HO CHI MINH CITY, UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEER

Application Based Internet of Things Report - LAB 1

Student: Vũ Minh Quang
ID: 1852699

HỒ CHÍ MINH CITY

Content

1 Introduction

In this first LAB, students are proposed to create a simple Thingsboard backend and Dashboard for an IoT application. Students are supposed to follow steps listed in the Implementation section to finish the first Lab.

2 Implementation

2.1 Step 1: Create account and a device

A refferent video is posted in the link bellow:

`https://www.youtube.com/watch?v=kWF5ZSkXfE4`

Please login to Thingsboard and create a device, named **IoT Project** for instance.

2.2 Step 2: Implement python source code

In this step, please create a github account and upload your source code to github. The link of your source code is required to present in this report.

`https://github.com/quangvuminh2000/IOT_CC01_SEM212/tree/master`

The manual video for this step can be found at:

`https://www.youtube.com/watch?v=pJKTgCq_J7Y`

At this step, two random values simulated for the temperature and humidity are sent to the server every 10 seconds.

2.3 Step 3: Simple Thingsboard dashboard

Design a simple dashboard with 2 labels to display the values of temperature and humidity. The manual for this step can be found at:

`https://www.youtube.com/watch?v=8eQOag5Ymfo`

2.4 Step 4: Use advanced UI in Thingsboard

Please use a UI in the Analogue Gause and Digital Gause in your dashboard, to present the value of temperature and humidity.

Publish your dashboard and present the link in this report

`https://demo.thingsboard.io/dashboard/bb04beb0-7806-11ec-9ed9-f9294d38ab44?publicId=0d613390-6d09-11ec-8159-03103585248e`

A manual video is posted at:

`https://www.youtube.com/watch?v=LFEllRi-5iU`

2.5 Step 5: Add a map to the dashboard

Finally, add a map to your dashboard. In this case, the **longitude** and **latitude** are required in your python source code. At this step, the latitude and longitude can be set to 10.8231 and 106.6297.

A manual video is posted at:

`https://www.youtube.com/watch?v=0XMqH8mdWi0`

3 Extra point (1 point)

Dynamic update the current longtitude and latitude. Explain your implementation in python source code such as the library which is used, some main python source code to get the value of longtitude and latitude.

3.1 Explanation

I use **geocoder** library to get the location of my router which is located inside my house.

The code I use is:

```python
import geocoder
latitude, longitude = geocoder.ip('me').latlng
```

Where `geocoder.ip('me')` means getting the information of my IP on the geolocation. Then I can get the latitude and the longitude according to my IP on google map simply by accessing the `latlng` attribute.

There is also an `generate_random_location` function inside the *utils.py* script. What it does is that, instead of letting the location being static at the current ip point on the map, the location will randomly move to a location in the 20 meters radius around the last position.

Here is my code for the `generate_random_location` function:

```python
def generate_random_location(x0, y0, r):
    """
    Generate the random location within the radius r of the position (↩
        x0, y0)

    Parameters
    ----------
    x0 : float
        Longitude of the starting location
    y0 : float
        Latitude of the starting location
    r : float
        The radius of the surrounding location (meters)

    Returns
    -------
    Tuple[float, float]
        New random position in the radius
    """

    # Convert radius to degree
    r_d = r/111000

    # Generate 2 iid values
    u, v = np.random.uniform(0, 1, 2)

    # Create a random manhattan distance from the current position
    w = r_d * math.sqrt(u)

    # Create a random orientation of the new position
    t = 2*math.pi*v

    # Generate new x,y such that x,y,w is 3 sides of a right triangle
    x = w*math.cos(t)
    y = w*math.sin(t)

    # Adjust the x-coordinate for the shrinking of the east-west ↩
      distances
    new_x = x/math.cos(math.radians(y0))

    # New position
    new_longitude = new_x + x0
    new_latitude = y + y0

    return (new_longitude, new_latitude)
```

Generally what it does is that it will first generate the random distance w in the range of $w \in (0 - r]$ where r is the input radius. Then it will create a random angle t such that $0 < t <= 2\pi$. Finally, it will create 2 number x, y such that x, y, w is 3 side of the right triangle where w is the length of the hypotenuse side with respect to angle t. It can easily being seen that the distance of the old and new points $(x0 + x, y0 + y)$ is strictly less than r.