IOT PLANT MONITORING SYSTEM

ECE 3502 – IOT DOMAIN ANALYST

J COMPONENT FINAL REPORT

Submitted to,

Dr. Sitharthan R

School of Electrical Engineering,

VIT University, Vellore,

By,

Saran DB (20BEE0130)

Rithami R (20BEE0148)

Praharshitha K (20BEE0331)

Gurijala Loknath Sai (20BEE0392)

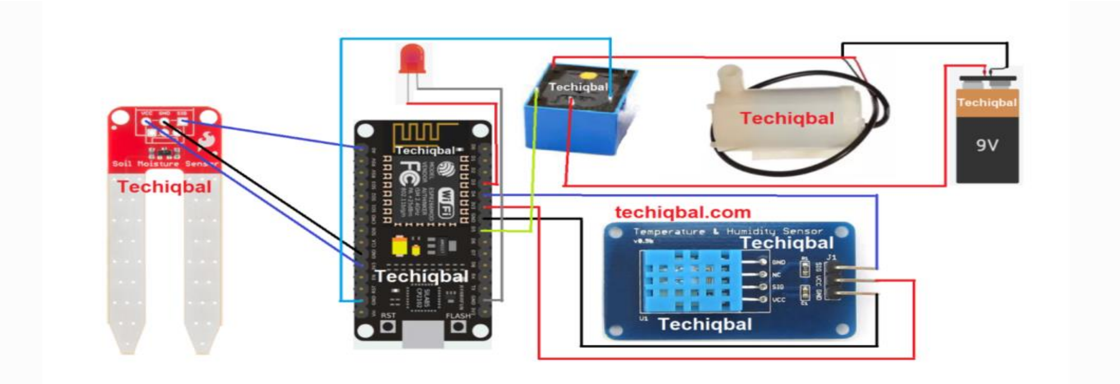
Problem Statement

IOT based plant monitoring systems address the existing problem of manual monitoring and maintenance of plants in agricultural and horticultural settings. Traditional methods of monitoring plant growth and health rely on manual inspections and record keeping, which can be time-consuming, labor-intensive, and prone to human error. Additionally, manual monitoring often doesn't provide real-time data, making it difficult to quickly identify and address issues as they arise. IOT based plant monitoring systems use sensors, devices, and software to automate and streamline the process, providing real-time data and alerts to help farmers and gardeners make informed decisions about their plants' care. This can lead to increased efficiency, reduced labor costs, and improved crop yields.

Apparatus Required

Sl. No	Equipment / Components	Range/ Type	Quantity
1	Esp8266	-	1
2	Arduino Uno	-	1
3	Soil moisture sensor	-	1
4	DHT 11 - temperature and humidity sensor	-	1
5	Jumper wires	-	1
6	breadboard	-	1
7	Solenoid valve	-	1
8	Relay	-	1
9	AC-DC Adaptor	-	1

Circuit Diagram



Objectives

1. Developing a system for monitoring various parameters of plants, such as temperature, humidity, soil moisture, using IoT sensors.

2. Creating a user-friendly interface for displaying real-time data and historical trends. This can be achieved using Blink IOT.

3. Automating the irrigation system based on the real-time data and historical trends.

4. Evaluating the effectiveness of the system in improving plant growth and health.

Procedure

The given prompt outlines the procedure for a smart irrigation system that utilizes temperature and moisture sensors controlled by an Arduino UNO and NodeMCU. The system collects data from these sensors and sends it through a regression model for future predictions. Such a system can be used in agricultural or gardening applications to automate the irrigation process and provide optimal moisture levels for plants. By using sensors to monitor temperature and moisture levels, the system can adjust the watering schedule as necessary, preventing overwatering or underwatering of plants. The use of regression models allows for data analysis and prediction, enabling the system to make adjustments in real-time based on historical data trends. Overall, this system provides a cost-effective and efficient solution for irrigation management.

