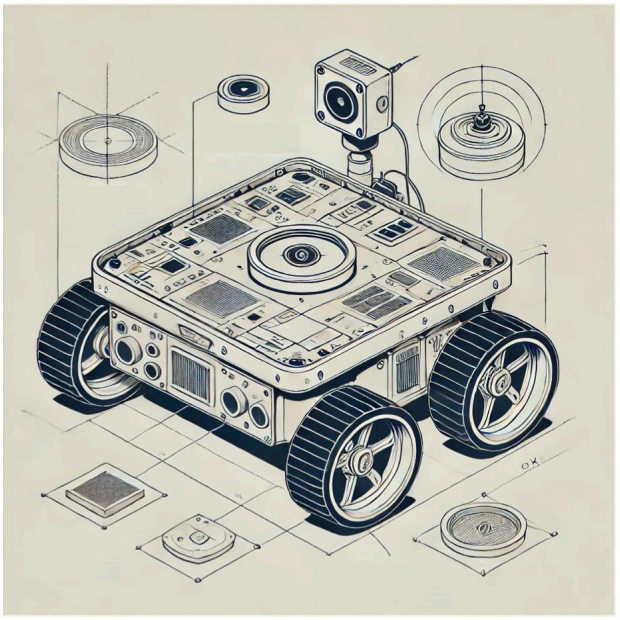
PROJET FIL ROUGE 2024-2025



Dossier de spécifications

Réalisé par :

- *TAYOU MBEDE Ryan Niel*
- *Imran WACHILL*
- *Hela BAKHTI*
- *Adji Aram SECK*
- *Wissal MEHREZ*
- *Djibril BA*

Sommaire

1. Contexte et objectifs

1.1 Contexte du projet

Le **Projet Fil Rouge** (PFR) s'inscrit dans le cadre d'un apprentissage progressif et technique sur deux semestres, visant à développer des compétences avancées en ingénierie robotique et en programmation. Structuré en deux phases distinctes, le projet a pour but de renforcer les connaissances acquises en cours et de les appliquer dans un environnement pratique, engageant les étudiants dans une démarche de conception, d'implémentation et d'intégration de solutions robotiques.

La première phase du projet (PFR1) se concentre sur le **développement de briques logicielles** essentielles, tandis que la deuxième phase (PFR2) a pour objectif d'intégrer ces modules logiciels dans une **plateforme robotique mobile**, permettant une mise en application des fonctionnalités dans des situations réelles. Ce cadre de projet permet aux étudiants de renforcer leurs compétences non seulement dans des domaines spécifiques comme le contrôle robotique et la vision par ordinateur, mais aussi dans des aspects transversaux tels que la gestion de projet, le travail en équipe et la validation de solutions techniques.

Dans ce rapport, seule la partie relative au PFR1 sera détaillée, couvrant spécifiquement le développement des briques logicielles de base nécessaires pour l'intégration future dans la plateforme robotique du PFR2.

1.2 Objectifs du projet PFR1

Le projet est organisé autour de plusieurs objectifs spécifiques, chacun contribuant à la réussite de l'ensemble du projet. Ces objectifs sont détaillés par phase et par domaine de compétence, afin de guider le développement et de servir de critères d'évaluation pour le projet.

Objectifs pour le PFR1 : Développement des Briques Logicielles

1. **Développement d'une Interface de Contrôle**
   - **Description :** Implémenter une interface utilisateur (IHM) permettant de commander le robot, soit via une application dédiée, soit via un joystick ou un autre dispositif de contrôle.
2. **Traitement d'images**
   - **Description :** Développer un module permettant de traiter des images. Seulement quelques formes et quelques couleurs seront traitées (voir suite du document).
3. **Implémentation de Commandes Vocales**
   - **Description :** Intégrer un module de reconnaissance vocale capable de traduire les instructions vocales en requête-commande, avec une analyse des données provenant des autres entrées pour en assurer la cohérence.
4. **Simulation des Déplacements et Visualisation**
   - **Description :** Intégrer un module de simulation permettant de visualiser les déplacements du robot en temps réel.
5. **Intégration de tout des différentes briques**
   - **Description :** Assembler les différentes briques logicielles développées (Traitement d'images, Commandes vocales, Interface de contrôle) et pouvoir simuler leur interactions.

2. Périmètre fonctionnel du projet

2.1 Commande et pilotage du robot

Dans le cadre du pilotage manuel, plusieurs modes de commande sont proposés pour permettre une interaction directe et flexible avec le robot, notamment via des dispositifs de contrôle physiques (un joystick par exemple ou une IHM sur un ordinateur…), ou des commandes vocales.

A. Mode manuel

Ce mode pilotage se réalise en interaction directe avec l'utilisateur, pouvant être effectué soit via un outil de contrôle, soit via des commandes vocales. Les deux modes sont détaillés dans la suite.

- Mode pilotage via un outil de contrôle (joystick et/ou application)

Fonction : IHM_user_to_robot()

Entrées : les requêtes-commandes, sous format structuré (exemple : JSON), sont envoyées par l'utilisateur par le biais d'un IHM ou joystick.

Action : le robot exécute la commande. Exemples de commande : "avancer", "reculer", "tourner", "arrêter".

Sorties : l'historique des déplacements du robot est stocké dans un fichier structuré (exemple : JSON) servant de base de données. Chaque action (exemples : "avancer", "tourner") est enregistrée avec des détails comme la direction, la distance ou l'angle, et un statut d'exécution (« réussi », « interrompu »).

Ci dessous un schéma expliquant le mode de fonctionnement global de ce mode.

Il est à préciser que le joystick n'est qu'un exemple d'équipement pouvant être utilisé pour le contrôle du robot : ceci sera réalisé dans la seconde partie du projet (PFR2). À noter que nous pourrions utiliser un smartphone à la place.

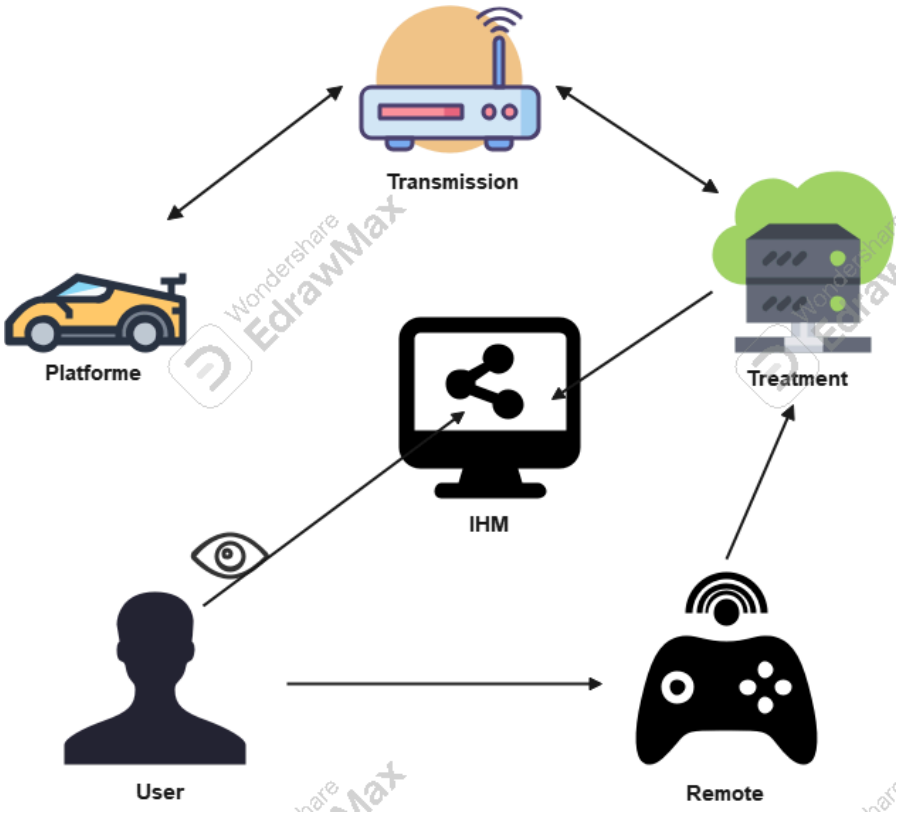


Figure 1 : Architecture Pilotage via outil de contrôle

La figure ci-dessus représente de façon globale le fonctionnement du mode de pilotage via un outil de contrôle. À noter que l'IHM est optionnelle : elle représente juste ici un moyen de faire parvenir à l'utilisateur une information.

