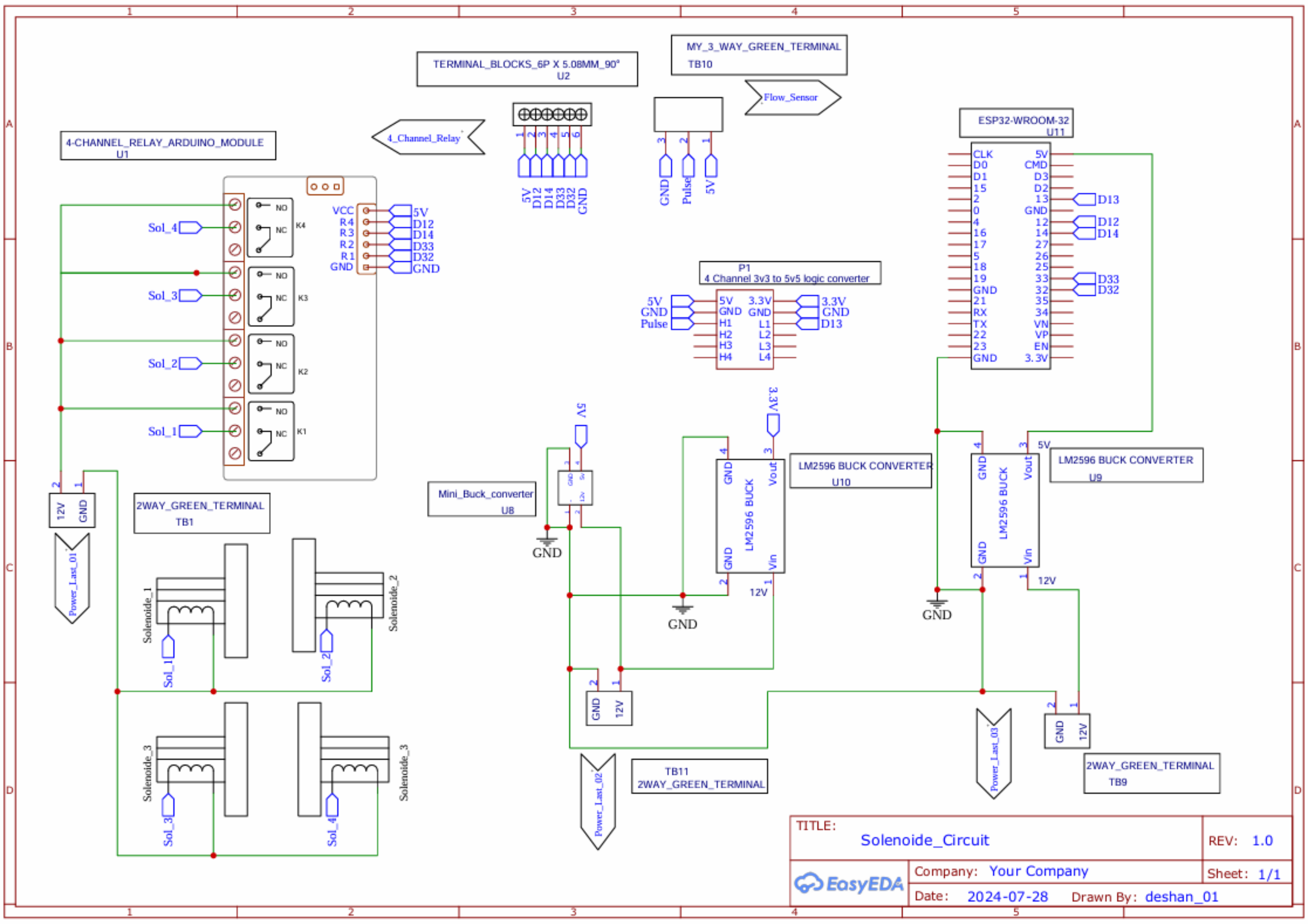


FIGURE 4 : SHEMETIC DIAGRAM FOR CONTROL SYSTEM

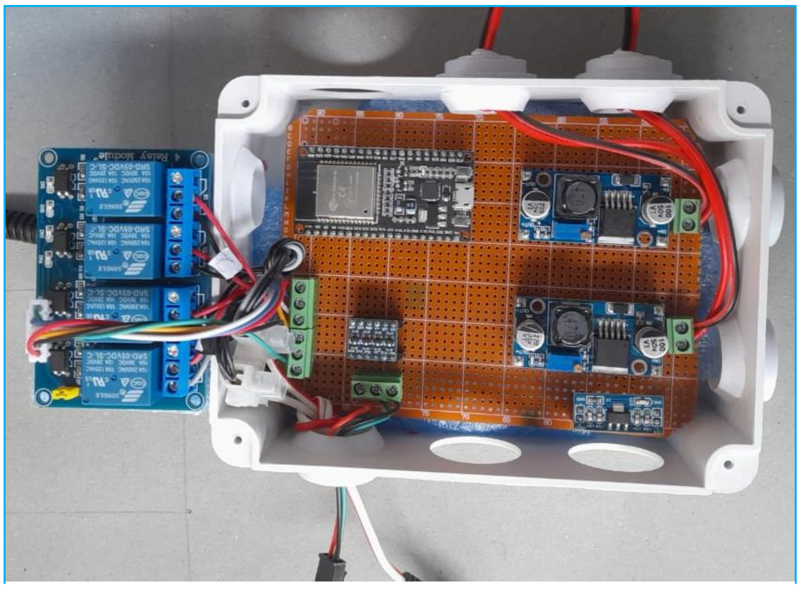


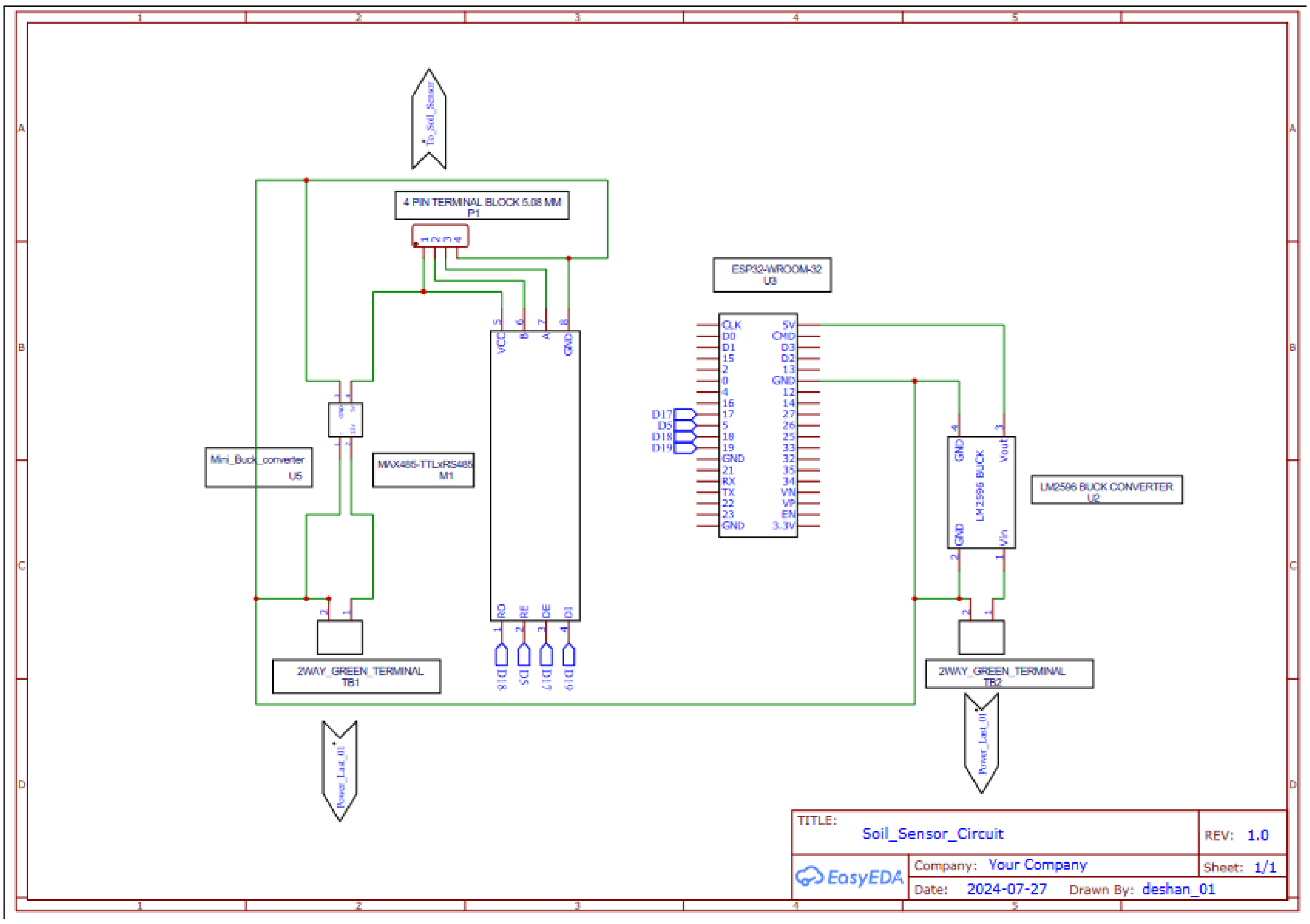
FIGURE 5 : CONTROL CIRCUIT ON DOT BOARD

4.1.2 Soil Sensor Circuit

This is named as device_01.

Component list for the soil sensor circuit

- ESP32, 38 pin
- MAX485 TTL module
- Soil Sensor
