Smart Campus:3D printer print-job monitoring using AWS IoT button, Raspberry Pi and Pi Cam

Ponmithiran Kannan
SUTD, Student,EPD
kannan_ponmithiran@mymail.sutd.edu.sg

Loy Shi Wei
SUTD, Student, EPD
shiwei_loy@mymail.sutd.edu.sg

Yap Zi Qi
SUTD, Student, EPD
ziqi_yap@mymail.sutd.edu.sg

ABSTRACT
In this paper, we discuss how to make a simple 3D printer monitoring device that sends a picture of your print job with a click of the AWS IoT button using Raspberry Pi and Pi Cam.

CCS Concepts

Keywords
Raspberry Pi, AWS, AWS Iot, Pi Camera

1. INTRODUCTION
Smart campus is a hot topic these days and we are looking at ways to make our lives easier and simpler so that we can focus on things that matters to us most while studying. Currently 3D printing is widely used in the campus for rapid prototyping but there is a shortfall of not being able to keep an eye on our print jobs, other than physically checking on it. Sometimes print jobs fail, and we may not know of this until we check on it. Therefore, by using an IoT print job monitor, this allows us to constantly monitor the 3D print job in progress and the print quality so 3D printer users need not check it manually.

2. Things Used for this Project

2.1 Hardware Used in this Project
Raspberry Pi Model B

Amazon Web Services AWS IoT button

Raspberry Pi Camera

Laptop

3D Printer

2.2 Software Used in this project:
Amazon Web Services (AWS) IoT

Amazon Web Services (AWS) Lambda

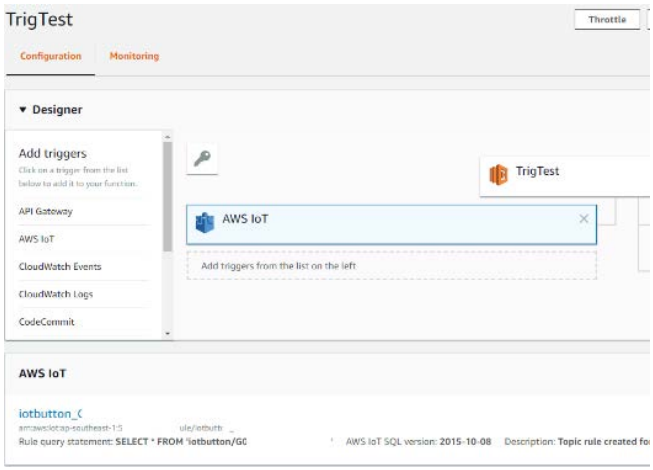
Amazon Web Services (AWS) SNS

3. Methodology

3.1 Configuring the AWS IoT Button
First, we connect the AWS IoT button to a WiFi network and link it to our AWS account. It must be connected to a WiFi network to install the required files and send messages to AWS IoT. More details on configuring the button can be found here: https://docs.aws.amazon.com/iot/latest/developerguide/configure-iot.html

3.2 Creating a Lambda rule with AWS Lambda
Next, we use AWS Lambda to create a new Lambda function. The method to create the Lambda function can be found here: https://docs.aws.amazon.com/iot/latest/developerguide/iot-lambda-rule.html?shortFooter=true

This allows us to configure the AWS IoT button to trigger the Lambda rule. By doing so, it allows us to process the incoming message and then call another AWS or third party service. Once done, it should look like the picture below:



3.3 Setting up the Raspberry Pi
We installed the OS on the new Raspberry Pi in order to use it – the steps to configure a new Raspberry Pi can be found on the Raspberry Pi site https://www.raspberrypi.org/help/videos/.

The Raspberry Pi should have internet connectivity, to connect to AWS IoT button and AWS. We also installed Putty to SSH into the Raspberry Pi so that we can use the Pi remotely from our computer. The software can be found on this website: https://www.putty.org/

Then run the following on the Raspberry Pi terminal to get the latest Kernel:

```
sudo apt-get install rpi-update

sudo rpi-update
```

To use the AWS IOT SDK, we run the following commands on the Raspberry Pi terminal:

```
sudo apt-get install cmake
```

1. Install OpenSSL 1.0.2 developer version:

Add the following text at the end of the file under this directory /etc/apt/sources.list.d/raspi.list:

```
deb http://ftp.debian.org/debian jessie-backports main
```

2. Update the package list:

```
sudo apt-get update
```

3. Install Open SSL:

```
sudo apt-get -t jessie-backports install libssl-dev
```

4. Install AWS IOT SDK:

```
sudo pip install AWSIoTPythonSDK
```

5. Install the GPIO python lib:

```
sudo pip install RPi.GPIO
```

3.4 Setup the AWS IOT Service

We will now setup the AWS IOT service with the following steps[1]:

1. Sign in into the AWS Management console, head over to AWS IOT service home page.

2. Expand the registry menu from the left pane and click on “Things”.

3. Click on “Register a thing or Create” depending on whether you already have a “Thing”.

4. Give the “Thing” a name, e.g. “printerbot”, then proceed to click on “Create”.

5. Click on the thing card on the “Things” page that you wind up on after creating the “motorRunner” thing.

6. Click on Interact from the left pane. Copy and save the Rest API endpoint, you will need this later.

7. Click on Security from the left pane and click on Create Certificate

8. Download **4 files** from this page before you close and save them for later use:
   1. A certificate for this thing
   2. A public key
   3. A private key
   4. Root CA certificate

   Click on Activate.

9. Click on Attach a policy and then click on Create a new policy.
In the Action field, type in iot:*, in the Resource Arn field type in *, check the “Allow” checkbox in the Effect section. Give the policy a name

3.5 Writing code for Raspberry Pi

1. Sign in to AWS Amazon > Services > AWS IoT > Settings > copy Endpoint
   This is your awshost

2. **Change** following things in the below program:
   a. awshost (from step 1)
   b. clientId (Thing_Name)
   c. thingName (Thing_Name)
   d. caPath (root-CA_certificate_Name)
   e. certPath (<Thing_Name>.cert.pem)
   f. keyPath (<Thing_Name>.private.key)

3. Paste aws_iot_pub.py & aws_iot_sub.py python scripts in folder where all unzipped aws files are kept.
4. Provide executable permission for both the python scripts.
5. Run aws_iot_sub.py script 6.

Code can be found at https://github.com/peltierchip/AWS-IoT-/blob/master/aws.py

Copy both .py file into your PI, we will run the program after we make the circuit which is just connecting the Pi Cam into to Raspberry at place it near your 3D printer.

Press the button to make sure that the Pi is working with no error code and login to your email to see if you at a picture of it.

Special thanks to Prof Tee Hui and AWS for the resources for this project.

4. REFERENCES

[1] Sumit Maingi, 2017. Make your first IOT device via AWS IOT Service and Raspberry Pi https://cloudncode.blog/2017/11/07/make-your-first-iot-device-via-aws-iot-service-and-raspberry-pi/