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

Nous avons imaginé le concept de **BotCop**, car la sécurité est devenu un thème central pour les habitations modernes. En effet, 75.000 cambriolages ont eu lieu en 2012 et en 2013 en Belgique [1].

Bien que de plus en plus d'habitations soient équipées de systèmes de surveillance électronique, ceux-ci ne sont pas toujours fiables. Ces systèmes sont généralement basés sur des capteurs de température et de mouvements, et peuvent se déclencher sans raison en été, lorsque les températures sont hautes.

Les systèmes de surveillance avec caméra sont très couteux. Notre projet permettrait, en complément avec un système meilleur marché, d'obtenir des images en temps réel de la maison.

Installer une caméra fixe n'est pas très utile, à moins de n'avoir qu'une seule pièce à surveiller. Il faut donc pouvoir déplacer cette caméra. Voilà donc pourquoi nous avons voulu créer un robot.

Concrètement, BotCop pourra se déplacer dans toute la maison et renvoyer l'image filmée par sa caméra à l'utilisateur.

L'utilisateur pourra donc :

- Contrôler le robot pour le déplacer,
- Contrôler la caméra, pour pouvoir regarder partout dans n'importe quelle direction,
- Visionner graphiquement le trajet vu du dessus dans l'interface

1. Source : `https://www.besafe.be/fr/diw/belgi-belgique`, visité le 13 août 2015

We imagined the **BotCop** concept, because security became a central theme for modern houses. Actually, 75.000 burglaries occurred in 2012 and in 2013 in Belgium [2].

Although more and more houses are equipped with electronic surveillance system, the latter are not always reliable. These systems usually are based on heat and movement sensors, and those can be triggered with no reason in the summer, when the temperatures are high.

Camera based surveillance systems are expensive. Our project would allow to get an image from the house in real time, with a cheaper system.

Set up a fixed camera is useless, unless there is only one room to monitor. This is why we need a mobile camera, and this is why we wanted to create a robot.

Concretely, BotCop can move in the house, and send the image back to the user.

The user will get the possiblity to :

- Control the robot to move it,
- Control the camera, to look everywhere in any direction,
- See a graphic representation of the path followed by the robot.

2. From `https://www.besafe.be/fr/diw/belgi-belgique`, visited on 13 august 2015

Chapitre 2

Matériel

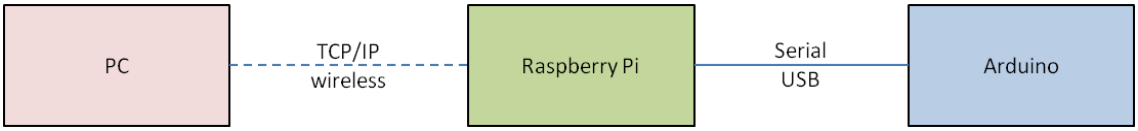

FIGURE 2.1: Architecture matérielle, et types de communication

Voici la liste complète du matériel que nous avons utilisé :

Kit Le kit que nous utilisons se compose des éléments suivants :
- Chassis Bundle
- Raspberry pi 2
- Arduino
- Batterie : TeckNet iEP387
- Carte micro SD
- Raspberry pi camera
- Wifi : Edimax EW-7811Un -150Mbps

Raspberry Pi Nous avons donc utilisé deux modèles de Raspberry pi lors de notre projet : le Raspberry Pi B+ et le Raspberry Pi 2. Bien que visuellement, ces deux modèles se ressemblent très fort, leurs caractéristiques sont différentes :

	Modèle B+	Modèle 2
Mémoire RAM	512 Mo	1 Go
Processeur	ARMv6	ARMv7 (4 coeurs)
Fréquence du processeur	700 MHz	900 MHz
Mémoire de stockage	MicroSD	MicroSD
Ports	4 USB 2.0, HDMI, RJ45 Jack (3.5mm), 40 broches GPIO	4 USB 2.0, HDMI, RJ45, Jack (3.5mm), 40 broches GPIO
Consommation	600 mA, 3.5 W	600 mA, 3.5 W
Prix	environ 30 €	environ 40 €
Système d'exploitation	Linux	Linux, ou Windows 10

Arduino C'est la carte Arduino qui fait le lien entre le Raspberry et les moteurs. Nous avons testé deux modèles différents :

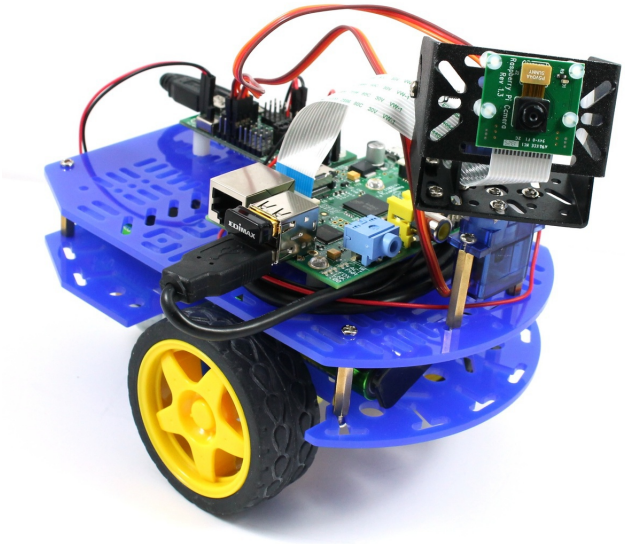

FIGURE 2.2: Robot complet

FIGURE 2.3: Raspberry Pi B+

FIGURE 2.4: Raspberry Pi 2

	Dagu mini driver	Dagu mini driver MKII
Processeur	ATMega8A	ATMega328P
Fréquence du processeur	16 MHz	16 MHz
Prix	environ 10 €	environ 12 €

