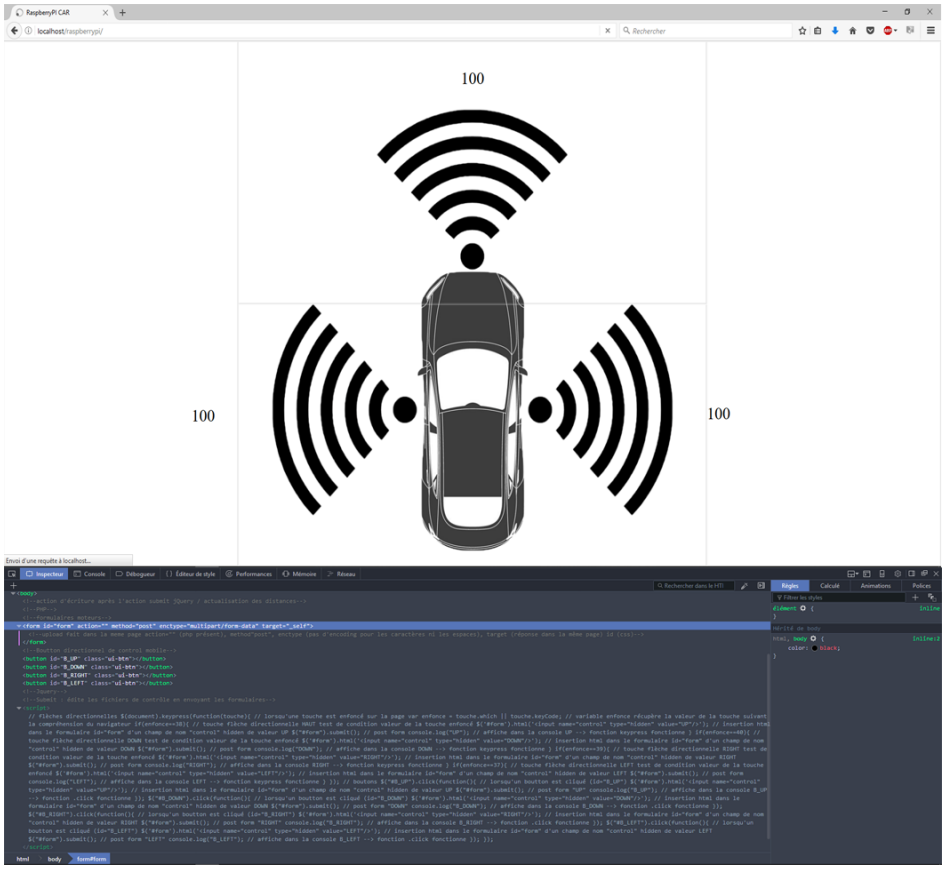
Raspberry-Pi Car



BOUFFETIER Thibault **CLAVIJO Matthis**

CLAVIJO Raphaël

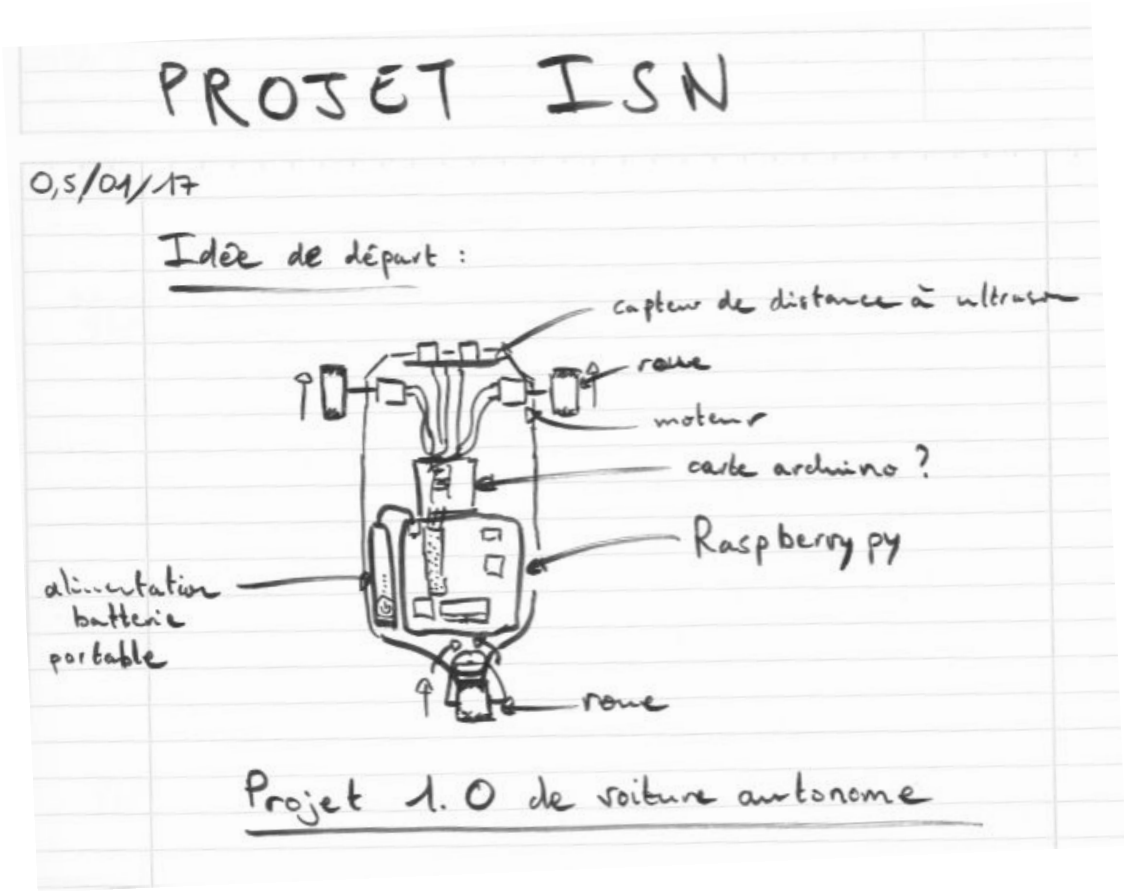
1) Présentation :

Le **Raspberry Pi Car** est un projet de voiture à partir d'un Raspberry Pi. Un Raspberry Pi est un micro-ordinateur de la taille d'une paume de main. Ce dernier propose différentes entrées et sorties. Il est alimenté de la même manière qu'un smartphone par un port femelle micro-USB type B. Ce qui permet par exemple de le raccorder à une batterie portable. Pour afficher la sortie vidéo il existe 2 moyens. On peut passer par un câble HDMI. Autrement on peut passer par le sans fil en ajoutant une clé usb wifi et en utilisant VNC viewer sur n'importe quelle plateforme (PC, tablette, téléphone). Le Raspberry Pi que nous utilisons a pour système d'exploitation Raspian. Ce dernier fournit des outils informatiques comme une boîte de commande linux, un editeur python, et des éditeurs d'autres langages de programmation. Le Raspberry Pi nous permet aussi d'héberger un serveur et les différents codes permettant le bon fonctionnement de la voiture. Ce projet contient deux modules. Le module d'une voiture autonome, la voiture est capable de rouler seule et de faire des choix pour éviter des obstacles fixes. Le Raspberry Pi contient également le module de la voiture manuelle. Un utilisateur à distance peut se connecter au serveur hébergé par le Raspberry Pi et contrôler la voiture à distance. L'utilisateur connaît en temps réel les distances des obstacles autour de la voiture.