- LM 2596 Adjustable buck converter
- AMS 1117 12V to 5V converter



- Power last modules × 2

<u>FIGURE 6 : SHEMETIC DIAGRAM FOR SOIL SENSOR CIRCUIT</u>

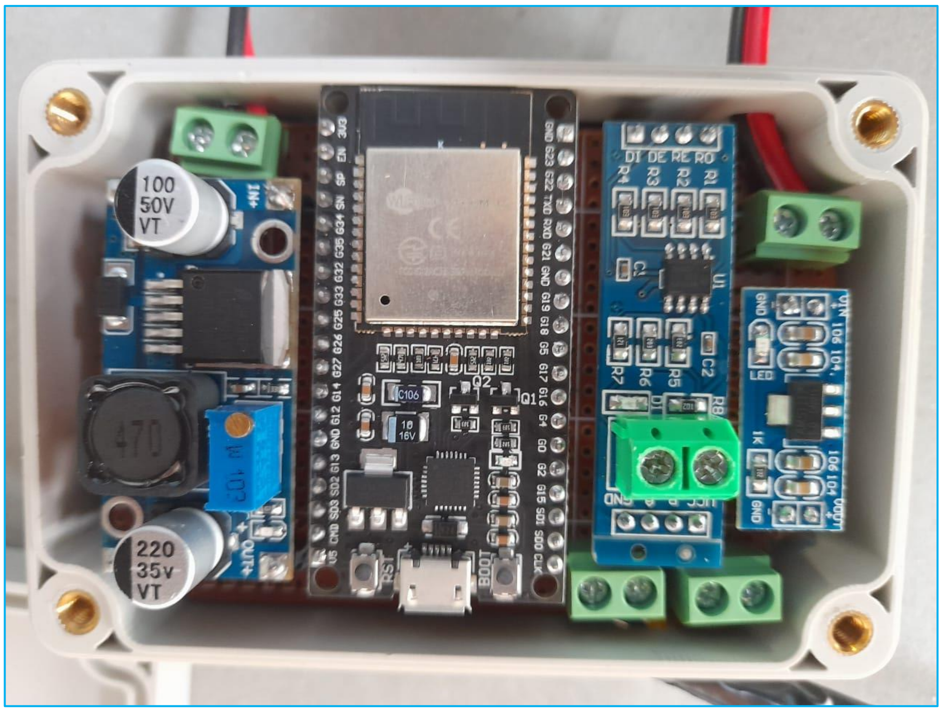
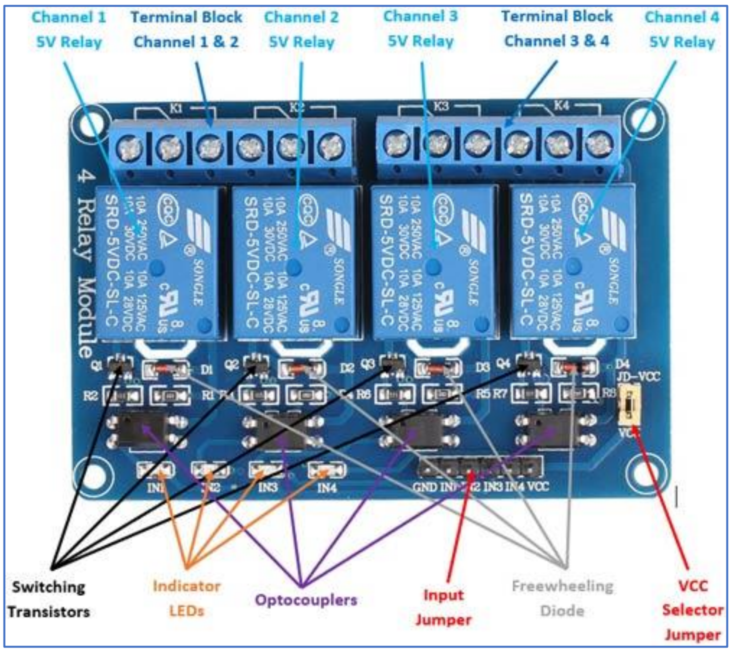


FIGURE 7: SOIL SENSOR CIRCUIT ON DOT BOARD

4.1.3 List of electronic components with specifications

1). SRD-5VDC-SL-C Module (4_Relay_Module)

