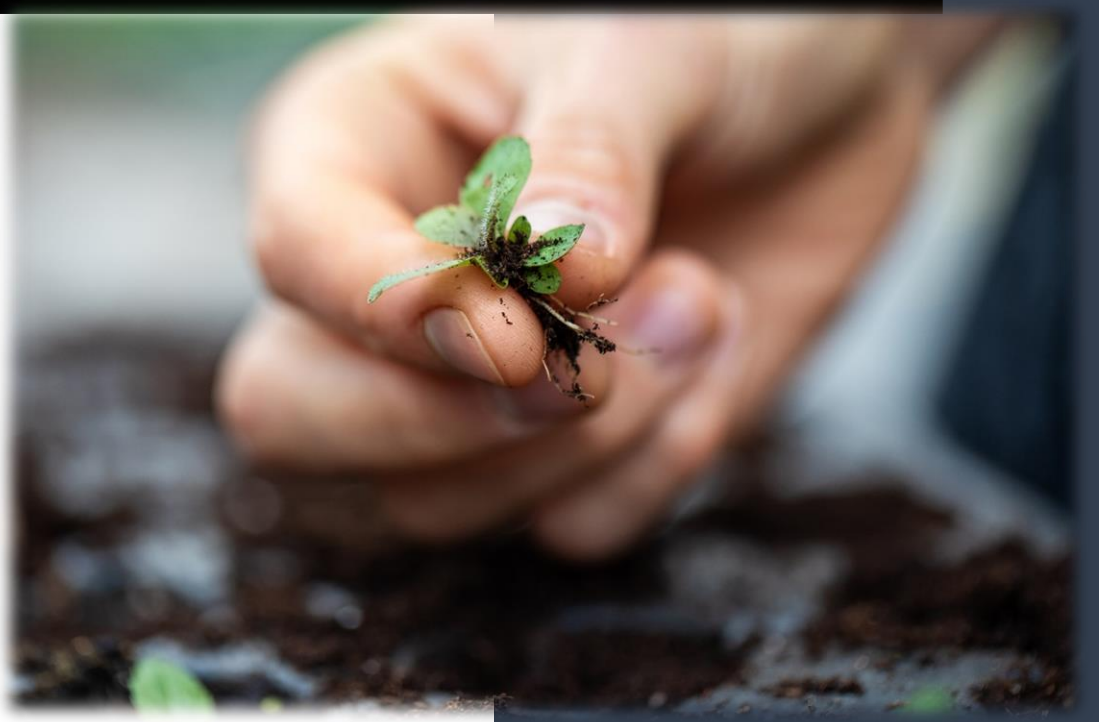
2024

IoT project - SmartGarden



Ben-Yair, Shalom Yehudah

Mazaki, Keren

Abstract – water, sun, fertilization etc. are important aspects for keeping a plant healthy, monitoring these stats and controlling them can help maintain a flourishing garden in our modern days. We propose a monitoring system that takes into consideration light, soil moisture and heat and helps the plant maintain a healthy life. the data communication is done via MQTT and AWS, visualization via Grafana and personal messages to the plant owner via mail or SMS, the paper continues and proposes hardware parts for practical use.

Keywords – smart garden, MQTT, AWS, Grafana, IoT.

1. Introduction

Keeping plants healthy and maintaining a flourishing garden is an ongoing task that demands focus and attention. In our modern world, it is not uncommon to lack the time or attention required to care for our gardens or indoor plants. With the development of technology and the automation of the world around us, monitoring our private gardens and automating them for better maintenance and care has become possible. Utilizing automated monitoring, watering, and fertilizing systems will leave the days of encountering dead plants due to neglect and the stresses of modern lifestyles, to the past.

