Final Report - Plant Aid Monitor (Group D)

Syndariah Johnson, Romello Turner, Diamond Watson

Description

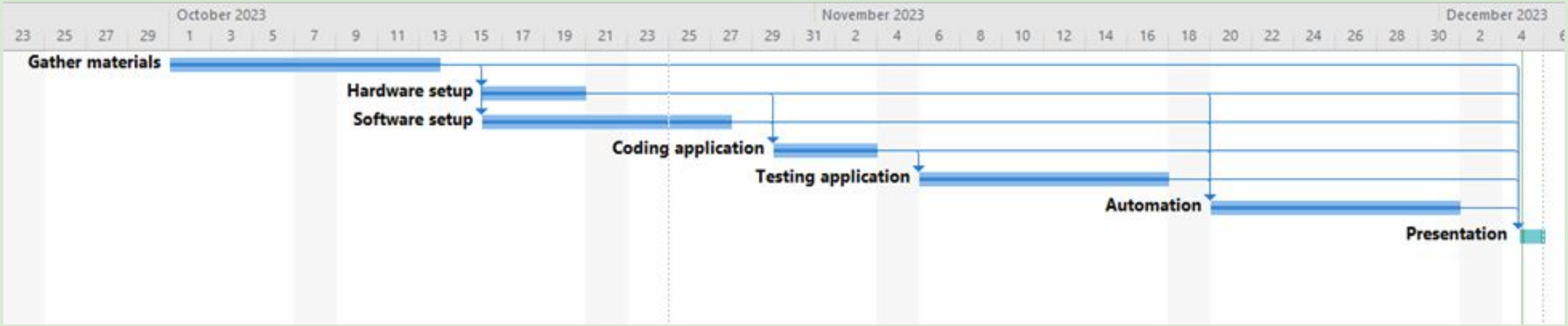
The plant aid monitor is an innovative system to enhance agricultural management. It combines advanced sensors to the plant’s environment to provide insights in real time on the plants soil conditions.

This helps the agriculture industry, such as farmers or anyone with plants/crops, to monitor their plants/crops and be able to identify any issues or signs of unhealthy conditions to be treated sooner than later. This improves time efficiency, crop productivity and sustainability.

Project Activities

		Task Name	Start	Duration	Finish	% Complete	Responsible party
1	✓	Gather materials	Sun 10/1/23	10 days	Fri 10/13/23	100%	Everyone
2	✓	Hardware setup	Mon 10/16/23	5 days	Fri 10/20/23	100%	Everyone
3	✓	Software setup	Mon 10/16/23	10 days	Fri 10/27/23	100%	Everyone
4	✓	Coding application	Mon 10/30/23	5 days	Fri 11/3/23	100%	Everyone
5	✓	Testing application	Mon 11/6/23	10 days	Fri 11/17/23	100%	Everyone
6	✓	Automation	Mon 11/20/23	10 days	Fri 12/1/23	100%	Everyone
7		Presentation	Tue 12/5/23	1 day	Tue 12/5/23	0%	Everyone