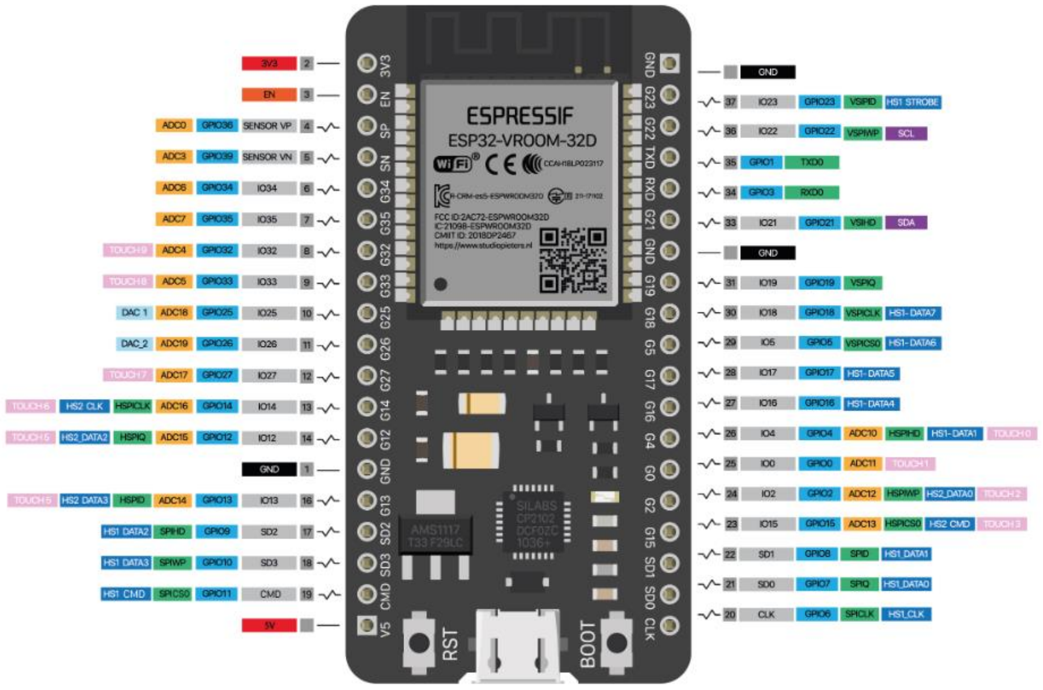


- Supply voltage – 3.75V to 6V
- Trigger current – 5mA
- Current when the relay is active - ~70mA (single), ~300mA (all four)
- Relay maximum contact voltage – 250VAC, 30VDC
- Relay maximum current – 10A

2). LM 2596 Adjustable Converter Module

- Input voltage: 4-35V
- Output Voltage: 1.5-35V (adjustable)
- Output current: rated current 2A, maximum 3A (heat sink required)

3). ESP32-WROOM-32 (ESP-WROOM-32) Pin


3V3 2
EN 3
ADC0 GPIO36 SENSOR VP 4
ADC3 GPIO39 SENSOR VN 5
ADC6 GPIO34 IO34 6
ADC7 GPIO35 IO35 7
TOUCH9 ADC4 GPIO32 IO32 8
TOUCH8 ADC5 GPIO33 IO33 9
DAC 1 ADC18 GPIO25 IO25 10
DAC_2 ADC19 GPIO26 IO26 11
TOUCH7 ADC17 GPIO27 IO27 12
TOUCH6 HS2 CLK HSPICLK ADC16 GPIO14 IO14 13
TOUCH5 HS2_DATA2 HSPIQ ADC15 GPIO12 IO12 14
GND 1
TOUCH5 HS2 DATA3 HSPID ADC14 GPIO13 IO13 16
HS1 DATA2 SPIHD GPIO9 SD2 17
HS1 DATA3 SPIWP GPIO10 SD3 18
HS1 CMD SPICS0 GPIO11 CMD 19
5V
GND
37 IO23 GPIO23 VSPID HS1 STROBE
36 IO22 GPIO22 VSPIWP SCL
35 GPIO1 TXD0
34 GPIO3 RXD0
33 IO21 GPIO21 VSIHD SDA
GND
31 IO19 GPIO19 VSPIQ
30 IO18 GPIO18 VSPICLK HS1- DATA7
29 IO5 GPIO5 VSPICS0 HS1- DATA6
28 IO17 GPIO17 HS1- DATA5
27 IO16 GPIO16 HS1- DATA4
26 IO4 GPIO4 ADC10 HSPIHD HS1-DATA1 TOUCH0
25 IO0 GPIO0 ADC11 TOUCH1
24 IO2 GPIO2 ADC12 HSPIWP HS2_DATA0 TOUCH2
23 IO15 GPIO15 ADC13 HSPICS0 HS2 CMD TOUCH3
22 SD1 GPIO8 SPID HS1_DATA1
21 SD0 GPIO7 SPIQ HS1_DATA0
20 CLK GPIO6 SPICLK HS1_CLK
3V3 EN SP SN G34 G35 G32 G33 G25 G26 G27 G14 G12 GND G13 SD2 SD3 CMD V5
GND G23 G22 TXD RXD G21 GND G19 G18 G5 G17 G16 G4 G0 G2 G15 SD1 SD0 CLK
ESPRESSIF
ESP32-VROOM-32D
RST
BOOT

- Operating Voltage = 5-6V (Preferred)
- Operating Current = 500mA

4). Electric Solenoid Valve 0.5-inch 12VDC NC Plastic Water (RL0047)

- Voltage: DC 12V
- Power: 8W
- Current: 0.6A

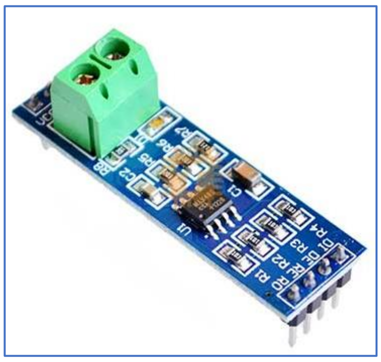
5). Soil NPKPHCTH Sensor (7 in 1) Agricultural NPK Sensor RS485 Modbus

Power Supply = 4.5-30V DC
Max Power Consumption = 0.5W at 24V DC
Output = RS485/4-20mA/0-5V/0-10V

6). YF-S201 Water Flow Sensor 1-30L/min 1.75MPa 0.5 inch

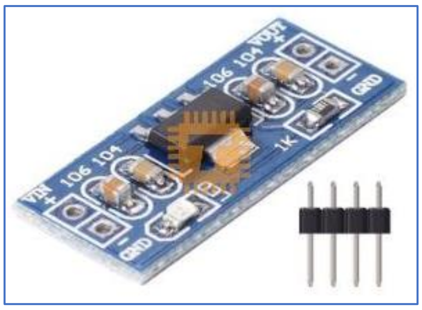
- Rated operating voltage-DC4.5 5V-24V
- Maximum operating current-15 mA(DC 5V)
- Operating voltage range-DC 5~18 V
- load capacity-≤10 mA(DC 5V)
- Allowable withstand voltage-Water pressure below 1.75Mpa

7). MAX485 TTL to RS485 MAX485CSA Converter Module



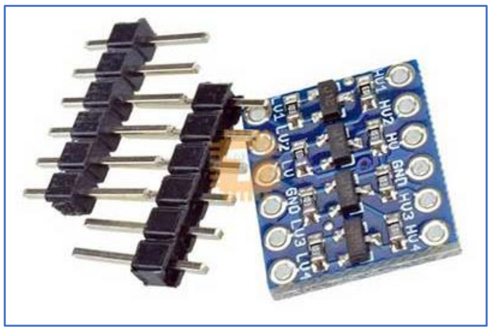
- MAX485 chip is a low power consumption for RS-485 communication, limit the slew rate transceiver
- Operating voltage: 5V

8). AMS1117 12V To 5V DC-DC Step-Down Converter



- Input voltage = DC 6.5V~12V
- Output voltage = DC 5V
- Output current = 800mA

9). 4-Channel 3.3V 5V Bi-Directional Logic Level Shifter Module



- HV 5V supply
- LV 3.3V power supply connection

10) PowerLast Power Backup

Output Voltage = 12V

Output Current = 2A

4.2 Software and Programming

In this smart irrigation system project, IoT integrated with ThingsBoard for the development of, dashboard and the entire hardware segments were scripted in Arduino IDE. It was aimed at measuring the status of the soil and its moisture regime and controlling the water supply flow and its operation in an automatic mode. Such instantaneous data generated from sensors and the points of interest on the dynamic dashboard were used to keep the cropping areas' soil condition at its best to enhance efficient and effectiveness of water usage and to minimize water wastage on irrigation. In the automation process of the system, the relay circuit, the flow meter and the soil moisture sensors were each gives specific roles in the system.