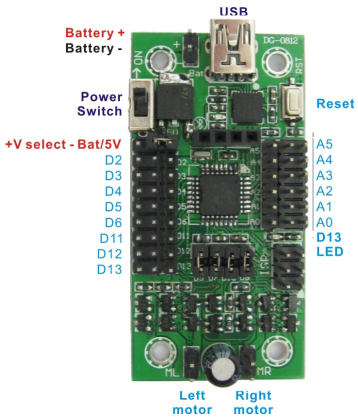

FIGURE 2.5: Dagu mini driver

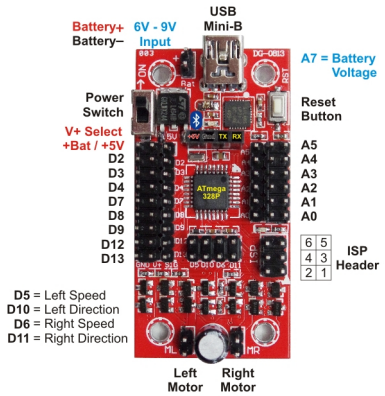

FIGURE 2.6: Dagu mini driver MKII

Chapitre 3

Implémentation

3.1 Le premier robot

Nous avions , dans un premiers temps, travaillé avec le Raspberry pi B+ et un contrôleur DC (DRV8833) pouvant gérer deux moteurs (1.2 A , 2.7-10.8 V). Cette solution, peut couteuse, nous a permis de travailler avec les pins GPIO du Raspberry.

Nos moteurs fonctionnaient avec une tension de 5 V, et étaient directement alimentés par ces pins. Nous avons beaucoup travaillé sur la partie électronique afin d'assembler au mieux un robot. Nous avions d'abord pensé implémenter directement dans le code l'envoi de signaux PWM dans les ports GPIO, avec une boucle dans le programme qui se répète toutes les 50 millisecondes et qui, selon les commandes envoyées, permet de faire avancer, tourner ou reculer le robot.

Les signaux PWM (Pulse Width Modulation) sont une technique qui nous permet d'obtenir une réponse analogique à partir d'un système fonctionnant en tout ou rien (système discret). Ce sont donc des ondes carrées dont le rapport de la largeur sur chaque période nous donne, dans ce cas-ci, la valeur de notre vitesse. La largeur de pulsation correspond au laps de temps au cours duquel le système a la valeur 1 (5 V). Cette largeur rapportée sur l'entièreté de la période, nous donne le pourcentage d'activité, le Duty Cycle.

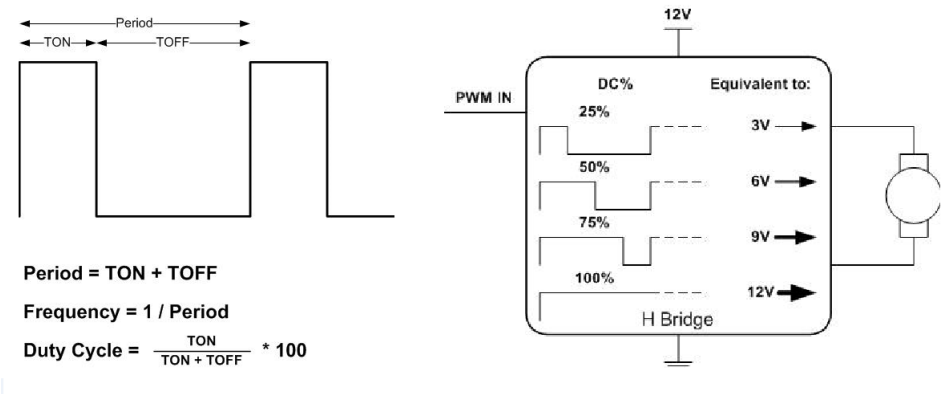

FIGURE 3.1: Principe des signaux PWM

Mais dans nos recherches, nous nous sommes aperçus que des librairies existaient déjà, et permettaient d'envoyer des signaux avec une seule commande, qui prenaient comme paramètre la vitesse relative des moteurs (100% pour faire tourner le moteur à pleine vitesse vers l'avant, par exemple)

Néanmoins, la qualité du matériel en notre possession n'était pas optimale : nous n'arrivions pas a fixer correctement les roues sur les axes des moteurs, et nous avons grillé le contrôleur DC lors de nos manipulations.

