# IOT Lab Assignment 5

Flask restful api

#### Flask (restful api)



A RESTful API is an application program interface (API) that uses HTTP requests to GET,
 PUT, POST and DELETE data.

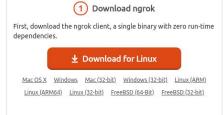
sudo pip3 install flask flask-restful

## Ngrok (get your pi a public url for your api)

- You will need to make a free account.
- https://ngrok.com

#### Download & setup ngrok

Get started with ngrok in just a few seconds.









### ngrok

- 1. Sudo dnf install unzip
- 2. curl --silent --remote-name --location -k https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-arm.zip
- 3. unzip ngrok-stable-linux-arm.zip
- 4. ./ngrok authtoken {auth token}

#### Run Ngrok, Get url, kill Ngrok

• ./ngrok http 5000 -log=stdout > /dev/null &

```
[rpi@localhost class5]$ ./ngrok http 5000 -log=stdout > /dev/null &
[1] a1452 t_url = "http://localhost:4040/api/tunnels" #Url with tunnel
```

• curl -s http://localhost:4040/api/tunnels | grep public\_url (look for https)

```
[rpi@localhost class5]$ curl -s http://localhost:4040/api/tunnels | grep public_url
{"tunnels":[{"name":"command line","uri":"/api/tunnels/command_line","uri":"/api/tunnels/command_line","uri":"/api/tunnels/command_line","uri":"/api/tunnels/command_line","uri":"/api/tunnels/command_line","uri":"/api/tunnels/command_line","uri":"/api/tunnels/command_line","uri":"/api/tunnels/command_line","uri":"/api/tunnels/command_line":["count":3,"rate1":1.5100269722698767e-8,
"rate5":0.0004965107274325478, "rate15":0.001225135678566729, "p50":14028042, "p99":14072521, "p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p999":14072521,"p99":14072521,"p99":14072521,"p99":14072521,"p999":1407252
```

Kill -9 1452 (if you want to stop ngrok for some reason)

#### Helloworld.py

```
[dpivonka@dhcp-41-217 examples]$ curl -X get https://9aafd449.ngrok.io/
{
    "hello": "world"
}
```

#### Hellouser.py

```
[dpivonka@dhcp-41-217 examples]$ curl -d '{"user":"dan"}' -X post https://9aafd449.ngrok.io/test
{
    "hello": "dan"
}
```

#### Broker

Broker is responsible for handling incoming light sensor data and storing it in the database

It is also responsible for handling incoming messages to turning on and off pi light

```
import paho.mqtt.client as mqtt
import RPi.GPIO as GPIO
from influxdb import InfluxDBClient
import datetime
broker address="192.168.4.19"
                                #broker address (your pis ip address)
GPIO.setmode(GPIO.BCM)
GPIO.setup(18,GPIO.OUT)
def on message(client, userdata, message):
    print(message.topic + " " + str(message.payload)) #print incoming messages
    if message.topic == '/light': -
    if message.topic == '/piled': ==
client = mqtt.Client() #create new client instance
client.connect(broker address) #connect to broker
client.on message=on message #set the on message function
client.subscribe("/light")
client.subscribe("/piled")
dbclient = InfluxDBClient('0.0.0.0', 8086, 'root', 'root', 'mydb')
client.loop start() #start client
try:
        pass #wait for ctrl-c
client.loop stop() #stop client
GPIO.cleanup()
```

#### API

API is responsible for querying the db for light average and returning that value

It is also responsible for sending mqtt messages out to turn on leds (these messages go to the esp and to the broker)

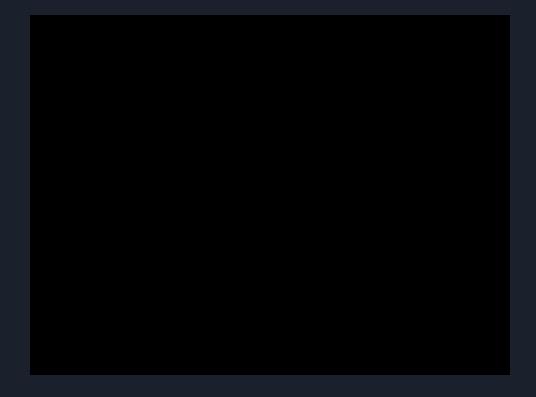
```
import paho.mqtt.client as mqtt
    from influxdb import InfluxDBClient
    from flask import Flask, request, json
    from flask restful import Resource, Api
    import datetime
    broker address="192.168.4.19" #broker address (your pis ip address)
    client = mqtt.Client() #create new client instance
    client.connect(broker address) #connect to broker
11
12
    dbclient = InfluxDBClient('0.0.0.0', 8086, 'root', 'root', 'mydb')
13
    app = Flask(name)
15
    api = Api(app)
17
    class Test(Resource):
        def get(self): ==
        def post(self): =
32
52
    api.add resource(Test, '/test')
53
    app.run(host='0.0.0.0', debug=True)
```

#### Assignment

- Send light sensor values to pi from esp8266
- Store those values in influxdb
- Create restful api
- Post endpoint sets state of led on pi or arduino {'device':'pi', 'state':'on"}
- Get endpoint returns avg light value over last 10 secs

Due by next class

## Assignment video



## Project Ideas

https://iotdesignpro.com/iot-projects

Start thinking about your project