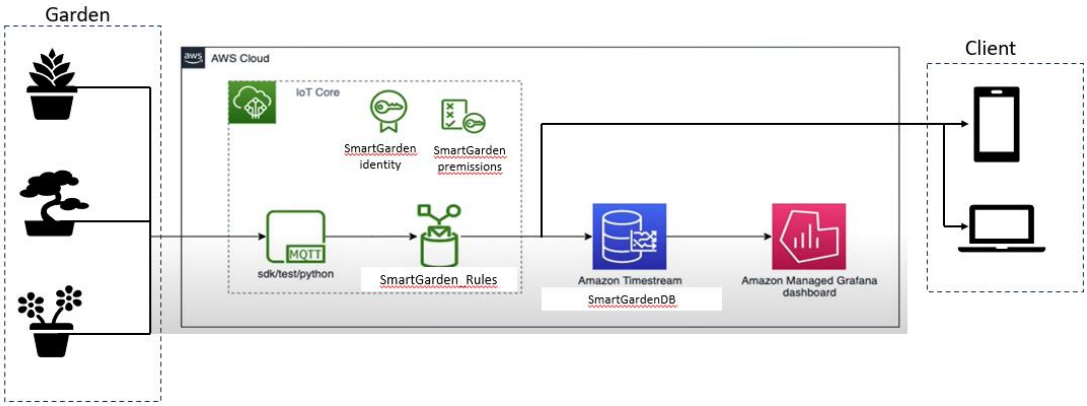
Numerous research projects have been conducted regarding the automation and monitoring of gardens. In [1], the publishers constructed an automatic smart garden watering system incorporating measurements of temperature, humidity, and soil moisture using a DHT11 sensor, a soil moisture sensor, a water pump, and an ESP8266 chip utilizing the MQTT (Message Queue Telemetry Transport) protocol. The monitoring system operates in real-time, allowing users to monitor anytime and anywhere via mobile devices. In [2], the authors developed an IoT-based smart garden system enabling users to monitor various plant parameters, including moisture level, temperature, humidity, and light conditions. The soil moisture can be regulated with the help of the watering system, which can be controlled manually by the user or operated in automatic mode. Fertilizer dispensers have also been integrated into the system. The system operates via Wi-Fi and is supported through a smartphone application and a web dashboard.

This paper proposes a smart garden system relying on AWS, utilizing simulated data such as temperature, soil moisture, and luminance level of plants. The data is transferred and analyzed via MQTT, with the user receiving real-time messages and instructions on tasks to perform.

2. Methodology

A. first design

Our initial design was based on the design from the HW Assignment, described in the illustration below:



We generated an MQTT topic that simulates data being sent from things that are connected to plants in real time. The MQTT message was then received in AWS and visualized in Amazon Grafana. In addition, we created an Email based notification service that notifies the user when needed. Let's elaborate on each step.

Firstly, we connected our PC as a device to AWS, using AWS IoT Core. we changed the pubsub.py file provided so it will generate the data randomly. at the beginning of the script, we initialized the plant's names, and defined the functions that were responsible for updating the data:

```
deviceNames = ['Mint', 'Tomatos', 'Bonsai', 'Cucumbers', 'Basil']

data = {device: {'Temperature': random.randint(15,35), 'Humidity': random.randint(50,70),
                 "Illuminance": random.randint(1000,5000)} for device in deviceNames}
average_array = np.zeros(5)
average = 0

def getTemperatureValues():
    for device in deviceNames:
        data[device]["Temperature"] += random.randint(-1, 1)
        if data[device]["Temperature"] > 50:
            data[device]["Temperature"] = 50
        if data[device]["Temperature"] < -5:
            data[device]["Temperature"] = -5
    return

def getHumidityValues():
    for device in deviceNames:
        data[device]["Humidity"] += random.randint(-2, 2)
        if data[device]["Humidity"] > 100:
            data[device]["Humidity"] = 100
        if data[device]["Humidity"] < 0:
            data[device]["Humidity"] = 0
    return

def getIlluminanceValues():
    for device in deviceNames:
        data[device]["Illuminance"] += random.randint(-100, 100)
        if data[device]["Illuminance"] > 5000:
            data[device]["Illuminance"] = 5000
        if data[device]["Illuminance"] < 0:
            data[device]["Illuminance"] = 0
    global average_array
    average_array = np.append(average_array[1:],(data["Basil"]["Illuminance"]))
    global average
    average = np.mean(average_array)
    return
```

