Sciences et technologies

Avenue V. Maistriau 8a
B-7000 Mons
+32 (0)65 33 81 54
scitech-mons@heh.be

WWW.HEH.BE

Rapport de gestion de projets.

Bachelier en informatique et systèmes – Finalité Télécommunications et réseaux – Milieu de cycle.

Année Académique 2022 – 2023
Groupe N°9
Rosin Guillaume
Danneau Thibaud
Yepgwa Takedo Fabiola

Table des matières

Tables des illustrations :

Introduction

Durant la semaine de projet, nous avons configuré une voiture autonome à l'aide d'un Raspberry Pi 3 B+ ainsi que les composants suivants qui nous ont été fournis par notre établissement scolaire :

Un module (voiture à 4 roues) équipé d'un Raspberry Pi 3 Model B+

- 3 capteurs Ultrasons (2 sur les côtés et 1 à l'avant).
- 1 capteur infrarouge situé à l'avant en-dessous du module.
- 1 capteur RGB situé à l'avant en dessus du module.
- 1 servo moteur et 2 moteurs à courant continus.
- 1 Driver Moteur L298N
- 4 piles rechargeables ainsi qu'un chargeur.

Notre équipe était composé de trois personnes en option développement et une personne en option sécurité. Mais à la suite d'un abandon nous nous sommes retrouvés à 3 pour réaliser le projet.

- Thibaud Danneau et Fabiola Yepgwa Takedo, étudiant en option développement.
- Rosin Guillaume étudiant en option sécurité.

Lors de cette semaine nous avons été mis en condition réelle. En effet, nous avons réalisé une voiture autonome. Pour nous mettre dans cette situation, nos professeurs ont interprété le rôle de clients et nous ont distribué un cahier des charges ainsi qu'un planning pour communiquer avec le client.

Lorsque nous avons rencontrés des problèmes que ce soit l'abandon d'un des membres du groupe, sur le matériel ou encore les erreurs dans le code, nous avons pu compter entièrement sur la bienveillance et l'aide de nos professeurs.

La méthode de travail que nous avons utilisé est la méthode Scrum, c'est une méthode agile de gestion de projets orientée informatiques privilégiant la communication et facilitant les réorientations opportunes.

Nous avons respecté 5 concepts de la méthode Scrum pour organiser notre travail :

- Nous avons défini un Scrum master : le rôle de Scrum master est un chef de projet qui est capable de faire sortir le meilleur de chacun pour réussir le projet. Il s'assure que le principe Scrum se déroule comme il se doit, il fixe les rôles, les timings et les objectifs. C'est un statut complexe à gérer, il faut être rationnel et communiquer avec l'ensemble de son équipe.

- Product Owner : c'est la personne qui va partager la réalisation du produit à réaliser avec l'équipe de développement. C'est le responsable de la bonne exécution du projet, il est en contact avec l'équipe de développement, le marketing et les clients.

- Scrum Board : c'est le tableau de bord du projet, il est disponible pour tous les membres de l'équipe. Il permet de suivre l'état d'avancement du projet. Les tâches y sont indiquées et qualifiées : à faire, en cours, en test et terminées.

- Le sprint : c'est la phase essentielle de développement du produit. Le but est de réaliser et de présenter un état d'avancement du projet au client.

- L'équipe : elle doit être autoorganisée et pluridisciplinaires.

1. Présentation

Comme nous l'avons dit précédemment, en début de semaine, nous avons reçu sous forme d'un cahier des charges, les objectifs de la semaine. Ils consistaient à développer un module mobile autonome dans le but de réaliser une course de formule 1 sur un circuit réalisé par M. Pietrzak.

Figure 1 - Photo du circuit

Le module devait donc être capable d'effectuer certaines tâches données par le cahier des charges :

- Être capable d'effectuer un tour complet du circuit.
- De s'arrêter une fois qu'il a franchi la ligne d'arrivée.
- Être capable d'éviter tout obstacle.
- Lui spécifier le nombre de tours à réaliser pour la course.
- De démarrer automatiquement lorsque le feu de signalisation devient vert.
- Il faudra évidemment réaliser du calibrage pour « optimiser » la trajectoire au maximum afin de gagner la course.

