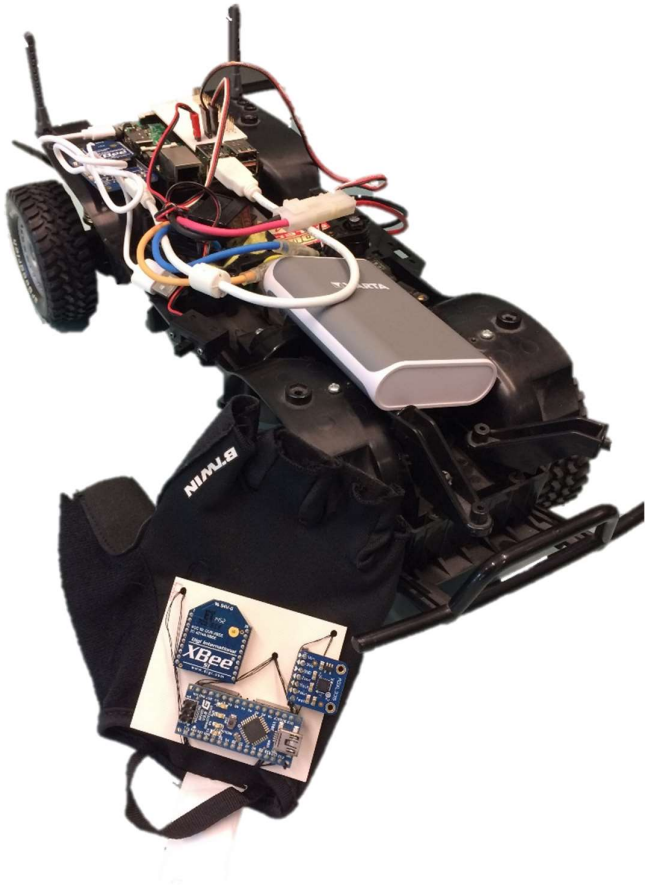

12/05/2017

Rapport de Projet IMA4

Voiture radiocommandée contrôlée par un gant

Thomas GOSSE, Bacem HAGUI

Sommaire

Table des matières

Introduction

Dans le cadre de notre quatrième année d'étude (dans la filière IMA de Polytech Lille), nous avons dû réaliser un projet mettant en application les compétences développées au cours de notre formation. Nous avons choisi de travailler sur le projet « Voiture radiocommandée contrôlée par un gant » qui consiste à réaliser une commande de contrôle sans fil pour une voiture radiocommandée ainsi qu'à contrôler les moteurs de cette dernière.

Le projet mobilise donc des connaissances acquises théoriquement, telles que la communication sans fil entre deux objets, la programmation informatique, mais aussi la réalisation de circuits électroniques et le contrôle de servo-moteurs.

De plus, ce projet s'inscrit dans un contexte où peu de nouveautés au niveau du contrôle des voitures radiocommandées ont été apportées depuis leur création. Nous pensons donc pouvoir créer une commande différente et intuitive, qui pourrait intéresser un large public.

Dans ce rapport, nous nous intéresserons dans un premier temps au cahier des charges du projet, ainsi qu'aux choix techniques mis en place pour y répondre. Ensuite nous présenterons la partie concernant la commande de la voiture. Dans une troisième partie, nous verrons comment a été réalisée la partie embarquée sur le véhicule, également le contrôle des moteurs et la calibration du contrôleur de moteur. Enfin, la dernière partie comportera les problèmes rencontrés et les résultats obtenus à l'issue du projet.