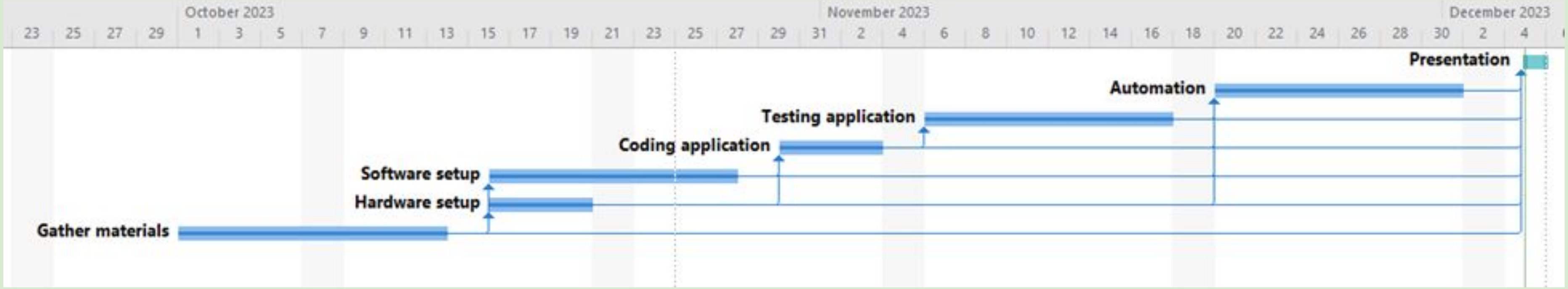
Gantt Chart



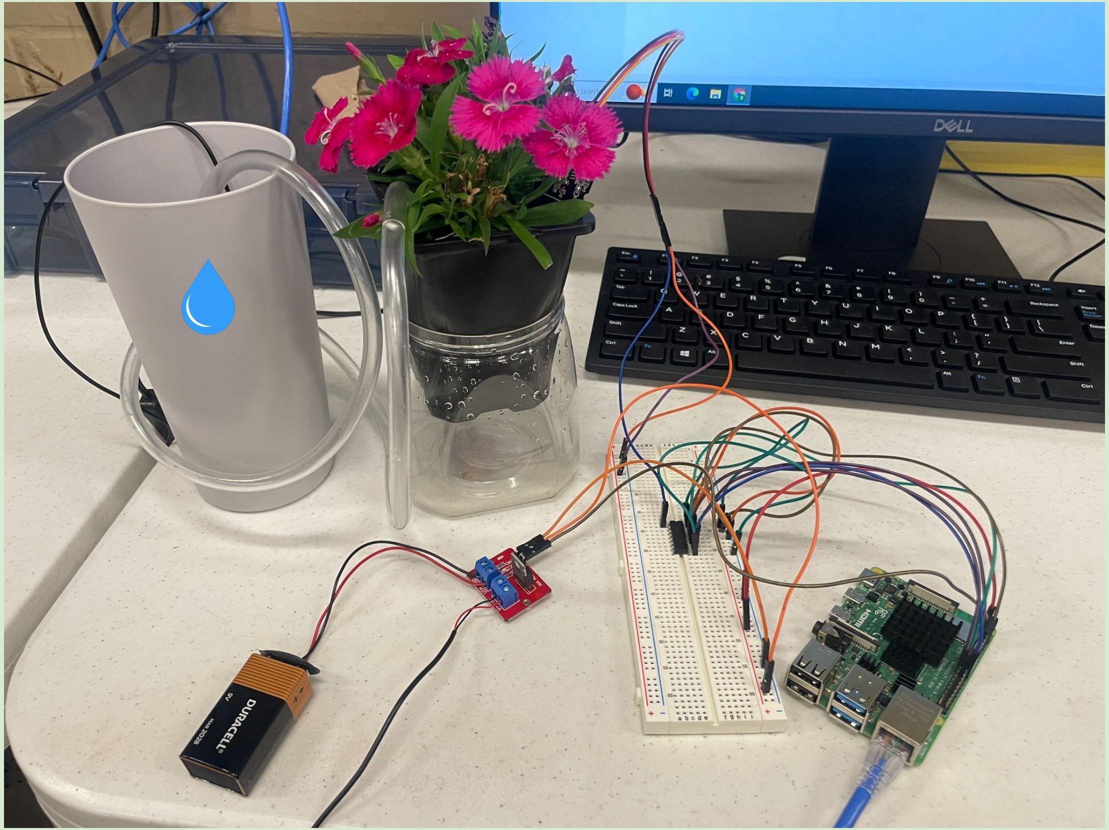
October 2023
November 2023
December 2023
23 25 27 29 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 2 4
Gather materials
Hardware setup
Software setup
Coding application
Testing application
Automation
Presentation

