Assignment 7 - report

The code displays a timer counting seconds and the soil moisture value from a sensor on an LCD. Every second, it updates the time on the LCD and reads the moisture level, displaying it on both the LCD and Serial Monitor. The timer increments by 1 second, and the moisture value is printed in real-time.

The result can be recorded in the video (Uploaded in the same folder) and can be reported as:

Time	Moisture level (.../1023)
5	945
26	930
60	923
70	914
94	903

112	880
130	866
140	833
166	801
220	755
240	740
277	719
302	709
365	704
588	702
632	695
646	700
669	709

Summary

- Initial Moisture Level: 945 (at 5 seconds)
- Final Moisture Level: 709 (at 669 seconds)
- The moisture level decreases gradually over time, indicating water absorption by the soil.

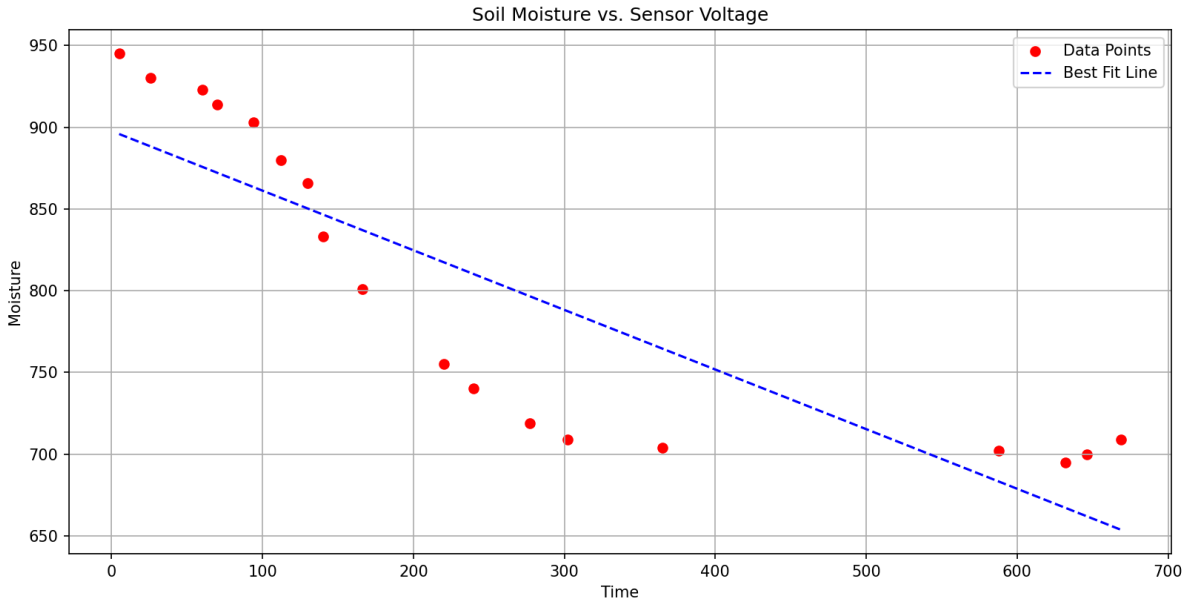
Soil Moisture Decrease Analysis

Time (s)	Soil Moisture (.../1023)	Decrease in Moisture (Previous - Current)
5 (Start of pump)	945	0
26	930	15
60	923	7
70	914	9
94	903	11
112	880	23

130	866	14
140	833	33
166	801	32
220	755	46
240	740	15
277	719	21
302	709	10
365	704	5
588	702	2
632	695	7
646	700	-5
669	709	9

Total increase in moisture = (first moisture value - balance moisture value)/ total time = (945 - 709) / 302 = **0.7814** (moisture value per second)

So, **moisture increased by 236 units in roughly 300 second**. This slow change in value of moisture increase could be explain by the small amount of water being added compare to the amount of soil. The second reason would come from the dryness of the soil which is harder to absorb water (In regions experiencing drought, the soil often becomes very dry and compacted, making it harder for water to penetrate. When rain comes after a long dry period, the water tends to run off the surface rather than being absorbed by the soil. This can lead to flooding)



=== > At the time of 300 and onwards, the value of moisture balance out.