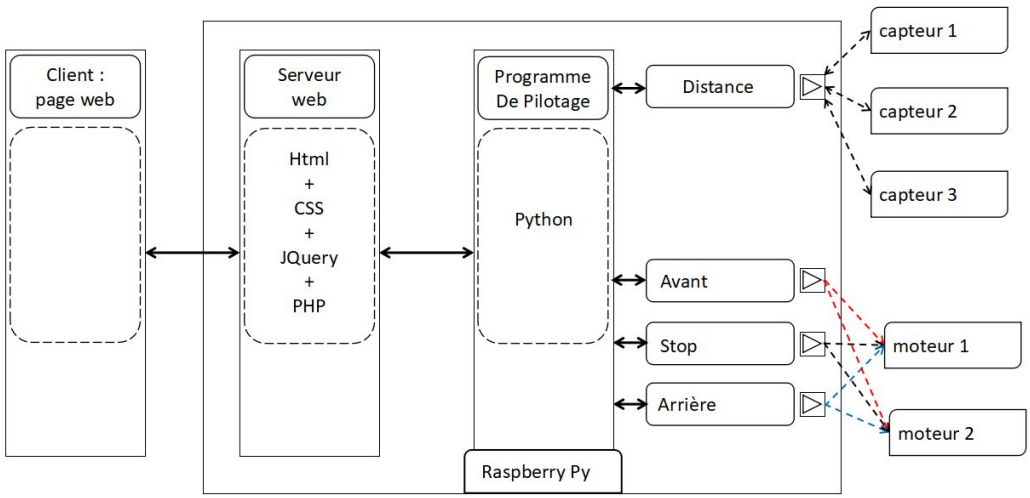
2) Analyse du besoin :

L'idée de départ était d'utiliser d'une manière originale le langage python et de ce fait faire un projet créatif. L'idée qui a rapidement émergée été celle d'une voiture semi-autonome contrôlable à partir d'un site web. Ce qui nous a poussé à choisir un projet alliant informatique et électronique est dû à notre attrait pour la technologie en général. Les voitures autonomes se développent de jours en jours avec de grands constructeurs comme Tesla, BMW, Mercedes. Le but intrinsèque de ce projet était donc de comprendre les bases du fonctionnement de tout code autonome.
Pour ce faire, nous avions prévu d'utiliser un raspberry pi pour héberger nos code python et qu'il puisse les lancer. Des projets de voiture autonome ont déjà été réalisés mais nous voulions créer notre propre projet et nous avons donc décidé de faire une voiture autonome et une voiture manuelle. Pour l'autonomie de la voiture, nous utilisons des émetteurs et des récepteurs à ultrasons. Cela nous permet de détecter des obstacles. Pour alimenter les moteurs des roues pour déplacer la voiture, nous utilisons l'énergie de piles AA. Grâce à une batterie portable à forte capacité, nous alimentons en énergie le raspberry pi.
De plus, nous voulions utiliser les connaissances que nous avions déjà acquises en python mais également apprendre de nouveaux langages et les faires interagir entre eux. Ainsi, nous avions l'ambition de mêler plusieurs langages de programmation et

les relier pour produire un projet cohérent alliant plusieurs modules divers autant en fonctionnalités qu'en méthodes de fonctionnement.



<u>Schéma des modules et des langages utilisés pour réaliser le projet :</u>



3) Répartition des tâches et démarche collaborative:

Hardware	Hardware-Software	Software
Thibault	Thibault	
		Raphaël
	Matthis	Matthis

Grâce à la plateforme Github, nous avons pu créer notre projet ISN avec tous les membres de notre groupe. Nous pouvions poster et mettre à jour nos codes sur la plateforme Github ce qui permettait à tous les membres de consulter les codes et l'avancement du projet. Nous avons donc pu travailler en équipe pendant nos leçons d'ISN, réfléchir sur l'adaptation du projet, informer les différents membres de l'équipe sur l'avancement des différents modules, en nous organisant et en se réunissant chez un membre de l'équipe. Nous avons également organisé des vidéos conférences pour travailler ensemble sur notre projet et nous concerter sur l'avancement de celui-ci.

