KENNESAW STATE UNIVERSITY

CPE 4040: Data Collection and Analysis, Spring 2023

Laboratory Report #7
Lab 7: A Data Project Using NodeRED, AWS IoT, and Streamlit

Team Members: Neal Jarzen & Clarence Barron

Electrical and Computer Engineering

Kennesaw State University

Faculty: Dr. Jeffrey L Yiin

Date of Lab Experiment: April 21, 2023

I. Objective

The objective of this lab was to build an end-to-end to collect and create a web dashboard using StreamLit. Also, we used the AWS IoT and S3 bucket cloud platforms to capture the data and display them on the created dashboard.

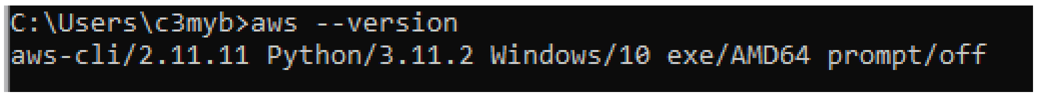
II. Material List

1. Raspberry Pi 3 or 4
2. Power supply adapter
3. Micro SD card (16+GB)
4. Ethernet cable
5. (optional) USB Keyboard, mouse and HDMI monitor or TV
6. Install Putty, Advanced IP Scanner and WinSC
7. Temperature Sensor
8. LED and resistors

III. Lab Procedures and Results

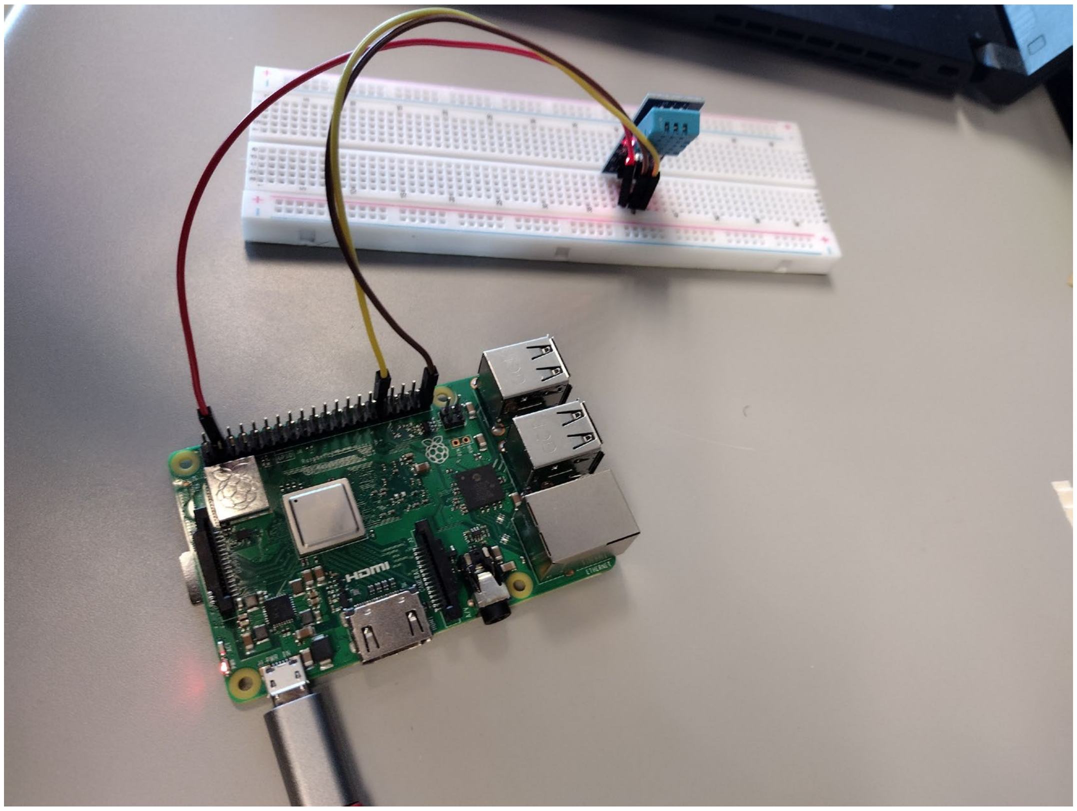
1) Plug in a boot the Raspberry Pi. Afterward, open the command prompt on your computer and make sure Python 3 is installed. If it is not installed, you can download it through https://www.python.org/downloads/.
2) Afterward, using "**pip install**" install the following libraries onto your system:
   a. Boto3
   b. Streamlit
   c. Plotly

   When installed, then go to https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html and follow the instructions to install AWS onto your computer. Afterward, type "**aws –version**" to make sure it is recognized. If it does not show up, you may have to reopen the terminal for the changes to take effect.



```
C:\Users\c3myb>aws --version
aws-cli/2.11.11 Python/3.11.2 Windows/10 exe/AMD64 prompt/off
```

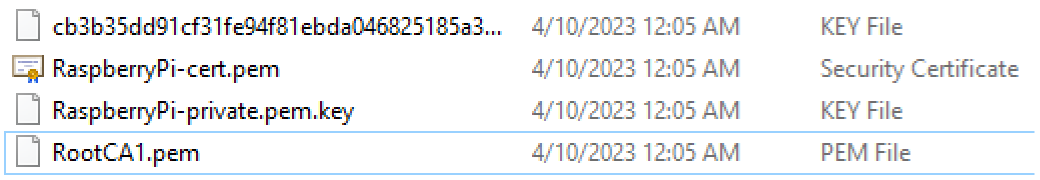
3) From Lab 4, set up the Raspberry Pi in the same configuration and use the same "**digitalOut.py**" script to verify that the sensor is working properly.



