Practica 3:
Censado y envió de datos a la nube

Pulido Bejarano Raymundo

6 de Octubre del 2020

1 Introducción

Esta Practica tiene como objetivo realizar el censado de Temperatura, Presion, Humedad, Aceleracion, y Orientacion, mediante una Cumulo de sensores incorporados en la llamada Sense Hat, donde para realizar este sensado se plantea la incorporacion de esta "Sense Hat", con la tarjeta de desarrollo Raspberry pi en su modelo 3b+, estos datos seran procesados mediante el lenguaje de programacion Python en su version 3.8, para posteriormente ser publicados usando el protocolo MQTT, a un broker perteneciente a una "nube" como puede ser Ubidots, como cualquier otra.

1.1 Hardware necesario

:

- Raspberry Pi 3b+
- Sense Hat
- Conexion a Internet

1.2 Sofware necesario

:

- Ssh
- Python 3.8
- Cuenta en la nube a mandar datos (en este caso Ubidots)

1.2.1 Paquetes necesarios para esta practica

:

- sense-hat
- paho-mqtt
- signal
- json

2 Desarrollo

Para desarrollar esta practica, decidi dividirla en dos modulos, el primero modulo estara enfocado en obtener los datos de la Sense Hat, y el segundo modulo estara enfocado en el manejo de la conexion de Python y la raspberry con la nube usando el protocolo MQTT. DOnde una vez desarrollado estos, se procedera a desarrollar el flujo principal del programa que sera el cual cordine toda la operacion del programa.

A su vez se agregaron a esta practica, el uso de argumentos para determinar las variables que seran sensadas y el uso de señales para al momento de temrinar el programa este cierre de una manera estable y ordenada, con el fin de que esta contara con mayor estabilidad y tuviera un comportamiento mas amigable.

Como anotacion previa a la explicacion acontinuacion se encontraran algunas variables "globales" utilizadas para el desarrollo de la practica.

La palabra *globales* esta entre comillas ya que en python no existen variables globales, si no de scope global, lo que refiere a que existen en el scope padre de todos.

```
1  args = parser.parse_args()
2  sense = SenseHat()
3  connected = False
4  BROKER_ENDPOINT = "industrial.api.ubidots.com"
5  PORT = 1883
6  MQTT_USERNAME = "BBFF-4aGyXQAP2KKxtwunvmyKsUHs87BVwe"  #
       Token
7  MQTT_PASSWORD = ""
8  TOPIC = "/v1.6/devices/"
9  DEVICE_LABEL = "Raspberry-Ray"
10 Varibles_Labels = []
11 mqtt_client_G = None
12 Activo = True
```

Variables empleadas en el desarrolo de la practica

El uso de todas ellas sera explicado durante el desarrollo de esta practica.

2.1 Recolección de datos desde la Sense Hat

Respecto a la recoleccion de datos de parte de la Sense Hat, python mediante su modulo "sense-hat" nos ofrece un gran cumulo de funciones que nos facilitan la obtencion de esta informacion.

Pero para el desarrollo de esta practica se decidio encapsular las varibles en dos distintas funciones con el fin de que sea sencillo el obtener los datos corelacionados entre si.

Para la obtenicion de estos datos se hara uso de una variable llamada "sense" la cual es una instancia de la clase de gestion de datos del modlo "sense-hat".

Por lo que estas dos funciones son las Siguientes:

- GetEnviromentalValues (Obtiene las Varibles del entorno como Humedad, Temperatura y Presion)
- GetMoveValues (Obtiene los datos de Orientacion,Giro y Aceleracion del dispositivo)

La función *GetEnviromentalValues* es la siguiente:

```
1  def GetEnviromentalValues():
2      """
3          return -> Humedad, Presion, Temperatura
4
5          Humedad          -> Humedad Relativa al ambiente
6          Presion          -> Presion actual en Milibars
7          Temperatura      -> Obtiene la temperatura del
               ambiente en Grados Celsius
8
9      """
10     global sense
11
12     return (sense.get_humidity(),sense.get_pressure(),sense.
           get_temperature())
```

Funcion GetEnviromentalValues

Esta funcion nos da como retorno una tupla de los valores mensionados anteriormente, lo que nos garantiza debido a las propiedades del lenguaje, que estos valores no podran ser modificados.

