MSA UNIVERSITY
جامعة أكتوبر للعلوم الحديثة والآداب

UNIVERSITY of GREENWICH

CSE Department – Faculty of Engineering - MSA
Fall 2024
CSE4531 IOT
Course Project
Course Instructor:Dr. Ehab Awad

Due Date 20/December/2024 11:59 PM on E-learning

Discussion inside lecture 21/December till 26/December inside lab as per lab slot

Student Name	**Abdelrahman Galal Ashour**	**Student ID**	**224249**
Student Name	**Abdelrahman Abubakr**	**Student ID**	**222235**
TA Name	**Eng. Hussein Mostafa**	**Grade:**	**/**

Project Title:

IoT-Based Automated Water Irrigation System using MQTT Protocol

Table of Contents

Project Overview

This project demonstrates the development of an IoT-based automated water irrigation system using MQTT for communication. The system monitors soil moisture levels, temperature, and humidity and automatically controls a water pump to optimize irrigation. It also displays real-time data on an I2C LCD and publishes system states to an MQTT topic. Simulation was carried out using PICSimLab, while real-time data transmission was achieved using the HiveMQ public broker.

Objectives

- To create a smart irrigation system to conserve water and optimize plant growth.
- To utilize MQTT for real-time monitoring and control.
- To integrate an I2C LCD for real-time data display.
- To implement and test the system in a simulated environment using PICSimLab.

Roles and Responsibilities

Abdelrahman Galal Ashour: Designed and implemented the IoT system and MQTT communication.

Abdelrahman Abubakr: Developed the simulation model in PICSimLab and tested the system.

Algorithm and external libraries

1. Initialize Ethernet and MQTT communication.
2. Continuously monitor soil moisture, temperature, and humidity using sensors.
3. Display real-time data on an I2C LCD.
4. Determine irrigation status based on predefined conditions:
   - Soil moisture < 30%, humidity < 60%, and temperature > 25°C: Turn on irrigation.
   - Soil moisture >= 30%, humidity >= 60% or temperature <= 25°C: Turn off irrigation.
5. Publish the system state to the MQTT topic /PLANT.

External Libraries Used:

- Ethernet.h: Manages the Ethernet connection for the Arduino.
- PubSubClient.h: Enables MQTT communication.
- DHT.h: Reads data from the DHT11 sensor.
- LiquidCrystal_I2C.h: Controls the I2C LCD display.

.

Code explaining

The updated code integrates an I2C LCD to display real-time data and incorporates new conditions for irrigation control based on soil moisture, temperature, and humidity.