4) Just like in Lab 6, we will use the AWS cloud network. Create a free account and open "IoT Core" under "Services". We need to register the temperature sensor as the device we are going to use for this lab. Afterward, we need to create and download the device certificate, public key, private key, and Root CA. Once downloaded, create a new policy and attach the newly created certificates in the policy.

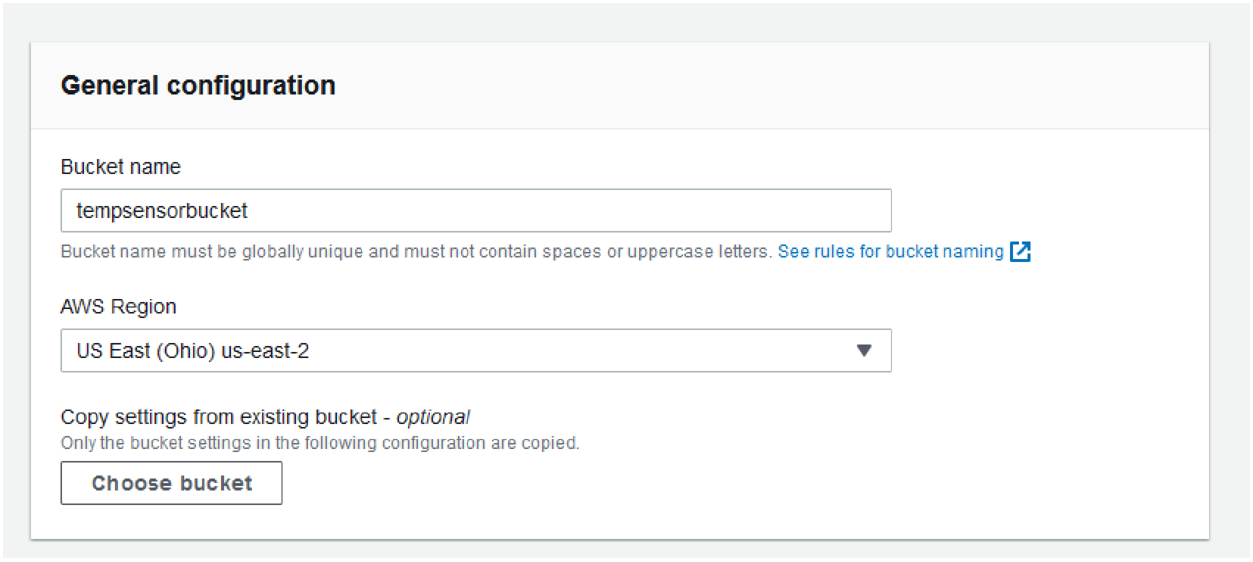
   After downloading the certificates and keys, change the following file names then import them to the Raspberry Pi under a new directory for your temperature sensor:

   - Device certificate: RaspberryPi-cert.pem
   - Private key: RaspberryPi-private.pem.key
   - Root CA: RootCA1.pem



cb3b35dd91cf31fe94f81ebda046825185a3...	4/10/2023 12:05 AM	KEY File
RaspberryPi-cert.pem	4/10/2023 12:05 AM	Security Certificate
RaspberryPi-private.pem.key	4/10/2023 12:05 AM	KEY File
RootCA1.pem	4/10/2023 12:05 AM	PEM File

5) For this lab, we are also going to make a S3 Bucket with in AWS. Navigate to the AWS Management system and search for "**S3**". Once found, go to the S3 dashboard, click "**Create Bucket**". Once on the page, enter a bucket name. (**Note:** It must a globally unique name). Under **Object Ownership**, click **ACLs Enabled.** Afterward, **Block Public Access settings for this bucket** and uncheck "**Block all public access**" then click "**Create Bucket**".


General configuration

Bucket name: tempsensorbucket

Bucket name must be globally unique and must not contain spaces or uppercase letters. See rules for bucket naming