Backwards schedule

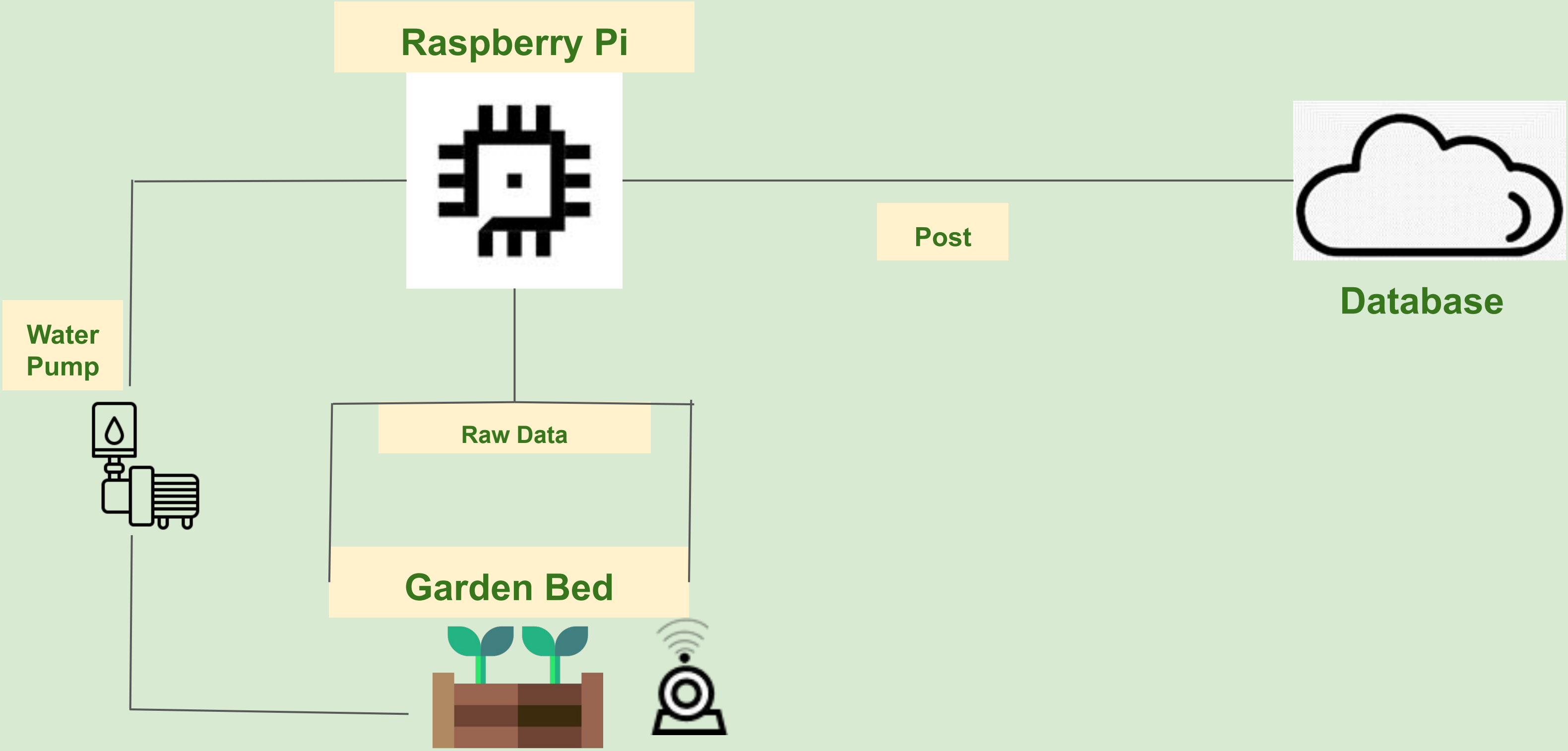
		Task Name	Start	Duration	Finish	% Complete	Responsible party
1		Presentation	Tue 12/5/23	1 day	Tue 12/5/23	0%	Everyone
2	✓	Automation	Mon 11/20/23	10 days	Fri 12/1/23	100%	Everyone
3	✓	Testing application	Mon 11/6/23	10 days	Fri 11/17/23	100%	Everyone
4	✓	Coding application	Mon 10/30/23	5 days	Fri 11/3/23	100%	Everyone
5	✓	Software setup	Mon 10/16/23	10 days	Fri 10/27/23	100%	Everyone
6	✓	Hardware setup	Mon 10/16/23	5 days	Fri 10/20/23	100%	Everyone
7	✓	Gather materials	Sun 10/1/23	10 days	Fri 10/13/23	100%	Everyone



Hardware setup



Avatar Model of Plant Aid Monitor



Python Code (From [Medium]())

```python
# Import necessary libraries
import RPi.GPIO as GPIO  # GPIO library for Raspberry Pi
import datetime          # Library for date and time operations
import spidev            # SPI interface library
import time             # Time-related functions


# Create a SPI connection
spi = spidev.SpiDev()
spi.open(0,0)
spi.max_speed_hz = 1000000  # Set SPI speed to 1 MHz

# Define a function to read data from MCP3008 ADC
def readData(channel):
    adc = spi.xfer2([1, (8 + channel) << 4, 0])
    data = ((adc[1] & 3) << 8) + adc[2]
    return data

# Define GPIO pin for the pump
pinPump = 4  # GPIO pin connected to the pump
needsWater = 350  # Threshold sensor value for dry air
```