4.2.1 Control System

The relay and flow meter circuit programming was done with a view of regulating water supply. The relays were set up to listen to triggers from the ThingsBoard control panel to control the water valves off site. When the dashboard highlighted certain low or high soil moisture levels a definite low or high value the right or appropriate relay was tripped to open or close the water supply. Furthermore, the flow meter was used to measure the current rate of water delivery and the displayed it on the dashboard. This information was useful in controlling the process of irrigation so as to ensure that the crops are watered as required.

Arduino Code for control Circuit,

```
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include <WiFi.h>

#define WIFI_AP "NONE" //
#define WIFI_PASS "reno0302" // "SLT#54321"

#define TOKEN "WmqYfL6H16TpjYihOx8A"
#define MQTT_SERVER "thingsboard.cloud"
#define MQTT_PORT 1883

#define GPIO1 32
#define GPIO2 33
#define GPIO3 14
#define GPIO4 12
#define GPIO1_PIN 1
#define GPIO2_PIN 2
#define GPIO3_PIN 3
#define GPIO4_PIN 4
#define LED_BUILTIN 2
#define SENSOR 13 // flow sensor input pin

constexpr uint32_t MAX_MESSAGE_SIZE = 1024U;

WiFiClient wifiClient;
PubSubClient client(wifiClient);
boolean gpioState[] = {false, false, false, false};
```

```
// Flow meter variables
long currentMillis = 0;
long previousMillis = 0;
int interval = 1;
boolean ledState = LOW;
float calibrationFactor = 4.5;
volatile byte pulseCount;
byte pulse1Sec = 0;
float flowRate;
unsigned int flowMilliLitres;
unsigned long totalMilliLitres;
float totalLitres; // Add this line

void IRAM_ATTR pulseCounter() {
  pulseCount++;
}

void setup() {
  Serial.begin(115200);
  Serial.println("Hello, ESP32!");

  // Set GPIO pins to OUTPUT and ensure they are off initially
  pinMode(GPIO1, OUTPUT);
  digitalWrite(GPIO1, HIGH);
  pinMode(GPIO2, OUTPUT);
  digitalWrite(GPIO2, HIGH);
  pinMode(GPIO3, OUTPUT);
  digitalWrite(GPIO3, HIGH);
  pinMode(GPIO4, OUTPUT);
  digitalWrite(GPIO4, HIGH);

  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(SENSOR, INPUT_PULLUP);

  pulseCount = 0;
  flowRate = 0.0;
  flowMilliLitres = 0;
  totalMilliLitres = 0;
  totalLitres = 0.0; // Initialize totalLitres
  previousMillis = 0;

  attachInterrupt(digitalPinToInterrupt(SENSOR), pulseCounter, FALLING);   //
flow sensor pulse count

  WiFi.begin(WIFI_AP, WIFI_PASS);
  InitWiFi();
  client.setServer(MQTT_SERVER, MQTT_PORT);
  client.setCallback(on_message);

  // Set initial state of GPIO to be off
  for (int i = 0; i < 4; i++) {
    gpioState[i] = false;
  }

  // Publish initial state to ThingsBoard
  publish_gpio_status_to_thingsboard();
}

void on_message(char* topic, byte* payload, unsigned int length) {
  Serial.println("On message");

  String payloadStr;
  for (unsigned int i = 0; i < length; i++) {
    payloadStr += (char)payload[i];
  }

  Serial.println("setGpioStatus: " + payloadStr);
  Serial.println("Topic: " + String(topic));

  DynamicJsonDocument json(256);
  DeserializationError error = deserializeJson(json, payloadStr);

  if (error) {
    Serial.print("deserializeJson() failed: ");
    Serial.println(error.c_str());
    return;
  }

  String methodName = json["method"].as<String>();

  if (methodName.equals("getGpioStatus1")) {
    String responseTopic = String(topic);
    responseTopic.replace("request", "response");
    String data = gpioState[0] ? "true" : "false";
    client.publish(responseTopic.c_str(), data.c_str());
  }
  else if (methodName.equals("getGpioStatus2")) {
    String responseTopic = String(topic);
    responseTopic.replace("request", "response");
    String data = gpioState[1] ? "true" : "false";
    client.publish(responseTopic.c_str(), data.c_str());
  }
  else if (methodName.equals("getGpioStatus3")) {
    String responseTopic = String(topic);
    responseTopic.replace("request", "response");
    String data = gpioState[2] ? "true" : "false";
    client.publish(responseTopic.c_str(), data.c_str());
  }
   else if (methodName.equals("getGpioStatus4")) {
    String responseTopic = String(topic);
    responseTopic.replace("request", "response");
    String data = gpioState[3] ? "true" : "false";
    client.publish(responseTopic.c_str(), data.c_str());
  }
  else if (methodName.equals("setGpioStatus")) {
    set_gpio_status(json["params"]["pin"], json["params"]["enabled"]);
    String responseTopic = String(topic);
    responseTopic.replace("request", "response");
    responseTopic.replace("\"", "");
    Serial.print("Response: ");
    Serial.println(responseTopic);
    String data;

    if (json["params"]["pin"] == 1){
      data = gpioState[0] ? "true" : "false";
      client.publish(responseTopic.c_str(), data.c_str());
    } else if (json["params"]["pin"] == 2){
      data = gpioState[1] ? "true" : "false";
      client.publish(responseTopic.c_str(), data.c_str());
    } else if (json["params"]["pin"] == 3){
      data = gpioState[2] ? "true" : "false";
      client.publish(responseTopic.c_str(), data.c_str());
    } else if (json["params"]["pin"] == 4){
      data = gpioState[3] ? "true" : "false";
      client.publish(responseTopic.c_str(), data.c_str());
    }

    // Publish updated relay statuses as attributes to ThingsBoard
    publish_gpio_status_to_thingsboard();
  }
}

void set_gpio_status(int pin, boolean enabled) {
  if (pin == GPIO1_PIN) {
    digitalWrite(GPIO1, enabled ? LOW : HIGH);
    gpioState[0] = enabled;
  } else if (pin == GPIO2_PIN) {
    digitalWrite(GPIO2, enabled ? LOW : HIGH);
    gpioState[1] = enabled;
  } else if (pin == GPIO3_PIN) {
    digitalWrite(GPIO3, enabled ? LOW : HIGH);
    gpioState[2] = enabled;
  } else if (pin == GPIO4_PIN) {
    digitalWrite(GPIO4, enabled ? LOW : HIGH);
    gpioState[3] = enabled;
  }
}
void InitWiFi() {
  Serial.println("Connecting to AP ...");
  WiFi.begin(WIFI_AP, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connected to AP");
}
```

```
void reconnect() {
  while (!client.connected()) {
    int status = WiFi.status();
    if (status != WL_CONNECTED) {
      WiFi.begin(WIFI_AP, WIFI_PASS);
      while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
      }
      Serial.println("Connected to AP");
    }
    Serial.print("Connecting to ThingsBoard node ...");
    if (client.connect("d2877340-fd4f-11ee-8c7f-a31d0e3c99f8", TOKEN, NULL)) {
      Serial.println("[DONE]");
      client.subscribe("v1/devices/me/rpc/request/+");
    } else {
      Serial.print("[FAILED] [ rc = ");
      Serial.print(client.state());
      Serial.println(" : retrying in 5 seconds]");
      delay(5000);
    }
  }
}

void publish_gpio_status_to_thingsboard() {
  DynamicJsonDocument json(256);
  json["GPIO1"] = gpioState[0] ? "true" : "false";
  json["GPIO2"] = gpioState[1] ? "true" : "false";
  json["GPIO3"] = gpioState[2] ? "true" : "false";
  json["GPIO4"] = gpioState[3] ? "true" : "false";
  String strPayload;
  serializeJson(json, strPayload);
  client.publish("v1/devices/me/attributes", strPayload.c_str());
}

void publish_flow_meter_to_thingsboard() { // Flow sensor
  DynamicJsonDocument json(256);
  json["FlowRate"] = flowRate;
  json["TotalMilliLitres"] = totalMilliLitres;
  String strPayload;
  serializeJson(json, strPayload);
  client.publish("v1/devices/me/attributes", strPayload.c_str());
  client.publish("v1/devices/me/telemetry", strPayload.c_str());
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  currentMillis = millis();  // flow sensor calculation begining
  if (currentMillis - previousMillis > interval) {
    pulse1Sec = pulseCount;
    pulseCount = 0;

    flowRate = ((1000.0 / (millis() - previousMillis)) * pulse1Sec) /
calibrationFactor;
    //flowrate = -0.0231*(flowrate^2) + 1.9599*flowrate - 41.458 // calibrated
value for flowrate
    previousMillis = millis();

    flowMilliLitres = (flowRate / 60) * 1000;
    totalMilliLitres += flowMilliLitres;
    totalLitres = totalMilliLitres / 1000.0;


    Serial.print("Flow rate: ");
    Serial.print(int(flowRate));
    Serial.print(" L/min\t");

    Serial.print("Output Liquid Quantity: ");
    Serial.print(totalMilliLitres);
    Serial.print(" mL / ");
    Serial.print(totalMilliLitres / 1000);
    Serial.println(" L");

    // Publish flow meter data to ThingsBoard
    publish_flow_meter_to_thingsboard();
  }
}
```