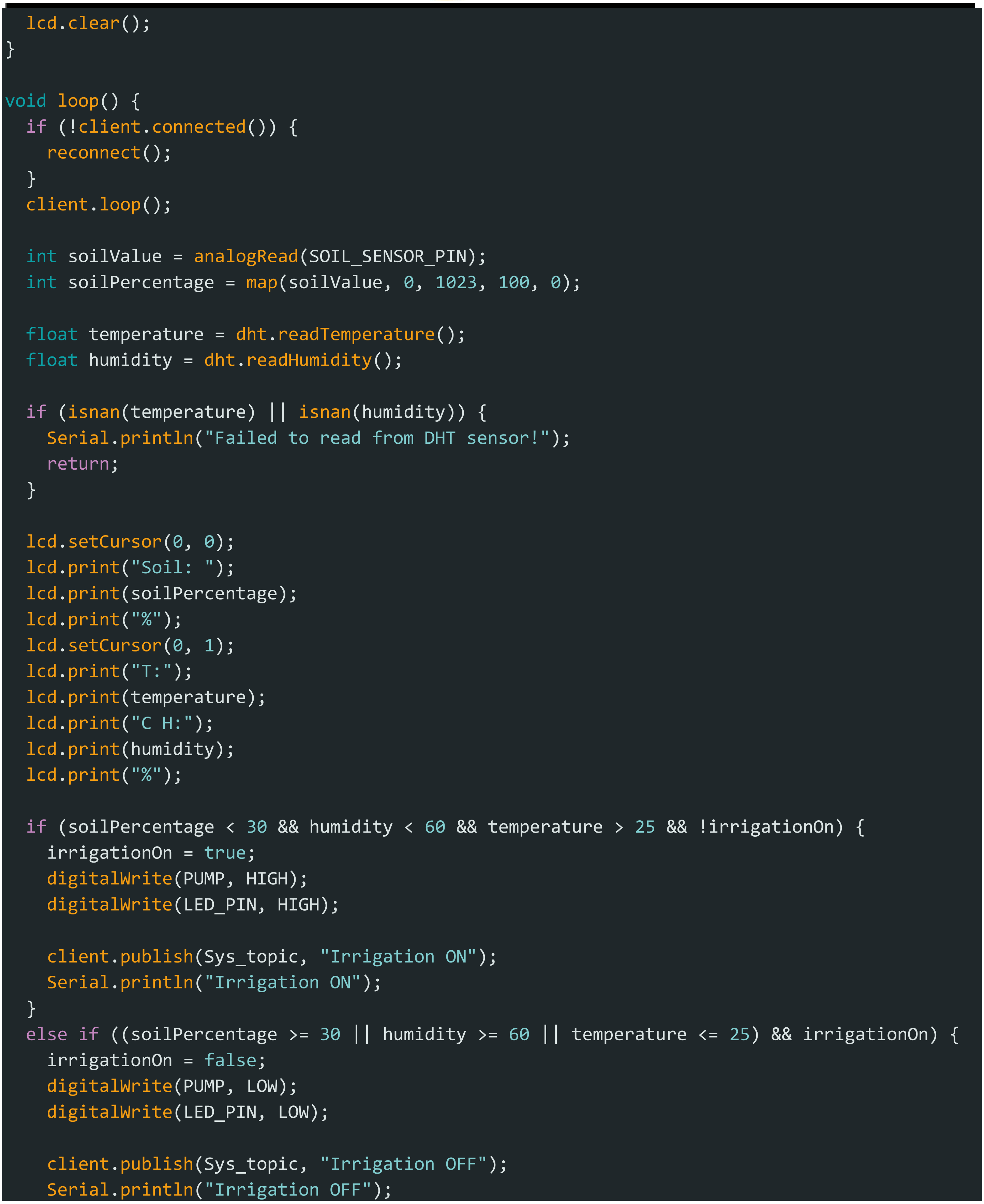
Key Features:

1. **Sensor Integration:**
    - The DHT11 sensor measures temperature and humidity.
    - A soil moisture sensor calculates the moisture percentage.
2. **Display:**
    - An I2C LCD shows soil moisture, temperature, and humidity values in real-time.
3. **Irrigation Logic:**
    - Irrigation is turned on when soil moisture is low (<30%), humidity is low (<60%), and temperature is high (>25°C).
    - Irrigation is turned off when any of the following conditions are met:
        - Soil moisture >= 30%.
        - Humidity >= 60%.
        - Temperature <= 25°C.
4. **MQTT Communication:**
    - The system publishes "Irrigation ON" or "Irrigation OFF" to the MQTT topic /PLANT based on the irrigation state.

The code of PICSIMLAB for Simulation

```
 #include <Ethernet.h>
#include <PubSubClient.h>
#include <DHT.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress ip(192, 168, 1, 177);

const char* mqtt_server = "broker.hivemq.com";
const int mqtt_port = 1883;
const char* Sys_topic = "/PLANT";

EthernetClient ethClient;
```

```
PubSubClient client(ethClient);

#define SOIL_SENSOR_PIN A0
#define LED_PIN 5
#define PUMP 6
#define DHT_PIN 7
#define DHT_TYPE DHT11

LiquidCrystal_I2C lcd(0x27, 16, 2);

DHT dht(DHT_PIN, DHT_TYPE);

bool irrigationOn = false;

void reconnect() {
  while (!client.connected()) {
    Serial.print("Connecting to MQTT...");
    if (client.connect("ArduinoClient")) {
      Serial.println("Connected!");
    } else {
      Serial.print("Failed, rc=");
      Serial.print(client.state());
      Serial.println(". Retrying in 5 seconds...");
      delay(5000);
    }
  }
}

void setup() {
  Serial.begin(9600);

  Ethernet.begin(mac, ip);
  client.setServer(mqtt_server, mqtt_port);

  dht.begin();

  lcd.init();
  lcd.backlight();

  pinMode(LED_PIN, OUTPUT);
  pinMode(PUMP, OUTPUT);
  digitalWrite(LED_PIN, LOW);
  digitalWrite(PUMP, LOW);

  lcd.print("System Ready");
  delay(2000);
```

