SWE30011 IoT Programming

Individual Assignment Report

Student name: Kien Quoc Mai
Student ID: 103532920
Tutorial 3 (Wednesday 2:30 am BA407)

Table of Contents

1. Summary

Agriculture is an essential field that is constantly evolving in order to meet the increasing demand for food caused by the rapidly growing populations all over the world. A majority of the consumption of freshwater resources across the world is for irrigation purposes. The problem with the traditional watering methods, such as overhead sprinklers and flood type, is that they are often inefficient since an excessive amount of water is used. In order words, the traditional irrigation methods provide more water than the actual amount of water needed by plants. Apart from the waste of water, excessive water can also lead to soil damage and disease. As a result, managing water resources in irrigation is one of the key challenges in agriculture. There has been a number of innovations designed to solve the irrigation problem, such as drip and underground irrigation. Despite being a highly efficient and widely adopted system, those solutions are not ideal since there is minimal control of watering quantity with respect to ambient and environmental factors. For instance, manual intervention is needed to adjust the dripping system when there is a change in weather conditions. Hence, more efficient and intelligent solutions need to be developed and implemented to improve the irrigation process.

In this assignment, an IoT-based system is proposed to continuously monitor ambient data via the use of sensors and automatically provide irrigation decisions. The system consists of an IoT node, various kinds of sensors and actuators, an edge device, a database, an MQTT broker and a user dashboard. Firstly, the IoT node, Arduino Uno, utilised sensors to measure soil moisture, temperature and humidity. It also controls a servo motor and an LCD screen. The Arduino's main responsibility is to capture data and control the irrigation system. Secondly, the edge device is a Raspberry Pi that receives, processes and stores the data from the IoT node. It consists of 2 services: the first service communicates with the IoT node and generates irrigation decisions, and the second service (backend service) is used to store the data into the database. The database is a simple SQL database that stores measurements from each sensor in its own table. Its also used to store configurations and rules for controlling the irrigation system. The next component is the dashboard that is implemented using NextJs. It is capable of displaying both historical and real-time data. In addition, the dashboard allows users to control the irrigation system manually as well as setting rules for automation. The last component in the system is the MQTT broker. The broker is hosted on HiveMQ cloud platform, and it is used by the edge device, the backend service and the dashboard to send and receive data. Regarding automation, the system provides 2 options: machine learning and rules. The "machine learning" option uses a pre-trained logistic regression model to predict irrigation decisions based on sensors' measurements. The "rules" option lets users create custom rules in the form of boolean expressions. Those rules are then stored inside the database, and they will be evaluated to generate irrigation decisions. For example, the rule "soilMoisture < 300" will activate the irrigation system if the soil moisture value falls below 300.