Python Code

```
# General GPIO settings
GPIO.setwarnings(False)  # Ignore GPIO warnings (irrelevant here)
GPIO.setmode(GPIO.BCM)   # Set GPIO pin numbering mode to BCM
GPIO.setup(pinPump, GPIO.OUT)  # Set the pump pin as an output
GPIO.output(pinPump, GPIO.LOW)  # Turn the pump off initially


# Read moisture data from channel 0 of the MCP3008
moisture = readData(0)

# Open a file to write time and current moisture level
f = open("/home/syndariah/IOTgarden/WateringStatss.csv", "a")
currentTime = datetime.datetime.now()

# Calculate and write the current moisture percentage and raw value
f.write((str(currentTime)) + "," + str(round((moisture - 330) / 450 * 100, 2)))
```

Python Code

```
# Check if the plants are too dry and need watering
if moisture > needsWater:
    t_end = time.time() + 4  # Pump will run for 4 seconds

    # Start pumping
    while (time.time() < t_end):
        GPIO.output(pinPump, GPIO.HIGH)

    GPIO.output(pinPump, GPIO.LOW)  # Turn the pump off
    #f.write("Plants got watered!\n")

f.write("\n")  # Add a line break for the next log entry
f.close()  # Close the file
GPIO.cleanup()  # Properly clean up used GPIO pins
```

Python Code

```python
import csv
def calculate_statistics(csv_file_path):
    # Initialize variables
    total_moisture = 0
    min_moisture = float('inf')
    max_moisture = float('-inf')

    # Open the CSV file
    with open(csv_file_path, 'r') as csv_file:
        # Create a CSV reader
        csv_reader = csv.DictReader(csv_file)

        # Iterate through each row in the CSV file
        for row in csv_reader:
            # Assuming 'moisture' is the column header in your CSV file
            moisture_value = float(row['moisture'])

            # Update total moisture, minimum, and maximum
            total_moisture += moisture_value
            min_moisture = min(min_moisture, moisture_value)
            max_moisture = max(max_moisture, moisture_value)

    # Calculate the average moisture and range
    num_rows = csv_reader.line_num - 1  # Subtract 1 to exclude the header row
    average_moisture = total_moisture / num_rows if num_rows > 0 else None
    moisture_range = (min_moisture, max_moisture)

    return average_moisture, min_moisture, max_moisture, moisture_range
```

Python Code

```
# Replace 'WateringStats.csv' with the actual path to your CSV file
csv_file_path = 'WateringStatss.csv'

# Calculate statistics
average_moisture, min_moisture, max_moisture, moisture_range = calculate_statistics(csv_file_path)

# Write the statistics to a new file
output_file_path = 'MeaningfulOutput.txt'
with open(output_file_path, 'w') as output_file:
    if average_moisture is not None:
        output_file.write(f'Average Moisture: {average_moisture:.2f}\n')
        output_file.write(f'Minimum Moisture: {min_moisture:.2f}\n')
        output_file.write(f'Maximum Moisture: {max_moisture:.2f}\n')
        output_file.write(f'Range of Moisture: {moisture_range}')
    else:
        output_file.write('No data in the CSV file.')

# Print a message indicating where the output is saved
print(f'Output saved to: {output_file_path}')
```

Python Code

```
import csv
import matplotlib.pyplot as plt

def calculate_statistics(csv_file_path):
    # ... (same as before)

# Replace 'WateringStats.csv' with the actual path to your CSV file
    csv_file_path = 'WateringStatss.csv'

# Calculate statistics
    average_moisture, min_moisture, max_moisture, moisture_range = calculate_statistics(csv_file_path)

# Bar chart
categories = ['Average Moisture', 'Minimum Moisture', 'Maximum Moisture']
values = [average_moisture, min_moisture, max_moisture]

plt.bar(categories, values, color=['blue', 'green', 'red'])
plt.ylabel('Moisture Level')
plt.title('Moisture Statistics')


# Create empty list
currentTime = []
moisture = []
# Read data from the CSV file
```

Python Code

```
# Read data from the CSV file
with open('WateringStatss.csv', 'r') as csvfile:
    csvreader = csv.DictReader(csvfile) # Create a CSV reader
    for row in csvreader:
        currentTime.append(row['currentTime']) # Append 'Activity Day' data to the list
        moisture.append(float(row['moisture'])) # Append 'Usage Charge' data to the list

# Initialize an empty scatter plot
plt.figure(figsize=(10, 6))
plt.xlabel('Current Time')
plt.ylabel('Moisture Level')
plt.title('Real-time Scatter Plot of Priscilla Moisture Level - Testing')

# Iterate through the data and update the plot in real-time
for i in range(len(currentTime)):
    plt.scatter(currentTime[i], moisture[i], marker='o') # Add a point to the scatter plot
    plt.pause(1)  # Adjust the pause duration as needed for the desired update rate
# Show the plot
plt.grid(True)  # Add a grid to the plot
plt.show()
```