Partie Personnelle # # # # # # # # # # # # # #

<h1> Présentation </h1>

Ma place lors de ce projet a été, avec Thibault, l'écriture de codes permettant à la voiture de se déplacer ou la modification de ces codes et également la mise en relation entre les codes de la voiture et ceux du serveur.
Raphaël, lui, était chargé du serveur. Nous avons décidé de créer trois codes différents.

<h2> Premier code </h2>

Le premier code développé a été créé pour obtenir une voiture autonome. Pour cela, la voiture nécessitait l'utilisation de capteurs à ultrasons pour connaître les distances entre la voiture et les obstacles. Nous disposions de trois capteurs à ultrasons. Tout d'abord nous avons codé pour le fonctionnement d'un seul capteur. Grâce à la relation physique permettant de connaître la distance en fonction de la vitesse et du temps, nous avons pu déterminer n'importe quelle distance entre un obstacle et la voiture. Ce code consiste à envoyer une onde depuis le capteur et à lancer un chronomètre au même moment. Lorsque l'onde revient au capteur, le chronomètre s'arrête et l'on obtient le temps. Ce temps est l'aller-retour de l'onde entre le capteur et un obstacle ; il est donc nécessaire de diviser ce temps par 2 pour obtenir le temps de l'aller. Sachant qu'une onde sonore se déplace à la vitesse de 340 mètres par seconde, nous avons toutes les données pour obtenir la distance entre le capteur et un obstacle. Puisque la vitesse est de 340 mètres par seconde, la mesure peut être qualifiée d'instantanée et nous avons donc la distance entre la voiture et l'obstacle en temps réel.

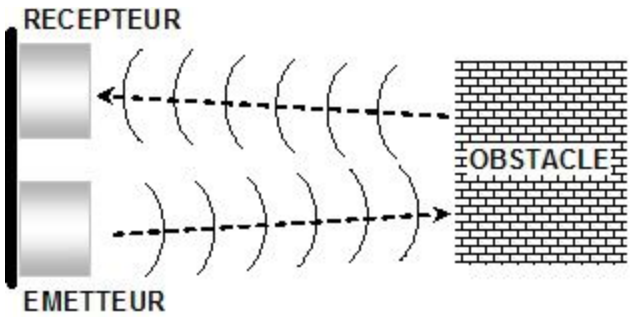


Schéma du fonctionnement d'un capteur

```
def DistanceMesurement(TRIG, ECHO) :


    GPIO.setmode(GPIO.BCM)
    # Configuration des Ports
    GPIO.setup(TRIG, GPIO.OUT)
    GPIO.setup(ECHO, GPIO.IN)

    # Initialisation broche Trig
    GPIO.output(TRIG, False)
    #print ("Waiting For Sensor 1 To Settle")
    time.sleep(0.1)

    time.sleep(0.1)
    # Impulsion de 10 micro secondes
    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)


    # Envoi impulsion et memorisation heure
    while GPIO.input(ECHO) == 0:
        pulse_start = time.time()


    # Attente reponse
    # memorisation heure
    while GPIO.input(ECHO) == 1:
        pulse_end = time.time()


    # Calcul de distance
    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration * 17150
    distance = round(distance, 2)
    return (distance)
```

Code du fonctionnement d'un capteur

Le code était fonctionnel et nous avons donc créé le même code pour les deux autres capteurs. Nous avons fait tourner les trois codes indépendamment mais ensemble les capteurs pouvaient recevoir les ondes des autres capteurs. Par conséquent les mesures étaient faussées et nous obtenions de mauvaises distances. Nous avons donc rassemblé les trois codes en un seul pour régler le problème. Les capteurs fonctionnent sous forme de boucle. Un capteur lance une onde et attend le retour de celle-ci puis un autre capteur fait de même. Comme cela, les capteurs ne peuvent recevoir les ondes des autres puisque les capteurs sont fonctionnels les uns après les autres.

```
while True :
    # Configuration des Ports
    avant = DistanceMesurement(16, 21)
    gauche = DistanceMesurement( 17, 22)
    droite = DistanceMesurement( 5, 6)
```

