PROJET PERI - Sorbonne (UPMC) 2018/2019

- AIDER Smail
- LAVALADE Pauline

Projet : Réseau de capteurs minimaliste

---

1. Configuration

1.1. Installation de serveur APACHE2

On doit d'abord installer un serveur HTTP qui va générer les pages HTML (générées en PHP).

- Se connecter sur la carte Raspberry Pi (numéro 42):
  `$ ssh -p 2242 pi@peri`
- Installation d'un serveur HTTP :
  `$ sudo apt-get install apache2`
- Lancer apache2 :
  `sudo systemctl start apache2`
- Arrêter apache2 :
  `sudo systemctl stop apache2`

1.2. Configuration de APACHE2

- Configuration des ports d'écoute :

Dans le fichier `/etc/apache2/ports.conf`, spécifier les ports à écouter.

Par défaut, il s'agit du port 80 (port par défaut pour HTTP).

Dans notre cas, il faut ajouter le port 8000.

- Création d'hôtes virtuels :

  Avec Apache, à chaque site web correspond en principe un hôte virtuel. Chaque hôte virtuel est défini par un fichier de configuration indépendant, qu'on trouve ou qu'on créé dans le répertoire `/etc/apache2/sites-available/`.

  Par défaut, on trouve le fichier `000-default.conf` et `default-ssl.conf`.

  Dans ce fichier on trouve :

  - `<VirtualHost *:8000>` : on accepte les connexions sur n'importe quelle IP du serveur (*) sur le port 8000.
  - `DocumentRoot "/var/www/projet"` : on placera les fichiers du site dans le répertoire **/var/www/projet**.
  - `ErrorLog /var/log/apache2/error.projet.log`, `CustomLog /var/log/apache2/access.projet.log combined` : il est pratique d'avoir des logs séparés pour chaque hôte virtuel afin de ne pas mélanger toutes les informations.
- Activer la configuration avec la commande `sudo a2ensite [nom du fichier sans son extension]` :

  `$ sudo a2ensite projet`
- Recharger la configuration d'Apache2 :

  `$ sudo systemctl reload apache2`
- Permissions :

Par défaut sur Ubuntu, Apache est exécuté par l'utilisateur www-data qui appartient au groupe www-data.
Pour des raisons de sécurité, il est recommandé de modifier le propriétaire des fichiers auxquels peut accéder Apache.

- Le propriétaire devrait être l'utilisateur qui va maintenir le contenu localement, mais le groupe propriétaire devrait rester www-data :
  `sudo chown -R $USER:www-data /var/www/projet`
- On change ensuite les permissions du contenu de manière à ce que l'utilisateur puisse le lire et le modifier, mais qu'Apache (dans le groupe www-data) ne puisse que le lire :
  `chmod -R a-rwx,u+rwX,g+rX /var/www/projet`

1.3. MySQL - MariaDB

On a besoin d'une base de données pour enregistrer les valeurs datées. Nous allons utiliser la base de données MariaDB.

- Installation d'un serveur MySQL :
  `$ sudo apt-get install mysql-server`
- Sécuriser le serveur MySQL :
  `$ sudo mysql_secure_installation`
- Se connecter sur le serveur MySQL :
  `$ sudo mysql -u root -p`

1.4. PHP

- Installation du PHP :
  `sudo apt-get install php libapache2-mod-php`

- Tester PHP :
    - Créer un fichier dans le répertoire root du site web : `$ sudo nano /var/www/html/info.php`
    - Copier le contenu suivant dans ce fichier :

```
<?php
phpinfo();
?>
```

    - Redémarrer le serveur Apache : `$ sudo systemctl restart apache2`
    - Visiter le fichier info.php en tapant la commande dans un navigateur : `peri:8042/info.php`
      Une page affichant la version du PHP installée doit apparaître.

1.5. Bibliothèques Python

On aura besoin d'utiliser MSQL et MQTT sous Python, il faut donc installer les bibliothèques nécessaires.
La méthode la plus simple c'est d'utiliser la commande `pip install <package>` qui permet d'installer un package avec toutes les dépendances nécessaires :

- Installation de MQTT : `$ pip install paho-mqtt`
- Installation de MYSQL API : `$ pip install mysql-connector-python`
- Parfois, il est nécessaire de mettre à jour la commande pip : `$ pip install --upgrade pip`

Il peut y avoir des situations où cette commande ne fonctionne pas, comme dans la carte Raspberry Pi. Dans ce cas, il faut suivre la méthode classique :

- Téléchargement des librairies (le code source) sur `https://pypi.org/` : `https://pypi.org/project/paho-mqtt/`, `https://pypi.org/project/mysql-connector-python/#files`, et autres si besoin.
- Installation :
  `$ cd paho/mqtt.python`
  `$ python setup.py install`

---

2. Sur l'ESP32

Il faut créer des tâches qui lisent périodiquement les capteurs et les publient vers la Raspberry Pi via un réseau sans fils.
– D'abord, il faut se connecter sur un réseau WIFI avec un login `dd-wrt` et un mot de passe `peri2019`.
– Ensuite, il faut créer un client MQTT qui va s'inscrire (subscribe) sur un serveur MQTT installé sur une carte Raspberry Pi. Pour cela, on a besoin de connaître l'adresse IP `192.168.1.42` de la carte et un numéro de port (par défaut: `1883`).
– Une fois le lien établi, on va pouvoir publier des messages (publish) sur un OUT_TOPIC `out_peri_al` que le serveur MQTT a déclaré. Pour cela, il faut définir un certain format pour faciliter le processus de décodage du message afin de stocker les mesures dans une base de données. Le message est constitué de :

- Le nom de la table dans laquelle on va stocker l'information
- La colonne concernée dans la table
- La valeur à stocker

Ces éléments seront séparés par des virgules. Le message doit respecter le format suivant : `db_table_name,row_name1,row_name2,value1,value2`. On peut indiquer autant de valeurs qu'on veut.
Par exemple, pour publier les mesures prises par un capteur ULTRASONIC Distance Sensors dans une table **distance_mesure** qui est composée d'une colonne **distance**, le message ressemblera à : `distance_mesure,distance,date,100,2012-04-19 13:08:22`.