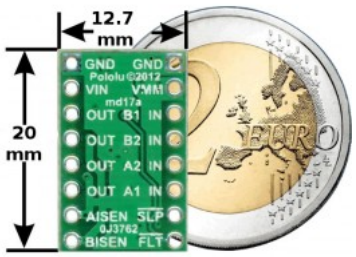

FIGURE 3.2: Contrôleur DRV8833

Nous avons réalisé à ce moment l'importance d'une bonne organisation de notre travail : nous avons opté pour une réalisation du projet étape par étape : c'est-à-dire que pour chaque chose à faire, nous la découpons en petites tâches "plus simples". Cette méthode présente deux avantages : nous voyons notre progression, car chaque étape est un pas vers la solution définitive, et si nous rencontrons un problème, nous pouvons revenir à une étape antérieure que nous savons fonctionnelle.

3.2 Construction du second robot

Nous avons donc pris l'initiative de changer pour un kit qui nous permet d'avoir des composants compatibles entre eux, et de meilleure qualité. Les moteurs sont donc directement relié aux pins de la carte Arduino, elle-même reliée au Raspberry avec un câble USB. L'ensemble du matériel utilisé est décrit en détail au chapitre 2. La carte Arduino sert à rajouter une couche d'abstraction. Concrètement, cela permet de ne gérer que de la communication sur le Raspberry. Le contrôle des moteurs est géré par la carte Arduino. En travaillant comme cela, nous ne devons plus gérer la partie électronique (génération de signaux PWM, ...).

3.2.1 Communication entre un PC et un Raspberry Pi

Pour notre robot, la création d'une connexion Socket entre un PC et un Raspberry était nécessaire.

La première étape est de créer une connexion entre un PC et ce même PC. Nous avons trouvé plusieurs exemples sur internet qui fonctionnent. Mais nous avons voulu aller plus loin pour cette étape :

- Ne pas envoyer une chaîne de caractère directement du code, mais la demander à l'utilisateur.
- Traiter la chaîne reçue sur l'autre programme, simplement l'afficher, afficher le premier caractère, ou si nous envoyons un nombre (envoyé sous forme de chaîne), récupérer ce nombre et effectuer une opération mathématique simple.
- Envoyer une chaîne (n'importe quoi, cela n'a pas d'importance), puis dès que l'autre programme reçoit cette chaîne, renvoyer une autre chaîne.
- Sur le PC, gérer avec Python les dates.

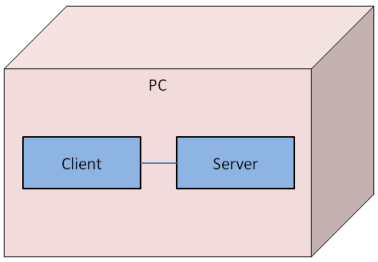

FIGURE 3.3: Première étape

Ainsi avec quelques étapes "faciles", nous pouvons affirmer :

- Nous savons ouvrir une connexion,
- Nous pouvons envoyer n'importe quelle information "simple" (types de données courants),
- Nous savons la récupérer et la traiter de l'autre coté,
- Nous savons estimer le temps que ça prend pour l'envoi (lors de nos tests, l'aller-retour prenait environ 0.7 secondes).
- Nous savons comment fonctionne le code, car au lieu de recopier bêtement d'internet, nous l'avons modifié en cherchant à comprendre son fonctionnement.

Ensuite, nous avons répété les mêmes processus entre un PC (sous Windows) et une machine virtuelle (sous Debian). Ces étapes sont normalement immédiates, puisque tous les codes ont déjà été écrits à l'étape précédente.

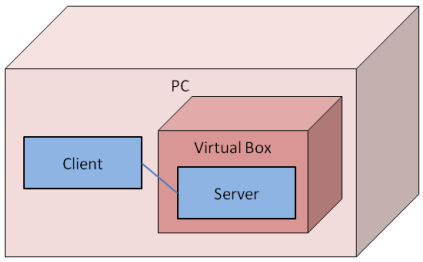

FIGURE 3.4: Deuxième étape

Enfin, nous avons refait la même chose entre le PC et le Raspberry. Nous avons dès lors atteint notre objectif, à savoir envoyer des instructions depuis le PC, vers le Raspberry.

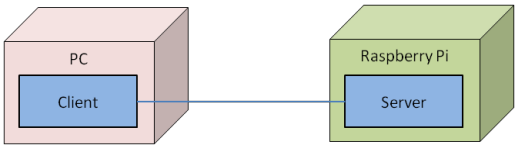

FIGURE 3.5: Troisième étape

3.2.2 Caméra

L'une des fonctionnalités principales du robot est de filmer et de renvoyer l'image en temps réel vers l'utilisateur.