- Soil Sensor Circuit

The programming of the soil sensor was majorly directed to getting real time data on moisture content of the soil and relay this information to ThingsBoard. The sensors were designed to record and transmit the soil parameters at a definite interval to the dashboard besides which the low and over-moisture levels were defined. If the soil moisture by the presented probes equaled to or transcended or, on the contrary, dropped below these numbers, the system initiated corresponding relays to correct the process of irrigation. Real time monitoring and control means that the crops would be watered optimally and excess water would not be used hence increasing the growth of the crops.

Arduino Code for soil sensor circuit,

```
#include <ModbusMaster.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include <WiFi.h>

// WiFi credentials
#define WIFI_AP "NONE"//
" #define WIFI_PASS "reno0302//"SLT#54321"

// ThingsBoard credentials
#define TOKEN "Ev9MRc5X4CZRudCVOseu"
#define MQTT_SERVER "thingsboard.cloud"
#define MQTT_PORT 1883

// Modbus settings
#define MAX485_DE 17 // DE
#define MAX485_RE 5  // RE
#define MODBUS_RX_PIN 18 // R0
#define MODBUS_TX_PIN 19 // D1
#define MODBUS_SERIAL_BAUD 4800

constexpr int MAX_VALUES = 10;
int values[MAX_VALUES] = {0}; // Initialize all values to 0
int currentIndex  = 0;
int sum = 0;

// Initialize Modbus and MQTT
ModbusMaster node;
WiFiClient wifiClient;
PubSubClient client(wifiClient);

void setup() {
  // Initialize serial communication
  Serial.begin(115200);

  // Initialize control pins for MAX485
  pinMode(MAX485_RE, OUTPUT);
  pinMode(MAX485_DE, OUTPUT);
  // Start with the receiver enabled
  digitalWrite(MAX485_RE, LOW);
  digitalWrite(MAX485_DE, LOW);

  // Initialize Modbus communication
  Serial2.begin(MODBUS_SERIAL_BAUD, SERIAL_8N1, MODBUS_RX_PIN, MODBUS_TX_PIN);
  Serial2.setTimeout(200);

  // Modbus slave ID 1
  node.begin(1, Serial2);

  // Callbacks for Modbus communication
  node.preTransmission(modbusPreTransmission);
  node.postTransmission(modbusPostTransmission);

  // Initialize WiFi and MQTT
  WiFi.begin(WIFI_AP, WIFI_PASS);
  InitWiFi();
  client.setServer(MQTT_SERVER, MQTT_PORT);
}

void modbusPreTransmission() {
  delay(500);
  digitalWrite(MAX485_RE, HIGH);
  digitalWrite(MAX485_DE, HIGH);
}

void modbusPostTransmission() {
  digitalWrite(MAX485_RE, LOW);
  digitalWrite(MAX485_DE, LOW);
  delay(500);
}

void InitWiFi() {
  Serial.println("Connecting to AP ...");
  WiFi.begin(WIFI_AP, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connected to AP");
}

void reconnect() {
  while (!client.connected()) {
    int status = WiFi.status();
    if (status != WL_CONNECTED) {
      WiFi.begin(WIFI_AP, WIFI_PASS);
      while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
      }
      Serial.println("Connected to AP");
    }
    Serial.print("Connecting to ThingsBoard node ...");
    if (client.connect("5428cc90-47d9-11ef-af43-61b559981b3b", TOKEN, NULL)) {
      Serial.println("[DONE]");
    } else {
      Serial.print("[FAILED] [ rc = ");
      Serial.print(client.state());
      Serial.println(" : retrying in 5 seconds]");
      delay(5000);
    }
  }
}

void publish_readings(float soilSensor, float averageSoilSensor) {
  DynamicJsonDocument json(256);
  json["soilSensor"] = soilSensor; // Individual soil moisture reading
  json["averageSoilSensor"] = averageSoilSensor; // Average soil moisture
reading
  String strPayload;
  serializeJson(json, strPayload);

  // Publish the JSON string to ThingsBoard telemetry
  client.publish("v1/devices/me/telemetry", strPayload.c_str());
  // Publish the JSON string to ThingsBoard attributes
  client.publish("v1/devices/me/attributes", strPayload.c_str());

  Serial.println("Telemetry and Attributes: " + strPayload);
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  uint8_t result;
  uint16_t data[1];
  float soilSensor;

  result = node.readHoldingRegisters(0x0000, 1); // Reading soil moisture
value from register 0x0000
  if (result == node.ku8MBSuccess) {
    Serial.println("Success, Received data:");

    data[0] = node.getResponseBuffer(0x00);

    soilSensor = data[0] / 10.0; // Convert to float value by dividing by 10
    //Serial.print("Soil Moisture: ");
    //Serial.print(soilSensor);
    //Serial.println(" %RH");
    int soilValue = data[0] / 10;

    sum -= values[currentIndex];          // Subtract the oldest value from
the sum
    values[currentIndex] = soilValue;     // Replace it with the new value
    sum += soilValue;                  // Add the new value to the sum
    currentIndex = (currentIndex + 1) % MAX_VALUES;  // Move to the next index

      // Calculate the average of the last 10 values
    float avgSoilValue = (float)sum / MAX_VALUES;

      // Create a JSON document
    StaticJsonDocument<200> doc;
    doc["soilSensor"] = avgSoilValue;
    doc["soilSensorval"] = soilSensor;

    // Serialize the JSON document to a string
    char buffer[256];
    size_t n = serializeJson(doc, buffer);

    // Publish the JSON string to ThingsBoard
    client.publish("v1/devices/me/telemetry", buffer, n);

    // Print the JSON string to the Serial Monitor
    Serial.println(buffer);

  } else {
    Serial.print("Failed, Response Code: ");
    Serial.print(result, HEX);
    Serial.println("");
    Serial.println("Check wiring and device address.");
  }

  delay(1000);
}
```

