بسم الله الرحمن الرحيم

Plant Monitoring System

Introduction

Traditional farming has high environmental cost in terms of the amounts of water poured into the farms and the large area of the farms.

Since the overall cost in terms of resources for running a farm is high, any optimization can make a significant difference. According to the world bank one third of the earth's land is dedicated to farming, optimizing water consumption for that wide area will have positive impacts on the availability of water for people.

Background Study

Multiple studies show that controlled farms can use less resources than conventional farming. A controlled cucumber crop in the USA required 28% less nitrogen and 23% less potassium per kilogram of fruit compared with a traditionally grown cucumber crop (Jovicich et al., 2007).

An intelligent plant care system was developed using IoT technology by Dr. B. Paulchamy et al. Where they provided control to the farm under test, the IoT system can also control the flow of water and can control CO2 emissions based on controlling the flow of nutrients to plants.

Another system by R. Rajkumar and R. Dharmaraj used Wireless Sensor Networks and monitored temperature and humidity, then those

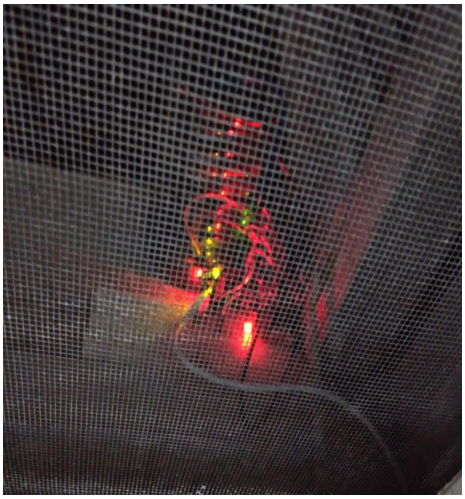
parameters were sent to the cloud using the Blynk platform then to the farmer through SMS so they could monitor their plants and add nutrients whenever needed.

In Melchizedek I. Alipio et al. 's work used a Bayesian network to automate data analytics in their farming system. To provide accurate metrics to farmers, they used sensors for pH, light intensity, water temperature and also humidity. The collected data composed a Bayesian network then Nutrient Film Technique was implemented and attached to the farm. The system also supported viewing those metrics via a web interface so it’s accessible anywhere.

Project Setup

Sensor Node

In our setup we can have multiple sensor nodes, each one will have a



factory assigned id by which the node obtains a global UUID to be used across the system.

After the sensor node boots up, it tries to connect to WiFi and keeps trying until it succeeds, after connecting to WiFi the node tries to reach out for the Fog server present in the same LAN. After that it performs time synchronization using the NTP protocol so packets sent to the Fog node will have a correct timestamp.

Once the node connects to the Fog server, it goes through a hand shaking process and exchanges its factory assigned ID with a UUID.

After the handshaking is complete, the node is now able to send data to the Fog node.

The sensor node sends its readings regularly to the fog node every 20 seconds, the interval shouldn't be very small as this way a lot of unnecessary readings will be sent to the server without much benefit. The system of the farm is stable given a small period of time.

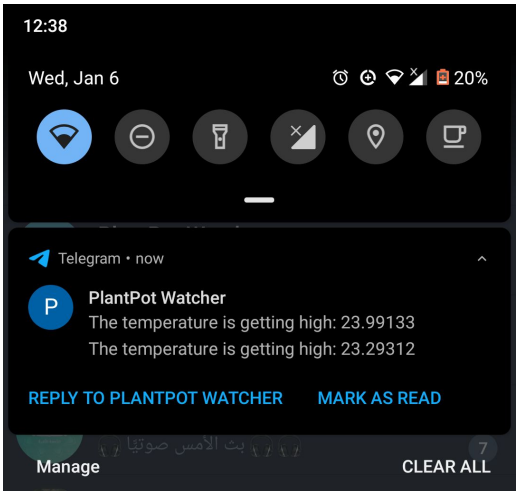
Fog Node

Once the Fog node receives the node message it adds it to a message queue, this allows for batching messages from sensor nodes together to avoid sending a lot of messages to the cloud server.