Nous avons utilisé une webcam, branchée sur un des ports USB du Raspberry. Le programme *Motion* permettant alors d'envoyer l'image sur le réseau. L'utilisateur peut alors voir ces images, en entrant l'adresse IP du Raspberry, suivie du numéro du port. Nous avons configuré *Motion* pour qu'il envoie sur le port 8081. L'adresse à entrer sur le navigateur est :

```
192.168.1.50:8081
```

Sur le kit que nous utilisons, un emplacement est prévu pour la picaméra, qui se connecte au Raspberry sur le port CSI (Camera Serial Interface)

FIGURE 3.6: Port CSI sur le Raspberry

Cette caméra présente toutefois un inconvénient : la nappe de câbles est trop courte, et si nous faisons faire des mouvements trop larges à la caméra (par exemple regarder à 90 degrés sur la gauche ou la droite), la nappe se détache du port.

Puisque nous ne savons pas remplacer la nappe, nous avons inclus dans le programme *ControlMotors.ino* une limite d'angle de 45 degrés à gauche et à droite par rapport à l'avant du robot.

Chapitre 4

Organisation

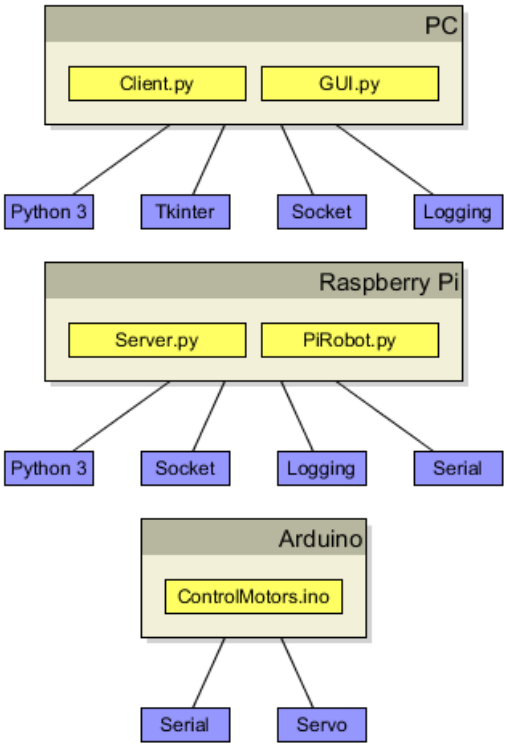

FIGURE 4.1: Logiciel

Notre code est divisé en trois parties :

- La partie Client, qui se trouve sur un PC (sous Windows, Linux, ...),
- La partie Serveur, qui se trouve sur le Raspberry Pi,
- La partie Contrôle, qui se trouve sur la carte Arduino.

4.1 PC

Client.py 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. A chaque fois qu'un bouton est poussé, une instruction est envoyée. Lorsque le bouton est relâché, une instruction `Stop` est envoyée.

GUI.py 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`).

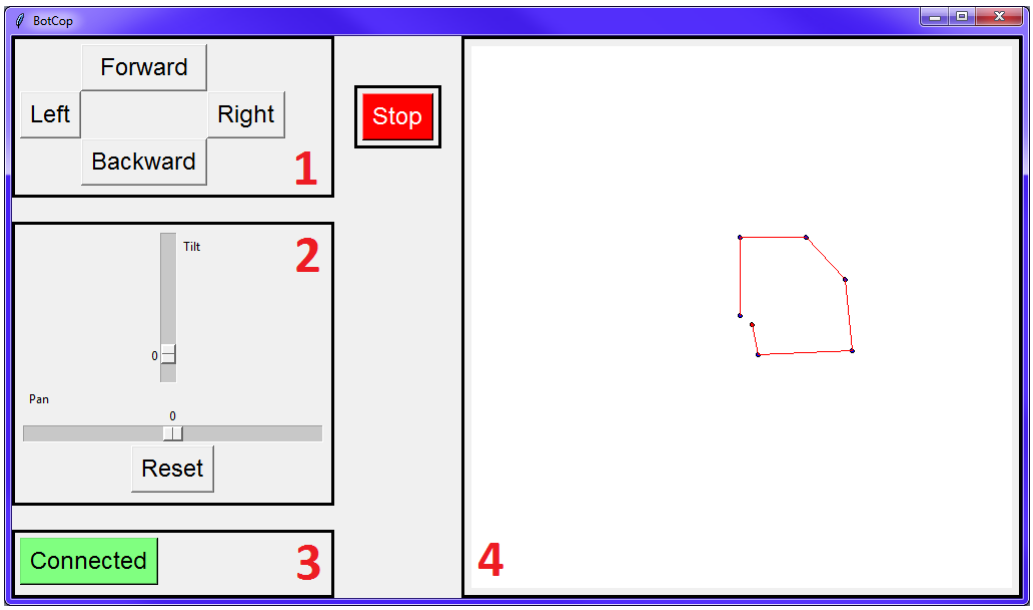

FIGURE 4.2: Interface graphique

L'interface est divisée en quatre zones, ou *frames* :

1 Contrôle du robot, pour le déplacer. Le robot continuera à avancer/tourner tant que le bouton est enfoncé, et s'arrête dès que le bouton est relâché.
2 Contrôle de la caméra, pour commander sa direction. Le bouton `Reset` permet de remettre la caméra à son état initial, c'est-à-dire regarder droit devant le robot, à l'horizontale.
3 Bouton de connexion. Tant que la connexion avec le Raspberry n'est pas établie, ce bouton restera rouge, et les autres boutons seront désactivés. Une fois que la connexion est établie, le bouton devient vert, et les autres boutons deviennent actifs.
4 Visualisation du trajet. Chaque déplacement du robot peut être vu graphiquement sur cette zone. Pour réinitialiser le graphique, double-cliquez dessus.

