DOSSIER PERSONNEL

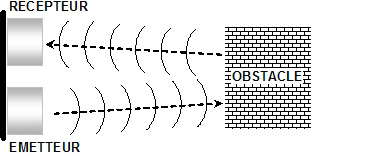
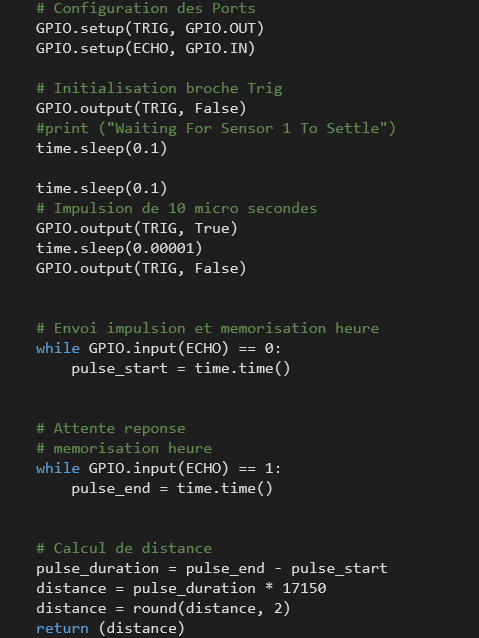
1. INTRODUCTION :

Je me suis principalement occupé de la structure du projet, c'est-à-dire savoir à quoi nous allions aboutir. J’ai été le responsable du hardware (électronique) et du codage de tous les programmes en python

Le travail à commencé par des recherches sur internet. J’ai commencé à voir les différents moyens d’utiliser le langage python d’une manière originale. Je savais que des projets alliant électronique et python existaient.

Il est en effet possible de maîtriser les GPIO (broches électriques) d’un Raspberry pi avec la console Linux ou avec un programme en python. J’ai alors eu l’idée de faire une voiture autonome en se basant sur un Raspberry pi. J’ai directement proposé l’idée à Matthis et Raphaël. En cherchant sur internet et j'ai commencé à voir les différents moyens de faire une voiture autonome à l'aide d'un Raspberry pi. Utiliser de l'électronique avec Python nécessite au minimum de connaître les boucles, les booléens, et les tests conditionnel. Cependant de nouvelles bibliothèques sont utilisées comme RPi.GPIO et time, de ce fait de nouvelles fonctions sont à disposition.

1. RADAR A ULTRASONS:

Plusieurs solutions s'offraient à nous utiliser une caméra des capteurs infrarouges et des capteurs à ultrasons le moins cher et le plus accessible (facilité de programmation) étaient les capteurs à ultrason. Le principe du fonctionnement d'un émetteur à ultrason et un récepteur à ultrason et analogue à celui haut-parleur et un micro (vibration de membrane). L’émetteur envoie une salve d’onde ultrasonore pendant un très cours lapse de temps. L’onde rebondit sur un obstacle (mur) puis est renvoyé sur le récepteur. Une onde ultrasonore à la même célérité (vitesse) qu’une onde sonore soit 343 m.s-1 environ. L’onde est dite *ultrasonore* car sa fréquence est élevé (>50 000kHz) ce qui la rend inaudible pour l’oreille humaine. On peut ainsi en fonction du temps entre l’émission et la réception en déduire par le calcul la distance parcourue par l’onde : **d = (v.T)/2**

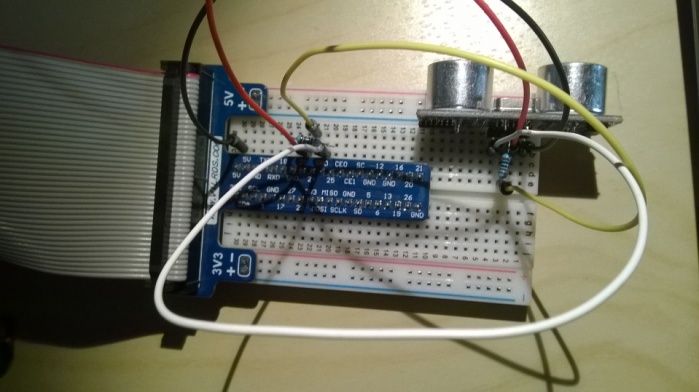
On défini :

