



UNIVERSIDADE  
**Estácio de Sá**

<b>Universidade</b>	Estácio de Sá
<b>Campus</b>	Polo de Cobilândia / Vila – Velha/ES
<b>Nome do Curso</b>	Desenvolvimento Full Stack
<b>Nome da Disciplina</b>	RPG0027- Vamos interligar as coisas com a nuvem
<b>Turma</b>	9001
<b>Semestre</b>	Segundo Semestre de 2024
<b>Integrantes do Grupo</b>	Tiago de Jesus Pereira Furtado
<b>Matrícula</b>	202306189045

**VILA VELHA  
2025**

## **Microatividade 5: Gerenciando e interagindo com o Hub IoT**

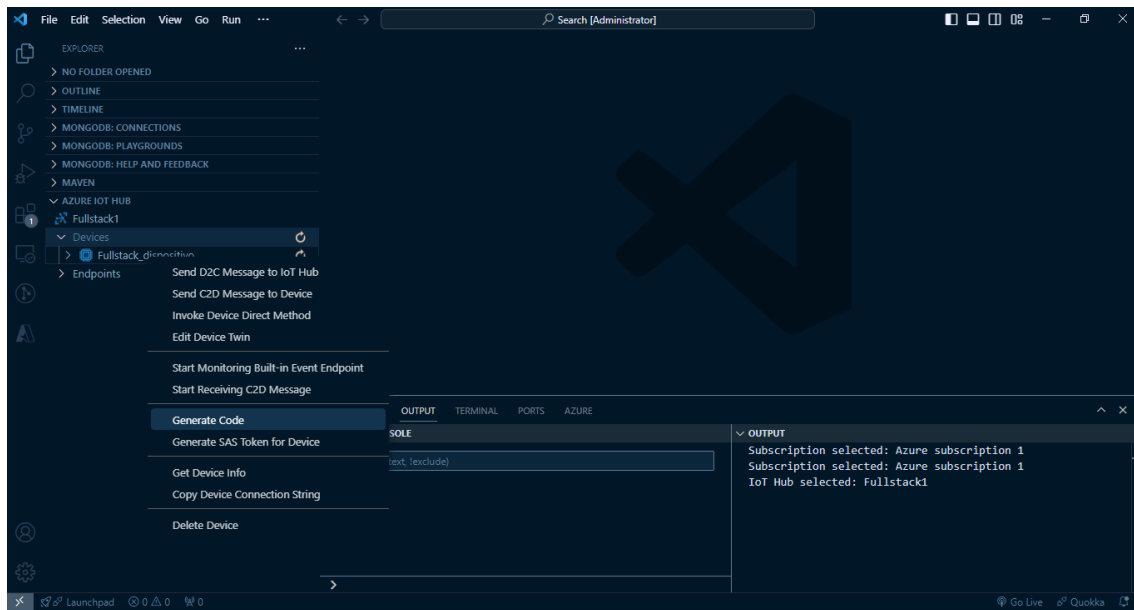
### **- Material necessário para a prática**

- Conta no Microsoft Azure.
- Navegador Web (Google Chrome, Firefox, MS Edge, Safari ou Opera).
- Visual Studio Code (VS Code).
- Raspberry Pi Azure IoT Online Simulator.