then, we updated the message that was being published:

```
while (publish_count <= message_count) or (message_count == 0):
    message = "{} [{}]".format(message_string, publish_count)
    print("Publishing message to topic '{}': {}".format(message_topic, message))
    messege = {}
    for device in deviceNames:
        messege[f"{device}_Temperature"] = data[device]['Temperature']
        messege[f"{device}_Humidity"] = data[device]['Humidity']
        messege[f"{device}_Illuminance"] = data[device]['Illuminance']
    messege["average_Basil_Illuminance"] = average
    message_json = json.dumps(messege, indent = 2)
```

The message was sent via MQTT, containing the data we collect for each plant, such as temperature, soil moisture and illuminance, with plant's name, and received to AWS:

▼ sdk/test/python

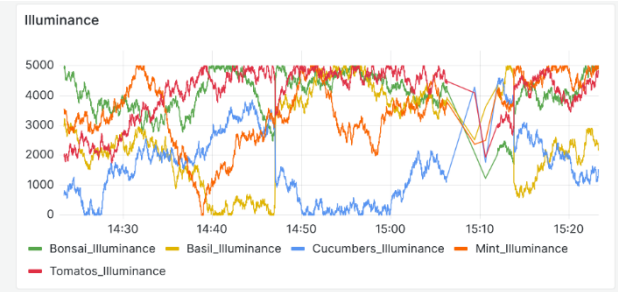

```
{
  "Mint_Temperature": 43,
  "Mint_Humidity": 46,
  "Mint_Illuminance": 1619,
  "Tomatos_Temperature": -23,
  "Tomatos_Humidity": 86,
  "Tomatos_Illuminance": 3375,
  "Bonsai_Temperature": 14,
  "Bonsai_Humidity": 92,
  "Bonsai_Illuminance": 2113,
  "Cucumbers_Temperature": 40,
  "Cucumbers_Humidity": 112,
  "Cucumbers_Illuminance": 4509,
  "basil_Temperature": 50,
  "basil_Humidity": 61,
  "basil_Illuminance": 1306
}
```

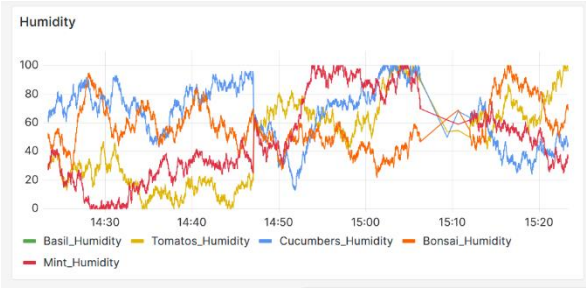

Next, we wanted to display the data through time in Grafana. To do so, we created a timestream rule, and a database with a table following the rule.
To make sure that the data is being transferred to the table, we created the following query:

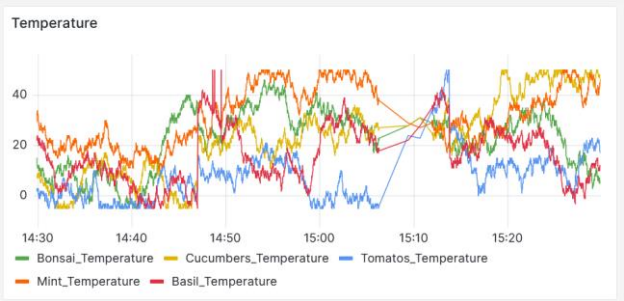
Query 1 +

```
1 select *
2 from
3 smart_garden_DB."smart_garden_table"
4 where measure_name = 'Bonsai_Illuminance'
```

Then, we created a workspace and displayed the data in Amazon Grafana:





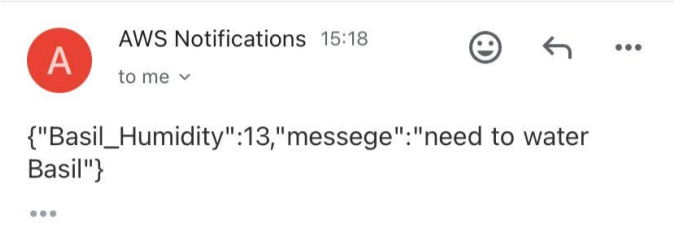


Secondly, we wanted to create a notification service that alerts the user according to specific rules, such as low humidity, high temperature or low illuminance. To do so, we connected our Email address to the SNS service in AWS, and created rules for the notifications. For example:

SQL statement

```
SELECT Basil_Humidity, "need to water Basil" as
messege FROM 'sdk/test/python' WHERE Basil_Humidity <
15
```

And the Email alert:



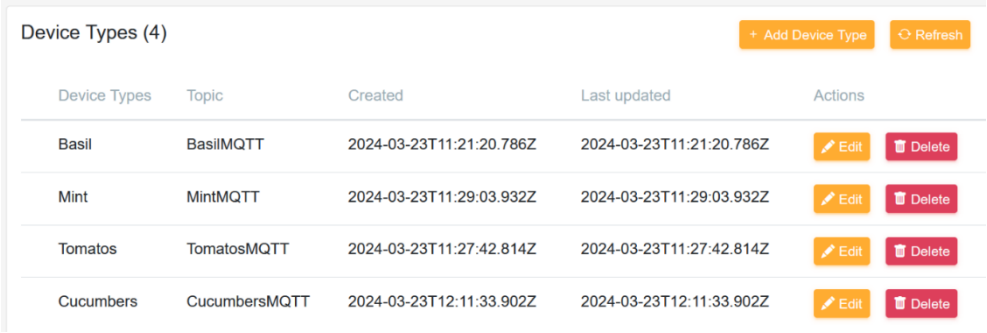
Alongside the messages, the visualization in Grafana allows us to better understand the patterns of each parameter.

B. Second Design:

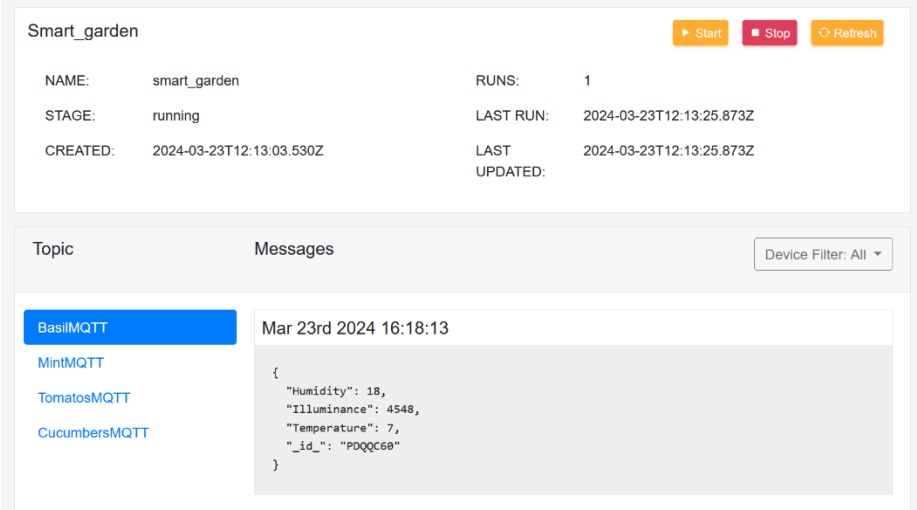
After implementing the first simulation, we wanted to use a more convenient way to simulate things, and to better handle MQTT messages.

To do so, we used an IoT simulator provided by AWS, following [6].