2. Conceptual Design

2.1 Design Diagrams

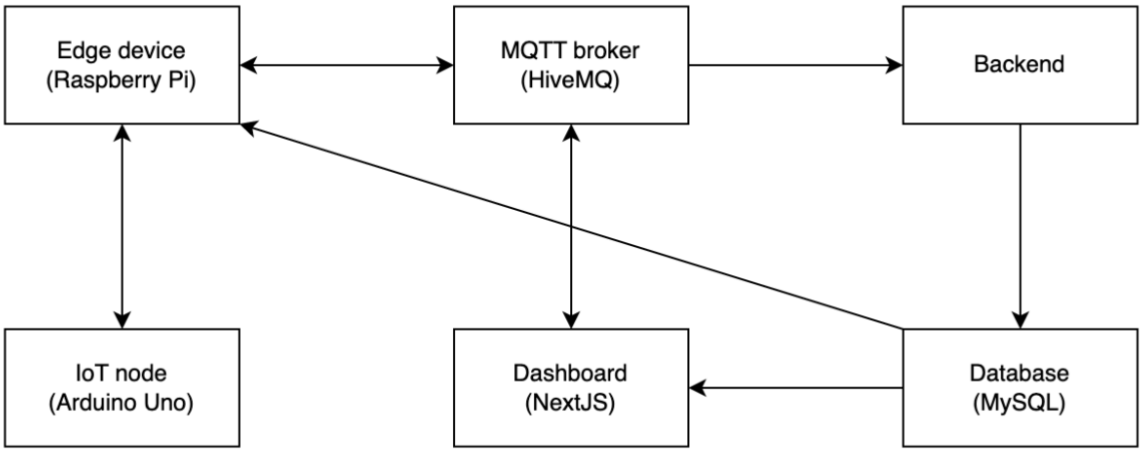


Figure 1: Block diagram

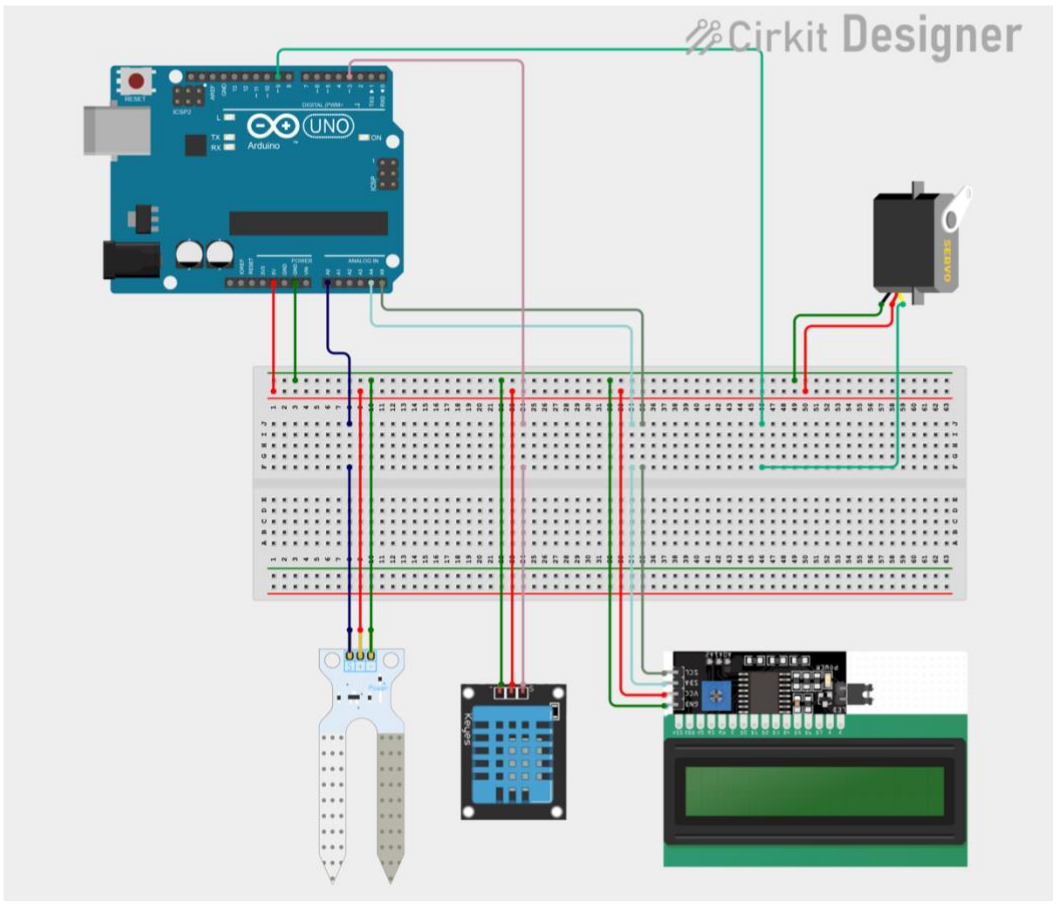


Figure 2: IoT node circuit design

2.2 Use Case Diagrams

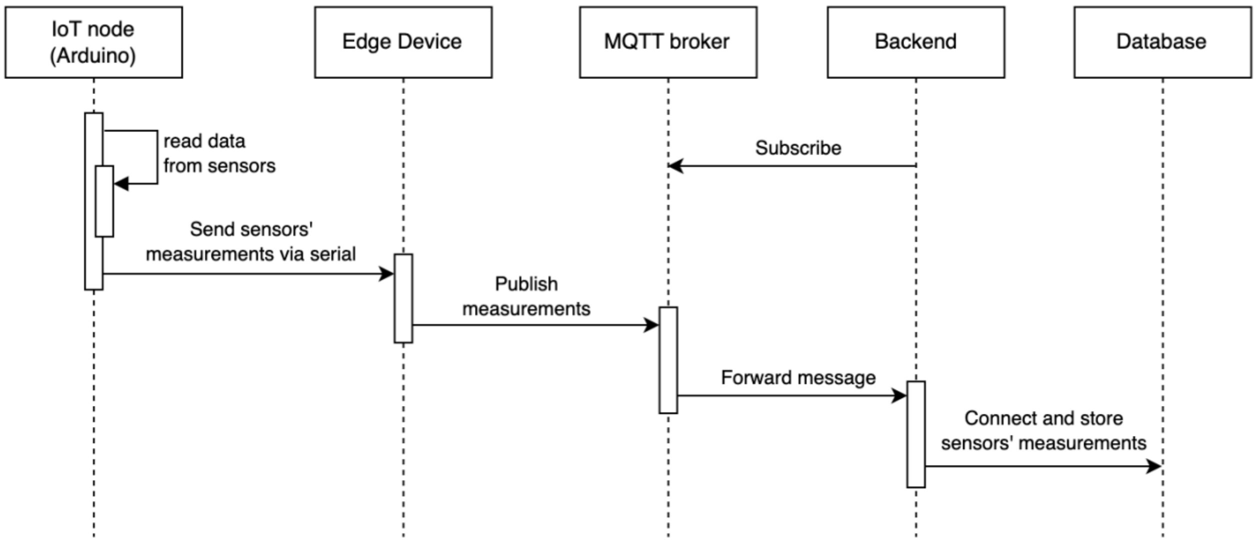


Figure 3: Read and record data