2. Description des composants

2.1. *Le Raspberry PI*

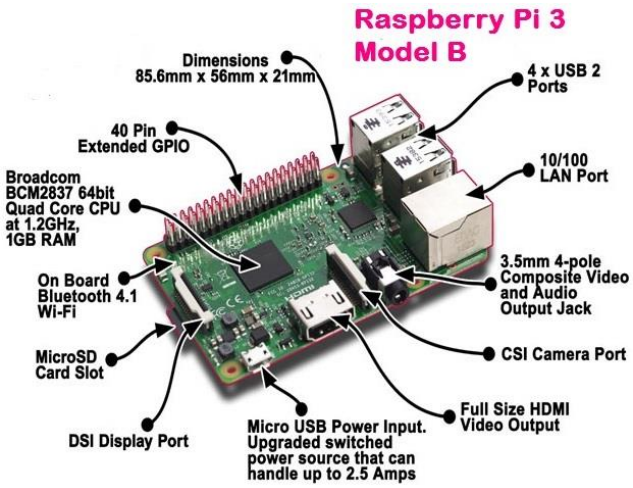

Figure 2 - binaryupdates.com - introduction-of-rasberry-pi-3-model-b

L'élément central et indispensable au bon fonctionnement de notre module est le Raspberry Pi 3 modèle B+, c'est grâce à ce dernier qu'elle pourra avancer, reculer, communiquer avec les différents capteurs et ainsi gérer les obstacles, la ligne d'arrivée ou le feu de signalisation via les différents scripts sauvegardés sur sa carte SD.

Nous avons déjà réalisé un premier rapport sur le Raspberry PI modèle 3 mais nous allons repasser rapidement en revue celui-ci afin de vous expliquer un peu plus en détails de quoi il est composé.

Il est équipé de 4 ports USB, un port HDMI, un connecteur d'E/S (entrée/sortie), d'un GPIO de 40 broches et d'un port Micro-SD. Pour le faire fonctionner, rien de bien compliqué, il nécessite une alimentation et une carte micro-SD munie d'un système d'exploitation (bootable).

Par rapport à notre projet, voici une explication générale du fonctionnement du Raspberry PI :

- Le Raspberry PI est alimenté par une batterie externe (Pi Juice) branché directement sur les broches du GPIO.
- Le système d'exploitation, les différentes librairies installées, les programmes installés et les scripts en python nécessaires au déroulement du projet et au pilotage des composants de la voiture sont stockés sur la carte micro SD. Le Raspberry ne possède pas de stockage intégré !
- Nous avons réalisé un point d'accès avec le Raspberry afin que nous puissions nous connecter à distance et le configurer avec plusieurs appareils. Cela nous a permis de réaliser des tests individuellement. Grâce à son Antenne WIFI et à sa haute permittivité il peut émettre et recevoir des données via la fréquence de bande 2.4 GHz tout en prenant un minimum de place sur le PCB.

- Les pins d'entrées et sorties du GPIO nous ont permis de connecter plusieurs types de composants à notre Raspberry Pi. Sur le Raspberry Pi 3 modèle B+ on y distingue 40 broches (26 broches étant utilisés comme entrées ou sorties numériques).

2.2. Les capteurs Ultrasons

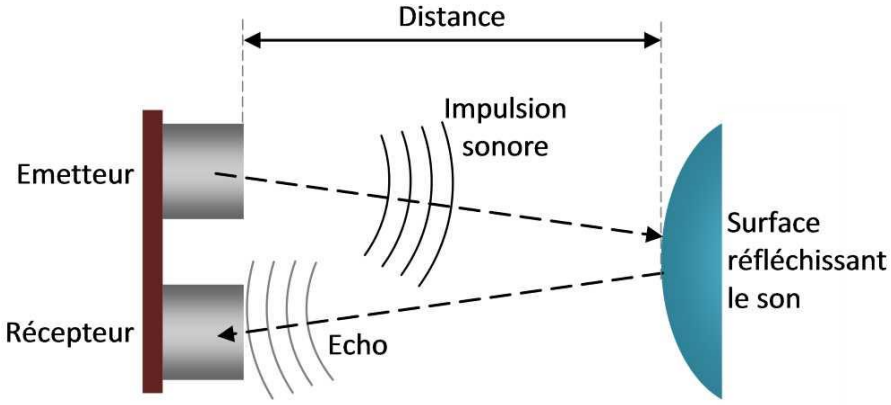

Figure 3 - arduino.blaisepascal.fr - capteur-de-distance-a-ultrasons

Au nombre de trois sur notre module, ils nous ont permis de récupérer les valeurs des distances se situant en face, à gauche et à droite du véhicule afin d'éviter les obstacles, de longer les murs et éventuellement de reculer pour réaliser une manœuvre.

Leur principe est simple, ils émettent à intervalles régulier de courtes impulsions ultrasoniques (autrement dit impulsions sonores) à haute fréquence. Elles se déplacent ainsi dans l'air à la vitesse du son sous forme d'écho au capteur. Ensuite, ils calculent la distance séparant l'objet du capteur en fonction du temps écoulé entre l'émission et la réception de l'impulsion.

