Synthèse de Projet : Tracker GPS

1/ Introduction

L'objectif de notre projet était de réaliser un **Tracker GPS** à l'aide d'un Raspberry Pi.
Le principe étant de suivre le dispositif depuis un serveur distant. Le **GPS (Grove)** est branché au Raspberry, les lignes de données sont récupérées et parsées par notre programme. Nous pouvons ensuite ainsi calculer la distance par rapport au point d'arrivée et la comparer avec la distance des données précédentes. Si la distance est supérieure, c'est à dire si le produit s'est éloigné du point d'arrivée, le Raspberry demande à un second capteur, le **Grove Buzze**r de le signaler à l'utilisateur par un bip sonore.

Ensuite ces mêmes données sont envoyées sur un serveur distant:
« http://www.wintz-combis.16mb.com/Application_Web/views/carte.php »

(Le site est hébergé sur Hostinger)
Les données comprennent, la latitude, la longitude, la date et l'utilisateur. L'utilisateur et le point d'arrivée sont spécifiques à notre programme. Les données sont envoyées grâce à un capteur Wi-Fi (car on a un Raspberry 2).
Enfin nous pouvons donc suivre en direct la progression du « colis » ou de l'utilisateur sur le site web (L'ajout d'un nouvel utilisateur se fait également sur le site web). Il suffit de charger la page et choisir le bon utilisateur pour afficher son trajet sur un carte (Google Maps) ornée d'une Polyline qui relie chaque point de passage envoyé par le Raspberry.

2/ Mode d'emploi

Scénario de déploiement du Tracker :

Imaginons que le président doive envoyer un colis confidentiel à l'étranger. Ne pouvant faire confiance à personne dans ce monde, il souhaite évidemment pouvoir surveiller le transporteur et connaître en permanence la **position exacte** de son colis (au risque de déclencher la troisième Guerre Mondiale). Comment peut-il faire ? Heureusement un de ses plus fidèles conseillers à entendu parler d'un nouveau dispositif de Tracking développé à Polytech' Montpellier par deux étudiants d'IG3.
Totalement intéressé, le président acquiert notre produit. Que lui reste-t-il à faire maintenant ?

Et bien c'est très simple, il doit simplement :

- Brancher le Raspberry à une batterie externe ou n'importe quel moyen d'alimentation (allume-cigare du transporteur par exemple).
- Connecter le Raspberry à internet via le partage de connexion.
- Se connecter sur http://www.wintz-combis.16mb.com/Application_Web/views/carte.php créer un nouvel utilisateur du nom de son transporteur.
- Connecter le gps sur le port RPISER du Raspberry
- Brancher le Buzzer sur la sortie Digitale 8
- Inscrire dans tracker.c le nom de l'utilisateur qu'il a créé ainsi que les coordonnées du point d'arrivée.

- Lancer le programme tracker.c
- Surveiller de très près le colis confidentiel sur la carte.
- Déclencher l'auto-destruction du colis si le transporteur fait fausse route (Facultatif)
- Éviter la troisième Guerre Mondiale (Facultatif)

3/ Moyens Matériels

Nous avons choisis d'utiliser un **Raspberry Pi** au lieu d'un Arduino tout d'abord car nous avions tous les deux déjà utilisés un Arduino en Peip 2 à Polytech' Nice et nous voulions comprendre le fonctionnement du Raspberry Pi. Ensuite car il était plus simple pour nous d'utiliser le Raspberry Pi pour envoyer des données sur un serveur distant.
Au niveau des capteurs, pour des raisons de budget nous n'avons utilisés que des capteurs déjà présent à Polytech. Le **Grove GPS**, le **Grove Buzzer** et le **module Wi-fi**.