La función *GetMoveValues* es la siguiente:

```
1  def GetMoveValues():
2      """
3          return -> compass,(Gyroscope),(Accelerometer)
4
```

```
        compass         -> The direction of North
        Gyroscope       -> (Pitch (X), Roll (Y), Yaw (Z))
            Representing the angle of the axis in degrees.
        Accelerometer   -> (Pitch (X), Roll (Y), Yaw (Z))
            Representing the angle of the axis in degrees.
    """
    global sense

    Gyroscope = tuple(sense.get_gyroscope().values())
    Acceleration = tuple(sense.get_accelerometer().values())

    return (sense.get_compass(),Gyroscope,Acceleration)
```

Funcion GetMoveValues

Esta funcion nos da como retorno una tupla de los valores mensionados anteriormente, lo que nos garantiza debido a las propiedades del lenguaje, que estos valores no podran ser modificados.

2.2 Gestion de conexion

Para realizar la implementacion de la coneccion con la nube mediante MQTT, se realizo la implementacino de 4 funciones, las cuales son:

- onConnect
- onPublish
- connect
- publish

La función "connect" sera descrita a continuación:

```
def connect(mqtt_client, mqtt_username, mqtt_password,
    broker_endpoint, port):
    global connected

    if not connected:
        mqtt_client.username_pw_set(mqtt_username, password=
            mqtt_password)
        mqtt_client.on_connect = on_connect
        mqtt_client.on_publish = on_publish
        mqtt_client.connect(broker_endpoint, port=port)
        mqtt_client.loop_start()

        attempts = 0

        while not connected and attempts < 5:
```

```
14              print("[INFO] Intentando Conectar...")
15              time.sleep(1)
16              attempts += 1
17
18      if not connected:
19          print("[ERROR] No se pudo conectar con el Broker")
20          return False
21
22      return True
```

Implementacion de la funcion "connect"

En esta funcion primeramente se agregan las credenciales y informacion necesaria para la conexion con el broker de la nube y se agregan las funciones que a continuacion mostraremos las cuales se ejecutaran en caso de que se publique o que se pueda conectar. Despues de esto se espera a que la coneccion se establesca y termina.

Ahora se mostrar la implementacion de las dos funciones que se ejecutaran en caso de que se pueda conectar con el broker y la que se ejecutara en caso de se publique algun dato.

```
1  def on_connect(client, userdata, flags, rc):
2      if rc == 0:
3          print("[INFO] Conectado con el Broker")
4          global connected
5          connected = True
6      else:
7          print("[INFO] Error, conexion fallida")
```

Implementacion de la funcion "onConnect"

```
1  def on_publish(client, userdata, result):
2      print("[INFO] Publicado!")
```

Implementacion de la funcion "onPublish"

Este par de funciones como dice su nombre en caso de que se pueda llevar acabo la conexion y en caso de que pueda publicar cada uno de los mensajes.

Respecto a la Funcion que nos permitira publicar, esta seria la funcion Publish, esta se encargara de tomar el paquete de datos creado y posteriormento enviarlo por el canal anteriormente creado.

```
def publish(mqtt_client, topic, payload):
    try:
        mqtt_client.publish(topic, payload)
    except Exception as e:
        print(f"[ERROR] Tuvimos un error, detalles: \n{e}")
```

Implementacion de la funcion "Publish"

2.3 Implementacion de señales y argumentos de programa

En esta seccion explicare la implementacion del uso de argumentos en el progrma y a su vez la incorporacion de señales, para gestionar un cierre correcto de los canales y del programa. Ya que de esta forma no existe la posibilidad de bloqueos en los canales de comunicacion.

2.3.1 Argumentos

Respecto a la incorporacion de argumentos, unicamente se permitio el usuo de un parametro llamado "Vars", el cual podra contar con 3 opciones:

- All (Esta opción incluira variable de Entorno y de Movimiento)
- Ent (Esta opción inluye unicamente las variable de Entorno)
- Mov (Esta opción inluye unicamente las variable de Movimiento)

Esto se puede ver implementado en las siguientes lineas de codigo:

```
"""
Gestion de Argumentos
    -   All     ->  Se mandaran las varibles de entorno y de
        movimiento
    -   Ent     ->  Se enviaran las variables de entorno (
        Presion,Temperatura,Humedad)
    -   Mov     ->  Se enviaran las variables de movimiento
        (Magnetometro,Gyroscopio,Acceleracion)
"""
parser = argparse.ArgumentParser()
parser.add_argument("Vars",help="Designe que variables se
    enviaran All, Ent, Mov")

args = parser.parse_args() # esto fue mostrado anteriormente
```

Implementacion de Argumentos

El valor de esta variable podra ser accedida por el desarrollador como elemento del objeto anteriormente designado. Recordar que en caso de que no se reciva el parametro no se podra ejecutar el programa.