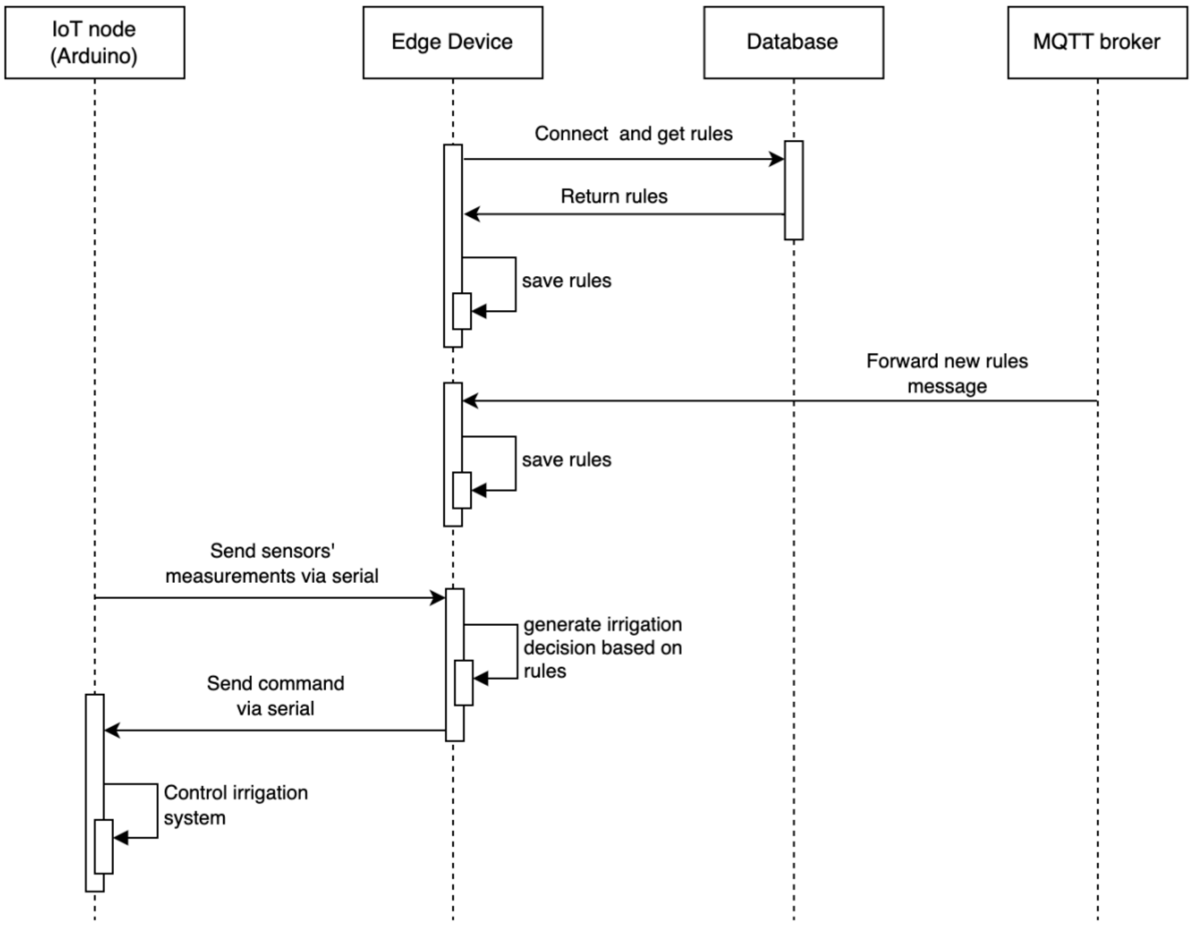

IoT node (Arduino)
Edge Device
Database
MQTT broker
Connect and get rules
Return rules
save rules
Forward new rules message
save rules
Send sensors' measurements via serial
generate irrigation decision based on rules
Send command via serial
Control irrigation system

