05

Ajustando código

Transcrição

Já conseguimos conectar com a plataforma da Amazon e publicar mensagens. Agora precisamos adaptar o código da publicação das mensagens para enviar dados reais de temperatura do nosso sensor.

Antes de trabalharmos com o sensor de fato, podemos primeiramente capturar a temperatura do processador do próprio Raspberry PI para termos uma noção de como vai funcionar e poder testar algo independente do sensor específico. Aliás, a temperatura do processador também é captada por um sensor.

O que vamos fazer é importar a função `CPUTemperature` da biblioteca `gpiozero`. Esta função já fornece o dado que precisamos e para testar, executamos o código no próprio console do Python. Primeiro digitamos `python` no terminal para entrar no console e logo depois disso o código a seguir.

```
from gpiozero import CPUTemperature
CPUTemperature()
```

Como resposta à execução do comando teremos algo como:

```
<gpiozero.CPUTemperature temperature=43.31>
```

Ótimo, temos a temperatura! Agora vamos alterar o código de publicação das mensagens para enviar este dado ao invés de outra informação aleatória. O arquivo que vamos editar é o `basicPubSub.py` que está dentro das pastas `aws/aws-iot-device-sdk-python/samples/basicPubSub/`. Vejamos os trechos que precisamos mudar.

Primeiro, logo no inicio do arquivo, temos várias importações e dentre elas, vamos inserir o *import* do `CPUTemperature`. O que antes estava assim:

```
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
import logging
import time
import argparse
```

> Como uma boa prática, lembre-se de duplicar o arquivo antes de alterá-lo. Caso algum problema aconteça, teremos sempre uma cópia de segurança. Uma forma rápida de fazer isso é usando o comando `cp` dentro da pasta `basicPubSub`: `cp basicPubSub.py basicPubSub.py.original`

Passa a ficar assim:

```
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
from gpiozero import CPUTemperature
import logging
import time
import argparse
```

Note que o arquivo em si é bem dividido em várias sessões. Cada sessão é descrita com um pequeno comentário. Temos o início do arquivo com um grande comentário e o os demais comentários funcionam como cabeçalhos. Eles são:

```
# Custom MQTT message callback

# Read in command-line parameters

# Configure logging

# Init AWSIoTMQTTClient

# AWSIoTMQTTClient connection configuration

# Connect and subscribe to AWS IoT

# Publish to the same topic in a loop forever
```

Usaremos esses tópicos para nos guiar, facilitando a busca por trechos específicos dentro do arquivo.

Para alterar o tópico padrão para um que faça mais sentido para o nosso projeto, alteraremos uma das linhas que estão dentro da sessão `Read in command-line parameters` . A linha que precisamos alterar é a seguinte:

```
parser.add_argument("-t", "--topic", action="store", dest="topic", default="sdk/test/Python", help='
```

Neste caso, mudaremos o valor do `default` para `telemetria/temperatura` . Já podemos salvar essas alterações antes de seguir para o próximo passo, que será alterar o *loop* de publicações do dado de temperatura.

```
parser.add_argument("-t", "--topic", action="store", dest="topic", default="telemetria/temperatura",
```

Todas as alterações de código em arquivos estão sendo feitas com o VI neste curso. Você pode utilizar outro editor que esteja habituado, mas caso queira saber mais sobre o VI ou utilizar o VIM (*semelhante ao VI com melhorias*), nós temos um curso sobre: Curso de Vim: Introdução e boas práticas a edição no terminal (https://cursos.alura.com.br/course/vim)