Python Services

A. Objectives

1. Learning about Python services
2. Sending email using Twilio Communication Cloud
3. Working with static web pages using Python Bottle
4. Using Cayenne for data control and monitoring
5. Quick introduction to JavaScripting

B. Time of Completion

This laboratory activity is designed for students with some knowledge of various Python services and it is estimated to take about 7-10 hours to complete.

C. Requirements

1. A Raspberry Pi 3 Model 3
2. 32 GByte MicroSD card → Give your MicroSD card to the lab instructor for a copy of Ubuntu.
3. USB adaptor to power up the Pi

D. Pre-Lab

Make Sure you have completed PART I.

E. Lab

The purpose of this lab is to learn about **Python services,** such as sending an SMS text, accessing the cloud, creating a webpage, and more. Before starting this section make sure you have completed Part I: Introduction to Python.

1. Sending SMS Message Using Twilio

Twilio is a developer platform for communications. Many companies including Lyft, Airbnb, or Netflix, use Twilio. In fact, Twilio powers communications for over 40,000 businesses around the world. Twilio has taken the global telecom network and turned it into a cloud communications platform with many features [1]. Some refer to Twilio as "AWS (Amazon Web Services) cloud for telecom"[2].

One of the powerful applications of Twilio is its ability to send text messages. In this section we learn how to use Twilio API to send text messages using Python in RPI. There are many very interesting projects that can utilize texting-to-send updates on things, such as motion captured by a camera, or temperature in a room, and much more. So, let's start:

1- Make sure you have an updated `pip` utility: `sudo apt-get install python-pip`

2- Then, install twilio using `pip` package manager utility: `sudo pip install twilio`

3- Using your laptop (remote machine) create an account with Twilio: https://www.twilio.com

4- Go to https://www.twilio.com/console/phone-numbers/incoming and receive buy a new phone number. I am not sure if you have to pay anything. If so, you can use my account. See course Canvas for account information.

5- Go to https://www.twilio.com/console/phone-numbers/verified and verify & add the number you want to send the text to.

6- Go to https://www.twilio.com/console and copy SID and AUTH TOKEN and save them.

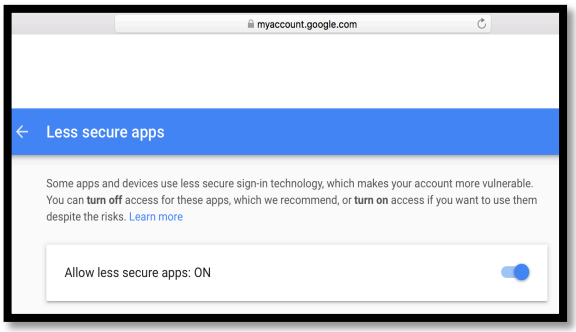
7- Log into your RPI using SSH. Create a directory called `Twilio`. In the directory create a file called `send_text.py`. Copy the code below into your file. Note that you need to make some changes to the file so it works. Pay attention to the indentations.

[1] Read here for more information: https://www.twilio.com/learn/twilio-101/what-is-twilio
[2] Read about AWS here: https://aws.amazon.com/what-is-aws/

```
#!/usr/bin/env python
from twilio.rest import Client
import numpy as np
# generate random integer values
from random import seed
from random import randint

account_sid = 'the id here'
auth_token = 'the token here'
client = Client(account_sid,auth_token)

#Generate a message:
x = generate a random number using random function
y = something here – need to fix it
fullMessage= text=(y+str(x)+' degrees')

message = client.messages \
    .create(
         body=fullMessage,
         from_='+1242323',
         to='+132423232'
     )
print(message.sid) # this would be somehting like: 85a11ef34d8ea1d9b4
```

Execute the `send_text.py` and make sure you receive a text message on the phone.

2. Sending an Email Using Google Account



Make sure you have a google Gmail account.

Note: You may want to create a new Gmail account to make sure your account is not locked due to sending too many emails!

Sign into your Gmail. Go to this link, below, and select Turn On:
https://www.google.com/settings/security/lesssecureapps.

