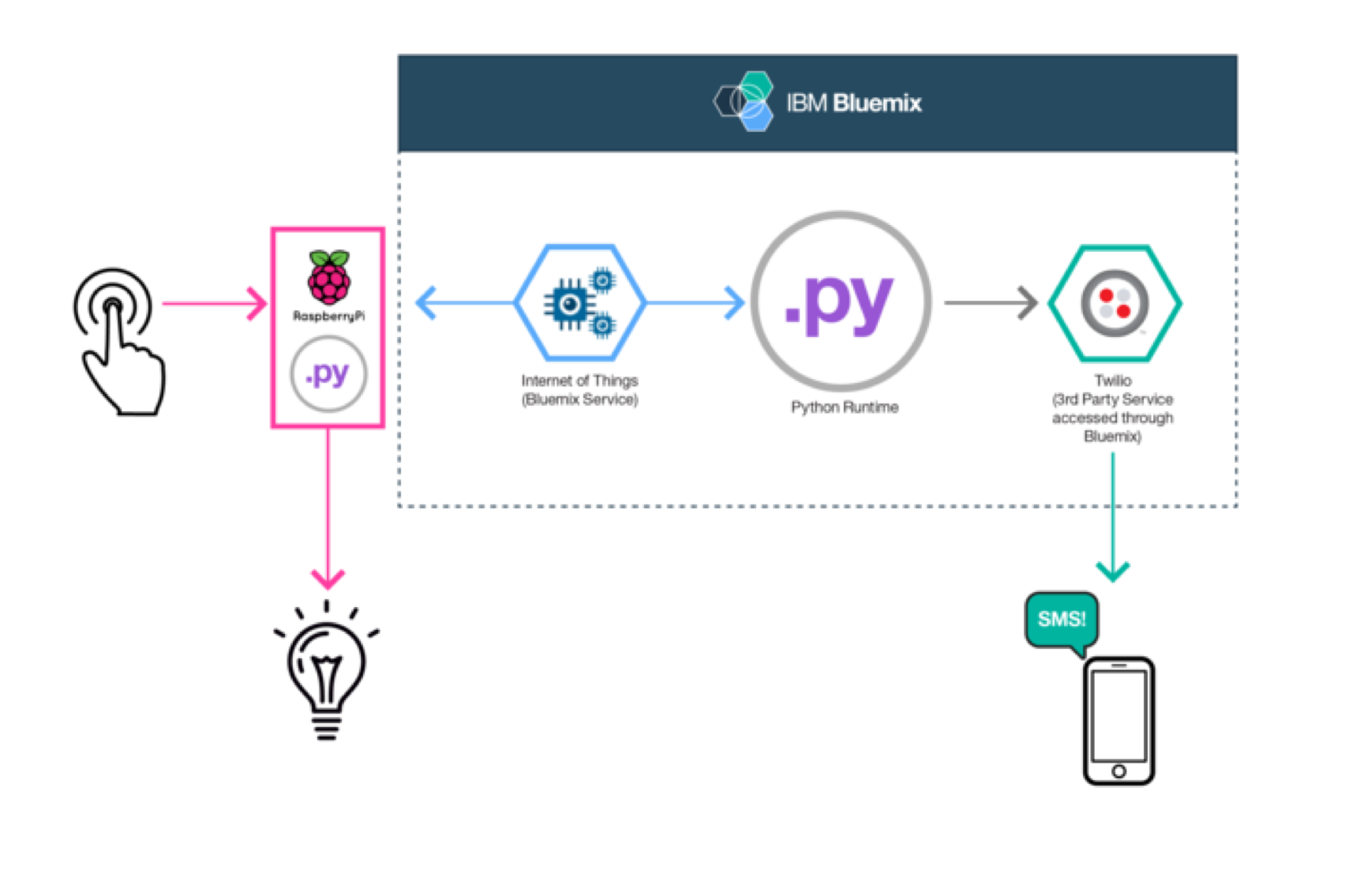
In this tutorial we are going to go through how to integrate a Python Flask webapp in Bluemix with the [Internet of Things Foundation](https://internetofthings.ibmcloud.com/) in Bluemix with a Raspberry Pi and two sensors on the Raspberry Pi. The tutorial also uses Twilio to interact with the Raspberry Pi.

Here is a simple architecture diagram.



