SOFE 4610U: Design and Analysis of IoT

Lab #5: Using Docker for Automated System Deployment on the Raspberry Pi

Devante Wilson and Youssef Osman
100554361 and 100715637

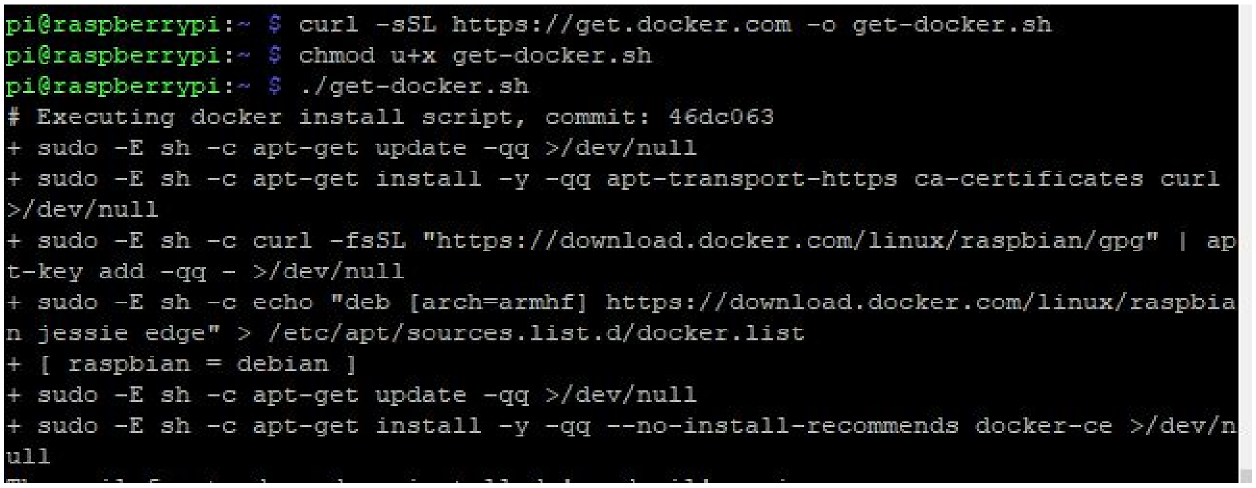
Introduction

The proceedings of this lab report show our lab team seeking to understand the design goals of *Docker*. We will demonstrate how to setup *Docker* on the Raspberry Pi. Additionally, we will see how to download and run a Docker image from Docker Hub. Finally, we will design our own custom Docker image for automated deployment of our IoT solution.

Lab Activity

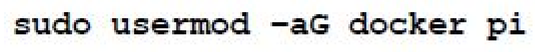
Software Setup

To setup *Docker* onto the RPi, we ran the following commands:



```
pi@raspberrypi:~ $ curl -sSL https://get.docker.com -o get-docker.sh
pi@raspberrypi:~ $ chmod u+x get-docker.sh
pi@raspberrypi:~ $ ./get-docker.sh
# Executing docker install script, commit: 46dc063
+ sudo -E sh -c apt-get update -qq >/dev/null
+ sudo -E sh -c apt-get install -y -qq apt-transport-https ca-certificates curl
>/dev/null
+ sudo -E sh -c curl -fsSL "https://download.docker.com/linux/raspbian/gpg" | ap
t-key add -qq - >/dev/null
+ sudo -E sh -c echo "deb [arch=armhf] https://download.docker.com/linux/raspbia
n jessie edge" > /etc/apt/sources.list.d/docker.list
+ [ raspbian = debian ]
+ sudo -E sh -c apt-get update -qq >/dev/null
+ sudo -E sh -c apt-get install -y -qq --no-install-recommends docker-ce >/dev/n
ull
```

