UMONS
Université de Mons

Faculté Polytechnique

Un robot contrôlé via un Raspberry Pi

Projet d'informatique

Rapport de projet

Raphaël LEJEUNE
Maximilien POTTIEZ

POLYTECH MONS

Sous la direction de Monsieur le Professeur
Mohammed BENJELLOUN

2015

ACADÉMIE UNIVERSITAIRE WALLONIE-BRUXELLES

Table des matières

Chapitre 1

Introduction

Décrire le but visé, l'utilité du robot.
Faire en français et en anglais !

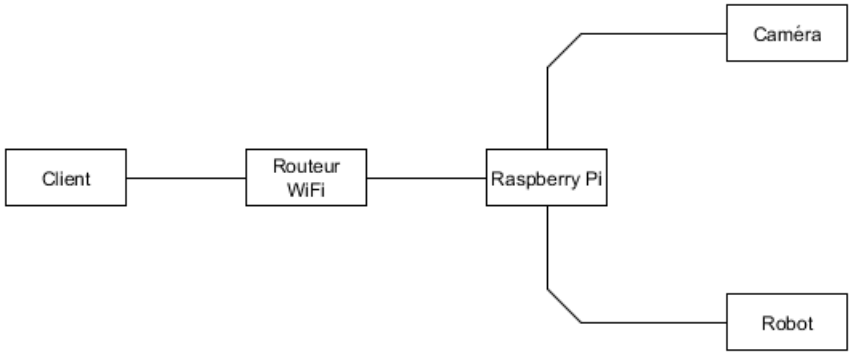

FIGURE 1.1: Matériel

Chapitre 2

Matériel

Lister le matériel utilisé, justifier sa présence

- Raspberry (pourquoi pas un Arduino, par exemple ?) - Moteurs (quel modèle ?) - Contrôleur (expliquer son utilité) - Batterie - Clé wifi - Webcam (caractéristiques techniques) - Réseau Wi-Fi opérationnel - ?

Chapitre 3

Software

Lors de la réalisation du projet, nous avons envisagé plusieurs pistes pour contrôler le robot.

3.1 Signaux PWM

Pour contrôler les moteurs, nous pouvons utiliser les signaux PWM (*Pulse Width Modulation*) : il s'agit d'ondes carrées périodiques.

Pour chaque période, on envoie une tension continue de 5 volts (qui correspond à la tension de base du Raspberry Pi, délivrée sur les ports GPIO) pendant une fraction de la période seulement. La tension « perçue » par le moteur est directement proportionnelle à cette fraction. Si cette fraction vaut par exemple 60 %, le moteur tournera comme si on envoyait 3 V, donc à 60 % de son régime maximum, si sa tension nominale est de 5 V.

Chapitre 4

Notre code

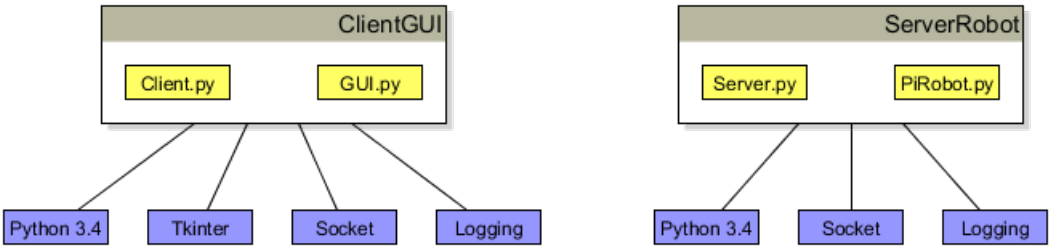

FIGURE 4.1: Logiciel

Notre code est divisé en deux parties :

- La partie client, qui tourne sur un PC (sous Windows, Linux, ...),
- La partie serveur, qui se trouve sur le Raspberry Pi.

Client.py Ce fichier sert à communiquer avec le Raspberry Pi. C'est dans ce fichier que sont récupérées toutes les requêtes de l'utilisateur, lorsqu'il appuie sur un bouton, par exemple.

GUI.py Ce fichier sert à tracer l'interface graphique. Nous avons voulu créer des fonctions dans ce fichier et les lier directement au bouton comme ci-après :

```
buttonstop = Button(frameRoot, text="STOP", command=buttonstopclick)
```

Mais cela a entraîné des erreurs de références circulaires (car *Client.py* dépend de *GUI.py* et *GUI.py* dépend de *Client.py*). Nous avons résolu ce problème en ne donnant pas de fonction au bouton. Dans *Client.py*, nous donnons explicitement la commande suivante :

```
GUI.buttonstop.bind("<Button-1>", buttonstopclick)
```

Cette seconde option présente deux avantages : on peut choisir le type d'événement à associer (bouton cliqué ou relâché, ...), et on peut aussi supprimer ce lien (avec la commande `unbind`).

Server.py C'est le programme qui tourne sur le Raspberry Pi. Son but est de recevoir les messages (depuis le client) et de les interpréter pour envoyer des instructions au robot.

PiRobot.py C'est une classe qui contient toutes les fonctions d'envoi de commandes au robot, ainsi quand *Server.py* reçoit un message, il n'a qu'à appeler la bonne fonction (par exemple la fonction `Stop`, qui envoie une commande au robot pour arrêter les moteurs).

Python 3.4 Ça peut sembler anodin, mais certaines fonctionnalités que nous utilisons n'existent pas ou existent sous un autre nom dans des versions antérieures de python. C'est pourquoi il est important de préciser que le code est exécuté avec Python 3.4.

Tkinter Nous nous servons de *Tkinter* pour tracer l'interface graphique. Il permet d'organiser assez facilement les éléments dans la fenêtre graphique.

Socket Comme son nom l'indique, *Socket* permet d'ouvrir une connection entre une machine hôte (appelée Serveur) et une ou plusieurs machines (appelées Clients). Une fois la connection établie, les commandes `send` et `recv` permettent d'échanger des données.

Logging *Logging* sert à générer un fichier .log, contenant diverses informations. A titre d'exemple, la commande suivante est appelée dans la fonction `buttonstopclick` :

```
logging.debug('Button STOP click')
```

Dans le fichier .log, on verra cette ligne :

```
2015-07-28 18:06:08,051 root  DEBUG Button STOP click
```

C'est une alternative au `print('Button STOP click')`, et qui permet de sauvegarder les actions faites lors de l'exécution d'un programme, même s'il est arrêté pour quelque raison que ce soit.

Chapitre 5

Qui a fait quoi

Environ 1 page par étudiant, détailler ce qu'il a fait

Chapitre 6

Conclusion

Difficultés rencontrées, limitations, améliorations possibles, ...

Annexe A

1234

Procédure d'installation ? Soit générer un exécutable, soit faire exécuter avec python 3.4 (pas 2.7)

En 2 parties : client et serveur, + configuration (adresse IP, ...)

Partie client : possibilité de donner l'image d'une machine virtuelle, correctement configurée