HEH be
Sciences
et technologies

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

WWW.HEH.BE

Conception de projet

Document de justification des choix technologiques

2024-2025
Groupe N°4

Hemeryck Quentin
Herbigniaux Néo
Carsault Mathias
Goffaux Benjamin
Bougma Adam
Marcel Alexandre

Commenté [Ui1]: mettre dans l'ordre alphabétique les noms

WALLONIE-BRUXELLES ENSEIGNEMENT
pôle hainuyer

Table des matières

1. Introduction

Dans le cadre du projet de voiture autonome, nous avons été amenés à faire plusieurs choix technologiques afin de répondre aux exigences du cahier des charges. Ces choix concernent à la fois le matériel utilisé (Raspberry Pi, capteurs, moteurs) et les outils logiciels (système d'exploitation, langage de programmation, bibliothèques, etc.).

Ce document a pour but d'expliquer de manière claire et concise les solutions que nous avons retenues, ainsi que les raisons qui ont motivé nos décisions. Nous avons pris en compte la compatibilité avec le matériel fourni, la simplicité d'intégration, les connaissances de l'équipe et la faisabilité dans le temps imparti.

2. Choix du matériel

Le matériel utilisé dans ce projet nous a été imposé par l'équipe enseignante, dans le cadre du cahier des charges. Il s'agit d'un kit basé sur le SunFounder Smart Video Car, accompagné de composants complémentaires fournis par l'école. Le cœur du système est un Raspberry Pi 3 Model B, utilisé pour piloter les moteurs, lire les capteurs et gérer la logique du programme.

D'autres cartes comme l'ESP32 ou l'Arduino étaient possibles, mais moins adaptées. L'Arduino ne permet pas de gérer un système complet ou une interface. Le Raspberry Pi 3, compatible avec Python, simple à utiliser et abordable (± 35 €), est un choix idéal pour ce projet.

Raspberry Pi 3 Model B

Ce modèle a été imposé comme carte principale. Il présente plusieurs avantages : suffisamment puissant pour le traitement en Python, équipé de ports GPIO pour les capteurs et actionneurs, et doté d'un module Wi-Fi intégré facilitant les connexions à distance (SSH, VNC). Ce choix nous a permis de travailler efficacement sans avoir à gérer des cartes séparées pour le traitement et la communication.

Capteurs utilisés

Le kit contient :

- Trois capteurs à ultrasons HC-SR04 : utilisés pour la détection d'obstacles. Ils sont reliés aux GPIO pour les signaux TRIG et ECHO.
- Un capteur infrarouge (line follower) : connecté au GPIO, il permet de détecter la ligne d'arrivée sur le circuit.

Ces capteurs ont été choisis pour leur compatibilité avec le Raspberry Pi et leur simplicité d'intégration. Le schéma fourni insiste sur la vérification de l'ordre des broches (GND, VCC, etc.) avant tout branchement.

Moteur et servomoteurs

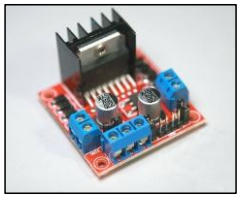
- Deux moteurs DC assurent la propulsion du véhicule (roues arrière).
- Un servomoteur est utilisé pour la direction des roues avant.

Les moteurs DC sont contrôlés via un driver L298N, et les servomoteurs sont gérés par un contrôleur PWM PCA9685. Ces modules sont tous préinstallés dans le châssis fourni, ce qui facilite leur mise en œuvre.

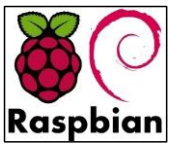
Alimentation

L'alimentation est divisée en deux parties :

- Le Raspberry Pi est alimenté par une batterie PiJuice connectée directement à son port GPIO.
- Les moteurs sont alimentés par quatre batteries 18650, reliées à un convertisseur de tension XL1509. Celui-ci stabilise la tension avant de l'envoyer au driver moteur L298N.



3. Choix du système d'exploitation



Nous avons utilisé Raspbian avec interface graphique car c'est le système d'exploitation officiel, recommandé et optimisé par Raspberry Pi. Ce système est simple à installer grâce à *Raspberry PI Imager* et à utiliser. Il permet de voir directement ce qui se passe sur le Raspberry, de lancer facilement les programmes et d'accéder aux fichiers sans passer par des commandes compliquées.

L'interface graphique est aussi très utile pour les tests. Nous avons utilisé comme client RealVNC pour contrôler le Raspberry Pi à distance en accédant à son GUI, car nous connaissions déjà cette solution et savions qu'elle fonctionnait bien. Cela nous a permis de gagner du temps lors de la configuration, grâce à l'interface graphique.

Ce choix est également important pour la suite du projet. Par exemple, si nous ajoutons une webcam, il sera plus facile d'afficher l'image en direct avec un système graphique que depuis un terminal. De plus, il pourrait nous permettre de montrer une interface avec les informations comme la vitesse et le classement.

4. Choix du langage de programmation

Le langage utilisé pour ce projet est Python 3. C'est un choix logique, car le Raspberry Pi est parfaitement compatible avec Python, et de nombreuses bibliothèques utiles sont déjà intégrées ou faciles à installer. Python permet aussi de contrôler les GPIO, les capteurs et les moteurs sans configuration complexe.

C'est aussi un langage que nous maîtrisons déjà, car nous avons appris à l'utiliser l'année passée, notamment pour coder en programmation orientée objet (POO). Cela nous permet de structurer le projet proprement, en organisant le code par modules et en appliquant de bonnes pratiques vues en cours.

Python est également bien adapté au travail en équipe. Sa syntaxe claire et sa flexibilité facilitent la lecture et la modification du code par tous les membres du groupe. Il reste aussi un bon choix pour les extensions futures du projet, comme l'ajout d'une interface graphique ou d'un flux vidéo via webcam.

5. Choix des outils de développement

Pour développer notre projet, nous avons utilisé des outils simples, efficaces et adaptés à notre environnement.

Le code a été écrit avec Visual Studio Code (VS Code), un éditeur que nous connaissions déjà et qui est très pratique.

Grâce à l'extension *Remote – SSH* de Microsoft, nous avons pu nous connecter directement au Raspberry Pi et éditer les fichiers à distance, sans devoir transférer quoi que ce soit manuellement.

Le suivi de version du code a été géré avec Git, via une plateforme GitHub partagée avec tous les membres de l'équipe. Cela nous a permis de travailler en parallèle, d'éviter les conflits et de garder un historique clair des modifications.

Enfin, pour organiser notre travail selon la méthode Scrum appris récemment en cours, nous avons utilisé Trello comme tableau de bord. Chaque User Story est représentée par une carte, avec son état (à faire, en cours, terminé), ce qui nous aide à suivre l'avancement du projet jour après jour

6. Conclusion

Les choix technologiques présentés dans ce document ont été faits en tenant compte des contraintes du projet fournis dans le cahier des charges du matériel imposé et des compétences acquises en cours. Nous avons privilégié des composants simples à intégrer, compatibles avec le Raspberry Pi, et bien documentés. Le langage Python, que nous maitrisons suite aux cours de l'année précédente est un excellent outil pour structurer notre code et répondre aux besoins du projet

Les outils de développement comme Visual Studio Code, Git et Trello nous ont permis de travailler efficacement en équipe, tout en respectant la méthode Scrum Le système Raspbian avec interface graphique a facilité les tests, les accès à distance, etc...

Ces choix nous ont permis de gagner du temps, d'éviter des erreurs et de nous concentrer sur la réalisation des fonctionnalités attendues.