U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

MASTER IN ELECTRICAL AND COMPUTERS ENGINEERING

DISTRIBUTED SYSTEMS

Publisher-Subscriber MQTT in Azure IoT for ESP32

STARTING GUIDE

Authors
Gonçalo Pereira
(up201503829)

João Santos
(up201504545)

Paula A Graça
(up201503979)

Supervisor
Prof. Luís Almeida

January 16, 2020

Contents

1 Introduction

The present document serves as a Starting Guide for the developed project in the course Distributed Systems. The implemented system consists in a MQTT publisher-subscriber communication system with Azure IoT Hub. The project is available at AzureIoT_MQTT_ESP32 Github repository and this guide describes how to set it up.

2 Setup Azure IoT

2.1 Create a Hub

In the implemented system, the IoT Hub serves as a broker, handling the exchange of messages, and providing other useful features like data analytics and message filtering. Therefore, it is necessary to create a Hub before doing anything else.

1. Sign up in Azure IoT Hub with your Microsoft account

2. Create a Hub, following the Microsoft's tutorial

2.2 Create IoT Devices

After creating a Hub, it is necessary to create endpoint devices that connect with the cloud. You need to have one device ID for every device you use simultaneously.

1. In the Hub web page, select "IoT devices" in the menu on the left and click "New"
2. Choose a device ID and leave all other options as default. Select Save
3. In the device menu there is all the relevant information about the device, such as its (primary) connection string that is going to be used later on

3 Setup ESP32

After preparing the cloud, it is time to configure the endpoint devices. In this section you will configure your computer to build, upload and monitor the ESP32. This steps are compitible with Windows, Linux and macOS.

3.1 Configure ESP IDF

1. Install a prebuilt toolchain: Windows / Linux / macOS
2. Install the ESP32 specific API/libraries by following the Espressif tutorial

3.2 Run the code

To test your setup, run the following code that publishes an "hello" message and is able to receive messages from the cloud.

1. Clone the project's repository to your PC
2. Go to the directory *esp-azure/examples/iothub_client_sample_mqtt/*
3. Run *make menuconfig* and use the arrows and enter keys to go to "Example configuration"
4. Introduce your WiFi SSID and password

```
 Status reason: IOTHUB_CLIENT_CONNECTION_OK

-> 12:58:10 SUBSCRIBE | PACKET_ID: 2 | TOPIC_NAME: devices/john/messages/de
vicebound/# | QOS: 1
-> 12:58:10 PUBLISH | IS_DUP: false | RETAIN: 0 | QOS: DELIVER_AT_LEAST_ONC
E | TOPIC_NAME: devices/john/messages/events/ | PACKET_ID: 3 | PAYLOAD_LEN:
 5
<- 12:58:10 SUBACK | PACKET_ID: 2 | RETURN_CODE: 1
<- 12:58:10 PUBACK | PACKET_ID: 3
-> 12:58:10 PUBLISH | IS_DUP: false | RETAIN: 0 | QOS: DELIVER_AT_LEAST_ONC
E | TOPIC_NAME: devices/john/messages/events/ | PACKET_ID: 4 | PAYLOAD_LEN:
 5
<- 12:58:11 PUBACK | PACKET_ID: 4
-> 12:58:11 PUBLISH | IS_DUP: false | RETAIN: 0 | QOS: DELIVER_AT_LEAST_ONC
E | TOPIC_NAME: devices/john/messages/events/ | PACKET_ID: 5 | PAYLOAD_LEN:
 5
<- 12:58:11 PUBACK | PACKET_ID: 5
-> 12:58:12 PUBLISH | IS_DUP: false | RETAIN: 0 | QOS: DELIVER_AT_LEAST_ONC
E | TOPIC_NAME: devices/john/messages/events/ | PACKET_ID: 6 | PAYLOAD_LEN:
 5
<- 12:58:12 PUBACK | PACKET_ID: 6
-> 12:58:13 PUBLISH | IS_DUP: false | RETAIN: 0 | QOS: DELIVER_AT_LEAST_ONC
```

Figure 1: ESP32 Sending messages

5. Introduce the connection string of the device you created earlier
6. Hit "Save" at the bottom and then "Exit" until you leave the menu
7. Plug the ESP32 to your computer
8. Run *make app-flash* to build and upload the code to ESP32
9. Run *make monitor* to see information about message exchange

If all went smoothly the information you see on your terminal should be similar to Figure 1. This shows the device subscribing to *devices/john/messages/devicebound/#* which means that the device *john* is listening for messages and then receives the SUBACK from the cloud. Then the device periodically publishes a message to *devices/john/messages/events* and receives the PUBACK.

4 Setup Azure Service

At this moment you have an endpoint device sending messages to the cloud but those messages are not being received by anyone else besides the cloud. In order to setup a publisher-subscriber mechanism, you have to run a service that communicates with the cloud. This service receives messages, filters them and, if those messages match a certain criteria, they are sent to other devices (subscribers).

4.1 Setup NodeJS

1. Install NodeJS on your computer
2. Open a terminal on the *pc* directory of the project's repository
3. Run *npm install*

4.2 Configure Code

1. Go to Azure IoT Hub portal and click on "Shared access policies" in the menu on the left
2. Select *iothubowner* and copy *Connection string-primary key*
3. On the *pc* directory of the project open the file *D2C2D.js* and find the line *var connectionString = '<connection-string>'*, replace it with the string you've just copied from Azure
4. Scroll down until you find the code represented in Figure 2. Here you can set which devices get which messages. The criteria for routing a message from a publisher to its subscribers can be set according to the sender or other properties of the received *message* object (e.g. the body of the message or its ID). As a first experiment you can set your device to receive messages from himself by setting *sendMessage* to the same device as the sender

```
if(sender == "john"){
  sendMessage("alice");
  //sendMessage("bob");
  //sendMessage("sarah");
  //sendMessage("sam");
  //sendMessage("james");
}
```

Figure 2: Publisher-Subscriber logic

5 Run the code

Now you have everything setup. This chapter describes how you can use the ESP32 in a publisher subscriber enviroment having Azure IoT Hub as a

broker.

1. Run *node D2C2D.js* and leave it running. This terminal will show every message exchanged between devices and the cloud.
2. Open another terminal window and go to the directory *esp-azure/examples/iothub_client_sample_mqtt/*
3. Run *make monitor* to show status messages of ESP32. In case you've setup your device to receive messages from itself, you will see the corresponding round-trip time.

6 Troubleshooting

In the case that any of the previous steps goes wrong, you can use Azure CLI to check if messages are being received by the cloud.

1. Install Azure CLI
2. Run *az login* and enter your Microsoft account credentials
3. Run *az iot hub monitor-events -n <iot-hub-name>* and see the messages that are being received by the cloud

If nothing appears on the screen check your connection string on *menuconfig* described in chapter 3.2 in topic 3.

References

[1] Espressif. *Espressif IoT Development Framework*. URL: https://docs.espressif.com/projects/esp-idf/en/v3.3.1/get-started/index.html.

[2] Gonçalo Pereira, João Santos, Paula A Graça. *Publisher-Subscriber MQTT for Azure IoT Hub*. URL: https://github.com/paulaaagraca/AzureIoT_MQTT_ESP32.

[3] Microsoft. *Azure IoT device SDK for C*. URL: https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-device-sdk-c-intro.