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

Documentation des choix logiciels

Année Académique 2024-2025
Groupe N°6
Debande Aurélien, De Coster Koryan, Duchenne Corentin,
Isembaert Nathan, Kozlenko Anastasiia, Rousche Aurélien.

WALLONIE-BRUXELLES ENSEIGNEMENT
pôle hainuyer

Choix des logiciels

Choix du langage : Python 3

Nous avons opté pour le langage Python 3 car il combine à la fois simplicité, puissance et accessibilité. Tous les membres de l'équipes ont déjà travillé avec python durant l'année passée, ce qui facilite notre travail. Il offre un écosystème très riche avec des bibliothèques pour presque tous les besoins : capteurs, moteurs, gestion du temps, entrées/sorties, tests, etc.
Python est préinstallé sur Raspberry Pi OS, ce qui nous fait gagner un temps précieux pour débuter rapidement les tests.

Choix de la programmation orientée objet

La POO est particulièrement adaptée à notre projet car elle permet de modéliser chaque composant de la voiture comme une entité autonome. Un capteur, un moteur ou un module de contrôle peut être représenté par une classe dédiée, avec ses propres méthodes et attributs. Cela rend le code plus lisible, modulaire et réutilisable. De plus, les tests unitaires sont facilités et permettent à plusieurs membres de l'équipe de travailler en parallèle sur différentes fonctionnalités sans conflit. Grâce à l'encapsulation, on peut aussi protéger les données critiques de chaque composant, en respectant de bonnes pratiques logicielles.

Choix de Git

Pour assurer une bonne gestion de version du code et faciliter le travail collaboratif, nous avons adopté Git comme outil de suivi de projet. Grâce à Git, chaque membre de l'équipe peut travailler sur sa propre branche sans perturber le travail des autres, tout en gardant une trace claire de chaque modification apportée. Cela permet aussi de revenir à une version antérieure en cas de bug ou d'erreur critique. En liant Git à GitHub, nous pouvons centraliser le développement, commenter les changements, documenter le projet et organiser efficacement notre travail en équipe.

Choix de Visual Paradigm

Visual Paradigm Online nous permet de créer l'uml de notre projet très facilement. Nous pouvons travailler en même temps sur l'uml et corriger les éventuelles erreurs aisément. De plus, nous avons déjà tous utilisé Visual Paradigm Online l'année passé ce qui facilite et rend plus rapide notre travail.

Choix de l'environnement : Raspbian OS / Raspberry Pi OS

Le système choisi pour faire tourner notre projet est Raspberry Pi OS . Il est spécialement conçu pour tirer le meilleur parti du matériel du Raspberry Pi.
Ce système est à la fois léger, fiable et parfaitement adapté aux projets embarqués comme le nôtre. Il offre un accès direct aux broches GPIO, une compatibilité native avec Python, et une documentation très fournie. Cela nous permet de travailler dans un environnement stable, bien documenté, et soutenu par une grande communauté.

Choix des bibliothèques Python

a) RPi.GPIO

C'est la bibliothèque de base pour interagir avec les broches GPIO du Raspberry Pi. Elle nous permet de gérer très simplement des éléments comme des LED, des boutons ou des capteurs. Elle est légère, très bien intégrée à l'environnement Raspberry Pi, et surtout idéale pour garder un bon niveau de contrôle sur les signaux électroniques.

b) Adafruit_PCA9685

Indispensable pour piloter les moteurs via le module PCA9685, cette bibliothèque nous permet d'exploiter 16 canaux PWM grâce au protocole I2C. Cela simplifie grandement le contrôle simultané de plusieurs moteurs en limitant l'utilisation des broches GPIO du Raspberry Pi.

c) unittest et unittest.mock

Tester son code, c'est indispensable, surtout sur un projet à plusieurs. La bibliothèque unittest nous permet de vérifier que chaque morceau de code fonctionne comme prévu. De plus comme nous travaillons avec des cerveaux moteurs ou autres, tester notre code permet de ne pas risquer d'endommager notre matériel.

d) busio

La bibliothèque busio, fournie par CircuitPython, nous permet d'utiliser facilement des bus de communication comme I2C ou SPI. Elle sert souvent en coulisse, par exemple quand on utilise Adafruit_PCA9685, mais elle est essentielle pour établir une communication fiable entre notre Raspberry Pi et les modules externes comme les capteurs ou les contrôleurs moteurs.

Compatibilité et intégration

Tous les outils et bibliothèques que nous utilisons sont pensés pour fonctionner ensemble. Ils sont compatibles avec Python 3, avec le système Raspberry Pi OS, et exploitent au mieux l'architecture ARM du Raspberry Pi. Leur installation se fait facilement avec pip ou apt, et chacun bénéficie d'une communauté active et de tutoriels.

Ce socle logiciel nous offre donc :

- Un développement plus sûr grâce aux tests unitaires,
- Une interaction directe et fiable avec le matériel de la voiture,
- Une base flexible pour ajouter des fonctionnalités comme le multithreading, la vision ou le suivi de ligne.

En bref, nous avons construit une pile logicielle cohérente, pédagogique et évolutive, parfaitement adaptée à notre projet embarqué.