Use the Python code below to send an email to your Gmail account from your RPI:

```
#!/usr/bin/env python
# Making Web request
# read about the module: https://docs.python.org/2/library/urllib.html
# THIS CODE WORKS ONLY FOR PYTHON 2
import urllib2
contents = urllib2.urlopen("http://python.org").read()
#print (contents)

#Send an email
import smtplib

GMAIL_USER = 'xxx@gmail.com'
GMAIL_PASS = 'yyy'
GMAIL_MSG = 'This is a test 2'
GMAIL_SUB = 'Hello World!'

SMTP_SERVER = 'smtp.gmail.com'
SMTP_PORT = 587

def send_email(recipient, subject, text):
     smtpserver = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
     smtpserver.ehlo()           # using Extended HELO
     smtpserver.starttls()       # will send your password with encryption
     smtpserver.ehlo
     smtpserver.login(GMAIL_USER, GMAIL_PASS)
     header = 'To:' + recipient + '\n' + 'From: '+ GMAIL_USER
     header = header + '\n' + 'Subject:' + subject + '\n'
     msg = header + '\n' + text + ' \n\n'
     smtpserver.sendmail(GMAIL_USER, recipient, msg) # function that send the email
     smtpserver.close()

send_email(GMAIL_USER , GMAIL_SUB, GMAIL_MSG)
```

When you run the code above, you should receive an email in your Gmail account.

PROJECT A: Write a Python code that when the temperature of the CPU in RPI exceeds 40 degrees Celsius you receive a text message or email. See Part I: Python lab to learn how you measure the CPU temperature. You need to make sure you can somehow demonstrate that this part works. If your code can do both text and email you receive **5 points extra credit**.

3. Creating a Web Server Using Bottle

When it comes to creating a web server, there are many different module choices such as web.py, Flask, Bottle, etc. Below are examples of using bottle and web.py modules. We will do a more elaborate example using bottle in later labs. Read about Bottle web framework here; https://bottlepy.org/docs/dev/tutorial.html

Use the following ways to install Bottle:

```
$ sudo apt-get install python-bottle
```

Attention: If you do this part on your RPI, you may want to use a monitor and keyboard to access your web page.

Create a directory called `bottle`. In the directory create a file called `index.py` and copy/paste the code below. Run the python code. Then open a web browser and paste the following in the URL: http://<IP_Address>:8080/hello/whatever. **Remember, you can use `localhost` as your IP address assuming the web page is opened on the local machine e.g., your RPI)**.

```python
#!/usr/bin/env python
from bottle import route, run, template

@route('/hello/<name>')
def index(name):
    return template('<b>Hello {{name}}</b>!', name=name)

run(host='10.0.0.218', port=8080)
```

Creating a web server using bottle – in your browser type in http://10.0.0.218:8080/hello/whatever

The most interesting aspect of bottle is that it uses Python standard library. It is possible to run *static* HTML pages using Bottle. Let's see how.

On your RPI create a directory called `bottle`. Then inside Bottle create another directory called `pubic`. In the bottle directory create a file called `app.py`. Also, create a file in `bottle/pubic/` directory and call it `home.html`. Copy and paste the code below in respective file.

```python
#!/usr/bin/env python
from bottle import route, run, static_file

@route('/<filepath:path>')
def server_static(filepath):
    return static_file(filepath, root='./public/')

run(host='10.0.0.218', port=8080, debug=True)
```

`bottle/app.py`

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Home page</title>
</head>
<body>
    <p>This is home page</p>

</body>
</html>
```

`bottle/public/home.html`

Go to bottle directory and run `app.py`. From a remote machine (your phone or laptop) open the browser and in the URL type in http://10.0.0.218:8080/home.html. Note that in the terminal on your RPI you should get something like this:

```
Bottle v0.12.13 server starting up (using WSGIRefServer())...
Listening on http://10.0.0.218:8080/
Hit Ctrl-C to quit.
```

In your case your IP address will be most likely different. You can change the `home.html` and completely redesign the web page. Also, note that every time a user accesses the webpage Bottle generates a record of the user and its IP address, along with the access time:

