

Integração de Sistemas Ciber-Físicos

2021 / 2022

Trabalho 2

Integração utilizando MQTT (Esquema *Publish/Subscriber*)

Duração: 4 aulas acompanhadas por docente

Entrega: 20 de Maio de 2022

1. Introdução

Neste trabalho vai ser explorada a integração através de uma abordagem *Publish/Subscriber* apresentada nas aulas teóricas. Como ponto de partida será utilizada a arquitetura do trabalho 1 em que uma aplicação em Python, lê os valores gerados pela simulação, mas ao contrário do Lab 1, neste caso, os dados extraídos pela aplicação em Python, serão enviados para tópicos criados no broker MQTT. Neste trabalho é sugerido que na fase inicial seja utilizado o broker MQTT Mosquitto. Após a escrita desses valores nos tópicos respectivos, uma aplicação em Node Red irá receber esses mesmos valores, pois a aplicação em Node Red deverá subscrever os tópicos.

Numa fase final será feita a **integração com uma plataforma *Cloud as a Service***. Através desta plataforma, os dados da aplicação em Python deverão ser enviados **tanto para o dashboard em Node Red como para uma aplicação adicional também para visualização**. Esta última pode ser por exemplo o dashboard do primeiro trabalho laboratorial alojado no Heroku, ou uma aplicação implementada pelos estudantes numa linguagem de programação diferente de Python (e.g., Java, C#).

2. Implementação

Na implementação pedida será utilizado o seguinte material:

- MQTT Mosquitto (fase inicial); Bebotte Cloud as a Service (fase final);
- Node Red;
- Python.

3. Planeamento das Aulas

- **Aula 1** – Instalação do broker MQTT e Node Red.

- **Aula 2** – Integração entre Python e Node Red utilizando MQTT, e início do desenvolvimento da interface gráfica em Node Red.
- **Aula 3** – Integração com a plataforma *Cloud as a Service* através de MQTT e com uma aplicação adicional à escolha através da mesma plataforma.
- **Aula 4** – Funcionalidades extra. Para este ponto podem ser consideradas funcionalidades como por exemplo:
 - Autenticação no dashboard do Node-red;
 - Processamento dos dados numa das aplicações (Node-red ou externa);
 - Actualização do *current rate* da data collection em Python através do Node Red.

Outras funcionalidades que os grupos tenham interesse em explorar devem ser verificadas com o docente.

4. Arquitectura Geral

Na fase final, a arquitectura geral do projecto deverá apresentar a seguinte estrutura:

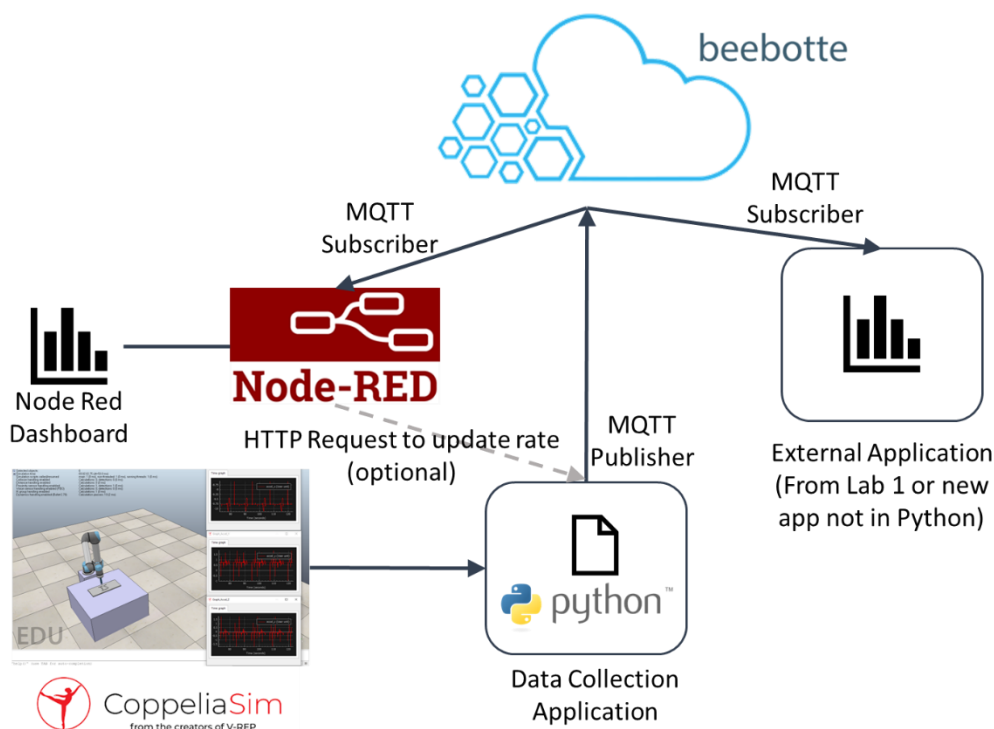


Figura 1 - Arquitectura geral do segundo trabalho laboratorial de ISCF.