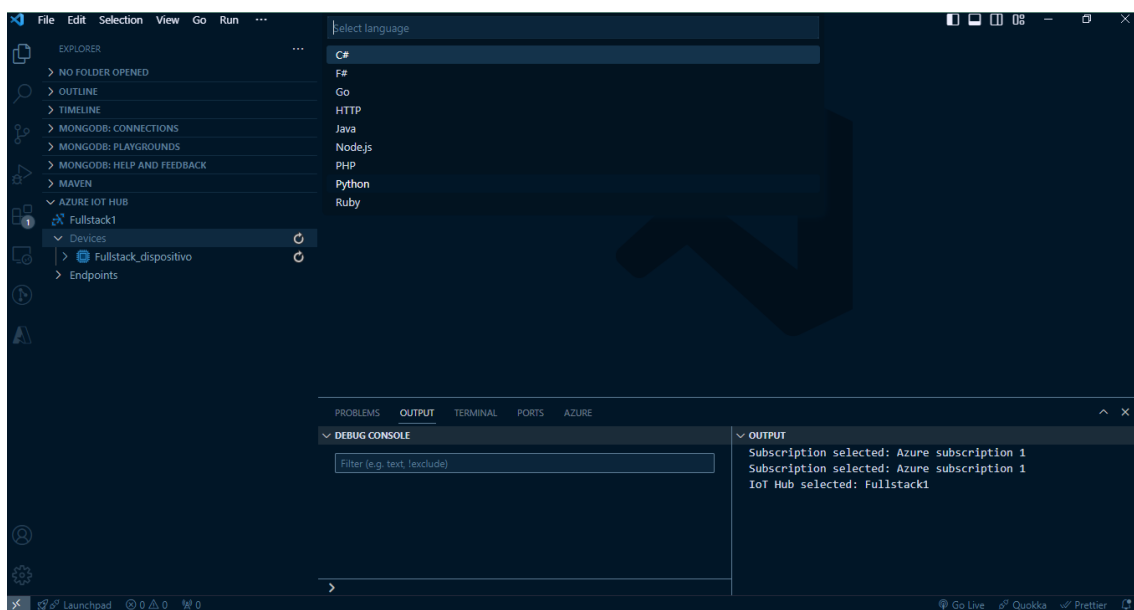
### **- Procedimentos**

Após a configuração da extensão do Hub IoT, você pode interagir com seus hubs, dispositivos e módulos na Paleta de Comandos ou no menu de ação na visualização Explorer do VS Code. Existem diversas tarefas que podem ser executadas a partir desta extensão e que serão listadas ao final da atividade. Nesta atividade nós vamos executar a atividade que gera um código para interação com o dispositivo IoT.

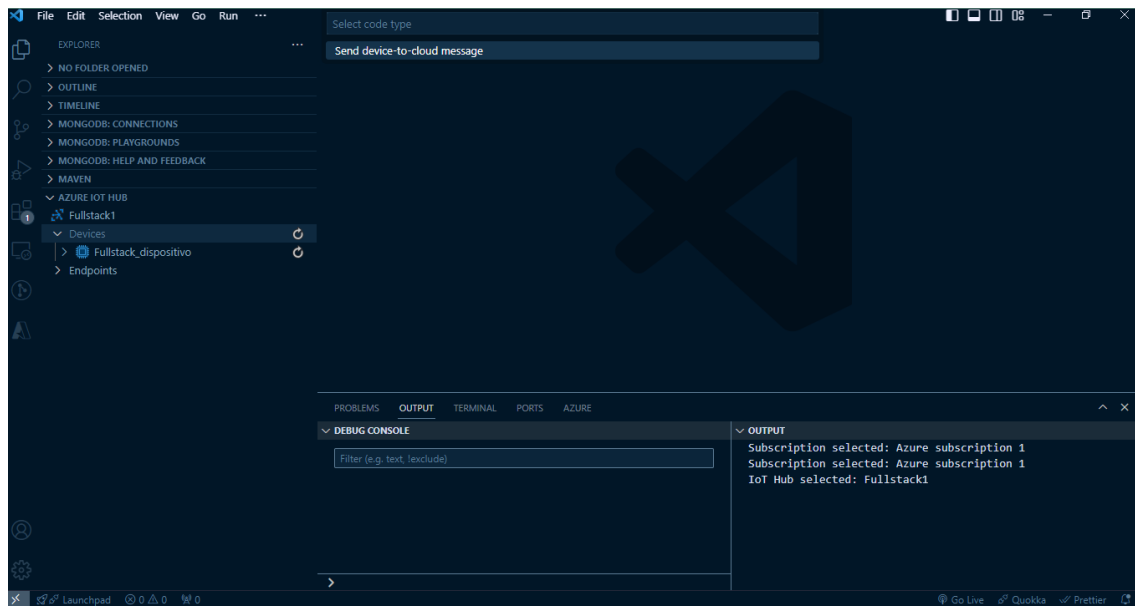
1. Abra o Visual Studio Code e no modo Explorer do VS Code, expanda a seção "Hub IoT do Azure" na barra lateral.
2. Selecione o dispositivo IoT para o qual você deseja gerar o código, clique com o botão direito e selecione a opção Generate Code (Gerar código).



