Building A Collection of Labs for Teaching IoT Courses

Xing Liu
Dept. of Computer Science and Information Technology
Kwantlen Polytechnic University
Surrey, Canada
xing.liu@kpu.ca

***Abstract*—This paper introduces a collection of labs that can be used for teaching Internet of Things (IoT) courses. An IoT system consists of physical devices, the Internet, and the cloud. The labs are designed to give students opportunity to experiment with these three aspects of IoT. On the physical devices side, the labs use a Raspberry Pi single board computer, selected sensors and actuators. On the software side, Python libraries are used for coding device interfaces and IoT applications. Amazon Web Services (AWS) IoT is the cloud platform used by the labs. A laptop computer with a Virtual Network Computing (VNC) client installed serves as the development platform which connects to the Raspberry Pi computer via a local WiFi network. The Raspberry Pi computer interacts with sensors and actuators and communicates with the AWS IoT cloud service through the Internet. The paper provides details on how the labs are developed. Test results are presented to illustrate how the labs work.**

Keywords-Internet of Things; IoT; teaching; courses; labs.

I. Introduction

The Internet of Things (IoT) has gone through rapid development in past few years. Numerous commercial products have been developed. The technology is being applied to many aspects of our life. In order to provide the much needed workforce for both development and applications of IoT technology, universities and technical institutions have started teaching IoT courses in their computer science, computer engineering, or information technology curriculum [1]-[4].

In order to help students understand the technical concepts of IoT, hands-on training is essential. Ideally, IoT courses should be taught along with a series of labs.

Although IoT course samples are not difficult to find on the Internet, detailed hands-on labs used in the courses are rarely available. Therefore, it is the author's intention to share such information in this paper.

The paper summarizes the author's experience in developing labs for an IoT course. Detailed descriptions are provided regarding setting up the lab platform, the use of sensors and actuators and the AWS IoT cloud platform, together with computer code snippets for the labs.

The paper is organized as follows. Section II gives an introduction to the Internet of Things. Section III describes the architecture of the system for the labs. Section IV explains the details of the selected labs. Test results are provided in Section V.

II. Internet of Things

According IEEE [5], IoT is "a network that connects uniquely identifiable things to the Internet. The things have sensing/actuation and potential programmability capabilities. Through the exploitation of unique identification and sensing, information about the thing can be collected and the state of the thing can be changed from anywhere, anytime, by anything".

Essentially, an IoT system has three components: physical devices, which are also called the *things*, the Internet, and the cloud. The simplified IoT model can be represented using Figure 1.

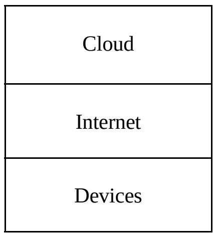


Figure 1. Simplified model of an IoT system

In order to provide students opportunities to understand various aspects of IoT, properly designed labs of an IoT course should cover these three components.

The learning objectives include having students understand IoT operations from a system perspective and gain hands-on skills in constructing IoT systems and developing related software. After completing the labs, students should understand what sensors and actuators are and what they can do. Students should also be able to design and build basic electronic interface circuits for sensors and actuators. Students will understand the concepts of data sampling, data collection and data transfer. Students will also understand how physical devices should identify themselves to cloud services and communicate with the services securely. Students are expected to be able to set up AWS IoT services to collect data from sensors and store the data in the cloud. The labs should enable students to use other cloud services to process data as well.

Preparing students for employment in the region is the rationale of selecting AWS and AWS IoT as the cloud platform for the labs.

III. SYSTEM DESCRIPTION

All labs are designed based on the system architecture as shown in Figure 2. The system has a Raspberry Pi single board computer, sensors, actuators, Internet connection, and the cloud. The Raspberry Pi computer, sensors, and actuators make a *thing* which is a physical device with computing power. The thing can send its data to the cloud via the Internet. A service in the cloud can interact with the thing via the Internet as well. The laptop computer connected to the Raspberry Pi serves as the development platform.