Algorithms Used

For Arduino Code,

Algorithm to read soil moisture sensor reading and map it to values 0-100 in Arduino

Here's an algorithm to read a soil moisture sensor reading and map it to a value between 0 and 100 using an Arduino:

Connect the soil moisture sensor to the Arduino's analog input pin.

Define a variable to store the sensor reading.

Read the sensor value using the analogRead() function and store it in the variable.

Map the sensor reading to a value between 0 and 100 using the map() function. The map() function takes the following arguments:

The input value (the sensor reading)

The minimum input value (the minimum sensor reading)

The maximum input value (the maximum sensor reading)

The minimum output value (0)

The maximum output value (100)

Example: int moistureReading = analogRead(A0); int moistureValue = map(moistureReading, 0, 1023, 0, 100);

For temperature and humidity:

Connect the DHT11 sensor to the Arduino board. The DHT11 sensor has three pins: VCC, data, and GND. Connect the VCC pin to the 5V pin on the Arduino board, the data pin to a digital pin on the Arduino board (e.g., pin 2), and the GND pin to the GND pin on the Arduino board.

Declare the pin to which the DHT11 data pin is connected:

Declare the type of the DHT sensor you are using. In this case, we are using the DHT11 sensor:

Create an instance of the DHT class:

In the setup() function, initialize the serial communication and the DHT sensor

In the loop() function, read the temperature and humidity values from the DHT sensor

For ML Algorithm,

Read the CSV file and store the time, temperature, and humidity data in separate arrays.

Convert the time instances in the CSV file to a numerical format that can be used for linear regression. One way to do this is to convert each time instance to the number of seconds that has elapsed since the first time instance. To do this, you can follow the steps below:

a. Choose a reference time instance from your data. This can be the first time instance in your dataset.

b. Convert each time instance to a datetime object.

c. Calculate the time difference between each time instance and the reference time instance in seconds.

d. Store the time differences in a separate array.

Perform linear regression on the time and temperature data to create a linear regression model. You can use the time differences calculated in step 2 as the input for the linear regression model and the temperature data as the output.

Once you have created a linear regression model, you can use it to make future predictions. To do this, you can follow the steps below:

a. Choose future time instances at which you want to make temperature predictions.

b. Convert the future time instances to the same numerical format used in step 2.

c. Use the linear regression model to predict the temperature at each future time instance.

Print the predicted temperature values.

You should now have a linear regression model that can predict future temperature values based on the time instances in a CSV file.

Code

i) Arduino Code

```
#include <Arduino.h>
#define SOIL_MOISTURE_PIN A0
void setup() {
 Serial.begin(9600);
pinMode(SOIL_MOISTURE_PIN, INPUT);
 pinMode(2,OUTPUT);
 pinMode(5,OUTPUT);
 digitalWrite(5,HIGH);
```

```
}
void loop() {
int sensorValue = analogRead(SOIL_MOISTURE_PIN);
int moisturePercentage = map(sensorValue, 0, 1023, 0, 100);
Serial.print("Soil moisture: ");
Serial.print(moisturePercentage);
  Serial.println("%");
 delay(1000);
if(moisturePercentage<50) digitalWrite(2,HIGH);
 else digitalWrite(2,LOW);
}
```