4.2 Raspberry Pi

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

PiRobot.py 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).

A chaque instruction reçue, une nouvelle chaîne de caractère est crée. Elle se présente de la manière suivante :

```
F180F180090020
```

On peut la décomposer en quatre parties :

```
F180 F180 090 020
```

Les deux premières parties sont les instructions pour les moteurs (le gauche puis le droit). La lettre (`F` ou `B`) indique le sens (AVANT ou ARRIÈRE), et les trois chiffres suivants indiquent la vitesse relative (255 pour 100% de la vitesse).

La troisième partie donne l'angle horizontal de la de la caméra (PAN). Par exemple `090` sert à orienter la caméra droit devant le robot.

La quatrième partie donne l'angle vertical de la caméra (TILT).

Enfin, cela n'est pas indiqué, mais la chaîne se termine par un caractère `\newline`, pour que le programme *ControlMotors.ino* sache que la chaîne est terminée.

4.3 Arduino

ControlMotors.ino reçoit la chaine d'instructions depuis le Raspberry, et la découpe pour envoyer les instructions à chaque moteur. Cette action est répétée à chaque fois qu'un caractère `\newline` est reçu.

4.4 Informations supplémentaires

Python 3 est la version de l'interpréteur utilisé. Nous précisons ce détail, car certaines fonctionnalités que nous utilisons ne portent pas les mêmes noms dans d'autres versions de Python, ou n'existent tout simplement pas. Sur le Raspberry, c'est la version 3.4, tandis que sur le PC, c'est la version 3.2.3. Il ne faut pas utiliser Python 2.7, car la compatibilité des programmes n'est pas assurée.

Tkinter sert à tracer l'interface graphique. Il permet d'organiser assez facilement les éléments dans la fenêtre graphique.

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

Serial permet d'ouvrir une connexion entre la carte Arduino et le Raspberry. La carte Arduino va se mettre à l'écoute sur un port choisi par l'utilisateur (port `\ttyUSB0`).

Logging sert à protocoler des informations, en les stockant dans un fichier .log. Par exemple, la commande suivante est appelée dans la fonction `buttonstopclick` :

