IoT Lab 11 – Raspberry Pi Camera Part 2

In this lab, we will explore using the Raspberry Pi Camera and AWS. We will use OpenCV to implement elementary machine vision with our Raspberry Pi and upload frames with motion in them to our AWS instance. If you are ambitious, you can stand up a webserver on AWS to display the images that were uploaded.

**Raspberry Pi Camera V2.1**

* Based on Sony IMX219 Sensor
* 8 Mpixel
* Video
  + 1080p@30fps
  + 720p@60fps
* Still Capture
  + 3280X2464

# Part 1 – Creating our AWS Instance and Logging in from the Raspberry Pi

I have created a classroom at AWS for the class. You should have received an invite to join the classroom at your NKU email. You should receive a $100 credit for the use of the services on AWS.

When you log into AWS Academy (<https://awsacademy.instructure.com>) it should take you to a Canvas site.

Select Modules. You should see the following.

Graphical user interface, application

Description automatically generated

Select the Learner Lab – Foundational Services

It should take you to this sceeen.

Graphical user interface, text, application, email

Description automatically generated

Click on Start Lab – Keep in mind there is a 4 hour limit. You can reset the time by clicking start lab again.

It takes a while for the lab to start. Wait until the AWS light turns green.



Click on the AWS link next to the green indicator. It will open a new window to the AWS Management Console.

A screenshot of a social media post

Description automatically generated

You can select Launch a virtual machine under Build a solution. It will take you to a launch wizard. You can scroll through and see all the prebuilt VM images you can choose.

Select the Ubuntu Server 18.04 LTS (HVM), SSD Volume Type for the VM we want to create. One advantage to this is that it is Free Tier eligible, so if you create your own AWS account, you could stand up a free version of this VM.

We want the Intel version of Ubuntu, so select 64-bit (x86).

A screenshot of a social media post

Description automatically generated

Click select, which will take us to the Instance Type selection. Let’s select the General purpose t2.micro instance, which is Free Tier eligible.

A screenshot of a social media post

Description automatically generated

At the bottom click Next: Configure Instance Details

On the next page, we can see all the different options we have for the Instance. We don’t need to make any changes here for our purposes.

A screenshot of a computer

Description automatically generated

Click on Next: Add Storage

On the storage page, increase the size for 8GiB to 24 GiB.

A screenshot of a social media post

Description automatically generated

Click on Next: Add Tags

We won’t make any changes on this page. Tags are used for companies to keep track of costs, assets, and monitoring purposes.

Click Next: Configure Security Group

The security group is, in essence, the firewall rules associated with our Instance.

A screenshot of a social media post

Description automatically generated

So we notice that ssh is allowed from any IP address. As a side note, this is the area that gets companies in trouble with cloud services is that they leave instances open to the Internet inadvertently.

Click on Review and Launch

Look over the items presented.

Then click on Launch

It will pop up a window similar to below about Key pairs, and this is for SSH.

A screenshot of a social media post

Description automatically generated

Select Create a new key pair from the drop-down

Give it a name, as shown below.

A screenshot of a social media post

Description automatically generated

Click on Download Key Pair and save the file to your Downloads directory. If you named it iot\_class, it would be named iot\_class.pem. This file is a private key to allow access to the AWS Instance.

Click on Launch Instances

You will see messages about the Instance (s) launching.

A screenshot of a social media post

Description automatically generated

Click on the button View Instances

It will take you to the EC2 Dashboard, where you can see the status of the EC2 Instance.

A screenshot of a cell phone

Description automatically generated

Get the public IPv4 address for the Instance. We will need that to connect to the Instance. **This IP address will be different each time you stop and start the Instance!** AWS does provide the ability to get a fixed IP address for a fee.

Now that our Instance is running, let us see if we can connect to it from our Raspberry Pi.

Connect to our Raspberry Pi.

Copy the key file to the Raspberry Pi. I use Bitvise since it provides both SSH and SFTP in the same session.

You need to remove read access for Group and Other to the iot\_class.pem file. Do the following

chmod 400 iot\_class.pem

At the bash prompt on the Raspberry Pi type the following command.

ssh -i filename.pem ubuntu@ip\_address

For example, to connect to this EC2 Instance ssh -i iot\_class.pem ubuntu@54.208.57.70

Accept the ECDSA fingerprint

It should log you into the EC2 Instance.

Once we verify that we can connect, log out of the EC2 Instance.

# Part 2 – Setting up Passwordless Access to our EC2 Instance

We want to be able to log in with SSH and transfer files to the EC2 Instance without having to use the private key file.

First, check to see if you already have existing SSH keys.

1. Log in to the Raspberry Pi
2. At the bash prompt type the following

**ls ~/.ssh**

1. If you see id\_rsa.pub and id\_rsa, you already have ssh keys and should skip the section Creating SSH Keys and proceed to the Getting the Public ssh key to the EC2 Instance authorized\_keys File section.

## Creating SSH Keys

If you don’t have existing SSH keys, we will create them.