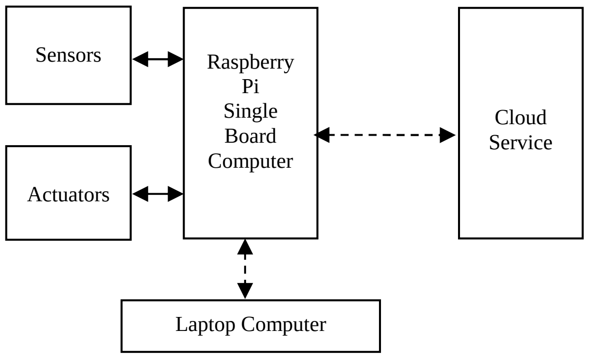


Figure 2. System architecture for running the labs

A. The Raspberry Pi Single Board Computer

The single board computer for the labs is a Raspberry Pi 3 model B with a 1.2GHz 64-bit quad-core ARMv8 CPU and 1 GB of RAM. It has both wired and wireless network interfaces. The wireless network interfaces include 802.11n Wireless LAN and Bluetooth. The Raspberry Pi has a powerful Input/Output interface called GPIO which stands for “general purpose input output”, as well as a camera interface. The “hard disk” of the Raspberry Pi is a SD card which can have tens of gigabytes of memory. Although the Raspberry Pi can have different types of operating systems, its “official” operating system is called Raspbian, which is of a Linux type. The Raspberry Pi can be programmed using popular programming languages such as C, JavaScript, Java, and Python. The collection of labs in this paper are based on Python.

B. Sensors

Sensors make the main component of an IoT system. They are used to measure environment parameters such as temperature, humidity, motion, light intensity etc, just to name a few. The labs of this paper use sensors that are readily available on the market and are of low cost, such as the DHT11 temperature/humidity sensor. Students are encouraged to obtain sensor kits designed for IoT applications as well. These kits not only have a variety of sensors, but also electronic components for building circuits, such as breadboards, wires, and resistors.

C. Actuators

Actuators in IoT systems generate movements, rotations, or other actions. The actuator used by the labs of this paper is a DC servo motor. The servo motor generates movements so that students understand what “actuation” means. The labs of this paper use a micro servo motor called SG90.

D. Camera

The camera used in the selected labs is a Raspberry Pi Camera with a Sony Exmor IMX219 Sensor and a resolution of 8 mega pixels. It has a fixed focus lens. The camera connects to the Raspberry Pi via an interface called Mobile Industry Processor Interface Camera Serial Interface Type 2 (MIPI CSI-2). The camera can be used to take still pictures or record video.

E. The Cloud

The labs use AWS IoT as its cloud service. AWS IoT is a popular Amazon web service for IoT applications. Physical devices connected with AWS IoT can interact with each other and with other AWS cloud services. Sensor data can be stored in the AWS cloud and can be analyzed there.

The main components of AWS IoT are: 1). A device gateway which enables physical devices to securely communicate with AWS IoT; 2). A message broker that helps devices and AWS IoT applications to publish and receive messages using the Message Queuing Telemetry Transport (MQTT) protocol; 3). A Rules Engine that provides message processing and integration capabilities, as well as to republish messages to other subscribers; 4). A Registry that allows users to register devices and their attributes; 5). A Device Shadow that is used to store and retrieve device state information; 6). A Device Provisioning Service that maintains device entries in the registry, certificates which devices use to authenticate with AWS IoT, and policies which determine what operations a device can perform in AWS IoT. The reason for choosing AWS IoT is to have students learn a popular industrial cloud service which is particularly useful for industry in the region.

IV. DETAILS OF SELECTED LABS AND TEST RESULTS

The labs are designed in such a way that students will learn the practical skills of the thing side first. These are the labs for driving sensors and actuators locally through the Raspberry Pi without an Internet connection. The students learn the basics of sensors and actuators, I/O interfaces, and how to interact with them through the Raspberry Pi. After the students have become familiar with the local thing side, additional labs will give them the opportunity to connect the thing to the cloud, experiment with sending data to the cloud and receiving instructions from the cloud. Further labs will enable students to learn data processing using other AWS cloud services such as machine learning.