3. Em seguida, selecione a linguagem que vai ser utilizada. O código será gerado na área do editor. As linguagens suportadas incluem C#, Go, HTTP, Java, Node.js, PHP, Python e Ruby. Nesta prática vamos utilizar Python, mas você pode escolher outra linguagem! Lembre-se que você precisa ter o interpretador/compilador instalado.



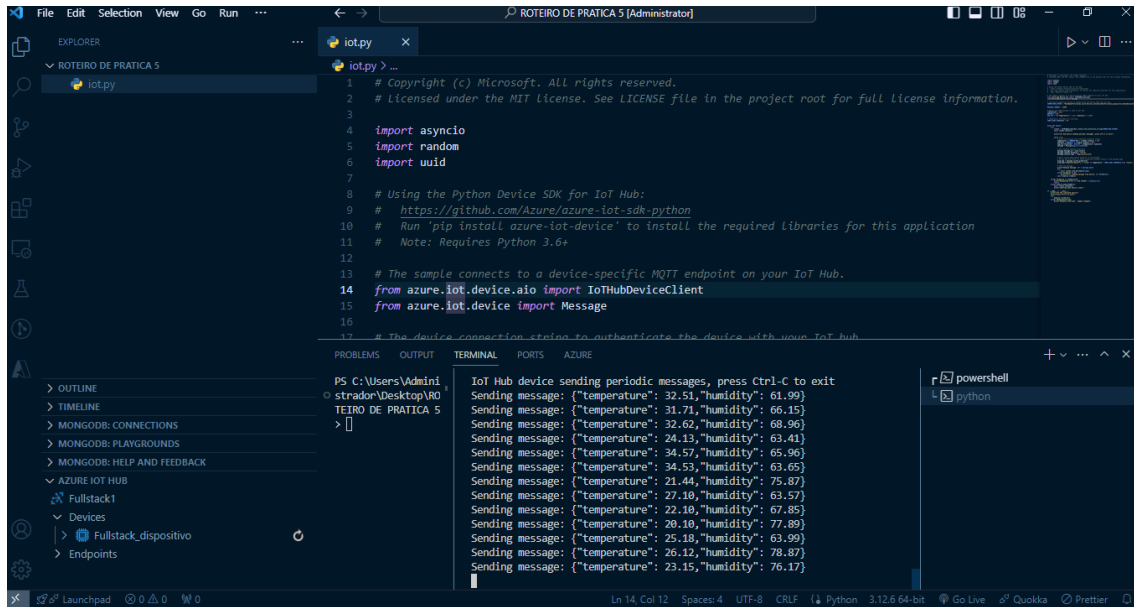
4. Dentre as opções que serão apresentadas de template de código (variam de acordo com a linguagem de programação selecionada) escolha “Send device-to-cloud message”.



