25/02/2017

Manuel du Programmeur

Projet Individuel : INFOB318-16-17-pds

Client : Bruno Dumas

François Georis
BLOC 3 INFO

Table des matières

1. Description

Le projet consiste en la création d'un serveur et de plusieurs boîtes connectées à celui-ci par Wifi.

1.1 Serveur

Le serveur est la machine qui va gérer la réception et l'affichage des données envoyées par les boîtes. Donc, c'est un ordinateur connecté à au moins un écran. Les données seront affichées par le biais de graphiques sur des page Web.

1.2 Boîte connectée

Une boîte connectée est composée d'une raspberry connectée à un InterfaceKitPhidget et aux phidgets adaptés à la boîte. Elle récupère des données et elle les envoie au serveur. Chacune possède un numéro pour l'identifier.

Il y a 3 types de boîtes connectées implémentées dans le projet :

- Boîte à bouton : elle possède de 1 à 8 boutons connectés à son InterfaceKitPhidget. Elle envoie pour chaque utilisation d'un bouton, un message au serveur pour l'en avertir.
- Boite Force : elle possède 4 capteurs de force connectés à son InterfaceKitPhidget. Elle envoie toutes les secondes la somme des valeurs enregistrées par les capteurs au serveur.
- Boite Température : elle possède 1 capteur de température force connecté à son InterfaceKitPhidget. Elle envoie la température maximale enregistrée sur la dernière seconde au serveur toutes les secondes.

2. Les outils de développements utilisés

2.1 Librairies

La version de java utilisée pour ce projet est Java 1.8.

Les différentes librairies utilisées dans ce projet sont :

- gson-2.8.0 qui est utilisée pour la création et la gestion des objets JSON. Vous pouvez la télécharger sur https://github.com/google/gson
- phidget21 qui est utilisée pour la gestion des phidgets. Vous pouvez la télécharger sur http://www.phidgets.com/docs/Language_-_Java
- sqlite-jdbc-3.15.1 qui est utilisée pour la gestion de la base de données. Vous pouvez la télécharger sur https://www.sqlite.org/
- d3.js qui est une librairie graphique utilisée pour la création des pages Web qui affichent les données graphiquement. Vous pouvez la télécharger sur https://d3js.org/

L'ensemble des librairies utilisées pour ce projet se trouve dans le dossier lib (\INFOB318-16-17-pds-AMELIORER\project\lib).

Il faudra installer les drivers des phidgets pour pouvoir les utiliser. Ils sont sur http://www.phidgets.com/docs/Operating_System_Support

2.2 Base de données

Il y a une seule base de données qui se trouve sur le serveur. C'est lui qui l'a créée si elle n'existait pas encore et elle portera le nom donné dans le fichier de Config (à bd). On utilise Sqlite pour gérer la base de données.

Le serveur crée une seule table par boîte qui s'est connectée à lui. Le nom des tables est défini par le mot boîte + le type de boîte + son numéro. (Exemple : BoiteTemp1).

Structure des différentes tables :

- Boîte Bouton est composée des colonnes Jour, Heure, Ind (numéro du bouton), Valeur.

```
CREATE TABLE BoiteBouton1 (
    Jour  INTEGER NOT NULL,
    Heure INTEGER NOT NULL,
    Ind   INTEGER NOT NULL,
    Valeur    INTEGER NOT NULL);
```

- Boîte Force est composée des colonnes Mois, Jour, Heure, Minute, Seconde, Valeur.

```
CREATE TABLE BoiteForce1(
    Valeur    REAL NOT NULL,
    Jour  INTEGER NOT NULL,
    Heure INTEGER NOT NULL,
    Minute    INTEGER NOT NULL,
    Seconde   INTEGER NOT NULL,
    Mois  INTEGER NOT NULL);
```

- Boîte Température est composée des colonnes Mois, Jour, Heure, Minute, Seconde, Valeur.
-
```
CREATE TABLE BoiteTemp1(
    Valeur    REAL NOT NULL,
    Jour  INTEGER NOT NULL,
    Heure INTEGER NOT NULL,
    Minute    INTEGER NOT NULL,
    Seconde   INTEGER NOT NULL,
    Mois  INTEGER NOT NULL);
```

Mois, Jour, Heure, Minute, Seconde correspondent à la date à laquelle le message a été envoyé par la boîte.

