Smart Soil Monitoring and Irrigation System using IOT

Amt Sharma
Computer Engineering Department
San Jose State University, San Jose, CA, USA
amit.sharma02@sjsu.edu

Jyotsna Priya
Computer Engineering Department
San Jose State University, San Jose, CA, USA
jyotsna.priya@sjsu.edu

Abstract—**The irrigation system is one of the core components in agriculture, and manual tracking of soil moisture and watering requires significant staff-hours. With manual irrigation systems, plants may be under-watered or overwatered, leading to lower production, higher cost, and wastage of resources. With this project, we propose a smart automatic irrigation system using the Internet of Things (IoT). We have developed a prototype that tracks the soil moisture and ambient temperature using sensors and sends this data to the cloud for further analysis and action. This smart irrigation system has a simple notification service incorporated that sends email or message notification to the user whenever the moisture level falls below the defined threshold value. It also automatically turns on the watering system at the same time. We have also built a dashboard through which the user can track the real-time sensor data and see graphs of the last ten days of historical data. The historical data may provide an insight into the water needs of different crops in different weather conditions. Based on this analysis, the user can set threshold values for the automatic watering system. This prototype IoT application can be scaled up to support large agriculture farms for production improvement and cost reduction.**

Index Terms - Cloud computing, internet of things, smart agriculture, web dashboard

I. Introduction

With easily available IoT infrastructure: hardware (sensor, relays and micro controllers), cloud services and Wi-Fi, many applications and systems are being developed for different fields from medical to constructions. Agriculture is also one of the important fields, which needs more advancement. Still majority of the farmers use conventional ways for farming. With continuous population growth, demand for food is increasing and we have limited resources in the form of agriculture land, water and manpower. Advancements in agriculture system are required to fulfill this demand. Irrigation system is one of the main components in agriculture and manual tracking of soil moisture and watering requires many man hours. Also, with manual irrigation system, plants can be under-watered or overwatered which leads to low crop yield and higher cost as well as waste of the resources (workforce, water, farming land).

Here we are trying to automate this important component of agriculture: soil moisture tracking and irrigation. We have developed a prototype of such IoT system which tracks the soil moisture as well as ambient temperature and send this data to cloud for tracking and analysis. Also, it automatically turns on the irrigation system when soil moisture level goes below threshold. User can track soil moisture and temperature values on web-based dashboard and email notification is also sent to the user. With history of moisture and temperature data, user can understand the need of water for different corps in different weather conditions. Based on this analysis user can set threshold values for automatic watering system.

II. State-of-the-Art Review

This section discusses some of the work that has been done in this area.

In their paper [1], Devika et. al. discuss about a automatic irrigation system that they developed using AtMega328 microcontroller and Arduino. In their setup, they have used moisture sensor to measure the soil moisture level. When the soil gets dry, the automatic pump starts. This paper gave an overview on how to get started with a smart irrigation system for our project.

In their paper [2], Elijah et. al. discuss about advantages of integrating IoT with Data analytics over the cloud for the advancement of smart agriculture. They also discuss several benefits and challenges of the same. This paper motivated us to integrate a historical overview of our sensor data in the dashboard to provide an analytics option.

III. Sytem Design

This Soil moisture and temperature monitoring and automatic irrigation system is based on low-cost hardware and cloud technologies are used to reduce the storage cost and made the accessible anytime, anywhere.

A. Hardware Details:

1. Raspberry Pi 4: This is low-cost computing system which is well sufficient for this requirement. It is used for edge computing and sending data to the cloud through Wi-Fi.
2. Adafruit Soil sensor: STEMMA Soil sensor-12C from Adafruit is used here. It is a capacitive sensor which sense soil moisture and ambient temperature values. It provides soil moisture ranging from 200 (very dry) to 2000 (very wet). This works better than mostly available low-cost resistive style sensors, which get oxidized soon and their resistive value goes up. This needs re-calibration of the code as per these higher values [3].
3. 5v DC Relay: This is used to control the power to water pump as per the minimum and maximum threshold for soil moisture.
4. Water pump: 5V DC water pump for irrigation system.
5. Battery: 9V DC battery for pump.
6. Jumper wires for connection.

B. Software Details

To implement our project, we used the programming languages, packages, and libraries as listed below:

- HTML/CSS: Hypertext Markup Language (HTML) was used to develop the dashboard layout and Cascading Style sheets was used to style the web application. In addition to our CSS, we also used Bootstrap, and jQuery libraries to make the web UI better.
- Python3
  This project is mostly based on Python 3 scripting language. We used Python SDK to connect the Raspberry Pi to AWS IoT core service. We also wrote our program to connect the hardware to MongoDB. The backend code and APIs for the web application is also written in Python3.
- Apache WSGI server with Flask
  We used Flask server to handle the API request coming from the web application
- JavaScript
  We used JavaScript to write the code to generate graphs on our dashboard
- To generate graphs, we also used two libraries Highcharts and Raphael. Sensor history data is displayed using Highcharts JS library and real-time data is displayed using Raphael JS library.
- To connect to MongoDB cloud, we used pymongo and dnspython library

C. Cloud Services Details

1) AWS IoT core service

Amazon Web Services (AWS) IoT core service helps the edge devices such as Raspberry Pi, Arduino board, etc. to connect to cloud. If the device is able to connect to the IoT core service, we can define various rules to send and process the data that those devices send. IoT core can not only connect to AWS cloud services, but it also provides and option to connect to Web server endpoints which can be our own application as well. With the help of this service, we get a way to connect and command our IoT device from our application. A diagram as provided by AWS is given in Figure 1.