The fog node runs 2 code threads to avoid blocking on listening the nodes and to enable both receiving data from the node and sending MQTT messages simultaneously. One added benefit is that using this method reduces the CPU load compared to polling, which in turn saves power and makes the system consume less energy.

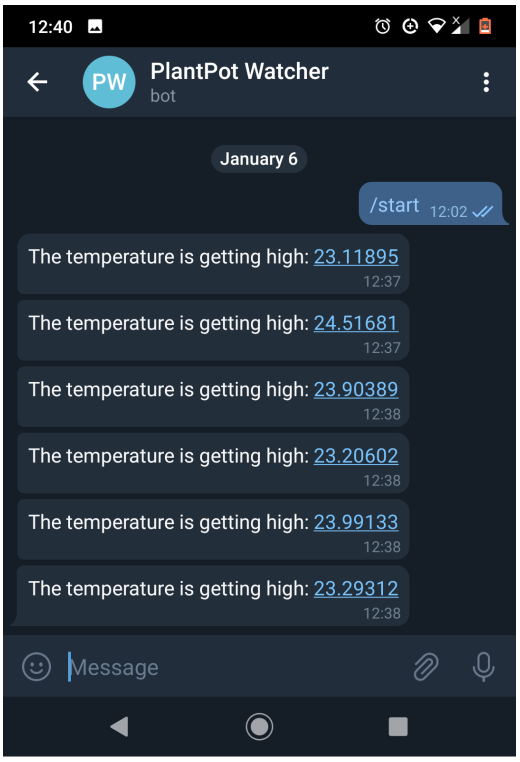
Notifications [New]

The fog node notifies the user using TelegramAPI where there are any readings outside the acceptable range.



In the shown example we notify the user when the temperature is above 23 degrees Celsius.

This works by making a telegram bot and using python code to integrate with the telegram server so it makes the bot send the message to the user.



It’s important to know the id of the user in order to be able to send them messages.

Cloud Server

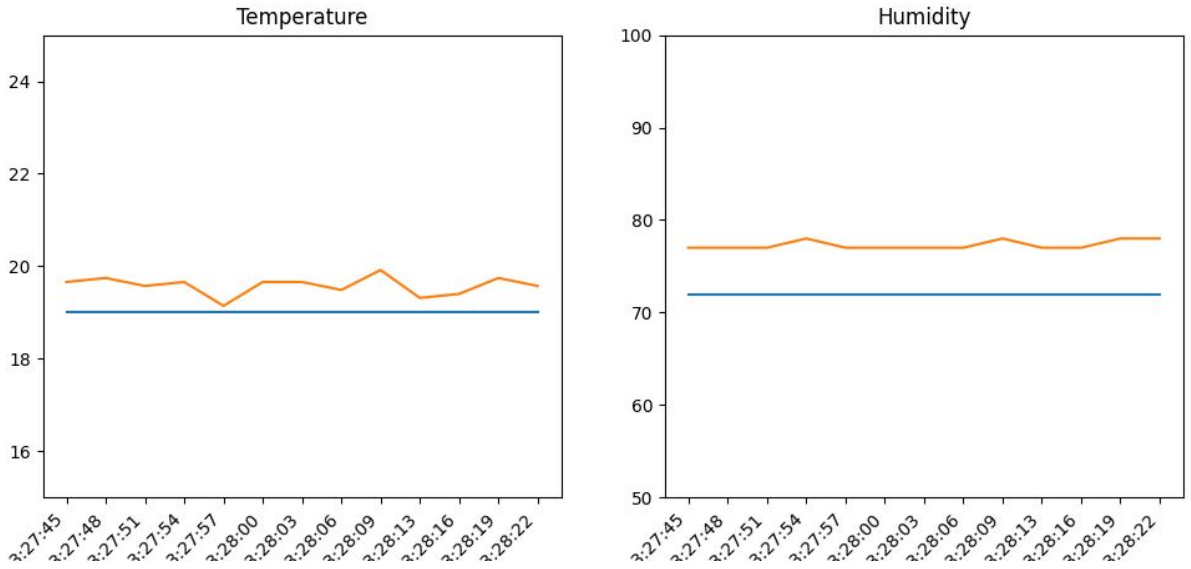
An MQTT server is running on the cloud to act as a broker between sensor nodes and the client. For the time being, the MQTT server performs no processing on the incoming data.