#Assignment Code

```
#include <Arduino.h>
#include <DHT.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);  // Set the LCD address (0x27) for a 16 chars a

#define M_PIN A1  // Define the moisture sensor pin

unsigned long previousMillis = 0;  // Variable to store last time the display was up
int secondCount = 0;  // Variable to count seconds

void setup() {
   Serial.begin(9600);  // Start the Serial communication
```

```
    pinMode(M_PIN, INPUT);  // Set the moisture pin as input to read analog data
    lcd.begin();  // Initialize the LCD
    lcd.backlight();  // Turn on the LCD backlight
    lcd.setCursor(0, 0);  // Set the cursor to the first position
    lcd.print("Time: 0s");  // Display initial time
}

void loop() {
    unsigned long currentMillis = millis();  // Get the current time in milliseconds

    // Check if 1 second has passed
    if (currentMillis - previousMillis >= 1000) {
        // Save the last time the display was updated
        previousMillis = currentMillis;

        // Increment the second count
        secondCount++;

        // Update the display with the new time
        lcd.clear();  // Clear the previous time
        lcd.setCursor(0, 0);  // Set cursor to the first line
        lcd.print("Time: " + String(secondCount) + "s");  // Display the time in secon
    }

    // Read moisture value
    int moisture = analogRead(M_PIN);

    // Print moisture value to Serial Monitor
    String displayText = "MOISTURE: " + String(moisture);
    Serial.println(displayText);

    // Display the moisture value on LCD (below the time counter)
    lcd.setCursor(0, 1);  // Set cursor to the second line
    lcd.print(displayText);
```

```
    delay(1000);  // Wait briefly before the next loop iteration
}
```

#Code for the graph above using matplotlib (reused from the lesson 6)

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import linregress

# Sample data: Voltage (sensor readings) and Gravimetric Soil Moisture (%)
voltage_values = [5, 26, 60, 70, 94, 112, 130, 140, 166, 220, 240, 277, 302, 365, 58
moisture_values = [945, 930, 923, 914, 903, 880, 866, 833, 801, 755, 740, 719, 70

# 22 - 12 = 10 / 12 = 0.833  | 13 - 8 = 5 / 8 = 62,5 %  | 11 - 8 = 3 / 8 = 37,5 %

# Perform linear regression to find the best-fit line
slope, intercept, r_value, p_value, std_err = linregress(voltage_values, moisture_va
print(f"Equation: y = {slope:.4f}x + {intercept:.4f}")
# Generate x values for the line
x_values = np.linspace(min(voltage_values), max(voltage_values), 100)
y_values = slope * x_values + intercept

# Plot the data points
plt.scatter(voltage_values, moisture_values, color='red', label='Data Points')
plt.plot(x_values, y_values, linestyle='--', color='blue', label='Best Fit Line')

# Labels and title
plt.xlabel( 'Time'  )#'Sensor Voltage (0-1023)')
plt.ylabel( 'Moisture' )#'Gravimetric Soil Moisture (%)')
plt.title('Soil Moisture vs. Sensor Voltage')
plt.legend()
plt.grid()
```

```
# Show the plot
plt.show()
```

#Code for soil-moisture-sensor-server/app.py

```
import json
import time
import paho.mqtt.client as mqtt
import threading

id = "2fdac31a-b52e-4ea4-a2f6-4e6629111b7e"

client_telemetry_topic = id + "/telemetry"
server_command_topic = id + "/command"
client_name = id + "soilmoisturesensor_client"
water_time = 5
wait_time = 20

def send_relay_command(client, state):
    command = { 'relay_on' : state }
    print("Sending message:", command)
    client.publish(server_command_topic, json.dumps(command))

def control_relay(client):
    print("Unsubscribing from telemetry")
    mqtt_client.unsubscribe(client_telemetry_topic)

    send_relay_command(client, True)
    time.sleep(water_time)
    send_relay_command(client, False)

    time.sleep(wait_time)

    print("Subscribing to telemetry")
```

```
    mqtt_client.subscribe(client_telemetry_topic)

# Callback function when a message is received
def handle_telemetry(client, userdata, message):
    payload = json.loads(message.payload.decode())
    print("Message received:", payload)

    if payload['soil_moisture'] > 450:
        threading.Thread(target=control_relay, args=(client,)).start()

# Callback function when connected to the broker
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected to MQTT Broker!")
        client.subscribe(client_telemetry_topic)
    else:
        print(f"Failed to connect, return code {rc}")

# Setup MQTT client
mqtt_client = mqtt.Client(client_name)
mqtt_client.on_connect = on_connect
mqtt_client.on_message = handle_telemetry

# Connect to the MQTT broker
try:
    mqtt_client.connect("test.mosquitto.org", 1883, 60)
except Exception as e:
    print("Failed to connect to broker:", e)
    exit(1)


# Keep the connection alive and process messages
mqtt_client.loop_forever()
```

```
(env) PS C:\Users\ADMIN\OneDrive\Documents\PlatformIO\Projects\HelloArduino\soil-moisture
-sensor-server> python app.py
Connected to MQTT Broker!
```

=⇒ Server is on and ready to subcribe and publish

#Code for controlling relay

```
#include <Arduino.h>

#define RELAY_PIN 7
void setup() {
pinMode(A0, INPUT);
pinMode(RELAY_PIN, OUTPUT);

}

void loop() {

int soil_moisture = analogRead(A0);
Serial.print("Soil Moisture: ");
Serial.println(soil_moisture);
if (soil_moisture > 450)
{
```

```
Serial.println("Soil Moisture is too low, turning relay on.");
digitalWrite(RELAY_PIN, HIGH);
}
else
{
Serial.println("Soil Moisture is ok, turning relay off.");
digitalWrite(RELAY_PIN, LOW);
}

delay(10000);

}
```