Ln: 104 Col: 0

Step by Step Instructions

1. Gather materials
   - Raspberry Pi4, soil moisture sensor(s) v1.2, MOS module, analog-to-digital convertor (MCP3008), submersible 3V DC water pump, PVC tube, breadboard, male-male wires, female-female wires, male-female wires, battery clip, Duracell 9v battery, plant(s) and soil, water, vase
2. Configure Python code
   - ```
     # Import necessary libraries
     import RPi.GPIO as GPIO  # GPIO library for Raspberry Pi
     import datetime          # Library for date and time operations
     import spidev            # SPI interface library
     import time              # Time-related functions

     # Create a SPI connection
     spi = spidev.SpiDev()
     spi.open(0, 0)
     spi.max_speed_hz = 1000000  # Set SPI speed to 1 MHz
     ```
     ```

Step by Step Instructions

```
# Define a function to read data from MCP3008 ADC
def readData(channel):
    adc = spi.xfer2([1, (8 + channel) << 4, 0])
    data = ((adc[1] & 3) << 8) + adc[2]
    return data

# Define GPIO pin for the pump
pinPump = 4  # GPIO pin connected to the pump
needsWater = 350  # Threshold sensor value for dry air

# General GPIO settings
GPIO.setwarnings(False)  # Ignore GPIO warnings (irrelevant here)
GPIO.setmode(GPIO.BCM)   # Set GPIO pin numbering mode to BCM
GPIO.setup(pinPump, GPIO.OUT)  # Set the pump pin as an output
GPIO.output(pinPump, GPIO.LOW)  # Turn the pump off initially
```

Step by Step Instructions

```
# Read moisture data from channel 0 of the MCP3008
moisture = readData(0)

# Open a file to write time and current moisture level
f = open("/home/syndariah/IOTgarden/WateringStatss.csv", "a")
currentTime = datetime.datetime.now()

# Calculate and write the current moisture percentage and raw value
f.write((str(currentTime)) + "," + str(round((moisture - 330) / 450 * 100, 2)))
```

Step by Step Instructions

```
# Check if the plants are too dry and need watering
if moisture > needsWater:
    t_end = time.time() + 4  # Pump will run for 4 seconds

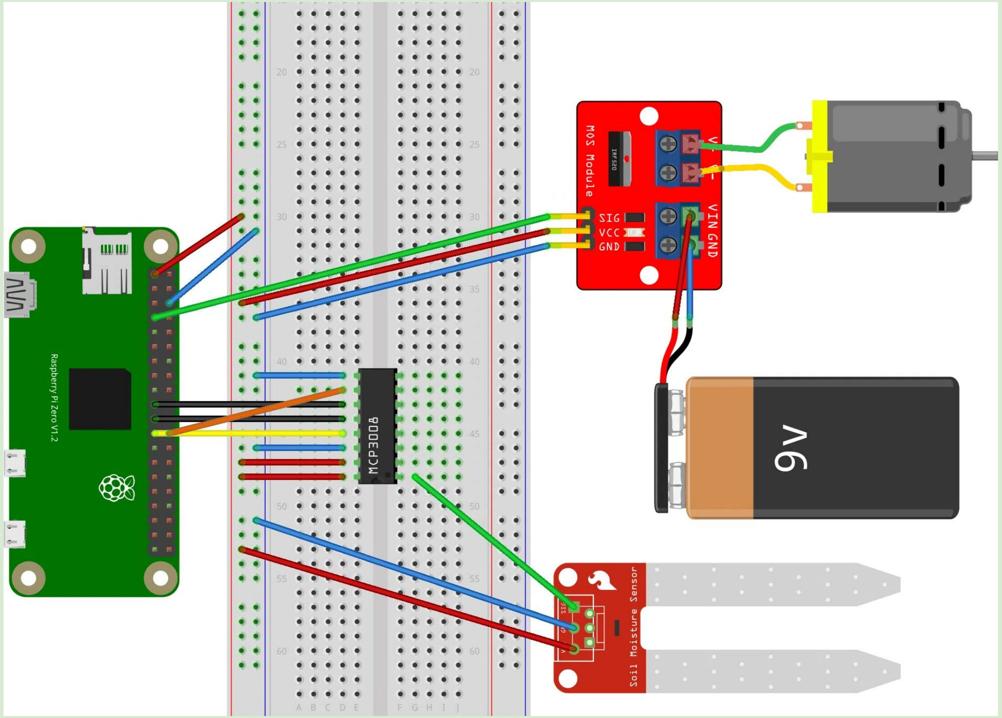
    # Start pumping
    while (time.time() < t_end):
        GPIO.output(pinPump, GPIO.HIGH)