```
10.0.0.62 - - [11/Nov/2019 22:16:21] "GET /home.html HTTP/1.1" 200 322
10.0.0.62 - - [11/Nov/2019 22:16:21] "GET /favicon
```

You can find more information about HTML coding here:
https://www.w3schools.com/html/default.asp

PROJECT B: Change your default html web page which you created (`home.html`) and add a button to switch the displayed messages or something similar. The main purpose is to learn how to use *buttons* (read the Hint below).

Hint: See https://www.w3schools.com/howto/howto_js_toggle_text.asp
for more information. Run the sample code below and see what happens. Make sure you play with the code and understand how it works. Notice that in this example we have a *JavaScript* code, identified by <script>…</script> tag pair that is embedded into an HTML code, inside the <body> </body> tag pair.

JavaScript[3] is the programming language for the web that makes web page design dynamic. Thus, using JavaScript we can update and change both HTML and CSS. Furthermore, we can calculate, manipulate and validate data.

Remember, HTML is NOT a programming language. HTML, as a markup language doesn't really "do" anything in the sense that a programming language does. HTML contains no programming logic. It doesn't have common conditional statements such as If/Else.

In this example when we click on the button, the `myFunction` changes and a different text appears!

[3] Practice with JavaScript codes here: https://www.w3schools.com/whatis/whatis_js.asp

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body>

<p>Click the button to swap the text of the DIV element:</p>

<p><button onclick="myFunction()">Click Me</button></p>

<div id="myDIV">Hello</div>

<script>
function myFunction() {
  var x = document.getElementById("myDIV");
  if (x.innerHTML === "Hello") {
    x.innerHTML = "Swapped text!";
  } else {
    x.innerHTML = "Hello";
  }
}
</script>

</body>
</html>
```

4. Canyenne IoT Platform

The basic idea behind the Internet of Things (IoT) is that an end device, e.g. RPI, can send its data to be stored and visualized, and possibly even analyzed. There are many cloud platforms that can do these tasks. One such cloud platform is *Cayenne*.

Cayenne is one of the easiest and more powerful IoT platform for developing beautiful UIs for IoT solution. Cayenne supports devices like Raspberry Pi, Arduino and much more. In this exercise we briefly explain how Canyenne can be setup and connected to the RPI. Remember the idea is to be able to do the following tasks:

- Use default setup in Cayenne to visualize RPI's temperature and CPU;
- Remotely control RPI's GPIO ports;
- Send notification, say via email, if a condition occurs;
- Create your own Python code to automatically send sensor information to Cayenne to visualize the data.

So, let's start.

4.1. Use Default Setup in Cayenne IoT Platform

In this section we learn how to use default setup in Cayenne to visualize RPI's temperature and CPU.

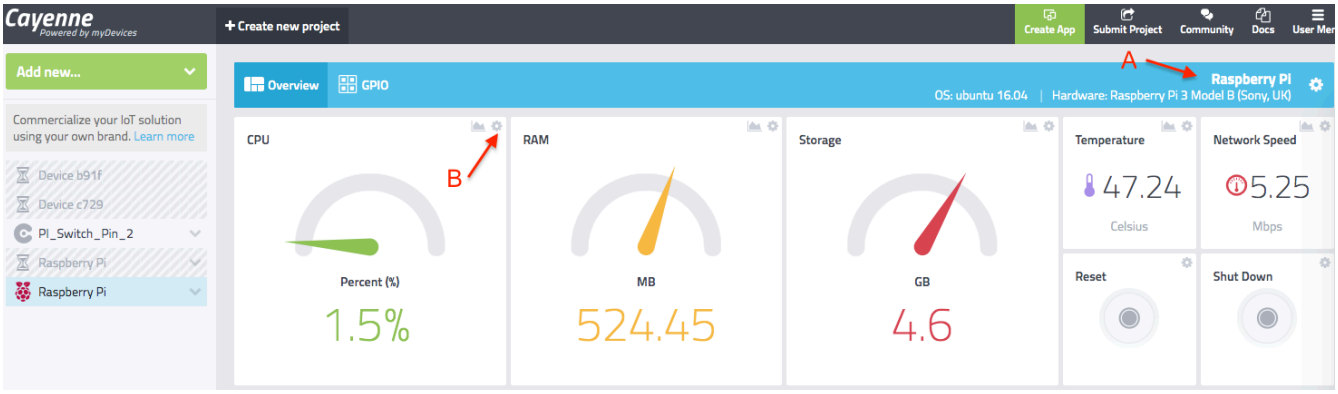
1- Using your laptop you need to go to https://cayenne.mydevices.com/ and create a free account. Keep your web browser on the laptop open. For the rest of this exercise, we call your laptop the *remote machine*.

