

CPE 4490

Intro to the Internet of Things (IoT)

Final Project: RezLok

Kwame Akuffo

Michael Young

John Herring (PM)

Andrew Bluhm

## **Project Background**

When deciding our project, we decided to use photoresistors as our sensor input since we had just used it in our Device Networks class for another project. While looking for a project idea from several DIY websites online, we discovered different options for smart locks and decided to test our knowledge by making our own IoT smart lock. This smart lock has buttons as photoresistors, and incorporates a Raspberry Pi which is connected to AWS to notify the user of when the door is being accessed.

## **Use Case**

Imagine your friend's house was robbed one night; they had a very basic lock that was picked by intruders and they wanted you, an IoT expert, to make them a secure lock, a RezLok, that would give them information about the lock from anywhere in the world.

## **Project Goal**

The goal for this project was for our team to create a simple, smart IoT door lock. We were able to accomplish this with the use of photoresistors which would act as a combination to unlock the door. More importantly, we wanted the photoresistor pattern to change each time the user wanted access to the building, and lastly, for the user to get informed of this new password via email.

## Materials

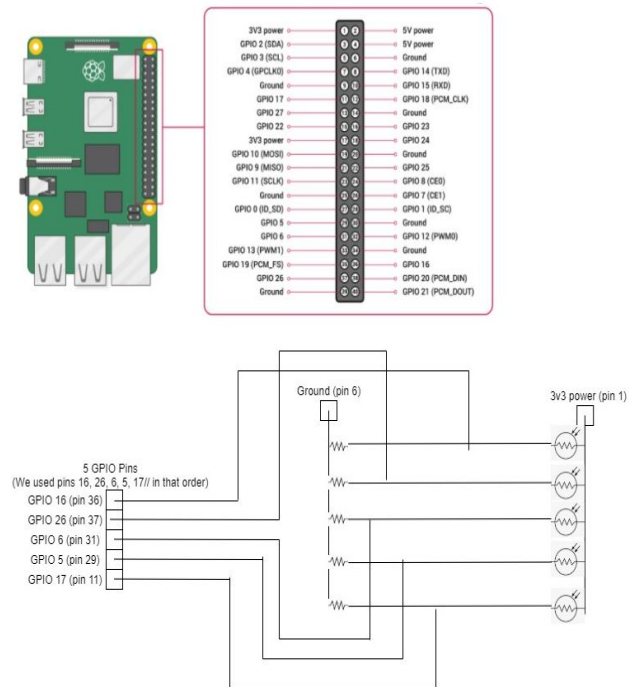
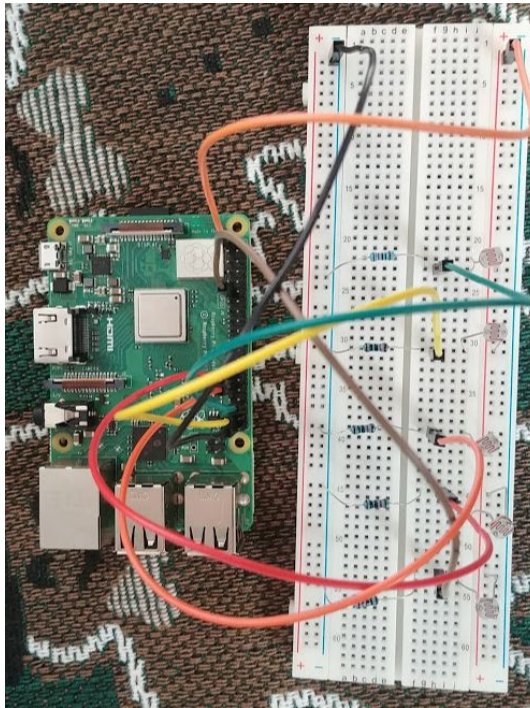
Materials required for this project are as follows:

- Raspberry Pi (for consistency model 3 B+ or higher version)
- Breadboard
- 5 Photoresistors
- 5 x 10K resistors
- FileZilla
- 7 x Male to female jumper wires
- AWS account (used for IoT core and SNS)

## Detailed procedures

1. Using the required materials, build a circuit with five photoresistors similar to the circuit shown below:
  - i) Connect a wire from the positive strip of your breadboard to a 3.3V GPIO pin.
  - ii) Align 5 photoresistors vertically, each with one pin of the photoresistor in the positive strip and the other pin horizontal from the pin in the positive strip. This pin will be in the terminal rail of the breadboard.
  - iii) Then, connect a wire from each photoresistor pin in the terminal rail to a GPIO pin.  
In this circuit, we use GPIO pins 16, 26, 6, 5, and 17 for the first, second, third, fourth, and fifth photoresistor, respectively.
  - iv) Then, in series with each wire connected to the GPIO pin, connect a 10k resistor with the other end in a negative rail.

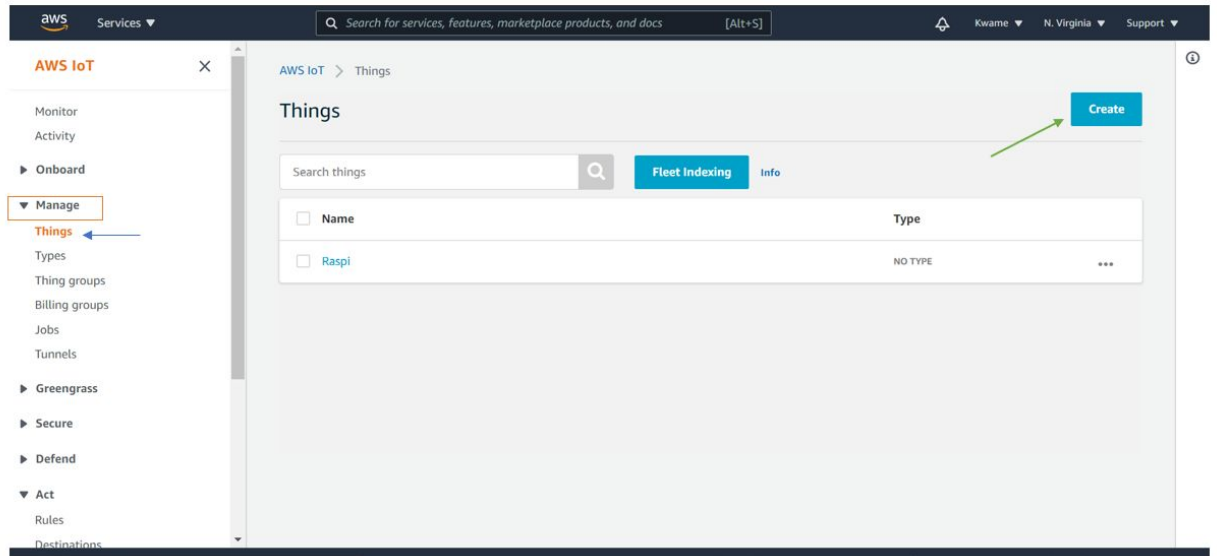
v) Ground this negative rail by connecting a wire from the rail to a GND GPIO pin on your raspberry pi. Your circuit is now complete.