4.2.2 Thingsboard Cloud

Used a dashboard on thingsboard cloud platform named as smart irrigation system under Fazenda Smart Agro. Created following widgets for monitoring the processes,

1. Measure the soil moisture percentage
2. Measure the flow rate
3. Display the soil moisture variations on graph
4. Display the flow sensor variation on graph
5. Calculate the total Liters for one day
6. Checking whether devices are active or not
7. Turn on/off solenoid on dashboard
8. Display the day-by-day water consumption

9. Threshold soil moisture values.
10. Display the critical alarms – There are two alarms
    - ➢ If the total consumption of water is more than 100L
    - ➢ If the soil moisture less than SoilMin or greater than SoilMax

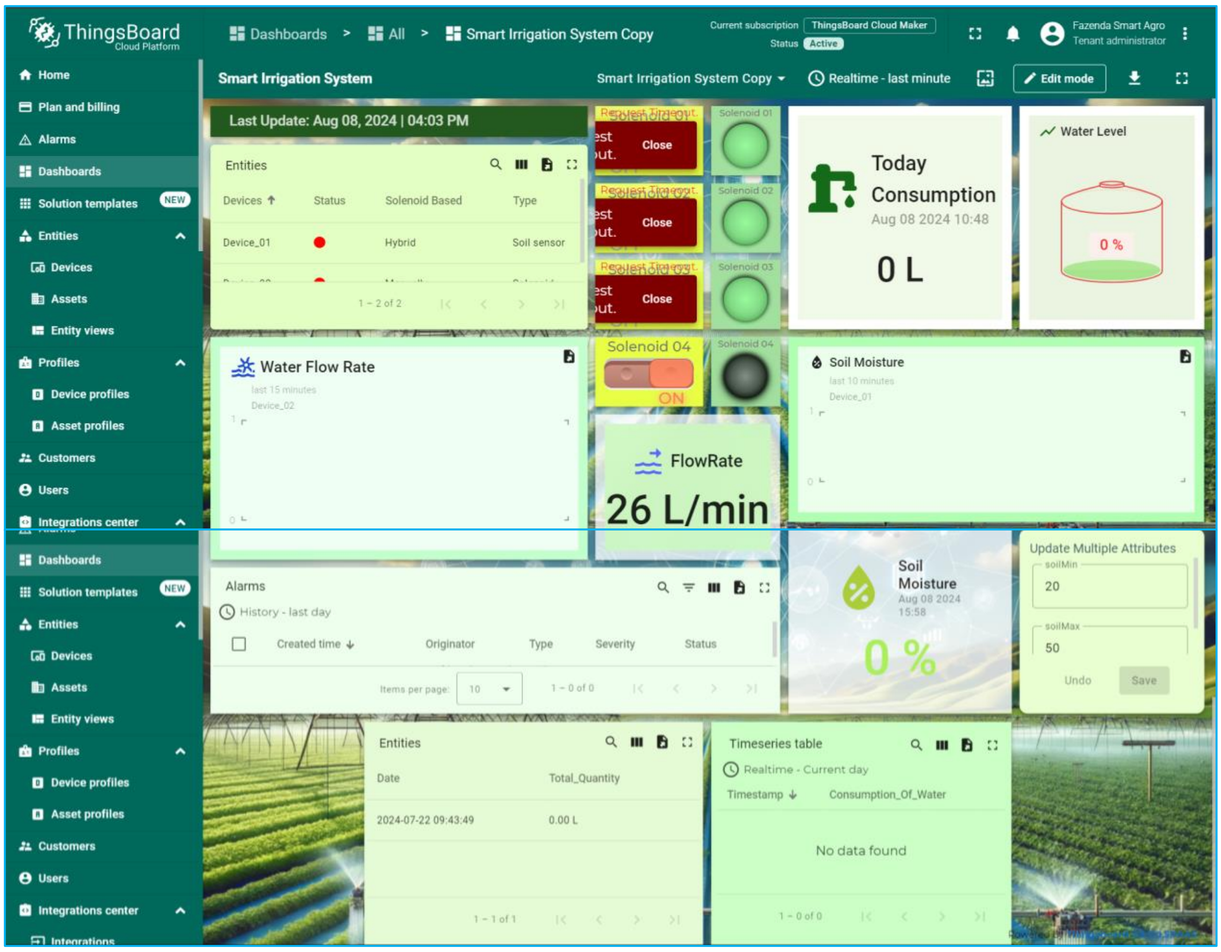


FIGURE 9: THINGS BOARD DASHBOARD

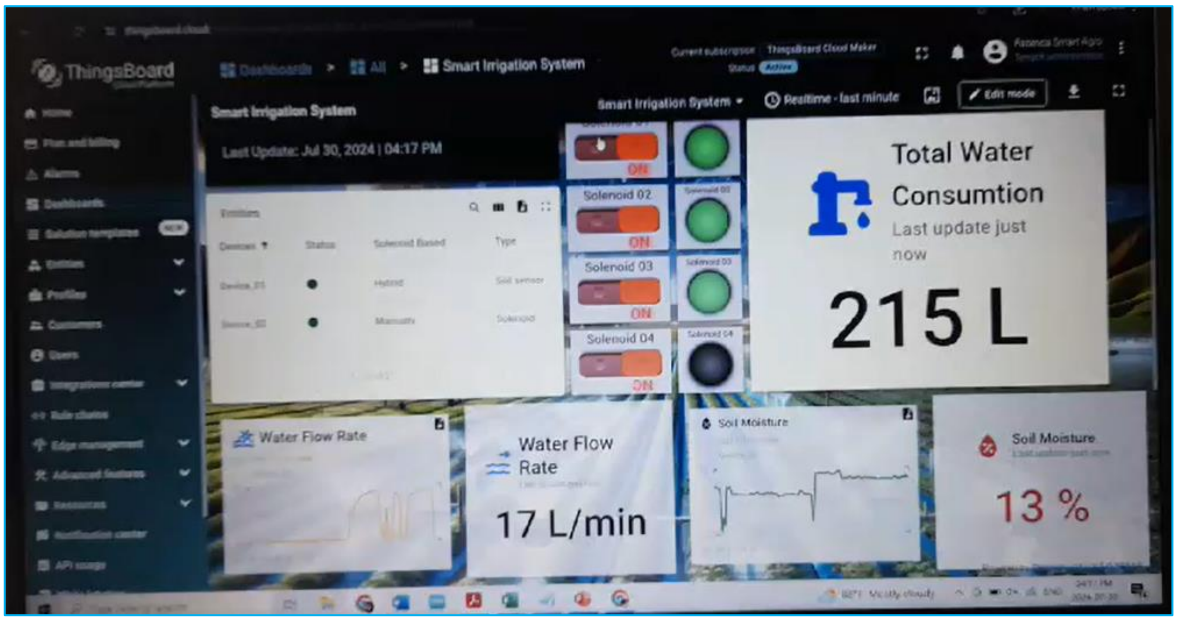


FIGURE 8: THINGSBOARD CLOUD DASHBOARD WHEN WORKING

5 Testing and Calibration

5.1 Deployment and Maintenance

Initial Testing

- Checked continuity and wire connection testing
- Calibrate sensors and ensure accurate readings.
- Test control logic and automated responses.
- Validate communication with the ThingsBoard dashboard.

Full Deployment

- Scale the installation to cover the full irrigation area.
- Perform comprehensive testing and fine-tuning.
- Train personnel on system operation and maintenance.

Ongoing Monitoring and Maintenance

- Regularly monitor sensor data and system performance via the dashboard.
- Schedule maintenance for hardware components.
- Update software as needed for improvements and bug fixes.