5. Após selecionar a linguagem e o template de código será apresentado o código para realizar a comunicação com o dispositivo IoT selecionado e configurado no Azure IoT Hub. A figura abaixo apresenta o código que foi gerado para a linguagem Python.

```
iot.py X
C: > Users > Administrador > Desktop > missao-pratica-5.0 > pratica-05 > iot.py > ...
1  # Copyright (c) Microsoft. All rights reserved.
2  # Licensed under the MIT license. See LICENSE file in the project root for full license information.
3
4  import asyncio
5  import random
6  import uuid
7
8  # Using the Python Device SDK for IoT Hub:
9  # https://github.com/Azure/azure-iot-sdk-python
10 # Run 'pip install azure-iot-device' to install the required libraries for this application
11 # Note: Requires Python 3.6+
12
13 # The sample connects to a device-specific MQTT endpoint on your IoT Hub.
14 from azure.iot.device.aio import IoTHubDeviceClient
15 from azure.iot.device import Message
16
17 # The device connection string to authenticate the device with your IoT hub.
18 CONNECTION_STRING = "HostName=Fullstack1.azure-devices.net;deviceId=fullstack_dispositivo;SharedAccessKey=TqHKGx2N0Y1sAekkiTZPhqY9LwZAbYZMa1JWKN07t+o="
19
20 MESSAGE_TIMEOUT = 10000
21
22 # Define the JSON message to send to IoT Hub.
23 TEMPERATURE = 20.0
24 HUMIDITY = 60
25 MSG_TXT = "{\"temperature\": %.2f, \"humidity\": %.2f}"
26
27 # Temperature threshold for alerting
28 TEMP_ALERT_THRESHOLD = 30
29
30
31 async def main():
32     try:
33         client = IoTHubDeviceClient.create_from_connection_string(CONNECTION_STRING)
34         await client.connect()
35
36         print("IoT Hub device sending periodic messages, press Ctrl-C to exit")
37
38         while True:
39             # Build the message with simulated telemetry values.
40             temperature = TEMPERATURE + (random.random() * 15)
41             humidity = HUMIDITY + (random.random() * 20)
42             msg_txt_formatted = MSG_TXT % (temperature, humidity)
43             message = Message(msg_txt_formatted)
44
45             # Add standard message properties
46             message.message_id = uuid.uuid4()
47             message.content_encoding = "utf-8"
48             message.content_type = "application/json"
49
50             # Add a custom application property to the message.
51             # An IoT hub can filter on these properties without access to the message body.
52             prop_map = message.custom_properties
53             prop_map["temperatureAlert"] = ("true" if temperature > TEMP_ALERT_THRESHOLD else "false")
54
55             # Send the message.
56             print("Sending message: %s" % message.data)
57             try:
58                 await client.send_message(message)
59             except Exception as ex:
60                 print("Error sending message from device: {}".format(ex))
61                 await asyncio.sleep(1)
62
63     except Exception as iot_hub_error:
64         print("Unexpected error %s from IoTHub" % iot_hub_error)
65         return
66     except asyncio.CancelledError:
67         await client.shutdown()
68         print('Shutting down device client')
69
70 if __name__ == '__main__':
71     print("IoT Hub simulated device")
72     print("Press Ctrl-C to exit")
73     try:
74         asyncio.run(main())
75     except KeyboardInterrupt:
76         print('Keyboard Interrupt - sample stopped')
77
```

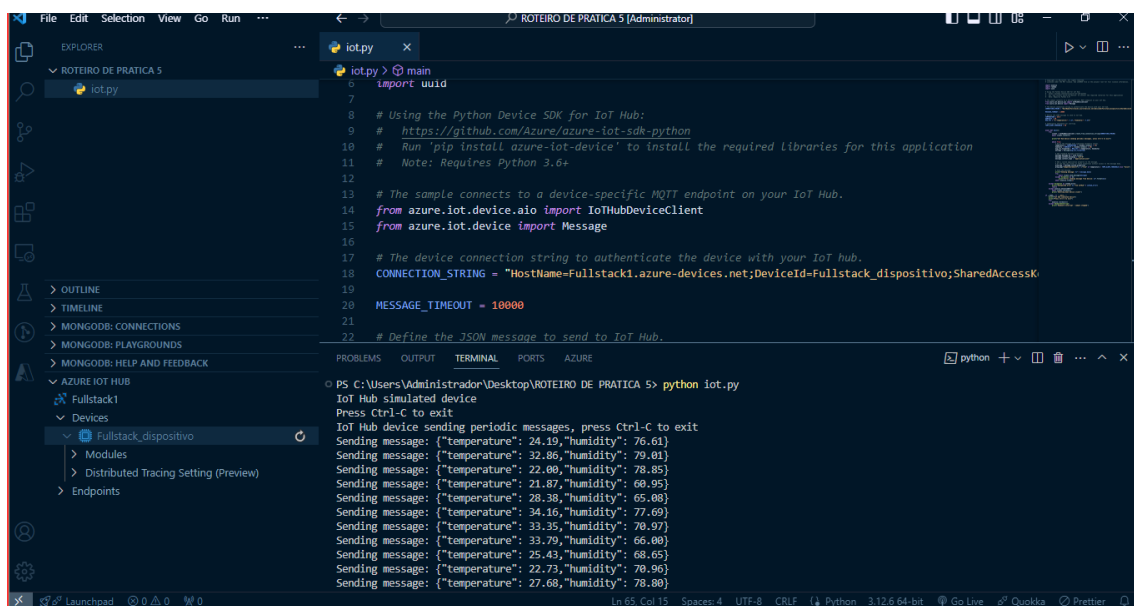
6. Execute o código e verifique na saída do console os dados coletados do dispositivo configurado no Hub IoT. Os dados apresentados são coletados do sensor de temperatura e umidade existentes no simulador.