2. AWS IoT Core implements the Message Queuing Telemetry Transport (MQTT) protocol by allowing us to publish to or subscribe from the Raspberry Pi. In this use case, we will subscribe to the Raspberry PI from the AWS IoT test environment. AWS also allows us to publish data to other AWS applications, such as AWS Lambda and DynamoDB Table. In this application, we will use the Amazon Simple Notification Service (SNS) to create a rule that sends a new password to the user's email each time access is enabled.

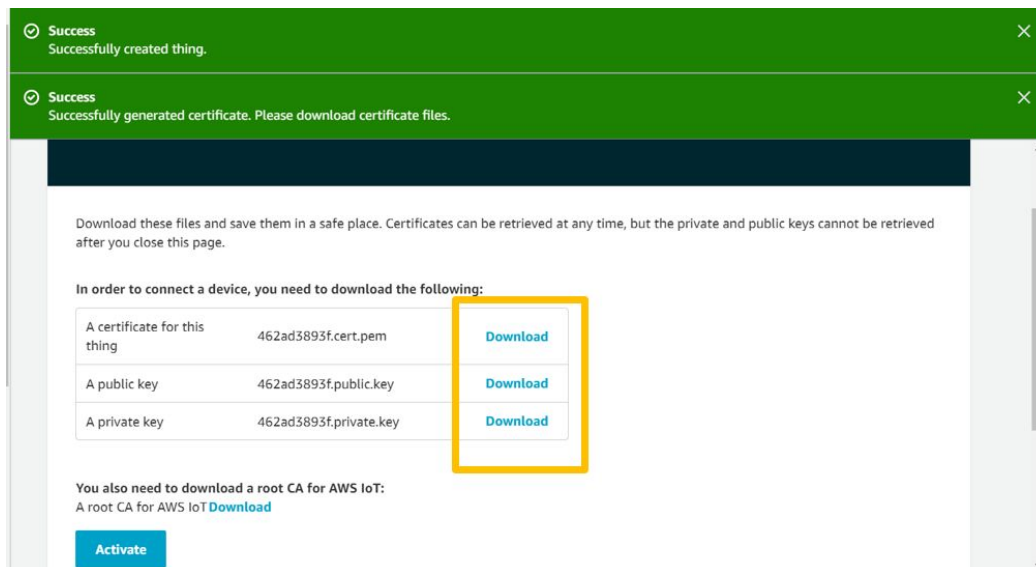
Let's set up AWS IoT:

- i) Create an AWS account. For this application, the free tier is sufficient.
- ii) Once your AWS account is created, navigate to AWS IoT Core, and create a thing:

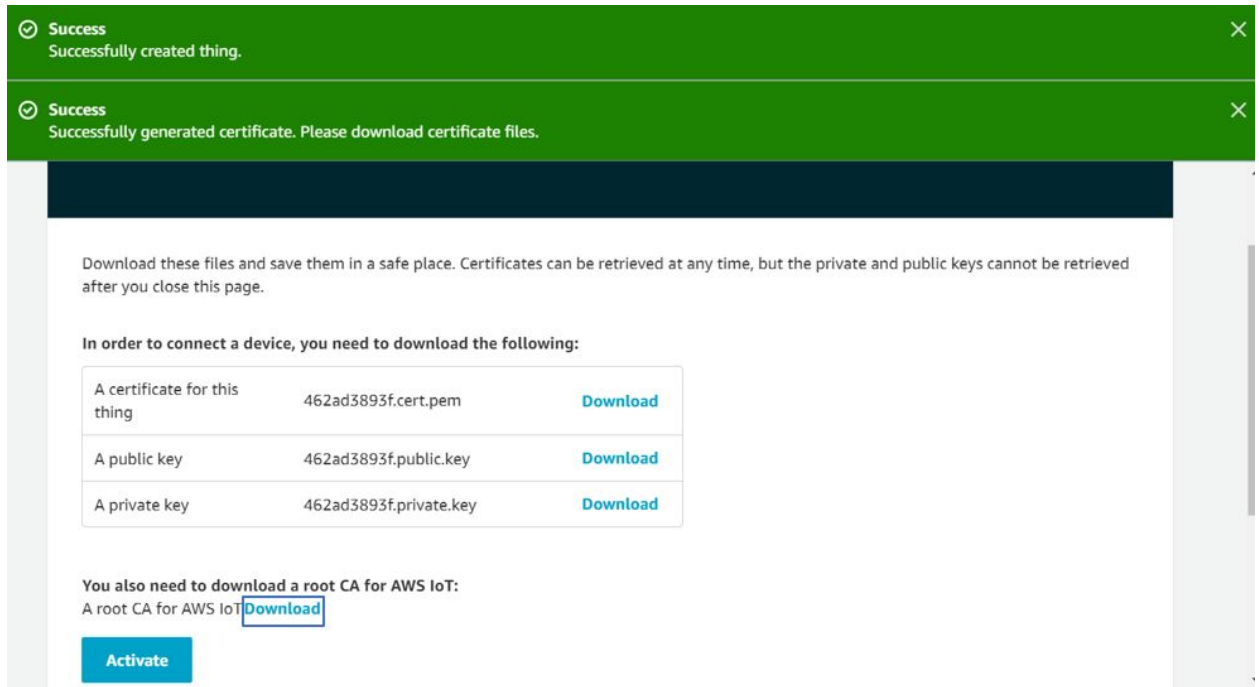


iii) Once you click on create, click on create a single thing. Give your thing a name, keep all the default information the same, and then click next. In our example (shown above), our thing is called Raspi.

iv) Next, add a certificate to your thing by clicking create certificate. Once a certificate is created, a “certificate for this thing,” a public key, and a private key should become available to you. Download all three keys. A picture of this is shown below:



v) Now, click on the very last download button shown below to download a root certificate authority.