2.3 Les pages Web et Wamp

Pour afficher les données reçues par le serveur, nous utilisons des pages Web écrites avec la librairie d3.js qui se trouvent dans \INFOB318-16-17-pds-AMELIORER\project\PageWeb. Elles sont constituées principalement d'exemples trouvés sur https://github.com/d3/d3/wiki/Gallery .

Pour pouvoir les ouvrir dans notre navigateur, il faudra utiliser WAMP. Donc, il faut simplement télécharger et installer Wamp sur la machine qui hébergera le serveur. Vous pouvez le télécharger sur http://www.wampserver.com/. Puis il vous suffira de copier-coller les pages Web de \INFOB318-16-17-pds-AMELIORER\project\PageWeb dans le dossier \www\ contenu dans votre dossier Wamp créé lors de l'installation.

Il faudra changer dans le fichier de propriétés de l'application l'adresse à laquelle les pages Web se trouvent selon leur emplacement sur le serveur.

3. Serveur

3.1 ServeurPrincipal

La classe ServeurPrincipal est la MainClasse pour le Serveur. C'est elle qui va gérer la création des threads EnvoieIP et Serveur et la création d'un socket pour une connexion TCP entre le serveur et une boîte qui la demande. Elle peut actuellement créer jusqu'à 1000 connexions avec différentes boîtes. Ce nombre paraît conséquent mais lorsqu'une boîte a un problème, elle se déconnecte et se reconnecte. Elle utilise donc une nouvelle connexion et donc un nouveau thread Serveur.

3.2 EnvoieIP

La classe EnvoieIP est le thread qui va gérer les demandes d'IP des boîtes. Lorsque qu'une boîte s'allume, elle ne connaît pas l'IP du serveur ; donc elle envoie un message « Hello » en broadcast et cette classe lui envoie un message UDP contenant l'IP du serveur.

3.3 Serveur

La classe Serveur va créer un Thread serveur selon le type de la boîte connectée. On connaît le type de boîte grâce au premier message qu'elle envoie qui est son type et son numéro.

3.4 ServeurBouton, ServeurTemp, ServeurForce

Les classes ServeurBouton, ServeurTemp, ServeurForce vont créer dans la BD la table qui correspond à la boîte connectée (si elle n'existe pas encore). Elles vont aussi créer les Threads UpdateDataScreen selon le type de boîte connectée (UpdateDataScreenBouton, UpdateDataScreenTemp, UpdateDataScreenForce), Reception selon le type de boîte connectée (ReceptionBouton, ReceptionTemp, ReceptionForce) et Envoie.

3.5 UpdateDataScreenBouton, UpdateDataScreenForce, UpdateDataScreenTemp

Les classes UpdateDataScreenBouton, UpdateDataScreenForce, UpdateDataScreenTemp sont les classes qui vont gérer le lancement de la/les pages Web du type de la boîte connectée sur le navigateur

Google Chrome. Elles vont gérer aussi la création et la mise à jour des fichiers de données utilisées par les pages Web grâce aux données de la BD.

