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 !

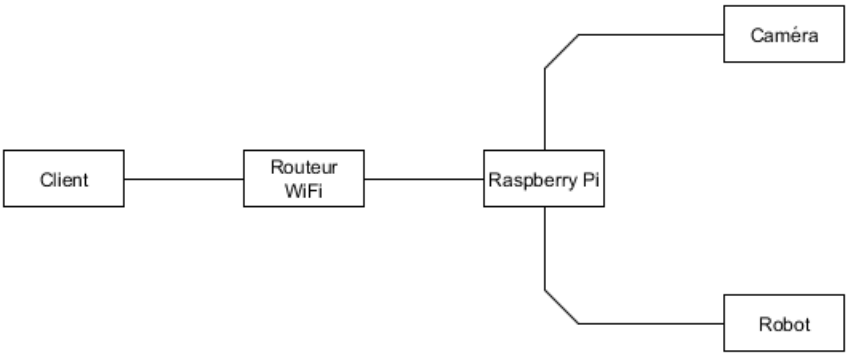

FIGURE 1.1: Matériel

Chapitre 2

Matériel

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

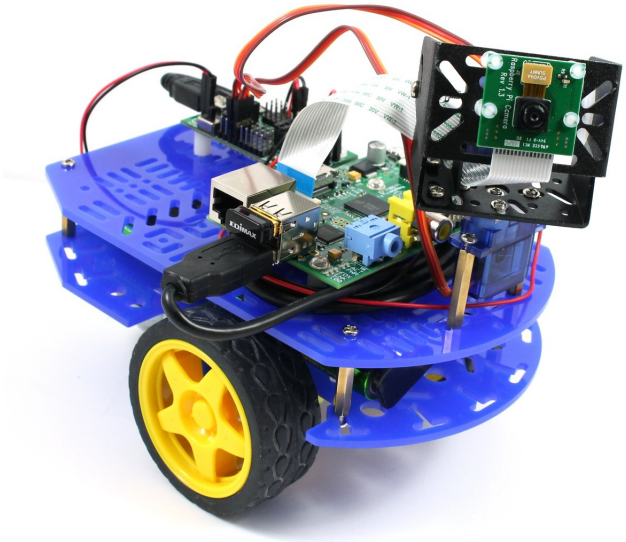

FIGURE 2.1: Robot complet

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

FIGURE 2.2: Raspberry Pi B+

FIGURE 2.3: Raspberry Pi 2

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

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

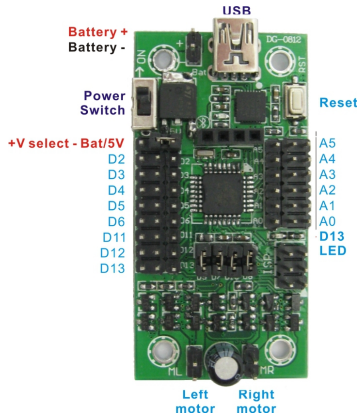

FIGURE 2.4: Dagu mini driver

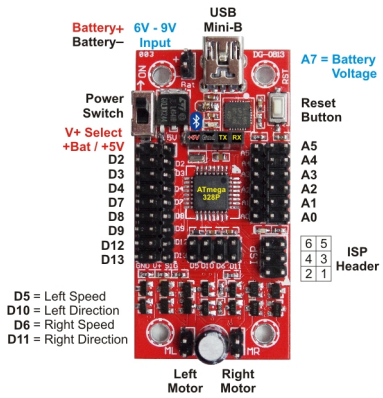

FIGURE 2.5: Dagu mini driver MKII

Chapitre 3

Organisation

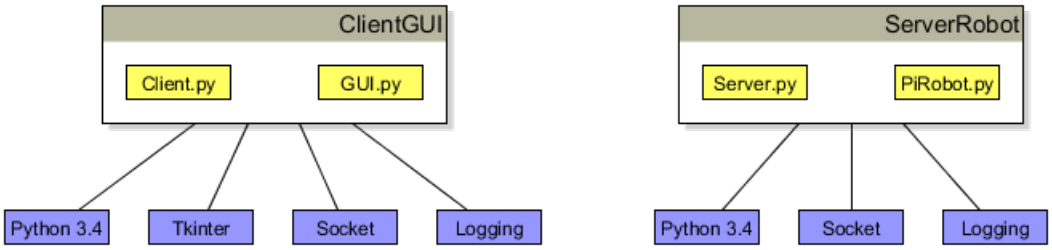

FIGURE 3.1: Logiciel

Notre code est divisé en trois parties :

- La partie client, qui tourne 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.