v : 17 150 cm.s-1

T = variable en secondes

d = résultat attendu en cm

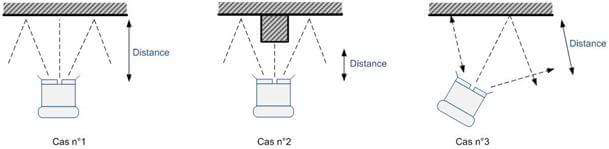
Dans le programme cela donne :

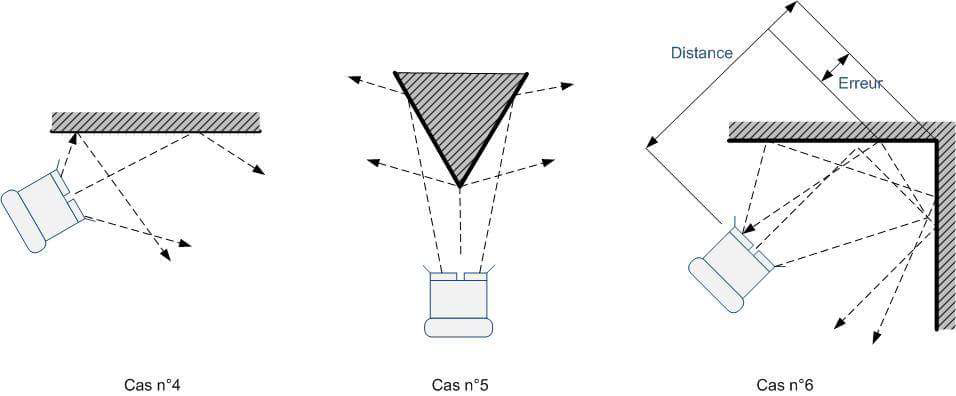


**Branchements pour 1 capteur à ultrasons**

J’ai utilisé une breadboard pour connecter les fils entre eux sans avoir à faire de soudures.

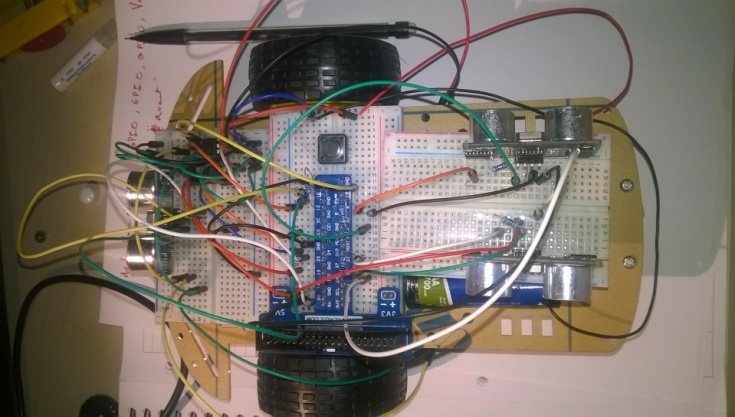
Un problème que j’ai rencontré avec les capteurs à ultrason est que ces derniers rencontrent des problèmes lorsque l’inclinaison des capteurs vis-à-vis d’un obstacle dépasse 30° :





Un problème s'est rapidement posé. Le code fonctionnait parfaitement pour un seul capteur faut que le but final était d'en avoir plusieurs. Le fonctionnement de plusieurs capteurs en simultané provoquait des conflits entre les ondes de chaque capteur. En effet les ondes interfèrent entre elles. J'ai donc adapté le code de manière à ce que les capteurs se déclenchent l'un après l'autre. (le capteur 2 attends que le capteur 1 ait calculé sa distance). De ce fait il n'y avait plus de conflits. J'avais ainsi le module qui permettait à la voiture de se repérer dans l'espace.