ii) IOT Code

```
#define BLYNK_TEMPLATE_ID "TMPLGonwzQTu"
#define BLYNK_TEMPLATE_NAME "SMART IRRIGATION USING IOT"
#define BLYNK_AUTH_TOKEN "NHdtZMbO6407cHXzn0pNK67HGGVLfKaL"

/* Comment this out to disable prints and save space */
#define BLYNK_PRINT Serial
#include <DHT.h>


#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
//#include "ACS712.h
//ACS712 sensor(ACS712_30A,A0);
// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "OPPO K10 5G";
char pass[] = "123456789";

DHT dht(D2, DHT11);    // Initialize DHT sensor
void setup()
{
 Serial.begin(115200);
 Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
 dht.begin();
pinMode(4,OUTPUT);

pinMode(0,INPUT);

pinMode(2,INPUT);

pinMode(3,OUTPUT);

digitalWrite(3,HIGH);

}
void loop()
{
float soilMoistureMapped=moist();

 Blynk.virtualWrite(V5, soilMoistureMapped);              // Send mapped soil moisture value to Blynk
dashboard
```

```
delay(1000);   // Wait for 1 second before taking next   reading

 Blynk.run();   // Update Blynk dashboard

float temperature = dht.readTemperature();    // Read      temperature from DHT11 sensor

float humidity = dht.readHumidity();
Blynk.virtualWrite(V4, temperature);
Serial.println(temperature);
Blynk.virtualWrite(V2, humidity);
if(soilMoistureMapped<20)
 {
Blynk.virtualWrite(V0,1);
 digitalWrite(3,HIGH);
 }
if(soilMoistureMapped>20)
 {
Blynk.virtualWrite(V0,0);
 digitalWrite(3,HIGH);
 }
/*else
 {
Blynk.virtualWrite(V0,LOW);
digitalWrite(D3,LOW);
}*/
//Serial.println(V4);
 //Serial.println(V5);
}


float moist()
{
int sensorValue = analogRead(A0);
int moisturePercentage = map(sensorValue, 0, 1023, 100, 0);
return moisturePercentage;
}
```

iii) ML code

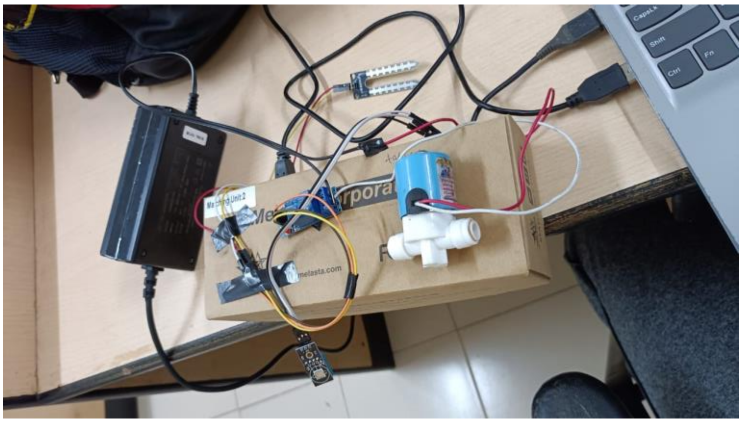
```
import csv
import re

# Open the Serial monitor data file
with open('temphum.txt', 'r') as f:
  # Read the lines from the file
lines = f.readlines()

# Define a regular expression pattern to match the expected format
pattern = r'^(\d+:\d+:\d+\.\d+) -> Temperature: ([\d\.]+) °C, Humidity: ([\d\.]+) %$'

# Create a new CSV file
with open('data.csv', 'w', newline='') as csvfile:
    # Create a CSV writer
    writer = csv.writer(csvfile)
```