2- Once you signed up, you will need to register/connect the RPI up to the account you just created. To do this just copy the two command lines shown after you sign up. The code looks something like this:

```
wget https://cayenne.mydevices.com/dl/rpi_bpn82i9.sh
sudo bash rpi_b8w82i9.sh -v
```

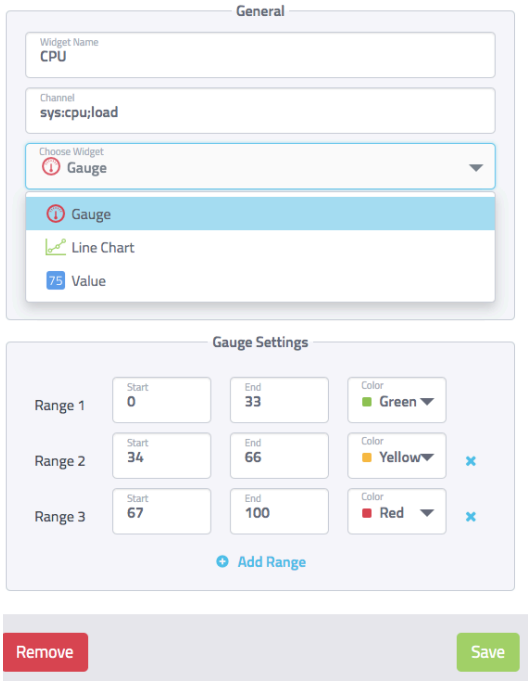
These two commands do the magic! Enter these into the RPI terminal. You can do this by doing SSH into your RPI. These files are unique for every new install. Note that it will take a few minutes to install the commands onto your RPI depending on how fast your internet connection is. The Cayenne web browser (e.g., your account) should automatically update with information on the installation process.

3- Once installed the dashboard on the remote machine (your Cayenne web browser) will display and should look like something below:



Note that in your screen you have several *widgets*. Each widget represents a parameter. You can move the widgets, resize them. Or change the way they look. Very cool, huh?

4- You can click on say CPU (the setting symbol in the top right corner of the CPU widget – (B) in the figure above) and change the configuration, say to Line Chart. Note that you can change the MIN and MAX In the chart.



In case Cayenne reports off-line, reset the RPI!

5- Using SSH log into your RPI and type `python` and run the following code:

```
>>> from math import *
>>> print (factorial(600000))
```

Note what happens to the CPU usage and CPU temperature of the RPI. Just for fun, you can SSH onto your RPI and run the above code three times. Observe what happens to the CPU and temperature. Try doing the same thing from several SSH windows and see how CPU reading changes.

6- Click where it says *Raspberry PI* on the right top corner of the screen (A). Then select Configure. Note how Cayenne has obtained all the information about your RPI, including your internal IP address.

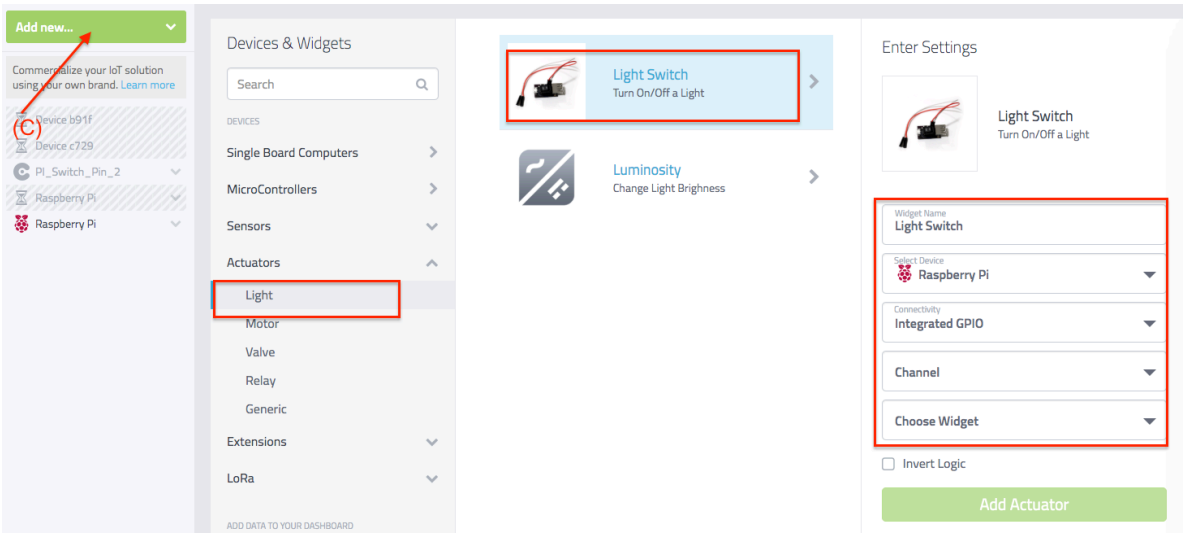
4.2. Remotely Control RPI Using Cayenne IoT Platform