2.3.2 Señales

La implementacion de señales ofrece al usuario confianza que al terminar el programa este podra ser ejecutado de forma segura, ya que si es cerrado abruptamente din contar con la gestion de señales no se cerraran los canales, y no se podra conectar nuevamente.

Por lo cual se implemento la gestion de señales, especificamente para la señal "SIGINT" la cual es generada cuando el usuario oprime "Ctrl + C", la cual es la combinacion mas usual para finalizar un programa.

Esto fue implementado en las siguientes lineas de Codigo:

```
def GestionarSenal(signum,stack):
    global mqtt_client_G,Activo
    if mqtt_client_G != None:
        mqtt_client_G.disconnect()
        mqtt_client_G.loop_stop()
    print("\nCerrando Canal de Comunicacion\n")
    Activo = False
    sys.exit(os.EX_OK)

signal.signal(signal.SIGINT,GestionarSenal)
```

Implementacion de Señales

2.4 Implementacion de Funcion Principal

Una vez con todos los modulos anteriormente comentados, procedi a crear la funcion principal del programa, la cual es la encargada de gestionar las funciones anteriores como tambien otorgarles los valores adecuados para su correcto funcionamiento.

```
def main(mqtt_client):
    global Varibles_Labels,args,MQTT_USERNAME,MQTT_PASSWORD,
        BROKER_ENDPOINT,PORT,connected,DEVICE_LABEL,TOPIC
    values = ()

    #Designar que variables seran enviadas
    if (args.Vars == "All"):
        Varibles_Labels = ["Humedad","Presion","Temperatura"
            ,"Orientacion","Gyroscopio","Acceleracion"]
        values = GetEnviromentalValues()
        values = values + GetMoveValues_raw()
    elif(args.Vars == "Ent"):
        Varibles_Labels = ["Humedad","Presion","Temperatura"
            ]
        values = GetEnviromentalValues()
```

```
13      elif(args.Vars == "Mov"):
14          Varibles_Labels = ["Orientacion","Gyroscopio","
                Acceleracion"]
15          values = GetMoveValues_raw()
16      else:
17          print("Error en el parametro, ingrese un valor
                correcto!")
18          sys.exit(1)
19
20      #Creando Payload
21      payload = {}
22      for indice,variable in enumerate(Varibles_Labels):
23          payload[variable] = values[indice]
24
25      payload = json.dumps(payload)
26      topic = "{}{}".format(TOPIC, DEVICE_LABEL)
27
28      if not connected:
29          connect(mqtt_client, MQTT_USERNAME, MQTT_PASSWORD,
                BROKER_ENDPOINT, PORT)
30
31      # Publicando valores
32      print(f"[INFO] Intentando publicar el payload:\n{payload
            }")
33
34      publish(mqtt_client, topic, payload)
```

Funcion Main

En la funcion main, primeramente identificamos el parametro que se mando y de esta forma determino que variables son las que se desea mandar al broker, una vez identificadas se cre el mensaje que se mandara al broker tomando el nombre de la caracteristica y el valor obtenido por la sense hat.

Una vez creado el objeto JSON y el topico de publicacion se realiza la conexcion con el broker en caso de que nuca se haya realizado, en caso contrario se procede a publicar los datos obtenidos anteriormente.

```
1   if __name__ == '__main__':
2       #global mqtt_client_G
3       mqtt_client = mqttClient.Client()
4       mqtt_client_G = mqtt_client
5       while Activo:
6           main(mqtt_client)
7           time.sleep(1)
```

Funcion gestion del main

3 Pruebas de funcionamiento

Ah continuacion se mostraran Pruebas del correcto fucionamiento del programa, y de que los datos obtenidos llegan correctamente a la nube de Ubidots.

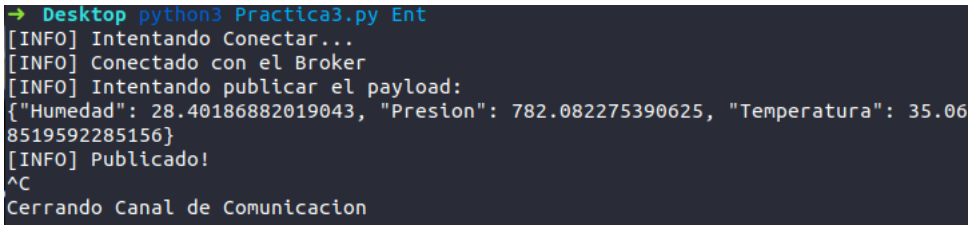

Figure 1: Funcionamiento del programa con el parametro "Ent"

Como se puede apreciar en la imagen anterior, en caso de que el parametro que se le pase al prpograma se "Ent" este publicara los valores de Presion, Humedad y Temperatura, al Broker de la nube.