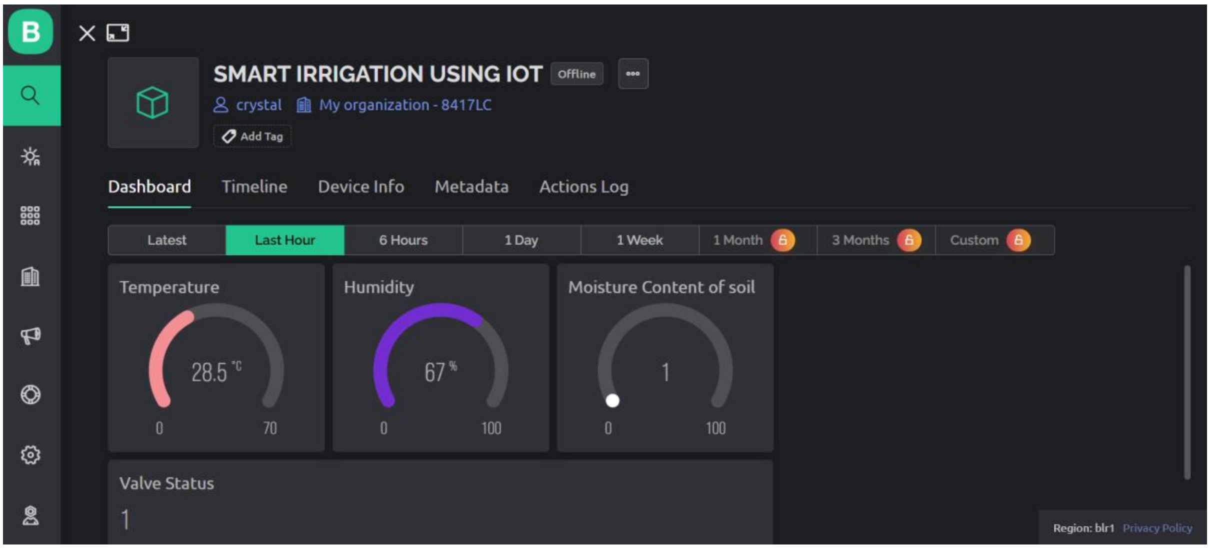
```
    # Write the headers to the CSV file
    writer.writerow(['Timestamp', 'Temperature', 'Humidity'])

    # Loop through the lines in the Serial monitor data
    for line in lines:
        # Use the regular expression to match the line
        match = re.match(pattern, line.strip())

        # Check if the line matches the expected format
        if match:
            # Extract the timestamp, temperature, and humidity from the match
            timestamp = match.group(1)
            temperature = match.group(2)
            humidity = match.group(3)

            # Write the data to the CSV file
            writer.writerow([timestamp, temperature, humidity])
        else:
            print('Line skipped:', line)

import pandas as pd
from sklearn.linear_model import LinearRegression

# Load the data from the CSV file into a pandas DataFrame
data = pd.read_csv('data.csv')

# Remove the Timestamp column from the DataFrame
data = data.drop(columns=['Timestamp'])

# Split the data into features (temperature and humidity) and target (None)
X = data[['Temperature', 'Humidity']]
y = None

# Create a LinearRegression model and fit it to the data
model = LinearRegression()
model.fit(X, y)

# Use the model to make predictions for future temperature and humidity values
future_data = [[25, 70], [30, 80], [35, 90]]
future_predictions = model.predict(future_data)

# Print the predicted temperature and humidity values for the future data
print(future_predictions)
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
from datetime import datetime, timedelta
```

```
# Load the data from the CSV file
data = pd.read_csv('data.csv')

# Convert the timestamp to datetime objects
data['Timestamp'] = pd.to_datetime(data['Timestamp'], format='%H:%M:%S.%f')

# Extract the hour and minute from the timestamp
data['Hour'] = data['Timestamp'].dt.hour
data['Minute'] = data['Timestamp'].dt.minute

# Split the data into features (hour and minute) and target (temperature)
X = data[['Hour', 'Minute']]
y = data['Temperature']

# Create a linear regression model
model = LinearRegression()

# Train the model
model.fit(X, y)

# Predict the temperature for the next 2 hours
current_time = data['Timestamp'].iloc[-1]
future_time = current_time + timedelta(hours=2)
future_hours = np.array([future_time.hour, future_time.minute]).reshape(1, -1)
future_temperature = model.predict(future_hours)

# Print the predicted temperature
print('Predicted temperature after 2 hours:', future_temperature)

# Plot the temperature over time
plt.plot(data['Timestamp'], data['Temperature'])
plt.xlabel('Time')
plt.ylabel('Temperature (°C)')
plt.title('Temperature over time')
plt.show()
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
from datetime import datetime, timedelta

# Load the data from the CSV file
data = pd.read_csv('data.csv')

# Convert the timestamp to datetime objects
data['Timestamp'] = pd.to_datetime(data['Timestamp'], format='%H:%M:%S.%f')
```