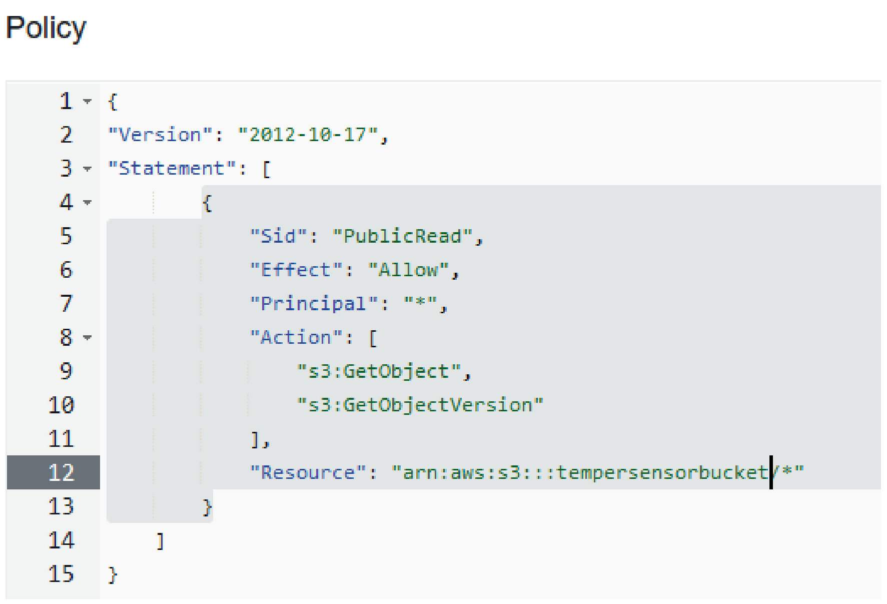
AWS Region: US East (Ohio) us-east-2

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

6) Under the **Permissions** tab, click edit next to the **Bucket policy**. Afterward, copy and paste the Bucket ARN within the code given below:

```
{
"Version": "2012-10-17",
"Statement": [
{
"Sid": "PublicRead",
"Effect": "Allow",
"Principal": "*",
"Action": [
"s3:GetObject",
"s3:GetObjectVersion"
],
"Resource": "arn:aws:s3:::<Paste-Your-Bucket-Name-Here>/*"
}
]
}
```

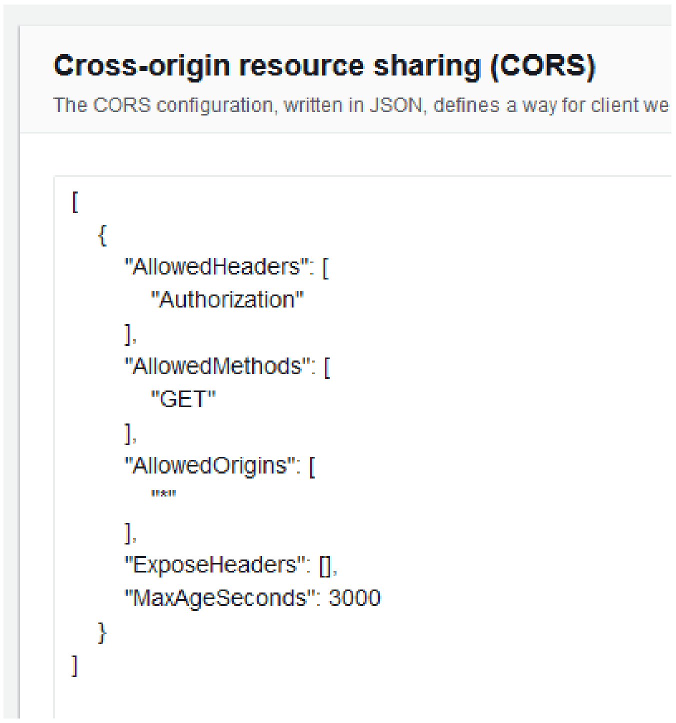


Policy

```
{
"Version": "2012-10-17",
"Statement": [
        {
            "Sid": "PublicRead",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::tempersensorbucket/*"
        }
    ]
}
```

Then paste the following code under CORS section and click save changes.

```
      [
{
"AllowedHeaders": [
"Authorization"
],
"AllowedMethods": [
"GET"
],
"AllowedOrigins": [
"*"
],
"ExposeHeaders": [],
"MaxAgeSeconds": 3000
}
]
```

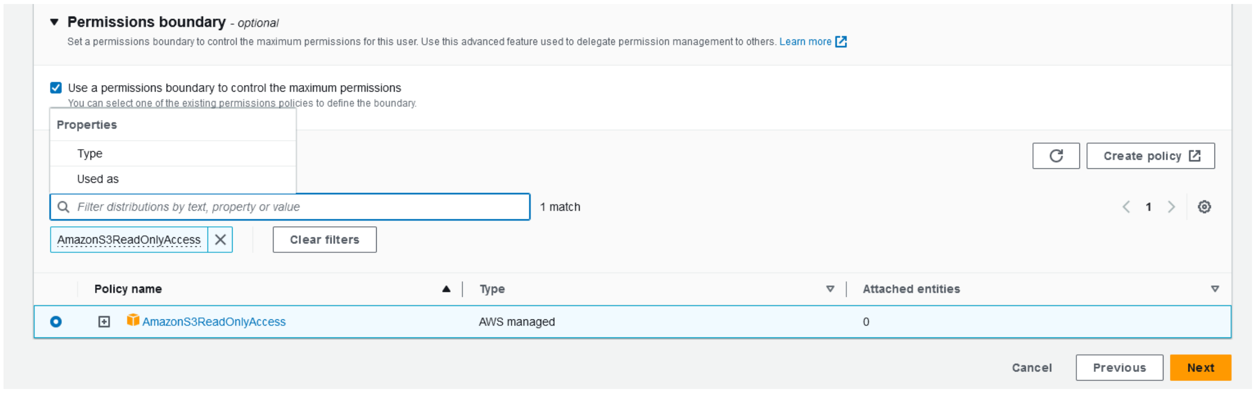


Cross-origin resource sharing (CORS)

The CORS configuration, written in JSON, defines a way for client we

```
[
    {
        "AllowedHeaders": [
            "Authorization"
        ],
        "AllowedMethods": [
            "GET"
        ],
        "AllowedOrigins": [
            "*"
        ],
        "ExposeHeaders": [],
        "MaxAgeSeconds": 3000
    }
]
```

7) Under **Access Control List**, go to **List** and **Read** and check "**Everyone (Public Access)**". Then acknowledge that it will be public, then click "**Save Changes**". After that, go to the **IoT Core** page then go to **Rules** under **Message Routing**, and create a new rule. Give it a name, then on the next page under **SQL Statement** and enter the following:



Permissions boundary - optional

Set a permissions boundary to control the maximum permissions for this user. Use this advanced feature used to delegate permission management to others. Learn more

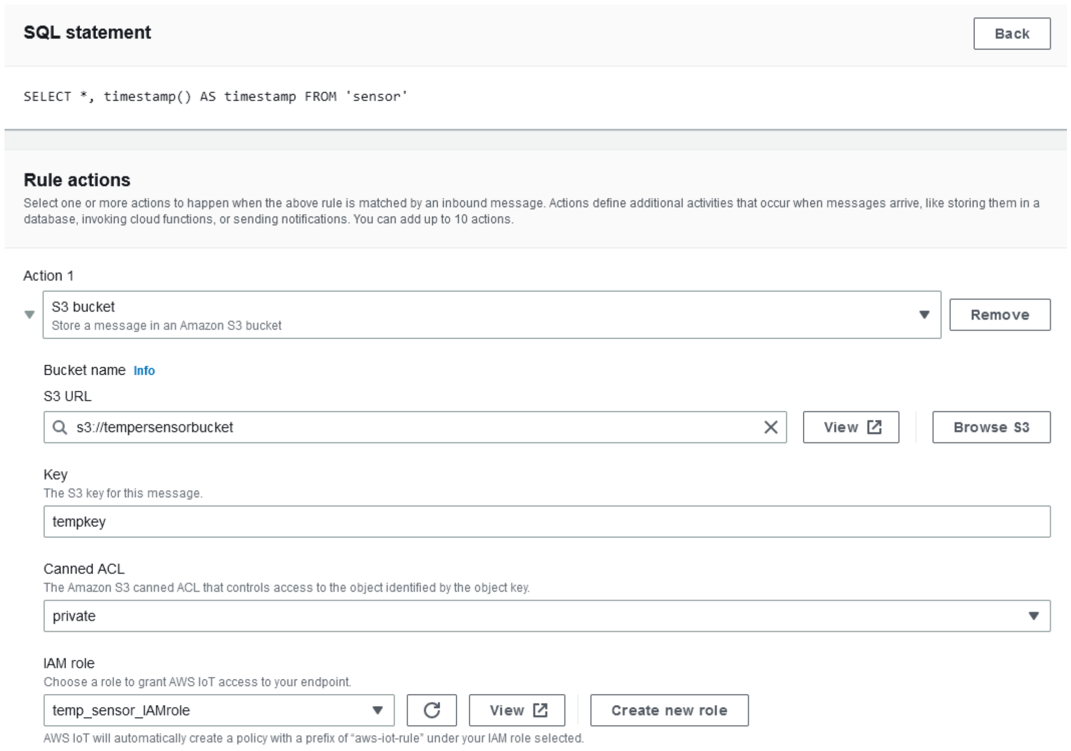
☑ Use a permissions boundary to control the maximum permissions

You can select one of the existing permissions policies to define the boundary.

	Policy name	Type	Attached entities
●	AmazonS3ReadOnlyAccess	AWS managed	0

```
SELECT *, timestamp() AS timestamp FROM '<Your-incoming-IoT-Topic-Here>'
```

After entering, go to **Rules Actions** and search for **S3 Bucket**. Then, go to your bucket name and under "**Key**" enter a key, then save it somewhere. You will need this key later.



SQL statement

```
SELECT *, timestamp() AS timestamp FROM 'sensor'
```

Rule actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. You can add up to 10 actions.

Action 1

S3 bucket
Store a message in an Amazon S3 bucket

Bucket name
S3 URL: s3://tempersensorbucket

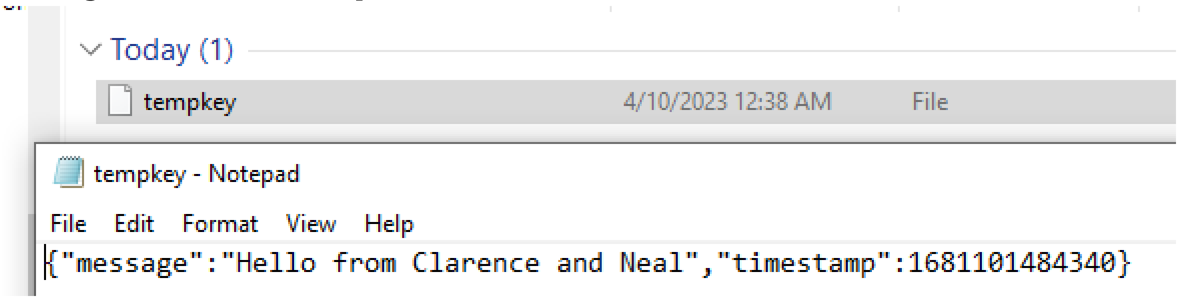
Key
The S3 key for this message.
tempkey

Canned ACL
The Amazon S3 canned ACL that controls access to the object identified by the object key.
private

IAM role
Choose a role to grant AWS IoT access to your endpoint.
temp_sensor_IAMrole

AWS IoT will automatically create a policy with a prefix of "aws-iot-rule" under your IAM role selected.

8) Go to the **IAM Role** and hit **Create new role**. Enter a name then create the role, then on the side bar click **MQTT test client** under **Test** go to **Publish a topic**. Enter the topic name of you have made, and enter a custom message in the field then hit public. Once published, go back to the S3 page and click onto your bucket, and, in the upper right corner, hit **Download** and open the file. You should see the message along with the timestamp.



