OBJECTIF DU TP : COMMUNICATION SANS-FIL (MQTT)

L'objectif du TP est de mettre en place la couche communication sans-fil, pour permettre aux robots de communiquer avec d'autres entités, et entre eux. Vous utiliserez **MQTT** (Message Queuing Telemetry Transport), qui est un protocole de messagerie ***publish-subscribe*** basé sur le protocole TCP/IP (https://mqtt.org/).

INTRODUCTION

Votre travail va consister à préparer tous les éléments nécessaires à la communication entre les robots EV3 et une entité de contrôle qui permettra de gérer la régulation d'une intersection (Figure 1 : Exemple d'infrastructure MQTT).

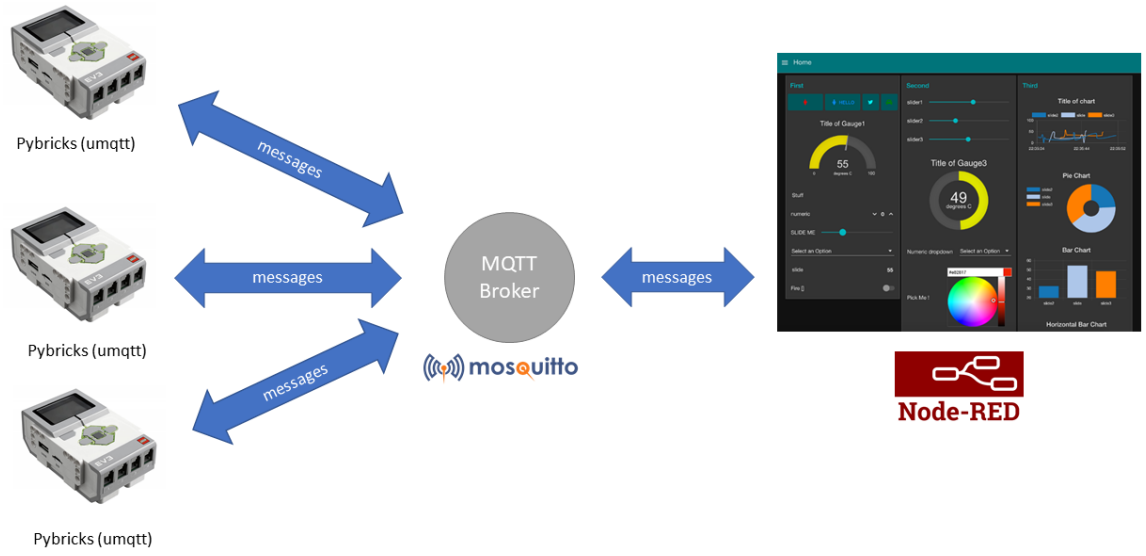

Figure 1 : Exemple d'infrastructure MQTT

MISE EN PLACE

On va différencier le **broker MQTT**, qui représente le centre d'échanges des messages, avec les **clients MQTT** qui vont échanger des messages (robots EV3 et autres entités).

BROKER MQTT

Pour commencer vous aurez besoin d'un **broker MQTT**. Vous pouvez, soit installer votre propre broker. Attention, le broker nécessite l'ouverture du port 1883 (8883 en cas de connexion sécurisée), il faut donc les droits nécessaires sur votre machine. Il existe plusieurs brokers, nous vous proposons d'utiliser **Mosquitto** (https://mosquitto.org/).

MQTT CLIENT : ROBOT EV3

Les robots EV3 ont besoin d'un **client MQTT**. La bibliothèque pybricks contient déjà le nécessaire à la communication MQTT, en ajoutant simplement la ligne ***from umqtt.robust import MQTTClient*** dans votre code python. Pour vous guider, vous pouvez vous inspirez de l'exemple suivant : https://ofalcao.pt/blog/series/ev3-micropython-and-iot.

MQTT CLIENT : CONTROLEUR D'INTERSECTION

Le contrôleur d'intersection a pour rôle d'observer l'état des robots, tout en attribuant des droits de passages pour gérer les conflits. Ce dernier a plusieurs objectifs :

1. Assurer la sécurité de l'intersection, en évitant les collisions.
2. Optimiser l'évacuation des véhicules, en formant des convois par exemple.
3. Donner un retour visuel de l'état de l'intersection (position des robots, droits de passage, etc.).