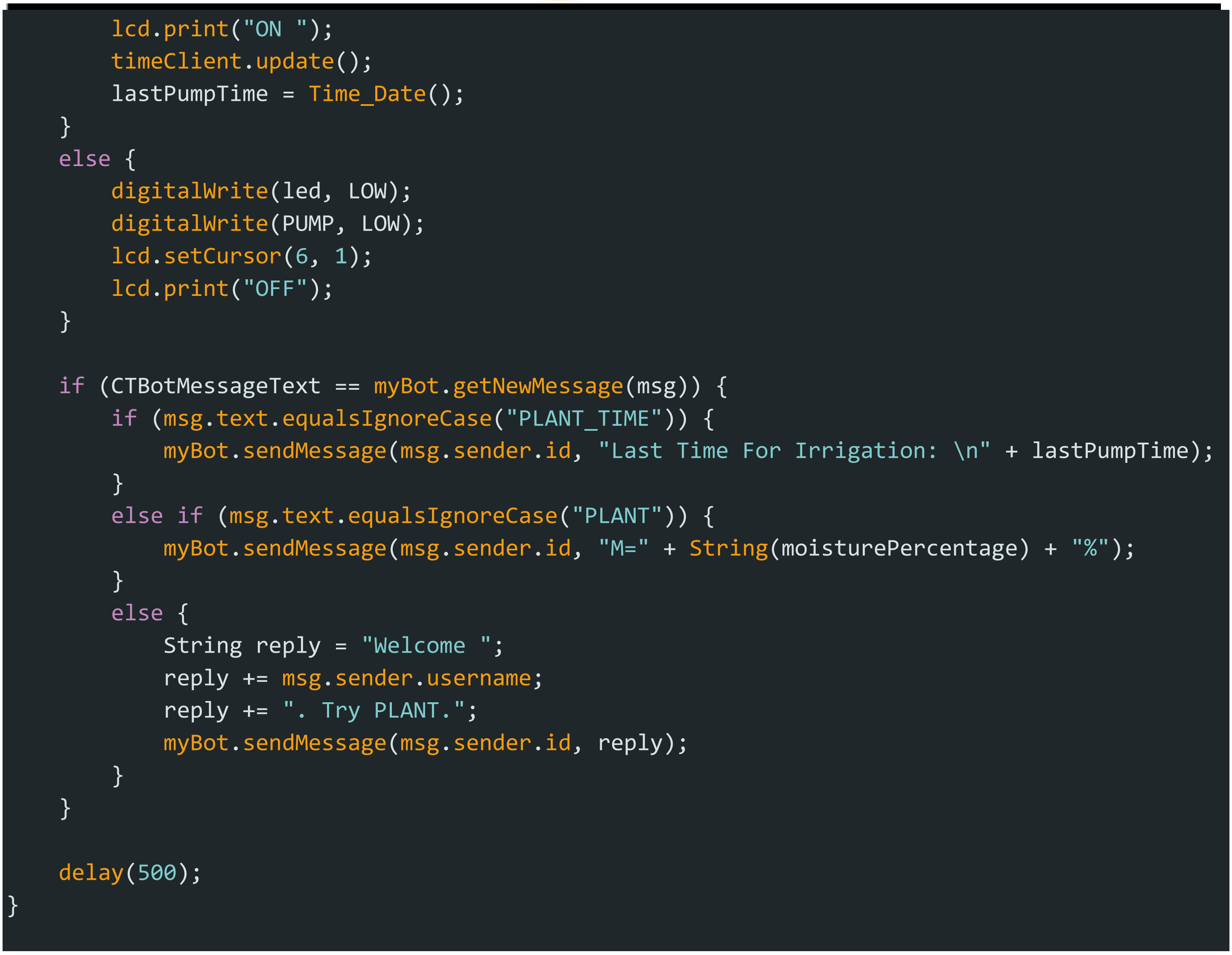


```
  lcd.clear();
}

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

  int soilValue = analogRead(SOIL_SENSOR_PIN);
  int soilPercentage = map(soilValue, 0, 1023, 100, 0);

  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  lcd.setCursor(0, 0);
  lcd.print("Soil: ");
  lcd.print(soilPercentage);
  lcd.print("%");
  lcd.setCursor(0, 1);
  lcd.print("T:");
  lcd.print(temperature);
  lcd.print("C H:");
  lcd.print(humidity);
  lcd.print("%");

  if (soilPercentage < 30 && humidity < 60 && temperature > 25 && !irrigationOn) {
    irrigationOn = true;
    digitalWrite(PUMP, HIGH);
    digitalWrite(LED_PIN, HIGH);

    client.publish(Sys_topic, "Irrigation ON");
    Serial.println("Irrigation ON");
  }
  else if ((soilPercentage >= 30 || humidity >= 60 || temperature <= 25) && irrigationOn) {
    irrigationOn = false;
    digitalWrite(PUMP, LOW);
    digitalWrite(LED_PIN, LOW);

    client.publish(Sys_topic, "Irrigation OFF");
    Serial.println("Irrigation OFF");
  }

  delay(2000);
}
```