4/ Moyens Humains

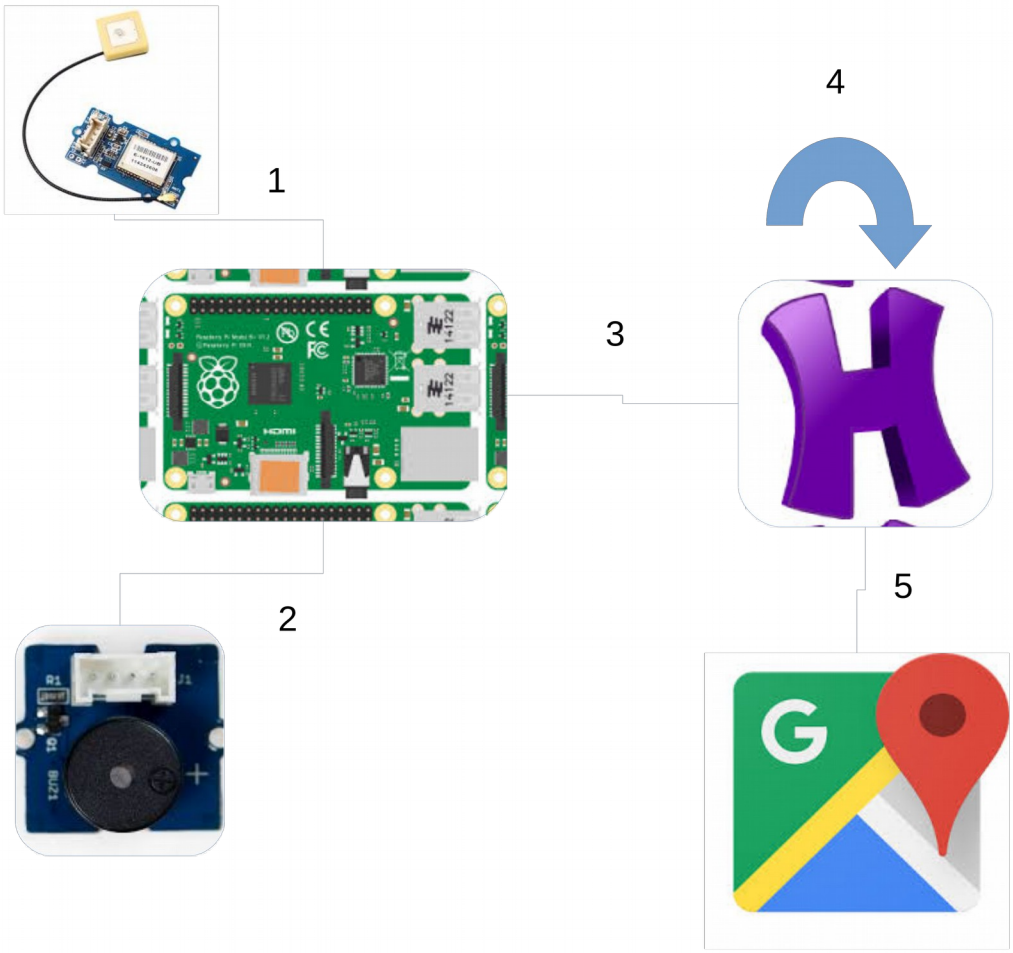
Nous avons **travaillé ensemble** la plupart du temps. D'abord dans une démarche de recherche pour comprendre le fonctionnement du GPS (Longue et infructueuse la plupart du temps). Nous avons trouvé dans un premier temps une librairie Python de *Dexter Indutries* qui utilisait le Grove GPS (Rien en C). Nous avons donc essayé de comprendre les programmes et de les tester . Les tests furent très peu concluant puisque le GPS a fonctionné une seule fois fin Décembre (grâce à monsieur Berry notamment qui nous a aidé et gentiment prêté sont Raspberry avant les vacances de Noël).

Durant les vacances, Mégane a continué de chercher des solutions pour faire fonctionner la librairie Python et trouver un moyen d'envoyer les données sur un serveur. Loïc, quant à lui, a développé une petite application web capable de recevoir et d'afficher les données envoyées par le GPS.

Enfin Janvier, dernière ligne droite, après analyse du code Python nous avons trouvé une probable solution en C pour lire les données envoyées par le GPS. Mégane s'est occupée donc d'essayer de lire dans le fichier /dev/ttyAMA0 tâche sur laquelle Loïc l'a rejoint pour essayer de trouver des solutions. Enfin nous avons fait des recherches pour trouver un moyen d'envoyer des requêtes HTTP en C, nous avons donc utilisé la commande curl de linux.

Il faut également noter la participation d'**Alexandre Bouthinon** qui nous a aidé à configurer l'ordinateur pour partager la connexion internet au Raspberry et dans les nombreuses tâches de débogage.

Architecture Matérielle :



1/ Le GPS reçoit des données des Satellites et les envoie sur au Raspberry .
2/ Le Raspberry traite les informations et le cas échéant envoie un signal au BIP
3/ Le Raspberry envoie les données sur le serveur.
4/ Le serveur stocke les données dans la base de données
5/ Le serveur envoie la carte au client.