- L'utilisateur se sert d'un outil (joystick ou IHM) pour transmettre une instruction au robot.
- L'instruction transite par "Treatment" qui se charge de traiter l'instruction (par exemple faire correspondre l'appui sur le bouton X du joystick à une instruction avancée compréhensible par le robot).
- Le résultat est passé à "Transmission" qui s'occupe de transmettre l'instruction au robot et de renvoyer la réponse du robot (par exemple que l'instruction a bien été réalisée) à "Treatment" à destination de l'utilisateur. Par exemple dans le cas d'une IHM l'utilisateur verra un message à l'écran.

Dans ce mode, le robot exécutera simplement les commandes de l'utilisateur sans vérifier la cohérence de celle-ci. Par exemple, supposons que le robot se trouve à 2m d'un mur et que l'utilisateur télécommande celui-ci pour qu'il avance de 3m alors le robot avancera tout simplement.

Cette fonctionnalité peut être divisée en trois :

- Côté utilisateur (**user interface**), une application ou un joystick permet de générer des instructions. Chaque instruction est vérifiée, puis traduite en actions spécifiques exécutables par le robot.
- Côté robot (**robot interface**), les actions spécifiques sont exécutées.
- Un programme (**link interface**), adossé au matériel adéquat, permet de "gérer" la communication entre l'utilisateur et le robot.

Figure 2 : Interfaces pour le pilotage via outil de contrôle

La figure ci-dessus représente l'interaction entre les différentes interfaces du mode de pilotage via un outil.

Un scénario possible peut être défini de la manière suivante :

- **User interface** transmet l'instruction d'avancer de 2 mètres.
- **Link interface** reçoit l'information et effectue quelques vérifications. Par exemple si l'information transmise est non altérée (complète).

- **Link interface** juge qu'il y a eu des erreurs de transmission et demande à **user interface** de renvoyer l'information.
- **User interface** envoie de nouveau l'information.
- **Link interface** analyse et juge que l'information est conforme.
- **Link interface** transmet l'information à **robot interface**.
- **Robot interface** reçoit l'information et exécute la commande.

- Mode pilotage vocal

Fonctions : speech_analysis()

Entrée : son (message vocal de l'utilisateur qui est un audio)

Sortie : affichage des actions

Action : Action du robot (exécution de la commande par le robot ou signalement d'un problème…)

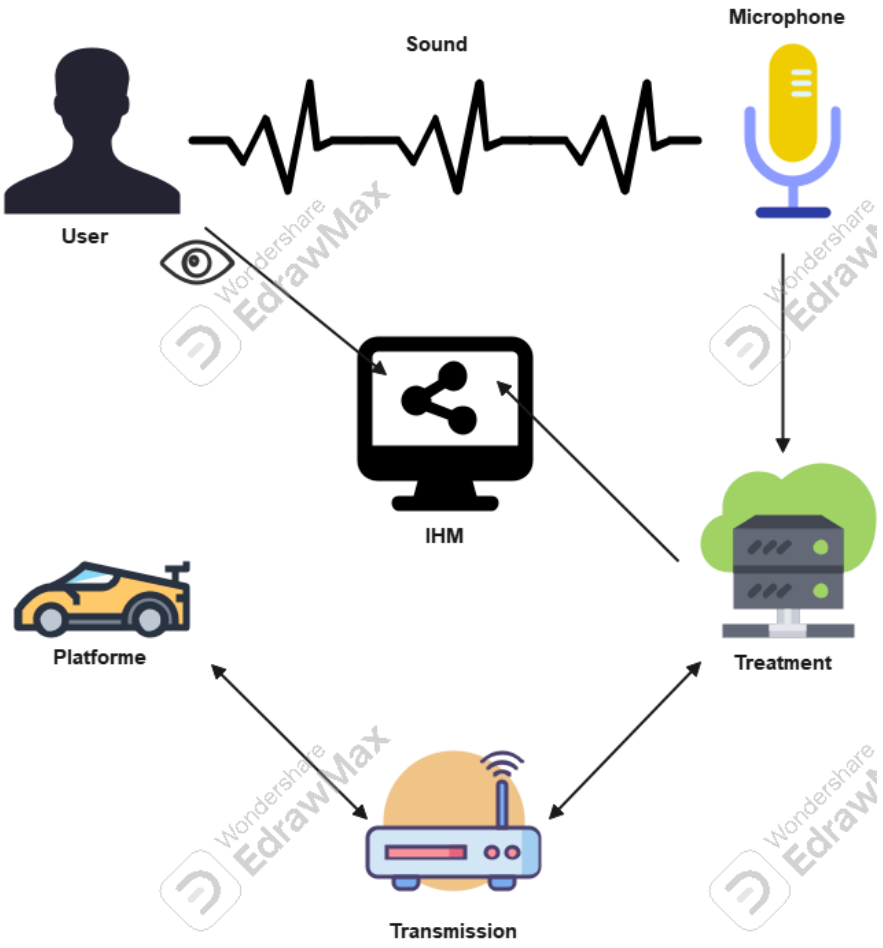


Figure 3 : Architecture Pilotage vocal

La figure ci-dessus représente de façon globale le fonctionnement du mode de pilotage vocal.

- L'utilisateur émet un son (instruction vocale).
- Le micro capte ce son puis le transmet au bloc de traitement (Treatment).
- Le bloc "Treatment" traite le son et prend une décision. Soit le son reçu n'est pas valide (ne correspond pas à une commande) alors il ne fait rien (ou optionnellement , nous pourrons demander à l'utilisateur d'émettre à nouveau par un message envoyé sur "IHM"). Si le son est valide alors le processus se poursuit.
- Le résultat est passé à "Transmission" qui s'occupe de transmettre l'instruction au robot et de récupérer la réponse du robot (par exemple que l'instruction a été bien réalisée ou qu'il ne peut pas avancer) à "Treatment" à destination de l'utilisateur. Par exemple, l'utilisateur verra un message sur l'IHM.

Contrairement au pilotage via un outil de contrôle, les instructions de l'utilisateur sont ici transmises via la voix. Ainsi, une fonctionnalité de transcription vocale est nécessaire : elle a pour rôle de traduire les instructions vocales de l'utilisateur en requêtes. Puis ses requêtes seront transmises à « link interface » et le processus décrit plus haut sera lancé. Ce mode de pilotage peut être résumé par le graphe ci-dessous.

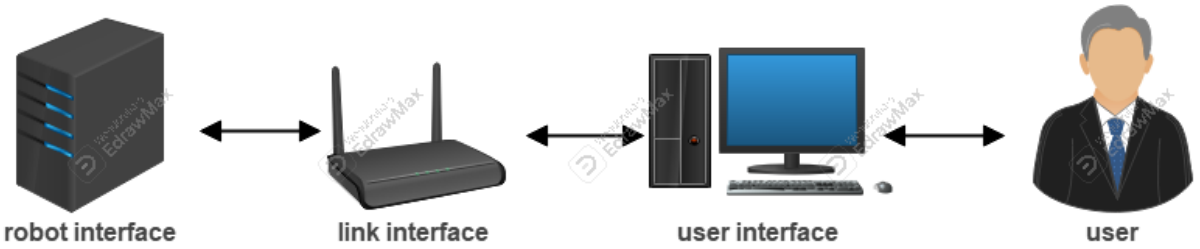


Figure 4 : Interfaces pour le pilotage vocal

Un scénario possible peut être le suivant :

- L'utilisateur donne une commande vocale.
- User interface traite la commande et constate qu'elle ne contient aucune information utile.
- User interface demande à l'utilisateur de donner à nouveau une commande.
- User interface valide la commande, la traite puis la transmet à link interface.
- Ensuite le scénario précédent (mode pilotage via une IHM) peut se dérouler à quelques exceptions près (voir ci-dessous **Particularité du mode automatique**).