A. Set Up the Raspberry Pi

Assembling a newly purchased Raspberry Pi is straightforward. One caution is that heat sinks should be installed to avoid overheat so students should not omit this step. Usually a new Raspberry Pi comes with the operating system Raspbian pre-installed. If not, Raspbian can be downloaded from the official Raspberry Pi website [6] and can be installed using software tools that work with SD cards.

It is recommended to install the Raspbian OS with a desktop-type GUI so that the user can connect to AWS IoT later via the web browser of Raspbian.

However, Raspberry Pi is “headless”, meaning there is no monitor or even a LCD screen connected initially. The most convenient way to use a Raspberry Pi is for it to share the monitor, keyboard and mouse with a laptop computer. However, separate keyboard and mouse are still needed to set up the Raspberry Pi the first time in order to enter initialization information, such as the SSID and password of a wireless network and to enable VNC on the Raspberry Pi.

Installing the software applications PuTTY, Angry IP Scanner, RealVNC, SDFormatter on the laptop computer before setting up the Raspberry Pi is very useful. The laptop computer only has to be on the same WiFi network as the Raspberry Pi in order to use RealVNC to connect to the Raspberry Pi.

In order to run Python programs on a Raspberry Pi, the Python engine has to be available. The way to verify that Python is installed properly on the Raspberry Pi is by typing the command “python” on a Raspbian command terminal and run a line of Python code such as `print "Hello, World!"`.

It is necessary to run all of the following commands to install or upgrade Python drivers on the Raspberry Pi before developing and running programs written in Python:

```
sudo apt-get update
sudo apt-get install python-dev python-pip
sudo pip install --upgrade distribute
sudo pip install ipython
sudo pip install --upgrade RPi.GPIO
```

Among the list of commands above, `python-pip` is a package management system used to install and manage software packages written in Python. `ipython` (Interactive Python) is a command shell. `RPi.GPIO` is the GPIO pin driver API, which is usually already included in Raspbian.

B. *Drive A LED*

Being able to light up a LED is essential for the labs because LEDs can be used conveniently as indicators of various activities in the IoT system. A LED can help detect if a circuit or a pin of an output port on the Raspberry Pi is working or if a sensor is triggered.

The circuit diagram for the LED lab is shown in Figure 3 where GPIO Pin 17 is used to drive the LED.

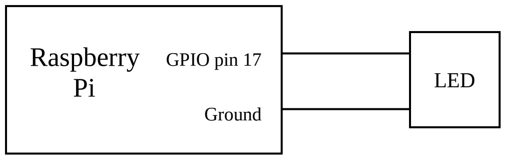


Figure 3. Circuit for driving a LED

Key Python code snippets for the LED lab is shown in Figure 4.

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
led_pin=17
GPIO.setup(led_pin, GPIO.OUT)
GPIO.output(led_pin, GPIO.HIGH)
GPIO.output(led_pin, GPIO.LOW)
```

Figure 4. Python code for the LED lab

In Figure 4, Line 1 imports the Python GPIO driver. Line 2 specifies the GPIO pin mode because there are two options for numbering the GPIO pins. The `GPIO.BCM` option above means that the pins are referenced by the "Broadcom SOC Channel" numbering system. In this option, the pin numbers have the prefix "GPIO". The other option is `GPIO.BOARD` which specifies the pin numbers based on the numbers printed on the circuit board such as “01”, “02”, and “40”. Line 4 sets up the `led_pin` for output. Line 5 turns the LED on and Line 6 turns the LED off. Students can write their own code to have the LED on for certain amount of time and at different intervals to create interesting display patterns.

C. *Read A Pushbutton*

Reading a pushbutton as shown in Figure 5 can be useful when a user wants to interact with the IoT system via a “real button” instead of a virtual button displayed on a graphical user interface. The circuit program is shown in the Figure 5.

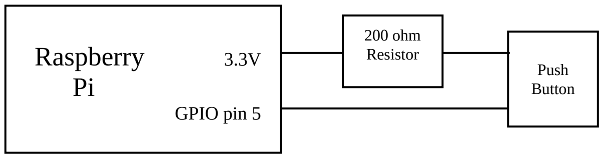


Figure 5. Circuit for reading a pushbutton

Key Python code snippets for the pushbutton lab is shown in Figure 6.