5.2 Soil sensor calibration

Components needed,

- Measuring cups
- Weight scale
- Dried soil
- A spray to distribute water evenly
- Tray
- NPK Soil sensor and thingsboard cloud dashboard

The below steps are followed,

Step_01 – Dried soil until moisture content 0% of soil.

Step_02 – Divided the 0-100% into several values and roughly calculated how much water should be added to the dried soil.

Step_03 – Measured a known quantity of soil and righted down the weight.

Step_04 – Add roughly calculated water to the dried soil and mixed well evenly using a spray.

Step_05 – Measured the total weight of the soil after adding water.

Step_06 – Righted down the displayed value of the soil sensor on the dashboard (Display/Given value)

Step_07 – Calculated the Actual soil moisture value and noted down a MS excel sheet

Step_08 – Repeat the same procedure.

Step_09 – Sketched a graph for soil sensor given values against actual values.

Step_10 – Generate an equation for the calibration of soil sensor using MS excel.

TABLE 1 : ACTUAL VALUES AND MEASURED VALUES FOR SOIL SENSOR

Actual value (%)	Measured value (%)	Error	Squared Error
0.0000	6	-6.0000	36.0000
5.1324	8	-2.8676	8.2231
15.1460	13	2.1460	4.6053
21.0420	18	3.0420	9.2538
29.6703	24	5.6703	32.1526
35.2158	39	-3.7842	14.3202
45.2747	54	-8.7253	76.1304
59.5604	62	-2.4396	5.9516
74.7272	64	10.7272	115.0728
84.1758	66	18.1758	330.3597
100.0000	68	32.0000	1024.0000

MSE:	150.5517828
Correlation	0.951751276

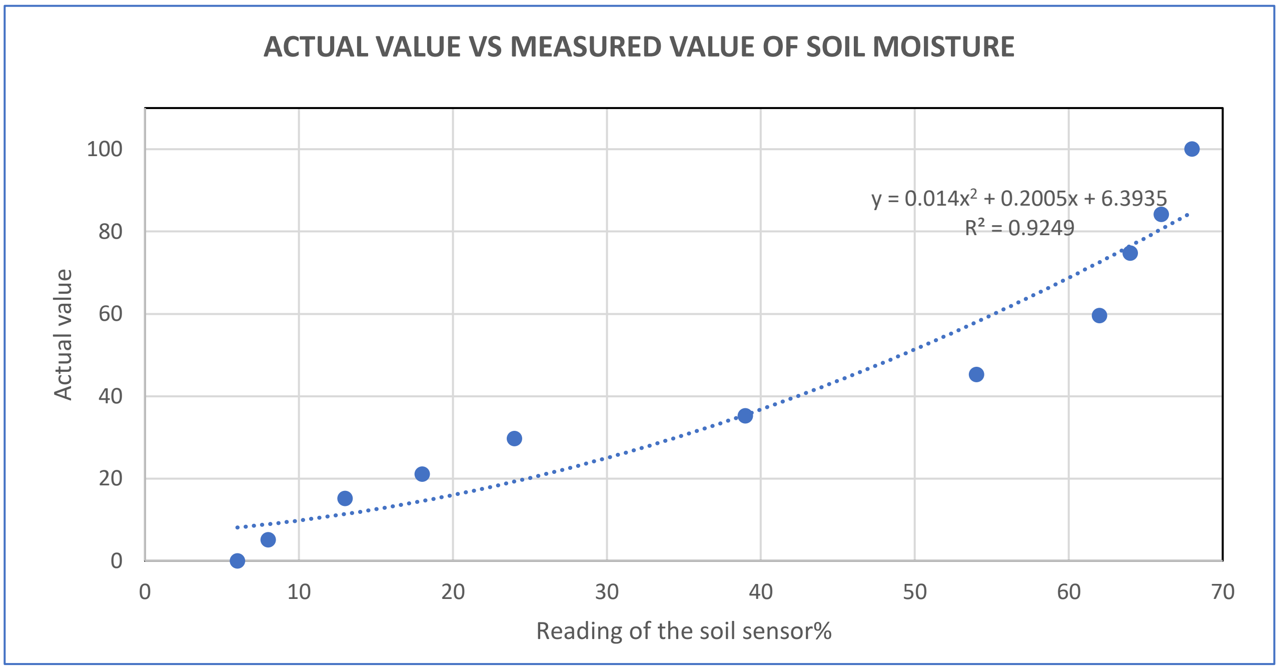


FIGURE 10 : GRAPH OF ACTUAL VALUE VS MEASURED VALUE OF SOIL SENSOR

*Equation for correction, (y = Actual value, x = Measured value on dashboard)

$y = 0.014x^2 + 0.2005x + 6.3935$

5.3 Flow sensor calibration

Components needed,

- Measuring cups
- Stop watch
- Flow sensor and dashboard

The below steps are followed,

Step_01 – Filled the tank with water up to some level. (3/4 tank)

Step_02 – Let to flow some water out from the pipes.

Step_03 – First turn on all the solenoids,

Step_03 – Set the stop watch and exactly 0.2L water was measured

Step_04 – Righted down the given calculated total liters on thinsboard dashboard.

Step_05 – Entered flow sensor given value and calculated the flow sensor actual value, into a excel sheet.

Step_06 – Repeat the same procedure (for three solenoids are open, two solenoids are open and one solenoid are open..) using thingsboard dashboard. And took at least three readings for each step.

Step_07 – Sketched a graph for flow sensor given values against actual values.

Step_08 – Generate an equation for the calibration of flow sensor using MS excel.

4 solenoid open

started volume in L	end volume in L	exact volume in L	duration (min)	sensor given reading(L/min)	Actual volume in L	actual reading (L/min)
48.83	104.67	55.84	1:19	42.41	0.2	0.1519
104.67	176.5	71.83	1:43	41.84	0.2	0.1165
176.5	249.86	73.36	1:46	41.53	0.2	0.1132

Average				41.93		0.1272

3 solenoids open 1 closed

started volume in L	end volume in L	exact volume in L	duration (min)	sensor given reading(L/min)	Actual volume in L	actual reading (L/min)
318.13	400.25	82.12	1:54	43.22	0.2	0.1053
400.25	499.41	99.16	2:16	43.75	0.2	0.0882
499.41	604.16	104.75	2:22	44.01	0.2	0.0845

Average				43.66		0.0927

TABLE 2 : ACTUAL VALUE AGAINST MEASURED VALUE OF FLOW SENSOR

Measured Value (L/min)	**Actual Value in (L/min)**	**Error**	**Squared Error**
42.41	0.1519	42.2581	1785.7470
41.84	0.1165	41.7235	1740.8505
41.53	0.1132	41.4118	1714.9372
43.22	0.1053	43.1147	1858.8808
43.75	0.0882	43.6618	1906.3493
44.01	0.0845	43.9255	1929.4496