Cahier des charges

Objectifs du projet

Le projet "Voiture radiocommandée contrôlée par gant" propose donc de réaliser entièrement le gant qui contrôlera la voiture, mais également de réaliser le système de réception et de traitement de données venant du gant (contrôle des servomoteurs). La voiture doit pouvoir se déplacer en suivant le mouvement de la main :

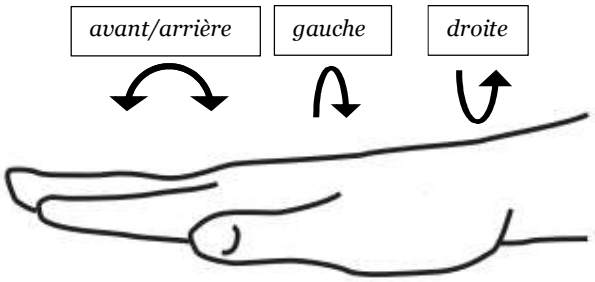

Figure 1 : différentes positions de la main pour contrôler la voiture

Nous prévoyons plusieurs positions (plus ou moins orientées) pour avant/arrière afin que le voiture puisse aller plus ou moins vite.

Choix techniques

Afin de répondre correctement au cahier des charges imposés, nous avons décidé de couper le projet en trois grandes parties :

- La partie commande composée d'un **gant** sur lequel nous collerons un **accéléromètre**. L'accéléromètre sera utilisé pour faire tourner la voiture à droite ou à gauche, mais également pour avancer et reculer, suivant le mouvement de la main ci-dessus. Afin de récupérer et traiter les données issues des capteurs, nous aurons besoin d'une plateforme **Arduino Nano** accrochée au gant.

- La partie embarquée sur la **voiture** sera composée d'une **Raspberry Pi**. Cette dernière aura pour fonction de recevoir les données envoyées par le premier et de commander les servomoteurs.

Elle sera également utilisée pour héberger un serveur Web (qui sera réutilisée dans la cadre des TP IMA5). Il faudra donc qu'elle soit paramétrée comme balise Wi-Fi.

- La partie transmission, assurée par une communication ZigBee et réalisée avec deux modules XBee : un connecté à l'Arduino sur le gant, et l'autre connecté en USB à la Raspberry Pi. Ce protocole permet une transmission de donnée sur une portée de 30m en intérieur et jusqu'à 100m en extérieur, ce qui est convenable pour le projet.

Enfin, nous utiliserons un PC sous Linux pour programmer ainsi que Fritzing pour la réalisation de cartes électroniques.

Réalisation de la commande

Choix de la carte

Comme présenté dans la partie précédente, la partie commande situé sur le gant est composée d'une Arduino. Nous avons commencé par prendre une Arduino Mini, qui ne disposait pas de port USB pour téléverser les programmes dessus. Nous avons d'abord essayé un montage avec un FTDI afin de connecter la carte au PC :

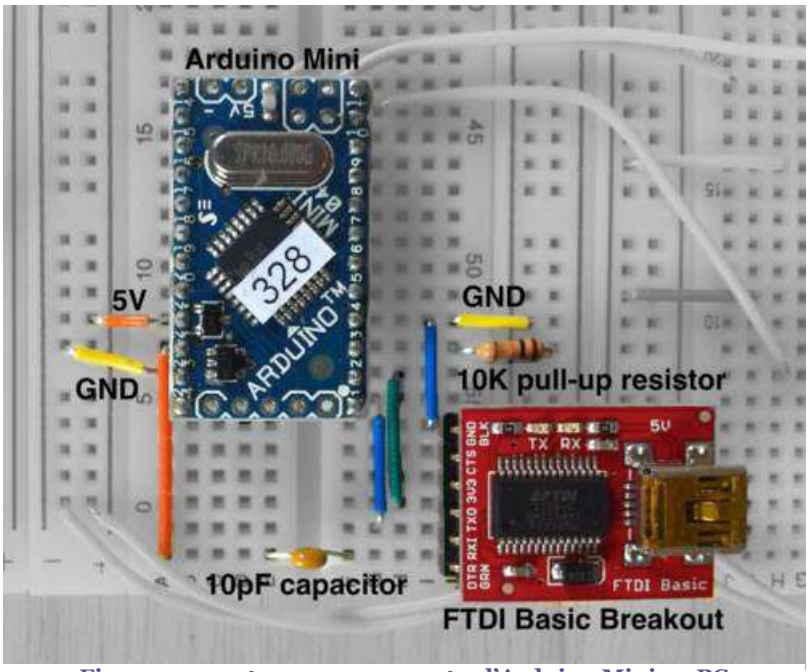

Figure 2 : montage pour connecter l'Arduino Mini au PC

Après plusieurs essais et en changeant le montage, nous n'arrivions toujours pas à charger de programme sur l'Arduino, nous avons donc décider de changer pour l'Arduino Nano, légèrement plus grande, mais disposant d'un port USB.