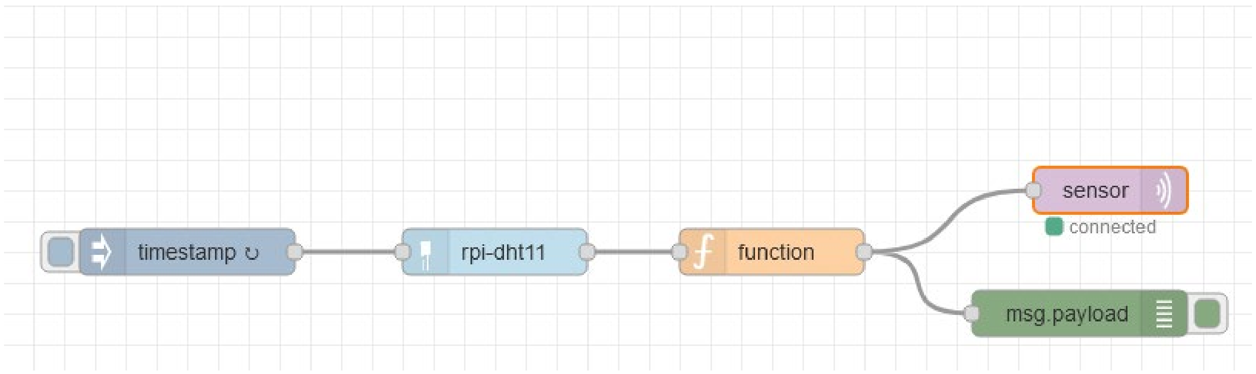
Today (1)

tempkey 4/10/2023 12:38 AM File

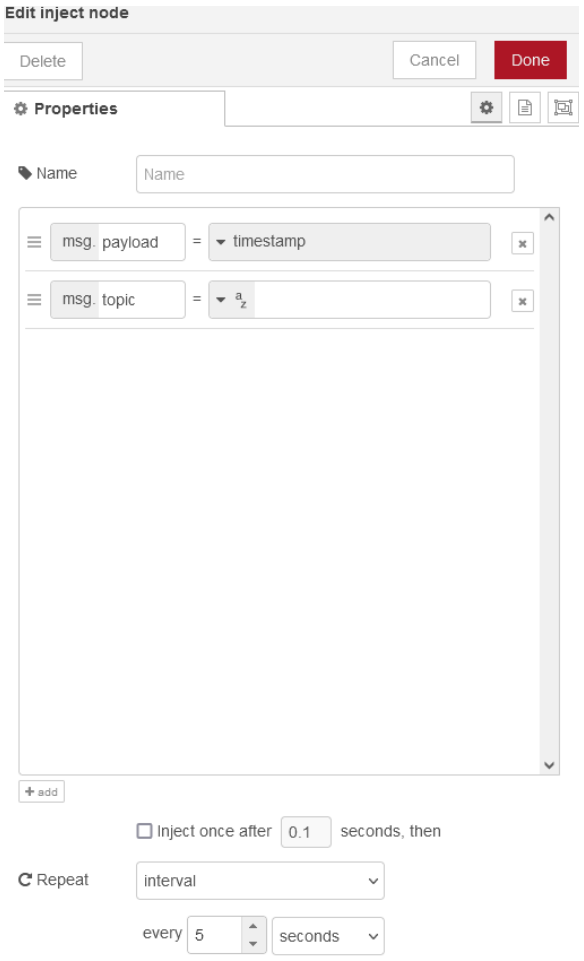
tempkey - Notepad

```
{"message":"Hello from Clarence and Neal","timestamp":1681101484340}
```

9) On the Pi, start up NodeRED and open the flow editior. In the Palette Manager, install **node-red-contrib-dht-sensor** (or if you are using a DHT-20 sensor install **node-red-contrib-aht20** instead). Once installed, create a new flow. Add an injection node, **rpi-dht22**, **function** node, **mqtt** out node, and **debug** node in the following configuration:



10) Once placed, edit the injection node to post every 5 seconds. In the **rpi-dht** node, use GPIO pin 12.