In the above diagram there is two flows:

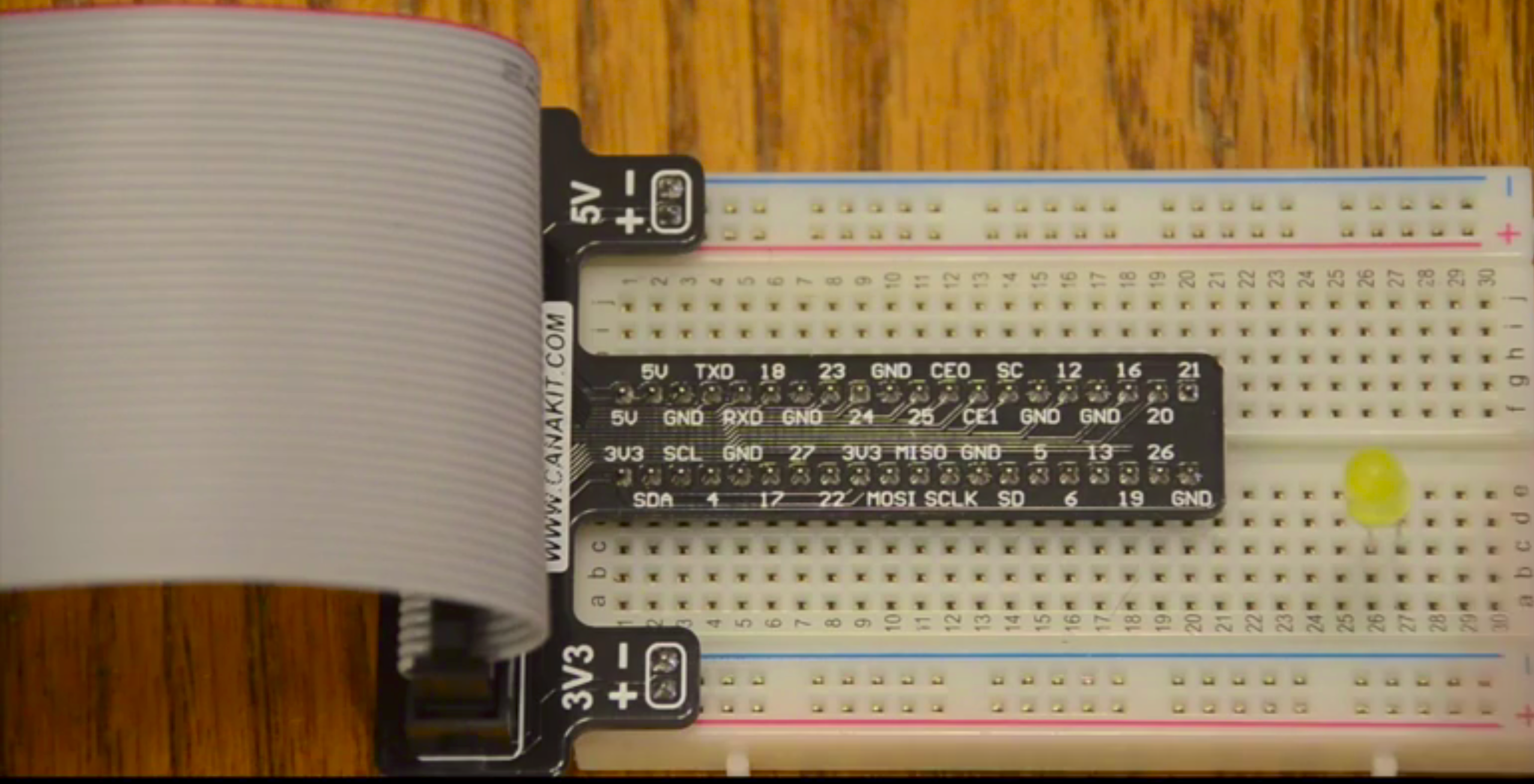
* The first flow involves pressing a HTML button in the Python Flask app in Bluemix to either turn on or off an LED on the Raspberry Pi.
* The second flow involves pressing a physical button on the breadboard on the Raspberry Pi and using the IoT service in Bluemix and our app in Bluemix it will send a text message of our choosing to a phone number we enter in the Python Flask app in Bluemix using Twilio.

## Setup

There is two parts to setting this up. If we take a look at the architecture diagram above we see that there is a Raspberry Pi piece and then a Bluemix piece. We are going to go through the Raspberry Pi piece first then we will go through the Bluemix piece.