3.5.1 mapageBoutonA

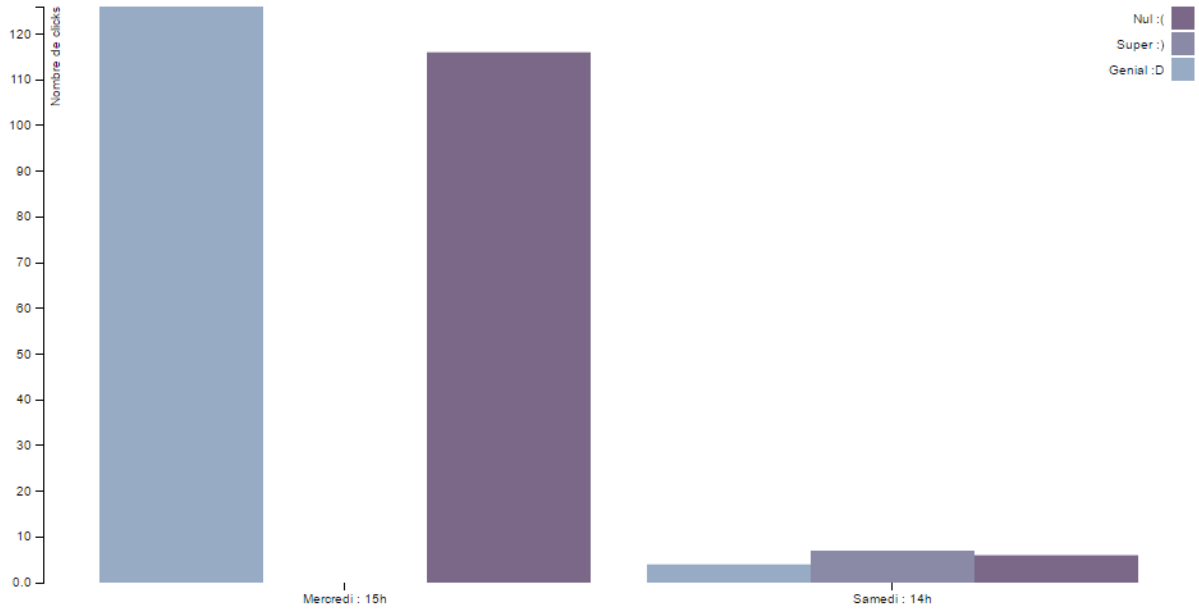
Voici le rendu visuel :

Voici un exemple du fichier de données :

```
1  bouton,click
2  Genial :D,130
3  Super :),7
4  Nul :(,122
5
```

3.5.2 mapageBoutonB

Voici le rendu visuel :



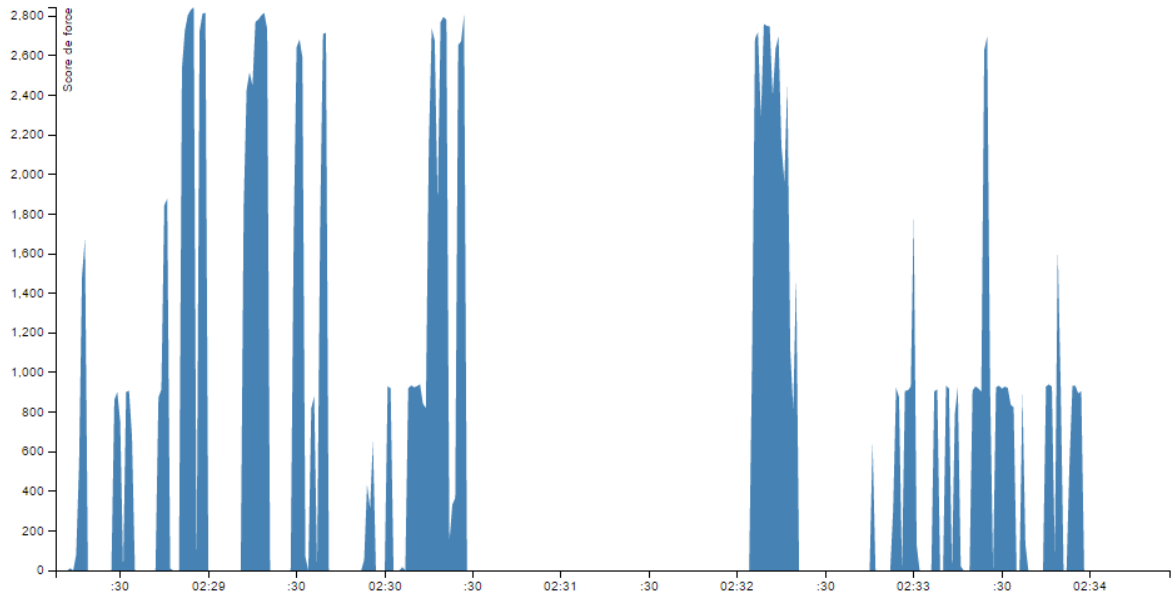
Voici un exemple du fichier de données :

```
State,Genial :D,Super :),Nul :(
Mercredi : 15h,126,0,116
Samedi : 14h,4,7,6

