Hervé Périn

Guillaume Pouilloux

Groupe DUT S4 B

Tuteur : Monsieur Ly

**PTi - Rapport de projet**

Diffusion d’informations GPS par une Raspberry Pi

Le vendredi 21 mars 2014



Département informatique de l’IUT de l’Université de Bordeaux

Dans le cadre du projet tuteuré du quatrième semestre de DUT Informatique, nous avons été confrontés au domaine des systèmes embarqués en travaillant avec une Raspberry Pi et un module GPS, connectés entre eux.

La Raspberry Pi est un ordinateur à processeur ARM de la taille d’une carte de crédit.

Il a été conçu par l’inventeur de jeux David Braden (Zoo Tycoon, …) et permet l’exécution de variantes du système d’exploitation libre GNU/Linux et des logiciels compatibles. Dans notre cas, la Raspberry Pi est équipée de Raspbian, dérivé de Debian.

L’objectif final de notre projet consiste à partager les informations du module GPS au travers d’un serveur web, accessible sur le réseau Wi-Fi émit par la Raspberry Pi.

Un smartphone peut donc se connecter en Wi-Fi sur le serveur web hébergé par la Raspberry Pi et obtenir les informations GPS directement sur son navigateur Internet.

Sommaire

x. Objectif et organisation

**Hervé**

x. Réalisation technique

x. Mise en place du réseau

**Hervé**

x. Développement du serveur web

**Guillaume**

x. Utilisation du module GPS

**Hervé (archi tech)**

**Guillaume (communication et utilisation)**

x. Problèmes rencontrés

**Guillaume**

x. le driver du dongle wi-fi

x. la paramétrisation de la communication avec le GPS

x. Evolutions possibles

x. Vers un affichage cartographique

**Guillaume**

x. SBF plutôt que NMEA ?

**Hervé**

x. pilotage complet du GPS par une interface web

**Hervé**

x. cgi ou une autre solution ?

**Guillaume**

Annexe : mode d’emploi complet (documentation technique)

**Développement du serveur web**

La Raspberry Pi est un ordinateur à processeur ARM et ne dispose pas des mêmes capacités qu'un ordinateur traditionnel. Lors de la mise en place d'un serveur web sur un tel appareil, il convient donc de se pencher sur des solutions ne consommant pas trop de ressources.

Des outils tels que Apache sont donc mis de côté dès le départ car ils gèrent le multi-threading. Un thread est créé pour chaque requête et le serveur risque vite d'être surchargé à cause du manque de ressources.

Nous nous intéressons alors à des outils comme Node.JS ou la librairie Python BaseHTTPServer qui n'utilisent pas le multi-threading pour gérer les requêtes.

Dans cette configuration, chaque requête est traitée par ordre d'arrivé de la même manière que dans une file. Ces technologies très légères permettent tout de même de très bon temps de réponse pour des sites web qui ne sont pas trop lourds, comme le notre.

Node.JS se base notamment sur le moteur Javascript V8 open source, développé par Google.

Un serveur Node.JS est donc léger et performant à la fois.

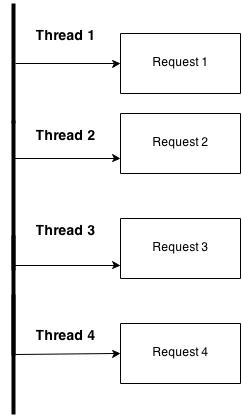


Figure 1: Multi-threading (e.g Apache, ..)