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

Dans le fichier .log, nous verrons cette ligne apparaître :

```
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.

Github est un système de contrôle de révision, et peut être utilisé pour plusieurs choses :

- Il permet d'enregistrer plusieurs versions d'un code, en montrant qui l'a mis en ligne, quand cela a été fait, et les différences avec la version précédente. C'est utile si nous nous rendons compte que les modifications apportées à un code rend celui-ci inutilisable, en permettant de retélécharger un code que nous savons fonctionnel. Sur la figure 4.3, on voit que Sero17 (c'est le pseudo de Raphaël) a modifié le fichier `ClientGUI\GUI.py` il y a dix jours.
- Il permet aussi de garder une copie des codes. Si l'ordinateur ou le Raspberry rencontre un problème, le code est sauvegardé.

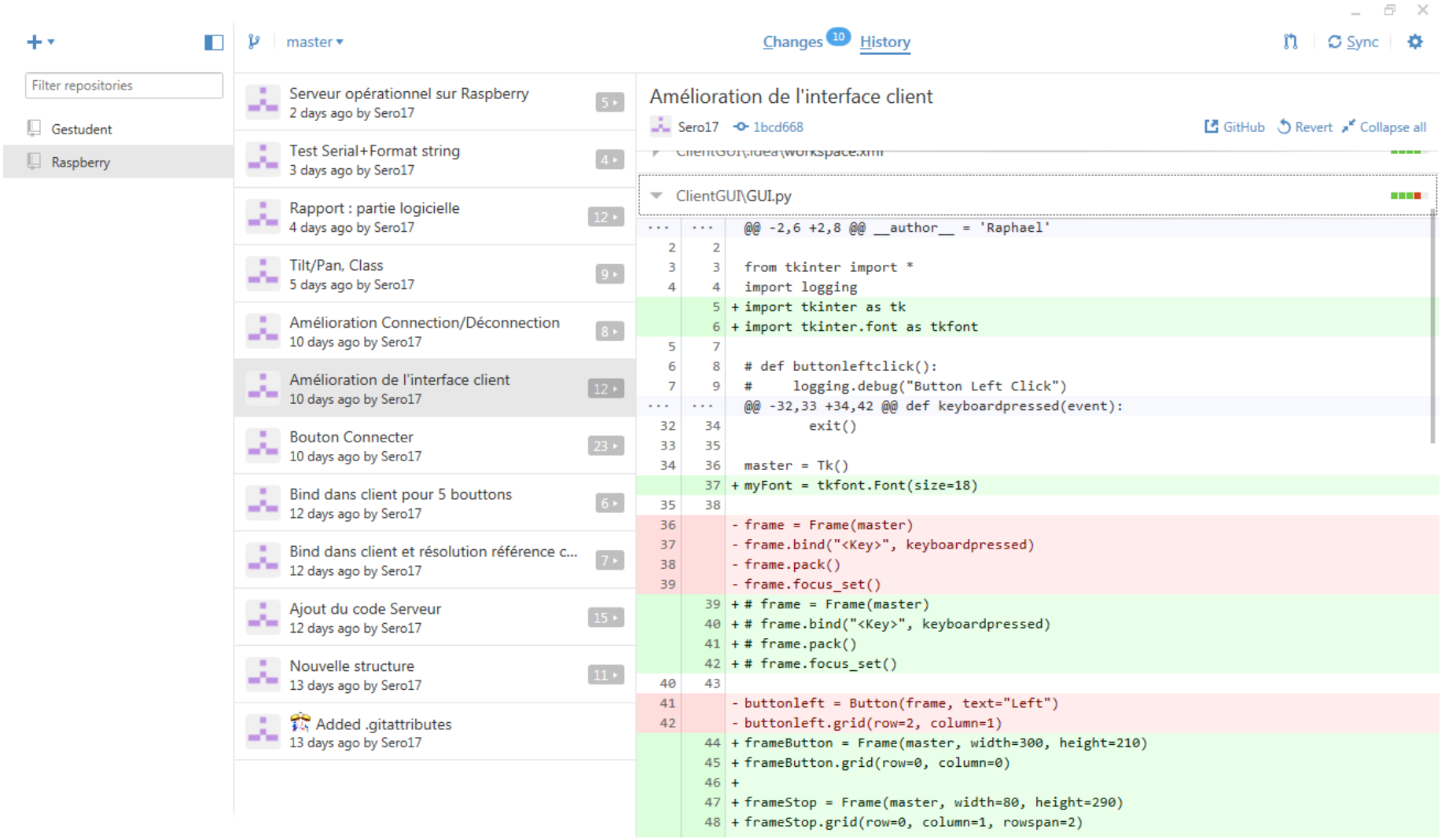

FIGURE 4.3: Interface de GitHub sous Windows

Chapitre 5

Le programme en action !

Pour contrôler le robot, il faut démarrer les programmes dans le bon ordre. Voici la marche à suivre :

Arduino La première chose à faire est de télécharger le code sur la carte Arduino. Pour cela, démarrez l'IDE Arduino, et ouvrez le fichier .ino que vous voulez mettre sur la carte.

Dans `Outils`, sélectionnez le modèle de carte (`Arduino Nano w/ATmega328`), le port (`ttyUSB0`) et le programmeur (`AVRISP mkII`).

Vérifiez le code, et téléversez-le sur la carte.

Notez que la carte Arduino peut garder en mémoire un programme, même si elle est mise hors tension.

Raspberry Pi Il faut démarrer le programme *Server.py* sur le Raspberry. Bien sur, on peut connecter un écran, un clavier et une souris. Dans ce cas, ouvrez un terminal, et allez dans le dossier ou se trouve le programme. Par exemple, si le programme se trouve dans le dossier *ServerRobot*, qui se trouve sur le bureau, entrez :

```
cd Desktop/ServerRobot/
```

Une fois dans le bon dossier, démarrez le programme avec :