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.

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`).

Server.py 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 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 3.2.3 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.

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

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 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.

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

- Garder une copie des codes. C'est la raison principale. Si l'ordinateur ou le Raspberry rencontre un problème, le code est sauvegardé.
- Il permet aussi 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 on se rend compte que les modifications apportées à un code rend celui-ci inutilisable, en permettant de retélécharger un code qu'on sait fonctionnel. Sur la figure 4.2, on voit que Sero17 (c'est le pseudo de Raphaël) a modifié le fichier `ClientGUI\GUI.py` il y a dix jours.

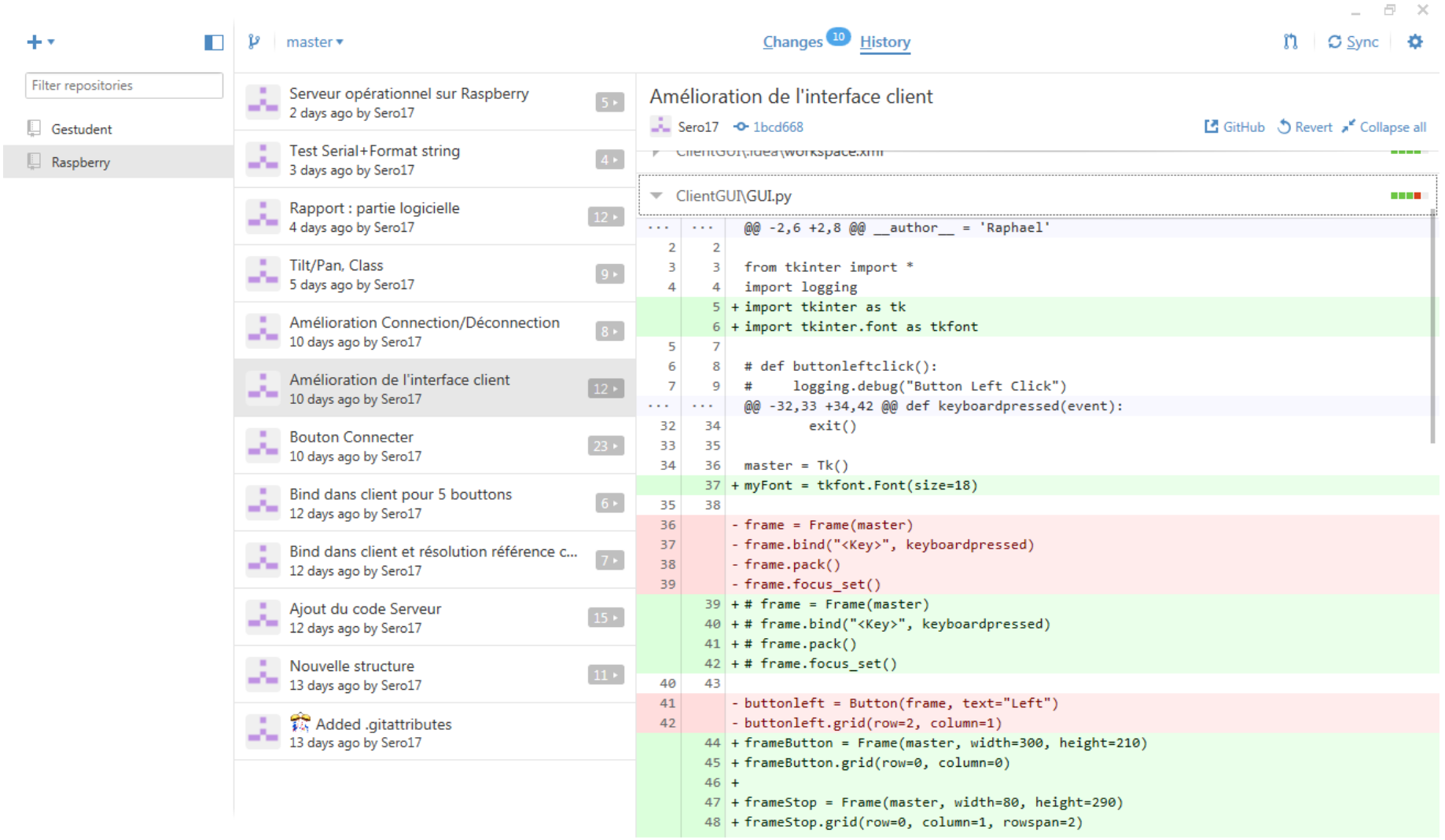

FIGURE 3.2: Interface de GitHub sous Windows

Chapitre 4

Implémentation

4.1 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.

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.

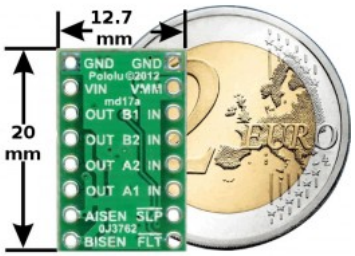

FIGURE 4.1: Contrôleur DRV8833

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, on la découpe en petites tâches "plus simples". Cette méthode présente deux avantages : on voit notre progression, car chaque étape est un pas vers la solution