Quelques exemples de commandes vocales :

- « Bonjour robot avance de 2 mètres » : les informations importantes sont « avance » et « 2 mètres ». Le module de transcription s'occupe de récupérer cette information

- « Bonjour robot » : ici par exemple, il y a aucune information qui doit provoquer une action. Le module de transcription peut demander par exemple à l'utilisateur de se répéter ou de ne juste rien faire. À noter que cette instruction est optionnelle.

Particularité du mode de pilotage vocal

Il faut noter que contrairement au mode de pilotage via un outil de contrôle, dans le mode vocal, la cohérence des commandes que le robot reçoit de l'utilisateur est analysée en fonction des données collectées par ses capteurs (exemple : micro) et l'historique des mouvements. Par exemple, l'utilisateur peut demander au robot de refaire le trajet précédent. Dans ce cas, il faudrait que le robot ait sauvegardé l'historique du déplacement précédent.

Scénario 01

- Utilisateur : « avance jusqu'à la balle rouge »
- Robot
    - 1er cas : le robot ne trouve pas la balle rouge alors il signale à l'utilisateur que ce n'est pas possible. Par exemple, un message est envoyé à destination de l'IHM.

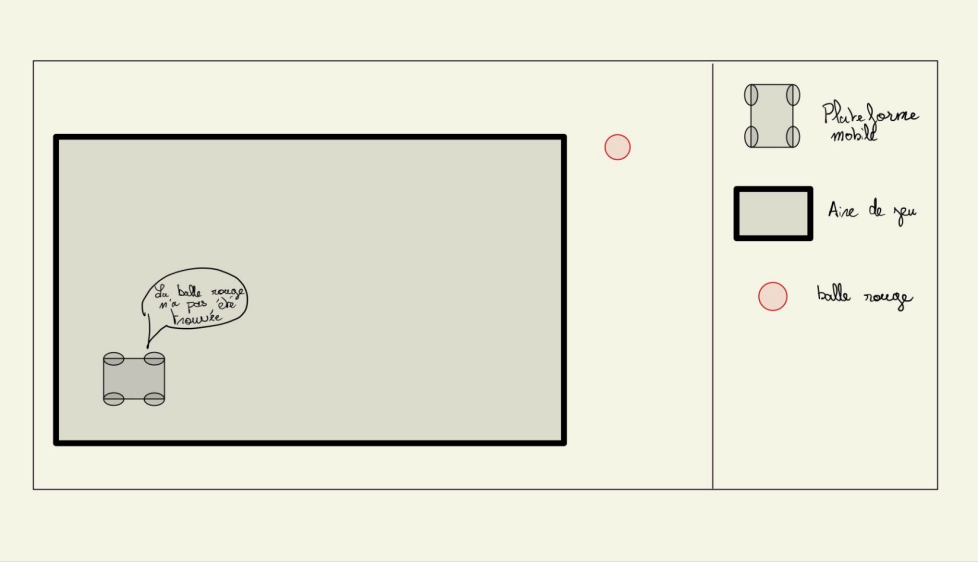


Figure 5 : Scénario 1, cas 1

    - 2éme cas : le robot repère la balle rouge. Alors c'est à lui de trouver un chemin qui mène à elle.

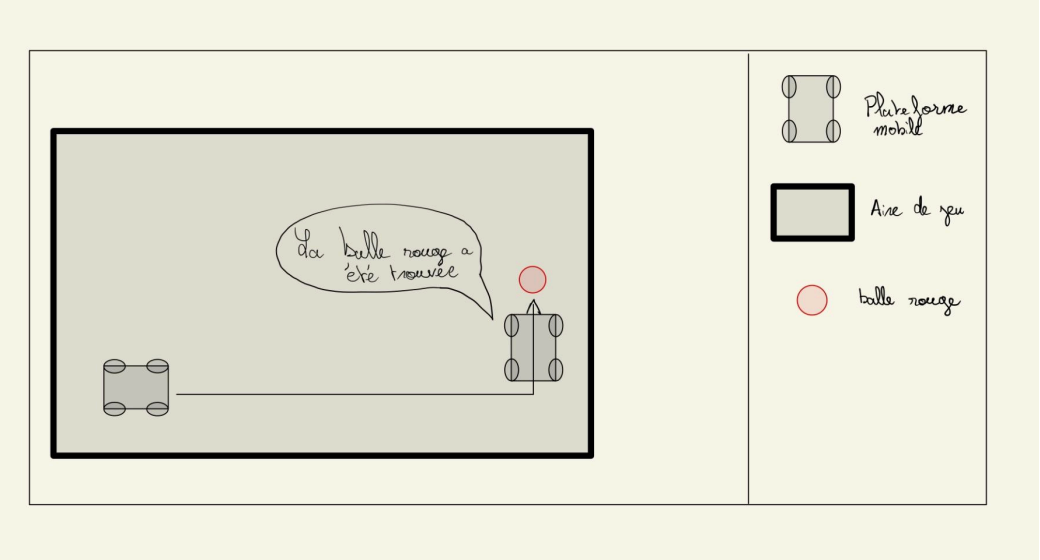


Figure 6 : Scénario 1, cas 2

Scénario 02

- Utilisateur : « avance de 3 mètres »
- Robot
    - 1er cas : le robot ne peux avancer de 3 mètres à cause d'un obstacle. Alors il signale à l'utilisateur que le déplacement demandé est impossible. Par exemple, un message est envoyé à destination de l'IHM.

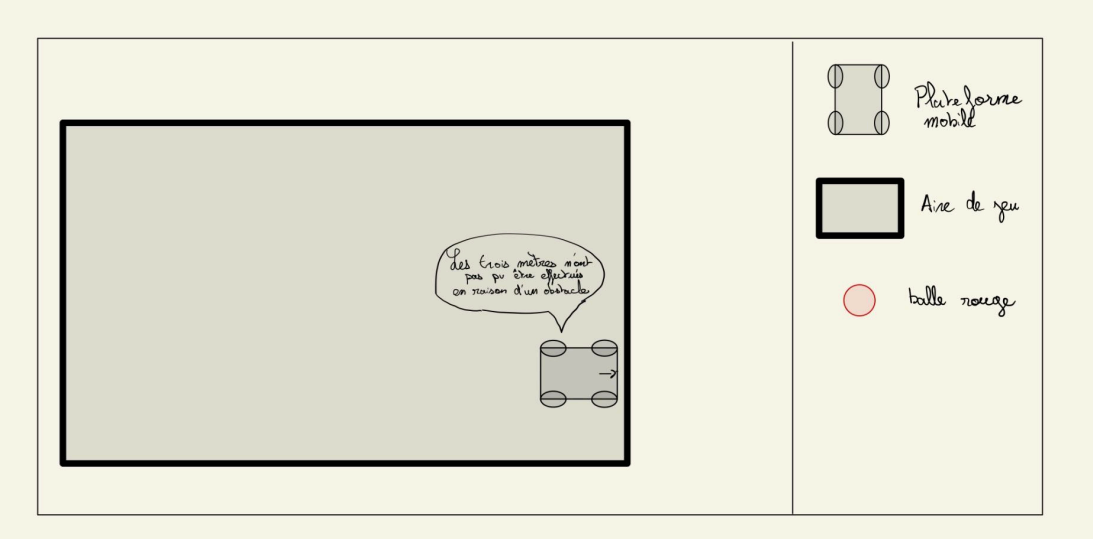


Figure 7 : Scénario 2, cas 1

    - 2éme cas : le robot avance et pendant qu'il avance il trouve un obstacles
        - 1er possibilité : il peut faire ses 3 mètres sans le percuter. Alors le robot réalise tout simplement ses 3 mètres.