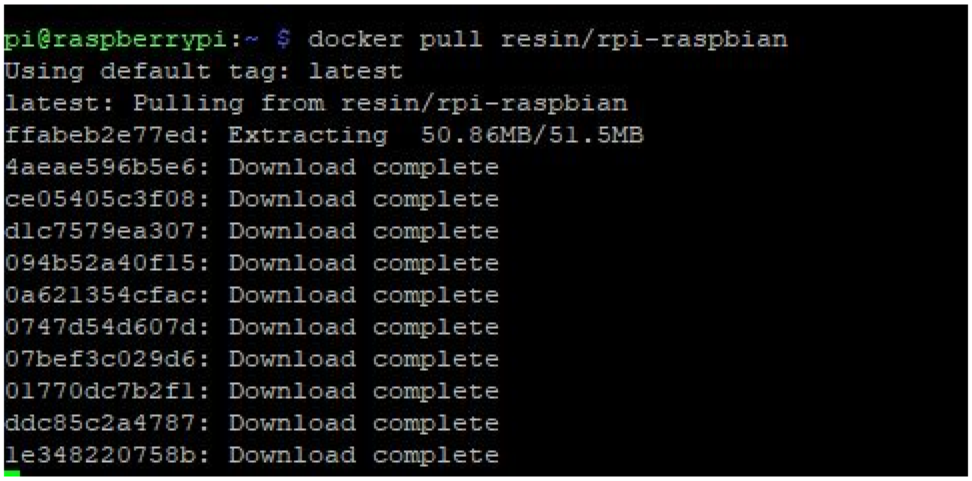
Once the install script finishes executing, we must now add the user “pi” to the *Docker* group so that we, as non-root, may access the *Docker* engine. To do so, we ran the following command:



```
sudo usermod -aG docker pi
```

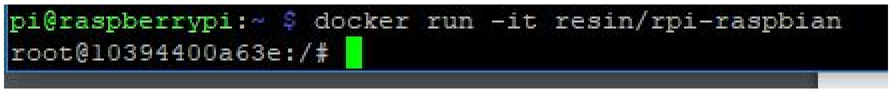
Then we logged out and logged back into the RPi as “pi”.

Once *Docker* is installed, we want to run the *rpi-raspbian Docker* image from Resin.io. But first, the image needs to be downloaded from the *Docker* Hub. To do so, the following was run:



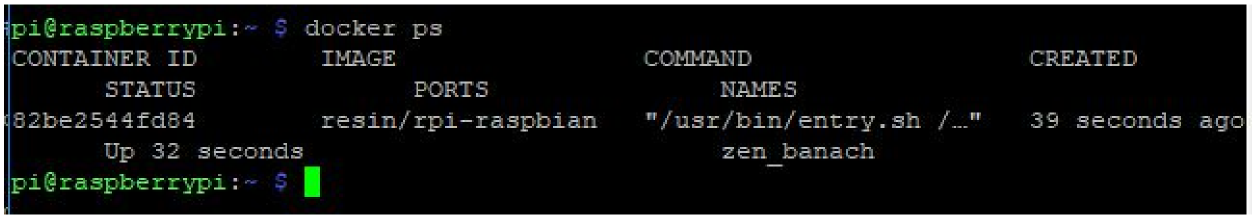
```
pi@raspberrypi:~ $ docker pull resin/rpi-raspbian
Using default tag: latest
latest: Pulling from resin/rpi-raspbian
ffabeb2e77ed: Extracting  50.86MB/51.5MB
4aeae596b5e6: Download complete
ce05405c3f08: Download complete
d1c7579ea307: Download complete
094b52a40f15: Download complete
0a621354cfac: Download complete
0747d54d607d: Download complete
07bef3c029d6: Download complete
01770dc7b2f1: Download complete
ddc85c2a4787: Download complete
1e348220758b: Download complete
```

We can now start the image and obtain an interactive bash shell. To do so:



```
pi@raspberrypi:~ $ docker run -it resin/rpi-raspbian
root@10394400a63e:/#
```

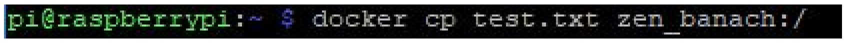
We will now list the running *Docker* containers. We executed the following:



```
pi@raspberrypi:~ $ docker ps
CONTAINER ID        IMAGE                 COMMAND                   CREATED
      STATUS              PORTS               NAMES
82be2544fd84        resin/rpi-raspbian   "/usr/bin/entry.sh /…"   39 seconds ago
      Up 32 seconds                           zen_banach
pi@raspberrypi:~ $
```