Figure 4: Control irrigation system using conditional rules

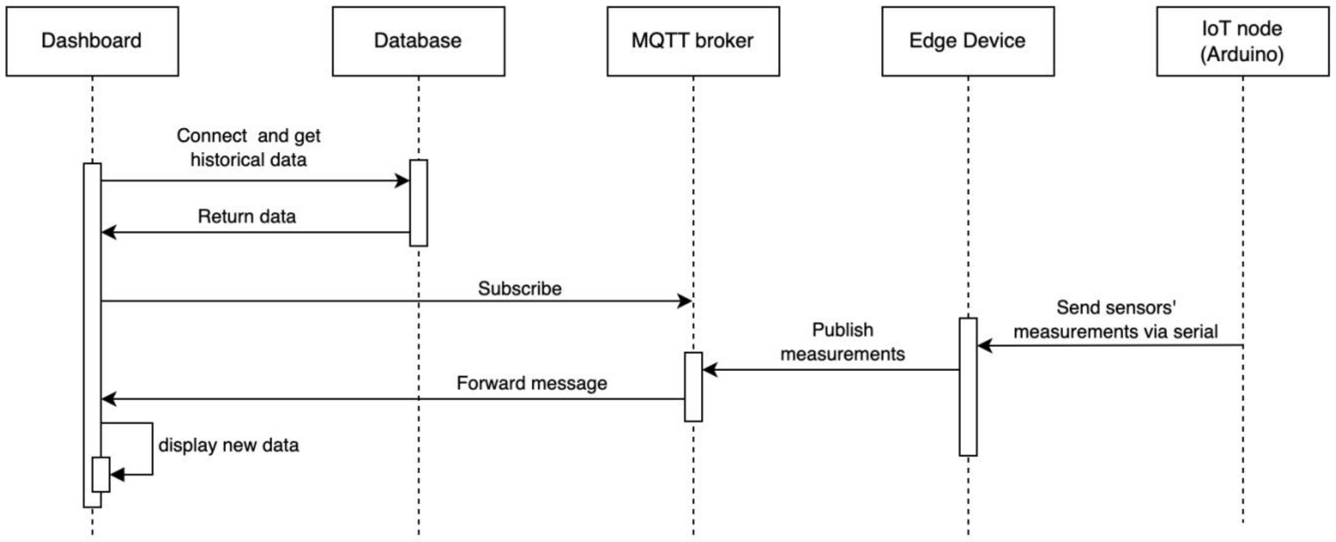


Figure 5: Display data on dashboard

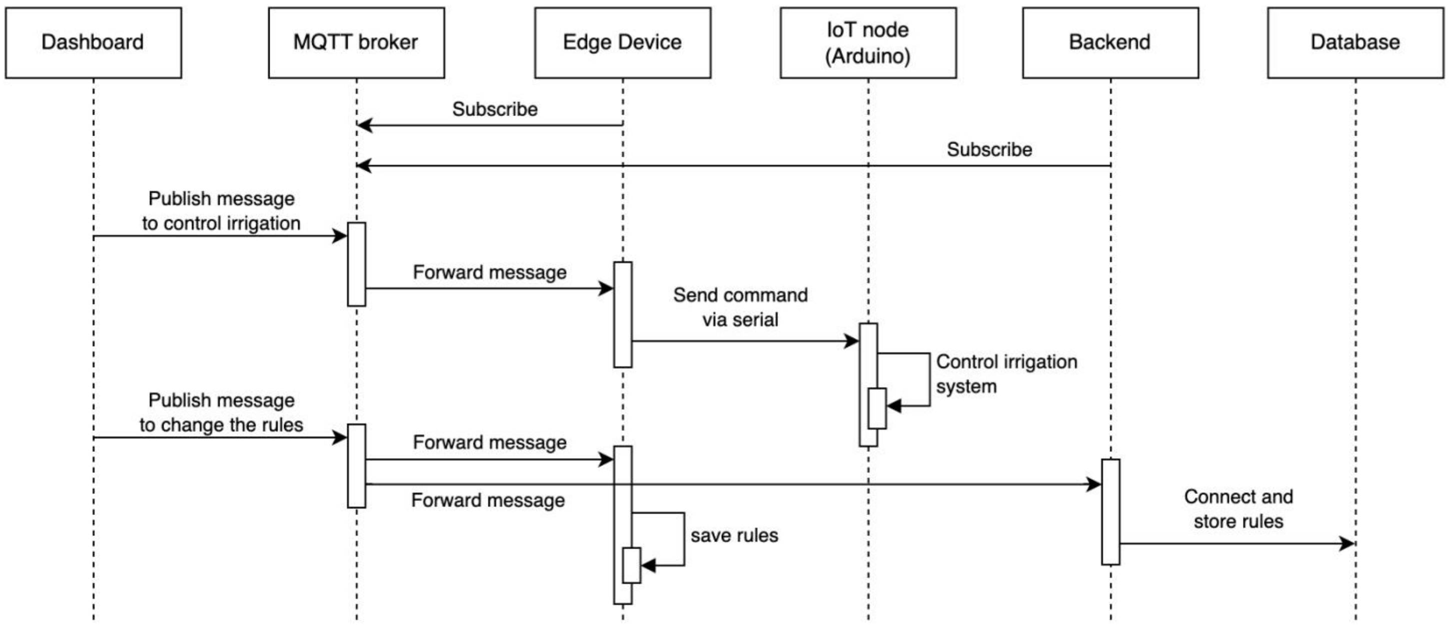


Figure 6: Control IoT node and change rules from dashboard

3. Implementation

3.1 Sensors:

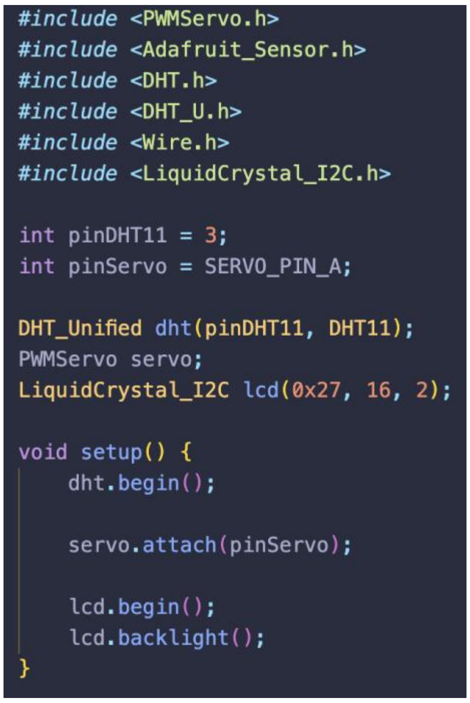
- Resistive soil moisture sensor module: This sensor is used to measure the relative moisture in the soil. Its measurement will tell if the soil is wet or dry. The soil moisture sensor has 3 pins: VCC, ground and signal. The VCC pin is connected to 5V output; the ground pin is connected to ground; the signal pin is connected to A0 since its output is analog signals.
- DHT11 sensor module: The module consists of a pull-up resistor and a DHT11 sensor. This module is used to measure temperature and relative humidity. Temperature and humidity are important factors affecting a plant's water need as they are directly related to water evaporation. For example, a high temperature can cause the water from leaves to evaporate quickly which increases the amount of water needed by a plant. The DHT11 sensor module has 3 pins: VCC, ground and signal. The VCC pin is connected to 5V output; the ground pin is connected to ground; the signal pin is connected to pin 3 (a digital pin) since DHT11 sensor uses its own digital communication protocol. In addition, the sampling rate of DHT11 is 1Hz which means it gives one reading for every second. The library Adafruit DHT sensor is used to control the DHT11 module.