    GPIO.output(pinPump, GPIO.LOW)  # Turn the pump off
    #f.write("Plants got watered!\n")

f.write("\n")  # Add a line break for the next log entry
f.close()  # Close the file
GPIO.cleanup()  # Properly clean up used GPIO pins
```

Step by Step Instructions

3. Assemble the hardware



4. Setup Cron job to your liking

5. Test the functions of the IOT Garden

Output Analysis

Variety of Meaningful Output

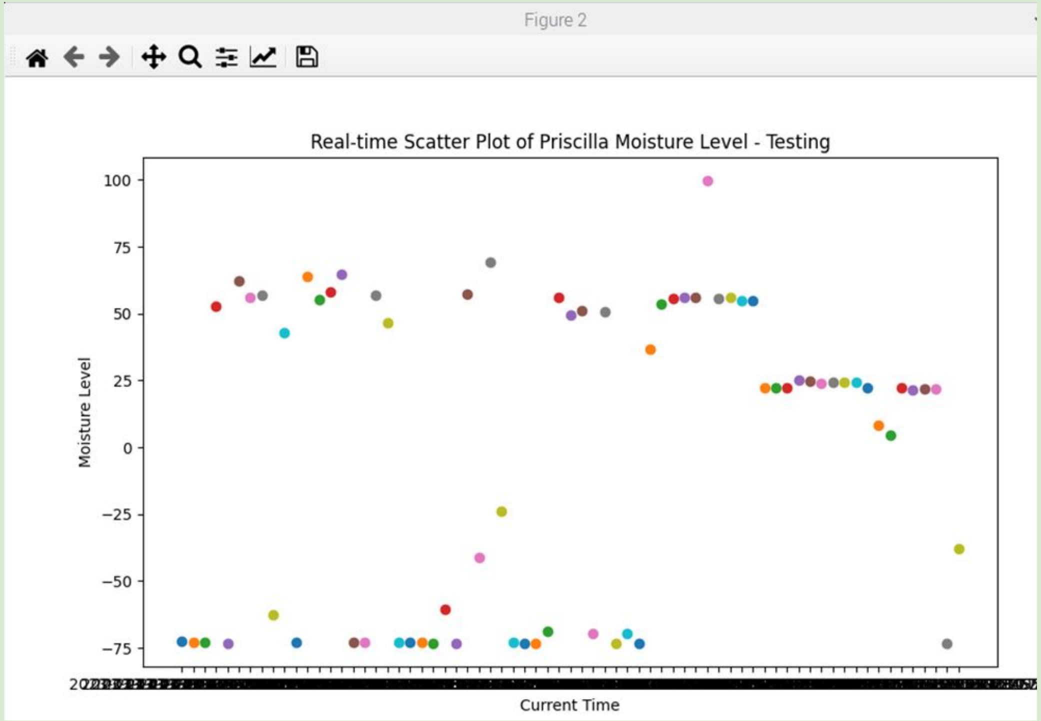
PiCode.py | WateringStatss.csv