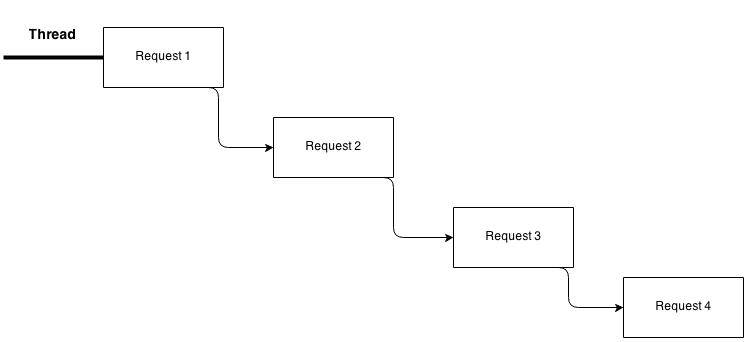


Figure 2: Single-threading (e.g Node.JS, ..)

Notre objectif est d'afficher quelques informations du GPS sur une page web. Nous décidons donc d'utiliser la librairie Python BaseHTTPServer. Cela nous permet également d'uniformiser le langage choisi entre le développement du site et la communication série avec le GPS.

Nous créons donc un répertoire *server* qui servira de racine du site web et dont l'arborescence est la suivante :

|  |
| --- |
| $ tree server/  server/  |-- cgi-bin  | |-- gps.py  |-- index.html  |-- public  | | |-- javascripts  | | | |-- images  | | | |-- layers-2x.png  | | | |-- layers.png  | | | |-- marker-icon-2x.png  | | | |-- marker-icon.png  | | | `-- marker-shadow.png  | | `-- jquery.min.js  | `-- stylesheets  | `-- style.css  `-- rpigpsserver.py  5 directories, 10 files |

Figure : Arborescence du répertoire server

Le fichier rpigpsserver.py est le fichier python qui permet l'exécution du serveur web sur le port 8088. Le contenu du fichier est présenté ci-dessous :

|  |
| --- |
| #!/usr/bin/python  import os  from BaseHTTPServer import HTTPServer  from CGIHTTPServer import CGIHTTPRequestHandler  os.chdir('/home/pi/rpi/rpi-gps/server')  handler = CGIHTTPRequestHandler  handler.cgi\_directories = ['/cgi-bin', '/htbin'] # this is the default  PORT = 8088  server = HTTPServer(('192.168.2.1', PORT), handler)  print 'serving at port : %d' % PORT  server.serve\_forever() |

Figure : Contenu du fichier rpigpsserver.py

Ce script python est exécuté au démarrage de la Raspberry Pi par CRON.

On se déplace à la racine du site web en utilisant *os.chdir* et on ajoute le support des CGI (Common Gateway Interface) car notre script de communication GPS est un script python CGI.

Enfin, le serveur est lancé à l'adresse de notre interface réseau et sur le port 8088.

Nous avons choisi le port 8088 pour ne pas casser les services tournant sur les ports 80 et 8080 de la Raspberry Pi et dont nous ne sommes pas à l'origine.

Nous avons fait le choix de rédiger un script CGI afin que le client soit à l'origine de l'exécution du script. A savoir, le client demande, à un instant t, les informations GPS qu'il peut récupérer via le script gps.py situé dans le répertoire cgi-bin.

Notre page web comprend donc un bouton "Refresh" qui permet à l'utilisateur de lancer la récupération des informations. Vous trouverez ci-dessous une capture d'écran du résultat obtenu ainsi que le contenu du fichier index.html.