In this section we learn how to remotely control RPI's GPIO ports and turn on/off an LED.

1- Connect an LED to GPIO 23 on your RPI.

2- Go to you remote machine and click on ADD NEW (shown on the figure below as (C) in the figure below). Select **Actuators**, then Light, Light Switch and then set Channel to Channel 23. Choose the widget as button.

3- Note that a widget should appear. Click on the widget and you should see the LED turning on and off!



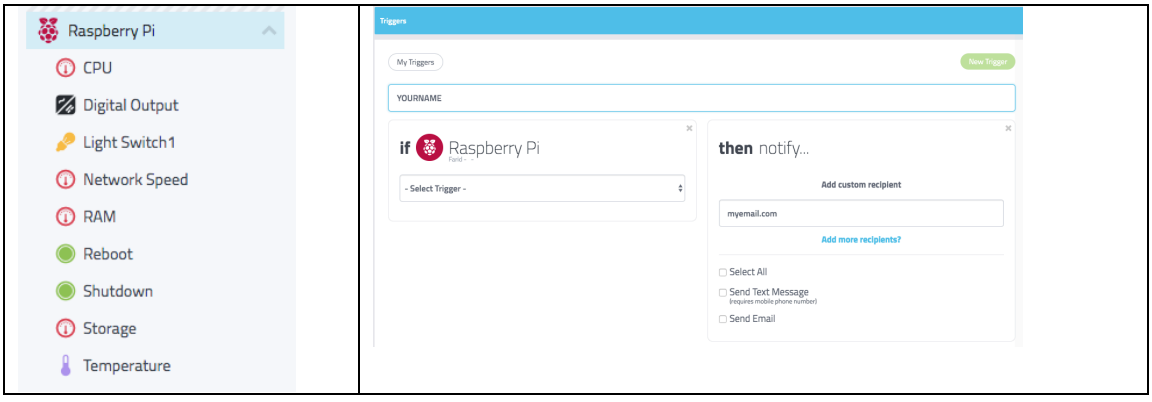
4- Note that under *Raspberry Pi* you see a new *Light Switch*. If you double click on it, you can change the properties of the switch. It is important that the Channel number cannot be changes unless you delete the switch and create a new one. Remember to save any change you make.

4.3. Send Notification Using Cayenne IoT Platform

In this section we learn how to send an email when a specific condition is triggered.

1- Click on ADD NEW…. and select **Trigger**.

2- Drag RASPBERRY PI Device and drop it into box after IF statement, as shown below. Then, select a trigger device such as your Light Switch. . Select SETUP NOTIFICATION and enter your email. Make sure you assign a name to the Trigger and **SAVE** it. Note that every time your light is ON you receive an email! Cool, huh? Always refresh your browser screen to make sure the changes have gone through.



4.4. Sending Sensor Data from RPI to Cayenne IoT Platform

Using its default *magic* somehow Cayenne sends and visualizes temperature and CPU data. The question is how is it done? In this section we develop our own Python code to transmit data from the RPI to Cayenne. Follow the step below:

1- Go to ADD NEW… and then BUILD YOUR OWN THING. Notice the information about MQTT username, password, client, etc. Also note that you are getting a message saying: *Waiting for board to connect….* Make sure you NAME YOUR DEVICE to something like *MyRemoteRPI*.