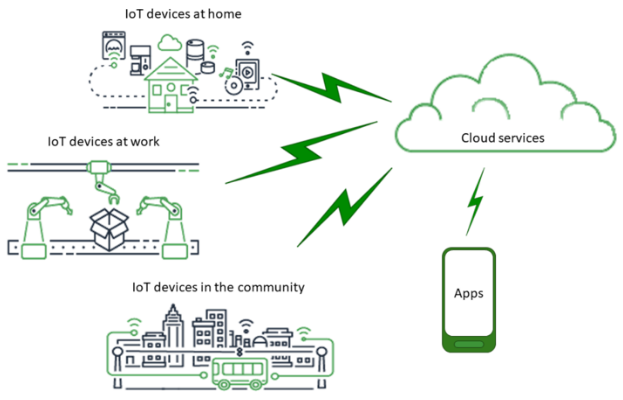


Figure 1. IoT devices and application connectivity through cloud

IoT core services use MQTT or WebSocket protocols to connect to the devices.

2) AWS SNS service

AWS Simple Notification Service (SNS) sends the data from a MQTT message to the destination provided by the user. It sends a push notification on the email id or phone number provided by the user.

3) AWS EC2 instance

AWS Elastic Compute Cloud (EC2) is a web service that provides secure, resizable compute capacity in the cloud. It provides options to choose from processor, storage, networking, operating system, and purchase model. For this project we used EC2 free tier instance with Ubuntu operating system.

4) MongoDB Atlas cloud

MongoDB Atlas is a global cloud database service for modern applications. It is a non-relational database which provides options to build clusters of databases. The databases have collections of documents stored in JSON format. MongoDB also provides options to connect our devices to store and send large data over the cloud. The driver application to connect to MongoDB cloud can be built in many languages such as NodeJS, Python, PHP, Perl, etc. The DB can also be accessed using Mongo Shell which can be installed on our local system.

D. *Circuit details:*

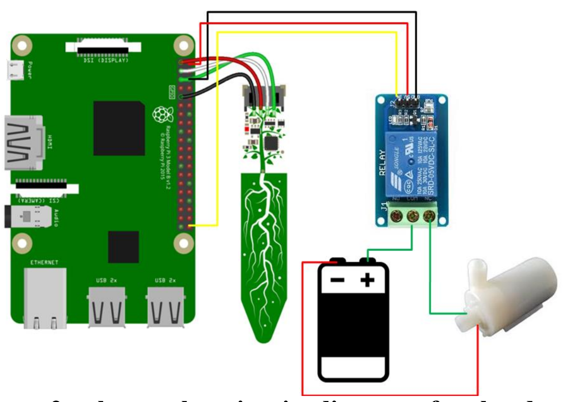


Figure 2. shows the circuit diagram for hardware components.

Connection details of circuit diagram of Fig.2:

1. Raspberry Pi and Soil sensor connections [4]:
   - Pi 3V3 power (pin 1) to sensor VIN
   - Pi GND (pin 9) to sensor GND
   - Pi SCL (pin 5) to sensor SCL
   - Pi SDA (pin 3) to sensor SDA
2. Raspberry Pi to relay connections [6]:
   - Pi 5V power (pin 2) to relay DC+
   - Pi GND (pin 6) to relay DC-
   - Pi GPIO 21 (pin 40) to relay IN
3. Relay, water pump and battery connections [6]:
   - Relay COM to battery +
   - Relay NC (not close) to water pump
   - Battery + to water pump

E. *Project Architecture:*

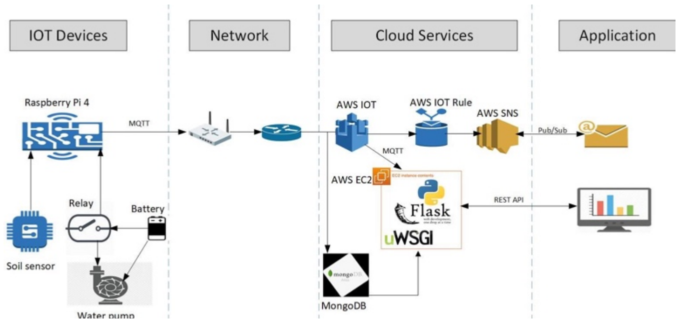


Figure 3. show architecture diagram of the project

As per the architecture diagram of Fig. 3, hardware (IoT devices) is connected as per the circuit diagram and Raspberry Pin is connected to the internet through Wi-Fi. Python script runs on the Pi and collects soil moisture and ambient temperature values from the sensor. Pi controls the water pump as per the defined moisture threshold value. Also, these sensor values and time stamps are sent to AWS and MongoDB Atlas. Data is sent to AWS using MOTT protocol and to MongoDB using TCP/IP socket. In case of moisture values reaches below threshold, AWS IoT rule and AWS SNS are used for notification email and SMS. Python Flash based web application for user dashboard runs on AWS EC2 instance which fetches data from AWS IoT and MongoDB. Details of these python script, cloud services and web application are provided in methodology section.