définitive, et si on rencontre un problème, on peut revenir à une étape antérieure qu'on sait fonctionnelle.

4.2 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.

4.3 PWM

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.

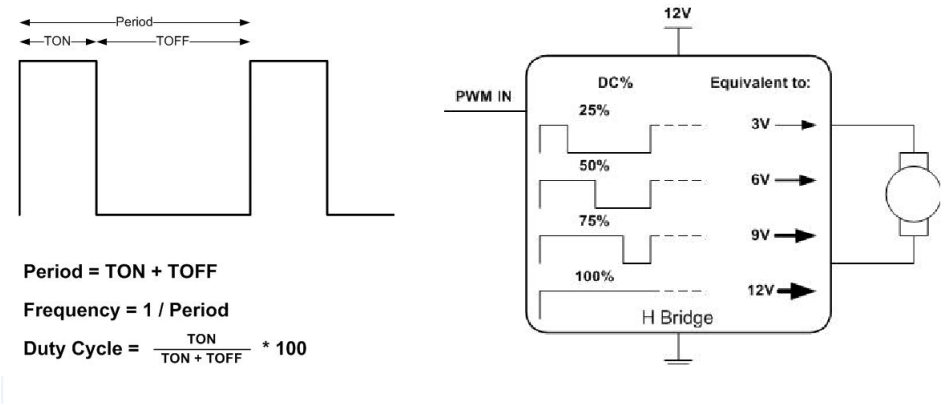

FIGURE 4.2: Principe des signaux PWM

4.4 Socket

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

La première étape est de créer une connection 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 on envoie 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.

Ainsi avec quelques étapes "faciles", on peut affirmer :

– On sait ouvrir une connection,
– On peut envoyer n'importe quelle information "simple" (types de données courants),
– On sait la récupérer et la traiter de l'autre coté,
– On sait estimer le temps que ça prend pour l'envoi (lors de nos tests, l'aller-retour prenait environ 0.7 secondes).
– On sait comment fonctionne le code, car au lieu de recopier bêtement d'internet, on l'a 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.

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.

Chapitre 5

Qui a fait quoi

Raphaël
Recherches sur connection socket
Programme client, interface, programme serveur sur raspberry, connection serial avec arduino Backup régulier des codes sur GitHub, ...

Chapitre 6

Conclusion

Difficultés rencontrées, limitations, améliorations possibles, ...

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
  ```