Commande d'un robot par Websocket

6 avril 2020 : Installation des serveurs Web et Websocket, et des clients associés

Un client Websocket ne peut rester actif que pendant un temps limité. Il faut le relancer régulièrement. Pour gérer ce problème on va utiliser un serveur Web, ce qui permettra d'envoyer des requêtes qui vont déclencher de manière indirecte le lancement ou l'arrêt d'un client websocket sur le robot.

En revanche, comme un serveur Web, un serveur Websocket peut rester actif en permanence.

Au niveau de l'IHM Web , le client Websocket peut être lancé quand la page HTML est chargée.

La solution suivante est proposée :

Un serveur Web est lancé sur un VPS (fichier serveur_web/serveur.js).

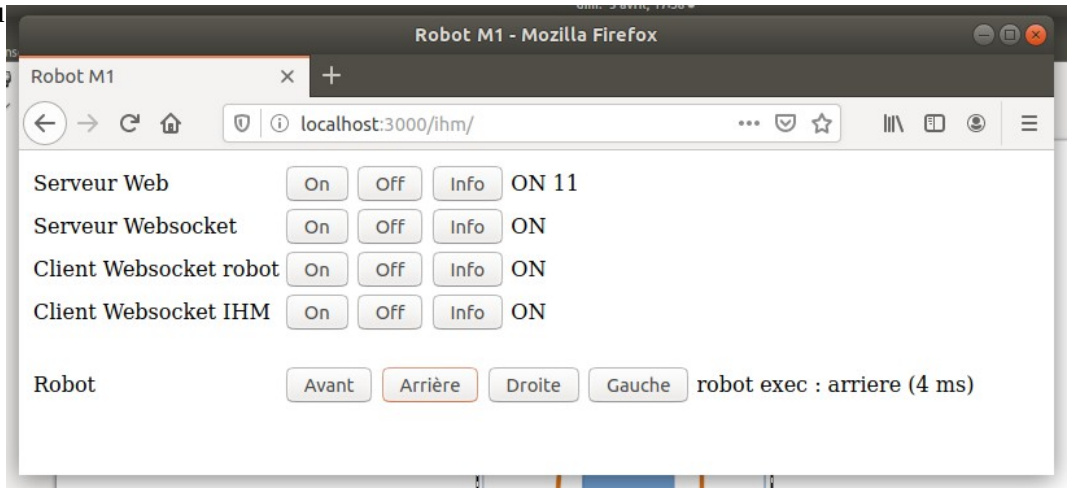
Le serveur Web permet de charger une page HTML sur un navigateur (fichier serveur_web/ihm/index.html). La page contient des boutons qui permettent les actions suivantes : lancer un serveur Websocket sur le VPS, lancer un client Websocket sur le navigateur, demander au robot de lancer un client Websocket. Quand les clients et le serveur Websocket sont lancés, d'autres boutons permettent d'envoyer des commandes au robot (avant, arrière, droite, gauche).

Un client Web est lancé sur le robot (fichier robot/robot.js). Il interroge régulièrement le serveur Web pour y chercher des commandes à exécuter : démarrer ou arrêter un client Websocket (fichier websocket/client_websocket_robot.js).

Ainsi, en veille, seuls deux processus sont actifs, un serveur Web sur le VPS et un client Web sur le robot. Quand le robot doit être démarré ou arrêté, le serveur Web du VPS démarre ou arrête un processus sur le VPS (serveur Websocket), le client Web du robot démarre ou arrête un processus sur le robot (client Websocket)

En mode utilisation du robot, on utilise uniquement le Websocket. Le robot reçoit toutes les commandes par le Websocket. Inversement, le robot transmet les données de ses capteurs par le Websocket.

Copie d'écran du navigateur. Quand tous les serveurs et tous les clients sont lancés, on peut envoyer des commandes au robot. Le temps d'exécution s'affiche en ms.



Test des programmes fournis

L'archive fournie (robot_m1.zip) contient des programmes directement exécutables (voit copie d'écran ci-dessus).

Il faut seulement initialiser un projet node (mkdir robot_m1, cd robot_m1, npm init) et faire les "npm install" (express, body-parser, axios, child_process, ws).

Il faut également télécharger jQuery (https://jquery.com). Il y a un fichier Javascript à télécharger. Il faut le copier dans js/jquery/jquery.min.js
jQuery est utilisé pour envoyer des requêtes HTTP au serveur Web, voir les "$.ajax" qui permettent d'envoyer une requête et de récupérer le résultat de manière **asynchrone** (une fonction s'exécute automatiquement quand le résultat arrive, voir "success" dans le "$.ajax").
jQuery est aussi utilisé pour afficher les résultats dans la page Web, voir les $("#…").html("…") qui permettent de modifier le contenu d'un bloc "span".

Il faut aussi modifier config/config.js qui contient 3 chemins absolus (rep_html, shell_serveur_websocket, shell_client_websocket_robot qui correspondent respectivement au dossier qui contient les fichiers HTML/JS/CSS à envoyer au navigateur, à la commande de lancement du serveur websocket, et à la commande de lancement du client websocket sur le robot)

Dans une première étape, il est conseillé de commencer par lancer les programmes en local depuis deux terminaux :

Sur Linux :

```
cd robot_m1

# lancement du serveur Web dans le terminal 1
node serveur_web/serveur_web.js

# lancement du client Web robot dans le terminal 2
node robot/robot.js

# chargement de la page dans un navigateur
http://localhost:3000/ihm

# puis clic sur Serveur Websocket ON
# puis clic sur Client Websocket Robot ON
# puis clic sur Client Websocket IHM ON

# attention pour chaque clic, la réponse arrive au bout d'environ 10s
# les temps de réponse ont été volontairement augmentés pour ne pas saturer le serveur Web
# en effet, les clients Web interrogent périodiquement le serveur Web (toutes les 10 secondes)

# ensuite, quand tous les programmes websockets ont été lancés,
# on peut envoyer des commandes au robot (avant, arrière, droite, gauche)
# les temps de réponse sont alors de quelques millisecondes
```

Test sur VPS

Créer un projet node sur le VPS. Faire les "npm install". Copier les dossiers config, message, serveur_web, shell, sebsocket.

Ensuite lancer le serveur Web.

Si un serveur Web a été lancé depuis un terminal, et si on ferme le terminal, le serveur Web reste actif. On peut ainsi le laisser actif en permanence sur le VPS. Ensuite on peut envoyer des requêtes au serveur Web pour lancer ou arrêter le serveur Websocket (processus).

Suite du projet

- **envoi de commandes à un robot réel**
- **modification de l'IHM pour pouvoir afficher en temps réel sur une carte montrant les obstacles à éviter, la position et l'orientation du robot**
- **programmation d'une mission pour le robot : avancer jusqu'à un obstacle situé au milieu d'une pièce, faire le tour de cet obstacle et revenir au point de départ. Le parcours effectué s'affiche automatiquement sur une carte quand le robot avance.**

Mission du robot : circuit tracé en orange à effectuer en mode automatique.