IV. Methodology

A. *Raspberry Pi implementation*

Python script runs on Raspberry pi collects sensor data and send it to the cloud. It also contains the logic of turning on and off of water pump. If moisture value goes below from threshold (set as 370 mu) it turns on the water pump and turn it off once water pumping duration is over. Flow diagram of Fig. 4 shows the logic for controlling water pump.

1) *Flow diagrams for water pump control:*

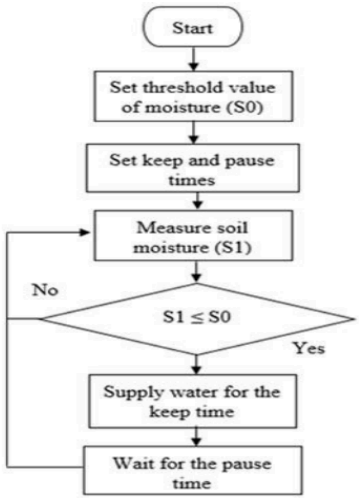
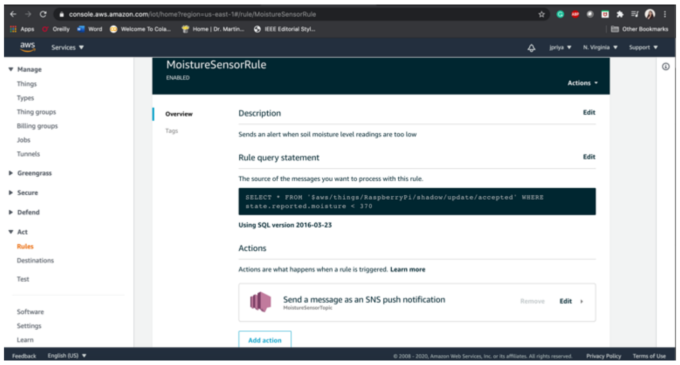

Start
Set threshold value of moisture (S0)
Set keep and pause times
Measure soil moisture (S1)
No
S1 ≤ S0
Yes
Supply water for the keep time
Wait for the pause time

Figure 4. Flow chart of water pump control for irrigation

B. *IoT rules*

1) *SNS rules overview*

The AWS SNS service can subscribe to MQTT protocol messages and act accordingly. We created a policy that subscribe to the topic that captures the value of moisture level. In our case whenever the moisture level drops below 370 MU, the user is notified via message as well as email. We enabled both option to test our program is working fine. In a real world setup, the user can avail any one of it. Figure 5 shows the rule for AWS SNS service. Figure 6 shows the email and message subscription of the user.


console.aws.amazon.com/iot/home?region=us-east-1#/rule/MoistureSensorRule

MoistureSensorRule
ENABLED

Description
Sends an alert when soil moisture level readings are too low

Rule query statement
The source of the messages you want to process with this rule.

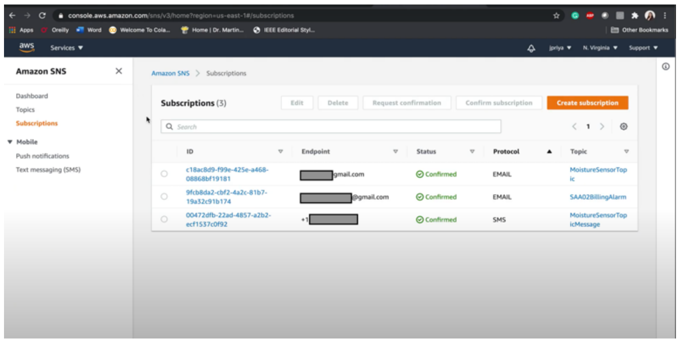
```
SELECT * FROM '$aws/things/RaspberryPi/shadow/update/accepted' WHERE
state.reported.moisture < 370
```

Using SQL version 2016-03-23

Actions
Actions are what happens when a rule is triggered. Learn more

Send a message as an SNS push notification
MoistureSensorTopic

Figure 5. AWS IoT rule


console.aws.amazon.com/sns/v3/home?region=us-east-1#/subscriptions

Amazon SNS > Subscriptions

Subscriptions (3)

ID	Endpoint	Status	Protocol	Topic
c18ac8d9-f99e-425e-a468-08868bf19181	[redacted]gmail.com	Confirmed	EMAIL	MoistureSensorTopic
9fcb8da2-cbf2-4a2c-81b7-19a32c91b174	[redacted]@gmail.com	Confirmed	EMAIL	SAA02BillingAlarm
00472dfb-22ad-4857-a2b2-ecf1537c0f92	+1[redacted]	Confirmed	SMS	MoistureSensorTopicMessage

Figure 6. AWS SNS services

2) AWS IoT policy overview:

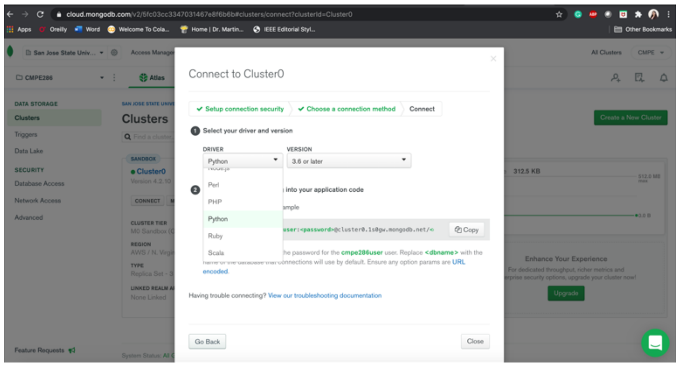
In AWS IoT core service, we created a Thing for our Raspberry Pi device. Then we created a policy to connect the Thing with the hardware which is shown in Figure 7.

```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:region:account:client/RaspberryPi"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/update",
        "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/delete",
        "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/get"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/update/accepted",
        "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/delete/accepted",
        "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/get/accepted",
        "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/update/rejected",
        "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/delete/rejected"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": [
        "arn:aws:iot:region:account:topicfilter/$aws/things/RaspberryPi/shadow/update/accepted",
        "arn:aws:iot:region:account:topicfilter/$aws/things/RaspberryPi/shadow/delete/accepted",
        "arn:aws:iot:region:account:topicfilter/$aws/things/RaspberryPi/shadow/get/accepted",
        "arn:aws:iot:region:account:topicfilter/$aws/things/RaspberryPi/shadow/update/rejected",
        "arn:aws:iot:region:account:topicfilter/$aws/things/RaspberryPi/shadow/delete/rejected"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:GetThingShadow",
        "iot:UpdateThingShadow",
        "iot:DeleteThingShadow"
      ],
      "Resource": "arn:aws:iot:region:account:thing/RaspberryPi"

    }
  ]
}
```

Figure 7. AWS IoT Policy

C. MongoDB setup

We connected MongoDB with the hardware using pymongo python library. We also used dnspython to connect using the encoded url provided by MongoDB for a individual users. Figure 8 shows a setup for the same.


cloud.mongodb.com/v2/5fc03cc3347031467e8f6b6b#clusters/connect?clusterId=Cluster0

Connect to Cluster0

✓ Setup connection security ✓ Choose a connection method Connect

1 Select your driver and version

DRIVER: Python
VERSION: 3.6 or later

2 into your application code

```
user:<password>@cluster0.1s0gw.mongodb.net/<
```

he password for the cmpe286user user. Replace <dbname> with the name of the database that connections will use by default. Ensure any option params are URL encoded.

Having trouble connecting? View our troubleshooting documentation

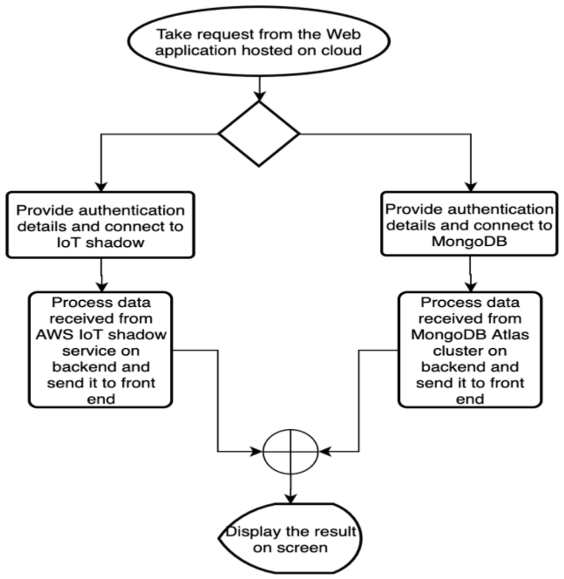
Figure 8. MongoDB connection setup

D. Dashboard implementation

To build a dashboard, we used HTML/CSS and JavaScript for the frontend and Flask and WSGI for the backend. For generating graphs we used Raphael JS and Highcharts.js libraries. The reason we used two tabs of dashboard is because the user might not need historical data option in private home setup. However, large historical sensor data can be useful for farmers who can not leave their automatic pumps ON all the time. Based on the last ten days of trend, they can decide at what range of time they can leave their pump connected to internet so that it starts automatically when the moisture level falls below a threshold value.

1) Flow Diagram

An overview of the process followed for the dashboard is given in Figure 9.



Take request from the Web application hosted on cloud
Provide authentication details and connect to IoT shadow
Provide authentication details and connect to MongoDB
Process data received from AWS IoT shadow service on backend and send it to front end
Process data received from MongoDB Atlas cluster on backend and send it to front end
Display the result on screen

Figure 9. Flowchart for the process followed for dashboard

V. Results

A. Raspberry Pi with Circuit and Sensors

As per circuit diagram of Figure 2 Raspberry Pi, soil sensor, relay and water pump connected. This setup is shown in Figure 10. For demo purpose we are using water glass as a replacement of wet soil and empty glass for dry soil. Water pump is kept inside the water source.