```
import RPi.GPIO as GPIO

push_pin=5
GPIO.setmode(GPIO.BCM)
GPIO.setup(push_pin, GPIO.IN,
           pull_up_down=GPIO.PUD_DOWN)

GPIO.add_event_detect(push_pin,
           GPIO.RISING, bouncetime=200)

def on_push_down(channel):
    print("Pushbutton is pressed.")

GPIO.add_event_callback(push_pin,
                 callback=on_push_down)
```

Figure 6. Python code for the pushbutton lab

In Figure 6, similar to the LED lab, the Python GPIO driver is imported and the pin mode is set to `GPIO.BCM`. In

Line 6, `pull_up_down=GPIO.PUD_DOWN` means that a value of `GPIO.HIGH` will be read by Python code when the button is pressed (because the circuit between the +3.3V pin and the GPIO pin 5 is closed). Line 8 means that the `push_pin` is set for rising edge detection and it will ignore further edges for 200ms to compensate for switch bounces. The last line registers the Python function `on_push_down` as a callback function, which means function `on_push_button` will execute after the pushbutton is pressed. A pushbutton press can be used to simulate any action that will trigger an event.

D. Read A Temperature Sensor and A Humidity Sensor

Measuring environmental parameters such as temperature and humidity is common for IoT applications. There are many types of temperature and humidity sensors available on the market. A popular sensor named DHT11 can measure both temperature and humidity. The circuit for using the DHT11 sensor is shown in Figure 7.

The DHT11 temperature/humidity sensor has three wires. The wire in the middle of the connector is the data wire. The other two wires are for power and the ground.

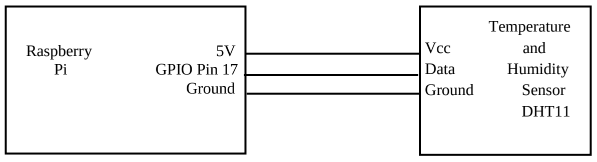


Figure 7. Circuit for the temperature/humidity measurement lab

Writing the Python code from the ground up to talk to the DHT11 sensor is complex because the code has to handle digital pulses and their encodings. However, a Python driver that makes application development much easier has been written and is available for download [7].

Two files should be downloaded from the above site: `dht11.py` and `dht11_example.py`. With the help of the `dht11.py` driver module, the key Python code for reading data from the DHT11 sensor can be made very simple and is shown in Figure 8.

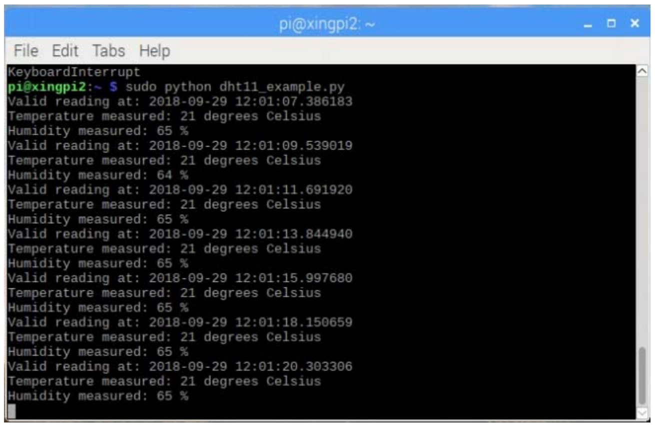
```
import RPi.GPIO as GPIO
import dht11

GPIO.setmode(GPIO.BCM)

instance = dht11.DHT11(pin=26)
result = instance.read()
if result.is_valid():
  print("Temperature: %d" % result.temperature)
  print("Humidity: %d %%" % result.humidity)
```

Figure 8. Python code for the temperature and humidity sensor lab

Some tests were performed in a residential home. Temperature and humidity data from the DHT11 sensor were recorded by taking a screenshot of a Raspbian terminal window in which the Python program prints its output. Test run results are shown in Figure 9.