The screenshot shows the Visual Studio Code editor with the file `iot.py` open. The file contains comments and code for connecting to an IoT Hub and sending periodic messages. The terminal output shows the execution of the script, which sends 15 messages with temperature and humidity data.

```
1 # Copyright (c) Microsoft. All rights reserved.
2 # Licensed under the MIT license. See LICENSE file in the project root for full license information.
3
4 import asyncio
5 import random
6 import uuid
7
8 # Using the Python Device SDK for IoT Hub:
9 # https://github.com/Azure/azure-iot-sdk-python
10 # Run 'pip install azure-iot-device' to install the required Libraries for this application
11 # Note: Requires Python 3.6+
12
13 # The sample connects to a device-specific MQTT endpoint on your IoT Hub.
14 from azure.iot.device.aio import IoTHubDeviceClient
15 from azure.iot.device import Message
16
17 # The device connection string to authenticate the device with your IoT hub.
```

```
PS C:\Users\Administrador\Desktop\ROTEIRO DE PRATICA 5> python iot.py
IoT Hub device sending periodic messages, press Ctrl-C to exit
Sending message: {"temperature": 32.51,"humidity": 61.99}
Sending message: {"temperature": 31.71,"humidity": 66.15}
Sending message: {"temperature": 32.62,"humidity": 68.96}
Sending message: {"temperature": 24.13,"humidity": 63.41}
Sending message: {"temperature": 34.57,"humidity": 65.96}
Sending message: {"temperature": 34.53,"humidity": 63.65}
Sending message: {"temperature": 21.44,"humidity": 75.87}
Sending message: {"temperature": 27.18,"humidity": 63.57}
Sending message: {"temperature": 22.18,"humidity": 67.85}
Sending message: {"temperature": 28.18,"humidity": 77.89}
Sending message: {"temperature": 25.18,"humidity": 63.99}
Sending message: {"temperature": 26.12,"humidity": 78.87}
Sending message: {"temperature": 23.15,"humidity": 76.17}
```