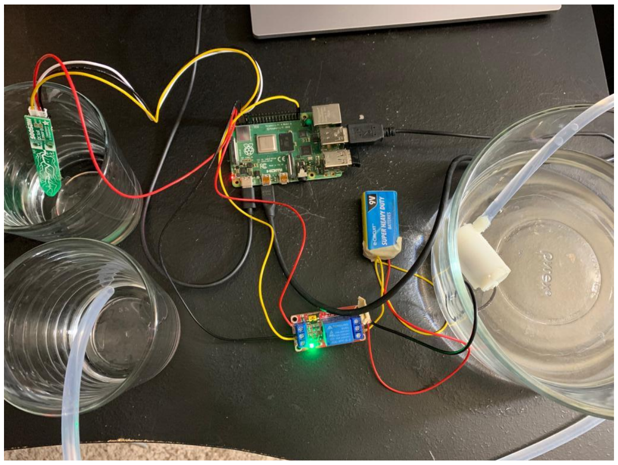
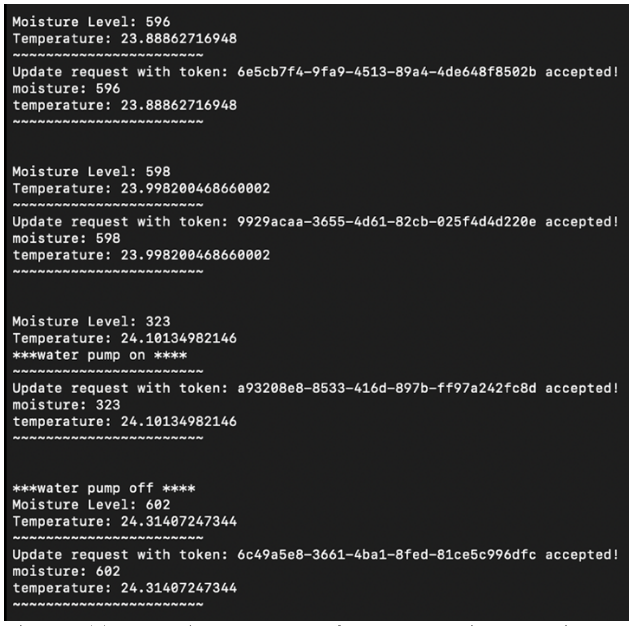


Figure 10. Raspberry Pi, sensor and actuator setup

B. Terminal Output of the Hardware

When python script is executed while keeping soil sensor inside water glass (Figure 10), it takes the moisture (596 mu) and temperature (23.8 C) values from the sensor and send it to cloud. When soil sensor is moved out from the water glass and kept into the empty glass, soil moisture values goes down till 323 mu, which is below threshold (370 mu). This lower moisture value turns on the water pump. Execution terminal of this script is shown in Figure 11.

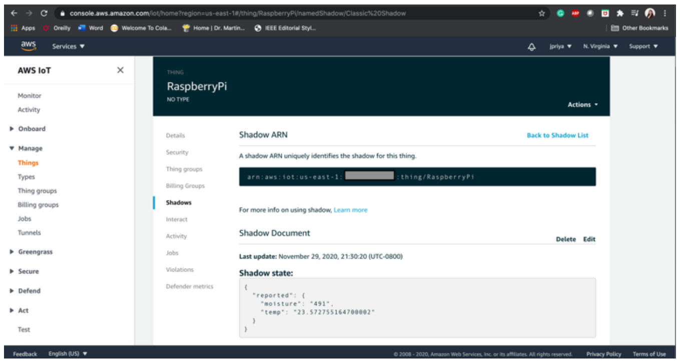


```
Moisture Level: 596
Temperature: 23.88862716948
~~~~~~~~~~~~~~~~~~~~~~
Update request with token: 6e5cb7f4-9fa9-4513-89a4-4de648f8502b accepted!
moisture: 596
temperature: 23.88862716948
~~~~~~~~~~~~~~~~~~~~~~


Moisture Level: 598
Temperature: 23.998200468660002
~~~~~~~~~~~~~~~~~~~~~~
Update request with token: 9929acaa-3655-4d61-82cb-025f4d4d220e accepted!
moisture: 598
temperature: 23.998200468660002
~~~~~~~~~~~~~~~~~~~~~~


Moisture Level: 323
Temperature: 24.10134982146
***water pump on ****
~~~~~~~~~~~~~~~~~~~~~~
Update request with token: a93208e8-8533-416d-897b-ff97a242fc8d accepted!
moisture: 323
temperature: 24.10134982146
~~~~~~~~~~~~~~~~~~~~~~


***water pump off ****
Moisture Level: 602
Temperature: 24.31407247344
~~~~~~~~~~~~~~~~~~~~~~
Update request with token: 6c49a5e8-3661-4ba1-8fed-81ce5c996dfc accepted!
moisture: 602
temperature: 24.31407247344
~~~~~~~~~~~~~~~~~~~~~~
```

Figure 11. Terminal output of python script running on Raspberry Pi

C. IoT shadow output

Figures 12 displays the latest data that is retrieved from the hardware. The data remains persistent in the shadow so if in case the sensors stop sending data the shadow will store the last time it got updated. By this, the dashboard can track if there is any problem with the hardware by checking the last updated data.



Shadow ARN

A shadow ARN uniquely identifies the shadow for this thing.