```
KeyboardInterrupt
pi@xingpi2:~ $ sudo python dht11_example.py
Valid reading at: 2018-09-29 12:01:07.386183
Temperature measured: 21 degrees Celsius
Humidity measured: 65 %
Valid reading at: 2018-09-29 12:01:09.539019
Temperature measured: 21 degrees Celsius
Humidity measured: 64 %
Valid reading at: 2018-09-29 12:01:11.691920
Temperature measured: 21 degrees Celsius
Humidity measured: 65 %
Valid reading at: 2018-09-29 12:01:13.844940
Temperature measured: 21 degrees Celsius
Humidity measured: 65 %
Valid reading at: 2018-09-29 12:01:15.997680
Temperature measured: 21 degrees Celsius
Humidity measured: 65 %
Valid reading at: 2018-09-29 12:01:18.150659
Temperature measured: 21 degrees Celsius
Humidity measured: 65 %
Valid reading at: 2018-09-29 12:01:20.303306
Temperature measured: 21 degrees Celsius
Humidity measured: 65 %
```

Figure 9. Test results for the DHT11 temperature and humidity sensor

E. Control A Servo Motor

The labs of this paper use the micro servo motor SG90. Although Raspberry Pi has pin GPIO18 designated for producing pulse width modulation (PWM) pulses, other pins can be used to drive the servo motor SG90 as well.

It takes 20ms of the pulse width for the SG90 to travel through its full rotational range. By design, when the pulse width of the PWM signal is 1 millisecond (ms), the position of the servo motor is at its very LEFT side. The duty cycle of this position is (1ms/20ms) x 100 = 5%. For pulse widths of 1.5ms and 2ms, the servo motor is at its MIDDLE position with a duty cycle of 7.5% and at its far RIGHT position with a duty cycle of 10%.

The circuit for the servo motor lab is shown in Figure 10.

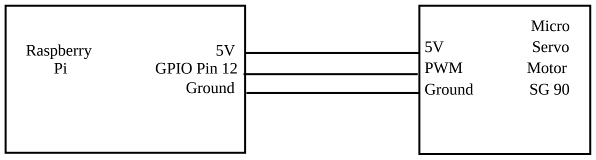


Figure 10. Servo motor control circuit

Key Python code for driving the servo motor is shown in Figure 11.

```
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)

pwm = GPIO.PWM(18,50)

pwm.start(5) #Start at 0 degrees
time.sleep(1)

pwm.ChangeDutyCycle(7.5) #turn to 90 degrees
time.sleep(1)

pwm.ChangeDutyCycle(10) # turn to 180 degrees
```

Figure 11. Key Python code for servo motor control

In Figure 11, `GPIO.PWM(18,50)`sets GPIO Pin 18 to output square wave pulses with 50 Hz of frequency. The two key Python functions are `start` and `ChangeDutyCycle`, which are used to position and rotate the shaft of the servo motor.

F. Control A Camera

The camera named Raspberry Pi Camera is the "official" camera to be used with Raspberry Pi. With this camera users can take still pictures, apply image effects and record video. Caution has to be exercised when installing the camera on the Raspberry Pi. Users have to make sure that the blue side of the ribbon cable face the Ethernet port and the silver side face the HDMI port. The camera has to be enabled under the configuration of Raspberry Pi as well. Another important set-up step is to enable VNC streaming of live images from the camera to the laptop computer. Otherwise no image displays can be seen on the screen of the laptop computer. This can be done by opening a terminal window on the Raspberry Pi, type command "`vncserver`", navigate to "Menu" then "Options" followed by "Troubleshooting", and select "Enable experimental direct capture mode". Figure 12 shows what a streamed image looks like on the screen of a laptop computer.

Figure 12. Pi camera captured image streamed to a laptop computer

In order to run Python code to control the camera, the Python module PiCamera has to be imported. Important functions are `start_preview`, `stop_preview`, and `capture` which is used to capture an image and `start_recording` which is used to record a video. Code samples are shown in Figure 13. Videos recorded can be viewed by running the command "`omxplayer`" followed by the name of the video file.