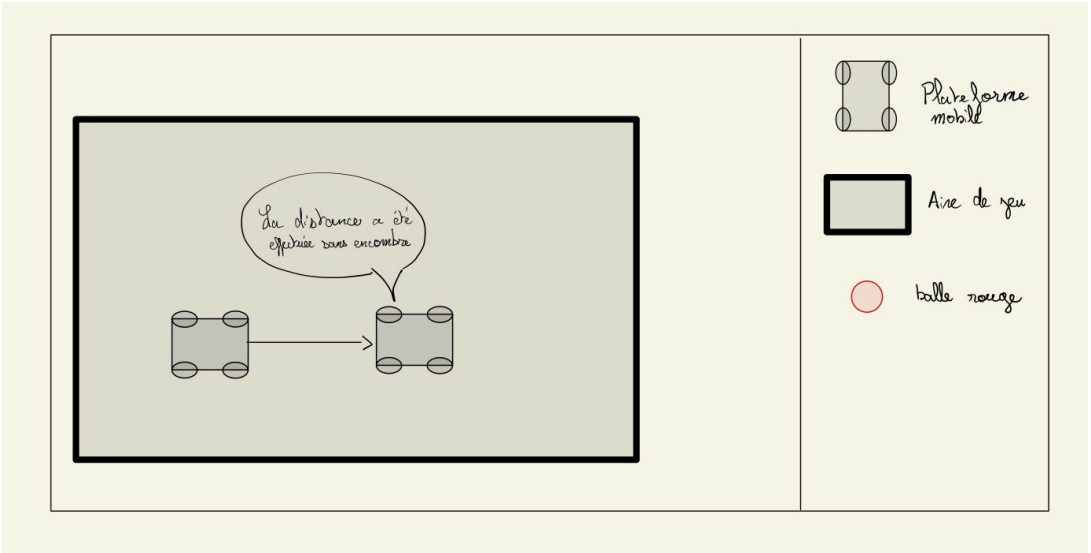


Figure 8 : Scénario 2, cas 2

- 3éme possibilité : il ne peut pas faire 3 mètres sans percuter l'obstacle. Alors le robot s'arrête et signale à l'utilisateur qu'il ne peut continuer, il signale également la distance manquante.

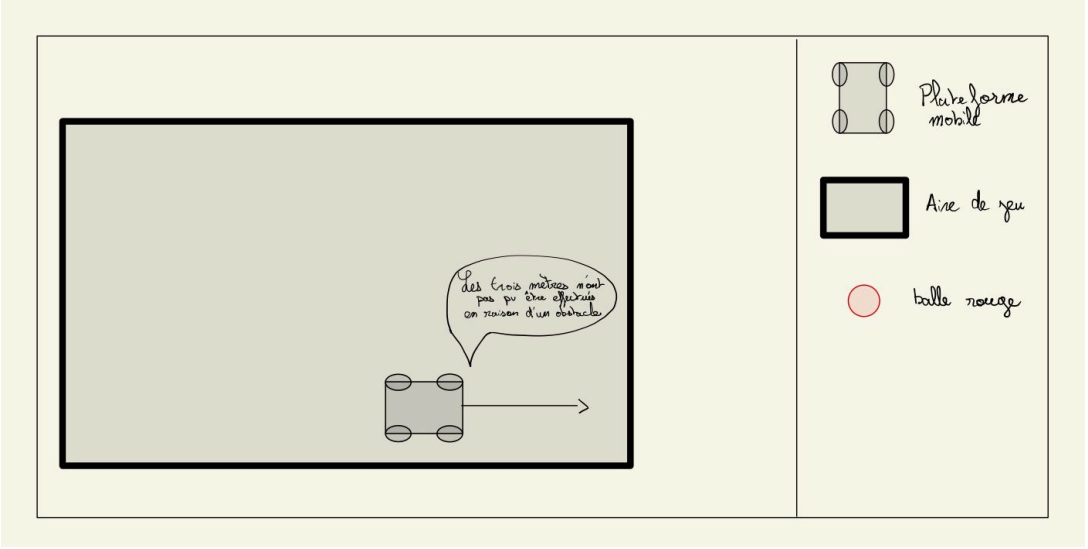


Figure 9 : Scénario 1, cas 3

Il est évident de noter que dans l'exemple 02 si le robot ne trouve pas d'obstacles pendant qu'il effectue l'instruction de l'utilisateur alors il exécute entièrement l'instruction.

B. Mode automatique

La partie automatique sera peu détaillée dans ce document, car elle sera approfondie et développée dans le cadre du PFR2.

Fonction : autom()

Entrées : Les données, converties en chaîne de caractères, proviennent de divers capteurs. Ainsi, pour garantir la compatibilité, les paramètres d'entrée sont en format chaîne de caractères.

Sortie : historique des déplacements (fichier enregistrant tous les déplacements).

Actions attendues: déplacement de la plateforme permettant de cartographier la pièce.

Dans ce mode, le robot se déplace de manière autonome en utilisant ses capteurs pour détecter son environnement et éviter les obstacles. Il peut reconnaître des objets spécifiques, comme des balles de couleur, grâce à une caméra embarquée. Le robot génère une carte de son environnement au fur et à mesure de ses déplacements, permettant une visualisation en temps réel de l'espace parcouru. Il est capable de naviguer aussi bien dans des environnements fermés que modifiables, et peut reproduire un trajet précédent si nécessaire.

Schématiquement, nous pouvons représenter ce mode par :

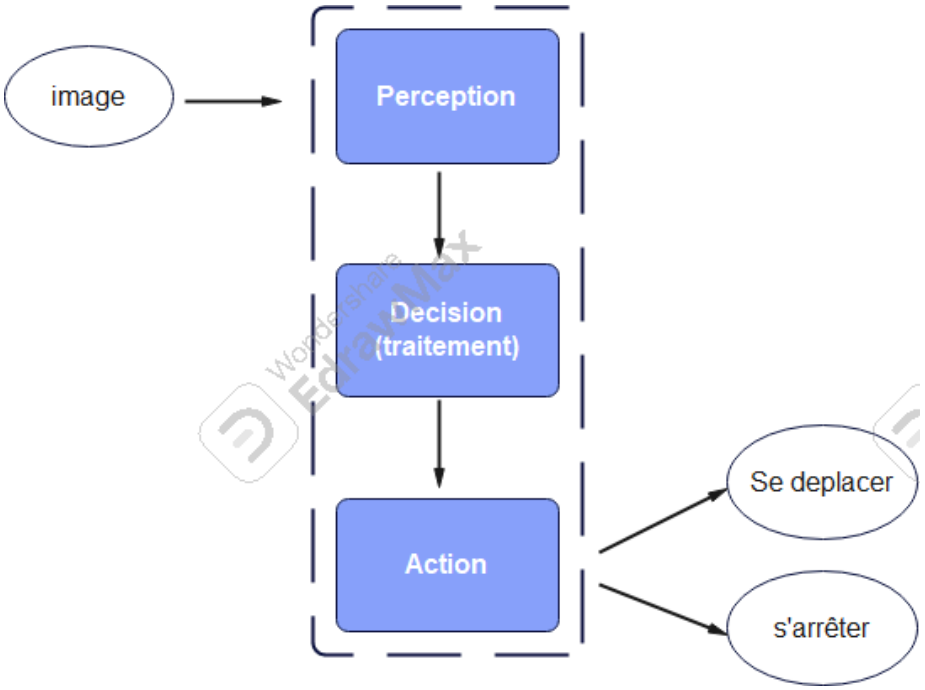


Figure 10 : Représentation du pilotage automatique

En mode automatique, le système doit fonctionner autour du triptyque perception-décision-action. Il perçoit son environnement (images), il traite les données collectées (images, historiques des précédents déplacements, cartographie réalisée

précédemment…) puis commande une action qui peut être de se déplacer ou de s'arrêter par exemple.