1. At the bash prompt enter the following

**ssh-keygen**

1. Check that the keys were created by typing the following at the bash prompt.

**ls ~/.ssh**

1. Both the id\_rsa.pub and id\_rsa files should be there.

## Getting the public ssh key to the EC2 Instance authorized\_keys File

The id\_rsa and id\_rsa.pub files are a private/public key pair based on the RSA algorithm. The file id\_rsa is the private key. The private key should never be shared if it ever becomes compromised stop using it. The id\_rsa.pub is the public key for ssh and can be freely shared.

To allow for passwordless access to the EC2 Instance, we need to add our public key to the authorized\_keys file on the EC2 Instance.

Unfortunately, the utility ssh-copy-id, which will copy the public key to the remote machine, won’t work in this case because we have a private key to log in. However, we can use some pipes and redirection to accomplish the same thing.

At the bash prompt type on one line.

**cat ~/.ssh/id\_rsa.pub | ssh -i iot\_class.pem ubuntu@ip\_addr 'cat >> ~/.ssh/authorized\_keys'**

This command takes the public key in id\_rsa.pub and pipes it into ssh. The command ssh logs into the EC2 Instance and types out what is in the pipe and appends it to the authorized\_keys file on the EC2 Instance.

Test that it worked. You should be able to type the following now and be automatically logged in.

**ssh ubuntu@ip\_addr**

# Part 3 – Using OpenCV to Perform Motion Detection

OpenCV is a machine vision library for Python. We will use it to perform background subtraction from a base frame. If the threshold of change is beyond a certain level, there is motion in the frame.

We need to some installs on our Raspberry PI’s to allow OpenCV to run. At the bash prompt on your Raspberry Pi, do the following.

pip3 install opencv-contrib-python==4.1.0.25

pip3 install imutils

sudo apt-get install -y libhdf5-dev

sudo apt-get install -y libatlas-base-dev

sudo apt-get install -y libjasper-dev

sudo apt-get install -y libqtgui4

sudo apt-get install -y libqt4-test

Note: Above the line pip3 install opencv-contrib-python==4.1.0.25

has two equal signs (==).

Download the code **motion\_lab.py** from Canvas and install it I the /home/pi/scripts directory on your Raspberry Pi. Review the code and the comments. The code is also reviewed in the lecture material.

When motion\_lab.py is running and detect motion, it will create a jpeg file in the scripts directory. It will have the name of the form imgYYYYMMDDHHMMSS.jpg where YYYYMMDDHHMMSS is Year, month, day, hours, minutes, and seconds.

Below is an example of what the picture will look like when motion is detected.

A picture containing indoor, toy, young, standing

Description automatically generated

The bounding boxes indicate where “motion” is detected.

**Test the program and save one of the images when motion is detected to be submitted with your lab writeup.**

## Uploading Our Images to AWS

Now that we can capture images, we want to move them to the AWS Instance. Modify the code to use the Python os module to perform a secure copy (scp) to the AWS Instance that was set up in part 1 and part 2 of this lab. We also want to remove the file from the Raspberry Pi once it has been copied to the AWS Instance.

On the AWS Instance create a directory named **motion** in ubuntu home using the following command at the bash prompt on the AWS Instance.

**mkdir ~/motion**

Look for these comments in motion\_lab.py, and you will put your code there.

**# Add your code here to use scp to copy the imagefile just created to the AWS server**

**# You will need to use the os module to execute the scp command as if it was typed**

**# in at the command line.**

A few hints to help.

1. The filename of the file you want to copy to the AWS server is stored in the variable **filename**.
2. The form of scp is scp filename ubuntu@ip\_addr:~/motion/filename.
3. Build the command that you want to execute as a variable containing a string and then use os.system(cmd) to execute it. Where cmd is the string containing the command to execute.
4. The cmd to delete a file on Linux is **rm filename**.
5. Remember, every time you restart the AWS Instance, you will have a different public IP address.

Once the program is working, log onto the AWS Instance and list the contents of the motion directory. **Take a screenshot of the list of directory contents to submit with your lab writeup.**

# Going Further

Some ideas for going further. Now that you can move files to an AWS Instance, it would be possible also to serve up the images.

# Lab Submission

For the lab submission, please write a lab report and answer the following questions. Make sure you use complete sentences and proper grammar in your writeup. Make sure you indicate what question you are answering.

1. Submit the modified Python program on Canvas directly that detects motion and uploads the images to the AWS Instance. Make sure your programs include an updated header with your name and what the code does. Also, add comments in the code that was added.
2. Submit a brief lab write up on Canvas that answers the following questions.
   1. Image of detected motion
   2. Screenshot of the list of images that were uploaded to your AWS Instance in the motion directory.
   3. Reflection
      1. Write a brief reflection. The reflection should be a paragraph or two in which you reflect on your lab experience. Discuss items such as the following:
      2. What was the most valuable feature of the lab?
      3. How did you prepare for this lab? What changes are you considering in preparing for your next lab?
      4. What did you learn from this experience?
      5. What advice would you give someone preparing for this lab for the first time?
      6. Don't think of the reflection as a question/answer section. Instead, this should be a well-written paragraph or two that discusses items like those listed above.

Submit your lab writeup and Python programs on Canvas.