Edit inject node

Name: Name

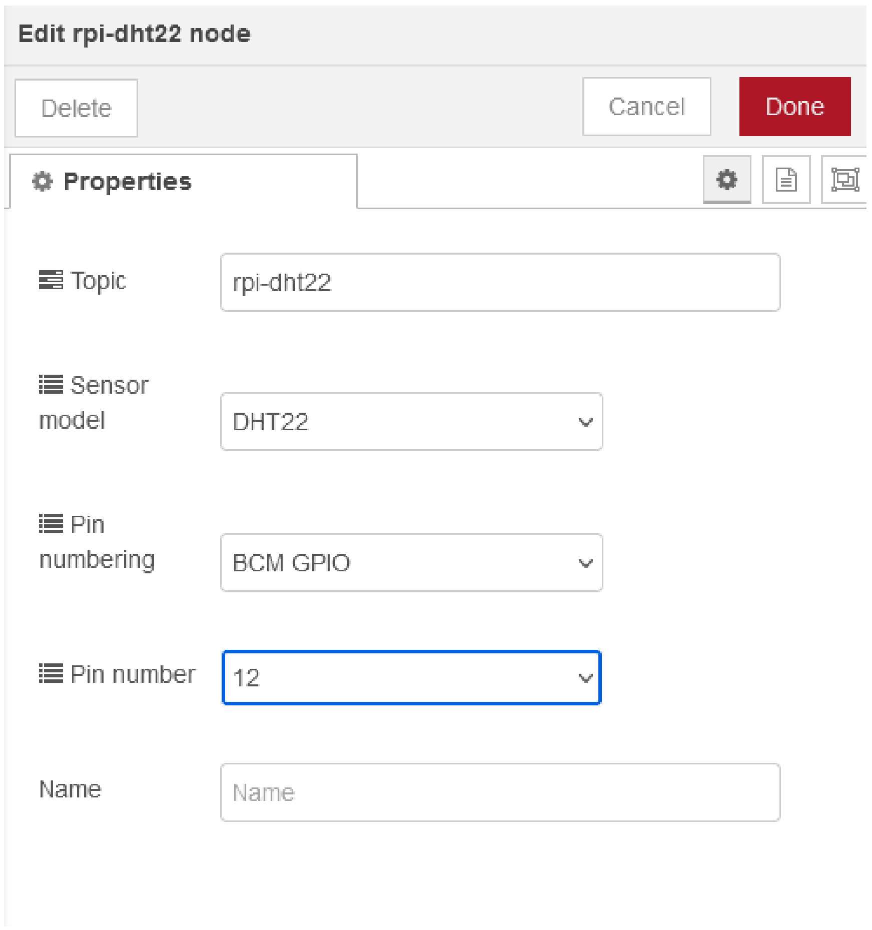
msg. payload = timestamp

msg. topic = ^a_z

☐ Inject once after 0.1 seconds, then

Repeat: interval

every 5 seconds


Edit rpi-dht22 node

Topic: rpi-dht22

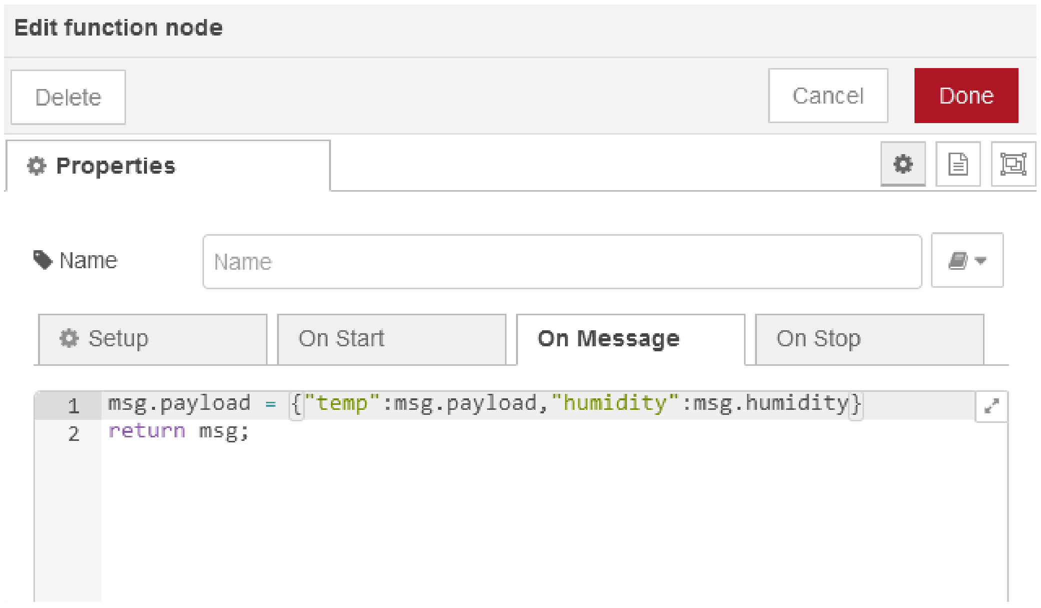
Sensor model: DHT22

Pin numbering: BCM GPIO

Pin number: 12

Name: Name

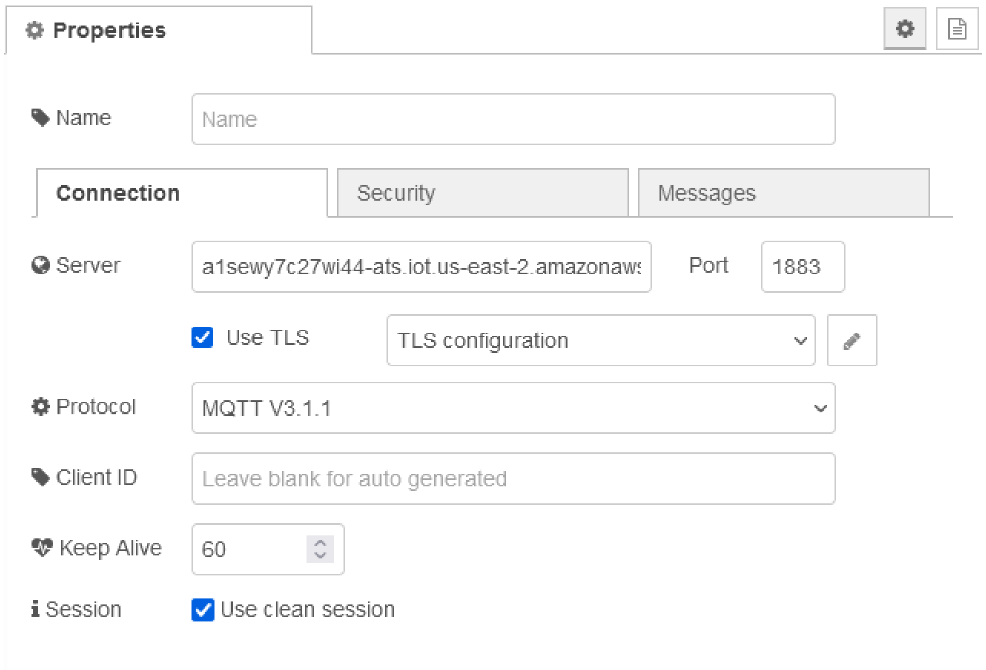
11), In the function node, add the following code:


Edit function node

Name: Name

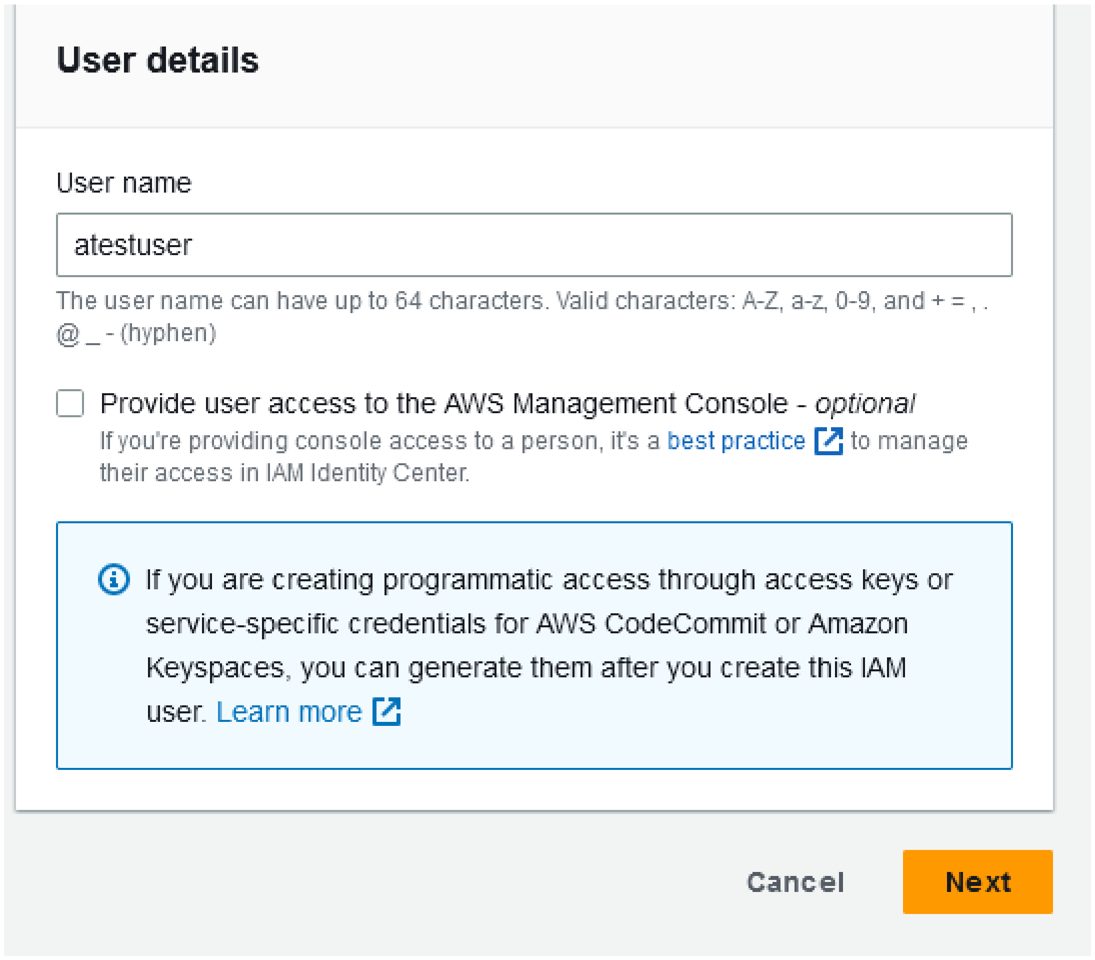
```
msg.payload = {"temp":msg.payload,"humidity":msg.humidity}
return msg;
```

12) Edit the **MQTT out** node. Click the edit button next to server and add a new MQTT broker. Then, navigate to AWS IoT > Settings > Device Data endpoint and copy the URL and paste it into the server. For the Client ID, add the device name. Check **Use TLS** and upload the AWS certificates, also in the topic field, make sure it is the same topic for the bucket. Then click deploy.



Name: Name
Server: a1sewy7c27wi44-ats.iot.us-east-2.amazonaws
Port: 1883
☑ Use TLS: TLS configuration
Protocol: MQTT V3.1.1
Client ID: Leave blank for auto generated
Keep Alive: 60
Session: ☑ Use clean session

13) Go back to the AWS Console and search **IAM** then click **Users** on the IAM menu. Go to **Add Users** and enter a name for the user and hit **Next**. Afterward, go to **Permissions Boundary** and select the check box. Then, search for **AmazonS3ReadOnlyAccess** and click next and create user.