Ci-dessous, le robot est matérialisé par le camion.

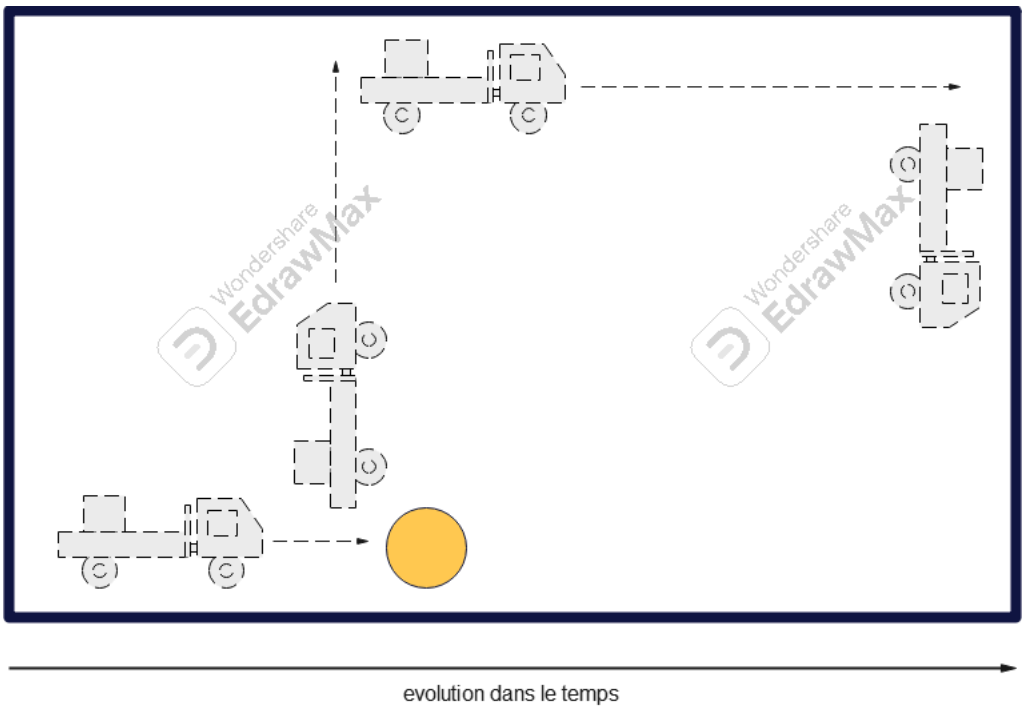


Figure 11 : Scénario pilotage automatique

Exemple de scénario :

- Le robot avance et un capteur (bloc "Perception") prend une image de ce qu'il y a devant lui et l'envoie en module traitement.
- Le bloc "Traitement" traite l'image et se rend compte qu'il y a une balle devant lui et qu'il ne peut tourner à droite. Après traitement, le module envoie l'action (tourner à gauche) à faire au module suivant.
- Le bloc "Action" reçoit l'information et s'assure qu'elle s'exécute.
- Puis le procédé est répété.

Il est à noter que lorsque plusieurs possibilités de déplacement sont valides, le robot peut en choisir une en fonction de la tâche à effectuer. Par exemple, à la fin du deuxième déplacement de notre robot, il est face à un mur. Mais vu que l'objectif est de cartographier toute la pièce, il décide de tourner à droite pour explorer un plus large espace.

2.2 Fonctionnalitées globales

Le projet est structuré autour de quatre fonctionnalités principales, chacune subdivisée en modules spécifiques pour répondre aux différents besoins de commande et de contrôle.

- **Commande vocale** : ce module permet de contrôler le système par des instructions vocales, offrant une interaction naturelle pour l'utilisateur.
- **Traitement d'images** : ce module traite les images capturées pour obtenir des informations pertinentes qui guideront le comportement du système.
- **Commande via outils** : ce module permet le contrôle direct du système par des dispositifs externes, donnant à l'utilisateur un accès manuel pour piloter les actions.
- **Simulation des déplacements** : cette fonctionnalité, développée en Python, simule les déplacements et actions du système pour valider et visualiser les comportements attendus.

Chaque fonctionnalité prend des types de données spécifiques en entrée (par exemple, les instructions vocales, les images capturées…) et produit des sorties destinées à la plateforme de simulation ou aux visualisations des actions effectuées (voir figure 12). Les sous-fonctionnalités seront détaillées dans les sections suivantes pour une compréhension approfondie de chaque module.

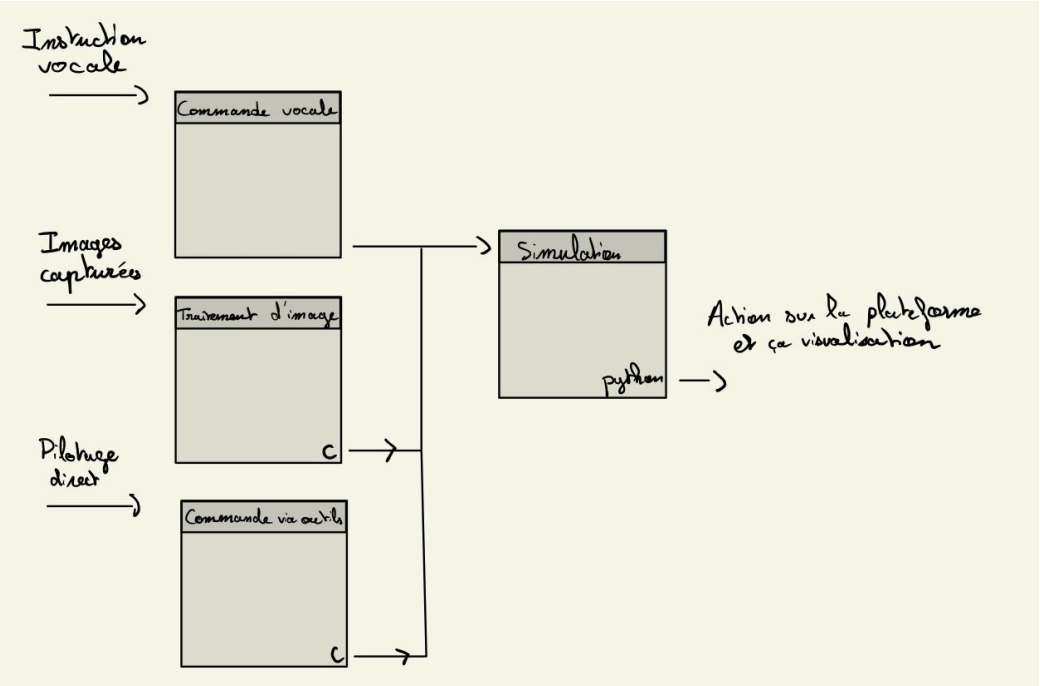


Figure 12 : décomposition des fonctionnalitées globales

2.3 Détails des fonctionnalitées

Le projet est détaillé en termes de fonctionnalités principales, chacune subdivisée en modules spécifiques. Cela inclut les interactions vocales, le traitement d'images, et la simulation des déplacements. Chaque module est décrit avec ses sous-fonctions, ses entrées et sorties, et son rôle au sein du système global, offrant ainsi une vue claire de la structure fonctionnelle et des exigences de chaque composant dans le projet.

Le schéma de la figure 13 résume les interactions entre les différentes fonctionnalités globales et leurs sous-fonctionnalités, offrant une vision simplifiée de la composition des différentes fonctionnalités globales décrites dans le tableau 1.