MSE:	1822.70238149
Correlation	-0.644319621

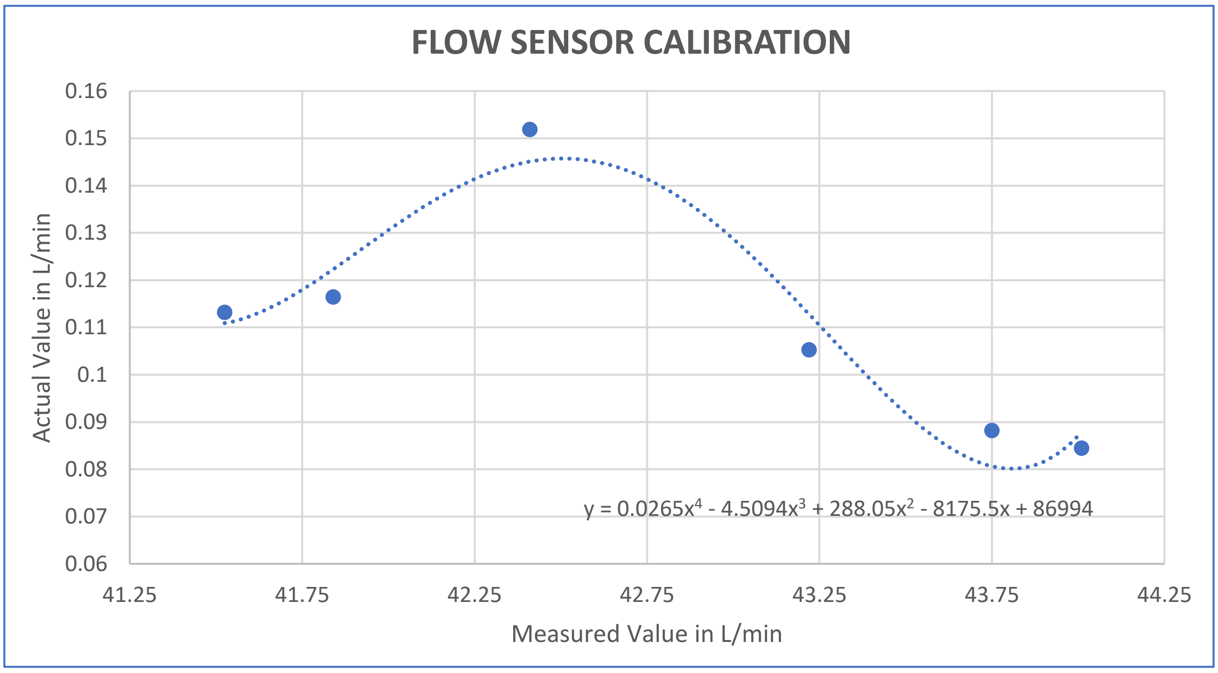


FIGURE 11 : THE GRAPH OF ACTUAL VALUE AGAINST MEASURED VALUE OF FLOW SENSOR

*Equation for correction, (y = Actual value, x = Measured value on dashboard)

$y = 0.0265x^4 - 4.5094x^3 + 288.05x^2 - 8175.5x + 86994$

6 Challenges and Solutions

6.1 Challenges

- By giving 3.3V to the esp32 is not sufficient for to communicate
- Initially, two power last were used for control system. One for solenoids and one for all the other modules. When giving power to the electronic components except solenoids, the esp32, microcontroller didn’t get power properly.
- Same thing happen when testing the soil sensor circuit also.
- There was a problem with calibrating soil sensor. When adding more water to the initially measured soil reading, mixing procedure was difficult and also unable to get whole sample for soil moisture actual testing in the lab (Some of soil particles are on the tray)
- When adding water more for the calibrating soil sensor, after 30% the water started to come up on the measuring cup.
- When calibrating flow sensor, the flow sensor gave readings on dashboard only for 4 and 3 solenoids were opened.

6.2 Solutions

Gave 5V to the esp32 then it worked

Used 3 power last for control circuit and used two power last for solenoid circuit.

Took a sample of soil moisture content when the content volume is getting increased.

After 30% of water adding, water is coming up on the measuring cups, Took the other readings like this, (30% to 100%)

Only took reading for the 4 and3 open valves. Otherwise, need to set the prototype to direct pipe line to increase pressure.

7 Conclusion and updating

Cabbage cultivation in Sri Lanka's picturesque highland hills offers peasant farmers opportunities to prosper while conserving precious water, yet success demands attentive planning and adoption of cutting-edge techniques. Where mists cloak verdant slopes and seasonal rains nourish the earth, cabbages thrive on fertile volcanic soil. With diligent study of local hydrology and careful placement of subsurface emitters delivering precise, measured flows, yield and efficiency can elevate small plots to levels supporting humble families for generations to come.

Skillfully arranged plantings and irrigation schedules in concert with the lay of the land maximize the bounty from each drop. Further, enriching the soil with natural amendments derived from the decomposition of once-living matter restores nutritional balance, leading to heartier vegetables and produce of superior quality for consumers and fair return for growers. Through innovation and care for Sri Lanka's life-giving waters and agricultural heritage, humble cultivators may find blessing in abundance.

Can update this one as a mobile application.

8 References/Bibliography

[1]. *5V four-channel Relay Module* (no date) *Components101*. Available at: https://components101.com/switches/5v-four-channel-relay-module-pinout-features-applications-working-datasheet (Accessed: 14 August 2024).

[2]. *MD0042 - LM2596S 3-40V to 1.5-35V 4A DC to DC adjustable step-down Buck Module* (no date) *TRONIC.LK*. Available at: https://tronic.lk/product/lm2596s-3-40v-to-1-5-35v-4a-dc-to-dc-adjustable-step-do (Accessed: 14 August 2024).

[3]. Pieters, A. (2022) *Esp32 – pinout*, *Home -*. Available at: https://www.studiopieters.nl/esp32-pinout/ (Accessed: 14 August 2024).

[4]. *RL0047 - electric solenoid valve 0.5-inch 12VDC NC plastic water* (no date) *TRONIC.LK*. Available at: https://tronic.lk/product/electric-solenoid-valve-0-5-inch-12vdc-nc-plastic-water (Accessed: 14 August 2024).

[5]. *Soil NPKPHCTH sensor (7 in 1) agricultural NPK sensor RS485 Modbus* (no date) *BDTronics.com*. Available at: https://www.bdtronics.com/soil-npkphcth-sensor-agricultural-npk-sensor-7-in-1-rs485-modbus.html (Accessed: 14 August 2024).

[6]. *MD0227 - YF-S201 water flow sensor 1-30L/min 1.75MPA 0.5inch* (no date) *TRONIC.LK*. Available at: https://tronic.lk/product/yf-s201-water-flow-sensor-1-30lmin-1-75mpa-0-5inch (Accessed: 14 August 2024).

[7]. *MD0053 - max485 TTL to RS485 MAX485CSA converter module for Arduino* (no date) *TRONIC.LK*. Available at: https://tronic.lk/product/max485-ttl-to-rs485-max485csa-converter-module-for-ardu (Accessed: 14 August 2024).

[8]. *2PCS AMS1117-5V DC-DC 6-12V to 5V buck step-down converter 25*11mm regulator* (no date) *eBay*. Available at: https://www.ebay.de/itm/124698847136 (Accessed: 14 August 2024).

[9]. *MD0067 - 4-channel 3.3V 5V Bi-Directional Logic Level Shifter Module* (no date) *TRONIC.LK*. Available at: https://tronic.lk/product/4-channel-3-3v-5v-bi-directional-logic-level-shifter-mo (Accessed: 14 August 2024).

[10]. *PowerLast power backup: Online shopping sri lanka: Wi-Fi devices, power backups, telephones* (no date) *PowerLast Power Backup | Online Shopping Sri Lanka: Wi-Fi Devices, Power Backups, Telephones*. Available at: https://eteleshop.slt.lk/product/powerlast-power-backup (Accessed: 14 August 2024).

[11]. *Cabbage production guide.pdf*. Available at: https://ati2.da.gov.ph/e-extension/content/sites/default/files/2023-03/Cabbage%20Production%20Guide.pdf (Accessed: 15 August 2024).

[12]. *Hordi crop – cabbage* (no date) *Department of Agriculture Sri lanka*. Available at: https://doa.gov.lk/hordi-crop-cabbage/ (Accessed: 15 August 2024).