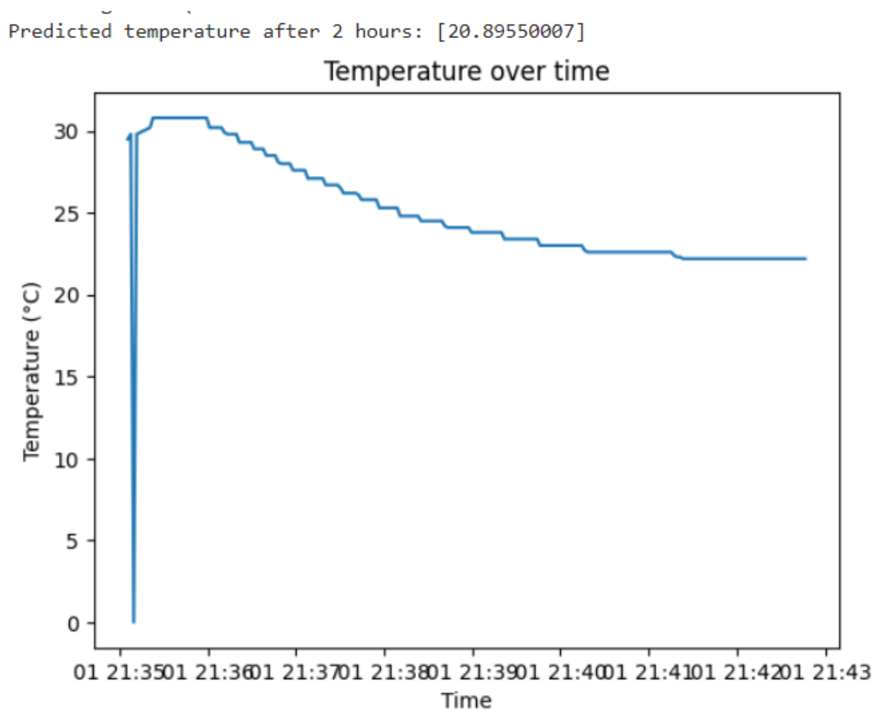
```
# Extract the hour and minute from the timestamp
data['Hour'] = data['Timestamp'].dt.hour
data['Minute'] = data['Timestamp'].dt.minute

# Split the data into features (hour and minute) and target (temperature)
X = data[['Hour', 'Minute']]
y = data['Temperature']

# Create a linear regression model
model = LinearRegression()

# Train the model
model.fit(X, y)

# Predict the temperature for the next 10 minutes
current_time = data['Timestamp'].iloc[-1]
future_time = current_time + timedelta(minutes=10)
future_timestamps = pd.date_range(current_time, future_time, freq='2min')
future_hours = np.array([future_timestamps.hour, future_timestamps.minute]).T
future_temperature = model.predict(future_hours)

# Print the predicted temperature
print('Predicted temperature for the next 10 minutes:', future_temperature)

# Plot the temperature over time
plt.plot(data['Timestamp'], data['Temperature'], label='Measured')
plt.plot(future_timestamps, future_temperature, label='Predicted')
plt.xlabel('Time')
plt.ylabel('Temperature (°C)')
plt.title('Temperature over time')
plt.legend()
plt.show()
```