5/ Architecture Logicielle :

Au niveau de l'architecture logicielle , on a donc un **programme principal en C** dans le Raspberry. Ce programme se charge de récupérer les données envoyées par le GPS, les traiter, envoyer les données au serveur. Le Buzzer quant à lui est activé par une fonction Python appelée depuis le programme tracker.c
Côté serveur, on donc la page carte.php, page sur laquelle on créé un nouvel utilisateur et on affiche les trajets sur une carte.

On a un controleur en php qui nous sert à recevoir les informations envoyées par le Raspberry action = « SEND_POS » , elle sert à ajouter les nouveaux utilisateurs action = « ADD » et enfin à sélectionner toutes les positions d'un utilisateur, action = « GET »
L'affichage de la carte se fait en javascript grâce à l'api javascript **Google Maps**

6/ Code

Sur le Raspberry:

Méthode qui lit dans le fichier **/dev/ttyAMA0** et renvoie un tableau contenant une latitude une longitude et une date (C).
Méthode qui calcule la distance entre le point courant et le point d'arrivée et le compare avec la distance précédente (C).
Méthode qui envoie envoie un signal au Buzzer pour qu'il sonne (Python).
Méthode qui envoie une requête POST au à la page (C) :
http://www.wintz-combis.16mb.com/Application_Web/controleur/controleur-carte.php

Côté serveur :

Méthode qui instancie et affiche une carte **Google Maps**
Méthode qui enregistre ou sélectionne des éléments dans une base de données.
Méthode qui renvoie un objet PDO
Méthode qui vérifie si un utilisateur existe.

Sources :

- Lecture dans /dev/ttyAMA0 est inspiré de la librairie de DexterIndustries : https://github.com/DexterInd/GrovePi
- Carte Google Maps : https://developers.google.com/maps/documentation/javascript/?hl=fr
- Formule de calcul de distance entre coordonnées GPS : https://fr.wikipedia.org/wiki/Orthodromie
- Objet PDO : http://php.net/manual/fr/book.pdo.php

7/ Perspectives

Quelles perspectives ? Si nous avions plus de temps nous aimerions **améliorer l'interface client** pour une utilisation simple et claire même pour des utilisateurs qui n'ont aucune connaissance. Ensuite, miniaturiser le Tracker et **le rendre autonome** pour qu'il puisse être utilisé à des fins de surveillances par exemple. Pour ceux on pourrait par exemple **remplacer le Wi-Fi par de la 3G**, une mini batterie et de cette manière il serait totalement autonome. Nous souhaiterions également utiliser seulement du C et donc réécrire la fonction Python qui déclenche le Buzzer
Cependant nous n'avons pas eu le temps de terminer correctement notre projet puisqu'il y a des bugs dans le code avec notamment la lecture dans le fichier /dev/ttyAMA0 qui nous a et nous pose encore problème. En effet, l'écriture permanente du GPS dans ce fichier bloque l'utilisation de scripts qui fonctionnent pour des fichiers statiques. Deuxième raison pour laquelle nous n'avons pas

pu terminer notre projet est du au temps que nous avons mis à faire fonctionner les premiers scripts avec le GPS (pas avant fin décembre) et ceux malgré de nombreuses tentatives. Ça nous a bloqué trop longtemps et empêcher d'avancer voir même de commencer le développement du projet. C'est pourquoi l'assemblage et le débogage ne sont pas encore terminés

Nous estimons pouvoir finir le projet en moins de moi sans personne supplémentaire. Une partie du temps afin de **déboguer la lecture dans /dev/ttyAMA0, tester notre programme sur le terrain** pour ensuite parfaire l'interface web et ajouter des fonctionnalités comme entrer le point d'arrivée. Enfin sur le Raspberry développer une mini interface qui demande les informations à l'utilisateur comme sont nom et ne nécessite plus de modifier directement le programme.

Enfin pour conclure, nous nous accordons à dire que le projet a été une **expérience très intéressante** en terme d'apprentissage, que ce soit en C même si aujourd'hui nous n'avons pas encore vraiment saisis les modalités de ce langage, le fonctionnement des capteurs et même en terme de **gestion** avec notamment l'utilisation de **Github**. Nous avons néanmoins pu prendre conscience de la puissances des micro-ordinateurs et des possibilités qu'ils nous offrent.S