2.3. Le capteur infrarouge

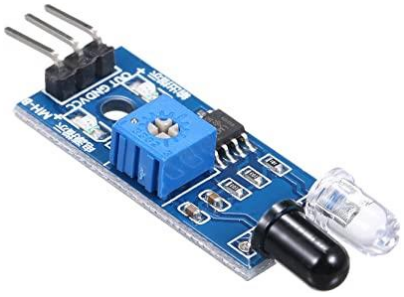

Figure 4 - SHUAIGUO Capteur de proximité Infrarouge IR Module de capteur d'évitement d'obstacle Infrarouge Compatible avec Arduino

Le fonctionnement de ce capteur est basé sur l'émission d'un faisceau de lumière infrarouge par l'émetteur intégré au capteur (dôme bleu), ce faisceau est ensuite réfléchi et est renvoyé sur le récepteur (dôme noir).

Lorsqu'un objet est présent dans la zone de détection du capteur la lumière infrarouge émise par l'émetteur est réfléchie par la surface de l'objet et renvoyée vers le récepteur. Le récepteur détecte cette lumière réfléchie et envoie un signal électrique au circuit de contrôle du capteur.

Le capteur TCRT5000 est constitué d'une LED infrarouge et d'un phototransistor sensible à la lumière. Les données sont transmises via deux broches, l'une fournissant une lecture continue via la broche analogique A0. La tension de sortie augmente avec la quantité de lumière infrarouge reçue. Cependant, le capteur peut être influencé par l'environnement et la source lumineuse peut fausser les mesures. Dans ce projet, nous utiliserons le TCRT5000 pour détecter la présence d'objets physiques tels que des virages et des murs, et pour identifier des couleurs allant du noir à l'uni. Comme pour une course, le capteur peut être utilisé pour suivre une ligne.

2.4. Le servo-moteur

Un servo-moteur est un type de moteur électrique qui est utilisé pour contrôler la position et la vitesse de placement de nos roues avec précision. Il maintiendra cette position jusqu'à l'arrivée d'une nouvelle instruction. Les instructions se font à l'aide d'impulsions électriques.

Le fonctionnement d'un servo-moteur est basé sur un système de boucles de rétroaction qui permettent d'ajuster la sortie du moteur. Le capteur de boucles de rétroaction se compose généralement d'un capteur de position, d'un microcontrôleur et d'un circuit de commande.

Pour expliquer le principe, lorsqu'on envoie une commande à notre servo-moteur, le microcontrôleur mesure la position actuelle de l'objet contrôlé à l'aide du capteur de position. Ensuite, il compare cette valeur à la valeur fournie par la commande et calcule la différence. Cette différence est utilisée pour ajuster la sortie du moteur jusqu'à ce que l'objet atteigne la position désirée.

Vous constaterez que chaque roue est positionnée sur un axe de rotation (partie bleu), qui est lui-même fixé sur un pivot sur le châssis de la voiture (partie en vert). Ainsi la « baguette » (partie rouge du schéma) permet de garder le parallélisme entre les roues. Sur cette baguette il est fixé également le bras de notre servo moteur qui permet de faire pivoter nos roues et permettre ainsi à la voiture de prendre une direction. Vous l'aurez compris le servo-moteur entre la baguette pour orienter nos roues.

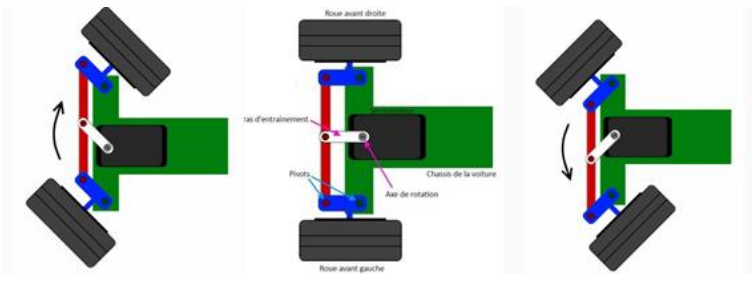

Figure 5 - eskimon.fr/tuto-arduino-602-un-moteur-qui-a-de-la-tete-le-servomoteur

2.5. Le capteur de couleur – TCS3472

Le capteur TCS34725 permet à notre module de détecter la couleur du feu de signalisation mis en place sur le circuit. L'objectif étant que notre module démarre à l'affichage de la couleur verte. Le fonctionnement de ce capteur est simple, le composant fournit un retour numérique des valeurs de détection de lumière rouge, verte, bleue et de lumière claire. Un filtre intégré directement sur la puce et situé sur les photodiodes de détection de couleur minimise la composante spectrale infrarouge de la lumière entrante et permet de prendre des mesures de couleur avec précision. Ces données sont transférées via un I²C à l'hôte.

Figure 6 - www.esp8266learning.com/esp8266-tcs34725-color-sensor.php

2.6. Driver Moteur – L298N

Ce composant va nous permettre de changer le signal de base lorsqu'il va passer sur la ligne d'arrivée, à nous de faire un compteur qui va s'incrémenter quand il va passer sur la ligne et changer d'état.