Client

On the other side of the internet a user/moderator can run the provided client to get the realtime data from sensor nodes, once the client starts it connects to the MQTT server and listens for a certain topic.

One improvement that could be made is enabling user accounts and allowing client programs to subscribe only to their corresponding topics.

Whenever the client receives a message from the topic, it updates and displays a table containing the data received so far.



Data Storage

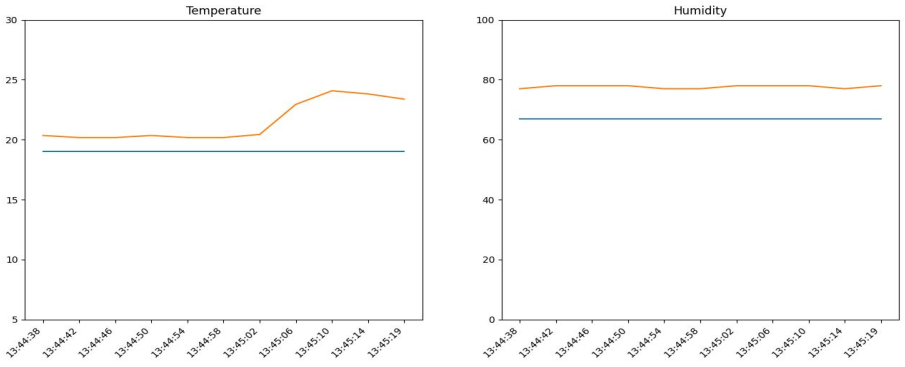
The client program stores the data it gets in a JSON format so later analysis can take place. The TinyDB Python module is used to act as a portable and low resources Database engine.


```
"1": {
        "data": {
            "air": {
                "humidity": 74,
                "temp": 19
            },
            "soil": {
                "humidity": 96,
                "temp": 19.48721
            }
        },
        "timestamp": 1609769567,
        "uuid": "aca24bec-3a0f-3c0a-80e8-baa33097e909"
    },
```


Results

Heat Test (soil)

In the following experiment, a heat source is applied to the temperature sensor, a change in the reading is noticed.

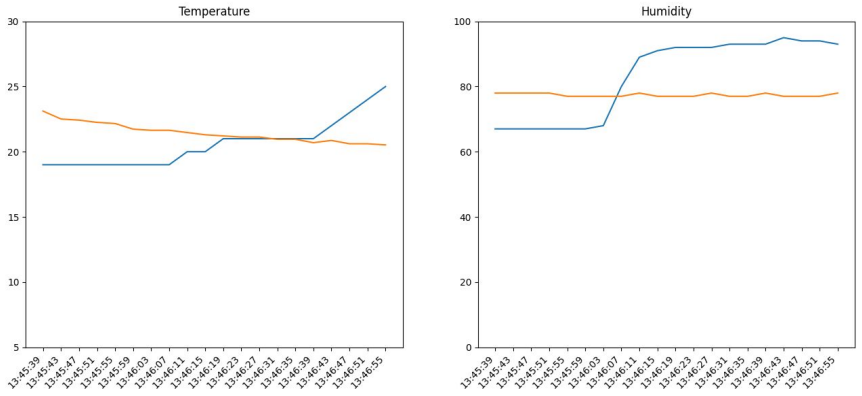


Heat & Humidity test (air)

By exposing the DHT-11 sensor to human breath which is both hot and humid we notice an increase in the temperature and humidity readings.