The Code Of Real Project

The Code of real project

```
#include <ESP8266WiFi.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include "CTBot.h"
#include <WiFiUdp.h>
#include <NTPClient.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);
WiFiUDP ntpUDP;
#define offset 10800
NTPClient timeClient(ntpUDP, "pool.ntp.org");

CTBot myBot;

const char* ssid = "Abody-IPhone";
const char* pass = "abody1234";
const char* token = "7036076213:AAFUFlBXpb4XpZ8_MHXVdZ9tJkR-lVc_Mss";
const uint8_t led = D7, PUMP = D8;
const int soilSensorPin = A0;
const int dryThreshold = 561;

String lastPumpTime = "Never";

void setup() {
    lcd.init();
    lcd.backlight();
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Starting TeleBot...");

    myBot.wifiConnect(ssid, pass);
    WiFi.begin(ssid, pass);
    timeClient.begin();
    timeClient.setTimeOffset(offset);

    myBot.setTelegramToken(token);

    if (myBot.testConnection())
    {
        lcd.clear();
        lcd.print("\ntestConnection OK");
    }
    else
    {
        lcd.clear();
        lcd.print("\ntestConnection NOK");
    }

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Irrigation systm");
    lcd.setCursor(0, 1);
    lcd.print("Pump=");
    lcd.setCursor(10, 1);
    lcd.print("M= ");
    pinMode(led, OUTPUT);
    pinMode(PUMP, OUTPUT);
    timeClient.update();
}

String Time_Date() {
    time_t epochTime = timeClient.getEpochTime();
    struct tm *ptm = gmtime(&epochTime);

    char currentDate[20];
    sprintf(currentDate, "DATE:%02d/%02d/%04d TIME:%02d:%02d:%02d",
            ptm->tm_mday, ptm->tm_mon + 1, ptm->tm_year + 1900,
            ptm->tm_hour, ptm->tm_min, ptm->tm_sec);
    return String(currentDate);
}

void loop() {
    TBMessage msg;
    int soilMoisture = analogRead(soilSensorPin);
    int moisturePercentage = map(soilMoisture, 697, 292, 0, 100);

    lcd.setCursor(10, 1);
    lcd.print("M= ");
    lcd.setCursor(12, 1);
    lcd.print(moisturePercentage);
    lcd.print("%   ");

    if (soilMoisture >= dryThreshold) {
        digitalWrite(led, HIGH);
        digitalWrite(PUMP, HIGH);
        lcd.setCursor(6, 1);
```