```
from picamera import PiCamera
camera = PiCamera()
camera.start_preview()
camera.capture('/home/pi/Desktop/image.jpg')
camera.stop_preview()
camera.start_preview()
camera.start_recording('/home/pi/video.h264')
sleep(10)
camera.stop_recording()
camera.stop_preview()
```

Figure 13. Sample Python code for using the Pi camera

G. Communicate with the Cloud

The cloud service used in the labs is Amazon's AWS IoT. In order to develop Python code running on the Raspberry Pi that can communicate with AWS IoT, a software development kit (SDK) has to be downloaded and installed on the Raspberry Pi. The SDK file can be downloaded from Amazon's website [8].

The SDK file should be unzipped and the user has to run the command "`sudo python setup.py install`" to install the SDK to a proper folder in the Raspberry Pi.

The SDK comes with several sample files which can be used as the starting point for development.

In order for the Raspberry Pi to communicate with AWS IoT, an identity of it named a "Thing" has to be created in AWS IoT. A set of security certificates have to be created and associated with the "Thing". These certificates have to be downloaded onto the Raspberry Pi which will use them to identify itself to AWS IoT when connecting. Policies have to be created to specify what the "Thing" is allowed to do in AWS IoT as well.

In the Python code running on the Raspberry Pi, an AWSIoTMQTTClient has to be created. This client will be used to publish messages to the AWS IoT cloud service and receive messages back from the AWS IoT cloud service via a "Topic" set up in the AWS Simple Notification Service (SNS).

Figure 14 shows the essential AWS IoT SDK functions needed for the Raspberry Pi to connect to AWS IoT and publish data to a topic in AWS SNS.

```
from AWSIoTPythonSDK.MQTTLib import
AWSIoTMQTTClient

myMQTTClient = AWSIoTMQTTClient("My Pi")

myMQTTClient.configureEndpoint("A3XXX.iot.e
u-west-1.amazonaws.com", 8883)

myMQTTClient.configureCredentials("/home/pi
/cert/RootCA.pem", "/home/pi/cert/xxxx-
private.pem.key", "/home/pi/cert/xxxxx-
certificate.pem.crt")

myMQTTClient.connect()
myMQTTClient.publish("my_topic",
"connected", 0)

payload = "sensor data"