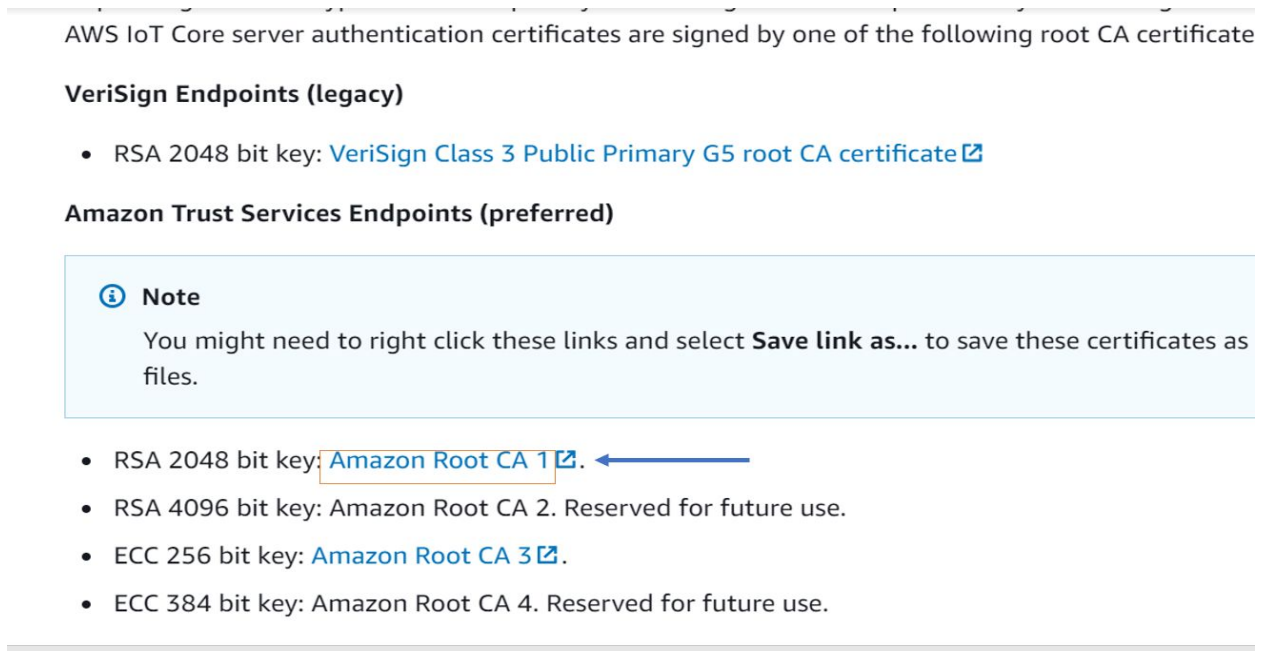


The screenshot shows two green success banners at the top. The first says "Success Successfully created thing." and the second says "Success Successfully generated certificate. Please download certificate files." Below these, a dark blue bar contains the text: "Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page." Underneath, it states: "In order to connect a device, you need to download the following:" followed by a table:

A certificate for this thing	462ad3893f.cert.pem	<a href="#">Download</a>
A public key	462ad3893f.public.key	<a href="#">Download</a>
A private key	462ad3893f.private.key	<a href="#">Download</a>

Below the table, it says: "You also need to download a root CA for AWS IoT:" followed by "A root CA for AWS IoT" and a [Download](#) button. At the bottom left is an [Activate](#) button.

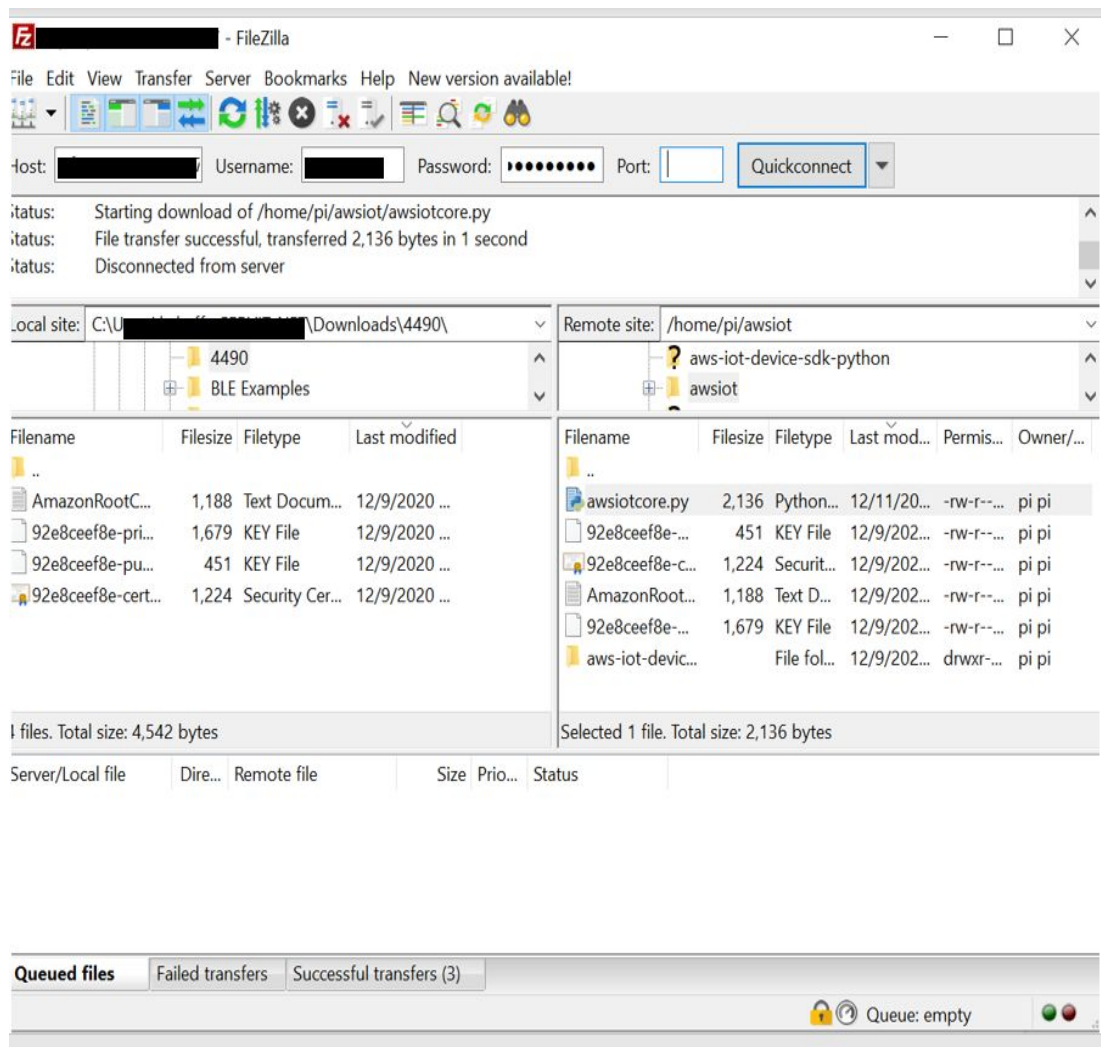
vi) Once the download button is clicked, you should be brought to a page that looks similar to the one below. Select Amazon Root CA1.