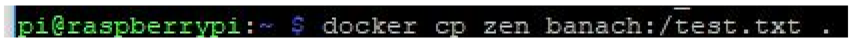
We made note of the assigned container name: zen_banach

To copy a file **into** the *Docker* container, we executed this command:



```
pi@raspberrypi:~ $ docker cp test.txt zen_banach:/
```

To copy a file **out** of the *Docker* container, we executed this command:

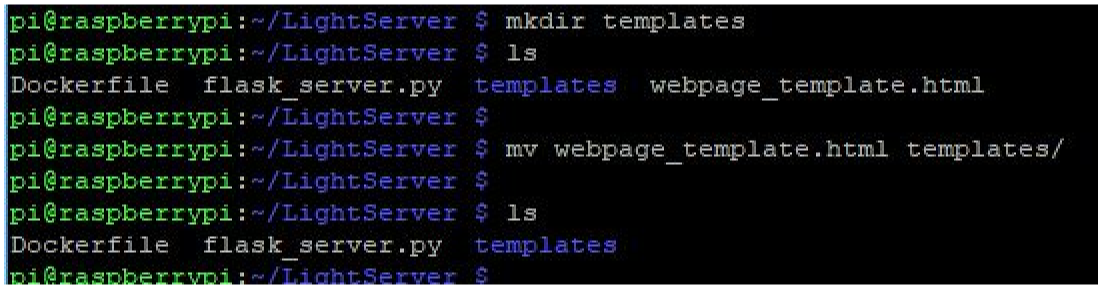


```
pi@raspberrypi:~ $ docker cp zen_banach:/test.txt .
```

We wrote a **Dockerfile** that will setup a server which will read data from the Cheerlights channel on ThingSpeak and set the background of a webpage accordingly. The server files are from Blackboard (**flask_server.py and webpage_template.html**)

We created a new directory called "LightServer", and switched into it via the following commands:

We copied the **flask_server.py** into this directory. A sub-directory named **templates** was created and then **webpage_template.html** was copied into that directory.



```
pi@raspberrypi:~/LightServer $ mkdir templates
pi@raspberrypi:~/LightServer $ ls
Dockerfile  flask_server.py  templates  webpage_template.html
pi@raspberrypi:~/LightServer $
pi@raspberrypi:~/LightServer $ mv webpage_template.html templates/
pi@raspberrypi:~/LightServer $
pi@raspberrypi:~/LightServer $ ls
Dockerfile  flask_server.py  templates
pi@raspberrypi:~/LightServer $
```