`arn:aws:iot:us-east-1:`[redacted]`:thing/RaspberryPi`

For more info on using shadow, Learn more

Shadow Document

Last update: November 29, 2020, 21:30:20 (UTC-0800)

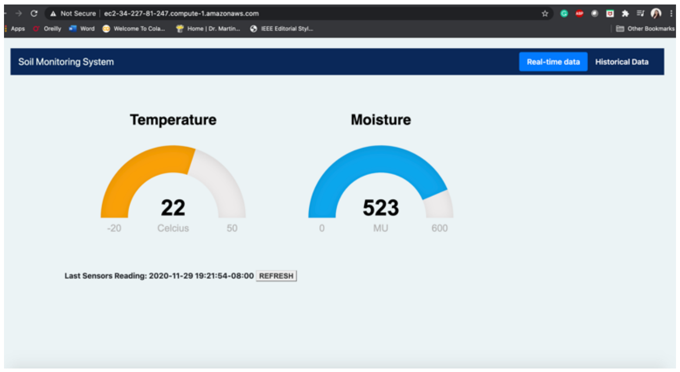
Shadow state:

```
{
  "reported": {
    "moisture": "491",
    "temp": "23.572755164700002"
  }
}
```

Figure 12. sensor data on AWS

D. Dashboard output

Figure 13 and 14 shows the real-time sensor data that is retrieved from the shadow and also the graphs of minimum moisture of last ten days. Second tab also shows the corresponding temperature and time at which the moisture level was the minimum.



Last Sensors Reading: 2020-11-29 19:21:54-08:00 REFRESH

Figure 13. The real-time data shown on the dashboard

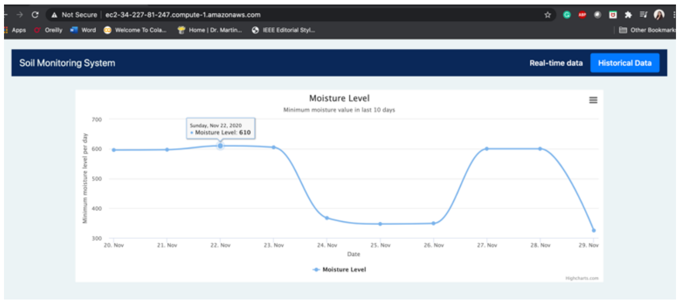


Figure 14. Historical data for minimum moisture level

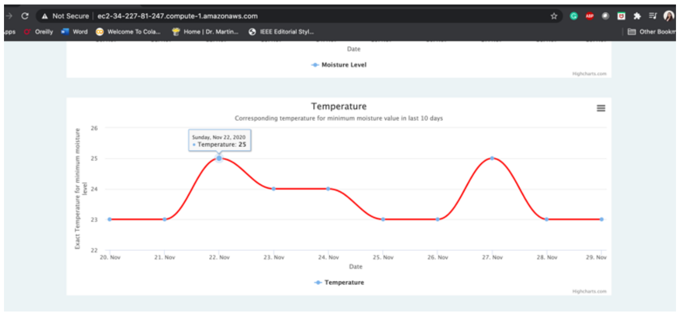


Figure 15. Historical data for temperature

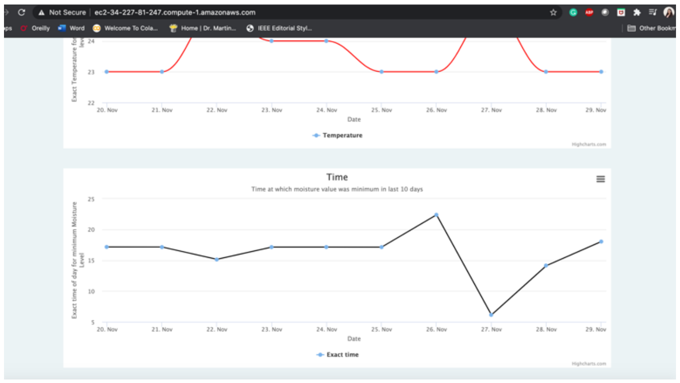
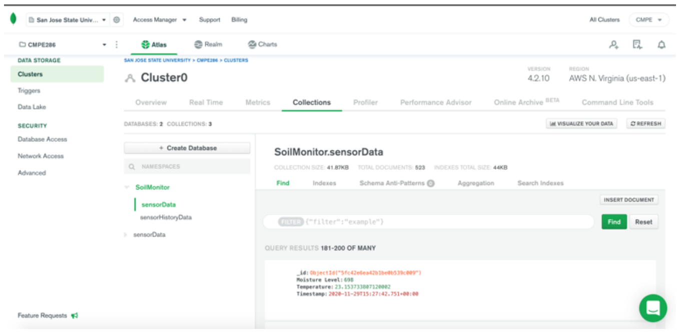


Figure 16. Historical data for time

E. MongoDB Data overview

Figure 17 shows the data that is stored on MongoDB on it dashboard. Figure 18 shows the queried data based on certain moisture level on Mongo shell terminal.