Le L298N possède deux canaux de sortie pour contrôler deux moteurs distincts. Chaque canal est composé de deux transistors H-bridge, qui permettent de contrôler la direction et la vitesse du moteur.

Pour contrôler un moteur, il faut appliquer une tension d'alimentation sur les bornes d'entrée du canal correspondant, ainsi qu'un signal de commande pour définir la direction de rotation et la vitesse du moteur.

Le signal de commande est généralement généré par un microcontrôleur ou un circuit de commande dédié. Il est constitué de deux signaux binaires, un pour la direction de rotation et un pour la vitesse. Le signal de direction permet de choisir le sens de rotation du moteur, tandis que le signal de vitesse permet de réguler la vitesse du moteur en modulant la largeur d'impulsion des signaux.

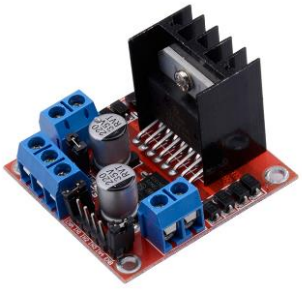

Figure 7 - www.eagle-robotics.com/accueil/162-l298n-driver-moteur

3. Programmation et structure des fichiers

3.1. Classe « drive »

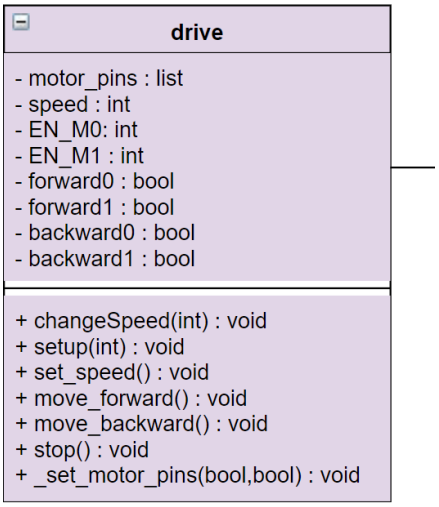

Figure 8 - Classe "Drive"

C'est la classe où sont définies toutes les méthodes pour faire fonctionner les moteurs à courant continu et donc les roues arrière du module. L'ensemble des algorithmes étant codés en orienté objet, la première "méthode" de la classe est « motor_pins » c'est le constructeur où sont l'ensemble des pins de sortie en mode BCM (c'est le mode que nous utilisons pour toutes les classes nécessitant des pins GPIO). Ensuite, nous avons la variable de vitesse « speed » qui est spécifiée à l'instanciation de la classe, les variables des pins d'entrée EN_M0 et EN_M1 pour la mise en marche de chacune des roues et les variables « forward » et « backward » pour le sens de chaque roue respectivement en avant et en arrière.

Ces différentes variables sont utilisées par les autres méthodes de la classe drive : « changeSpeed » permet de modifier la vitesse définie à la création d'un objet drive. « Setup » est la méthode qui permet d'initialiser le mode des pins en BCM et de créer l'objet de type PWM pour initialiser la fréquence des moteurs grâce à sa méthode « frequency ». La méthode « setup » est appelée dans le constructeur pour ne pas avoir à le faire après la création d'un objet drive. La méthode « set_Speed » permet d'initialiser la vitesse des roues récupérées chez le constructeur, donc à l'instanciation de la classe. Les méthodes « move_forward » et « move_backward » permettent respectivement de faire avancer les roues vers l'avant et vers l'arrière en utilisant les variables évoquées ci-dessus. La méthode « stop » permet d'arrêter les moteurs en mettant les pins de sorties à l'état « low » et la méthode « _set_motor_pins » permet de changer l'état des pins de sortie en fonction des valeurs des variables « forward » et « backward », l'un des pins de chaque moteur est en état « high » quand l'autre est en « low » ce qui conditionne le fait que les roues tournent en avant ou en arrière. Cette dernière est utilisée par les méthodes « move_forward » et « backward ».