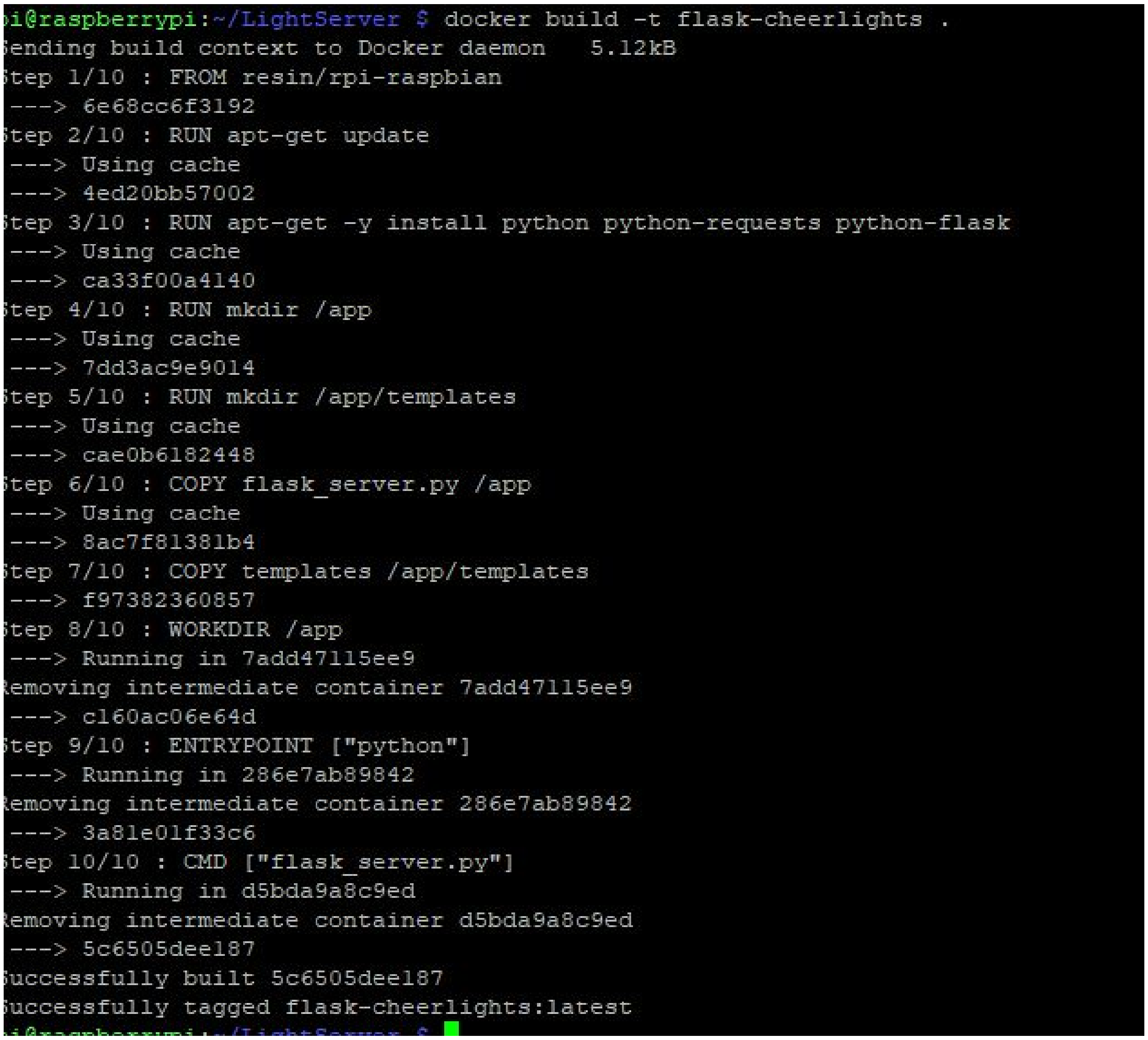
We created a new file named **Dockerfile** and edited it via the following commands:

```
touch Dockerfile
nano Dockerfile
```

We added these lines to the **Dockerfile**:

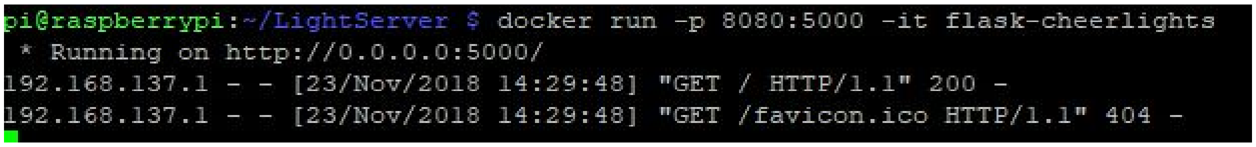
```
FROM resin/rpi-raspbian
RUN apt-get update
RUN apt-get -y install python python-requests python-flask
RUN mkdir /app
RUN mkdir /app/templates
COPY flask_server.py /app
COPY templates /app/templates
WORKDIR /app
ENTRYPOINT ["python"]
CMD ["flask_server.py"]
```

We built our new *Docker* image with this command:



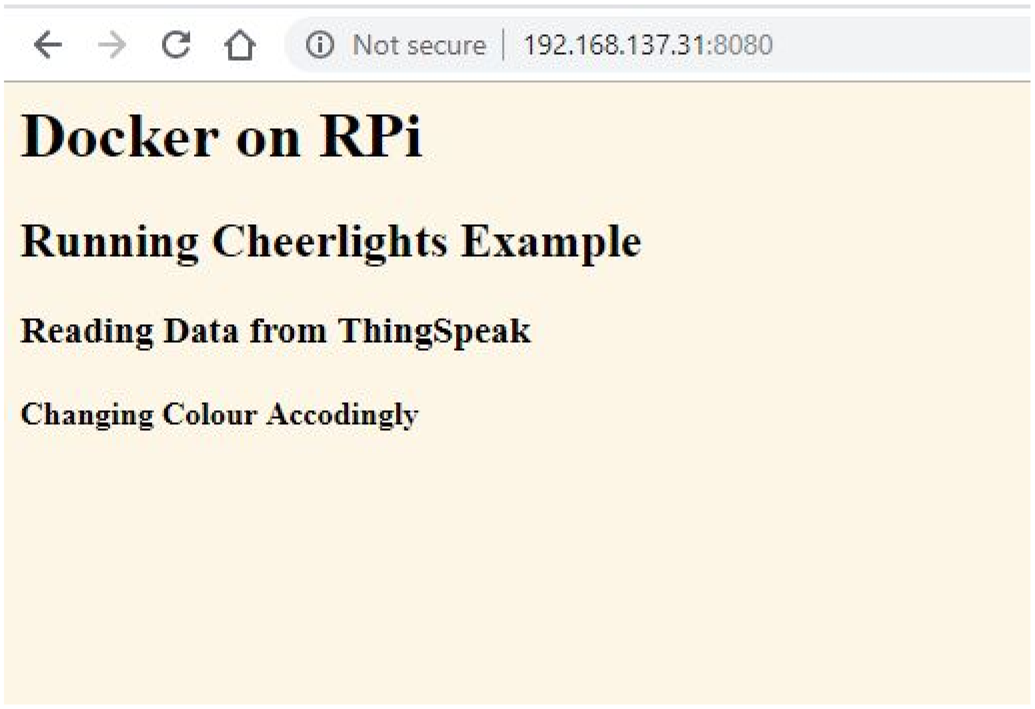
```
i@raspberrypi:~/LightServer $ docker build -t flask-cheerlights .
ending build context to Docker daemon   5.12kB
tep 1/10 : FROM resin/rpi-raspbian
 ---> 6e68cc6f3192
tep 2/10 : RUN apt-get update
 ---> Using cache
 ---> 4ed20bb57002
tep 3/10 : RUN apt-get -y install python python-requests python-flask
 ---> Using cache
 ---> ca33f00a4140
tep 4/10 : RUN mkdir /app
 ---> Using cache
 ---> 7dd3ac9e9014
tep 5/10 : RUN mkdir /app/templates
 ---> Using cache
 ---> cae0b6182448
tep 6/10 : COPY flask_server.py /app
 ---> Using cache
 ---> 8ac7f81381b4
tep 7/10 : COPY templates /app/templates
 ---> f97382360857
tep 8/10 : WORKDIR /app
 ---> Running in 7add47115ee9
emoving intermediate container 7add47115ee9
 ---> c160ac06e64d
tep 9/10 : ENTRYPOINT ["python"]
 ---> Running in 286e7ab89842
emoving intermediate container 286e7ab89842
 ---> 3a81e01f33c6
tep 10/10 : CMD ["flask_server.py"]
 ---> Running in d5bda9a8c9ed
emoving intermediate container d5bda9a8c9ed
 ---> 5c6505dee187
uccessfully built 5c6505dee187
uccessfully tagged flask-cheerlights:latest
```

Will ran the image with this command:



```
pi@raspberrypi:~/LightServer $ docker run -p 8080:5000 -it flask-cheerlights
 * Running on http://0.0.0.0:5000/
192.168.137.1 - - [23/Nov/2018 14:29:48] "GET / HTTP/1.1" 200 -
192.168.137.1 - - [23/Nov/2018 14:29:48] "GET /favicon.ico HTTP/1.1" 404 -
```

We loaded the webpage in the browser by inputting the RPi IP address (the address we used for SSH) into the URL bar and got the following output.