Explanation of the step-like response

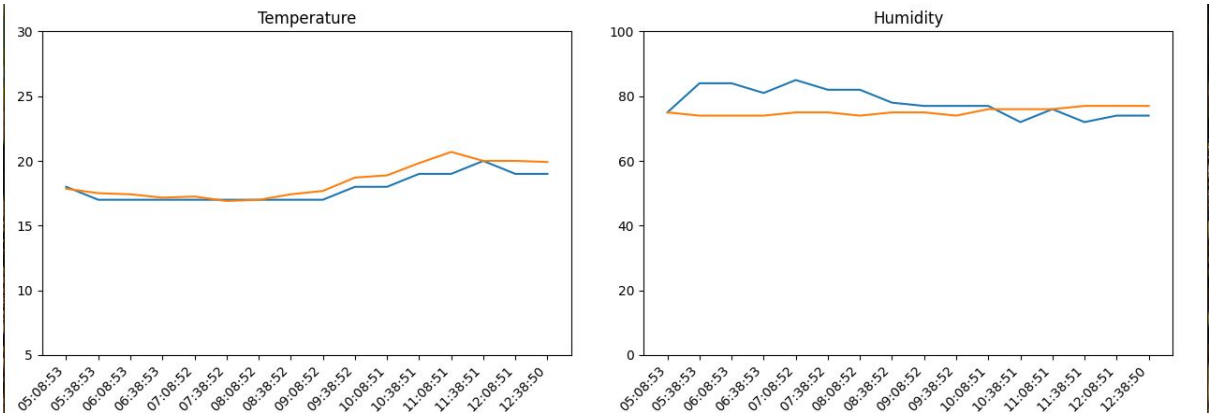
The line looks like a step because DHT-11 is a slow sensor and doesn't respond quickly to changes, so many readings are sent from the Sensor node with the same value.



Extended Period Observation

In the following data, we observe change over the period of 7 hours, from 5AM to 12PM, we notice that the temperature is rising while the humidity is falling down.

This is expected because as the sun is rising the temperature rises and the water resident in the air (humidity) becomes less and less, we notice the temperature decline after 11.38AM, which happens due to the accumulating clouds.




```
{'data': {'air': {'humidity': 84, 'temp': 17}, 'soil': {'humidity': 74, 'temp': 17.50775}}, 'timestamp': 1609825137, 'uuid': '0c5288e6-e989-3519-b7a6-e3a9c8fdcaf9'}
{'data': {'air': {'humidity': 84, 'temp': 17}, 'soil': {'humidity': 74, 'temp': 17.42166}}, 'timestamp': 1609826937, 'uuid': '0c5288e6-e989-3519-b7a6-e3a9c8fdcaf9'}
{'data': {'air': {'humidity': 81, 'temp': 17}, 'soil': {'humidity': 74, 'temp': 17.16332}}, 'timestamp': 1609828737, 'uuid': '0c5288e6-e989-3519-b7a6-e3a9c8fdcaf9'}
{'data': {'air': {'humidity': 85, 'temp': 17}, 'soil': {'humidity': 75, 'temp': 17.24944}}, 'timestamp': 1609830537, 'uuid': '0c5288e6-e989-3519-b7a6-e3a9c8fdcaf9'}
{'data': {'air': {'humidity': 82, 'temp': 17}, 'soil': {'humidity': 75, 'temp': 16.9049}}, 'timestamp': 1609832337, 'uuid': '0c5288e6-e989-3519-b7a6-e3a9c8fdcaf9'}
{'data': {'air': {'humidity': 82, 'temp': 17}, 'soil': {'humidity': 74, 'temp': 16.99105}}, 'timestamp': 1609834137, 'uuid': '0c5288e6-e989-3519-b7a6-e3a9c8fdcaf9'}
{'data': {'air': {'humidity': 78, 'temp': 17}, 'soil': {'humidity': 75, 'temp': 17.42166}}, 'timestamp': 1609835937, 'uuid': '0c5288e6-e989-3519-b7a6-e3a9c8fdcaf9'}
{'data': {'air': {'humidity': 77, 'temp': 17}, 'soil': {'humidity': 75, 'temp': 17.6799}}, 'timestamp': 1609837737, 'uuid': '0c5288e6-e989-3519-b7a6-e3a9c8fdcaf9'}
{'data': {'air': {'humidity': 77, 'temp': 18}, 'soil': {'humidity': 74, 'temp': 18.71258}}, 'timestamp': 1609839538, 'uuid': '0c5288e6-e989-3519-b7a6-e3a9c8fdcaf9'}
{'data': {'air': {'humidity': 77, 'temp': 18}, 'soil': {'humidity': 76, 'temp': 18.8847}}, 'timestamp': 1609841338, 'uuid': '0c5288e6-e989-3519-b7a6-e3a9c8fdcaf9'}
{'data': {'air': {'humidity': 72, 'temp': 19}, 'soil': {'humidity': 76, 'temp': 19.83169}}, 'timestamp': 1609843138, 'uuid': '0c5288e6-e989-3519-b7a6-e3a9c8fdcaf9'}
{'data': {'air': {'humidity': 76, 'temp': 19}, 'soil': {'humidity': 76, 'temp': 20.69369}}, 'timestamp': 1609844938, 'uuid': '0c5288e6-e989-3519-b7a6-e3a9c8fdcaf9'}
Connected with result code 0
{'data': {'air': {'humidity': 72, 'temp': 20}, 'soil': {'humidity': 77, 'temp': 20.00396}}, 'timestamp': 1609846738, 'uuid': '0c5288e6-e989-3519-b7a6-e3a9c8fdcaf9'}
{'data': {'air': {'humidity': 74, 'temp': 19}, 'soil': {'humidity': 77, 'temp': 20.00396}}, 'timestamp': 1609848538, 'uuid': '0c5288e6-e989-3519-b7a6-e3a9c8fdcaf9'}
{'data': {'air': {'humidity': 74, 'temp': 19}, 'soil': {'humidity': 77, 'temp': 19.91781}}, 'timestamp': 1609850338, 'uuid': '0c5288e6-e989-3519-b7a6-e3a9c8fdcaf9'}
```