3.2 Actuators:

- LCD I2C: This module consists of an LCD, an I2C module and a potentiometer. The LCD is used to display the latest measurements from the sensors and the current irrigation status. As a result, users can view information about the system. The LCD I2C module has 4 pins: VCC, ground, SDA (data signal) and SCL (clock signal). The VCC pin is connected to 5V output; the ground pin is connected to ground; the SDA pin is connected to A4; the SCL pin is connected to A5. It communicates with the Arduino Uno using the I2C protocol. The library LiquidCrystal_I2C is used to control the LCD I2C.
- SG90 servo motor: A servo motor is used to turn the irrigation on and off. By changing the rotational angle, it can lower the tip of a pipe which allows water to flow through. The servo motor has 3 pins: VCC, ground and signal. The VCC pin is connected to 5V output; the ground pin is connected to ground; the signal pin is connected to pin 9, which is a PWM pin. The library PWMServo is used to control the motor.

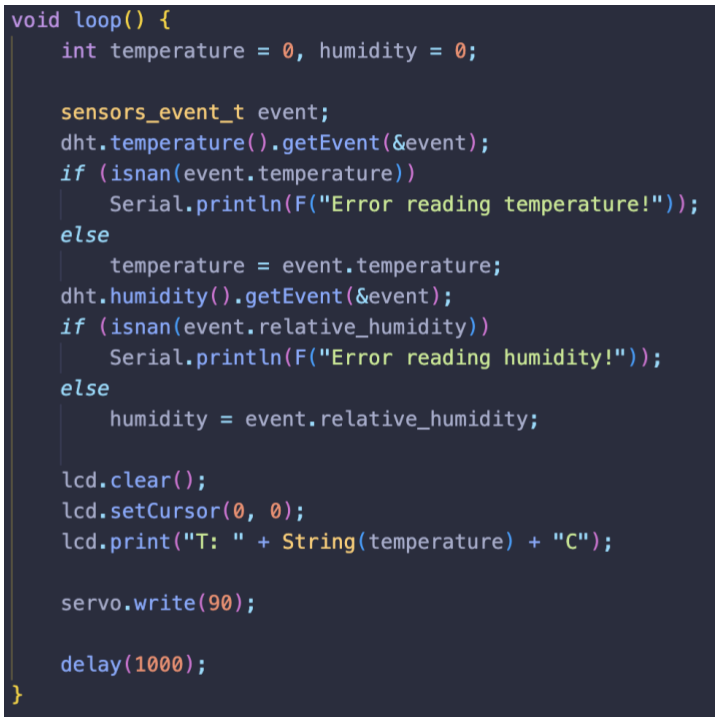
3.3 Software/Libraries:

- Additional libraries used for controlling sensors and actuators are Adafruit DHT sensor, LiquidCrystal_I2C and PWMServo.
  Setup:



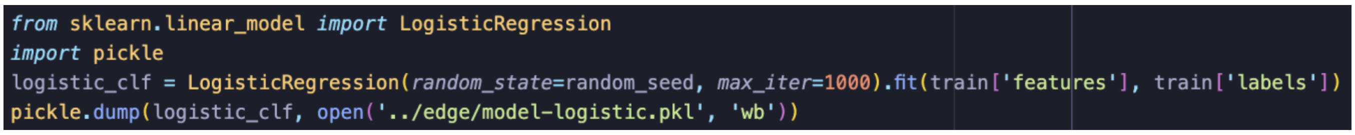
```
#include <PWMServo.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

int pinDHT11 = 3;
int pinServo = SERVO_PIN_A;

DHT_Unified dht(pinDHT11, DHT11);
PWMServo servo;
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
    dht.begin();

    servo.attach(pinServo);

    lcd.begin();
    lcd.backlight();
}
```

  Sample usage:



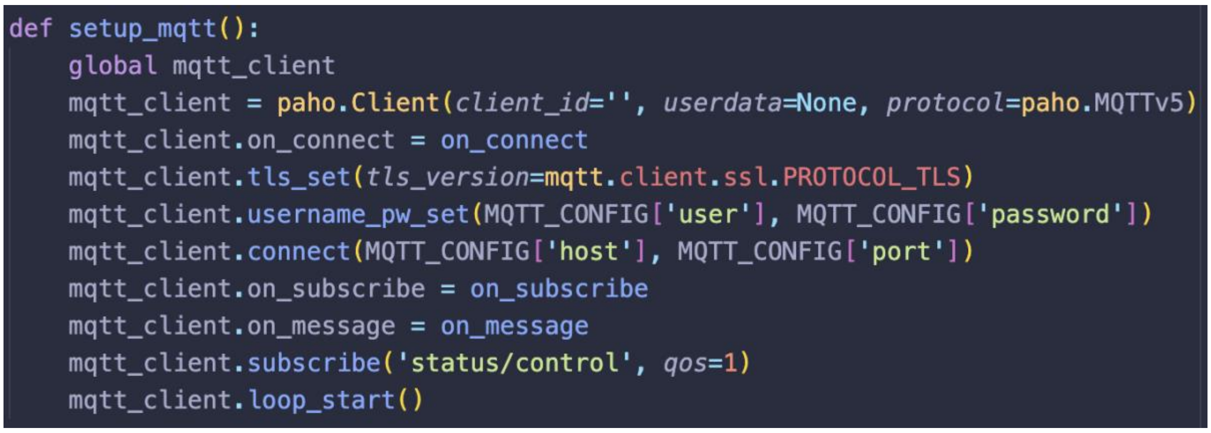
```
void loop() {
    int temperature = 0, humidity = 0;

    sensors_event_t event;
    dht.temperature().getEvent(&event);
    if (isnan(event.temperature))
        Serial.println(F("Error reading temperature!"));
    else
        temperature = event.temperature;
    dht.humidity().getEvent(&event);
    if (isnan(event.relative_humidity))
        Serial.println(F("Error reading humidity!"));
    else
        humidity = event.relative_humidity;

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("T: " + String(temperature) + "C");

    servo.write(90);

    delay(1000);
}
```

- To predict and provide irrigation decisions, a simple machine learning model is used. In specific, a logistic regression model is trained using scikit-learn, a popular python library for machine learning. The model takes the sensor measurements (soil moisture, temperature and humidity) as inputs and outputs a boolean value deciding whether to turn the irrigation system on or off. After being trained, the model is exported to be used by the edge device.

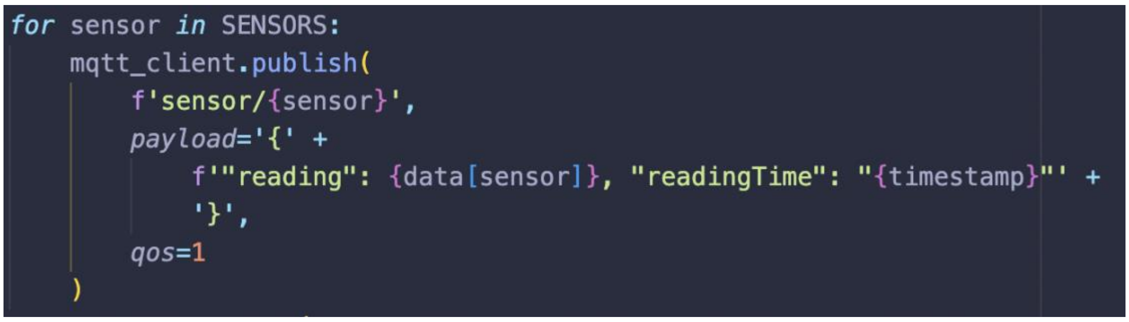


```python
from sklearn.linear_model import LogisticRegression
import pickle
logistic_clf = LogisticRegression(random_state=random_seed, max_iter=1000).fit(train['features'], train['labels'])
pickle.dump(logistic_clf, open('../edge/model-logistic.pkl', 'wb'))
```

- MQTT protocol is used to facilitate communication to and from the edge device. The broker is hosted on HiveMQ Cloud. Client libraries like paho (used on the edge device) and mqtt-react-hooks (used on the dashboard application) are used to subscribe and publish messages.



