08

Trabalhando com JSON

Transcrição

Antes mesmo de adicionarmos todo o código necessário para enviarmos os dados como JSON, vamos remover alguns trechos de código que são desnecessários para nosso projeto. Primeiro removeremos o *import* do `logging` pois ele é o responsável por imprimir todas aquelas mensagens que aparecem no console. Elas são desnecessárias pra gente. Então logo no início do arquivo removeremos a seguinte linha:

```
import logging
```

E vamos aproveitar que estamos na sessão de imports e fazer logo o import `datetime` para que possamos criar o timestamp de cada mensagem e também o JSON para podermos formatar as mensagens.

```
import datetime
import json
```

Assim teremos:

```
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
from gpiozero import CPUTemperature
import time
import datetime
import argparse
import json
```

A próxima alteração é remover a verificação de uso de *WebSockets* e toda a sessão de `Configure logging` . O trecho que removeremos está logo abaixo:

```
if args.useWebsocket and args.certificatePath and args.privateKeyPath:
    parser.error("X.509 cert authentication and WebSocket are mutual exclusive. Please pick one.")
    exit(2)

if not args.useWebsocket and (not args.certificatePath or not args.privateKeyPath):
    parser.error("Missing credentials for authentication.")
    exit(2)

# Configure logging
logger = logging.getLogger("AWSIoTPythonSDK.core")
logger.setLevel(logging.DEBUG)
streamHandler = logging.StreamHandler()
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
streamHandler.setFormatter(formatter)
logger.addHandler(streamHandler)
```

Na sessão `Init AWSIoTMQTTClient`, temos uma outra verificação por WebSockets. Não estamos utilizando essa forma de comunicação, então podemos remover o `if` inteiro e continuar apenas com o trecho do `else`. O código que estava assim:

```
myAWSIoTMQTTClient = None
if useWebsocket:
    myAWSIoTMQTTClient = AWSIoTMQTTClient(clientId, useWebsocket=True)
    myAWSIoTMQTTClient.configureEndpoint(host, 443)
    myAWSIoTMQTTClient.configureCredentials(rootCAPath)
else:
    myAWSIoTMQTTClient = AWSIoTMQTTClient(clientId)
    myAWSIoTMQTTClient.configureEndpoint(host, 8883)
    myAWSIoTMQTTClient.configureCredentials(rootCAPath, privateKeyPath, certificatePath)
```

Passa a ficar assim:

```
myAWSIoTMQTTClient = AWSIoTMQTTClient(clientId)
myAWSIoTMQTTClient.configureEndpoint(host, 8883)
myAWSIoTMQTTClient.configureCredentials(rootCAPath, privateKeyPath, certificatePath)
```

Agora podemos ir diretamente para a sessão `Publish to the same topic in a loop forever` e criar a mensagem da forma correta.

Nossa mensagem será composta pelos mesmos campos que serão armazenados no banco de dados. Serão 3 campos. O timestamp que é a data e hora da mensagem, um identificador do sensor, que no caso será seu nome e por último, a temperatura.

Para a formatação da mensagem, usaremos a função `dumps` do `json` informando os chaves entre aspas e os valores separados por dois pontos. Lembrando que no JSON temos o seguinte formato: **{ "CHAVE" : VALOR }**. Assim teremos:

```
payload=json.dumps({ "timestamp": str(datetime.datime.now()), "sensor":"rpi-sensor01", "temperatura'
```

Note que para o timestamp, precisamos fazer uso da função `str` para que ela converta o resultado da função `now` para um texto. O código final do *loop* que envia as mensagens estará assim:

```
while True:
    cpu=CPUTemperature()
    payload=json.dumps({ "timestamp": str(datetime.datime.now()), "sensor":"rpi-sensor01", "temperat
    myAWSIoTMQTTClient.publish(topic, payload, 1)
    time.sleep(2)
```

Já podemos executar o script `start.sh` novamente, salvando o arquivo `basicPubSub.py` antes, claro. E visualizar o resultado. Note que não terá mais milhares de mensagens de log sendo impressas ao executar o script. Apenas as mensagens do tópico. Assinando novamente no `Test` da plataforma, teremos as mensagens sendo impressas também, agora formatadas como JSON.

Validando JSON

Uma boa prática ao ver as mensagens sendo impressas no terminal ou mesmo na plataforma é copiar uma das saídas, de chave a chave e validar a formatação do JSON em ferramentas como o JSONLint (https://jsonlint.com). Isso ajuda a detectarmos possíveis problemas antes de tentarmos gravar determinado dado no banco de dados.

Não precisamos validar todas as saídas, mas pelo menos uma é uma boa ideia. Isso dará garantias que não esquecemos uma vírgula ou mesmo uma aspas.

Com isso finalizamos esta parte! Por enquanto ainda estamos trabalhando com a temperatura do próprio Raspberry PI, coisa que vamos modificar posteriormente, mas já podemos ver alguns resultados.