## Raspberry Pi Setup Steps

1. Sign up for a Bluemix account, visit [http://bluemix.net](http://bluemix.net/?cm_mmc=developerWorks-_-dWdevcenter-_-bluemix-_-lp) in your web browser and click “Sign-up” in the top right. We require a couple bits of information
2. Wait for an email to arrive, it should only take a couple minutes. There should be a link in the email that says “Click here to complete your registration”, click that. Sign in with the username and password you created from step 1.
3. Obtain a Raspberry Pi, a bread board, wires, a LED, and a button (your best bet is getting a starter kit such as the [Canakit](http://www.canakit.com/))
4. Place an LED in the breadboard (place the shorter side to the left)



5. Place a 220 ohm resistor in the breadboard connecting the short side (the left side of the LED) to the ground rail (the rail with the – sign). Make sure to place the side of the resistor with the red stripes closest to the LED  
*NOTE:* The way the breadboard works is connections run vertically not horizontally

6. Connect a wire from the ground rail to the GND on the pinout board

7. On the right side of the LED, connect a wire to the the port labeled 17.  
*NOTE:* The way the breadboard works is connections run vertically not horizontally.

8. Place the button in the breadboard, it takes a bit of force to press it in all the way. It won’t break it.

9. Connect a wire from the right side of the button to the ground rail.

10. Connect a wire from the ground rail to a GND port, it doesn’t matter which one it is

11. Lastly, connect a wire from the left side of the button to GPIO port 18.

12. Next we need to either use the console for our Raspberry Pi and the terminal application on the device itself or use SSH. I am going to use SSH.

13. To get the IP address of your Raspberry Pi the easiest thing is to open up the terminal app on the Raspberry Pi, it is under accessories -> Terminal

14. Type “ifconfig” without the quotes. If you are connected over Ethernet the IP address will be under eth0, if you are connected over wifi, the IP address will be under wlan0

15. Next we need to ssh into the device.  
—Windows, download Putty and use that to connect to the IP address  
—Mac and Linux, open the Terminal app, type ssh pi@myipaddress where myipaddress is the IP address of the Raspberry Pi, ex. 192.168.1.65

16. The password is raspberrypi

17. We need to run some commands to update our Raspberry Pi, run the following. It will ask you to confirm with the “Y” key

sudo apt-get update

sudo apt-get upgrade

18.Next we need to install a helper library for using GPIO

git clone <git://git.drogon.net/wiringPi>

cd wiringPi

./build

19.Next we need to install the IoT library on the Raspberry Pi

curl -LO <https://github.com/ibm-messaging/iot-raspberrypi/releases/download/1.0.2/iot_1.0-1_armhf.deb>

sudo dpkg -i iot\_1.0-1\_armf.deb

20.We need to get the device ID of our raspberry Pi do this run the following and save the output

service iot getdeviceid

#example output

The device id is b827eba5b236

* We will want to copy the id b827eba5b236, yours obviously will be different

21. Open up a web browser and goto [bluemix.net](http://bluemix.net/), and click on “Catalog” in the top. Scroll down to the very bottom and click “Internet of Things”.

22. Give the service a name, use iot-python (you must do this exactly or later things won’t work), for App choose “Leave unbound”, click “Create”.

23. On the left of the next page click “Launch dashboard”.

24. At the top click on “Devices”.

25. Click “Add Device”.

26. For the second field (it says e.g. mydevice type) type in exactly raspberrypi (you need to have it spelled like this or there will be issues). For device ID paste in the device ID we got from step 20, mine is b827ba5b236. Click continue.

27. On the next page it will show something like below, copy this and switch back to terminal on your Raspberry Pi.

|  |  |
| --- | --- |
| 1  2  3  4  5 | org=pwftki  type=raspberrypi  id=000000000000  auth-method=token  auth-token=cXQaGx8o!a9HwxM-ka |

28.Choose your favorite text editor but I am going to use vi. Type the following to open the file we want to edit.

|  |  |
| --- | --- |
| 1 | sudo vi /etc/iotsample-raspberrypi/device.cfg |

29. To paste the text press the “i” key. Then paste the text, this will depend on the OS you are on.

30. To save the file hit the “Esc” key. Then Type “:wq” and then press “Enter”. That will save the file.

31. Let’s restart the IoT service on our Raspberry Pi to start sending the data to the IoT service

sudo service iot restart

32. Next we need to download the Python code to run on the Raspberry Pi, run the following:

cd ~

git clone <https://github.com/IBM-Bluemix/python-iot-raspberry-> [pi.git](https://github.com/IBM-Bluemix/python-iot-raspberry-pi.git)

cd python-iot-raspberry-pi

33. Now we need to install the package manager for Python:

|  |
| --- |
| sudo apt-get install python-pip  sudo pip install -r requirements.txt |