User details

User name

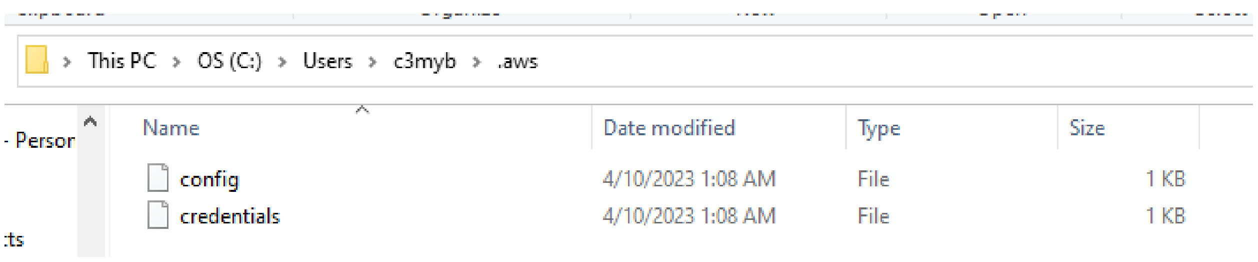
atestuser

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)

☐ Provide user access to the AWS Management Console - *optional*
If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.

If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. Learn more

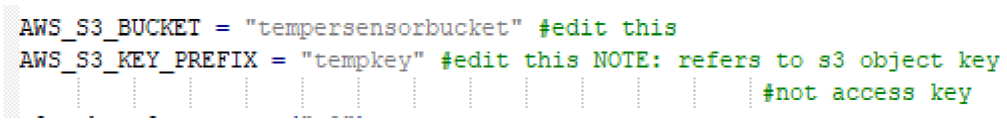
14) Under the new user, click **Security Credentials** and under **Access Keys**, go to **Create access key.** From there, you will go to **Access key best practices & alternatives** and copy the Access and Secret access key. Then, on your computer type "**aws configure**". Enter your keys when prompted and skip the other prompts by hitting enter. To verify, make sure it was created under C:\Users\[user]\.aws



This PC > OS (C:) > Users > c3myb > .aws

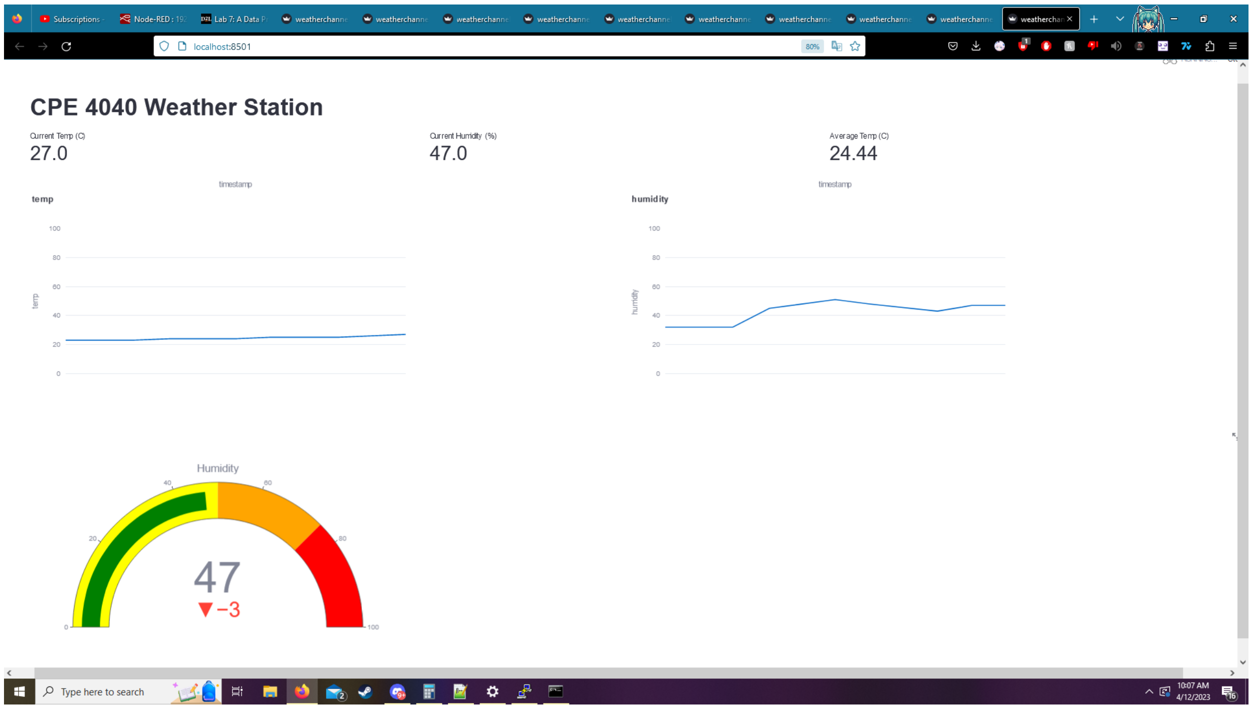
Name	Date modified	Type	Size
config	4/10/2023 1:08 AM	File	1 KB
credentials	4/10/2023 1:08 AM	File	1 KB

15) Download the **app.py** script and in a text editor. Change following lines to the name of your S3 bucket and your object key.



```
AWS_S3_BUCKET = "tempersensorbucket" #edit this
AWS_S3_KEY_PREFIX = "tempkey" #edit this NOTE: refers to s3 object key
                                                         #not access key
```

16) On the command prompt, enter **py streamlit run app.py** and it will automatically open a url to go to the web browser. There it will display the data in real time.



CPE 4040 Weather Station

Current Temp (C) 27.0

Current Humidity (%) 47.0

Average Temp (C) 24.44

IV. Conclusion

Setting up the lab itself was not hard. However, the tough part was trying to get the data to display data among the lines. Once that was dealt with, the rest was easy to set up. Also, this lab is a lab we would love to do again to get more data.