L'accéléromètre

Le capteur que nous avons décidé d'utiliser est l'ADXL335, un capteur déjà utilisé lors d'anciens projets et facile à mettre en œuvre avec l'Arduino. Il est alimenté en 3,3V par cette dernière et l'axe X et Y sont reliés directement aux pins analogiques de la carte Arduino (A0 et A1).

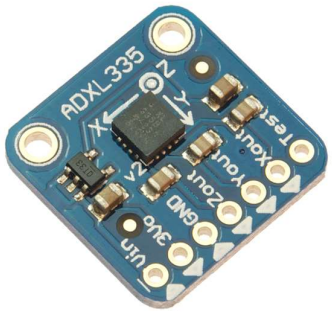

Figure 3 : ADXL335

Emission des données

Afin d'assurer la transmission de données entre l'Arduino et la Raspberry, nous avons choisi ZigBee. C'est un protocole à consommation réduite utilisée pour la communication entre des petits modules. On peut le comparer au Bluetooth pour sa courte distance d'émission, cependant, la technologie ZigBee demeure moins chère et plus simple d'utilisation. ZigBee est utilisé dans la bande 2,4GHz.

Sur le gant, nous devions donc fixer notre émetteur. Pour cela, nous avons choisi un module XBee classique.

La datasheet et le schéma avec les pins du module (ci à droite) nous ont permis de connecter le XBee à l'Arduino Nano. Le module est alimenté via le 3,3V de l'Arduino.

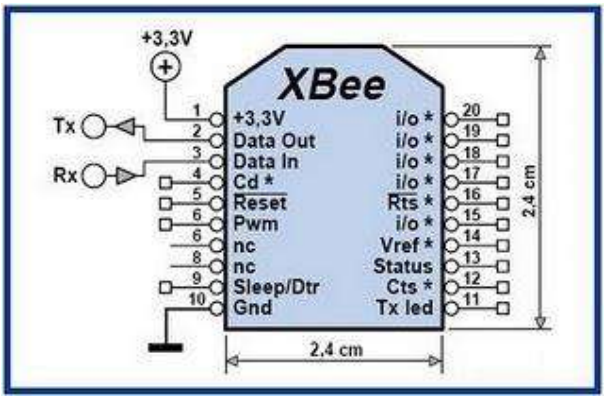

Figure 4 : pins du XBee

Le module XBee étant connecté sur le Tx et le Rx de l'Arduino, il suffit d'écrire sur la liaison série de l'Arduino pour que le module XBee transmettre les données dans son canal d'émission.

Le canal a été paramétré à l'aide du logiciel X-CTU (proposé pour XBee) : nous avons choisis « OxOC ». La programmation de la carte est faite via l'IDE Arduino et le code est simplement le suivant :

```
int val = analogRead(A0);
 Serial.print(val);
```

Afin de tester la connexion, nous avons connecté un module XBee à un PC et paramétré son canal sur OxOC également afin de visualiser les données reçues (Ce second module est utilisé sur la Raspberry Pi) ci-après :