The screenshot shows the Visual Studio Code editor with the file `iot.py` open. The file contains comments and code for connecting to an IoT Hub and sending periodic messages. The terminal output shows the execution of the script, which sends 15 messages with temperature and humidity data.

```
1 # Copyright (c) Microsoft. All rights reserved.
2 # Licensed under the MIT license. See LICENSE file in the project root for full license information.
3
4 import asyncio
5 import random
6 import uuid
7
8 # Using the Python Device SDK for IoT Hub:
9 # https://github.com/Azure/azure-iot-sdk-python
10 # Run 'pip install azure-iot-device' to install the required Libraries for this application
11 # Note: Requires Python 3.6+
12
13 # The sample connects to a device-specific MQTT endpoint on your IoT Hub.
14 from azure.iot.device.aio import IoTHubDeviceClient
15 from azure.iot.device import Message
16
17 # The device connection string to authenticate the device with your IoT hub.
18 CONNECTION_STRING = "HostName=Fullstack1.azure-devices.net;DeviceId=Fullstack_dispositivo;SharedAccessKey=
19
20 MESSAGE_TIMEOUT = 10000
21
22 # Define the JSON message to send to IoT Hub.
```

```
PS C:\Users\Administrador\Desktop\ROTEIRO DE PRATICA 5> python iot.py
IoT Hub simulated device
Press Ctrl-C to exit
IoT Hub device sending periodic messages, press Ctrl-C to exit
Sending message: {"temperature": 24.19,"humidity": 76.61}
Sending message: {"temperature": 32.86,"humidity": 79.01}
Sending message: {"temperature": 22.08,"humidity": 78.85}
Sending message: {"temperature": 21.87,"humidity": 68.95}
Sending message: {"temperature": 28.38,"humidity": 65.08}
Sending message: {"temperature": 34.16,"humidity": 77.69}
Sending message: {"temperature": 33.35,"humidity": 70.97}
Sending message: {"temperature": 33.79,"humidity": 66.00}
Sending message: {"temperature": 25.43,"humidity": 68.65}
Sending message: {"temperature": 22.73,"humidity": 70.96}
Sending message: {"temperature": 27.68,"humidity": 78.80}
```

A seguir são apresentadas outras tarefas de gerenciamento que podem ser desenvolvidas a partir da extensão do VS Code.

1. Você pode executar as seguintes tarefas de gerenciamento do hub IoT a partir da extensão:
  - [Criar um hub IoT](#) e selecioná-lo como o hub IoT atual para sua extensão;
  - [Selecionar um hub IoT existente](#) como o hub IoT atual para sua extensão;
  - [Listar pontos de extremidade internos e personalizados existentes](#) para o hub IoT
  - atual selecionando o botão Atualizar para a seção Hub IoT do Azure na exibição
  - Explorer;
  - [Copiar a cadeia de conexão do hub IoT atual](#) para sua área de transferência;
  - [Gerar um token SAS para o hub IoT atual](#) e copiá-lo para sua área de transferência.
2. Você pode executar as seguintes tarefas de gerenciamento de dispositivos para o hub IoT atual a partir da extensão:
  - [Criar um dispositivo no Hub IoT](#);
  - [Criar um dispositivo no IoT Edge](#);
  - [Listar dispositivos existentes](#) selecionando o botão Atualizar para o hub IoT atual na exibição Explorer ou especificando o comando Hub IoT do Azure: Listar Dispositivos na Paleta de Comandos;
  - [Obter informações sobre o dispositivo selecionado](#) como um documento JSON, mostrado no painel de Saída do VS Code;
  - [Editar o dispositivo gêmeo](#) para o dispositivo selecionado, como um documento JSON no editor do VS Code;
  - [Copiar a cadeia de conexão do dispositivo selecionado](#) para a área de transferência;
  - [Gerar um token SAS para o dispositivo selecionado](#) e copiá-lo para sua área de transferência;

3. Você pode executar as seguintes tarefas de gerenciamento de módulo para o dispositivo selecionado no hub IoT atual:

- [Criar um módulo](#);
- [Listar módulos existentes](#) selecionando o botão Atualizar para o dispositivo atual na exibição Explorer do VS Code;
- Obter informações sobre o módulo selecionado como um documento JSON, mostrado no painel de Saída do VS Code;
- [Editar o módulo gêmeo](#) para o módulo selecionado, como um documento JSON no editor do VS Code;
- [Copiar a cadeia de conexão do módulo selecionado](#) para a área de transferência;
- [Invocar um método direto para o módulo selecionado](#) e exibir os resultados no painel de Saída do VS Code;
- Excluir o módulo selecionado do dispositivo atual.