- En rouge, la fonctionnalité de **commande vocale**, qui comprend les sous-fonctionnalités **réception de la commande vocale** et **analyse de la réception**.
- En jaune, le module de **traitement d'image**, incluant les sous-fonctionnalités **traitement de la forme**, **traitement de la couleur**, et **traitement de la position**.
- En vert, la fonctionnalité de **commande via outils**, avec les sous-fonctionnalités **IHM (interface homme-machine)**, permettant la communication **utilisateur vers robot** et **robot vers utilisateur**.

Fonctionnalité	Fonction	Nom de la fonction	Entrée	Type d'entrée	Sortie	Type de sortie	Objectifs
Commande vocale	Réception commande vocale	speech_reception	Commande vocale	Audio	Texte	Texte	Permet de recevoir et transcrire les commandes vocales de l'utilisateur pour le robot.
	Analyse de la réception	speech_analysis	Texte	Texte	Instruction pour à exécuter	Texte	Permet d'analyser la commande vocale pour identifier l'action demandée par l'utilisateur.
Traitement d´images	Traitement de la forme	image_form	Image de l'environnement	Image extraite d'un flux vidéo	Forme de l'objet	Texte	Analyse de l'environnement afin de stocker la forme en vue d'un possible appel ultérieur.
	Traitement de la couleur	image_color	Image de l'environnement	Image extraite d'un flux vidéo	Couleur de l'objet	Texte	Analyse de l'environnement afin de stocker la couleur en vue d'un possible appel ultérieur.
	Traitement de la position	image_position	Image de l'environnement	Image extraite d'un flux vidéo	Position de l'objet	vecteur: (x,y)	Analyse de l'environnement afin de stocker la position en vue d'un possible appel ultérieur. Mais aussi pour prévenir des obstacles alentour afin d'éviter les collisions.
Commande via outils	IHM (Utilisateur vers robot)	IHM_user_to_robot	Commandes utilisateurs	événementielles (par exemple l'appui ou le click sur un bouton)	Commande pour le robot	Réel	Traduire les requêtes utilisateurs en commandes compréhensibles par le robot
	IHM (robot vers utilisateur)	IHM_robot_to_user	Notifications robot	Texte	Affichage sur l'IHM	Texte	Afficher les informations transmise par le robot
Simulation	Simuler les fonctionnalitée suivantes : Commande vocale ,Traitement d'images, Commande via outils	simulation	Choix de l'utilisateur	Événementielles (sélection depuis un terminal)	Visualisation de l'action	-	Le robot devra pouvoir simuler ses déplacements sur une carte virtuelle : - Visualisation des déplacements : représentation en temps réel des trajets effectués par le robot. - Rejeu des déplacements : possibilité de rejouer un parcours précédemment effectué par le robot.

Tableau 1 : détails des fonctionnalitées

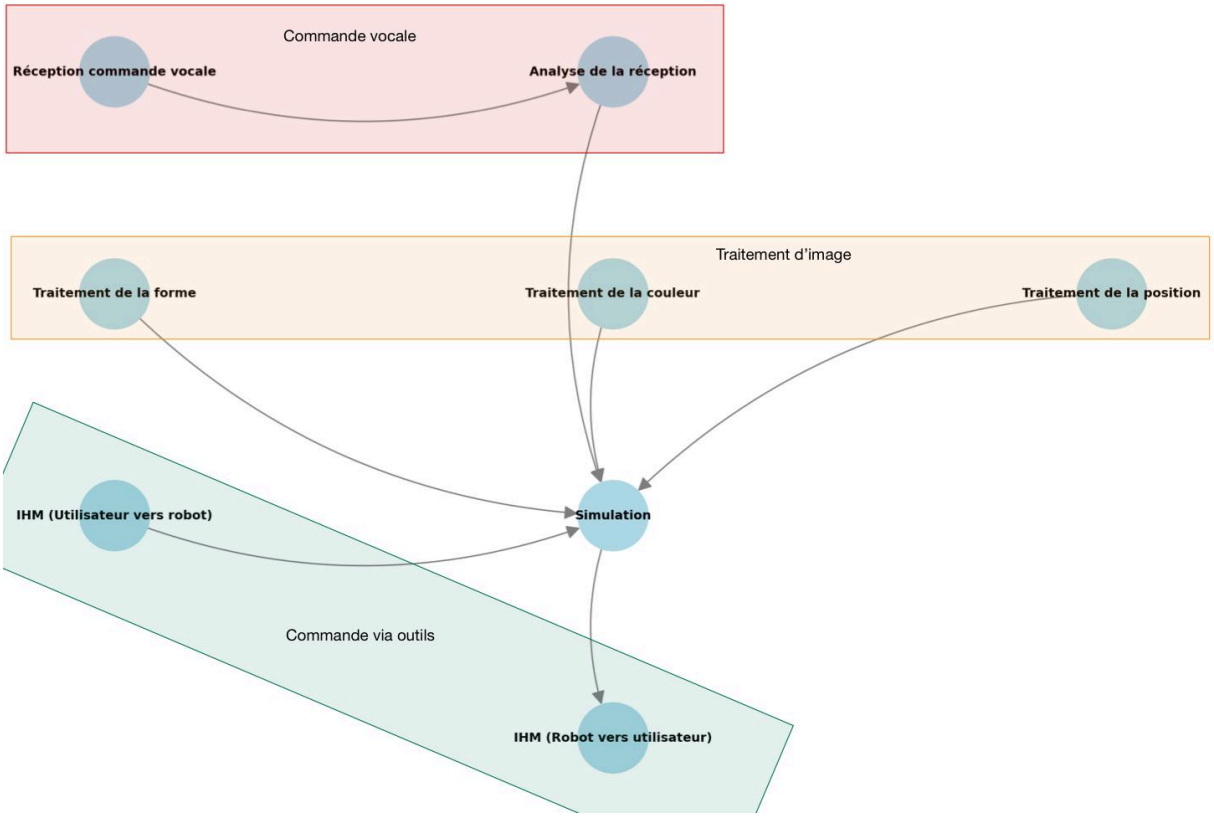


Figure 13 : schéma des fonctionnalitées

2.4 Détail des requêtes

A. Requêtes textes

Les requêtes textes correspondent à l'instruction vocale française ou anglaise et les requêtes commandées sous les sorties comprises, traduites et les ordres exécutables par la plateforme.

Requête-texte français	Requête-texte anglais	Requête-commande
Avance	Move forward	Go_forward(dist=default_distance)
Tourne à droite	Turn right	Turn_right(angle=default_angle)
Tourne à gauche	Turn left	Turn_left(angle=default_angle)
Recule	Backwards	Go_backwards(dist=default_distance)
Arrêt*	Stop	Stop

Tableau 2 : requêtes textes

Les requêtes pourront être suivies d'une valeur, par défaut en mètres pour les distances et des degrés pour les angles. Exemple : "Avance de 2 mètres" → Commande : "Go_forward(dist=2)"

Particularité pour la requête "arrêt" : elle est prioritaire sur n'importe quelle autre instruction. Lorsque le robot exécute une instruction, si la requête d'arrêt d'urgence est reçue alors il s'arrête : les activités en cours sont alors annulées…

Remarque : la requête "arrêt" sera développée optionnellement.

B. Autres eléments de requêtes

Requête-texte français	Requête-texte anglais	Requête-commande
Aller à	Go to	Go_to(shape=X,color=X)
Rouge	Red	color=Red
Bleu	Blue	color=Blue
Jaune*	Yellow	color=Yellow
Vert	Green	color=Green
Orange*	Orange	color=Orange
Cube	Cube	shape=Cube
Balle	Ball	shape=Ball

Tableau 3 : requêtes composées

Les requêtes composées permettent d'enrichir les commandes en spécifiant des paramètres comme la couleur et la forme des objets cibles (exemple : "aller à la balle rouge"). Ces options offrent une interaction plus précise et permettent au robot de mieux identifier et manipuler différents objets dans son environnement.