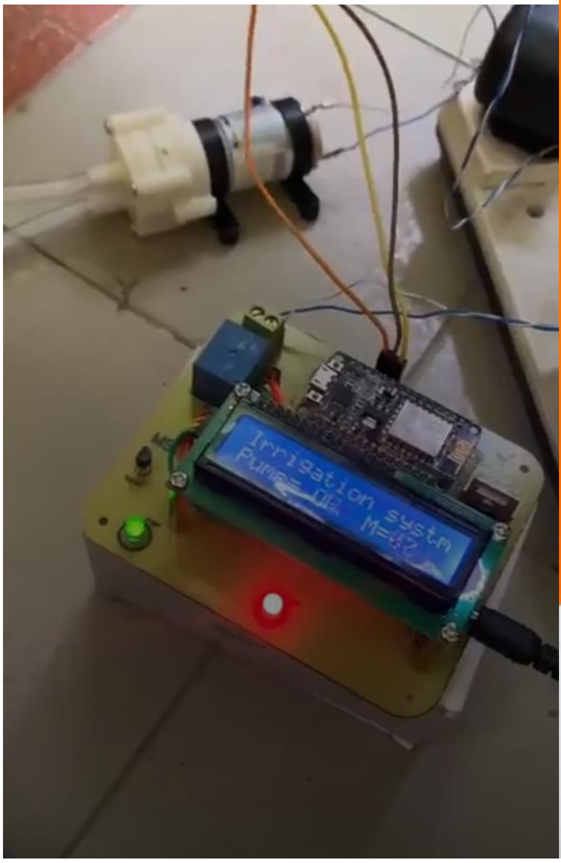


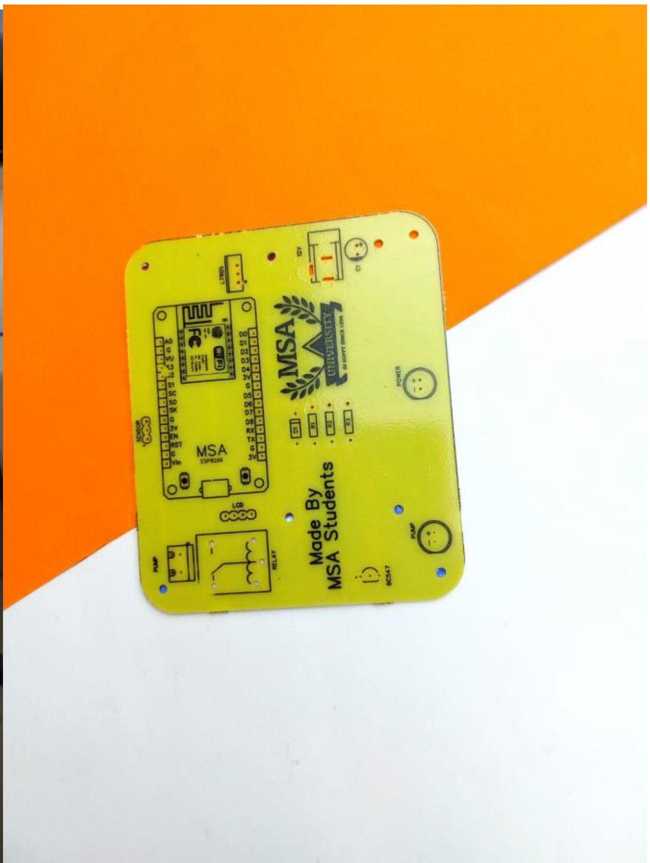
```
        lcd.print("ON ");
        timeClient.update();
        lastPumpTime = Time_Date();
    }
    else {
        digitalWrite(led, LOW);
        digitalWrite(PUMP, LOW);
        lcd.setCursor(6, 1);
        lcd.print("OFF");
    }

    if (CTBotMessageText == myBot.getNewMessage(msg)) {
        if (msg.text.equalsIgnoreCase("PLANT_TIME")) {
            myBot.sendMessage(msg.sender.id, "Last Time For Irrigation: \n" + lastPumpTime);
        }
        else if (msg.text.equalsIgnoreCase("PLANT")) {
            myBot.sendMessage(msg.sender.id, "M=" + String(moisturePercentage) + "%");
        }
        else {
            String reply = "Welcome ";
            reply += msg.sender.username;
            reply += ". Try PLANT.";
            myBot.sendMessage(msg.sender.id, reply);
        }
    }

    delay(500);
}
```

