Practica 3 - CoAP

Pulido Bejarano Raymundo

7 de Diciembre del 2020

1. Introducción

En la actualidad cando hablamos de comunicación atreves de internet es imposible no hablar en algún punto del protocolo HTTP, el cuales el encargado de realizar la comunicación en las páginas web, pero sin dudas en la misma conversación se llegaría a hablar sobre REST, y sobre como este nos permite comunicar el Frontend y Backend de una forma sencilla, clara y estandarizada, ya que mediante el uso de sus métodos de operación, sus agentes y su gestión de sesiones, lo convierte en un excelente prospecto para realizar las comunicaciones entre servidor y cliente, pero cuando hablamos de dispositivos de recursos limitados, los cuales comúnmente deben operar en condiciones adversas como un ancho de banda realmente limitado o alguna otra situación, se deben de priorizar el costo computacional, la latencia que este genere y el ancho de banda necesario para realizar la comunicación, con esto en mente se desarrolló el protocolo CoAP, siendo una adaptación de REST, corre sobre HTTP, tomando lo mejor de REST, pero modificándolo para mejorar su rendimiento, empezando desde tomar la elección de implementarlo sobre UDP en vez de TCP, entre muchas otras modificaciones o adaptaciones.

Por lo expuesto anteriormente y mucho más es que el protocolo CoAP es una excelente opción para realzar comunicación en IoT, por lo que para esta práctica se nos encomendó la tarea de realizar la envió de los datos censados mediante los sensores en la tarjeta SenseHat, mediante el protocolo CoAP, para de esta forma enviar los datos a una nube, la cual para finalidades de esta práctica será la nube de ThingsBoard

2. Desarrollo

Para el desarrollo de esta práctica decidí utilizar el lenguaje de programación en su versión 3.8, a su vez algunas bibliotecas externas desarrolladas por la comunidad.

Sin olvidar que también ilustraré una manera sumamente sencilla de implementación del mismo programa haciendo uso de Node-Red

A su vez se necesita crear una cuta en la plataforma de Thingsboard y configurar el dispositivo al que reciba los datos como publico.

A continuación, les mostraré el código desarrollado para realizar la comunicación que se nos solicitó, par que posteriormente realice una breve explicación y mostrar los resultados obtenidos

```
1 #!/usr/bin/env python
2 from sense_hat import SenseHat
3 import socket
4 import sys
```

```
5  import json
6
7  from coapthon.client.helperclient import HelperClient
8  from coapthon.utils import parse_uri
9
10 __author__ = "Pulido Bejarano Raymundo"
11
12 client = None
13 sense = SenseHat()
14
15 """
16     Funciones del SenseHat
17 """
18
19 def GetEnviromentalValues():
20     """
21         return -> Humedad, Presion, Temperatura
22
23         Humedad         -> Humedad Relativa al ambiente
24         Presion         -> Presion actual en Milibars
25         Temperatura     -> Obtiene la temperatura del ambiente en
               Grados Celsius
26
27     """
28     global sense
29
30     #return (sense.get_humidity(),sense.get_pressure(),sense.
           get_temperature())
31     return {"Humedad":sense.get_humidity(),"Presion":sense.
           get_pressure(),"Temperature":sense.get_temperature()}
32
33 def GetMoveValues_raw():
34     """
35         return -> (OrientatioAxis),(GyroscopioAxis),(
               AccelertionAxis)
36
37         OrientatioAxis      -> (Pitch (X), Roll (Y), Yaw (Z)) axis
38         GyroscopioAxis      -> (X,Y,Z) representing the rotational
               intensity of the axis in radians per second.
39         AccelertionAxis     -> (X,Y,Z) representing the
               acceleration intensity of the axis in Gs.
40
41     """
42     global sense
43     sense.set_imu_config(True,True,True)
44     OrientatioAxis  = tuple(sense.get_orientation().values())
45     GyroscopioAxis  = tuple(sense.get_gyroscope_raw().values())
46     AccelertionAxis = tuple(sense.get_accelerometer_raw().values())
47
48     return (OrientatioAxis,GyroscopioAxis,AccelertionAxis)
49
50 def GetMoveValues():
51     """
52         return -> compass,(Gyroscope),(Accelerometer)
53
54         compass         -> The direction of North
```

```
55          Gyroscope       -> (Pitch (X), Roll (Y), Yaw (Z))
               Representing the angle of the axis in degrees.
56          Accelerometer   -> (Pitch (X), Roll (Y), Yaw (Z))
               Representing the angle of the axis in degrees.
57      """
58      global sense
59      Gyroscope = tuple(sense.get_gyroscope().values())
60      Acceleration = tuple(sense.get_accelerometer().values())
61      return (sense.get_compass(),Gyroscope,Acceleration)
62
63  def main():
64      global client
65      path  = None
66      payload = GetEnviromentalValues()
67      url = "coap://demo.thingsboard.io/api/v1/hgT6xWy23FyQ1lS0zSK4/
           telemetry"
68      host, port, path = parse_uri(url)
69      client = HelperClient(server=(host, port))
70      for n in range(10):
71          client.post(path, json.dumps(payload),no_response=True)
72          payload = GetEnviromentalValues()
73
74      client.stop()
75
76  if __name__ == '__main__':
77      main()
```