VERSION 4.2.10 REGION AWS N. Virginia (us-east-1)

DATABASES: 2 COLLECTIONS: 3

SoilMonitor.sensorData

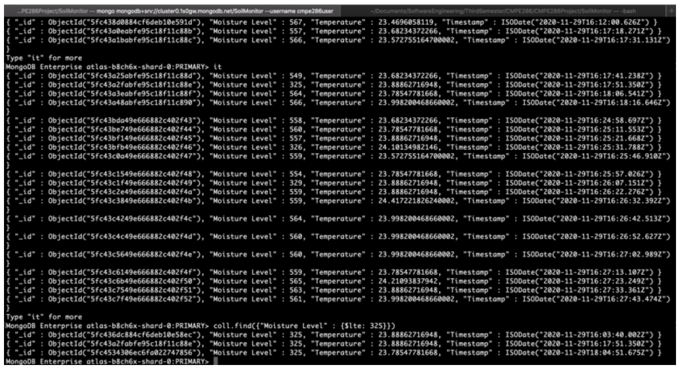
COLLECTION SIZE: 41.87KB TOTAL DOCUMENTS: 523 INDEXES TOTAL SIZE: 44KB

FILTER {"filter":"example"}

QUERY RESULTS 181-200 OF MANY

```
_id: ObjectId("5fc42e6ea42b1be0b539c009")
Moisture Level: 698
Temperature: 23.153733807120002
Timestamp: 2020-11-29T15:27:42.751+00:00
```

Figure 17. MongoDB data overview



```
...PE286Project/SoilMonitor — mongo mongodb+srv://cluster0.1s0gw.mongodb.net/SoilMonitor --username cmpe286user    ~/Documents/SoftwareEngineering/ThirdSemester/CMPE286/CMPE286Project/SoilMonitor — -bash
{ "_id" : ObjectId("5fc438d0884cf6deb10e591d"), "Moisture Level" : 567, "Temperature" : 23.4696058119, "Timestamp" : ISODate("2020-11-29T16:12:00.626Z") }
{ "_id" : ObjectId("5fc43a0eabfe95c18f11c88b"), "Moisture Level" : 557, "Temperature" : 23.68234372266, "Timestamp" : ISODate("2020-11-29T16:17:18.271Z") }
{ "_id" : ObjectId("5fc43a1babfe95c18f11c88c"), "Moisture Level" : 566, "Temperature" : 23.572755164700002, "Timestamp" : ISODate("2020-11-29T16:17:31.131Z")
}
Type "it" for more
MongoDB Enterprise atlas-b8ch6x-shard-0:PRIMARY> it
{ "_id" : ObjectId("5fc43a25abfe95c18f11c88d"), "Moisture Level" : 549, "Temperature" : 23.68234372266, "Timestamp" : ISODate("2020-11-29T16:17:41.238Z") }
{ "_id" : ObjectId("5fc43a2fabfe95c18f11c88e"), "Moisture Level" : 325, "Temperature" : 23.88862716948, "Timestamp" : ISODate("2020-11-29T16:17:51.350Z") }
{ "_id" : ObjectId("5fc43a3eabfe95c18f11c88f"), "Moisture Level" : 564, "Temperature" : 23.78547781668, "Timestamp" : ISODate("2020-11-29T16:18:06.541Z") }
{ "_id" : ObjectId("5fc43a48abfe95c18f11c890"), "Moisture Level" : 566, "Temperature" : 23.998200468660002, "Timestamp" : ISODate("2020-11-29T16:18:16.646Z")
}
{ "_id" : ObjectId("5fc43bda49e666882c402f43"), "Moisture Level" : 558, "Temperature" : 23.68234372266, "Timestamp" : ISODate("2020-11-29T16:24:58.697Z") }
{ "_id" : ObjectId("5fc43be749e666882c402f44"), "Moisture Level" : 560, "Temperature" : 23.78547781668, "Timestamp" : ISODate("2020-11-29T16:25:11.553Z") }
{ "_id" : ObjectId("5fc43bf149e666882c402f45"), "Moisture Level" : 557, "Temperature" : 23.88862716948, "Timestamp" : ISODate("2020-11-29T16:25:21.668Z") }
{ "_id" : ObjectId("5fc43bfb49e666882c402f46"), "Moisture Level" : 326, "Temperature" : 24.10134982146, "Timestamp" : ISODate("2020-11-29T16:25:31.788Z") }
{ "_id" : ObjectId("5fc43c0a49e666882c402f47"), "Moisture Level" : 559, "Temperature" : 23.572755164700002, "Timestamp" : ISODate("2020-11-29T16:25:46.910Z")
}
{ "_id" : ObjectId("5fc43c1549e666882c402f48"), "Moisture Level" : 554, "Temperature" : 23.78547781668, "Timestamp" : ISODate("2020-11-29T16:25:57.026Z") }
{ "_id" : ObjectId("5fc43c1f49e666882c402f49"), "Moisture Level" : 329, "Temperature" : 23.88862716948, "Timestamp" : ISODate("2020-11-29T16:26:07.151Z") }
{ "_id" : ObjectId("5fc43c2e49e666882c402f4a"), "Moisture Level" : 559, "Temperature" : 23.88862716948, "Timestamp" : ISODate("2020-11-29T16:26:22.276Z") }
{ "_id" : ObjectId("5fc43c3849e666882c402f4b"), "Moisture Level" : 559, "Temperature" : 24.417221826240002, "Timestamp" : ISODate("2020-11-29T16:26:32.392Z")
}
{ "_id" : ObjectId("5fc43c4249e666882c402f4c"), "Moisture Level" : 564, "Temperature" : 23.998200468660002, "Timestamp" : ISODate("2020-11-29T16:26:42.513Z")
}
{ "_id" : ObjectId("5fc43c4c49e666882c402f4d"), "Moisture Level" : 560, "Temperature" : 23.998200468660002, "Timestamp" : ISODate("2020-11-29T16:26:52.627Z")
}
{ "_id" : ObjectId("5fc43c5649e666882c402f4e"), "Moisture Level" : 560, "Temperature" : 23.998200468660002, "Timestamp" : ISODate("2020-11-29T16:27:02.989Z")
}
{ "_id" : ObjectId("5fc43c6149e666882c402f4f"), "Moisture Level" : 559, "Temperature" : 23.78547781668, "Timestamp" : ISODate("2020-11-29T16:27:13.107Z") }
{ "_id" : ObjectId("5fc43c6b49e666882c402f50"), "Moisture Level" : 565, "Temperature" : 24.21093837942, "Timestamp" : ISODate("2020-11-29T16:27:23.249Z") }
{ "_id" : ObjectId("5fc43c7549e666882c402f51"), "Moisture Level" : 563, "Temperature" : 23.88862716948, "Timestamp" : ISODate("2020-11-29T16:27:33.361Z") }
{ "_id" : ObjectId("5fc43c7f49e666882c402f52"), "Moisture Level" : 561, "Temperature" : 23.998200468660002, "Timestamp" : ISODate("2020-11-29T16:27:43.474Z")
}
Type "it" for more
MongoDB Enterprise atlas-b8ch6x-shard-0:PRIMARY> coll.find({"Moisture Level" : {$lte: 325}})
{ "_id" : ObjectId("5fc436dc884cf6deb10e58ec"), "Moisture Level" : 325, "Temperature" : 23.88862716948, "Timestamp" : ISODate("2020-11-29T16:03:40.002Z") }
{ "_id" : ObjectId("5fc43a2fabfe95c18f11c88e"), "Moisture Level" : 325, "Temperature" : 23.88862716948, "Timestamp" : ISODate("2020-11-29T16:17:51.350Z") }
{ "_id" : ObjectId("5fc4534306ec6fa022747856"), "Moisture Level" : 325, "Temperature" : 23.78547781668, "Timestamp" : ISODate("2020-11-29T18:04:51.675Z") }
MongoDB Enterprise atlas-b8ch6x-shard-0:PRIMARY>
```