```
currentTime,moisture
2023-11-14 20:00:24.196143,-72.67
2023-11-14 20:06:53.416933,-72.89
2023-11-14 20:11:46.917413,-72.89
2023-11-14 20:21:10.937940,52.89
2023-11-14 20:22:38.596014,-73.33
2023-11-14 20:23:07.321645,62.44
2023-11-14 20:24:07.080872,56.0
2023-11-14 20:25:21.678229,57.11
2023-11-14 20:26:18.689232,-62.67
2023-11-14 20:26:59.320457,42.89
2023-11-14 20:27:01.029799,-73.11
2023-11-14 20:27:36.386040,63.78
2023-11-14 20:34:49.213512,55.33
2023-11-14 20:35:13.772587,58.0
2023-11-14 20:35:31.389629,64.89
2023-11-14 20:44:05.414050,-72.89
2023-11-14 20:44:25.928117,-72.89
2023-11-14 20:44:46.063931,57.11
2023-11-15 11:50:01.556568,46.44
2023-11-15 11:50:46.187585,-73.11
2023-11-15 11:51:05.506387,-73.11
2023-11-15 11:51:17.562564,-73.11
2023-11-15 11:51:35.076808,-73.33
2023-11-15 11:51:55.308769,-60.44
2023-11-15 11:52:29.479876,-73.33
2023-11-15 11:54:05.008638,57.33
2023-11-15 11:56:03.908103,-41.33
2023-11-15 11:56:05.653492,69.11
2023-11-15 11:57:19.167857,-23.78
2023-11-15 11:57:42.884938,-72.89
```

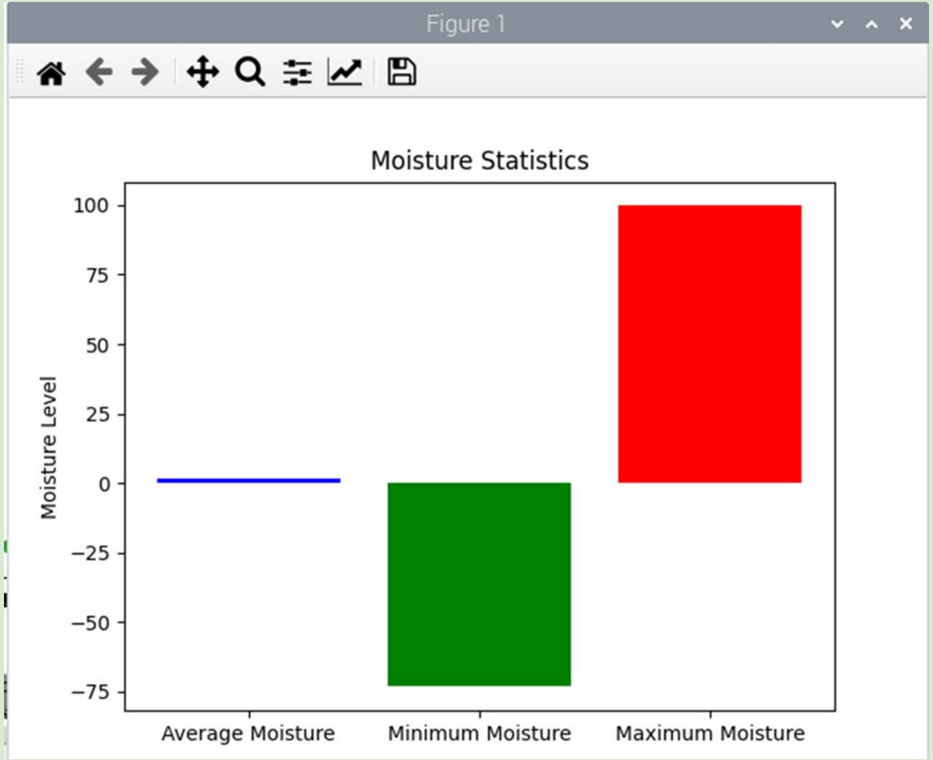
PiCode.py | WateringStatss.csv