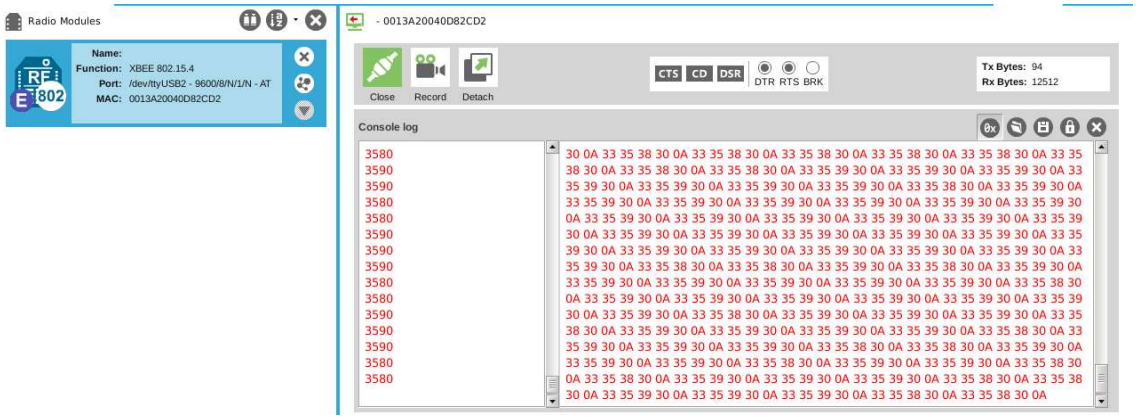

Figure 5 : données reçues venant de l'accéléromètre

Le programme dans sa version finale permet à l'Arduino d'envoyer les valeurs de l'axe X et de l'axe Y une fois sur deux chacune leur tour, tel un petit protocole propre à notre système (voir le code lié à ce rapport). De plus, en fonction de l'inclinaison de la main, différentes valeurs sont envoyées, afin de gérer plusieurs vitesses pour la voiture. Ainsi, pour avancer peu rapidement, l'Arduino envoie « 2 », pour avancer plus vite « 3 » et pour reculer « 1 ». Lorsque l'accéléromètre est penché à droite, la valeur envoyée est « 11 », et « 16 » pour la gauche.

Création du gant

Pour faciliter les connexions entre les 3 éléments du gant et se passer de câbles, nous avons réalisé une carte électronique liant l'Arduino Nano, l'accéléromètre et le XBee. La carte est réalisée sur Fritzing et les barrettes ensuite soudées à la main :

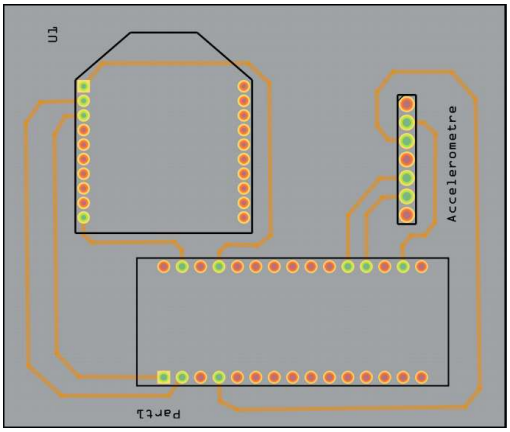

Figure 6 : PCB de la carte du gant sous Fritzing

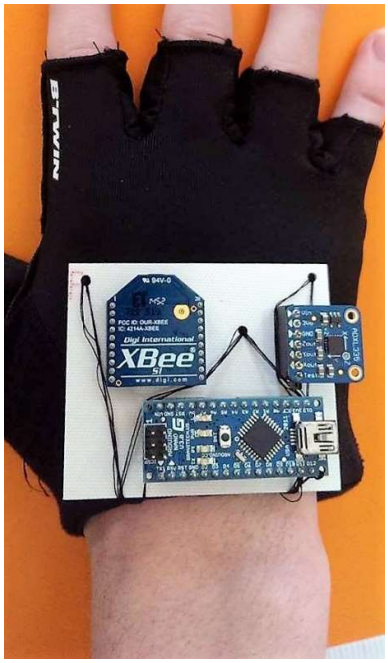

Figure 7 : plaque fixée sur le gant

La partie gant est donc fonctionnelle, la suite du rapport porte sur la partie embarquée de notre projet.

Le système embarqué

Paramétrage de la Raspberry Pi

La Raspberry Pi étant le cœur de notre système embarqué sur la voiture, nous avons commencé tout d'abord par sa configuration. La première étape était d'activer la liaison série, nous avons monté la carte SD de la Raspberry sur un ordinateur et ajouté la ligne "enable_uart=1" dans le fichier config.txt. Étant désormais capable de se connecter en liaison série sur la Raspberry nous avons poursuivi la configuration. Grâce à l'outil minicom (configuré comme suivant : 115200 bauds, 8N1 = 8 data bits, pas de bit de parité et 1 bit de stop), sur */dev/ttyUSBn* (avec n un numéro entier qui désigne le port su lequel la Raspberry est connecté et sans flux de contrôle) nous pouvions nous connecter à la Raspberry.