Figure 18. Data as viewed on Mongo shell terminal

VI. Conclusion

This IoT system tracks the soil moisture as well as ambient temperature and send this data to cloud for tracking and analysis. Also, it automatically turns on the watering system when soil moisture level goes below threshold. User can track soil moisture and temperature values on web-based dashboard and email notification is also sent to the user. With history of moisture and temperature data, user can understand the need of water for different crops in different weather conditions. Based on this analysis user can set threshold values for automatic watering system. This protype IoT application can scaled up to support large agriculture farms for high crop yield and cost reduction.

VII. Future Scope

- Sensor to check water level in water source can also be added, to save the water pump in cases of low water level.
- Industrial application of this prototype at large-scale can result in reduced manual hours and increased efficiency of farmers. This application can be built as a SaaS application to handle multiple users and multiple plants per user.
- Dashboard historical data graphs can show multiple times at which moisture level was the minimum. As of now it shows the latest time.
- A machine learning model can be trained and embedded in the dashboard. The ML model can predict the most suitable time for irrigation.

VIII. Project Contribution

Amit Sharma:
- Raspberry Pi setup and integration of moisture and temperature sensors and sending data to cloud
- Instead of DC power solar power can be used for better utilization of power resources.
- Automatic pump setup and coding
- Simple mail service
- Connection of Hardware with MongoDB and AWS IoT

Jyotsna Priya
- Dashboard implementation for real-time data using IoT shadow, and MongoDB
- Hosting web application on cloud
- Simple message service
- Connection of Hardware with MongoDB and IoT
- Historical graphs implementation

Source-code: https://drive.google.com/drive/folders/12ZezKZvNIEjOkMLBq3Fewae17ous_mdV?usp=sharing

References

[1] C. M. Devika, K. Bose and S. Vijayalekshmy, "Automatic plant irrigation system using Arduino," 2017 IEEE International Conference on Circuits

and Systems (ICCS), Thiruvananthapuram, 2017, pp. 384-387, doi: 10.1109/ICCS1.2017.8326027.

[2] O. Elijah, T. A. Rahman, I. Orikumhi, C. Y. Leow and M. N. Hindia, "An Overview of Internet of Things (IoT) and Data Analytics in Agriculture: Benefits and Challenges," in *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3758-3773, Oct. 2018, doi: 10.1109/JIOT.2018.2844296.

[3] https://www.adafruit.com/product/4026.

[4] https://learn.adafruit.com/adafruit-stemma-soil-sensor-i2c-capacitive-moisture-sensor/python-circuitpython-test

[5] https://docs.aws.amazon.com/iot/latest/developerguide/iot-moisture-tutorial.html

[6] https://www.electronicshub.org/control-a-relay-using-raspberry-pi/