Code de fonctionnement des capteurs sous forme de cycle

Il y a un capteur à l'avant de la voiture et un capteur de chaque côté de la voiture. Il nous restait donc à développer des conditions pour que la voiture soit autonome à partir des données fournies par les capteurs. Lorsque la voiture rencontre un obstacle en face d'elle, elle regarde les distances à sa droite et à sa gauche et choisit de tourner dans la direction où la distance est la plus grande. Si la distance devant la voiture est trop courte, la voiture recule. Notre voiture possède deux roues avec moteurs et un galet à l'arrière de celle-ci. Cette disposition est comparable à la disposition des trains classiques chez les avions. 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 char.

```
if (avant < DistTropMinAvant) :
    MotorControl(23, 24, "arrière")
    MotorControl(19, 26, "arrière")
    print ("marche arrière")
    sleep(1.5)


elif (avant > DistMinAvant) and (gauche > DistMinGauche) and (droite > DistMinDroite) :

    MotorControl(19, 26, "avant")
    MotorControl(23, 24, "avant")

    print ("marche avant")

elif (avant < DistMinAvant) and (gauche > droite) and (gauche > DistMinGauche) :
    GPIO.cleanup()
    sleep (0.5)
    MotorControl(23, 24, "stop")
    MotorControl(19, 26, "avant")
    sleep (0.3)
    print("virage à gauche")

elif (avant < DistMinAvant) and (droite > gauche) and (droite > DistMinDroite) :
    GPIO.cleanup()
    sleep (0.5)
    MotorControl(19, 26, "stop")
    MotorControl(23, 24, "avant")
    sleep (0.3)
    print("virage à droite")

|

elif (avant < DistMinAvant) and (gauche < DistMinGauche) and (droite < DistMinDroite) :

    MotorControl(23, 24, "stop")
    MotorControl(19, 26, "stop")
    print ("arrêt des moteurs")
    print ("opération manuelle requise")
```

Condition pour que la voiture effectue une décision

<h2> Deuxième code </h2>

Nous avons ensuite écrit un code pour pouvoir contrôler manuellement la voiture à distance. Ce code fonctionne comme celui de la voiture autonome. La différence est bien sûr que celui-ci n'est pas autonome, c'est-à-dire qu'il n'y a pas de conditions pour que la voiture prenne des décisions. Pour que le client puisse contrôler la voiture à distance, nous avons installé un serveur apache2 avec une relation php sur la voiture (raspberry pi). Il y a une relation client-serveur en sous-réseau (sur un même réseau) permettant des échanges d'informations entre les deux. Le client contrôle la voiture grâce aux flèches directionnelles de son clavier ou en cliquant sur des zones de l'écran sur smartphone. Pour plus de précisions sur le serveur, je vous invite à consulter le dossier de mon collègue Raphaël Clavijo. J'ai effectué le lien entre les requêtes client et le code pour diriger la voiture et le lien pour que les distances des capteurs s'affichent chez le client. Je n'ai pu bien sûr le faire tout seul ne connaissant pas les requêtes du client au serveur. Je me suis donc concerté avec Raphaël pour trouver un moyen pour effectuer ce lien. Après l'explication mutuelle de nos codes, je lui ai proposé d'écrire dans des fichiers texte les distances des capteurs et les requêtes du client. La page web affiche les distances des capteurs en temps réel grâce à l'écriture en temps réel des distances dans ces fichiers. Les requêtes sont également traitées en temps réel grâce à la lecture du fichier requêtes par mon code. Grâce à ce lien, le serveur et le client peuvent échanger en temps réel des informations et le code de la voiture les traites en temps réel. Le client peut ainsi contrôler la voiture manuellement.

```
def distances(Capteuravant,Capteurgauche,Capteurdroite):

    Capteuravant=str(Capteuravant)
    Capteurgauche=str(Capteurgauche)
    Capteurdroite=str(Capteurdroite)
    fichier = open("/var/www/RaspberryPI/capteurs/avant.txt", "w")
    fichier.write(Capteuravant)
    fichier.close()
    fichier = open("/var/www/RaspberryPI/capteurs/gauche.txt", "w")
    fichier.write(Capteurgauche)
    fichier.close()
    fichier = open("/var/www/RaspberryPI/capteurs/droite.txt", "w")
    fichier.write(Capteurdroite)
    fichier.close()
```