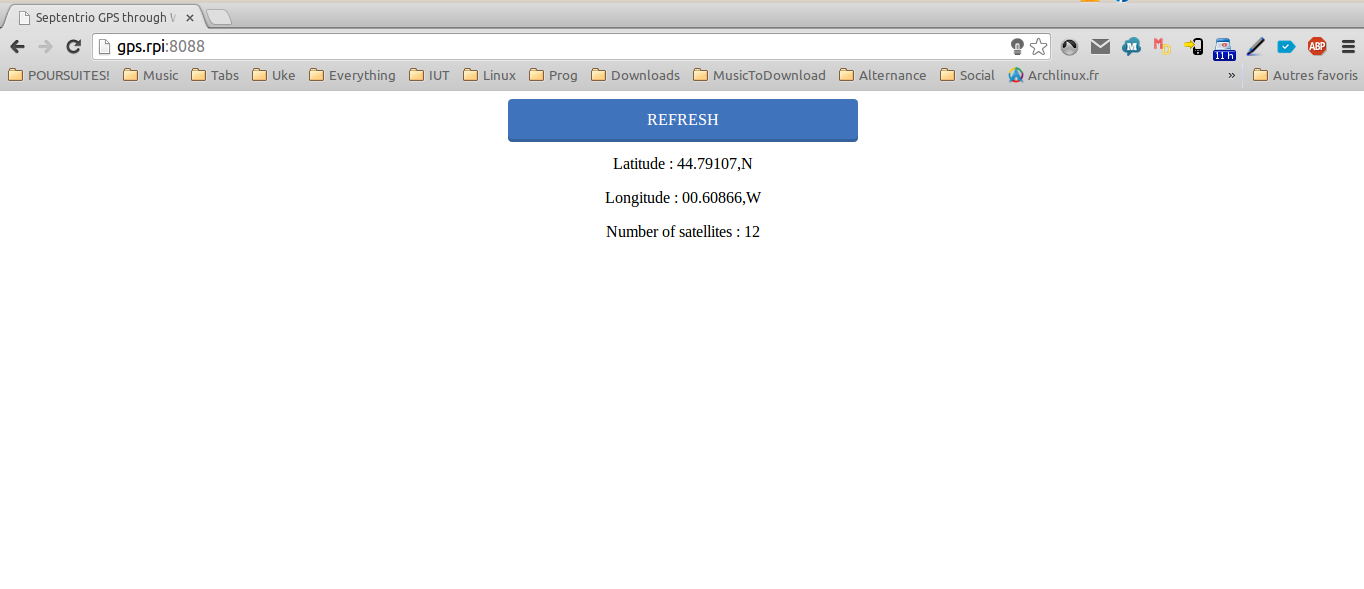


Figure : Interface web

Mise en place de la connexion à la Raspberry Pi

Choix de la technologie et des logiciels

Le smartphone devait se connecter à la Raspberry Pi en wifi. Pour cela nous disposions d’un dongle wifi.

Suite à des recherches de solutions sur Internet, nous avons décidé d’utiliser le logiciel « hostapd » pour configurer un point d’accès wifi.

Faits marquants de la réalisation

Le problème des drivers …

L’accès wifi concrètement

Au démarage de la Raspberry Pi, le point d’accès wifi est mis en place automatiquement. Pour vous y connecter :

SSID : « raspberry »

Pour le modifier : ligne ssid du fichier /etc/hostapd/config

Mot de passe : « projetgps »

Pour le modifier : ligne pwd du fichier /etc/hostapd/config

C’est un service (deamon) qui se lance au démarage de l’OS. Voilà les commandes pour le gérer :

Pour démarrer le service :

$ start hostapd

Pour éteindre le service :

$ stop hostapd

Pour redémarrer le service :

$ restart hostapd

Commandes exécutables en tant que « root ».

L’échange d’information avec le GPS

Méthode de communication avec le GPS

Le smartphone devait se connecter à la Raspberry Pi en wifi. Pour cela nous disposions d’un dongle wifi.

Suite à des recherches de solutions sur Internet, nous avons décidé d’utiliser le logiciel « hostapd » pour configurer un point d’accès wifi.

Faits marquants de la réalisation

Le problème de la réception des infos …

Le GPS concrètement

???

Le serveur d’achange d’information

Choix de la technologie et des logiciels

Quel type de logiciel ? Mettre un serveur web comme Apache est une mauvaise idée sur la Raspberry Pi à cause de ces performances. Nous avons donc recherché d’autres serveur web léger tel que Node.js, … mais finalement, c’est avec python que nous avons « créé » notre serveur web. Ce dernier peut répondre sur le port 80 à des requêtes.

Faits marquants de la réalisation

???

L’accès wifi concrètement

???