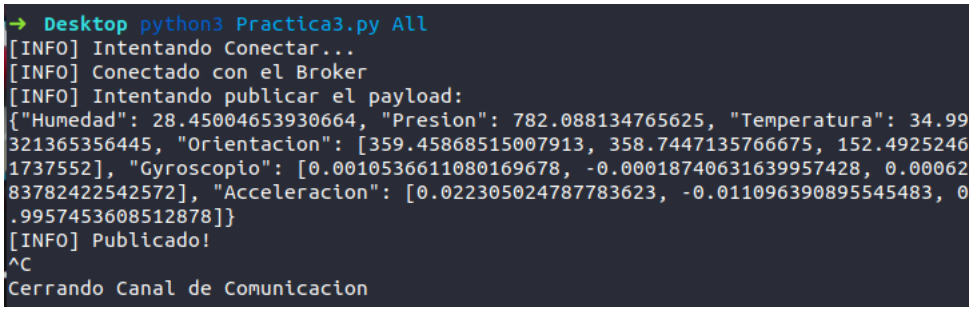

Figure 2: Funcionamiento del programa con el parametro "All"

Como se puede apreciar en la imagen anterior, en caso de que el parametro que se le pase al prpograma se "All" este publicara todos los valores censados como son Aceleracion,Giroscopio, Presion, Humedad y Temperatura, al Broker de la nube.

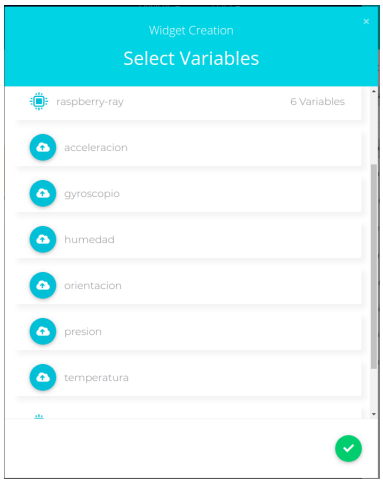

Figure 3: Modal para seleccionar el dispositivo y la variable que subcribira

En la imagen anterior se puede observar la pantalla en la que se puede seleccionar el dispositivo y el topico al que nos podremos subscribir, lo que en este caso es el dispositivo y los datos obtenidos anteririmente por el programa.

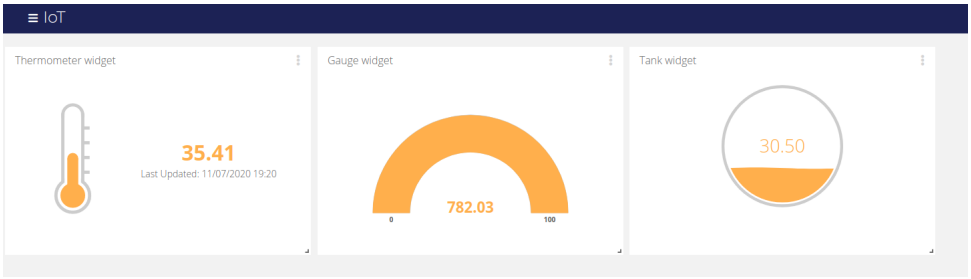

Figure 4: El DashBoard donde se vizualisan las variables de Entorno

En la imagen anterior podemos observar, la interfaz de la nube a la que se enviaron los datos, como algunos widgets, donde se muestran los datos al momento.

4 Conclusion

En esta practica se pudieron realmente apreciar el uso y comportamiento del Protocolo MQTT, ya que al realizar la gestion de la informacion, conexion y gestion, se puede crear una arquitectura mental que ofrece un entendimiento a profundidad de este, puesto que la Sense Hat cuenta con condiciones especiales para operar algunas de esus fucniones, y el modulo de paho requiere un grado

de conocimiento basico-medio, pero un entendimiento claro de los conceptos y la operacio.

Por lo que podria cerrar afirmando que esta Practica me parecio sumamente relevante, dejandome dudas como si se puede realizar envio de archvios como imagenes, videos o archivos de texto, por decir alguno.