3.2. *Classe « voiture »*

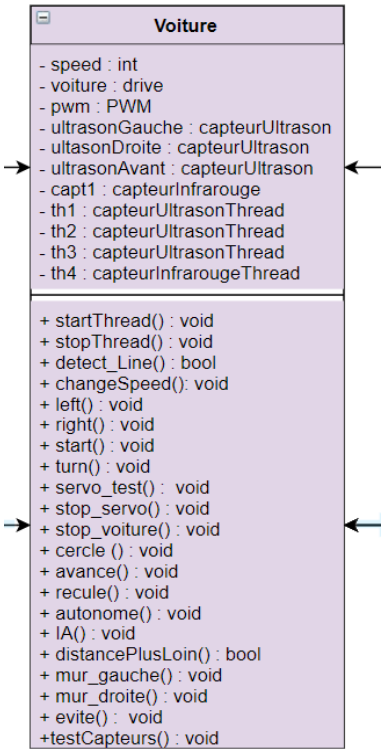

Figure 9 - Classe "Voiture"

Dans cette classe sont implémentées toutes les fonctionnalités pour le déplacement du module. Son constructeur fait intervenir une variable “speed” similaire à celle de la classe drive vue précédemment puisqu’elle va être utilisée dans l’instanciation de l’objet voiture de type drive utilisé pour une association des deux classes en vue d’utiliser des méthodes de la classe drive pour la définition de méthodes de la classe voiture.

On distingue également la variable « pwm » de type PWM qui va servir pour la communication avec le servomoteur pour la définition des différents angles de rotation des roues avant.

Les variables ultrason et capt1 sont des objets de type « capteurUltrason » et « capteurInfrarouge », classes que nous verrons plus bas ; ces objets permettent de communiquer avec les différents capteurs et d’en récupérer les valeurs. Les variables th1 à 4 sont des variables des classes de thread nécessaires pour recevoir les valeurs des capteurs simultanément.

Concernant les méthodes de cette classe, les méthodes « startThread » et « stopThread » permettent de mettre en marche et arrêter le fonctionnement des threads pour les capteurs, la méthode « detect_line » comme son nom l’indique est la méthode qui utilise l’objet faisant appel au capteur infrarouge pour détecter la ligne sombre qui est la ligne d’arrivée. La méthode « changeSpeed » fait appel

à la méthode du même nom de la classe drive pour modifier la vitesse définie à l'instanciation de la classe.

Les méthodes « left, right, start et turn » sont les méthodes qui permettent d'ajuster l'angle de rotation des roues avant : les méthodes « left, right et start » sont prédéfinies et permettent respectivement de tourner à gauche, à droite et de revenir à la position d'origine selon un angle constant tandis que la méthode « turn » prend en paramètre une valeur entière qui permet la rotation à l'angle voulu. La méthode « sevo_test » est une méthode test qui a été définie pour vérifier le fonctionnement des méthodes précédentes.

Les méthodes « stop_servo » et « stop_voiture » sont les méthodes qui permettent de mettre en arrêt les moteurs du module : la première permet uniquement d'arrêter le servomoteur et donc les roues avant et la seconde fait appel à « stop_servo » et à la méthode de l'objet voiture de type drive pour stopper les moteurs à courant continu en plus du servomoteur.

La méthode cercle a été définie pour effectuer un cercle dans le sens horlogique et anti-horlogique, l'une des tâches du premier sprint. Les méthodes avance et recule permettent de faire tourner les roues vers l'avant ou vers l'arrière en utilisant les méthodes « move_forward, move_backward et setSpeed » de l'objet voiture.

Autonome et IA sont les méthodes qui ont été mises en place pour effectuer le tour du circuit, une autre des tâches de notre sprint 2. La méthode autonome n'étant pas opérationnelle, nous avons implémenté la méthode IA qui utilise le résultat de la méthode « distance_plus_loin » pour comparer la distance la moins élevée parmi celles renvoyées par chacun des capteurs entre le module et un obstacle et ainsi déterminer dans quelle direction tourner tout en ralentissant pour une distance trop proche renvoyée par le capteur avant.

La méthode « distance_plus_loin » en question compare les distances renvoyées par les capteurs de gauche et de droite et renvoie un booléen spécifique en fonction de la distance la plus élevée : « True » pour la distance à droite et False pour la distance à gauche.

Les méthodes « mur_gauche » et « mur_droite » sont les méthodes permettant de longer un mur en mettant à profit les méthodes « turn », avance et « stop_voiture ». La méthode « evite » est la méthode qui permet d'éviter les obstacles en fonction des données des capteurs et « testCapteurs » est la méthode utilisée pour vérifier que les valeurs des capteurs sont correctes à des fins de débogage.