- *FICHIER SOURCE :* `esp.ino`

---

3. Sur la Carte Raspberry Pi3

3.1. Client MQTT (Serveur)

Fichier `mqtt_server.py` :
On a utilisé l'API **Paho™ MQTT Python Client** pour implémenter un serveur MQTT afin de recevoir les données publiées par les capteurs et les enregistrer dans une base de données.

- Créer une instance de client : `paho.mqtt.client.Client("MQTT_Serveur")`.
- Implémenter la fonction **on_connect()** :
  Elle est appelée au moment de l'établissement de la connexion au broker. Dans cette fonction, on s'inscrit (subscribe) sur le

TOPIC `out_peri_al`.

- Implémenter la fonction **on_messsage()** :
  Cette fonction est appelée à la réception d'un message de la part d'un client MQTT. Elle établit une connexion vers la base de données `db_connect()`, insère la donnée `db_insert_msg()` et finalement, ferme la base de données `db_close()`.
- Se connecter sur le broker MQTT `connect()` sur l'adresse IP `localhost` et le port `1883`.
- Rentrer dans une boucle infinie en attente des requêtes à traiter `loop_forever()`.

Fichier `db.py` :
Ce fichier contient les fonctions nécessaires pour manipuler la base de données. Ces fonctions sont appelées uniquement par le *Client MQTT(Serveur)*.

- `db_connect(username, password, db_name)`:
  Cette fonction établit une connexion vers la base de données MariaDB `connect()`. Ensuite, pour interagir avec la base de données, il faut instancier un objet curseur `cursor()`. Elle renvoie deux paramètres : l'objet correspondant à la connexion (db_connextion) et le curseur ainsi établi.
- `db_close(db_connection)`: fermer la base de données.
- `db_insert_msg(db_connection, cursor, data):`
  Cette fonction permet d'insérer les données (data) passées en paramètres (string) dans la base de données. Elle décode le message respectant le format `b_table_name,row_name1,value1,...`, extrait les données,

construit une requête SQL et l'exécute avec `cursor.execute(SQL)`.

3.2. Serveur HTTP

Le serveur http (Apache) permet de générer les pages HTML (en PHP). Il extrait les données de la base de données et les affiche sous forme d'un graphe dynamique. Les fichiers qui composent le code du serveur sont :

Fichier `index.php` :

Ce fichier propose à l'utilisateur le choix de la mesure à afficher `<select id="mesure">` et inclut le fichier JavaScript *index.js*.

Fichier `index.js` :

L'API utilisée pour visualiser les données sous forme d'une courbe est le **CanvasJS**. D'abord, il faut déclarer un graphe avec le constructeur `Chart()`. Ensuite, il faut créer une fonction `updateChart()` qui permet de mettre à jour le graphe. Cette fonction doit être appelée périodiquement et cela avec `setInterval(function(){ updateChart() }, updateInterval);`.

La fonction *updateChart()* récupère les nouvelles valeurs à afficher grâce à la fonction `db_get_lastvalue()`. Cette dernière utilise **AJAX**, plus précisément, l'objet `XMLHttpRequest` pour échanger des données avec un serveur Web en arrière plan. Cela signifie qu'il est possible de mettre à jour des parties d'une page Web **sans recharger** toute la page.

La requête émise est de type GET dont on va indiquer le fichier PHP qui va consulter la base de données et renvoyer une nouvelle valeur :

`xmlhttp.open("GET","db_getdata.php?param="+table, true);` `xmlhttp.send();`. Le paramètre **table** permet d'indiquer au fichier PHP le type de la mesure (ex. chaleur), donc le nom de la table à consulter.

Fichier `db_getdata.php` :
Dans ce fichier, on utilise le PHP pour accéder à la base de données, faire une requête SQL pour récupérer la valeur la plus récente de la table concernée et ensuite, renvoyer le résultat sous forme d'une String avec un `echo`. La table à consulter est récupérer via la commande `$_GET('param')`.

3.3. Base de données MariaBD

Dans cette section, on va parler de la gestion de la base de données.

- Se connecter à MariaBD (sous Linux) :
  `$ mysql -u root -p -h localhost`
- Création d'une base de données, ex. ENCLOS :
  `$ CREATE DATABASE enclos;`
  `$ USE ENCLOS;`
- Création d'une table, ex. Distance :

```
$ CREATE TABLE distance (
id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
distance FLOAT NOT NULL,
date DATETIME);
```

- Ajout d'une donnée :

```
INSERT INTO distance (distance, date)
VALUES(150, "2012-04-19 13:08:22");
```

---

4. Mots de passe

- Raspberry Pi : `raspberry`
- MySQL Server : `peri2019`

---

5. Sources

- MQTT

  https://github.com/eclipse/paho.mqtt.python#installation

  https://www.eclipse.org/paho/clients/python/docs/
- BDD

  https://mariadb.com/resources/blog/how-to-connect-python-programs-to-mariadb/

  https://mariadb.com/kb/en/library/mariadb-basics/

  https://sql.sh/sgbd/mariadb
- APACHE

  https://hostadvice.com/how-to/how-to-install-apache-mysql-php-on-an-ubuntu-18-04-vps/

  https://doc.ubuntu-fr.org/apache2
- PHP

  https://www.php.net/manual/fr/book.mysqli.php

BONNE VACANCES :)