Code permettant la liaison client-serveur

<h2> Troisième code </h2>

Le code de ce troisième module n'a pas fonctionné. Ce code mêlait l'autonomie de la voiture et les commandes manuelles du client. Ce code devait rendre la voiture semi-autonome, c'est à dire que lorsque le client conduisait la voiture trop près d'un obstacle, la voiture devait s'arrêter puis faire un choix autonome. En théorie ce code devait être simple puisque que les deux parties du code (autonomie et manuelle) existaient déjà. Quelques ajustements et le tour était joué mais un problème majeur est survenu. En effet, le Raspberry Pi n'était pas assez rapide pour détecter un obstacle pendant le pilotage de la voiture et ne pouvait donc réagir comme il aurait dû. Ce code n'a donc pas été validé par notre groupe.

<h2> Intégration et validation</h2>

Comme Thibault et moi avons développé les codes de manière indépendante, nous avons dû beaucoup échanger et effectuer des séries de tests pour analyser les parties des codes à améliorer ou à optimiser pour un meilleur rendu final.

Les premiers tests effectués ont été les tests pour savoir si la voiture réagissait bien face à un obstacle. Nous avons pu constater lors de ces tests que la voiture ne s'arrêtait pas face à un objet. Ce problème était dû à des distances de sécurité choisies arbitrairement mais trop courtes pour que le raspberry pi puisse réagir à temps, avant la collision. Nous avons donc revu nos distances de sécurité et optimisé notre code pour que le raspberry pi puisse réagir un tout petit peu plus vite. Nous avons donc réalisé encore des séries de tests pour déterminer la distance parfaite pour que la voiture puisse réagir et tourner sans problèmes.

Nous avons ensuite réalisé des tests pour voir si l'utilisateur pouvait correspondre avec la voiture. Il n'y a pas eu de problème pour la correspondance entre les deux mais le temps de latence était énorme entre les ordres de l'utilisateur et la réalisation de ceux-ci. Raphaël a donc optimisé son code pour que le temps de latence soit réduit. Nous avons également fait des séries de tests avec des temps de rafraîchissement de la page différents. Pourquoi ? Car le but est que l'utilisateur puisse avoir le temps d'envoyer un ordre mais également, que lorsque la voiture ne reçoit plus d'ordre, elle s'arrête aussitôt. Il faut donc que le temps de latence entre la fin de l'ordre et l'arrêt de la voiture ne soit pas trop grand de manière à ce que l'utilisateur puisse arrêter la voiture où il veut.

Ces séries de tests ont été essentielles au sein de notre projet car elles nous ont permis de finaliser totalement notre projet.

<h1> Bilan du Projet et Perspectives</h1>

Notre projet au tout début de sa conception, était de réaliser une voiture plus qu' autonome, une voiture semi-autonome. Puis après réflexion, nous nous sommes dit que pour ce projet nous allions devoir coder une partie autonome et une partie manuelle. Nous avons donc, à ce moment là, décidé de rajouter ces deux autres modules à la voiture. L'utilisateur ou, si l'on voit plus loin, l'entreprise commercialisant cette voiture pourrait décider de mettre en vente ces trois types de voitures différentes. Cependant, le dernier module, la voiture semi-autonome s'est heurté à un problème de capacité de notre modèle de Raspberry pi et donc nous ne possédons que les deux autres modules (autonome et manuel).