3.3. Classe « PWM »

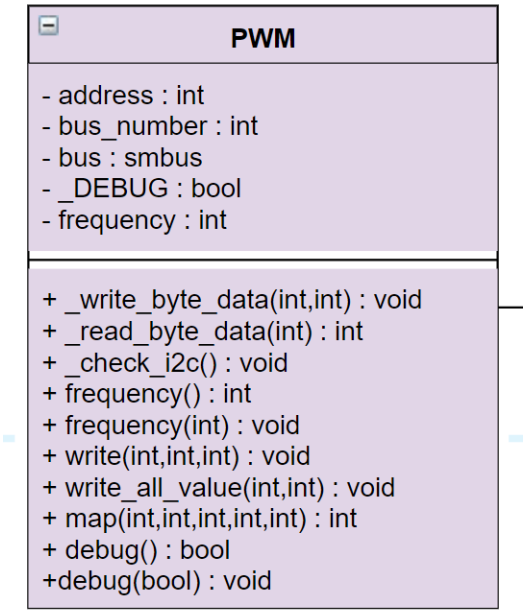

Figure 10 - Classe "PWM"

Cette classe est la librairie utilisée par les classes drive et voiture précédentes. Elle est importée dans ces différentes classes pour l'utilisation de ses méthodes. Pour ce faire, un objet « pwm » de type PWM est créé dans le constructeur de la classe voiture et dans la méthode setup de la classe drive.

Dans la classe drive, la classe PWM est instanciée dans la méthode setup pour avoir accès à sa méthode ''frequency'' et initialiser la fréquence des moteurs ainsi que la méthode ''write'' pour initialiser la vitesse récupérée par le constructeur aux pins d'entrée des deux moteurs à courant continu.

Dans la classe voiture, l'objet « pwm » est créé directement dans le constructeur puisqu'il est utilisé par plusieurs méthodes de la classe ; il est principalement présent pour avoir accès à la méthode « write » qui va permettre de définir le degré de rotation des roues contrôlées par le servomoteur.

3.4. Classes « capteurs »

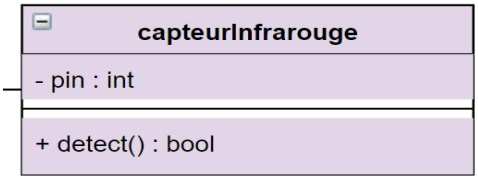

Figure 11 – Classe "Capteur Infrarouge"

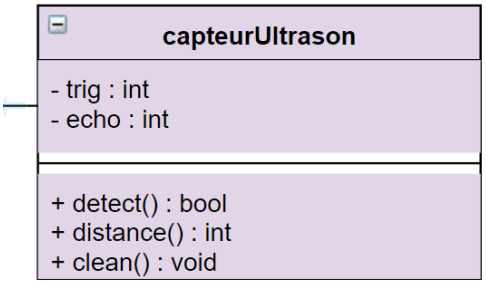

Figure 12 – Classe "Capteur Ultrason"

Les classes « capteurInfrarouge » et « capteurUltrason » sont les classes qui vont permettre au module et ses moteurs de communiquer avec les capteurs. Chacune de ces classes contient une méthode qui renvoie les données reçues par les capteurs en question. « capteurInfrarouge » prend comme paramètre la variable pin qui est définie dans le constructeur.

3.5. Classes « threads »

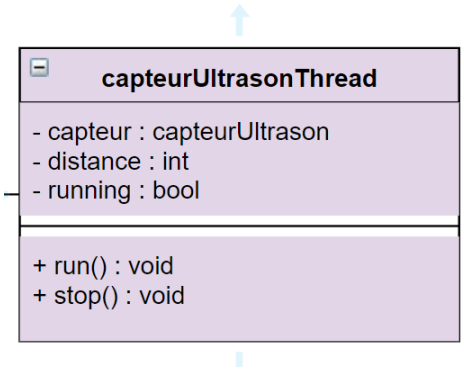

Figure 13 - Thread "Capteur Ultrason"

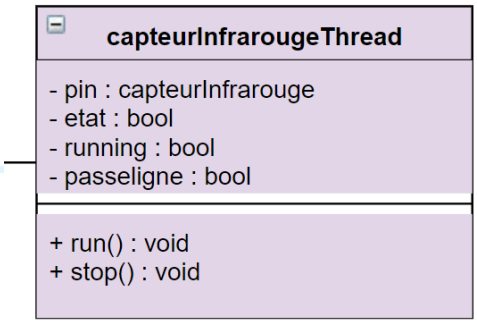

Figure 14 - Thread "Capteur infrarouge"

Les classes capteurUltrasonThread et capteurInfrarougeThread sont des classes qui vont permettre de gérer le multithreading. Le multithreading permet de réaliser plusieurs tâches en simultané, ce qui est nécessaire pour que les capteurs fonctionnent tous en même temps.

Les deux capteurs héritent de la classe Thread, c'est une classe qui vient de la bibliothèque threading.

Le capteurUltrasonThread prend en paramètre dans son constructeur un objet capteurUltrason. Cette classe sert à mesurer la distance à l'aide d'un capteur à ultrason et stock cette distance dans une variable "distance" de la classe capteurUltrason. La méthode "run" contient une boucle qui s'exécute tant que la méthode "stop" n'est pas appelée. Dans cette boucle, la méthode "distance" du

capteur est appelée et la distance est stockée dans une liste de maximum 20 éléments, la liste est de type file (c'est-à-dire que les derniers éléments ajoutés sont supprimés pour accueillir les nouveaux). La moyenne de la liste est ensuite calculée et stockée dans la variable "distance".