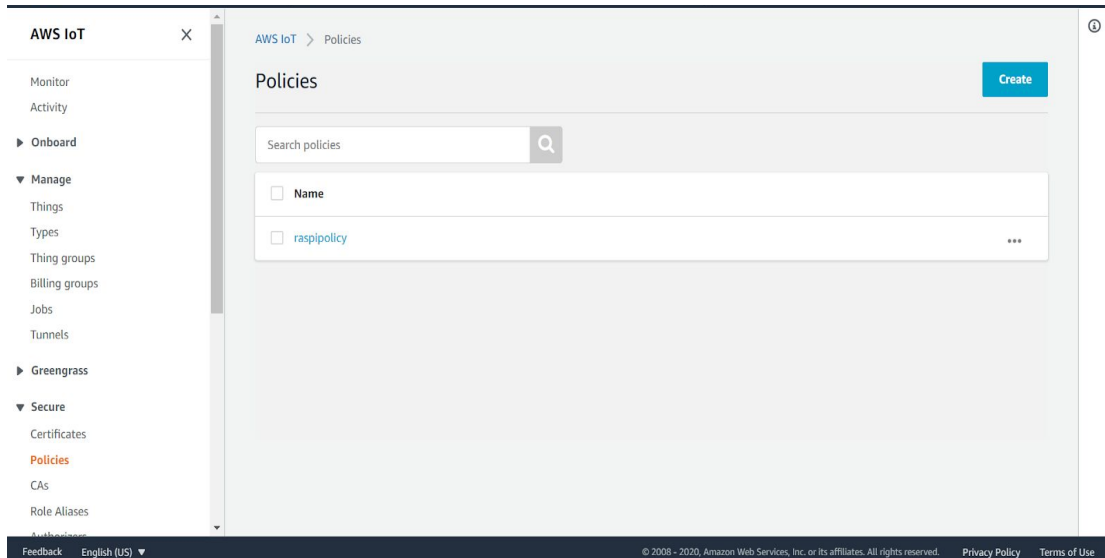
The screenshot shows the "AWS IoT Core server authentication certificates are signed by one of the following root CA certificate" page. It lists two categories: "VeriSign Endpoints (legacy)" and "Amazon Trust Services Endpoints (preferred)". Under "VeriSign Endpoints (legacy)", there is a bullet point: "RSA 2048 bit key: [VeriSign Class 3 Public Primary G5 root CA certificate](#)". Under "Amazon Trust Services Endpoints (preferred)", there is a note box that says: "Note You might need to right click these links and select **Save link as...** to save these certificates as files." Below the note box, there are four bullet points: "RSA 2048 bit key: [Amazon Root CA 1](#)", "RSA 4096 bit key: Amazon Root CA 2. Reserved for future use.", "ECC 256 bit key: [Amazon Root CA 3](#)", and "ECC 384 bit key: Amazon Root CA 4. Reserved for future use." A blue arrow points to the "Amazon Root CA 1" link.

vii) Once the certificate page opens. Download the page to a local folder on your PC. The saved document should be a .pem file.

viii) Now, we have to transfer all four downloaded files to our Raspberry Pi. To accomplish this, we use FileZilla. In the host section, put in the IP address of your Raspberry Pi and fill in the username and password. Since this is an SSH connection, the secure port is 22. Press connect, then, drag and drop files from your computer to your preferred location on the Raspberry Pi. Below is a screenshot similar to what your FileZilla should look like.



ix) Now we have to create a policy within AWS. Return to the AWS IoT Console web page, click on secure, then click on policy. A picture is shown below. Note that in the image below, a policy has already been created.



x) Click on Create (also shown in the picture above). Enter a name for this policy. Note that this name can be whatever you wish. Action describes whether this policy will be used for subscribing or publishing. Inserting `iot:*` in this field enables the policy for both subscribing and publishing. Clear the generated Resource ARN and insert a `*`. Doing this gives the policy access to any ARN and not just the one generated for us. Lastly, hit allow and then create. A picture of what creating your policy should look like is shown below:



Create a policy to define a set of authorized actions. You can authorize actions on one or more resources (things, topics, topic filters). To learn more about IoT policies go to the [AWS IoT Policies documentation page](#).

Name

Raspi

### Add statements

Policy statements define the types of actions that can be performed by a resource.