Output

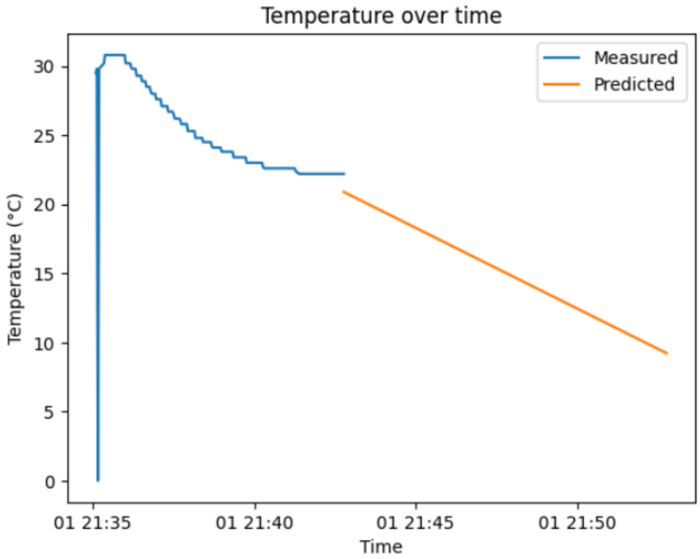



SMART IRRIGATION USING IOT
Offline
crystal
My organization - 8417LC
Add Tag
Dashboard
Timeline
Device Info
Metadata
Actions Log
Latest
Last Hour
6 Hours
1 Day
1 Week
1 Month
3 Months
Custom
Temperature
28.5 °C
0
70
Humidity
67 %
0
100
Moisture Content of soil
1
0
100
Valve Status
1
Region: blr1
Privacy Policy

Graphs


Predicted temperature after 2 hours: [20.89550007]
Temperature over time
Temperature (°C)
30
25
20
15
10
5
0
01 21:35
01 21:36
01 21:37
01 21:38
01 21:39
01 21:40
01 21:41
01 21:42
01 21:43
Time

```
Predicted temperature for the next 10 minutes: [20.89550007 18.56188161 16.22826315 13.89464468 11.56102622  9.22740776]
```



Link for files

https://drive.google.com/drive/folders/1r8Si5fC7_R7LOlYKqKBK4Vo0yAbKSxR2?usp=share_link

Conclusion

In conclusion, an IoT-based plant monitoring system can provide numerous benefits for plant growth and management. By using sensors to collect data on soil moisture, temperature, light intensity, and other environmental factors, the system can optimize plant growth conditions and alert users to any issues that may arise.

The real-time monitoring and remote access provided by an IoT-based system can also make plant management more efficient and cost-effective, as it allows for timely intervention and reduces the need for manual labour.

References

- IoT-Based Hydroponic Plant Monitoring and Control System to Maintain Plant Fertility from INTEK Jurnal Penelitian, 2022.

- Automatic Plant Watering and Monitoring System using NodeMCU from 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2019
- IoT based smart garden monitoring system using NodeMCU microcontroller from International Journal of Advanced and Applied Sciences, 2020.
- IoT Backyard: Smart Watering Control System from 8 Seventh ICT International Student Project Conference
- Implementation of Real-Time Monitoring on Agricultural Land of Rice Plants Using Smart Sensor from The 3 rd International Conference on Electrical, Telecommunication and Computer Engineering (ELTICOM), 2019.
- Raspberry Pi SCADA Zonal based System for Agricultural Plant Monitoring from 6th International Conference on Information Science and Control Engineering (ICISCE), 2019.
- Systematic review of Internet of Things in smart farming from wileyonlinelibrary.com/journal/ett, 2019
- IoT Based Low Cost Smart Indoor Farming Management System Using an Assistant Robot and Mobile App from Electrical Power, Electronics, Communications, Controls and Informatics Seminar, 2021
- Optimal Plant Growth in Smart Farm Hydroponics System using the Integration of Wireless Sensor Networks into Internet of Things from Advances in Science, Technology and Engineering Systems Journal, 2017
- Internet of Things based Wireless Plant Sensor for Smart Farming from Indonesian Journal of Electrical Engineering and Computer Science,2018