Cependant si l’un des capteurs n’obtient pas de distance le programme s’arrête. Les capteurs sont disposés de manière à ce que la voiture choisisse la direction où elle a le plus de place pour évoluer sans rencontrer d’obstacle. La voiture possède deux moteurs à l’avant et une roue libre pivotant à 360° à l’arrière. Ainsi pour que la voiture tourne à gauche, le moteur de la roue droite s’active vers l’avant et pour que la voiture tourne à droite, le moteur de la roue gauche s’active vers l’avant. Notre voiture tourne donc de la même façon qu’un tank ou un petit avion au sol.

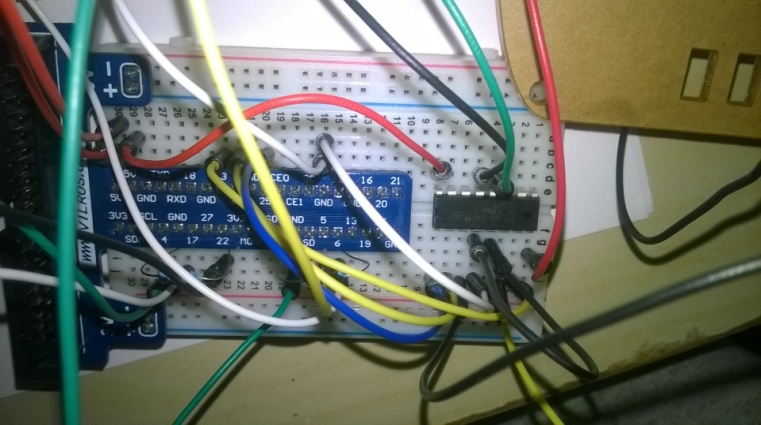


Prototype ayant pour but de définir l’emplacement des radars

Schéma simplifié de l’emplacement des capteurs

1. LES MOTEURS

Il a donc fallu trouver comment faire fonctionner un moteur dans un sens et dans l'autre. Pour cela une micro puce L293D était nécessaire pour inverser le sens du courant et ainsi le sens de rotation d’un moteur. Cette puce permet aussi de ne pas envoyer trop de voltage au Raspberry pi, ce qui pourrait provoquer des dommages électroniques.



Puce L293D

J'avais déjà intégré les bases du langage Python le code des moteurs s’est avéré au final relativement simple à élaborer. Pour faire tourner un moteur, on envoie du courant électrique issu de 4 piles AA (6 Volts) vers la puce L293D. On défini un port d’entrée et de sortie pour les GPIO. Si les deux broches de commande sont HIGH, ou les deux LOW, le moteur sera arrêté. Mais si une broche est HIGH et une autre en LOW, il tourne dans une direction, la direction est inversé si les valeurs des 2 broches sont inversées.

Cette fois ci j’ai rencontré des difficultés au niveau du hardware. En effet le câblage est complexe et l’erreur n’a pas sa place. La seconde difficulté à été de faire fonctionner 2 moteurs en même temps. Je ne pouvais pas ici utiliser la même méthode qu’avec les radars. Je me suis donc intéressé à la fonction multiprocess(), j’ai rapidement arrêté mes recherches dans ce domaine devant la complexité de la fonction. J’ai trouvé le moyen de faire tourner les moteurs l’un après l’autre sans arrêter le premier.