Codigo desarrollad - practicaCoAP.py

En el código anterior se encuentran diversas funciones, e las cuales las primeras se encargan de obtener los datos de los sensores y regresarlos en el formato necesario el cual es un diccionario o tupla, ahora refiriéndonos a la función main, en esta función primeramente defino algunas variables y genero el primer payload, el cual serán los datos obtenidos del senseHat, después describo la URI del servicio, la estructura necesario para Thingsboard está en su documentación oficial, posteriormente creamos la URI necesaria para el modulo, posteriormente creo el cliente como tal, esto habilita la conexión, y nos permite realizar cualquier tipo de conexión como la post en este caso.

Debajo se crea un ciclo que en este caso realiza el envió de 10 mediciones continuas y después termina.

3. Pruebas de comportamiento

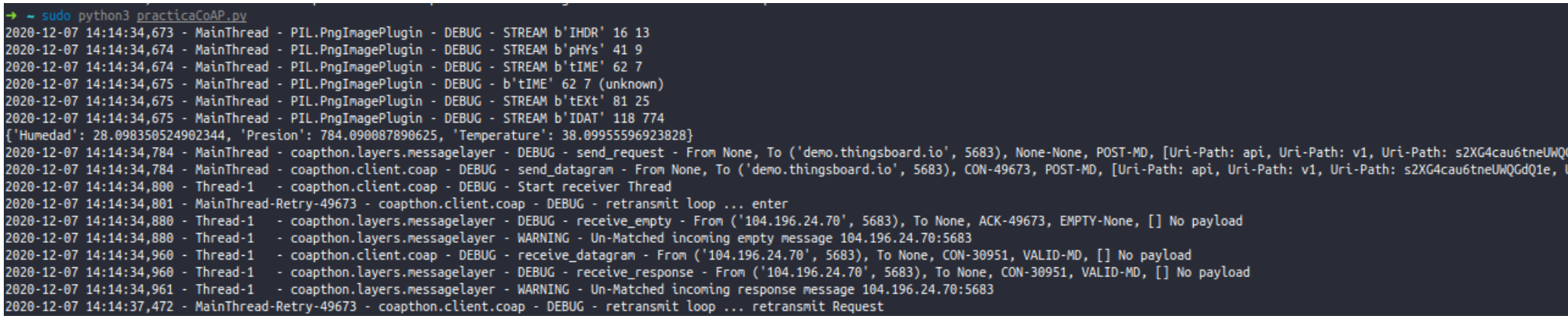

Figura 1: Ejecucion de la practica

En la figura anterior se puede observar la ejecución y el envió de la información a la nube, de forma correcta