Grâce au protocole MQTT, il est possible de programmer le contrôleur dans presque n'importe quel langage. Cependant il serait intéressant ici d'utiliser **Node-RED** (https://nodered.org/). En effet Node-RED est un outil simple et adapté à la gestion des application basées sur les évènements et l'échange de messages, à travers son éditeur (Figure 2 : Editeur Node-RED). De plus Node-RED possède un module **dashboard** permettant de visualiser facilement les données (https://flows.nodered.org/node/node-red-dashboard)., ce qui peut être intéressant pour visualiser la position des robots, l'état des droits de passage, etc. (Figure 3 : Dashboard Node-RED).

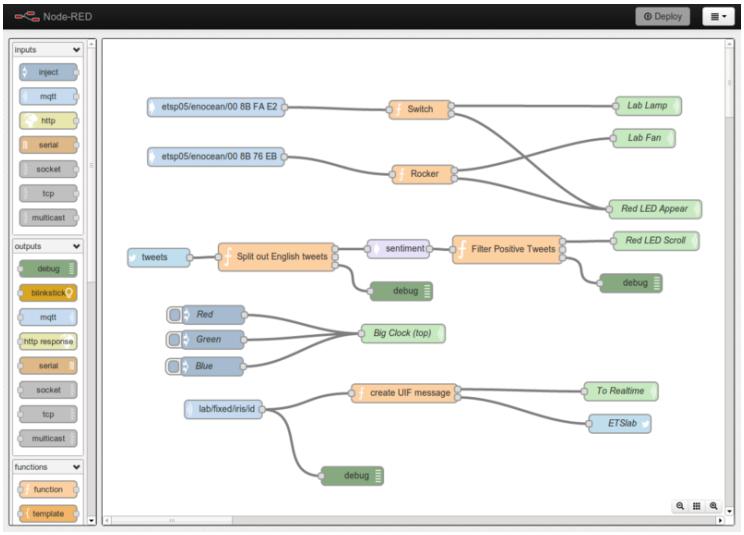

Figure 2 : Editeur Node-RED

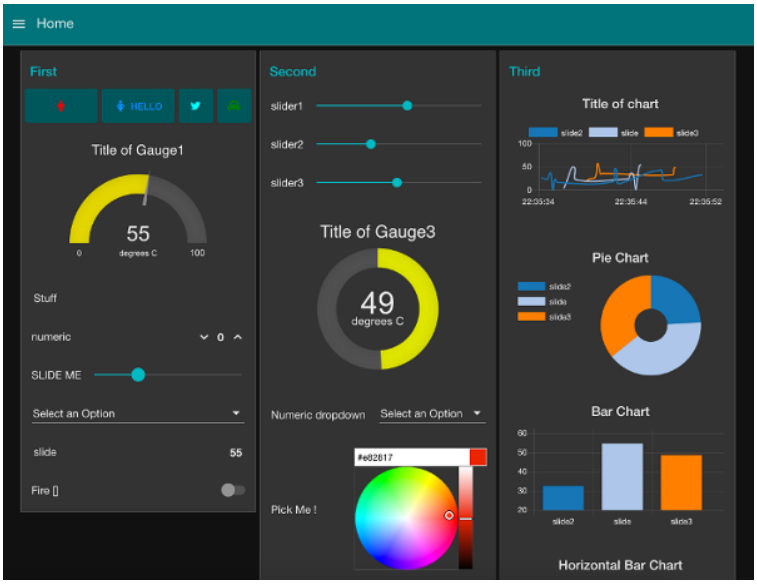

Figure 3 : Dashboard Node-RED

MQTT CLIENT : POUR TESTER

Il existe des clients permettant facilement d'envoyer et de recevoir des messages, tel que MQTT.fx (https://mqttfx.jensd.de/). Vous pouvez en utiliser un pour tester les échanges de messages et la connexion au broker MQTT. Il existe également des clients sur mobile tel que MQTT Dash sur Android.

CAS D'APPLICATION

L'application concerne la mise en place d'une intersection coopérative des robots selon l'une des trois modalités suivantes ordonnées selon le degré de complexité :