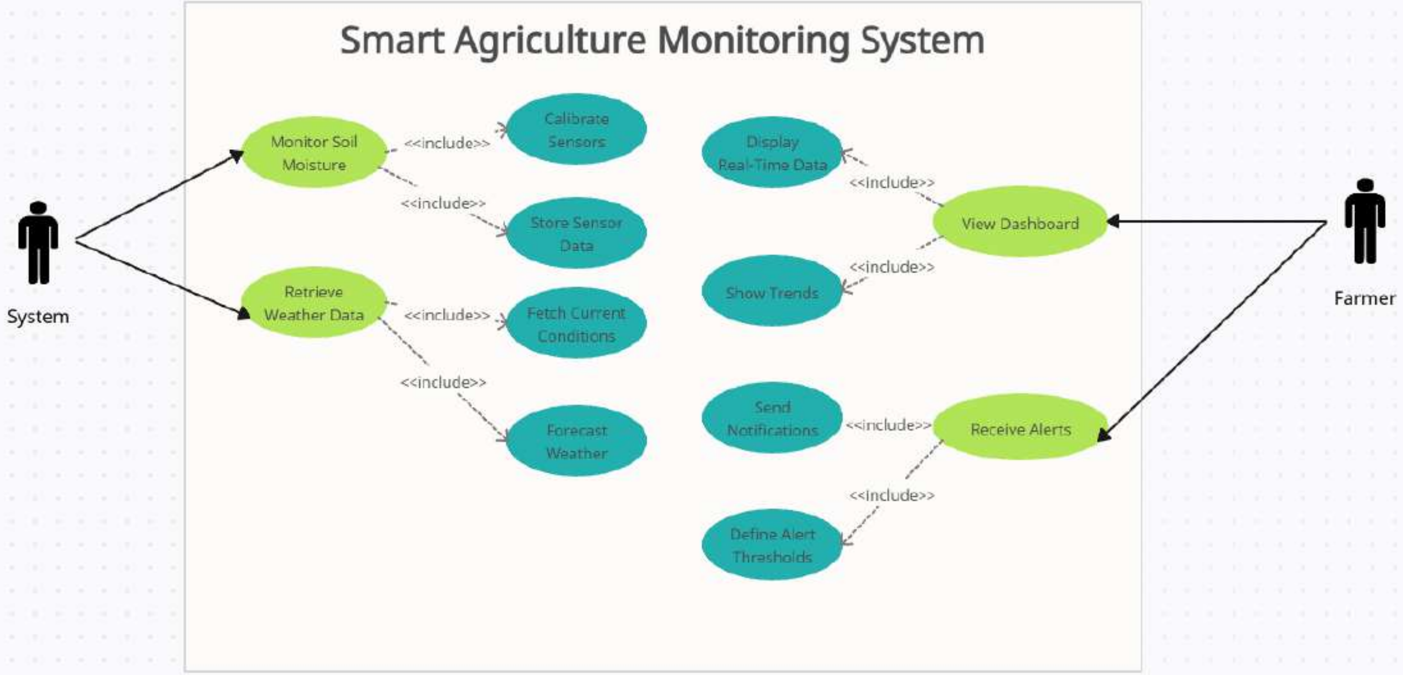
The video of the project

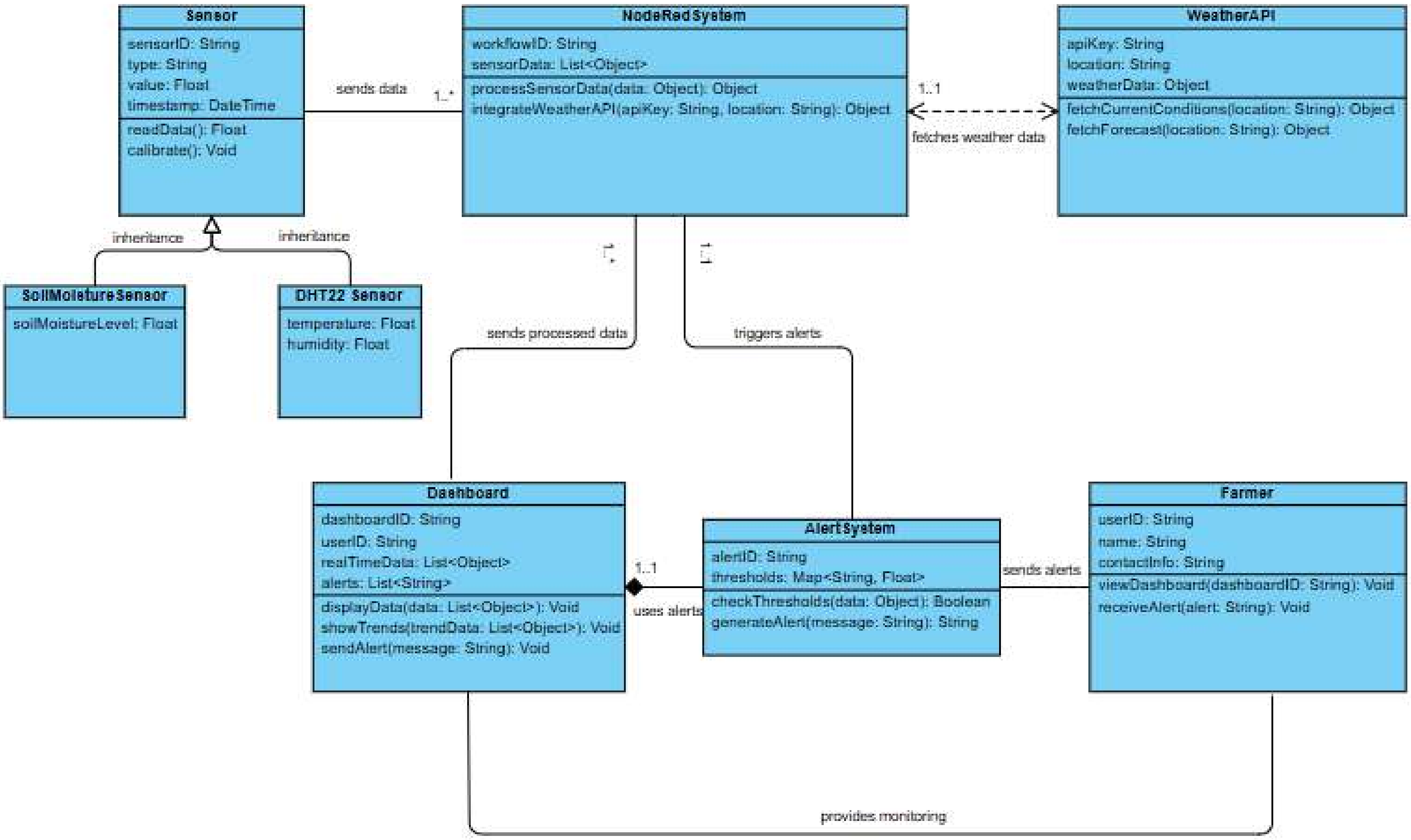
https://drive.google.com/file/d/1nZZ5jvwr1VnVj0CzuMdYJhyTJop9Koc2/view