4. Você pode executar as seguintes tarefas interativas para os recursos no hub IoT atual:

- [Gerar código](#) em uma linguagem de programação selecionada para executar uma tarefa comum, como enviar uma mensagem do dispositivo para nuvem, para o recurso selecionado;
- [Enviar uma mensagem do dispositivo para nuvem \(D2C\) para o Hub IoT](#) para o dispositivo selecionado;
- Iniciar e interromper o [monitoramento do ponto de extremidade do evento interno](#)
- [para o hub IoT atual](#) e exibir os resultados no painel de Saída do VS Code;
- [Enviar uma mensagem da nuvem para dispositivo \(C2D\) para o dispositivo](#)
- [selecionado](#) para o hub IoT atual e exibir os resultados no painel de Saída do VS Code;
- Iniciar e interromper o [monitoramento de mensagens C2D para o dispositivo](#)
- [selecionado](#) para o hub IoT atual e exibir os resultados no painel de Saída do VS Code;
- [Atualizar as configurações de rastreamento distribuído para dispositivos](#);
- Iniciar e interromper o [monitoramento dos pontos de extremidade de Hubs de](#)
- [Eventos personalizados para o hub IoT atual](#) e exibir os resultados no painel de saída do VS Code.

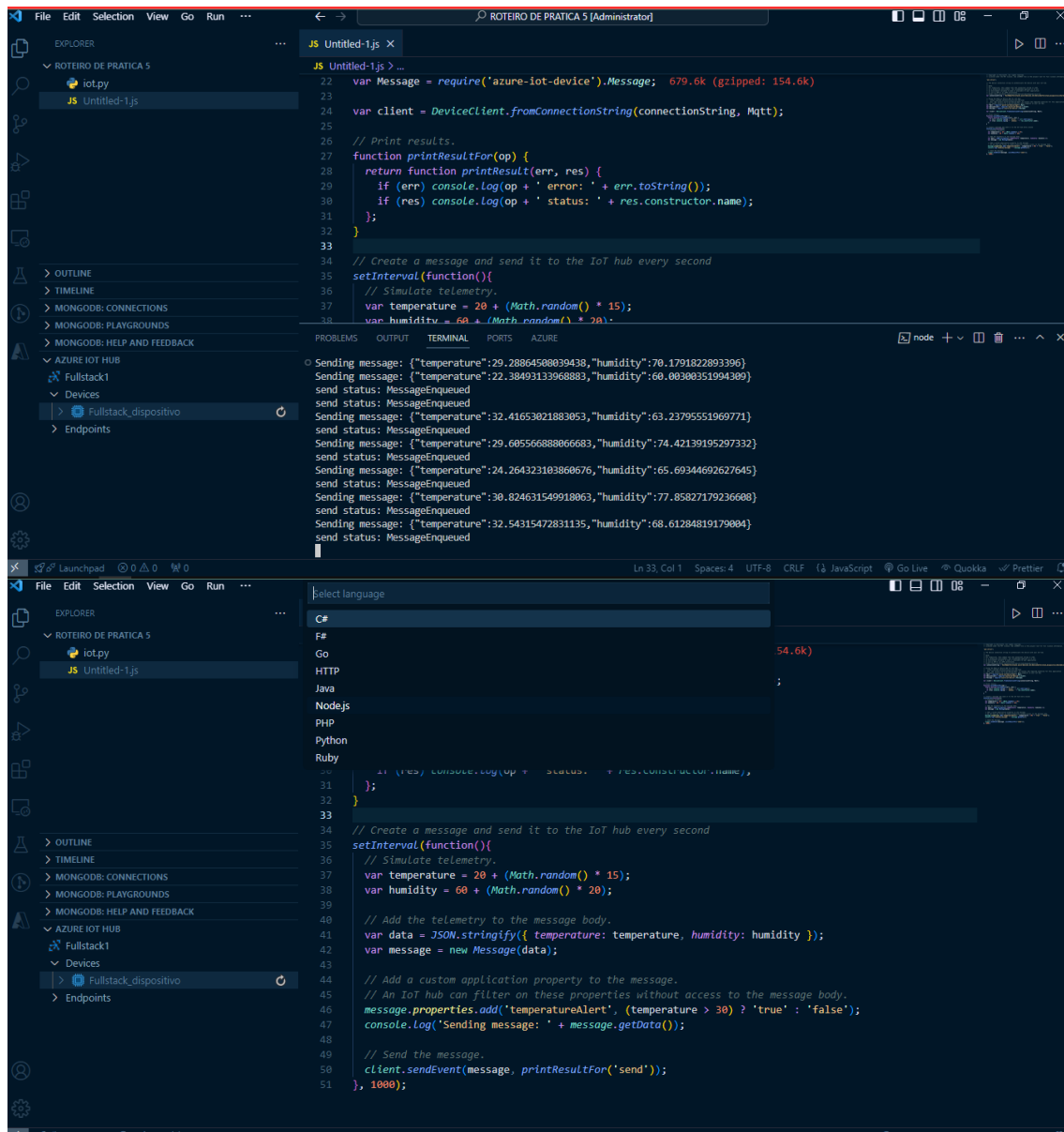
5. Você pode executar as seguintes tarefas interativas para dispositivos do [Azure IoT](#)