Advanced mode

Action

iot:\*

Resource ARN

\*

Effect

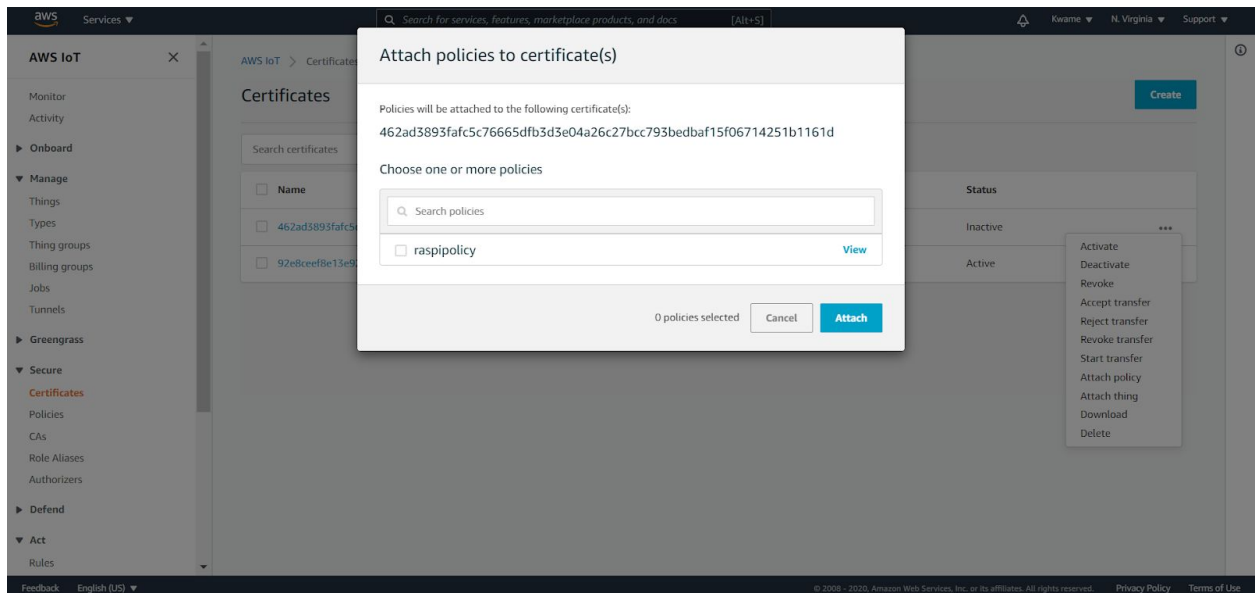
☒ Allow ☐ Deny

Remove

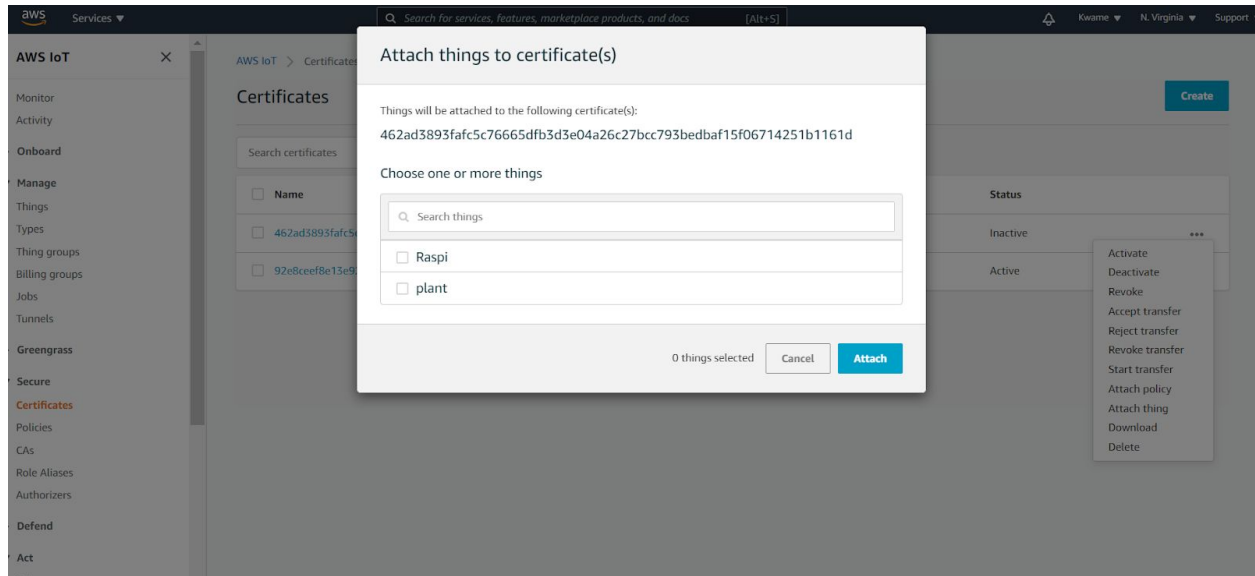
Add statement

Create

xi) Now we must associate this policy to our certificate. Return to the certificate tab from before, and hit the three dots on the certificate you created to attach policy. Here you can attach the policy that you recently created. Please see a picture below:

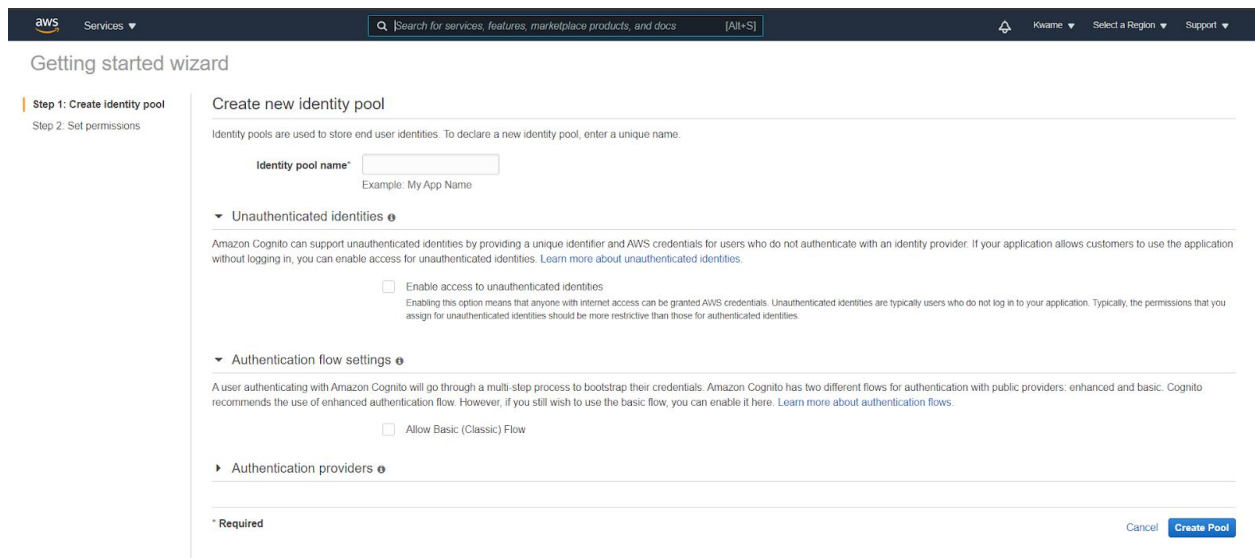


xii) Now we want to add a thing to this certificate, click on the three dots located on the certificate label and attach a thing:



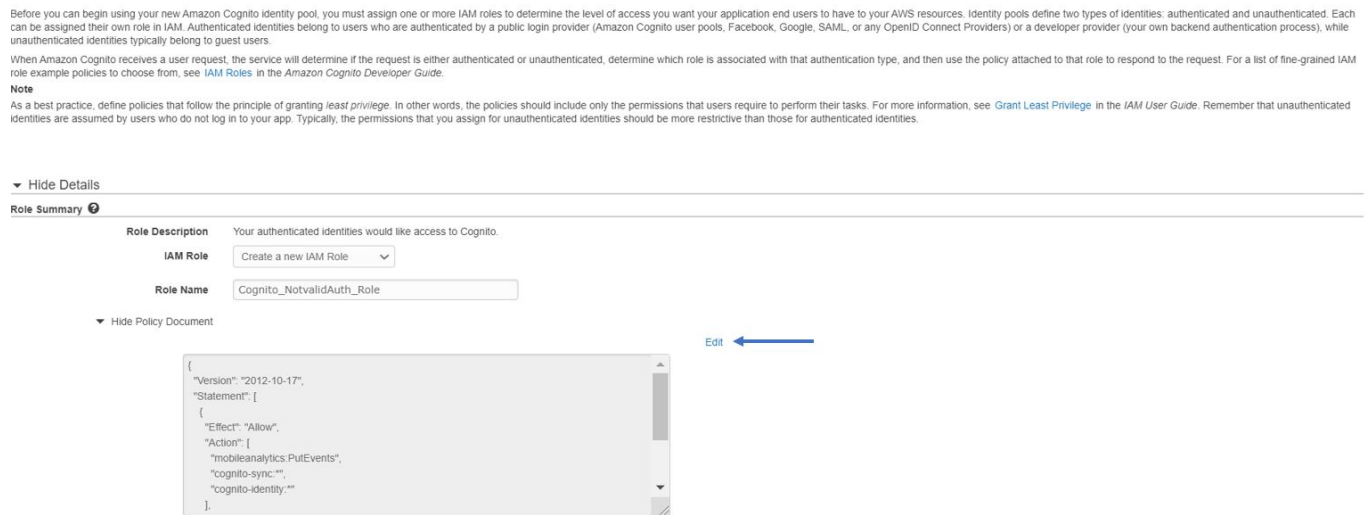
xiv) Make sure to click on the three dots to activate the certificate.

xv) Now, we will set up our cognito. In the search bar at the top of the AWS page, search for Cognito. Click on Manage Identity Pools, then on “create new” identity pool. You should now be on the following page:

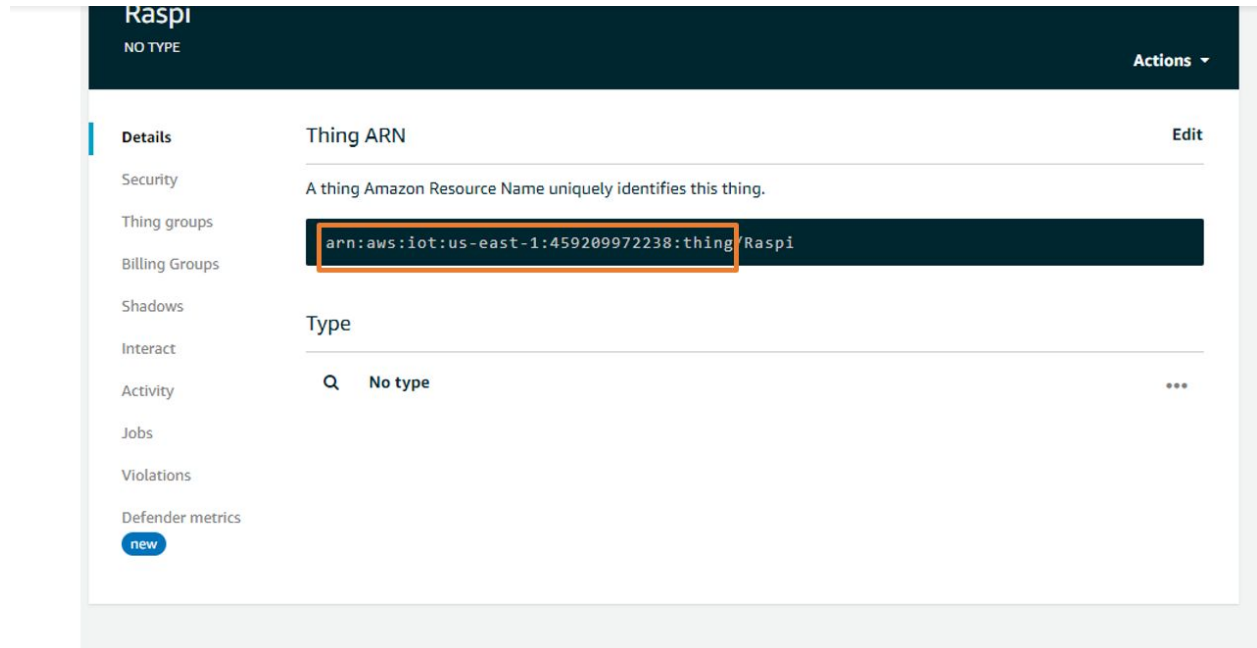


xvi) Enter a name for your identity and check the “enable to unauthenticated identities” box in the “Unauthenticated identities section”. Leave everything else as the default and click on create pool.

xvii) Once the identity is created, click on view details and then on “view policy document” under Role summary. Next click on edit. An image of this step is shown below:



xviii) In the policy document, scroll down to the resource, and then insert the ARN created for our “thing”. In order to obtain this ARN, return to the things page. Click on the thing you created, and you should see your ARN. Copy the ARN up until the forward slash and paste it in the resource field. Lastly, click allow. An image on how to obtain the ARN is shown below:

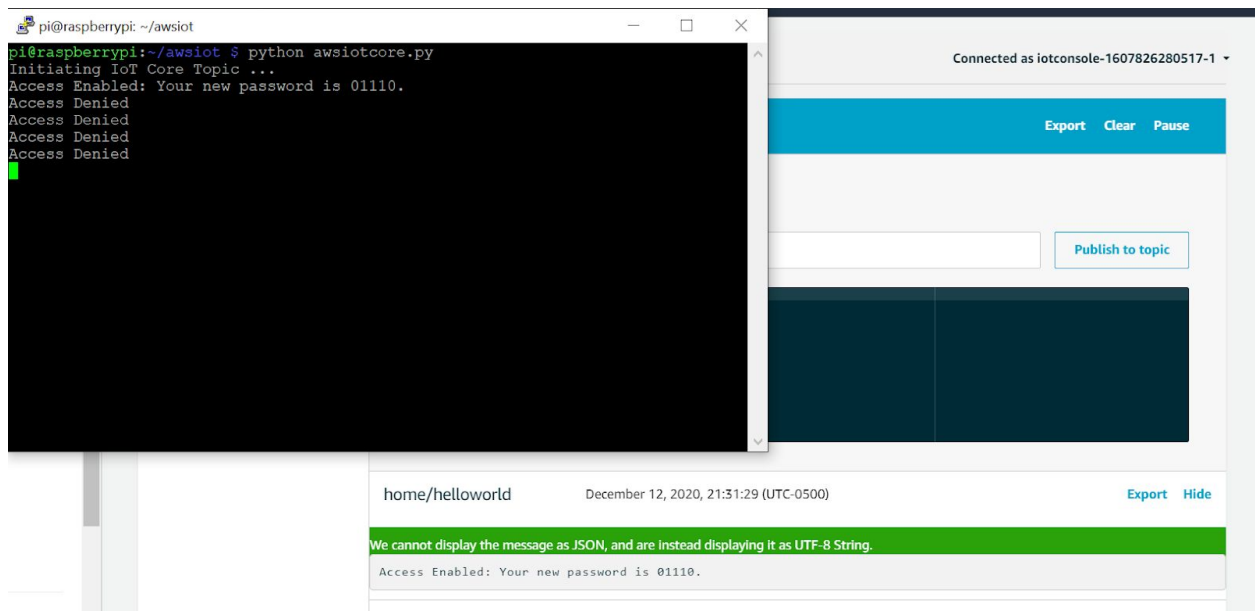


xix) Now, remote into your raspberry pi via terminal and create a python file with the following source code used for this project:

<https://drive.google.com/drive/folders/17UQFTh9NZAJ5e5tRgeqL9FARocAusCgh?usp=sharing>

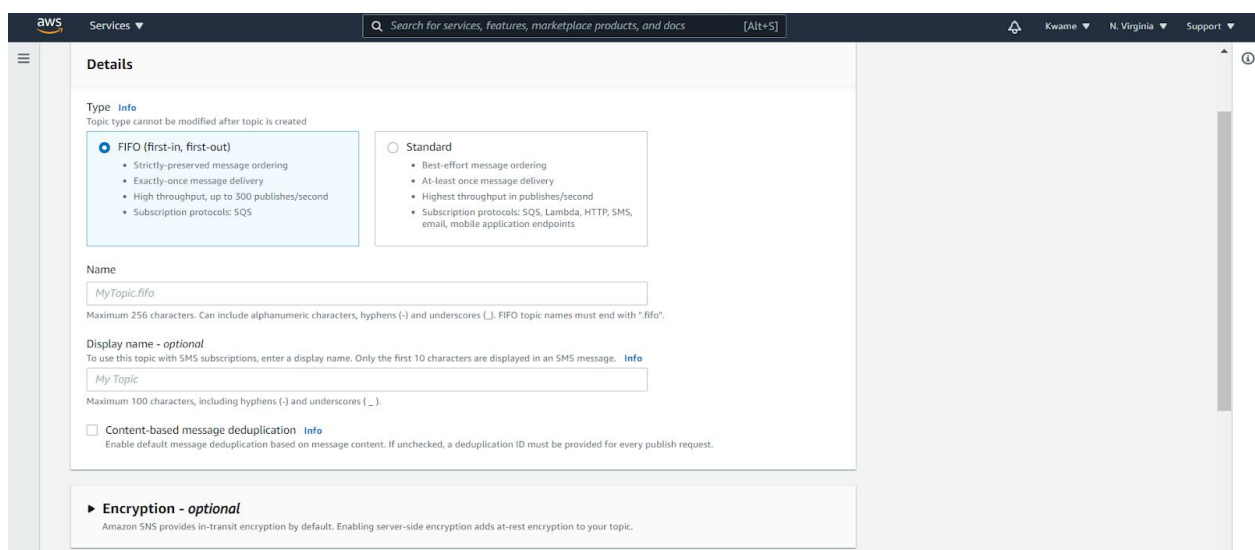
xx) Be sure to put in your information in the following fields. For the client ID, you can call it whatever you would like. For the endpoint, be sure to insert your endpoint. To obtain your endpoint, go to settings located on the bottom left of the AWS IoT Console and you should be able to copy and paste this endpoint into the python code. Lastly, for the configuring of credentials, be sure to use your private key, root ca permission file, and the certificate permission file. These documents should be in the directory you are currently working in.





3) Creating a rule in SNS: We will use the Amazon Simple Notification Service (SNS) to create a rule that sends a new password to the user's email each time access is enabled.

i) Return to the AWS console and search for Amazon SNS. Open SNS, and on the left hand side, click on Topic to create a topic. Select the standard option for type, and choose a name of your preference. Create the topic.



iv) Next, click on create subscription. Select email as the protocol and enter your preferred email address below. Note that this email address does not have to be the same one being used for AWS. Finally, create the subscription. Check in your email for a message from AWS and confirm the subscription.

The screenshot shows the 'Create Subscription' form in the AWS Management Console. The form is titled 'Details' and contains the following fields:

- Topic ARN:** A text input field containing 'arn:aws:sns:us-east-1:459209972238:newtopiccccc' with a clear button (X) on the right.
- Protocol:** A dropdown menu with 'Email' selected. Below the dropdown is the text 'The type of endpoint to subscribe'.
- Endpoint:** A text input field containing 'test@example.com'. Below the field is the text 'An email address that can receive notifications from Amazon SNS.'

Below the form fields, there is a light blue informational box with an information icon (i) and the text: 'After your subscription is created, you must confirm it. Info'.