34. One last bit, we need a config file for our app before we can start it. Lets run the following:

vi ~/device.cfg

35. Remember this is vi again, so remember the shortcuts for inserting text and saving it from step 29,30. The contents of the file should look something like below, replacing yourapikey, yourdeviceid, youriotorg, and yourapitoken with the correct values. To generate the API key and token we can get them from going back to the web browser and going to the Internet of Things Foundation. At the top click on API Keys. Click “New API Key”. The values that it gives you will be the values you use for the yourapitoken and yourapikey. yourdeviceid is the value we got from step 20  
youriotorg is from step 27, it is also in your config file; in my case my value is pwftki. Let’s save the file (remember, “Esc”, “:wq”, “Enter”)

[application]

org=youriotorg

id=yourdeviceid

auth-method=apikey

auth-key=yourapikey

auth-token=yourauthtoken

36. OK, so now we can launch/start the app on the Raspberry Pi. To do that run the following:

|  |  |
| --- | --- |
| 1  2 | cd ~/python-iot-raspberry-pi  python client.py |

## Bluemix App Setup Steps

1. We will need to install the Cloud Foundry CLI to deploy our app to Bluemix. To do this head over to <https://github.com/cloudfoundry/cli/releases>. Choose the appropriate installer for your platform, download it and run the installer.
2. So let’s open up a new terminal Window, we will need to do the rest on our development machine/laptop.
3. We need to download the Python code for our app. Run the following. If you don’t have git installed follow these instructions [here](http://git-scm.com/book/en/v2/Getting-Started-Installing-Git).

git clone git clone <https://github.com/IBM-Bluemix/python-iot-raspberry-pi.git>

cd python-iot-raspberry-pi

4. In your web browser go back to the tab that you have Bluemix open with. In the top click “Catalog”, scroll down to find “Twilio”. Click on that.

5. If you already have a Twilio account and want to use that skip to step 6. If you don’t have a Twilio account on the right hand side click “Register at Twilio”.

6. Twilio will ask you for some information and you will need to verify your phone number. This is important as only numbers verified with the free plan will work with your app. If you want to be able to text any number you need to pay for Twilio, its $1/month/phone number.

7. Once you are signed up for your account, head to <https://www.twilio.com/user/account/voice-messaging>. Near the top right there will be a twistie that says “Show API Credentials”, click that. There is two pieces of information here, the Account SID and Auth Token. We will need to copy these back to the Bluemix tab we have open with Twilio.

8. Paste your Account SID in the field in Bluemix that says Account SID, paste your Auth Token in the Auth Token field.

9. For the Service name you must type in this exactly iot-twilio

10. For the App, choose “Leave Unbound”

11. Click Create.

12. Switch back to the terminal that you have on your local dev machine, not your Raspberry Pi.

13. We need to login to Bluemix, to do that, type cf login -a https://api.ng.bluemix.net. It will ask you for your username and password. This is from step 1 in the Raspberry Pi section above.

14. To deploy our application all we need to do now is type cf push myappname where myapp name is a unique name you choose for your app.  
NOTE: If you get an error mentioning a route is taken, choose a different name and run cf push with a new app name.

15. It will take about a minute or two to deploy your application but eventually you will get some output that looks like the following:

requested state: started

instances: 1/1

usage: 1G x 1 instances

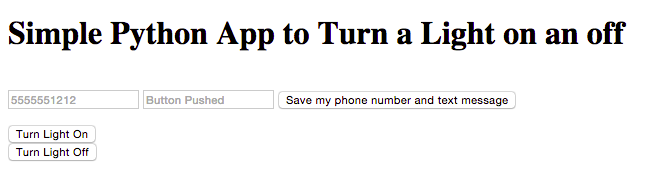
urls: testapp-jbs.mybluemix.net

last uploaded: Fri Jul 31 00:25:17 UTC 2015

stack: lucid64

buildpack: SDK for Node.js(TM) (ibm-node.js-0.12.7)

If everything went well you should have a page that looks like the following:



An important note here, do not press the button on the Raspberry Pi until you enter a phone number and text message here, if you do the app will crash. If you do this you can restart you app with cf restart myappname, where myappname is the name of the app you chose above.

When you click the “On” button it should turn the light on for you. If you press the “Off” button it should turn the light off.