Nous avons modifié le fichier */etc/network/interfaces* pour définir une adresse IP statique de l'appareil :

```
auto eth0
iface eth0 inet static
  address 172.26.78.112
  netmask 255.255.240.0
  gateway 172.26.79.254
```

De cette manière, la Raspberry Pi est accessible en SSH lorsque l'on connecte dessus un câble Ethernet jusqu'à un PC. Afin que la carte puisse béneficier de la connexion internet, il a également fallu ajouter l'adresse IP du serveur DNS de l'école dans le fichier */etc/resolv.conf*.

Comme nous voulions mettre notre Raspberry Pi en point d'accès, nous avons ajouté au fichier */etc/network/interfaces* les lignes suivantes :

```
allow-hotplug wlan0
  iface wlan0 inet static
  address 192.168.100.1
  netmask 255.255.255.0
```

Cela donne donc l'adresse de la Raspberry Pi sur le réseau local. Nous avons également installé un serveur DHCP avec la configuration nécessaire pour que le point d'accès délivre des adresses aux appareils s'y connectant. Enfin, la dernière étape est l'installation du daemon « Hostapd » qui permet la création du point d'accès Wi-Fi (et qui est également à configurer). La Raspberry Pi est détectable par les appareils sous le nom de « gosse_hagui ».

Réception des données

Ayant des modules XBee déjà configuré à l'aide du logiciel X-CTU, nous avons branché un de ces derniers sur un des ports USB de la Raspberry Pi et avons écrit un simple programme en C pour tester la bonne réception des données envoyées par le premier module. Pour lire les valeurs arrivant sur la liaison série de la Raspberry Pi, nous utilisons la fonction *fopen* sur le fichier */dev/ttyUSB0* pour l'ouvrir. Afin de lire les valeurs, nous utilisons *fscanf*, qui imprime tout simplement ces dernières dans une variable, ensuite utilisée pour les PWM.

Le programme affichait simplement les valeurs reçues sur la liaison série. La communication entre les deux modules étant réussie, nous avons continué sur le contrôle des moteurs.

Contrôle des moteurs

Pour contrôler les moteurs, nous avons utilisé des PWMs générée sur les broches de la Raspberry Pi (GPIO) grâce à une librairie appelée WiringPi.

Ces librairies permettent de créer des signaux PWM contrôlés par logiciel sur n'importe quelle broche de GPIO. Nous avons utilisé les trois fonctions suivantes :

- *int wiringPiSetupGpio (void)* : initialisation de la librairie WiringPi et permet l'utilisation des fonctions qui suivent (retourne 0 si tout c'est bien passée et 1 au cas d'erreur).

- *int softPwmCreate (int pin, int initialValue, int pwmRange)* : autorise une GPIO à être contrôlée par une PWM. Un numéro de pin GPIO prise en paramètre par pin, la valeur max (range) dans *pwmRange*, et une valeur initiale pour *initialValue* (0 pour « OFF » et 100 pour valeur maximale par exemple). Cette fonction retourne également 0 si tout s'est bien passé, tout autre valeur doit être vérifiée.

- *void softPwmWrite (int pin, int value)* : mise-à-jour de la valeur (le paramètre value est la nouvelle valeur) sur une pin donné (paramètre pin). La pin doit être préalablement initialisé et la nouvelle valeur doit être dans l'intervalle sinon cette commande sera ignorée.

Le fonctionnement des PWMs a d'abord été testé avec des LEDs. Nous avons ensuite directement relié la Raspberry Pi aux moteurs. Le code est plutôt simple et effectue, pour chaque valeur reçue, un traitement associé (une valeur de PWM) sur le moteur concerné Par exemple, si on reçoit « 2 », cela signifie que l'utilisateur a penché sa main, la Raspberry envoie la valeur « 11 » de PWM pour faire avancer la voiture à vitesse réduite.