2- Do SSH onto your PI. Create a directory called `Cayenne`. In this director create a file called *send.py*. Add the code shown below to *send.py* file [4]. Do not change the RED colored code. However, you need to make sure you add your USERNAME, PASSWORD, etc., properly.

3- Run the code on your RPI. You should see some responses similar to this:

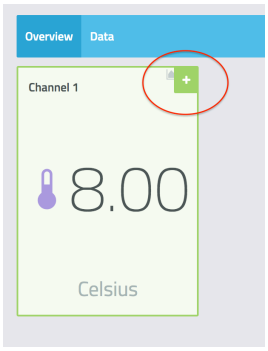
```
PUB v1/511058c0-/things/fdc79b70-fc6e/data/1
temp,c=33

PUB v1/511058c/things/fdc79b70-fc6eb1bf/data/1
temp,c=34
```

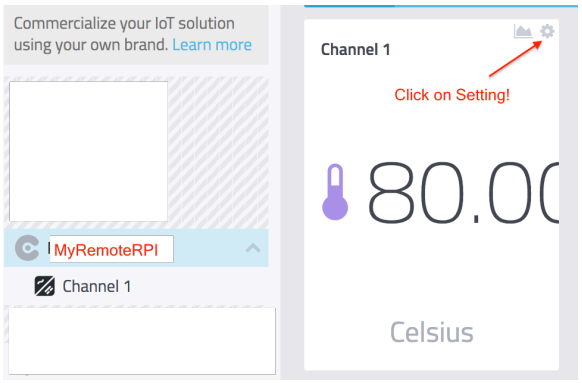
In the above Reponses data/1 represents channel 1. Also note that in the code we are using `client.virtualWrite(1, i, "temp", "c")` `1` represents Channel 1, temp represents the type, and C represent the unit. The attributes of `virtualWrite` can be found here: https://community.mydevices.com/t/data-types-for-cayenne-mqtt-api/3714/7

```
import cayenne.client #Cayenne MQTT Client
import time

# Cayenne authentication info. This should be obtained from the Cayenne Dashboard.
MQTT_USERNAME  = "5118c0-110"
MQTT_PASSWORD  = "4e6939"
MQTT_CLIENT_ID = " 6757b1"

# The callback for when a message is received from Cayenne.
def on_message(message):
  print("message received: " + str(message))
  # If there is an error processing the message return an error string, otherwise return
nothing.

client = cayenne.client.CayenneMQTTClient()
client.on_message = on_message
client.begin(MQTT_USERNAME, MQTT_PASSWORD, MQTT_CLIENT_ID)
# For a secure connection use port 8883 when calling client.begin:
# client.begin(MQTT_USERNAME, MQTT_PASSWORD, MQTT_CLIENT_ID, port=8883)

i=0
timestamp = 0

while True:
  client.loop()
  if (time.time() > timestamp + 10):
    client.virtualWrite(1, i, "temp", "c")
    timestamp = time.time()
    i = i+1
```

[4] Note that Cayenne IP address is 34.236.59.34 as we write this lab. Cayenne is operating on port 8883. All the MQTT messages are encrypted using TLSV1.2. These information can be obtained using `sudo tcpdump -i wlan0 src 34.236.59.34.`

4- Go to your Cayenne browser on the remote machine. Note that by default channel 1 values are being displayed automatically. Click on + and notice that Channel 1 appears under MyRemoteRPI. Click on setting as shown below and change the reading to a graph. Note that we refer to the messages from your remote machine to Cayenne server as *uplink* message and the messages from the server to the end note (RPI) are referred to as *downlink* messages.





The problem for engineering students is that Canyenne makes the entire IoT solution too easy!! But the fact is this simplicity reduces the flexibility in terms of what we can accomplish. For example, if you want to send a message to your phone when the CPU's temperature goes above 50, you cannot easily do it with the existing widgets, at least not yet.