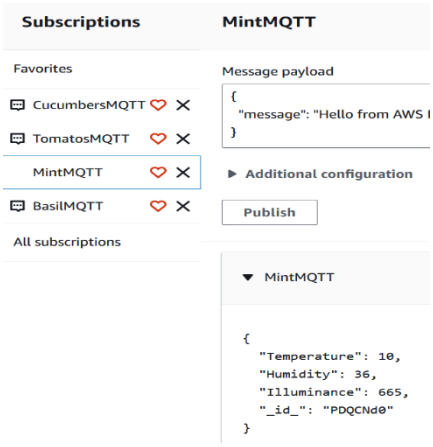
We created a device for each plant, where each device sends properties of the plant, and has a different MQTT topic:



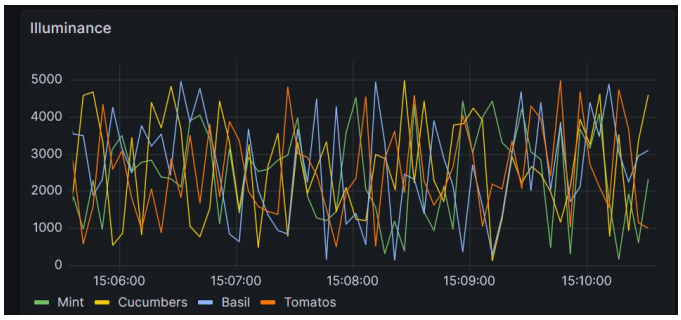
Then, we created the simulation, adding one device of each type:

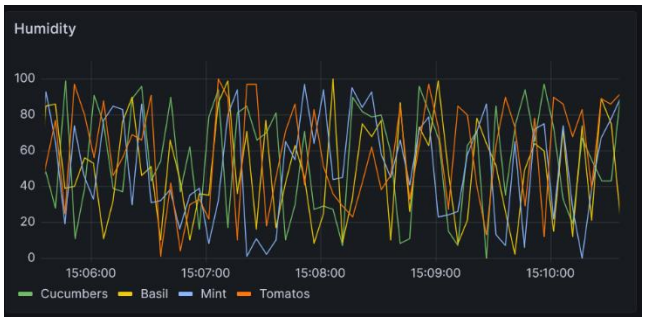


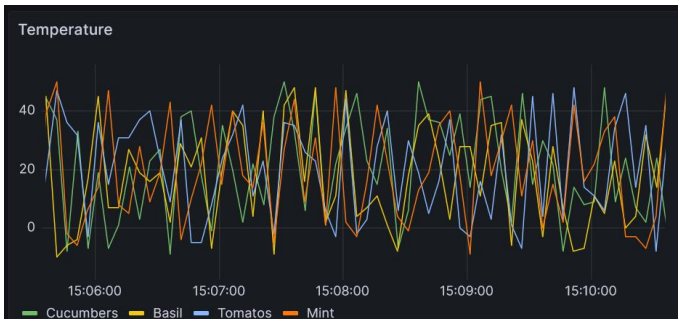
We confirmed that the messages were being received to the IoT Core Service by subscribing to the topics:

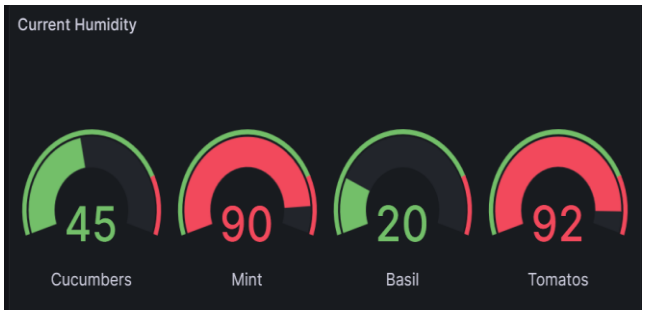


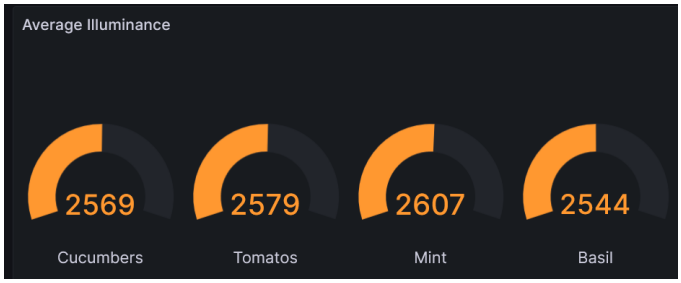
Lastly, we displayed the data in Grafana, like before:

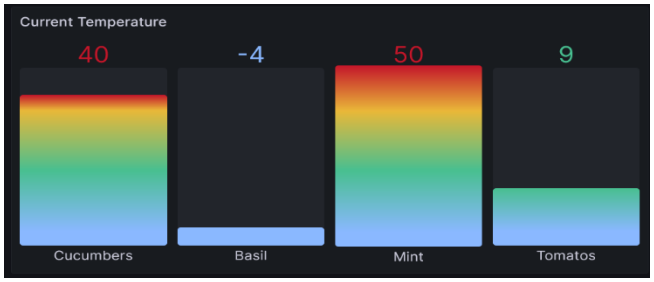












By simulating the data in this manner, we could better control and adjust the messages, and easily separate the plants from each other, by creating different topics for each of them.

C. Hardware requirements (for practical use)

When implementing the smart garden, hardware components are required for measuring all the parameters that we discussed.

For temperature and humidity measuring, the DHT11 is a perfect match, for soil moisture measuring, you can use a soil moisture sensor by SparkFun, and for illuminance indication, a simple LDR.

- **DHT11:** A low-cost, high-reliability and long-term stability humidity & temperature sensor. It includes a resistive-type humidity measurement component and an NTC temperature measurement component. It has 4 legs, one connects to the supply voltage, one to the ground and the other to the data, using a microcontroller (Arduino for example|). [3].

- **SparkFun Soil Moisture Sensor:** A simple device used for measuring the moisture in soil. The two large, exposed pads function as probes for the sensor, together acting as a variable resistor. The more water that is in the soil means the better the conductivity between the pads will be, resulting in a lower resistance and a higher out signal. the sensor has 3 pins: VCC, GND and SIG, which need to be connected to a microcontroller. [4].

- **LDR (light dependent resistor):** a resistor whose resistance varies depending on the amount of light falling on its surface. The resistor works on the principle of photo conductivity. When the light falls on its surface, the material's conductivity reduces because the electrons in the valence band of the device are excited to the conduction band. This makes the electrons jump from the valence band to conduction band and the conductivity increases [5].

3. Conclusions

In this paper we presented our IoT project – SmartGarden, we showed the methodology and work flow and dived into the various sensors one will may use for a practical implementation. We discussed two different simulation types an continued to communication and visualization. The system is low cost and affordable but also efficient and simple – for everyday and everyone's use.

4. References

[1] Denny Kurniawan, Rizki Jumadil Putra, Agung Bella, Muhammad Ashar and Khen Dedes, "Smart Garden with IoT Based Real Time Communication using MQTT Protocol", 2021 ,*7th International Conference on Electrical, Electronics and Information Engineering (ICEEIE)*, DOI: 10.1109/ICEEIE52663.2021.9616869.[online]:https://ieeexplore.ieee.org/document/9616869

[2] Gauri R Choudhari, Pratik A Dagale, Isha S Dashetwa , Rutuja R Desa and Abha A Marathe, "IoT-based Smart Gardening System", 2023*, J. Phys. Conf. Ser. 2601 012006* .[online]:https://iopscience.iop.org/article/10.1088/1742-6596/2601/1/012006

[3] DATASHEET: DHT11 Humidity & Temperature Sensor, [online]:https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf

[4] DATASHEET: SparkFun Soil Moisture Sensor, [online]: https://www.hackster.io/sparkfun/products/sparkfun-soil-moisture-sensor-with-screw-terminals1/specs

[5] WatElectronics, "What is a Light Dependent Resistor and Its Applications", 2019, [online]: https://www.watelectronics.com/light-dependent-resistor-ldr-with-applications/#google_vignette

[6] Salman Djingueinabaye, "IotDataToTimestream", *GitHub proj.* [online]:https://github.com/salmandjing/IotDataToTimestream?tab=readme-ov-file