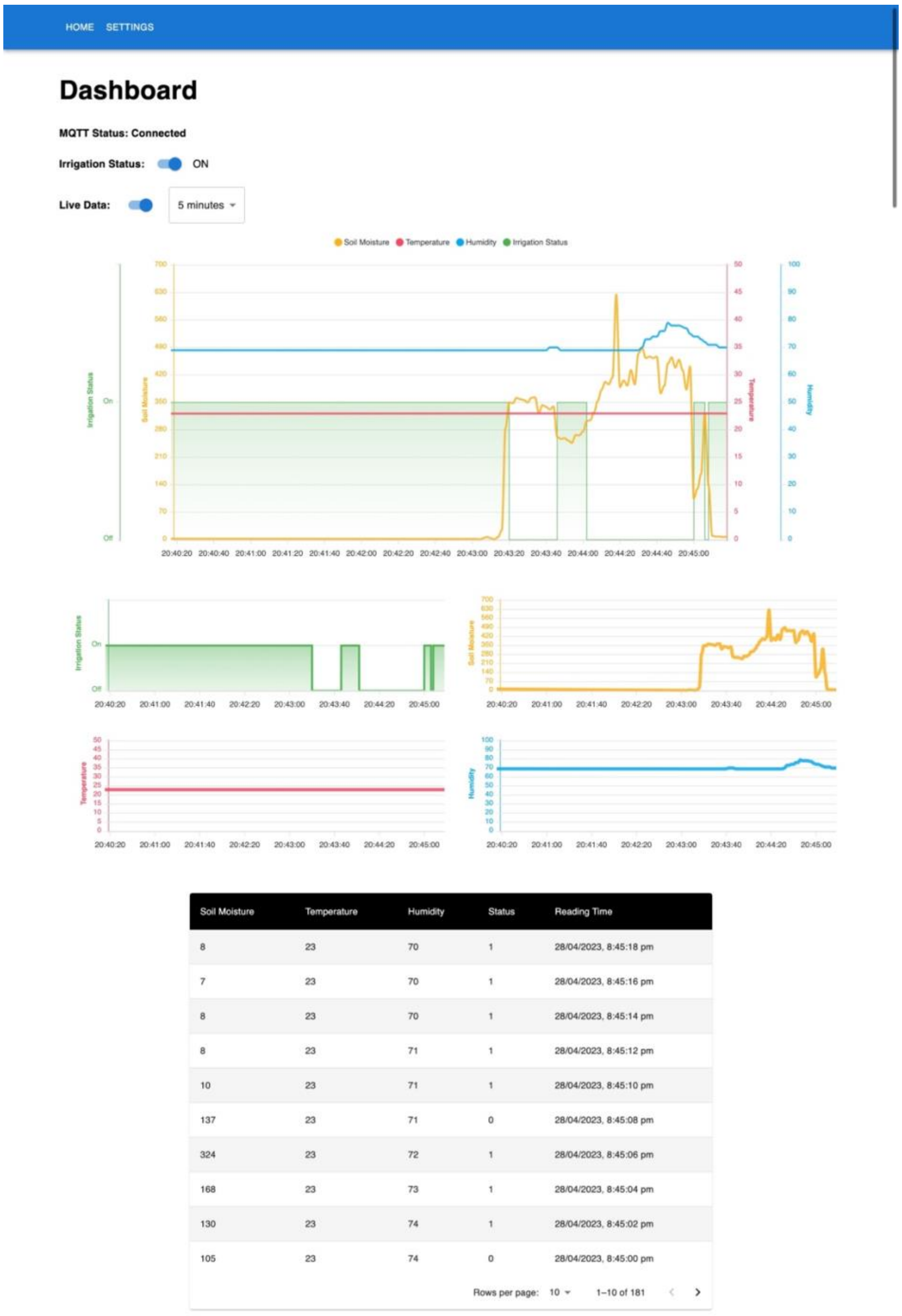
```python
def setup_mqtt():
    global mqtt_client
    mqtt_client = paho.Client(client_id='', userdata=None, protocol=paho.MQTTv5)
    mqtt_client.on_connect = on_connect
    mqtt_client.tls_set(tls_version=mqtt.client.ssl.PROTOCOL_TLS)
    mqtt_client.username_pw_set(MQTT_CONFIG['user'], MQTT_CONFIG['password'])
    mqtt_client.connect(MQTT_CONFIG['host'], MQTT_CONFIG['port'])
    mqtt_client.on_subscribe = on_subscribe
    mqtt_client.on_message = on_message
    mqtt_client.subscribe('status/control', qos=1)
    mqtt_client.loop_start()
```



```python
for sensor in SENSORS:
    mqtt_client.publish(
        f'sensor/{sensor}',
        payload='{' +
            f'"reading": {data[sensor]}, "readingTime": "{timestamp}"' +
            '}',
        qos=1
    )
```

3.4 Dashboard:

The dashboard is implemented using NextJs, Material UI and Apexchart. It utilises line charts, area charts and a table to visualise the data. Moreover, it has a simple form to change the irrigation rules.