Remarque : la priorité est de traiter 3 couleurs (rouge, vert et bleu), les autres couleurs seront optionnelles. Et au moins deux forme (Cube et une balle)

2.5 Génération d'un fichier de log

Le système devra générer un fichier de log : il s'agit d'un journal qui stocke les résultats obtenus lors des opérations.

3. Exigences techniques

3.1 Environnement de développement

Langage C

Le langage C est choisi pour développer les fonctionnalités de bas niveau, notamment celles liées à l'interaction directe avec les capteurs et les actionneurs. En raison de sa rapidité d'exécution et de son faible niveau d'abstraction, le langage C permet une gestion précise des ressources et des délais, essentielle pour les fonctionnalités critiques du projet.

Langage Python
Python est utilisé pour la partie logicielle liée aux interactions utilisateur, notamment la reconnaissance vocale et la simulation visuelle. Grâce à sa flexibilité et à son riche écosystème de bibliothèques (Turtle), Python facilite l'intégration d'interfaces utilisateur, la gestion des commandes vocales, ainsi que la visualisation des données de déplacement et de simulation.

Environnement Unix/Linux
Le développement et l'exécution du projet se feront sous un environnement Unix/Linux, qui offre une grande stabilité et compatibilité avec les outils open-source utilisés dans le projet. De plus, cet environnement est particulièrement adapté aux systèmes embarqués, facilitant ainsi la transition vers la plateforme physique finale du projet.

3.2 Technologies et outils utilisés

Bibliothèque Turtle (Python)
Utilisée pour simuler les déplacements du robot sur une carte virtuelle, Turtle permet de visualiser les trajets en temps réel, facilitant les tests des algorithmes de navigation avant l'intégration sur la plateforme physique.

Socket (Python)
Cette bibliothèque assure la communication bidirectionnelle entre l'utilisateur et le robot, via Wi-Fi ou Bluetooth, permettant d'envoyer et recevoir des commandes et retours en temps réel.

Reconnaissance vocale
Ce module Python convertit les commandes vocales en texte, facilitant l'interprétation des instructions orales pour le pilotage du robot.

4. Contraintes et attentes

4.1 Contraintes de développement

Le développement du projet Fil Rouge est encadré par plusieurs contraintes techniques et méthodologiques visant à respecter les exigences de performance, de compatibilité et de modularité nécessaires pour atteindre les objectifs du projet.

1. Le projet utilise les langages imposés **C et Python** : C pour le traitement bas-niveau et la gestion des capteurs, et Python pour les interactions vocales et la simulation. L'environnement **Unix/Linux** est également requis pour garantir la compatibilité avec les outils open-source et les systèmes embarqués, essentiels pour le déploiement sur la plateforme robotique finale.

2. **Modularité et évolutivité du code**
   Pour permettre une intégration future dans la plateforme physique (PFR2), le développement doit être modulaire. Chaque fonctionnalité (commande vocale, traitement d'images, simulation) doit être encapsulée dans des modules indépendants et bien documentés. Cette modularité doit faciliter les tests, les mises à jour, ainsi que la réutilisation et l'extension du code pour d'autres applications ou futures évolutions.

4.2 Critères de validation

Pilotage via outil

La plateforme doit pouvoir être pilotée sans difficulté via un outil (joystick, IHM), voir **Figure 1.** Le tableau ci-dessous présente les différents éléments à tester. Il est à préciser que dans la colonne instructions nous avons mis les significations littérales des instructions que la plateforme reçoit. Par exemple, supposons que l'utilisateur contrôle la plateforme via un joystick et que la touche X permet d'avancer. Lors de l'appui sur la touche X, nous enverrons une instruction à la plateforme pour lui demander d'avancer.

Instructions	**Résultat**
avance	la plateforme avance
recule	la plateforme recule
tourne à gauche	la plateforme tourne à gauche
tourne à droite	la plateforme tourne à droite

Tableau 4 : instructions test du pilotage via outils

Fiabilité des commandes vocales

Le système doit interpréter et exécuter correctement les commandes vocales, à condition que la transcription automatique soit précise. Cette précision sera mesurée par la capacité du module à comprendre les instructions, garantissant ainsi une interaction fluide entre l'utilisateur et le robot. Etant donné que nous ne développons pas le bloc chargé de transcrire (transformer l'oral en écrit), nous testerons juste la capacité de notre programme à extraire les informations textuelles (utiles pour le contrôle de la plateforme).

Ci-dessous un tableau pour couvrir globalement les configurations possibles. Il est à noter que certaines instructions présentées dans le tableau ci -dessus seront testées optionnellement car peuvent ne pas être réalisées. Il s'agit en particulier des instructions complexes.

Instructions	**Éléments extrait**
Avance de 2 mètres	avance, 2, mètres
avance vers la balle rouge	avance, balle, rouge
Tourne à gauche et recule	gauche, recule
Bonjour comment tu vas	*"rien qui suscite une action"*
Contourne l'obstacle sans passer à côté de la balle rouge*	la plateforme contourne l'obstacle sans passé à côté de la balle rouge

Tableau 5 : instructions test du pilotage vocal

Remarque : gérer le contour d'un objet est optionnel*.

Précision de l'analyse des images

Le robot doit être capable d'identifier et de localiser les objets dans son environnement avec une précision élevée. Cette précision inclut la reconnaissance de la **forme (balle et cube) et de la couleur (rouge, vert, bleu)** des objets, qui doivent être **clairement nommés**. Par exemple, le système devra être en mesure de signaler qu'il y a « un cube bleu», garantissant ainsi que le robot distingue chaque objet individuellement et de manière fiable.

Simulation fonctionnelle et représentative (prévue en phase 2 du PFR1)

Les déplacements simulés du robot doivent être conformes aux attentes en termes de trajectoire et d'interactions avec l'environnement. La simulation doit représenter les distances et proportions de manière **proportionnelle à la réalité** pour permettre une interprétation visuelle fidèle de l'espace. Cette exactitude permet non seulement de tester le robot dans un environnement contrôlé, mais aussi de garantir que les actions simulées sont applicables dans le monde réel.

Il est important de préciser que dans lors de simulation nous ne prendrons pas forcément en compte l'évitement d'obstacle. Cette fonction sera optionnelle.

5. Livrables attendus

- Briques logicielles : code source des modules de commande vocale, de traitement d'images, et de simulation.
- Une interface textuelle sera également développée pour permettre le choix des exécutions de programme et des modes de fonctionnement. Cette interface, sous forme de menu, fonctionnera dans le terminal ou l'invité de commande, avec un affichage simple et fonctionnel, sans mise en forme esthétique.
- Documentation technique : spécifications des modules, guide d'installation et manuel d'utilisation.
- Rapport de tests : démonstration des fonctionnalités développées avec des exemples concrets (simulation des déplacements, interaction vocale, etc.).

6. Planning

6.1 Planning général

Le projet Fil Rouge est structuré en plusieurs étapes réparties tout au long du semestre. Ce planning permet de garantir que chaque fonctionnalité est développée, testée et documentée de manière cohérente et progressive, en respectant les délais fixés. Voici le détail des étapes et des livrables attendus à chaque phase.

1. Constitution des équipes et première rencontre avec le tuteur

- **Date limite** : 20 septembre 2024
- **Objectif** : Former des équipes hétérogènes de 5 à 6 étudiants, validées par l'équipe pédagogique. Cette étape comprend également une première rencontre avec le tuteur pour clarifier les objectifs du projet, les attentes, et la méthodologie de travail.
- **Livrable** : Liste des membres de l'équipe et validation de la composition par le tuteur.

2. Analyse du cahier des charges et préparation du dossier de spécifications

- **Période** : Du 18 septembre au 16 octobre 2024
- **Objectif** : Comprendre en profondeur le cahier des charges, analyser les fonctionnalités à développer, et préparer un dossier de spécifications détaillant chaque module, les contraintes techniques, et le périmètre fonctionnel.