```
python3 Server.py
```

Ce n'est pas très intéressant d'avoir un robot relié à un écran avec un câble HDMI. On peut se connecter à distance au Raspberry en utilisant Putty[1]. Introduisez l'adresse IP du Raspberry, le numéro de port (22), et le type de connexion (SSH). Lorsque la connexion sera établie, entrez l'identifiant `pi` et le mot de passe `raspberry`. Ensuite, il suffit de démarrer le programme comme décrit ci-dessus.

PC Enfin, démarrez sur le PC le programme *Client.py*, et appuyez sur le bouton `Connect` en bas à gauche.

1. Téléchargez Putty sur `http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html`

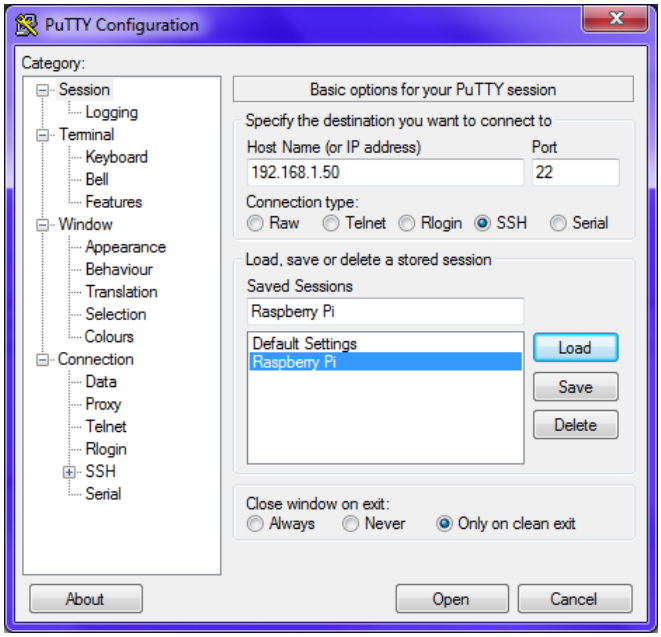

FIGURE 5.1: Configuraton de Putty

Chapitre 6

Conclusion

Ce projet nous a permis de découvrir et d'apprendre plusieurs choses : la programmation sur carte, la possibilité de contrôler un moteur à partir d'un programme, le langage Python, ... Mais aussi à organiser un programme quand celui-ci est en plusieurs parties. Il faut en effet avoir une vision globale de l'architecture logicielle pour pouvoir s'y retrouver.

Gérer un timing a été le plus difficile dans ce projet, notamment à cause de nombreux imprévus (matériel défectueux, ...). De même, c'était pour nous la première fois que nous avons du travailler en équipe sur un projet d'informatique.

Dans ce projet, nous avons surtout été limités par le matériel. Nous avons perdu beaucoup de temps pour nous rendre compte qu'un des composants ne fonctionne plus (cela nous est arrivé plusieurs fois). Un autre problème que nous avons eu est que le deuxième modèle de carte Arduino demande une tension supérieure à ce que la batterie peut fournir.

Pour encore améliorer notre robot, on pourrait imaginer rajouter des fonctonnalités pour la caméra : possibilité d'enregistrer une image ou un film, et éventuellement de déclencher cet enregistrement si un mouvement quelconque est détecté. On pourrait ajouter une lampe (pour que le robot puisse filmer dans le noir).

Annexe A

Procédure d'installation

A.1 Installation de Raspbian sur le Raspberry Pi

Voici la procédure à suivre pour installer Raspbian sur le Raspberry :

- Téléchargez NOOBS sur `https://www.raspberrypi.org/downloads/`, et extrayez les fichiers du zip.
- Formatez la carte SD (avec `https://www.sdcard.org/downloads/formatter_4/`). Sélectionnez la carte SD et dans les options choisissez *FORMAT SIZE ADJUSTMENT ON*, puis cliquez sur `Format`.
- Copiez les fichiers extraits de NOOBS sur la carte SD, puis éjectez la carte et insérez-la dans le Raspberry Pi.
- Connectez le Raspberry à un écran avec un câble HDMI, et connectez aussi un clavier et une souris sur les ports USB (un récepteur sans fil convient aussi).
- Alimentez le Raspberry. Attention il est déconseillé de brancher/débrancher des câbles lorsque le Raspberry est sous tension. Si vous souhaitez changer l'écran ou un autre périphérique, éteignez d'abord le Raspberry et débranchez-le.
- Sélectionnez Rasbian comme OS et la langue, ainsi que la configuration du clavier, et cliquez sur *Installer*.
- Après l'installation (compter environ 25 minutes), redémarrez le Raspberry Pi.
- Dans les options proposées, choisissez *Enable Boot to Desktop/Scratch*, et *Desktop Log in as user 'pi'*. Ensuite dans *Advanced Options*, choisissez *SSH*, et *Enable*. Allez sur *Finish* (avec Tabulation) et redémarrez le Raspberry.
- Si vous utilisez un écran avec un câble VGA et un adaptateur, il faut modifier `\boot\config.txt`. Il faut décommenter ou ajouter les lignes suivantes :
  ```
  hdmi_force_hotplug=1
  hdmi_group=2
  hdmi_mode=69
  hdmi_drive=2
  ```
  Le numéro 69 correspond à une résolution d'écran de 1920x1200, et une fréquence de 60 hz. Pour voir quel numéro correspond, voir `https://www.raspberrypi.org/documentation/configuration/config-txt.md` .
- Configurez le WiFi (voir section suivante), et mettez à jour le système en entrant ces deux commandes :

  ```
  sudo apt-get update
  sudo apt-get upgrade
  ```

A.2 Configurer le WiFi sur le Raspberry Pi

Nous allons configurer le Raspberry pour se connecter au routeur, et lui assigner une IP fixe.

- Lorsque le Raspberry est hors tension, insérez la clé WiFi dans un port USB.
- Toutes les informations concernant le réseau peuvent être retrouvées grâce à un autre appareil connecté à ce réseau : entrez dans un terminal ces deux commandes :

  ```
  sudo route -n
  ifconfig
  ```

- On édite un premier fichier de configuration : entrez cette commande :

  ```
  sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
  ```

  Ajoutez les paramètres du réseau Wi-Fi, et sauvegardez :

  ```
  network={
     ssid="<SSID du réseau local>"
     psk="<Clé de sécurité>"
     proto=RSN
     key_mgmt=WPA-PSK
     paiwise=CCMP
     auth_alg=OPEN
  }
  ```

- On édite un autre fichier : entrez cette commande :

