Raspberry : premier websocket client et serveur

Le websocket serveur

Au niveau du Raspberry, le **Websocket serveur** est créé avec le langage **Python** et nécessite une **programmation asynchrone** des échanges de messages entre le client et le serveur. Le **module asyncio** est donc **nécessaire** pour gérer un websocket en Python.

Pour installer pip3

```
$ sudo apt install python3-pip
```

Installation du module python websocket

- lancez un terminal et puis installez le module websocket en utilisant le gestionnaire de paquets Python3 pip3 :

```
$ pip3 install websockets
```

Programme python de base du serveur

Le **websocket serveur** doit être en **permanence** en attente des **connexions clientes** en précisant :

- quel **traitement** faire dès qu'il y a une connexion client en précisant la **fonction à exécuter**,
- et préciser l'**adresse IP et le port d'écoute**.

Voici l'**instruction de base** de la **création** d'un websocket serveur pour le Rapsberry :

```
lancement_serveur = websockets.serve(echange,'10.3.141.1', 5678)
```

Explications : \ * le premier paramètre est la fonction qui sera exécutée à chaque connexion cliente. Comme cette fonction ne doit pas être exécutée maintenant, on ne met pas de parenthèses. * l'adresse IP est celle du point d'accès Wifi du Raspberry ; le choix du port est libre du moment que la valeur ne soir pas celle d'un port réservé à des services réseaux existant. Ici le port 5678 est utilisé. Liste des ports réseaux : http://www.frameip.com/liste-des-ports-tcp-udp/ </WRAP> * envoyer des messages <code python> await websocket.send(“Message à envoyer.”) </code>

Le mot clé **await**, en association avec le mot clé **async** permet de ne pas bloquer le reste du programme le temps du traitement de l'instruction

* recevoir des messages <code python> messagerecu = await websocket.recv() </code> Voici un premier programme **serveur.py** avec l'utilisation d'une **boucle d'événemen**t. Dès q'une connexion est **établie**, ce serveur envoie le message **Bonjour** :
<code python> #!/usr/bin/env python3 import asyncio import websockets # Definir la fonction qui sera appelee par le serveur a la connexion d'un client async def echange(websocket,path): # recevoir un message ; messagerecu = await websocket.recv() print(messagerecu) # envoyer un message await websocket.send(“Bonjour”) lancement*serveur = websockets.serve(echange,'10.3.141.1', 5678)*
*# Creation de la boucle d'evenement (event loop) loop = asyncio.get*event*loop() loop.run*until*complete(lancement*serveur) loop.run_forever() loop.close() </code> * pour lancer le serveur ouvrez le terminal et lancez l'exéuction du programme python : <code shell> $ python3 serveur.py </code> ===== Programme javascript de base du client : la page index.html ===== Dans la page Web HTML, le script **javascript** va créer un **client WebSocket** en utilisant la bibliothèque (API) WebSocket afin de **communiquer** avec le **serveur WebSocket** du Raspberry grâce au **protocole WebSocket**. ==== Création d'un objet Websocket ==== L'instruction suivante permet d'ouvrir une connexion websocket vers le serveur : <code javascript> var websockets = new WebSocket(“ws:*10.3.141.1:5678/”); </code>*

Explications :

- la **variable websocket** va contenir la connexion vers le serveur,
- le **protocole** est **ws** suivi de l'**adresse IP et du port d'écoute** du serveur websocket.

==== Recevoir des données du serveur ==== Quand un **message arrive** du serveur, un événement (**event**) **message** est envoyé à la fonction **onmessage()**. Pour utiliser ce message voici un exemple de code : <code javascript> ws.onmessage = function (event) { afficheEtat(event.data); alert(event.data); }; </code> ==== Envoyer des données au serveur ==== Les messages sont **envoyés** avec la fonction **send()**. Cependant, les connexions étant asynchrones, l'envoi du **premier message** immédiatement après la création de la connexion **peut échouer**. Il est alors préférable <code javascript> ws.onopen = function (event) { ws.send("J'envoie un premier message au serveur."); }; </code> ==== Le code complet de ce premier exemple ==== Voici le code HTML complet de ce premier exemple. Une balise **%%%% est utilisé pour visualiser la réponse du serveur : <code html> <!DOCTYPE html> <html> <head> <meta charset="UTF-8"> </head> <body> En attente d'un message du serveur <script> *selectionner la balise avec son id var affichemessage = document.getElementById('messagerecu');* creation du websocket client vers le websocket serveur du Raspnerry var ws = new WebSocket("ws:*10.3.141.1:5678/*"); *ws.onopen = function (event) { ws.send("J'envoie un premier message au serveur."); }; ws.onmessage = function (event) {* affiche le message recu dans la balise affichemessage.innerHTML = event.data; }; </script> </body> </html> </code>**

Les messages **textuels** échangés lors d'une connexion Websocket sont au **format UTF-8**.

==== Les activités ... ====

Je reviens à la liste des activités.