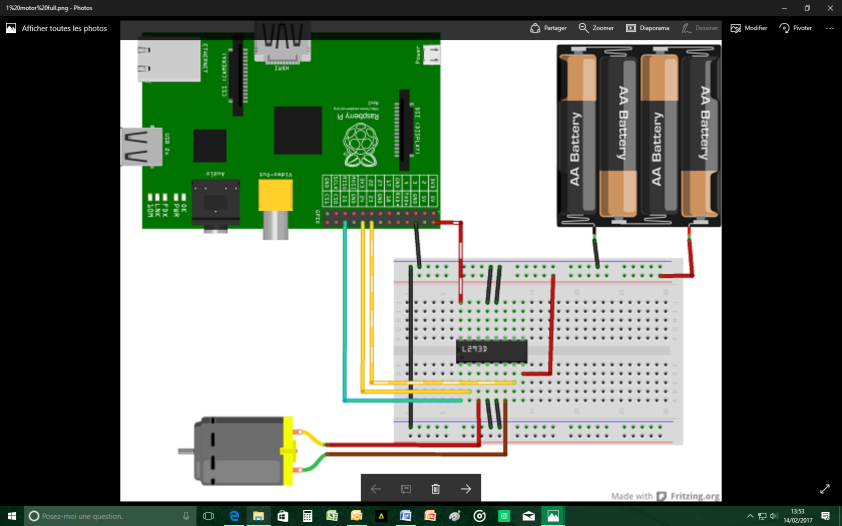


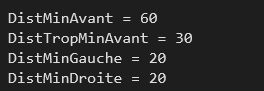
Schéma électronique pour le fonctionnement d’un moteur

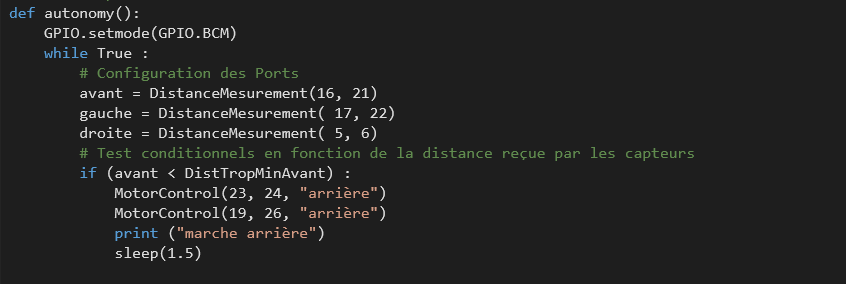
1. LE CODE AUTONOME

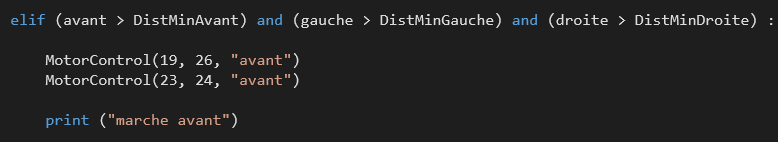
J'ai donc pu commencer à élaborer un programme commun entre les capteurs et les moteurs. J'ai rapidement définit un algorithme qui servirait de base à la structure du code et des cas de figure dans l'espace qui permettrait à la voiture de prendre des décisions seule. La fonction autonomy() se base sur un principe simple : Une boucle infinie initialise le code, il récupère ensuite les distances avant, droite, et gauche. L’une après l’autre dans un très court lapse de temps. Des tests conditionnels permettent à la machine de prendre des « décisions » dans l’utilisation des moteurs pour faire avancer, tourner, aller en arrière la voiture. J’ai travaillé en collaboration avec Matthis sur code. J’ai expliqué jusqu’au plus petit détail et au fur et mesure de l’avancement de la voiture les codes qui la contrôle. De telle sorte que lorsque nous avons fait le lien entre le machine et la page web tout le monde puisse participer même en ayant un domaine de prédilection.

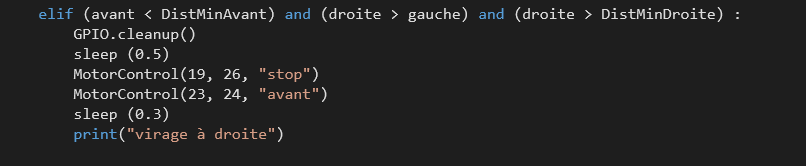
**Ci-dessous un moyen plus visuel de comprendre le code :**

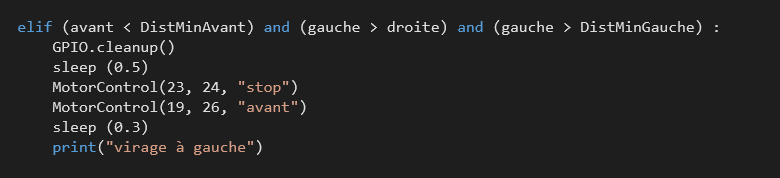
*On défini tout d’abord les distances minimales que la voiture ne devra pas dépasser.*