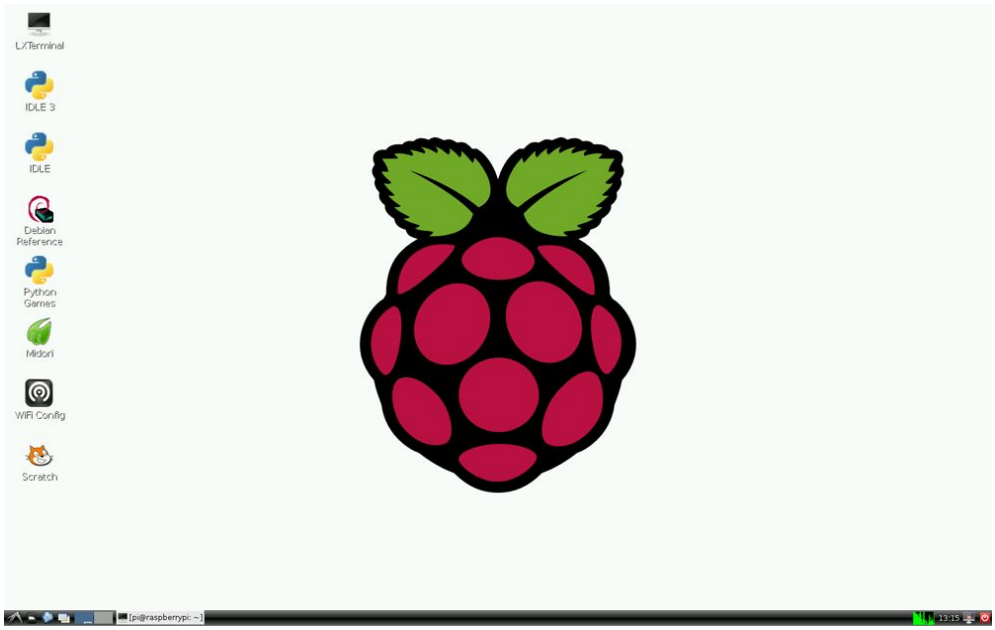
Ce projet m'a apporté énormément. Tout d'abord en travail de groupe puisque cela m'a appris à travailler à plusieurs sur un projet de longue durée.

Cela m'a également appris la communication entre les personnes au sein du groupe, indispensable puisque nous avions chacun des responsabilités différentes.
Le projet apprend aussi à s'organiser ensemble et avancer en parallèle pour la bonne réalisation du projet.

Ce projet m'a aussi appris à me confronter à des problèmes comme celui de faire tourner un code dont le résultat physique (électronique) attendu n'est pas celui obtenu.

Techniquement, ce projet m'a également appris à mettre en œuvre du code python, à le tester et à m'adapter à un nouvel environnement unix.

Avec un Raspberry pi plus puissant, le module semi-autonome pourrait être retravaillé.



Annexes

Code autonome:
https://github.com/ecolelasource92/Raspberry-Pi_Car/blob/master/MODULES/RADAR_MOTOR%20main.py

Code manuel:
https://github.com/ecolelasource92/Raspberry-Pi_Car/blob/master/MODULES/MANUAL.py
Et
https://github.com/ecolelasource92/Raspberry-Pi_Car/blob/master/MODULES/MANUAL2.py

Code serveur:
https://github.com/ecolelasource92/Raspberry-Pi_Car/blob/master/SERVEUR/index.php
Et
https://github.com/ecolelasource92/Raspberry-Pi_Car/blob/master/SERVEUR/Principal/index.php

Bases de données:
https://github.com/ecolelasource92/Raspberry-Pi_Car/tree/master/SERVEUR/capteurs
Et
https://github.com/ecolelasource92/Raspberry-Pi_Car/tree/master/SERVEUR/moteur