  ```
  sudo nano /etc/network/interfaces
  ```

  Ajoutez les paramètres du réseau Wi-Fi, et sauvegardez :

  ```
  auto lo

  iface lo inet loopback
  iface eth0 inet dhcp

  allow-hotplug wlan0
  auto wlan0

  iface wlan0 inet static
  address 192.168.1.50
  netmask 255.255.255.0
  broadcast 192.168.1.255
  gateway 192.168.1.1
  wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
  iface default inet dhcp
  ```

  L'adresse 192.168.1.50 est l'adresse IP que j'ai choisie pour le Raspberry. Il faut choisir une adresse en dehors de la plage d'adresse réservée pour le dhcp.
- Réinitialisez le réseau avec ces deux commandes, ou redémarrez le Raspberry.

  ```
  sudo ifdown wlan0
  sudo ifup wlan0
  ```

Annexe B

Qui a fait quoi

B.1 Raphaël LEJEUNE

Lors de ce projet, je me suis occupé de la partie logicielle : j'ai effectué les recherches suivantes :

Configuration Comment configurer le Raspberry pour qu'il puisse se connecter à internet. Il fallait aussi lui assigner une IP fixe, pour faciliter les communications avec la PC.

Communication Comment envoyer des informations depuis le PC vers le Raspberry, et depuis le Raspberry vers la carte Arduino. Ces recherches incluent aussi la compréhension des exemples trouvés sur internet, dans le but d'avoir une meilleure maitrise des programmes que j'écris.

Interface Comment créer une interface graphique en Python. Une fois Tkinter trouvé, le même travail de recherche, d'essais et de compréhension du fonctionnement du package a du être fait.

Caméra J'ai aussi mené quelques recherches pour la caméra, et pour récupérer l'image via le réseau.

Après les travaux de recherche, j'ai écrit les trois programmes :

PC L'interface graphique : j'ai agencé les boutons pour avoir une fenêtre aussi organisée et logique que possible. J'ai aussi décidé que les messages envoyés seraient du type "FORWARD" ou "STOP".

Raspberry Le serveur : j'ai géré deux types de communication (Socket et Serial). J'ai placé tout ce qui concerne l'envoi vers la carte Arduino dans une classe séparée, tandis que dans le programme principal, j'ai utilisé le formatage de chaîne pour faciliter le travail sur la carte Arduino.

Arduino Ici mon travail a surtout été de récupérer la chaîne reçue du Raspberry, de la découper pour pouvoir envoyer les instructions aux moteurs.

J'ai utilisé GitHub pour garder une trace de chaque étape que j'ai réalisée. Lors de la rédaction du rapport, j'ai rédigé toutes les parties consacrées au logiciel (organisation, procédures d'installation, ...)

B.2 Maximilien POTTIEZ

En ce qui me concerne , je me suis occupé de la recherche, de l'achat et du montage du matériel. Je me suis intéressé à l'électronique du projet et à la manipulation de Linux. J'ai appris à souder grâce à l'aide du service d'électronique. De plus, il a fallu s'intéresser et comprendre de nombreux concepts et outils tels que les signaux PWM et leur fonctionnement à travers les contrôleurs. Le projet m'a permis de maitriser des éditeurs de texte comme vim ,le langage python et différentes de ses bibliothèques, de même que SSH et bien d'autres concepts utiles. De nombreuses directions furent prises lors de la conception de ce projet et beaucoup de concept appris ne se retrouvent pas dans le produit final tel que l'utilisation d'opencv, pygame et autres.

Annexe C

Sources

Voici la liste des sites internet que nous avons consultés pour ce projet

– http ://www.dawnrobotics.co.uk/dagu-arduino-mini-driver-board/
– http ://www.dawnrobotics.co.uk/dagu-arduino-mini-driver-mkii/
– http ://blog.dawnrobotics.co.uk/2013/11/getting-started-with-the-dagu-arduino-mini-driver-board/
– http ://blog.dawnrobotics.co.uk/2014/06/programming-raspberry-pi-robot-using-python-opencv/
– http ://www.dawnrobotics.co.uk/raspberry-pi-camera-robot-chassis-bundle/
– http ://blog.oscarliang.net/connect-raspberry-pi-and-arduino-usb-cable/
– http ://stackoverflow.com/questions/20107700/serial-receiving-from-arduino-to-raspberry-pi-with-pyserial-stops-after-a-while
– https ://www.raspberrypi.org/forums/viewtopic.php ?t=50243&p=389997
– https ://www.raspberrypi.org/documentation/configuration/wireless/wireless-cli.md
– http ://lifehacker.com/how-to-clone-your-raspberry-pi-sd-card-for-super-easy-r-1261113524
– https ://docs.python.org/3.2/library/socket.html
– http ://www.robot-maker.com/forum/tutorials/article/78-rcerda-un-robot-raspberry-pi-pour-100-120/
– http ://www.fred-j.org/ ?p=366
– http ://lea-linux.org/documentations/Raspberry_Pi