```
2023-11-15 12:03:44.426990,-73.33
2023-11-15 12:06:19.124598,-68.89
2023-11-15 12:06:46.564372,56.22
2023-11-28 10:57:35.820737,49.56
2023-11-28 10:59:02.094824,51.11
2023-11-28 10:59:28.731577,-69.78
2023-11-28 11:00:06.896129,50.89
2023-11-28 11:00:45.583821,-73.33
2023-11-28 11:02:04.650471,-69.56
2023-11-28 11:23:23.412944,-73.33
2023-11-28 11:24:39.056411,36.89
2023-11-28 11:25:04.229258,53.78
2023-11-30 09:50:27.417941,55.56
2023-11-30 09:52:50.205753,56.22
2023-11-30 09:58:57.087462,56.0
2023-11-30 10:01:08.675108,99.78
2023-11-30 10:01:36.404339,55.78
2023-11-30 10:02:30.216088,56.0
2023-11-30 10:03:06.394052,54.67
2023-11-30 10:03:12.758321,54.89
2023-12-05 15:49:32.872498,22.44
2023-12-05 15:51:24.000275,22.44
2023-12-05 15:52:47.119375,22.44
2023-12-05 16:14:48.594150,25.11
2023-12-05 16:15:48.012592,24.89
2023-12-05 16:18:22.090087,23.78
2023-12-05 16:18:45.204316,24.22
2023-12-05 16:20:46.698915,24.22
2023-12-05 16:21:12.261061,24.22
2023-12-05 16:24:25.710866,22.44
2023-12-05 16:25:10.295196,8.22
```

Variety of Meaningful Output



Variety of Meaningful Output



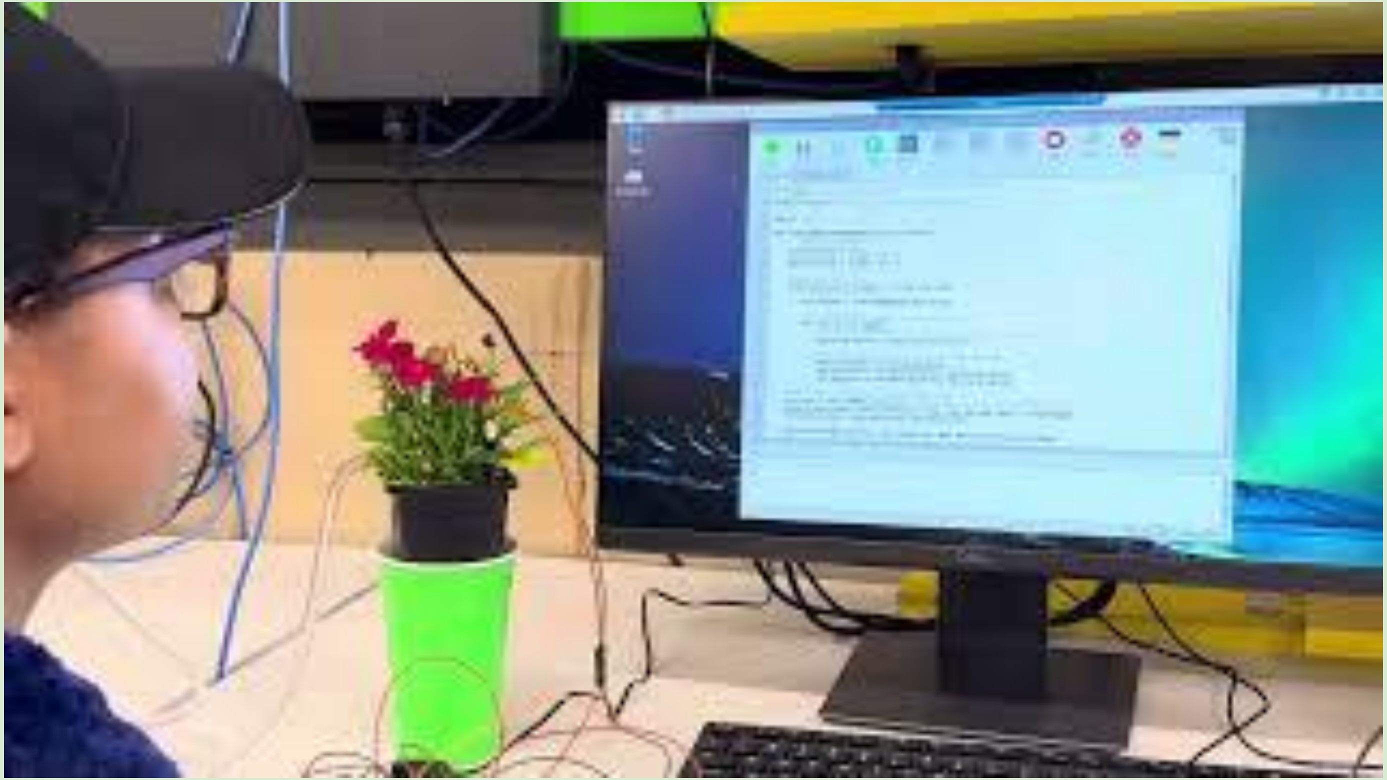
Variety of Meaningful Output

- Mean soil moisture: 1.72
- Maximum soil moisture: 99.78
- Minimum soil moisture: -73.33
- Range of soil moisture: -73.33 – 99.78

Business Use

- Reduce cost
  - The statistics found from the soil can be used to reduce the amount of effort needed to aid the plants by people and more reliant on a systematic approach. Doing this could reduce the costs of amount of care needed or result in fewer employee wages.
  - Efficient water usage and automation lead to reduced operational costs over time.
- Water Conservation
  - Optimizes water usage by delivering the right amount of water to each crop, reducing waste and environmental impact.
- Increased Crop Yield
  - Precision irrigation contributes to healthier plants, potentially increasing yield and quality.
- Remote Management
  - Allows for one to make timely decisions and adjustments without the need for on-site presence.

Project Execution



https://youtu.be/aHdIilp1kvo?si=2KEjkf-1sM_ws1Qa