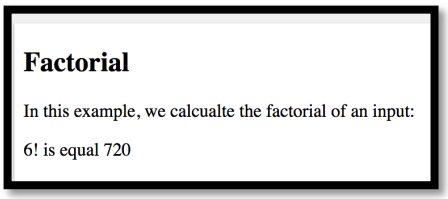
Furthermore, although Cayenne is very simple to utilize, one of its biggest short coming is that in the free version we cannot embed the graph into our own personal web page. Therefore, we cannot make the data easily available to public without providing our passcode for Cayenne.

PROJECT C: Write a Python code that interacts with Cayenne and sends a random number between 1-100 on channel 1 and RPI's temperature on channel 2. Plot the data. If the random number on channel 1 is larger than 60 (or something) you should receive an emails from Cayenne. Make sure you can demonstrate your project.

5. Programming Exercises & Questions

Complete all the projects above and answer the questions below:

1. What is Flask?
2. What is Twilio used for?
3. What is `pip`? What version are you using?
4. What is AWS?
5. What is the difference between PHP and JavaScript in term of the need to have a server?
6. How do you write a conditional statement in HTML?
7. What is Cayenne?
8. Using the messages generated from Bottle can you tell what is the message code if a user tries to access a page that does not exist?
9. What can be done in order for the following URL to work properly: http://127.0.0.1:8080/home.html.
10. Refer to JS tutorial (https://www.w3schools.com/js/js_variables.asp) and learn a simple JS program. Change the program such that the user can enter an input and the program calculate the factorial of the input (e.g., 5!). You output should look exactly like the figure shown. Note that in this case, the input is 6!. You can use the following code to ask for an input. Note that `inputNumber` is just a variable and can be anything (e.g., `Love_EE_465`)




```
var inputNumber = prompt('Please enter an integer');
```

F. Credits

Special thanks to online resources and all SSU students who assisted putting together this lab.

G. References

1- Raspberry Pi Cookbook by Simon - http://www.raspberrypicookbook.com/raspberry-pi-cookbook-ed2/ - Chapter 5
2- Install Geany: https://www.youtube.com/watch?v=jyOW9MbM4Uw
3- Python for Data Analysis by Wes McKinney – Appendix A
4- Great web page to lean many topics using Python: https://www.geeksforgeeks.org/display-hostname-ip-address-python/ - examples are based on Python 3.
5- Python network programing: https://docs.python.org/3/library/ipaddress.html
6- Using Twilio: https://www.hackster.io/matthew-wagner/sending-texts-with-the-raspberry-pi-faaab3

Appendix – Under Construction for Future Lucky Students!

11. Data Visualization Using Google Doc

It is also relatively simple to use Google doc to visualize our data from RPI. Here is an example of what we can do in this section: http://bit.ly/2JKNJx0 [5].

To accomplish this we need to learn about three specific tasks:

1- Be able to plot different types of graphs from Google Drive.
2- Be able to update our Google Drive from Line Command directly.
3- Create a web page which can display our published plot from Google Drive.

Obviously, we bore you continue you need to have a Google account!

Read this to automatically import:
https://stackoverflow.com/questions/26854563/how-to-automatically-import-data-from-uploaded-csv-or-xls-file-into-google-sheet

Very slow in updating – not a streaming data. Good for data that is not fast and streaming. It can be uploaded every 30 min.

Here is the data:

https://docs.google.com/spreadsheets/d/1BjVntUwl-VPanwhawDMmh-nRKZlsiHC9SqFKohJd6TY/edit?usp=sharing

Here is the web page:

http://web.sonoma.edu/users/f/farahman/subpages/junk/junk2.html

```
1.  <html>
2.  <head><title> MY PAGE </title>
3.
4.  <!--****** http://web.sonoma.edu/users/f/farahman/subpages/junk/junk2.html -->
5.  </head>
6.
7.  <body>
8.
9.  <h1>Welcome to My Webpage!</h1>
10. <p>Here is my PI's CPU performance: - make sure you update the page to get the latest.</p>
11. <p>
12. <iframe width="490" height="303" seamless frameborder="0" scrolling="no" src="https:/
    /docs.google.com/spreadsheets/d/e/YOURSHARED_LINK "></iframe>
13. </p>
14.
15. </body>
16. </html>
```

[5] To create a short link you can use https://app.bitly.com/.