myMQTTClient.publish("my_topic",payload,0)
```

Figure 14. Key Python code for communicating with AWS IoT

Sensor data can be used to build a "payload" which is published to AWS IoT. The AWS SNS service needs to subscribe to the topic which the MQTT client running on the Raspberry Pi publishes sensor data (payload) to. When data is being published, AWS SNS will receive and display them almost instantly. Users have the option to suspend the reception of data as well.

Figure 15 is a screenshot that shows a Thing in AWS IoT. This Thing is essentially a virtual representation of the

Raspberry Pi and its associated sensors in the AWS IoT cloud.

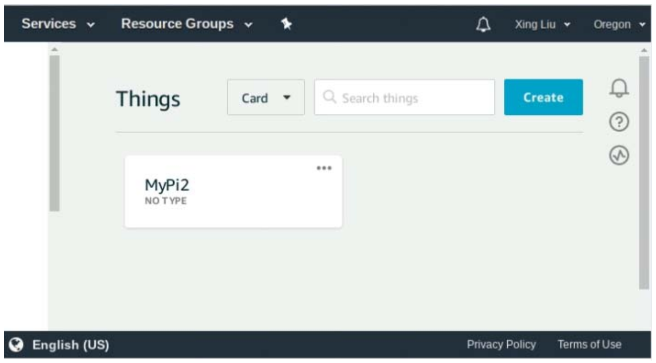
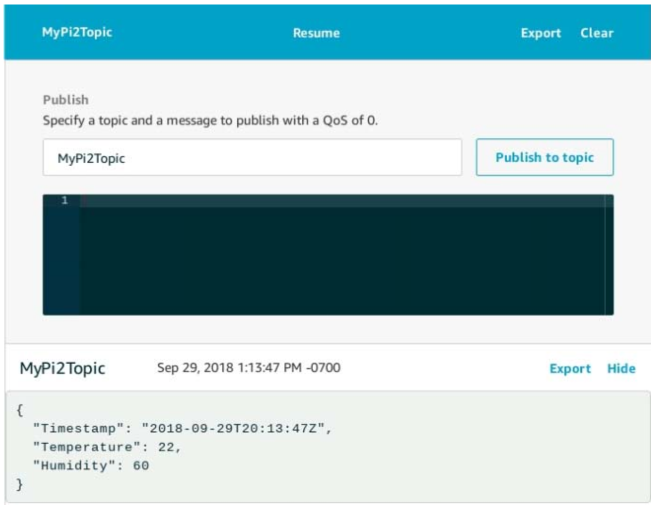


Figure 15. Screenshot that shows a "Thing" in AWS IoT

The AWS SNS service provides controls for publishing and subscribing to topics and a window for showing live topic data. Figure 16 is a screenshot that shows sensor data published by the DHT11 sensor via the Raspberry Pi and received by AWS SNS on a topic named "MyPi2Topic".


MyPi2Topic
Resume
Export
Clear
Publish
Specify a topic and a message to publish with a QoS of 0.
MyPi2Topic
Publish to topic
MyPi2Topic
Sep 29, 2018 1:13:47 PM -0700
Export
Hide

```
{
  "Timestamp": "2018-09-29T20:13:47Z",
  "Temperature": 22,
  "Humidity": 60
}
```

Figure 16. Screenshot showing sensor data received by AWS SNS

H. Labs under Development

Although the above set of labs have covered the three aspects of IoT, i.e., physical devices, the Internet and the cloud, some labs still need to be developed. For example, labs for generating email notifications and text messages; labs for controlling physical devices from the cloud; labs that allow one device to control another device via the cloud. Labs are also needed to teach students how to transfer large amount of data such as images and videos to the AWS cloud and store them there. These labs are still under development.

V. CONCLUSION

This paper has introduced a collection of labs that can be used in teaching IoT courses. The labs give students opportunities to learn different aspects of IoT ranging from physical devices to the cloud. Details of the labs and key Python code snippets are provided.

ACKNOWLEDGMENT

The author would like to express his gratitude for the support provided by the Provost's office of Kwantlen Polytechnic University.

REFERENCES

[1] X. Liu and O. Baiocchi, "An IoT Course for A Computer Science Graduate Program", International Conference on: Communication, Management and Information Technology (ICCMIT'16), Cosenza, Italy, April 26-29, 2016.

[2] Ryerson University, "Course Series in The Internet of Things (IoT)", https://ce-online.ryerson.ca/ce/calendar/default.aspx?id=5&section=program&mode=program&sub=atd&cert=CSTIOT00. Accessed on October 13, 2018.

[3] Northern Alberta Institute of Technology, "IOT/IOE Certificate", http://www.nait.ca/program_home_101497.htm. Accessed on October 13, 2018.

[4] Kwantlen Polytechnic University, "INFO 4381: Internet of Things and Applications", http://www.kpu.ca/calendar/2018-19/courses/info/index.html. Accessed on October 20, 2018.

[5] R. Minerva, A. Biru, and D. Rotondi, "Towards a definition of the Internet of Things (IoT)", 27 May 2015, IEEE Internet Initiative. https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf. Accessed on September 29, 2018.

[6] Raspberrypi.org, https://www.raspberrypi.org/downloads/. Accessed on October 13, 2018.

[7] DHT11 Python library, https://github.com/szazo/DHT11_Python. Accessed on October 13, 2018. Accessed on October 13, 2018.

[8] AWS SDK, https://s3.amazonaws.com/aws-iot-device-sdk-python/aws-iot-device-sdk-python-latest.zip. Accessed on October 13, 2018.