Dashboard

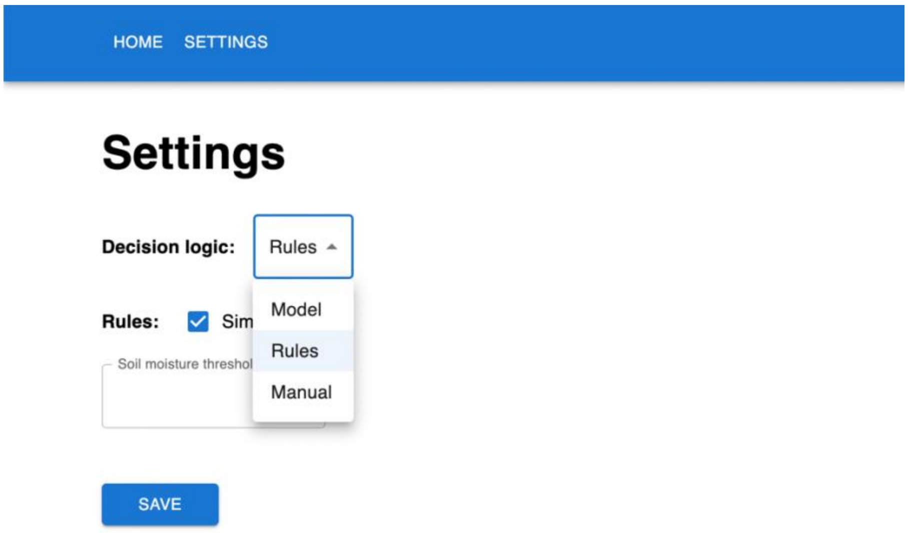
MQTT Status: Connected

Irrigation Status: ON

Live Data: 5 minutes

Soil Moisture	Temperature	Humidity	Status	Reading Time
8	23	70	1	28/04/2023, 8:45:18 pm
7	23	70	1	28/04/2023, 8:45:16 pm
8	23	70	1	28/04/2023, 8:45:14 pm
8	23	71	1	28/04/2023, 8:45:12 pm
10	23	71	1	28/04/2023, 8:45:10 pm
137	23	71	0	28/04/2023, 8:45:08 pm
324	23	72	1	28/04/2023, 8:45:06 pm
168	23	73	1	28/04/2023, 8:45:04 pm
130	23	74	1	28/04/2023, 8:45:02 pm
105	23	74	0	28/04/2023, 8:45:00 pm

Rows per page: 10 1–10 of 181



Settings

Decision logic: Rules

Rules: ✓ Sim

Soil moisture threshol

4. Resources

1. Agarwal, T. (2019, August 5). *DHT11 sensor definition, working and applications*. ElProCus - Electronic Projects for Engineering Students. https://www.elprocus.com/a-brief-on-dht11-sensor/
2. *Arduino—LCD I2C | Arduino tutorial*. (n.d.). Arduino Getting Started. Retrieved 28 April 2023, from https://arduinogetstarted.com/tutorials/arduino-lcd-i2c
3. *Arduino—Soil moisture sensor | Arduino tutorial*. (n.d.). Arduino Getting Started. Retrieved 28 April 2023, from https://arduinogetstarted.com/tutorials/arduino-soil-moisture-sensor
4. *DHT11, DHT22 and AM2302 sensors*. (n.d.). Adafruit Learning System. Retrieved 28 April 2023, from https://learn.adafruit.com/dht/using-a-dhtxx-sensor
5. *HiveMQ cloud: HiveMQ documentation*. (n.d.). Retrieved 28 April 2023, from https://docs.hivemq.com/hivemq-cloud/introduction.html#get-started
6. Nawandar, N. K., & Satpute, V. R. (2019). IoT based low cost and intelligent module for smart irrigation system. *Computers and Electronics in Agriculture*, *162*, 979–990. https://doi.org/10.1016/j.compag.2019.05.027
7. *Servo motor basics with Arduino | Arduino documentation*. (n.d.). Retrieved 28 April 2023, from https://docs.arduino.cc/learn/electronics/servo-motors
8. Vaishali, S., Suraj, S., Vignesh, G., Dhivya, S., & Udhayakumar, S. (2017). Mobile integrated smart irrigation management and monitoring system using IOT. *2017 International Conference on Communication and Signal Processing (ICCSP)*, 2164–2167. https://doi.org/10.1109/ICCSP.2017.8286792

9. Appendix

Scripts on edge devices:

- https://github.com/Legacy107/SWE30011-Smart-Irrigation/blob/main/edge/edge.py
- https://github.com/Legacy107/SWE30011-Smart-Irrigation/blob/main/backend/backend.py

Scripts on Arduino Uno:

- https://github.com/Legacy107/SWE30011-Smart-Irrigation/blob/main/main/main.ino

Frontend application:

- https://github.com/Legacy107/SWE30011-Smart-Irrigation/tree/main/frontend