1- Feu de circulation coopératif
2- Accès coopératif
3- Peloton virtuel

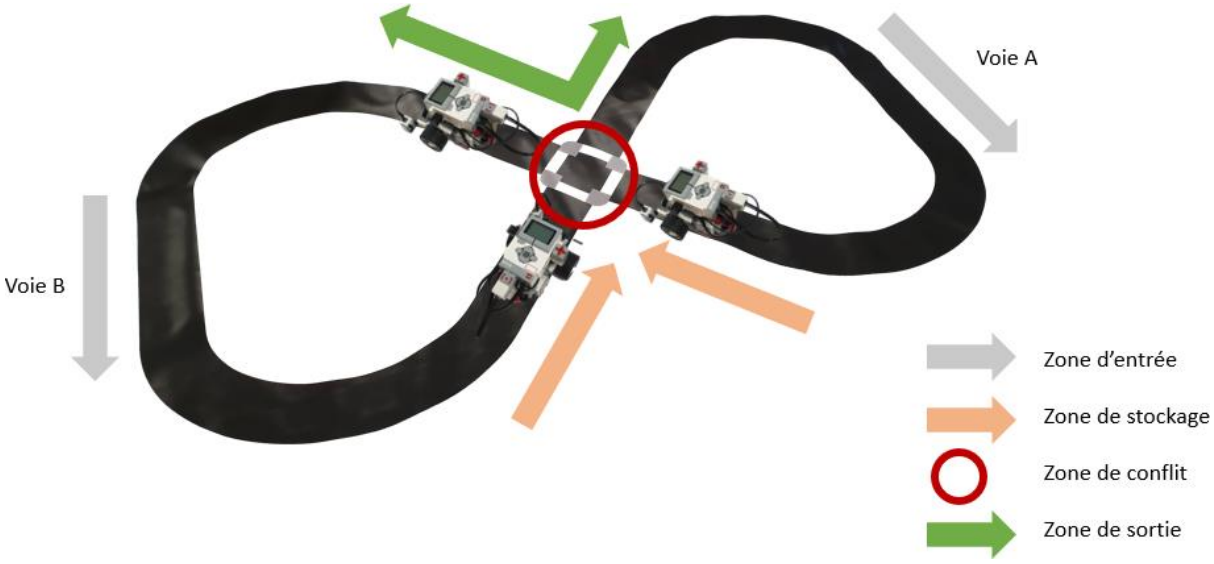

Figure 4. Intersection coopérative

Le choix de la modalité dépend de l'état d'avancement du projet.

FEU DE CIRCULATION COOPERATIF :

L'accès à la ressource est organisé selon un découpage temporel du temps d'accès à la zone de conflit. Deux temps d'accès distincts sont définis : l'un pour la voie A (phase A) et l'autre pour la voie B (phase B). Entre ces deux phases, un temps de rouge intégral est instauré, durant lequel aucune voie n'est autorisée à accéder à la zone de conflit.

Ce rouge intégral permet de laisser le temps nécessaire au robot ayant eu l'autorisation de passage de libérer complètement la zone avant que le suivant ne s'y engage. Les robots n'ayant pas le droit d'accès doivent, quant à eux, s'immobiliser avant la limite de la zone de stockage.

ACCES COOPERATIF

Les robots situés en amont du croisement déclarent leur arrivée dans la zone de stockage ainsi que leur distance par rapport à la zone de conflit. Sur cette base, le robot le plus proche de la zone de conflit obtient le droit de traversée, tandis que les autres doivent attendre leur tour en s'arrêtant avant la limite de la zone de stockage.

Une fois le passage autorisé, le robot doit signaler sa sortie afin de libérer la zone de conflit et permettre l'accès au robot suivant.

Ainsi, la zone de conflit est gérée comme une ressource à accès exclusif, garantissant qu'un seul robot puisse y évoluer à la fois.

PELOTON VIRTUEL

À l'approche de la zone de conflit, les robots forment un peloton virtuel. Autrement dit, lorsqu'ils atteignent la zone de stockage, ils maintiennent une distance de sécurité constante par rapport au robot qui les précède.

Une liste de présence est alors établie : chaque robot y enregistre sa position, exprimée par sa distance par rapport à la zone de conflit.

Lorsque le robot termine sa traversée et entre dans la zone de sortie, il signale sa libération de la zone, ce qui met à jour la liste.

ETAPES A REALISER

Votre travail pourra s'effectuer en plusieurs étapes :

1. Préparer un réseau local avec un broker MQTT (ou se connecter au réseau local mis en place en salle, à savoir le routeur TP-Link de TP).
2. Vérifier la connexion au broker, via un client de test (MQTT.fx ou autre).
3. Vérifier la connexion au broker, via un robot EV3 (python avec umqtt).
4. Vérifier la connexion au broker, via votre contrôleur (Node-RED ou autre).
5. Tester des échanges de messages entre votre robot EV3 et le client de test.
6. Tester des échanges de messages entre votre contrôleur et le client de test.
7. Tester des échanges de messages entre votre robot EV3 et le contrôleur.
8. Structurer le code du robot avec des Threads pour séparer l'envoi et la réception des messages de la boucle principale d'exécution.
9. Réfléchissez et mettez en place une structure de ***topics*** en vue d'échanger des informations entre les robots et le contrôleur (position des robots, droits de passages etc.), tout en réfléchissant au niveau de qualité de chaque message (Qos 0 ou 1).