**Détail des fonctions du code autonome :**

**Import** >> importer une bibliothèque et les fonctions qui lui sont associées

**GPIO.setmode(GPIO.BCM)** >> permet de configurer la numérotation des ports du Raspberry Pi

**GPIO.setup()** >> permet de configurer un port du Raspberry Pi en entrée ou en sortie pour le courant

**GPIO.cleanup()** >> RPi.GPIO fournit une fonction intégrée GPIO.cleanup () pour « nettoyer » tous les **ports utilisés.**

**If () :** >> Test conditionnel qui se traduit en français pas « si ».

**and ()** >> « et »

**Elif () :** >> Test conditionnel qui suit un if() : qui se traduit par « sinon si »

**Try  :** >> « essayer »

**Except KeyboardInterrupt : >>** suit un « try : », il permet de sortir d’un programme à l’aide d’un « ctrl + c » au clavier

**while() :** >> une boucle « tant que », elle continue de tourner tant que la condition n’est pas vérifiée.

**Print ()** >> littéralement « afficher » du texte, un tableau, des chiffres.

**Sleep()** >> La méthode sleep () suspend l'exécution pour un nombre de secondes donné. L'argument peut être un nombre à virgule pour indiquer un temps plus précis.

**time.time()** >> enregistre le temps dans une valeur au moment de son utilisation

Nous avons travaillé sur le code autonome en groupe afin définir ce que nous voulions exactement. Une fois que le code autonome fut partiellement fonctionnel, Raphaël et Matthis se sont concentrés sur le moyen de contrôler le Raspberry Pi à distance.

L'idée de départ a servi de fil directeur, il fallait faire une page web en HTML avec une interface utilisateur en PHP.  Nous avons trouvé la solution en groupe de faire communiquer le Raspberry Pi avec la page web : Le langage python offre la possibilité de lire des fichiers textes (.txt) et de modifier leur contenu. Il en va de même pour le PHP. Nous nous sommes donc servis de cette fonctionnalité pour échanger des informations entre le Raspberry Pi avec la page web. Ci-dessous un schéma explicatif :

La dernière étape a été de tester encore et encore la voiture dans différents environnements afin de définir les distances minimales optimales pour une meilleure réactivité.

Nous avons progressivement abandonné l’idée de faire un code semi-autonome à cause de sa complexité et nous avons choisi de faire un code totalement autonome et un code totalement manuel.

J’ai ajouté des codes bonus qui permettent d’allumer des LED en fonction du côté vers lequel la voiture va tourner. Seulement ce code ralentissait la réactivité

1. BILAN PERSONNEL

Le projet et modulaire comme l’électronique, chaque partie de code peut être détaché et réutilisé pour avoir une autre fonction. Il a été pensé de manière a ce qu’il puisse être amélioré et modifié. De nombreuses améliorations peuvent encore être faites au niveau du matériel, la fluidité et de l’interface. Mais je suis fier du projet que j’ai mené au sein d’un groupe, voir le produit de son travail se mouvoir seul sous ses yeux est une sensation très satisfaisante.

Ce projet m’a permit d’en apprendre plus sur l’usage des outils numériques : l’environnement Linux, de différents éditeurs de code, de nouveaux langages (Python, HTML, PHP, CSS). J’ai fortement apprécié chercher par moi-même les toutes informations dont j’avais besoin, et de ce fait apprendre par moi-même. Dans ce groupe chacun avait son domaine de prédilection, c’est une expérience très stimulante de faire partager ses connaissances et recevoir celle des autres. J’avais toujours eu envie de faire un projet sur un Raspberry Pi et je me suis amuser à faire ce projet. Mais j’ai aussi découvert que l’informatique peut-être extrêmement frustrant, à cause des bugs, des solutions que l’on passe des jours à chercher pour enfin trouver la solution en 10 minutes. De plus l’électronique rajoutais une contrainte, ce n’est pas une science toujours exacte. Enfin ce projet m’a appris à être persévérant et à ne jamais abandonner face à une difficulté et de toujours trouver une solution. Je pense que cette expérience me servira beaucoup dans mes études même si je ne m’oriente pas vers l’informatique car le monde de demain sera inévitablement numérique.

SCHEMA BILAN