[Edge](#) no hub IoT atual:

- [Criar uma implantação para o dispositivo IoT Edge selecionado](#) e exibir os resultados no painel de Saída do VS Code;
- Se você tiver um manifesto de implantação apropriado, [crie uma implantação em](#)
- [escala para vários dispositivos IoT Edge](#) e exiba os resultados no painel de Saída do VS Code.

Um outro Método que foi selecionado fora o que foi proposto na atividade foi o do número 4 como mostra nas imagens abaixo:

- [Gerar código](#) em uma linguagem de programação selecionada para executar uma tarefa comum, como enviar uma mensagem do dispositivo para nuvem, para o recurso selecionado;



The first screenshot shows a VS Code editor with a JavaScript file named 'Untitled-1.js'. The code is for an IoT device that sends temperature and humidity data to an Azure IoT Hub every second. The code includes comments in Portuguese and uses the 'azure-iot-device' library. The terminal output shows the device sending messages with simulated temperature and humidity values.

```
22 var Message = require('azure-iot-device').Message;
23
24 var client = DeviceClient.fromConnectionString(connectionString, Mqtt);
25
26 // Print results.
27 function printResultFor(op) {
28   return function printResult(err, res) {
29     if (err) console.log(op + ' error: ' + err.toString());
30     if (res) console.log(op + ' status: ' + res.constructor.name);
31   };
32 }
33
34 // Create a message and send it to the IoT hub every second
35 setInterval(function(){
36   // Simulate telemetry.
37   var temperature = 20 + (Math.random() * 15);
38   var humidity = 60 + (Math.random() * 20);
39 }, 1000);
```

The second screenshot shows the same VS Code editor with a language selection menu open. The menu lists various programming languages, and 'JavaScript' is selected. The code in the background is the same as in the first screenshot, but with some additional comments in Portuguese.

```
30 // Add the telemetry to the message body.
31 var data = JSON.stringify({ temperature: temperature, humidity: humidity });
32 var message = new Message(data);
33
34 // Add a custom application property to the message.
35 // An IoT hub can filter on these properties without access to the message body.
36 message.properties.add('temperatureAlert', (temperature > 30) ? 'true' : 'false');
37 console.log('Sending message: ' + message.getData());
38
39 // Send the message.
40 client.sendEvent(message, printResultFor('send'));
41 }, 1000);
```

## - Resultados esperados 🌟

Ao finalizar está microatividade, é esperado que você reconheça as ações que podem ser desenvolvidas através da extensão do VS Code, como supervisionar, controlar e otimizar dispositivos IoT conectados pelo Azure IoT Hub.

Segue abaixo o link do vídeo gravado onde e feito essa atividade

Na qual foi proposto acima:

<https://youtu.be/7N4kN-InWNk>