Afin que les valeurs de PWM envoyées soient cohérentes avec le moteur, nous avons dû paramétrer le contrôleur de moteur pour la vitesse :

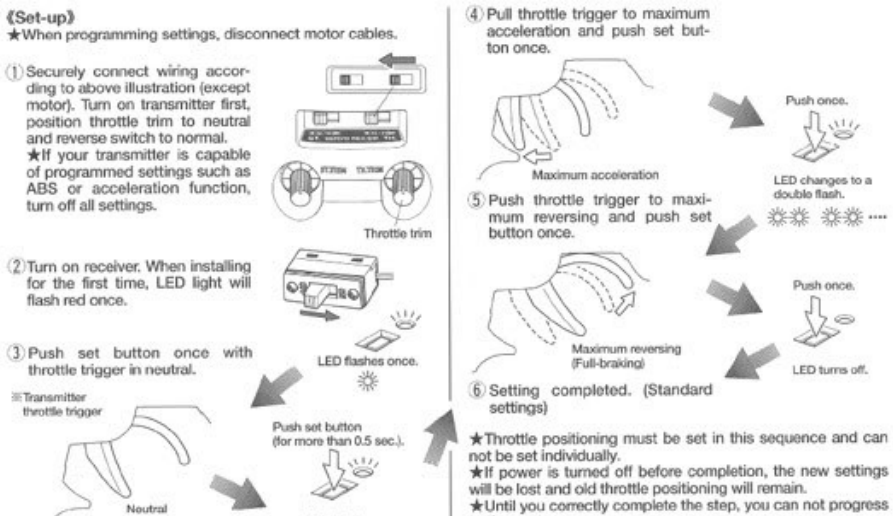

Figure 8 : notice de calibrage du contrôleur de moteur

Nous avons simulé des différentes positions de la télécommande par les valeurs de PWM désirées. La configuration convenant le mieux était une range de 20 pour la PWM de la vitesse, la valeur 10 étant la valeur d'arrêt (<10 une vitesse négative, >10 une vitesse positive).

Câblage du système sur la voiture

Comme pour la partie sur le gant, nous avons réalisé un PCB pour rassembler tous les éléments de la partie embarquée. De plus, ce « shield » pour Raspberry nous permet d'avoir une masse commune à tous les éléments, conditions nécessaires pour envoyer un signal PWM aux moteurs.

Le schéma de câblage est le suivant :

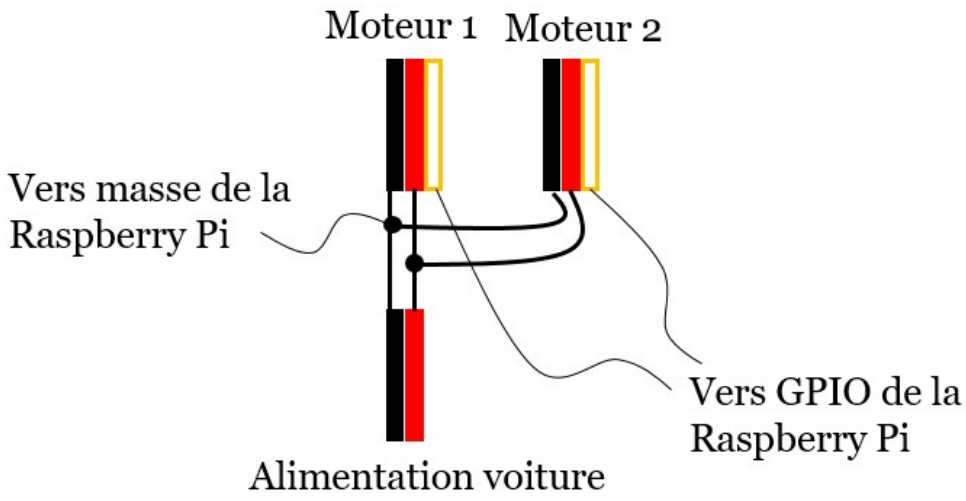

Figure 9 : câblage entre les éléments

Le PCB, également réalisé sous Fritzing et soudé à la main est le suivant :