some photos of the project



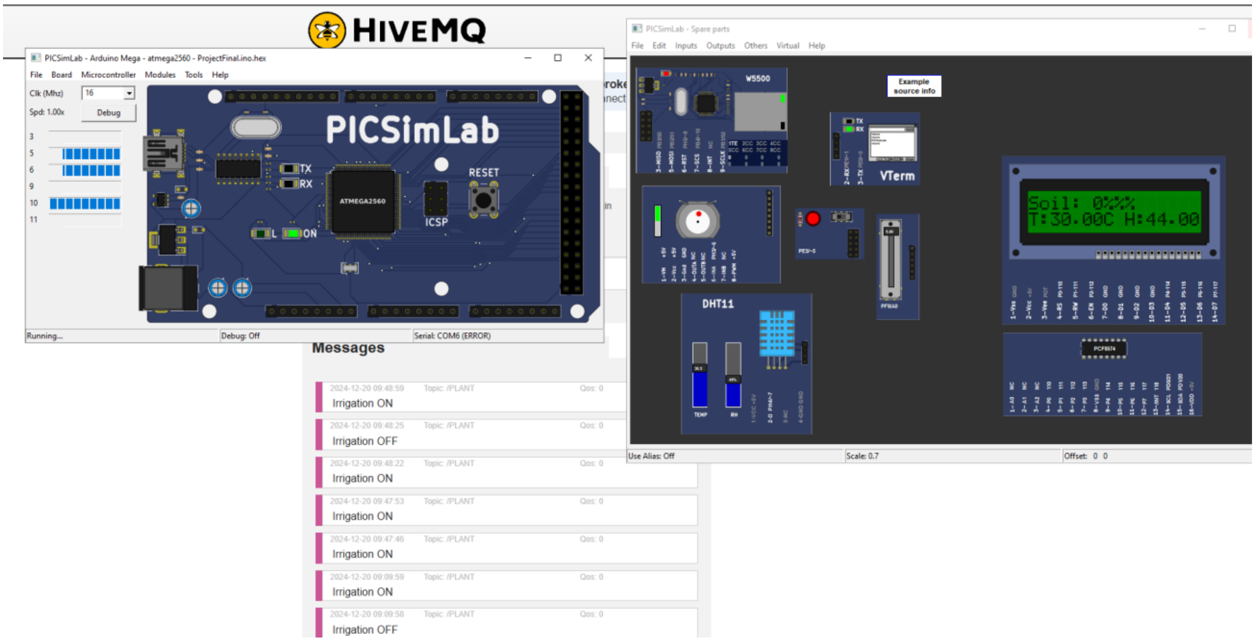



Smart Agriculture Monitoring System
System
Farmer
Monitor Soil Moisture
Retrieve Weather Data
Calibrate Sensors
Store Sensor Data
Fetch Current Conditions
Forecast Weather
Display Real-Time Data
Show Trends
View Dashboard
Send Notifications
Define Alert Thresholds
Receive Alerts
<<include>>


Sensor
sensorID: String
type: String
value: Float
timestamp: DateTime
readData(): Float
calibrate(): Void
sends data
1..*
NodeRedSystem
workflowID: String
sensorData: List<Object>
processSensorData(data: Object): Object
integrateWeatherAPI(apiKey: String, location: String): Object
1..1
fetches weather data
WeatherAPI
apiKey: String
location: String
weatherData: Object
fetchCurrentConditions(location: String): Object
fetchForecast(location: String): Object
inheritance
inheritance
SoilMoistureSensor
soilMoistureLevel: Float
DHT22 Sensor
temperature: Float
humidity: Float
1..*
1..1
sends processed data
triggers alerts
Dashboard
dashboardID: String
userID: String
realTimeData: List<Object>
alerts: List<String>
displayData(data: List<Object>): Void
showTrends(trendData: List<Object>): Void
sendAlert(message: String): Void
1..1
uses alerts
AlertSystem
alertID: String
thresholds: Map<String, Float>
checkThresholds(data: Object): Boolean
generateAlert(message: String): String
sends alerts
Farmer
userID: String
name: String
contactInfo: String
viewDashboard(dashboardID: String): Void
receiveAlert(alert: String): Void
provides monitoring

Output and results

1. When soil moisture is below 30%, humidity is below 60%, and temperature is above 25°C:
   - The pump and LED turn on.
   - The MQTT broker receives the message "Irrigation ON".
   - The LCD displays real-time values of soil moisture, temperature, and humidity.
2. When soil moisture is >= 30%, humidity is >= 60%, or temperature is <= 25°C:
   - The pump and LED turn off.
   - The MQTT broker receives the message "Irrigation OFF".
   - The LCD continues to display real-time values.



Serial: COM6 (ERROR)

```
Soil: 0%%
T:30.00C H:44.00
```

Messages

```
2024-12-20 09:48:59   Topic: /PLANT   Qos: 0
Irrigation ON
2024-12-20 09:48:25   Topic: /PLANT   Qos: 0
Irrigation OFF
2024-12-20 09:48:22   Topic: /PLANT   Qos: 0
Irrigation ON
2024-12-20 09:47:53   Topic: /PLANT   Qos: 0
Irrigation ON
2024-12-20 09:47:46   Topic: /PLANT   Qos: 0
Irrigation ON
2024-12-20 09:09:59   Topic: /PLANT   Qos: 0
Irrigation ON
2024-12-20 09:09:58   Topic: /PLANT   Qos: 0
Irrigation OFF
```

The Potentiometer here acts as a soil sensor but we cannot add it in PICSIMLAB unfortunately.

References

1. Arduino Documentation: Ethernet and PubSubClient libraries.
2. https://www.instructables.com/How-to-Make-Automatic-Irrigation-System-Using-Ardu/
3. MQTT Websocket Client
4. PICSimLab User Guide
5. DHT Sensor Documentation: https://learn.adafruit.com/dht.