Le capteurInfrarougeThread prend en paramètre dans son constructeur un objet capteurInfrarouge. Cette classe sert à détecter la présence d'une ligne à l'aide d'un capteur infrarouge et à stocker cette information dans une variable "etat". La méthode "run" contient une boucle qui s'exécute tant que la méthode "stop" n'est pas appelée. Dans cette boucle, la méthode "detect" du capteur est appelée pour vérifier la présence de la ligne d'arrivé. Si la ligne d'arrivée est détectée, la variable "etat" passe à True, sinon elle est mise à False. La variable "passeligne" est également mise à jour.

3.6. *Classe « captCourant »*

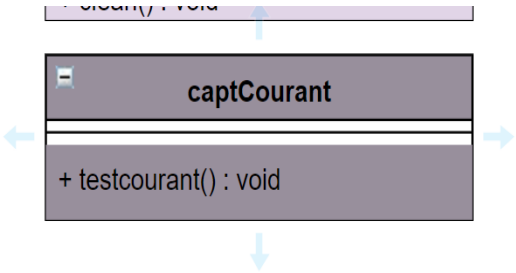

Figure 15 - Classe "captCourant"

L'une des tâches explicitement demandée pour la réalisation de ce projet était de surveiller l'évolution du courant en fonction de l'utilisation de nos moteurs à l'aide d'un capteur de courant. Ce dernier a été défini dans la classe « captCourant » avec une méthode « testCourant » qui utilise la librairie INA219 importée avec un objet instancié de cette librairie pour utiliser des méthodes comme voltage ou current et qui renvoie sous forme de ''print'' les valeurs de tension, de puissance et d'intensité enregistrées sur le moment selon l'activité des moteurs.

Pour se faire, nous avons lancé le script de cette classe en même temps que celui utilisé pour faire tourner les moteurs à leur maximum afin de déterminer quelles valeurs maximales il ne fallait pas dépasser et donc pour quelles valeurs il fallait stopper les moteurs. Cependant, malgré le fait que tous nos tests aient été couronnés de succès, nous n'avons pas réussi à implémenter cette classe dans notre script d'instanciation parce que nous avions des erreurs et que nous n'avions pas suffisamment de temps pour les déboguer.

4. Problèmes rencontrés

La réalisation de ce projet a fait face à plusieurs obstacles tout du moins dans notre groupe.

Tout d'abord, la familiarisation avec la méthode Scrum a été assez compliquée compte tenu de toutes les « contraintes » qu'elle présentait, ce problème s'est cependant arrangé au fil des jours pendant lesquels la répartition des tâches et le remplissage des Scrum plannings sont devenus une habitude.

Nous avons également eu du mal à mettre en place une cohésion de groupe effective étant donné que les groupes pour ce projet ont été formés aléatoirement et que nous ne connaissions pas tous vraiment ce qui s'est également arrangé au fil du projet.

Ensuite nous avons fait face à l'abandon de l'un des membres de notre groupe ; de quatre membres, nous sommes donc passés à trois sans compter le fait que nous avions du retard dans la réalisation de nos tâches et que nous nous retrouvions en effectif réduit à deux jours de la présentation finale, cet évènement a fortement mis à mal l'évolution du travail au sein du groupe ce qui a impacté la réalisation de nos tâches et conditionné que certaines soient faites et d'autres non.

Pour continuer, nous avons rencontré plusieurs fois des soucis avec le câblage de notre module. Les câbles se déconnectant de nos composants (n'étant pas souder). Nous avons perdu pas mal de temps sur notre temps de travail avant de réaliser que l'erreur venait du câblage.

Pour terminer, nous avions également eu un souci au niveau de nos capteurs Ultrason.

5. Améliorations qu'on aurait pu apporter

Concernant les améliorations qui auraient pu être apportées au niveau du module, nous aurions préalablement pu effectuer celles que nous n'avons pas pu réaliser par manque de temps et d'effectif notamment la mise en place de la classe pour la détection des couleurs pour le démarrage automatique et/ou l'arrêt automatique de la voiture en fonction de la couleurs des feux, la réalisation de plusieurs tests unitaires que nous n'avons pas pu terminer ou encore l'implémentation de la classe pour le capteur de courant afin que la tâche soit terminée et que le module soit exempt de problèmes liés à une charge électrique trop importante.

Dans le cahier des charges, il y avait également des tâches bonus à effectuer comme la connexion du module en Bluetooth via une application Android qui aurait pu être une activité supplémentaire pour développer nos compétences.