Docker on RPi

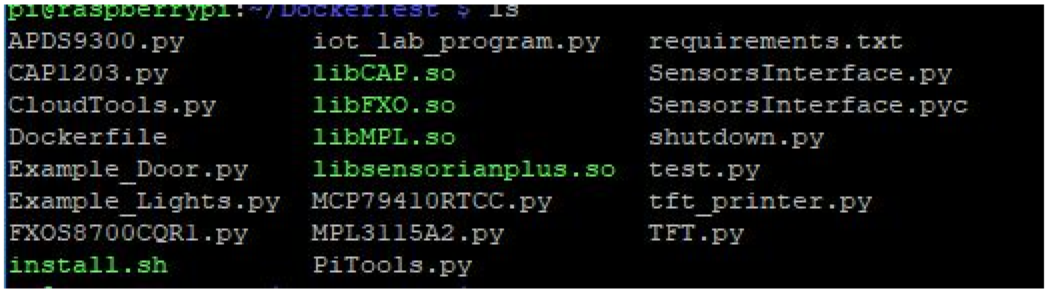
Running Cheerlights Example

Reading Data from ThingSpeak

Changing Colour Accodingly

Mini IoT Project

Copy all of the SensorsInterface/Python files into the Docker directory. Also copy /usr/lib/libbcm2835.so to the directory.



```
pi@raspberrypi:~/DockerTest $ ls
APDS9300.py          iot_lab_program.py    requirements.txt
CAP1203.py           libCAP.so             SensorsInterface.py
CloudTools.py        libFXO.so             SensorsInterface.pyc
Dockerfile           libMPL.so             shutdown.py
Example_Door.py      libsensorianplus.so   test.py
Example_Lights.py    MCP79410RTCC.py       tft_printer.py
FXOS8700CQR1.py      MPL3115A2.py          TFT.py
install.sh           PiTools.py
```

Copy the below lines into the Dockerfile:

FROM resin/rpi-raspbian

RUN apt-get update

RUN apt-get install -y python python-requests build-essential

COPY libbcm2835.so /usr/lib

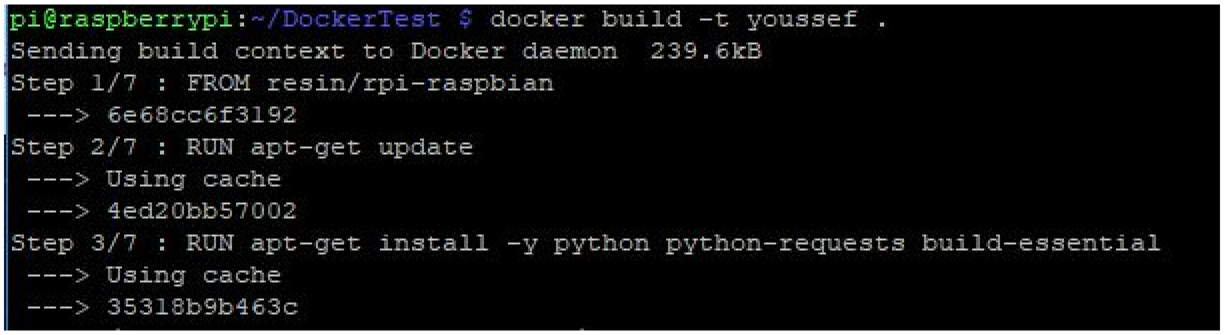
COPY iot_lab_program.py /

COPY . /

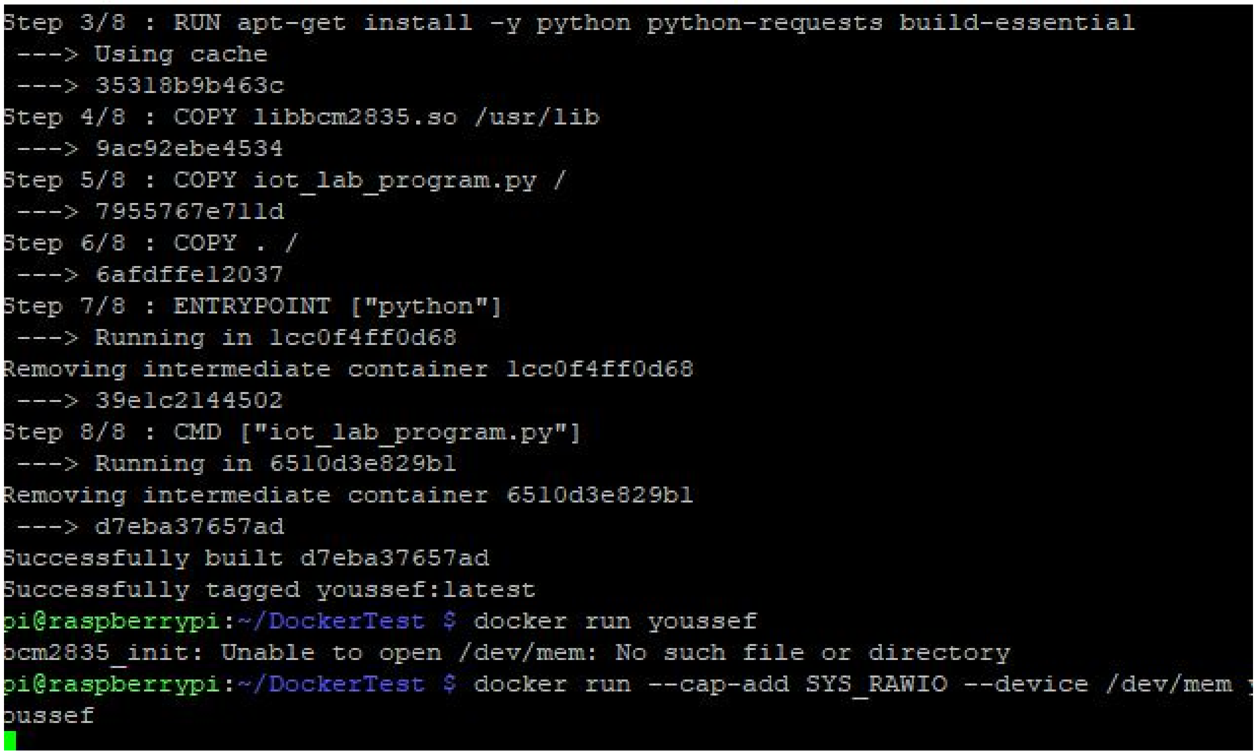
ENTRYPOINT ["python"]

CMD ["iot_lab_program.py"]

Then build the image as below image:



```
pi@raspberrypi:~/DockerTest $ docker build -t youssef .
Sending build context to Docker daemon  239.6kB
Step 1/7 : FROM resin/rpi-raspbian
 ---> 6e68cc6f3192
Step 2/7 : RUN apt-get update
 ---> Using cache
 ---> 4ed20bb57002
Step 3/7 : RUN apt-get install -y python python-requests build-essential
 ---> Using cache
 ---> 35318b9b463c
```



```
Step 3/8 : RUN apt-get install -y python python-requests build-essential
 ---> Using cache
 ---> 35318b9b463c
Step 4/8 : COPY libbcm2835.so /usr/lib
 ---> 9ac92ebe4534
Step 5/8 : COPY iot_lab_program.py /
 ---> 7955767e711d
Step 6/8 : COPY . /
 ---> 6afdffe12037
Step 7/8 : ENTRYPOINT ["python"]
 ---> Running in 1cc0f4ff0d68
Removing intermediate container 1cc0f4ff0d68
 ---> 39e1c2144502
Step 8/8 : CMD ["iot_lab_program.py"]
 ---> Running in 6510d3e829b1
Removing intermediate container 6510d3e829b1
 ---> d7eba37657ad
Successfully built d7eba37657ad
Successfully tagged youssef:latest
pi@raspberrypi:~/DockerTest $ docker run youssef
bcm2835_init: Unable to open /dev/mem: No such file or directory
pi@raspberrypi:~/DockerTest $ docker run --cap-add SYS_RAWIO --device /dev/mem y
oussef
```

Conclusion

By the end of the lab session, our team successfully learned how to setup *Docker* on the Raspberry Pi. Additionally, we saw how to download and run a Docker image from Docker Hub. Finally, we designed our own custom Docker image for automated deployment of our IoT solution.