```

3.5.3 mapageForce

Voici le rendu visuel :

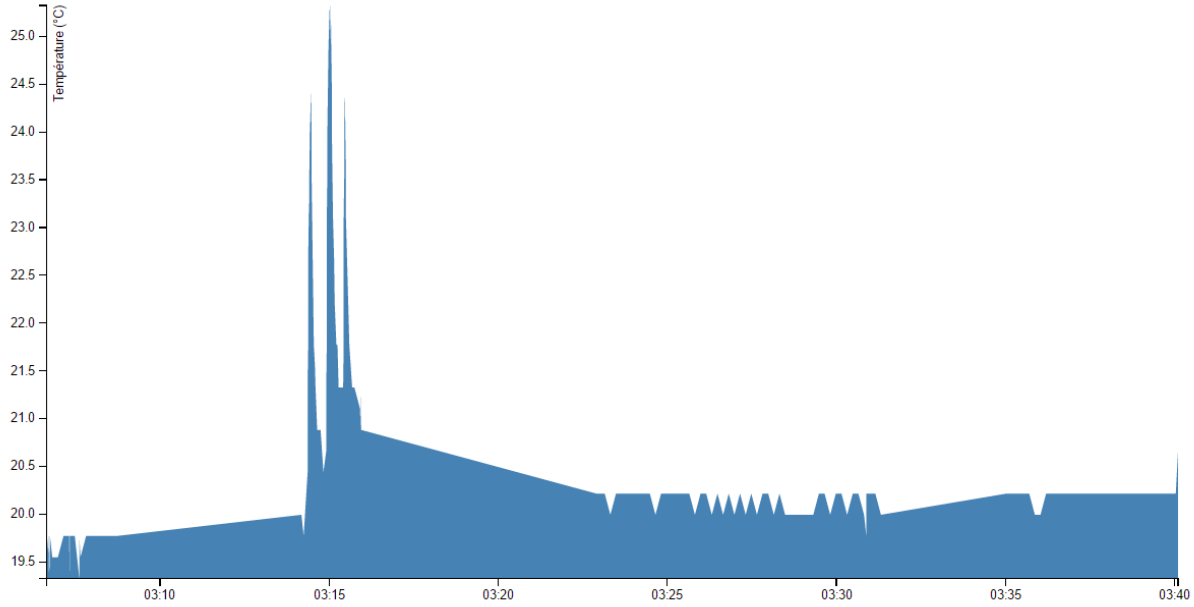


Voici un exemple du fichier de données :

```
1  date    close
2  8-28-14-56  0.0
3  9-28-14-56  0.0
4  10-28-14-56 0.0
5  11-28-14-56 0.0
6  12-28-14-56 0.0
7  13-28-14-56 11.0
8  14-28-14-56 0.0
9  15-28-14-56 79.0
10 16-28-14-56 535.0
11 17-28-14-56 1487.0
12 18-28-14-56 1678.0
```

3.5.4 mapageTemp

Voici le rendu visuel :



Voici un exemple du fichier de données :

```
1  date    close
2  50-54-15-42 19.1032
3  50-54-15-42 27.9912
4  50-54-15-42 32.6574
5  50-54-15-42 30.2132
6  50-54-15-42 25.9914
7  50-54-15-42 21.3252
8  50-54-15-42 15.548
```

3.6 ReceptionBouton, ReceptionTemp, ReceptionForce

Les classes ReceptionBouton, ReceptionTemp, ReceptionForce vont gérer la réception des données envoyées par les boîtes connectées et elles vont insérer ces données dans la BD.

Les message reçus sont sous le format Json :

- ReceptionBouton :

  {"Heure":11,"Valeur":999,"Index":3,"Jour":0}

- ReceptionTemp :

  {"Heure":11,"Minute":55,"Seconde":35,"Valeur":1500,"Jour":19,"Mois":1}

- ReceptionForce :

  {"Heure":11,"Minute":54,"Seconde":7,"Valeur":20.15,"Jour":19,"Mois":1}

3.7 Envoie

La classe Envoie va envoyer le nombre de paquets de données reçus depuis le dernier envoi à la boîte connectée.