At the bottom of the form, there are two optional sections:

- Subscription filter policy - optional:** This policy filters the messages that a subscriber receives. Info
- Redrive policy (dead-letter queue) - optional:** Send undeliverable messages to a dead-letter queue. Info

At the bottom right of the form, there are two buttons: 'Cancel' and 'Create subscription' (which is highlighted in orange).

i) Now we must create a rule for the payload from our home/helloworld topic to be sent to our e-mail. Return to the AWS console, click on Act located on the left panel. Then click on Rules. Choose a name for your rule, and choose the latest SQL version.

ii) In the rule query, insert `SELECT * FROM 'home/helloworld'`

Rule query statement
Edit

---

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









```
SELECT * FROM 'home/helloworld'
```

Using SQL version 2016-03-23

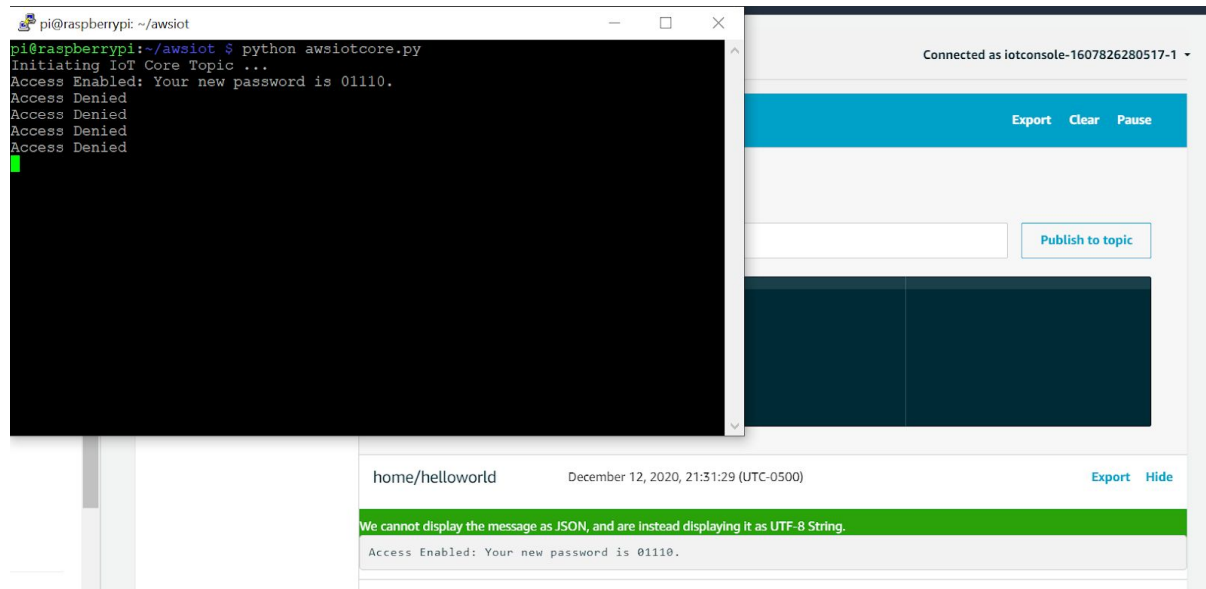
iii) Then, right below the rule query statement, we must create an action. Select the send a message as SNS push notification and click configure action. You should now see this screen.



Select an action.

<input type="radio"/>		Insert a message into a DynamoDB table DYNAMODB
<input type="radio"/>		Split message into multiple columns of a DynamoDB table (DynamoDBv2) DYNAMODBv2
<input type="radio"/>		Send a message to a Lambda function LAMBDA
<input checked="" type="radio"/>		Send a message as an SNS push notification SNS
<input type="radio"/>		Send a message to an SQS queue SQS
<input type="radio"/>		Send a message to an Amazon Kinesis Stream AMAZON KINESIS
<input type="radio"/>		Republish a message to an AWS IoT topic AWS IOT REPUBLISH
<input type="radio"/>		Store a message in an Amazon S3 bucket S3

iv) Choose the topic created in SNS. Set message format as raw, and lastly create a role. Finally, click create rule. Now, each time the AWS IoT core test environment receives an “Access Enabled” message, a copy of this message gets sent to our email. This is shown below:



**AWS Notifications** <no-reply@sns.amazonaws.com> 9:31 PM (1 hour ago)  
to me ▾  
Access Enabled: Your new password is 01110.  
\*\*\*

## Source Codes

This is our source code file. It is the same code, but to provide for better availability we made it a text file, and a python file. This is the only python code needed to be put on the Raspberry Pi to complete our final project:

<https://drive.google.com/drive/folders/17UQFTh9NZAJ5e5tRgeqL9FARocAusCgh?usp=sharing>

Here is our final presentation video with slides and a demo:

[https://kennesawedu-my.sharepoint.com/:v:/g/personal/nakuffo\\_students\\_kennesaw\\_edu/EdQACynCXyNHr6WzHEGW7D4BS3rjohtsKZ0j149V0Ap62Q](https://kennesawedu-my.sharepoint.com/:v:/g/personal/nakuffo_students_kennesaw_edu/EdQACynCXyNHr6WzHEGW7D4BS3rjohtsKZ0j149V0Ap62Q)

## **Lessons learned**

Our team learned some really valuable lessons during this project. Going into this assignment, none of our team members knew anything about AWS IoT aside from our warm-up lab. With this project we were able to acquire so many valuable tools to be able to use for the future, in school and out of school. We were able to experience how AWS IoT works with other programs and services such as SNS (Simple Notification Service). Something for us to think about is the fact that we barely scratched the surface of AWS. We used IoT core with SNS, but AWS provides for so many other services to use. AWS offers everything from IoT, to game development and machine learning tools.

It took our team quite a bit of time to get our Raspberry Pi connected to the AWS IoT. Luckily we were able to find some online resources along with the warm-up lab to complete this task. This was one of the biggest challenges faced, because with AWS IoT, one must make sure that the certificates on the board match with the certificate that AWS provides.

Another lesson that none of us had expected to run into is the fact that the Raspberry Pi only has GPIO pins, no actual analog reading pins. We were a bit surprised by this because our group has been working with the Beaglebone Black in another class which has specified pins for reading in an analog value.

This project was a fun and insightful way to end the class. Through the RezLok application, we were able to demonstrate our understanding on the fundamental principles of IoT, which has invigorated so many new technologies in different industries. We hope to build on this project and explore more ways we can make this application robust.

## Works Cited

- Rishab. "Connecting Raspberry Pi to AWS Iot Core : Setup and Code Using Python and AWS IOT." *YouTube*, YouTube, 3 Aug. 2020, [www.youtube.com/watch?v=kPLafcrng-c](https://www.youtube.com/watch?v=kPLafcrng-c).
- "Using Beam to Connect to AWS IoT Core." *Using Beam to Connect to AWS IoT Core* | *SORACOM Developers*, developers.soracom.io/en/start/aws/beam-iotcore/