Projet PIC : Serveur et base de données

Pour le projet PIC de la ruche connectée, le choix du micro controleur se porte sur le raspberry pi. Qu'en est-il du serveur qui fera tournner le système ? Avant d'évoquer les informations trouvées, modélisons notre base de données(ceci est une première version).
Notre modèle se basera selon les différents capteurs qui équiperont la ruche, à savoir le poids et les températures intérieure et extérieure. Nous identifions aussi la ruche par un id et aussi un id du ruchier à qui il appartient.

Ruche : {
	id : Entier,
	idRuchier : Entier,
	poids : Entier,
	tInt : Entier (température intérieure),
	tExt : Entier (température extérieure)
}

Le modèle fait, intéressons nous à la base de données. Notons que notre système doit gérer un ensemble de ruches, le ruchier. Il nous sera amené à implémenter une solution comportant aussi plusieurs ruchiers. Il n'y aura pas d'intéractions entre les ruchiers, ni d'ailleurs entre les ruches. Il nous faut alors un serveur et une base de données capable d'enregistrer des données de plusieurs sources. MongoDB est un bon choix car c'est ce pourquoi il a été crée. Simple et facile à utiliser, il peut enregistrer plusieurs milliers de données sans faire ramer le serveur. Nous verrons som implémentation dans la partie serveur.

Parlons donc du serveur. Ici le choix du serveur est indépendant d'un besoin spécifique. En effet aujourd'hui tous les systèmes de serveur sont très puissants et ce n'est le plus souvent par l'aisance envers une technologie que l'on choisit tel serveur plutôt qu'un autre. Qu'en est-il de notre projet ? Le système est relativement simple. À un instant t, un ruchier déclenche les mesures de toutes ses ruches, reçoit ces données et les envoie vers la base de données. De ce principe, nous avons choisie node js. Très puissant, s'utilise beaucoup dans l'univers de l'embarqué (domotique) et très rapide. Son architecture répond à nos besoins. Aussi, il s'installe facilement sur un raspberry et forment une paire gagnante. La suite de ce document nous montre comment l'installer ainsi que l'implémentation de la base de données.

Avant de vouloir installer node js, il faut s'assurer de la compatibilité de la version du raspberry avec node js. Cette version ne doit pas être ARM6, auquel cas node ne fonctionnera pas. Il faut une version supérieure ou égale à ARM7. Il faut aussi savoir qu'il existe une version spéciale de node js pour raspberry : la v0.10.28-linux-arm-pi. Dans tous les cas, il est préférable de télécharger la dernière version stable de node js. En ligne de commande, il faut écrire les instructions suivantes :
// Mise à jour de la iste des packages
```
sudo apt-get update
```
// Mettre à jour tous les packages du système
```
sudo apt-get dist-upgrade
```
// Télécharger et installer la dernière version de node js
```
curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
```
// L'installer sur sa machine
```
sudo apt-get install -y nodejs
```
// Vérifier sa bonne installation
```
node -v
```

L’utilisation de node js se fera via un framework node très connu nommé Express.
L’implémentation minimale d’un serveur node js se présente alors comme suit :

```
// import du module « express »
var express = require(‘express ‘) ;

var app = express() ;

var port = process.env.PORT || 5000 ;

app.listen(port, () => console.log(`Server running on port ${port}`)) ;
```

Et voilà, juste ces 4 lignes nous suffisent pour démarrer un serveur node sur n’importe quelle machine.

Pour pouvoir communiquer avec la base de données mongoDB, nous utiliserons un module nommé Mongoose. Grâce à lui, la manipulation de la base de données sera plus simple.

D’autres modules seront aussi utilisés : bodyParser, jwt pour les contrôles d’accès à l’application et un module gsm.
Concernant le module gsm, deux choix s’offrent à nous : serialport et node-gsm-modem. Un choix définitif n’est pas encore fait. En attendant d’avoir le matériel, il nous faire des tests d’implémentation afin de décider lequel des deux sera pris.