Recommendations & Future Work

Automatic Node Discovery

The current configuration requires that the node knows the IP address of the fog node before running so it can connect with it. This can be eliminated if we make the server send UDP broadcast messages to all the members of the network, then sensor nodes can listen on the same broadcast port and know the IP address of the fog node.

This will allow us to add nodes on the fly and they'll automatically start communicating with the fog node without any extra setup to add nodes.

Signal Filtering

Currently, the sensors are queried only when the reading is about to be sent to the fog node, which might give unstable results.

A better approach is to use a decaying weight model by having multiple readings in short intervals and give higher weight to the readings at the time closest to reporting to the fog node and less weight to the nodes that are away from that time. This way we can avoid having faulty readings on the sending time.

Batch Updates

The fog node currently pushes readings to the MQTT server whenever a sensor node sends data, a better approach would be sending less frequent and larger packets to the MQTT server, which would improve energy efficiency.

User initiated actions [New]

After integrating the telegram API it's possible for the user to issue commands to the fog node (which in turn can control the Arduino) using the chat, if the user sends messages to the bot chat page in Telegram the code can read this message and act accordingly.

An example would be to water the plants if the user types "water plant" to the bot in the chat.

References

- Pitakphongmetha, Jumras, et al. "Internet of things for planting in smart farm hydroponics style." 2016 International Computer Science and Engineering Conference (ICSEC). IEEE, 2016.
- Harsharn S. Grewala, Basant Maheshwaria, Sophie E. Parksb,"Water and nutrient use efficiency of a low-cost hydroponic greenhouse for a cucumber crop: An Australian case study", Agricultural Water Management, vol. 98, pp.841–846, March 2011.
- Jovicich, E. & Cantliffe, D. & Simonne, E. & Stoffella, P.. (2007). Comparative water and fertilizer use efficiencies of two production systems for Cucumbers. Acta Horticulturae. 731. 235-242. 10.17660/ActaHortic.2007.731.32.

- Dr. B. Paulchamy, N. Balaji, S. Dinesh Pravatha, P. Harish Kumar,T. Joel Frederick, "A Novel Approach for Automating & Analyzing Hydroponic farms Using Internet of Things",International Journal of Scientific Research in Computer Science, Engineering and Information Technology, Volume 3,Issue 3,April 2018,pp.1230-1234.
- R.Rajkumar, R.Dharmaraj, ”A Novel Approach for Smart Hydroponic Farming Using IoT”, International Journal of Engineering Research in Computer Science and Engineering, Volume 5, Issue 5, May 2018,pp.18-23.
- Melchizedek I. Alipio, Allen Earl M. Dela Cruz, Jess David A. Doria and Rowena Maria S. Fruto,”A Smart Hydroponics Farming System Using Exact Inference in Bayesian Network”,2017 IEEE 6th Global Conference on Consumer Electronics (GCCE 2017).
- TinyDB