Au sujet de notre organisation de groupe, nous aurions effectivement pu améliorer la partie « répartition des tâches » et mieux interroger les capacités et compétences des uns et des autres avant de s'attribuer des tâches, cela nous aurait probablement fait perdre moins de temps.

Nous aurions également dû mieux mettre à disposition notre temps pour optimiser la période de réalisation de certaines tâches mais encore une fois avec un effectif réduit et la nécessité de mettre en ordre tous nos documents de la méthodologie Scrum, nous n'étions que deux à jouer le rôle de développeur et de testeur sans compter que notre Scrum master, fort de ses attributions initialement importantes devait également s'occuper du câblage du module.

6. Conclusion

En conclusion, l'objectif de ce projet était de concevoir une voiture autonome capable d'effectuer un nombre déterminé de tours de circuit dans un temps record.

Durant cette semaine de projet, nous avons dû faire preuve de travail d'équipe, de persévérance, de compréhension et nous avons appris à structurer notre travail avec la méthode Scrum.

Un projet intense, nous n'avons malheureusement pas validé toutes les « Users stories » mais nous avons abouti à la réalisation d'un tour de circuit et ainsi à être qualifié pour faire la course avec l'ensemble des modules.

Enfin, nous souhaitons remercier l'ensemble des professeurs qui ont contribué à la création de ce projet. Également les professeurs qui ont fait preuve de bienveillance et qui ont aidé pour la réalisation de notre projet.

Figure 16 - Photo de classe - fin du projet de "gestion de projets"

7. Bibliographie

Installation SSH :

Fr, R. P. (2020, 6 février). *Connectez-vous en SSH à votre Raspberry Pi pour la contrôler depuis votre ordinateur*. Raspberry Pi FR. https://raspberry-pi.fr/connecter-ssh-raspberry-pi/

Documentation sur le Capteur Ultrason :

Engineers, L. M. (2022b, juin 11). How HC-SR04 Ultrasonic Sensor Works & # 038 ; Interface It With Arduino. *Last Minute Engineers*. https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/

Capteurs à ultrasons / Principe / microsonic - Capteur à ultrason. (s. d.-b). https://www.microsonic.de/fr/support/capteurs-%C3%A0-ultrasons/principe.htm

Capteur à ultrasons HC-SR04 HCSR04. (s. d.). Robot Maker. https://www.robot-maker.com/shop/capteurs/13-telemetre-a-ultrasons-hc-sr04-13.html

Datasheet :

http://www.robot-maker.com/shop/img/cms/datasheet-capteur-ultrasons-hc-sr04.pdf

Servo-moteur :

Fonctionnement d'un servomoteur | Kollmorgen. (2020, 29 septembre). https://www.kollmorgen.com/fr-fr/blogs/principes-de-fonctionnement-dun-servomoteur

Leveugle, X. (2022, 28 avril). Quelles différences entre le moteur à courant continu et le servomoteur ? *Transmission Aquitaine*. https://www.transmission-aquitaine.com/quelles-differences-entre-le-moteur-a-courant-continu-et-le-servomoteur

Datasheet :

https://www.st.com/resource/en/datasheet/l298.pdf

Capteur Infrarouge :

Alicia. (2022, 7 septembre). *Capteur optique infrarouge TCRT 5000. | modules et capteurs*. Le Disrupteur Dimensionnel. https://ledisrupteurdimensionnel.com/arduino/capteur-optique-infrarouge-tcrt-5000/

Découvrez la constitution des capteurs. (s. d.). OpenClassrooms. https://openclassrooms.com/fr/courses/5224916-developpez-un-robot-mobile-connecte-par-bluetooth/5411806-decouvrez-la-constitution-des-capteurs

Datasheet :

https://www.vishay.com/docs/83760/tcrt5000.pdf

Capteur RGB :

TCS34725 - capteur de couleur RGB + Filtre IR + LED blanche. (s. d.). MCHobby - Vente de Raspberry Pi, Arduino, ODROID, Adafruit. https://shop.mchobby.be/fr/autres-capteurs/1513-tcs34725-capteur-de-couleur-rgb-filtre-ir-led-blanche-3232100015135-adafruit.html

Datasheet :

Industries, A. (s. d.). *RGB Color Sensor with IR filter and White LED - TCS34725*. https://www.adafruit.com/product/1334

L298N H-bridge :

Instructables. (2017). Arduino Modules - L298N Dual H-Bridge Motor Controller. *Instructables*. https://www.instructables.com/Arduino-Modules-L298N-Dual-H-Bridge-Motor-Controll/

Datasheet :

https://www.st.com/resource/en/datasheet/l298.pdf

Méthode Scrum

Fernandez, A. (2020). Qu'est-ce que Scrum, méthode de développement agile. *Management et Performance, piloter.org*. https://www.piloter.org/projet/methode/scrum.htm#def