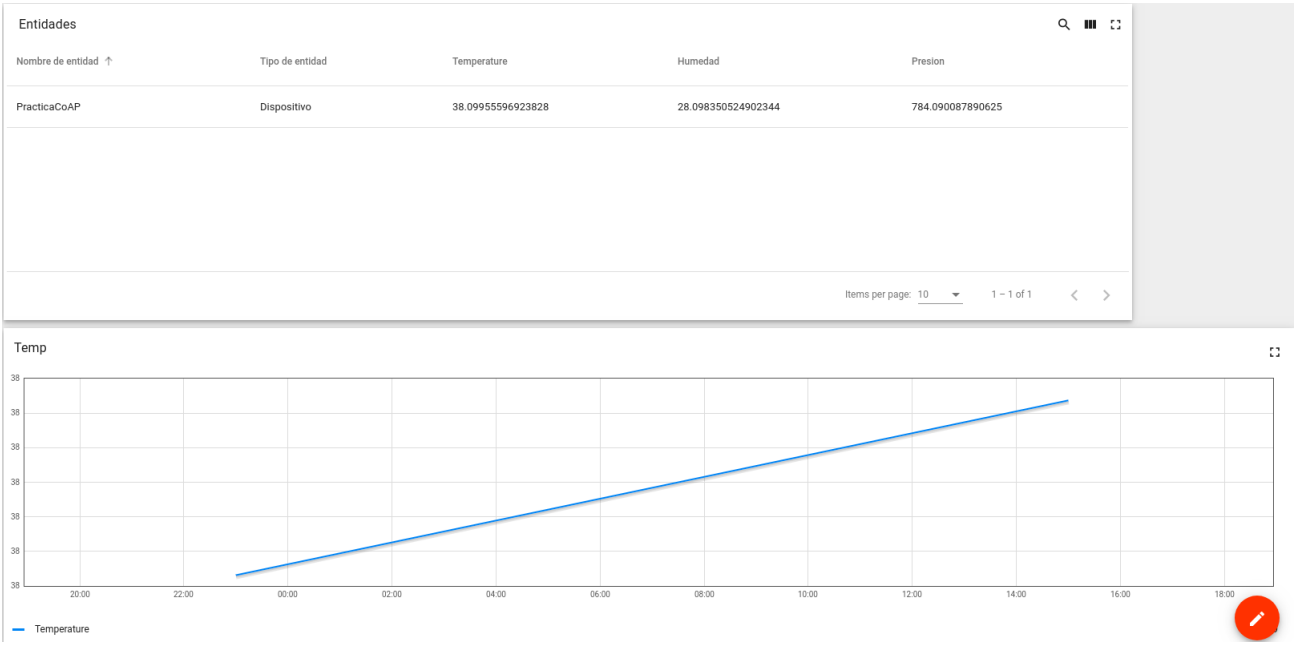

Figura 2: Datos en la nube

En la figura anterior se pueden observar el histórico de los datos que enviado en transcurso entre ayer y hoy, como también los últimos datos recibidos.

A continuación les mostraré la implementación del mismo programa implementado mediante node-red

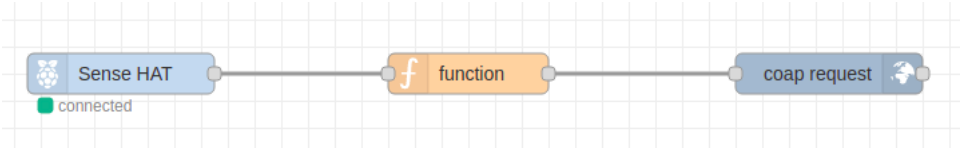

Figura 3: Implementación con node-red

Como se puede apreciar claramente la implementación haciendo uso de node-red, es absolutamente más sencilla que utilizando python, ya que ese incluso a primera vista podrías describir que proceso se está realizando, es asombrosa la sencillez y facilidades que nos brinda node-red

4. Conclusion

Para finalizar este reporte, me gustaría agregar que él puede entender de mejor manera el comportamiento del protocolo CoAP, que por mucho que sea similar a REST, son diferentes, y no se debe de confundir y pensar que un servidor que acepte REST podrá ser accedido mediante CoAP, además la importancia y trasfondo de la URI's y como estas describen todo el manejo de los datos como el protocolo en la Red, nos otorga un gran entendimiento de como es que se realiza la transmisión de datos atreves de la red, y que sin estas cualquier tipo de comunicación sería sumamente complicada y confusa.