- **Activités** : Sessions de travail en groupe pour décomposer le projet, séances de TD pour structurer le dossier de spécifications.
- **Livrable** : Dossier de spécifications complet, incluant la description des fonctionnalités, les contraintes techniques, le planning prévisionnel, et la répartition des tâches par membre de l'équipe.
- **Date limite de dépôt** : 18 octobre 2024

3. Décomposition en modules et début du développement

- **Période** : 4 novembre - 20 décembre 2024
- **Objectif** : Décomposer les fonctionnalités principales en modules indépendants et commencer le développement en C et Python. L'accent est mis sur la création des briques logicielles de base (commande vocale, traitement d'images, simulation).
- **Activités** : Codage des différents modules, test unitaire de chaque fonctionnalité, ajustements basés sur les premiers retours du tuteur.
- **Livrable intermédiaire** : Rapport d'avancement, décrivant l'état de développement de chaque module et les éventuels obstacles rencontrés.
- **Jalon intermédiaire** : 4 décembre 2024 (séance TD pour présenter l'état d'avancement et recevoir des conseils sur la validation)

4. Tests et intégration des modules

- **Période** : 6 janvier - 13 janvier 2025
- **Objectif** : Assembler les différentes briques logicielles, effectuer des tests de compatibilité entre modules, et s'assurer que les fonctionnalités se comportent comme prévu. Ce jalon permet de vérifier l'intégration et la coopération entre les modules de commande vocale, traitement d'images, et simulation.
- **Activités** : Tests d'intégration, correction des erreurs identifiées, ajustements finaux.
- **Livrable** : Rapport de test et de validation interne, contenant les résultats des tests et les ajustements apportés aux modules.

5. Finalisation du projet et préparation des livrables finaux

- **Période** : 13 janvier - 24 janvier 2025
- **Objectif** : Finaliser tous les aspects du projet, s'assurer que la documentation technique est complète, et préparer la démonstration scénarisée des fonctionnalités.
- **Activités** : Préparation de la démonstration, vérification de la qualité du code, organisation des livrables finaux.
- **Livrables finaux** :
    - **Rapport global** : Un document résumant l'ensemble des étapes du projet, les décisions techniques, les résultats obtenus, et un bilan individuel des contributions.
    - **Dépôt du code** : Code structuré, lisible et commenté, déposé avec un manuel d'utilisation pour faciliter la prise en main.

- **Démonstration scénarisée** : Présentation fonctionnelle des briques logicielles développées, simulant une interaction complète entre l'utilisateur et le robot via les commandes vocales, le traitement d'images, et la simulation de déplacements.

- **Date de validation et dépôt final** : 24 janvier 2025 (à confirmer avec l'équipe pédagogique)

6.2 Gantt

Pour compléter cette organisation, nous avons réparti les responsabilités et les tâches entre les équipes comme suit. Les équipes ne sont pas fixes, et il est possible que des changements de membres soient envisagés pour optimiser le travail en fonction des besoins du projet.

Équipe 1 (Jaune) : Composée de **Adji Aram SECK** et **TAYOU MBEDE Ryan Niel** , cette équipe se concentre sur le traitement des aspects visuels :

- **Traitement de la forme** : identification et analyse de certaines formes dans l'environnement.
- **Traitement de la couleur** : reconnaissance et gestion des informations colorimétriques.
- **Traitement de la position** : localisation des objets pour une meilleure interaction avec le système.

Équipe 2 (Orange) : Composée de **Djibril BA** et **Hela BAKHTI**, cette équipe se spécialise dans l'interaction vocale et la création d'interfaces utilisateur et robot :

- **Réception des commandes vocales** : gestion des commandes transmises vocalement pour interaction avec le robot.
- **Analyse de la réception** : extraction et traitement des données de commande.
- **IHM (Utilisateur vers robot)** : mise en place de l'interface utilisateur pour l'envoi de commandes au robot.
- **IHM (Robot vers utilisateur)** : développement de l'interface pour que le robot envoie des informations vers l'utilisateur.

Équipe 3 (Rouge) : Composée de **Imran WACHILL** et **Wissal MEHREZ**, cette équipe se charge de la simulation et de l'intégration du système :

- **Simulation** : vérification des fonctions développées par les autres équipes dans un environnement simulé pour assurer la qualité des interactions.
- **Intégration du système** : regroupement et coordination des fonctionnalités développées pour garantir une parfaite cohésion.

Avec cette approche flexible et cette répartition des tâches, nous nous assurons que chaque aspect du projet progresse en parallèle, tout en permettant une entraide et une adaptation en cas de besoin. Cette organisation collaborative maximise la productivité et permet de tester et intégrer les différentes fonctions de façon fluide et structurée.

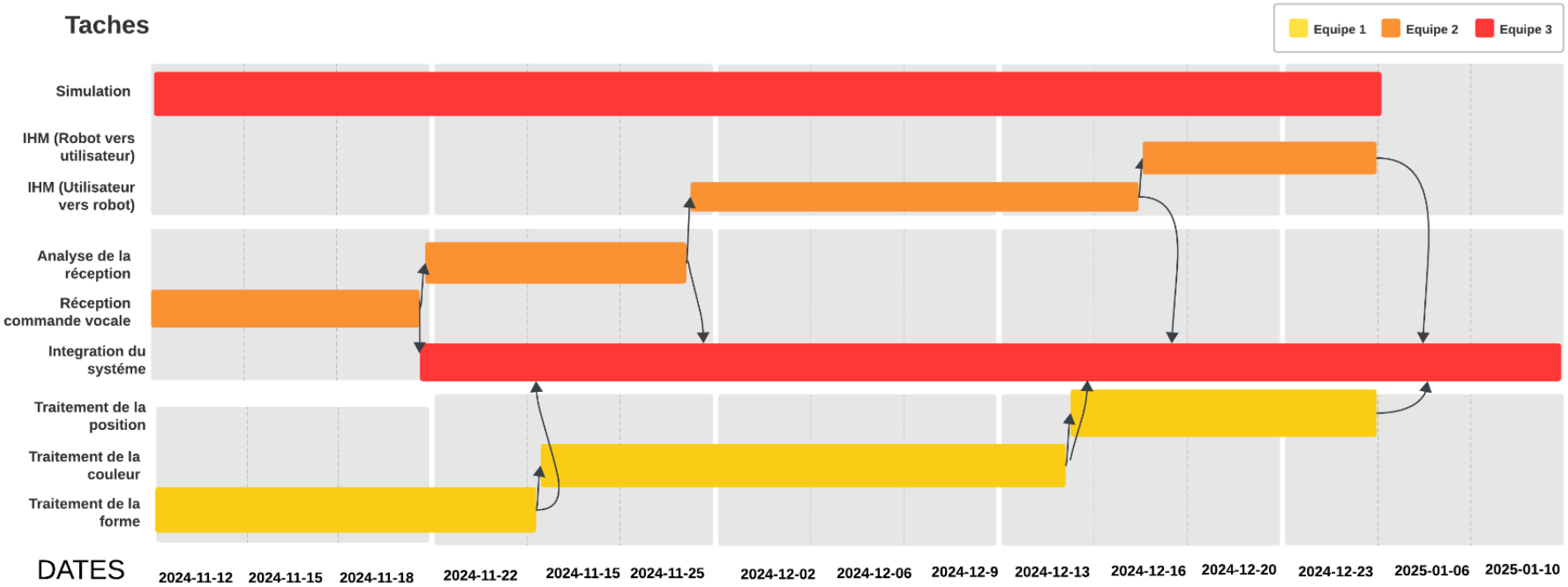


Figure 14 : Diagramme Gantt

7. Conclusion

Ce cahier des charges vise à aligner les objectifs et les attentes du client et de l'équipe projet pour garantir le bon déroulement du projet Fil Rouge. Il décrit les fonctionnalités à développer, les contraintes techniques et les livrables attendus, et servira de référence tout au long du projet. Toute modification devra être validée par le client.