4. Boîte connectée

4.1 Raspberry

Pour que la boîte connectée active notre application au démarrage de la raspberry, nous allons créer un fichier lancerJar.sh :

```
screen -dmS serveur sudo  java -jar /home/pi/Desktop/ClientTemp.jar
```

Le chemin d'accès du .jar dépend d'où vous l'avez mis sur votre raspberry. Ici, il est sur le Desktop.

Il faut ajouter au fichier /etc/rc.local la ligne bash /home/pi/Desktop/lancerJar.sh :

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser
# Make sure that the script will "exit 0" on success or
# value on error.
#
# In order to enable or disable this script just change
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

bash /home/pi/Desktop/lancerJar.sh
```

Voilà le fichier avec la modification. Le chemin d'accès du lancerJar.sh dépend d'où vous l'avez mis sur votre raspberry. Ici, il est sur le Desktop.

Dans notre cas, il faut placer le .jar dans le Desktop. Il faut mettre le .jar selon le type de phidget que l'on va connecter.

Exemple : le .jar pour une boiteBouton possède comme main classe ClientBouton.

4.2 ClientBouton, ClientSensorForce, ClientTemperature

Ces classes contiennent la fonction Main des différents types de boites. Elles vont récupérer l'IP du serveur, créer le Thread d'envoi des données (EnvoieBouton, EnvoieForce, EnvoieTemp) et le Thread de réception de la réponse du serveur (Reception). Elle gère les problèmes de déconnection entre la boîte et le serveur.

4.3 Envoie

Envoie est un Thread qui va initialiser la connexion avec l'interfaceKitPhidget et lui permettre d'envoyer des paquets de données au serveur quand un bouton est actif. Donc, lorsque l'on appuie sur un bouton connecté à l'interfaceKitPhidget, la boîte envoie un paquet de données au serveur avec heure, valeur, index (id du port de l'interfaceKitPhidget auquel le bouton est connecté)et le jour.

Exemple :

{"Heure":11,"Valeur":999,"Index":3,"Jour":0}

4.4 EnvoieForce

EnvoieForce est un Thread qui va initialiser la connexion avec l'interfaceKitPhidget et enverra au serveur toutes les secondes le maximum enregistré pour la somme des 4 capteurs de forces pendant la dernière seconde + le mois, jour, heure, minute, seconde. Les données sont envoyées en format JSON.

Exemple :

{"Heure":11,"Minute":55,"Seconde":35,"Valeur":1500,"Jour":19,"Mois":1}

4.5 EnvoieTemp

EnvoieTemp est un Thread qui va initialiser la connexion avec l'interfaceKitPhidget et enverra au serveur toutes les secondes le maximum enregistré par le capteur de température pendant la dernière seconde + le mois, jour, heure, minute, seconde. Les données sont envoyées en format JSON.

Exemple :

{"Heure":11,"Minute":54,"Seconde":7,"Valeur":20.15,"Jour":19,"Mois":1}

4.6 Reception

Reception est un Thread qui reçoit les réponses du serveur qui sont le nombre de paquets de données reçus par le serveur (int). Il vérifie qu'il n'y a pas trop de perte de données.

4.7 OpenNewPhidget

OpenNewPhidget est une classe qui possède la fonction initIK qui va initialiser la connexion avec l'interfaceKitPhidget. Création des Listener attach et detach du phidget et utilisation du SensorChangeListener qui est en paramètre.

4.8 LedAffichage

LedAffichage est un Thread qui va gérer l'allumage des leds. Les leds vont s'allumer selon la valeur récupérée par le capteur. L'allumage s'adapte à la selon du maximum enregistré.

4.9 DemandeIPServeur

DemandeIPServeur est le Thread qui va gérer l'envoi de demande en broadcast de l'IP du serveur jusqu'à obtenir une réponse et la réception de cette IP.