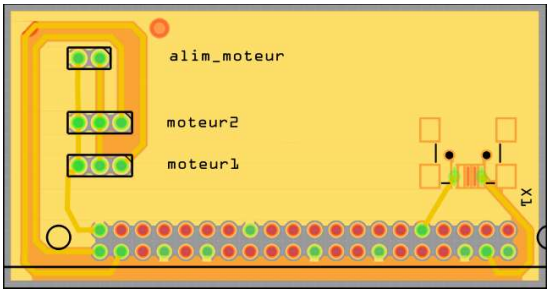

Figure 10 : shield pour la Raspberry Pi

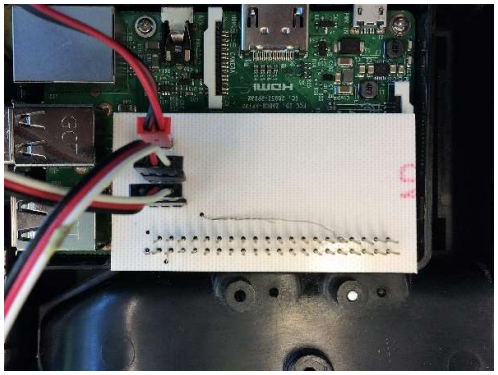

Figure 11 : shield fixé directement sur la Raspberry Pi

La dernière partie concerne les problèmes rencontrés ainsi que les résultats obtenus à la suite de ce projet.

Problèmes rencontrés et résultats

Tout au long tu projet, nous avons réalisé des tests pour valider chaque étape et ainsi faire fonctionner toute les parties ensemble.

Trouver les valeurs de PWM nous a pris beaucoup de temps sur la fin du projet car il fallait tester beaucoup de range et de valeurs afin d'obtenir un résultat correct :

- Sur le moteur qui fait avancer la voiture, il a fallu effectuer plusieurs calibrages avec la procédure ci-avant. Au début, la voiture roulait tout le temps à pleine vitesse, nous avons réussi à ajouter une vitesse lente et une vitesse plus rapide. Cependant, la marche arrière ne fonctionne pas systématique malgré des valeurs de PWM envoyées qui semblent correctes.

- Pour ce qui est du servomoteur qui s'occupe de la direction, cela a été plus facile mais par soucis de fluidité, nous n'avons gardé que 3 positions : droite, gauche et milieu. En effet, ce servomoteur est sensible et ajouter plusieurs autres positions n'apportait rien de remarquable.

Nous avons également eu quelques problèmes avec le shield de la Raspberry Pi. En effet, celui-ci ayant été mal soudé la première fois, nous avons dû le refaire car la voiture ne fonctionnait plus. Nous nous sommes alors rendu compte de l'importance d'avoir une masse commune à tous les éléments : sans cela, le signal PWM n'était pas envoyé.

Finalement, la voiture est contrôlable facilement sur une distance d'une vingtaine de mètres et réagi comme prévu aux mouvements de la main. (voir la vidéo sur notre wiki)

Conclusion

A travers ce projet, nous avons pu mettre en pratique tout ce que nous avons appris au cours de notre formation en IMA, de la programmation en C et en Arduino, en passant par la communication entre deux entités, mais également la réalisation de cartes électroniques. Nous nous sommes servis d'une librairie développée par d'autres personnes, ce qui nous a appris à nous adapter, à collecter des informations, et à intégrer dans un projet une production qui n'est pas notre.

Ce projet nous a également permit de travailler à deux, donc de nous répartir les tâches et de gérer un projet commun. L'utilisation de git était également appréciable car c'est un outil très utilisé.

La réalisation, les finitions apportées (telles que la création du gant par exemple), mais aussi les problèmes rencontrés nous ont montré qu'il n'y a pas que le code qui compte et que dans un vrai projet, tout doit être pris en compte.

Cela nous apporte donc une expérience supplémentaire dans le milieu des systèmes communicants