5. Avaliação

A avaliação do trabalho tem a seguinte ponderação:

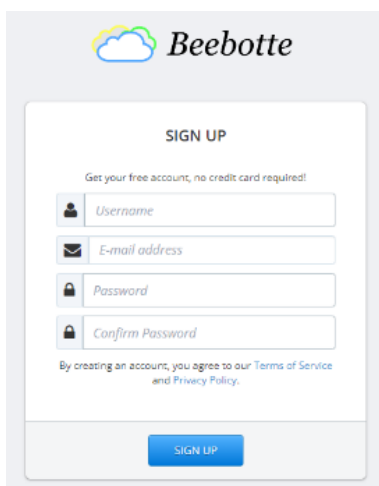
- Integração correcta entre Python e Node Red utilizando MQTT:
 - 14 valores
- Apresentação dos dados utilizando as funcionalidades do Node Red:
 - 16 valores
- Integração com a cloud e aplicação adicional:
 - 18 valores
- Funcionalidades extra:
 - 20 valores

Anexo I – Plataforma Beebotte (*Cloud as a Service*)

Para a parte final (aulas 3 e 4) deste trabalho será utilizada a plataforma *Cloud as a Service* Beebotte, focada na integração de diferentes aplicações de tempo real no contexto da *Internet of Things*. A plataforma Heroku poderia ser usada neste contexto para o mesmo propósito com o seu add-on CloudMQTT, contudo a associação de um método de pagamento é obrigatória inclusivamente nos planos *free*, pelo que se optou pela Beebotte cujo plano *free* é suficiente para os requisitos deste trabalho e suporta também a utilização de MQTT.

Os passos seguintes representam a preparação necessária na plataforma para a integração com este projecto:

- 1) Criação de conta no site <https://beebotte.com/register>:



- 2) Na tab *channels* criar um canal para o segundo trabalho laboratorial. O nome do canal é arbitrário, sendo que posteriormente deve ser criado um *resource* para cada valor que se pretende publicar. Assim, o tópico para o *publisher* dos valores *accel_x* seria **“ISCF_Exemplo/accel_x”**:

My Channels

Create and manage your Channels. Check the [tutorials](#) and the [documentation](#) to get started

Create New

Your don't have any configured channel. Create one now: [Create Channel](#)

Create a new channel

ISCF_Exemplo

Channel Description

☐ Public

Configured Resources

Resource	Resource Description	Unit	SoS	Action
accel_x	Resource Description	number	<input type="checkbox"/>	X
accel_y	Resource Description	number	<input type="checkbox"/>	X
accel_z	Resource Description	number	<input type="checkbox"/>	X

[Resource](#)

[Cancel](#) [Create channel](#)

- 3) Abrindo o novo canal é possível verificar o *Channel Token* como indicado na figura do lado esquerdo. Este token deve ser usado na autenticação do cliente para a ligação ao Bebotte. No caso do cliente MQTT em Node-Red, este token deverá ser preenchido no *Username* da tab "Security" na configuração do MQTT server (nos nodes MQTT In ou Out, tal como no tutorial). O campo *Password* pode ser deixado em branco.

ISCF_Exemplo

Channel Token: token. [redacted]

Configured resources

accel_x
accel_y
accel_z

Edit mqtt out node > Add new mqtt-broker config node

[Cancel](#) [Add](#)

Properties

Name: Name

Connection: [Security](#) [Messages](#)

Username: [redacted]

Password: [redacted]

- 4) Os restantes valores devem ser preenchidos como indicado no link <https://beebotte.com/docs/mqtt>. Nessa mesma página é fornecido um exemplo de como criar um publisher em Python recorrendo ao package Paho-MQTT utilizado também no tutorial.

Docentes

Ricardo Peres ricardo.peres@uninova.pt

José Barata jab@uninova.pt