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

Gestion de projet

Projet Voiture 2IRT

Choix technologiques

Année Académique
2024 - 2025

Groupe
5

Membres
Colle Joulian
Deneyer Tom
Kruczynski Mathis
Mauroit Antoine
Staquet Esteban
Vangeebergen Augustin

Table des matières

1 Choix Technologiques Matériels

1.1 Le Raspberry Pi comme Solution Optimale

Le choix du Raspberry Pi 3 Modèle B comme plateforme centrale pour ce projet s'est imposé face aux microcontrôleurs traditionnels comme Arduino pour plusieurs raisons fondamentales :

- **Puissance de calcul** : Contrairement aux microcontrôleurs classiques, le Raspberry Pi intègre un processeur quad-core à 1.2GHz et 1GB de RAM, permettant d'exécuter un système d'exploitation complet (Raspbian) et de gérer simultanément :
  - Le traitement des données des capteurs en temps réel
  - Les algorithmes de navigation (relativement) complexes
  - La communication réseau
- **Connectivité intégrée** : Le Raspberry Pi offre nativement :
  - Wi-Fi 802.11n et Bluetooth 4.1 pour les communications sans fil
  - 4 ports USB pour connecter des périphériques additionnels
  - Interface HDMI pour le débogage
- **Multitâching** : La capacité à exécuter plusieurs processus en parallèle est cruciale pour :
  - Gérer simultanément les capteurs et les moteurs
  - Maintenir une connexion SSH active
  - Exécuter un serveur web pour l'interface de contrôle
- **Écosystème logiciel** : L'environnement Linux permet d'utiliser :
  - Python 3 comme langage principal avec toutes ses bibliothèques
  - Des outils professionnels de versioning (Git)
  - Des frameworks de test et d'intégration continue

1.2 Limitations des Microcontrôleurs Traditionnels

Les solutions comme Arduino ou STM32, bien que performantes pour certaines applications, présentent des limitations majeures pour ce projet :

- **Capacité de traitement insuffisante** pour les algorithmes avancés de détection d'obstacles et d'optimisation de trajectoire
- **Absence de système d'exploitation** rendant complexe :
  - La gestion concurrente des tâches
  - Le débogage avancé
  - Les communications réseau
- **Connectivité limitée** nécessitant des shields additionnels pour :
  - Le Wi-Fi/Bluetooth
  - Le stockage de données
  - Les interfaces utilisateur
- **Espace mémoire restreint** incompatible avec :
  - Le stockage des logs de diagnostic
  - L'exécution de bibliothèques complexes
  - La gestion de protocoles réseau complets

1.3 Comparaison avec d'Autres Solutions

TABLE 1 – Comparatif des plateformes matérielles

Caractéristique	Raspberry Pi 3	Arduino Mega	STM32F4
Processeur	Quad-core 1.2GHz	16MHz	180MHz
Mémoire	1GB RAM	8KB RAM	192KB RAM
Système d'exploitation	Linux	Aucun	RTOS optionnel
Connectivité	Wi-Fi/BT intégrés	Requiert shields	Requiert modules
Langages supportés	Python, C++, etc.	C/C++	C/C++
Prix	~35€	~40€	~25€

Ce tableau montre clairement l'avantage du Raspberry Pi en termes de rapport performance/prix pour les besoins spécifiques de ce projet.

2 Choix Technologiques Logiciels

2.1 Système d'Exploitation

Le choix s'est porté sur **Raspbian** (version GUI) pour plusieurs raisons techniques :

- **Compatibilité matérielle** : Support natif de tous les composants du projet
- **Stabilité** : Distribution officielle maintenue par la fondation Raspberry Pi
- **Outils intégrés** : Inclut par défaut :
  - Gestionnaire de paquets apt
  - Interfaces de configuration matérielle
  - Environnement de développement complet

2.2 Langage de Programmation

Python 3 a été sélectionné comme langage principal pour :

- **Productivité** : Syntaxe claire permettant un développement rapide
- **Bibliothèques spécialisées** :
  - RPi.GPIO pour le contrôle des GPIO
  - NumPy pour les calculs matriciels
  - Matplotlib pour la visualisation des données
- **Communauté active** : Large base d'utilisateurs et documentation abondante
- **Portabilité** : Fonctionne sur tous les systèmes d'exploitation

2.3 Outils de Développement

L'environnement logiciel comprend :

- **Git** : Pour le versioning et la collaboration en équipe
- **SSH/TightVNC** : Permet le développement à distance
- **UnitTest** : Framework de tests unitaires intégré à Python

2.4 Architecture Logicielle

L'application suit une architecture modulaire :

- **Couche matérielle** : Drivers pour les capteurs et actionneurs
- **Couche contrôle** : Algorithmes de navigation et détection
- **Couche communication** : Gestion des connexions réseau
- **Interface utilisateur** : Console et interface web

Cette architecture permet une maintenance aisée et une bonne séparation des préoccupations.