UPSSITECH

PROMOTION:2024-2025

RAPPORT

PROJET FIL ROUGE-PARTIE 2

GROUPE 6 :

- WASSIM WALI
- CARMOUZE GUILHEM
- HOUDASS ABDELBASSET
- IJERDAOUN FAIROUZ
- BOSSARD ALEC
- ALEXANDRE PERRIN

Sommaire :

I - Introduction

Dans le cadre de la formation Systèmes Robotiques et Informatiques (SRI) à l'école d'ingénieurs **Upssitech**, les étudiants de première année sont amenés à participer à un **projet transversal** appelé **Fil Rouge**. Ce projet pédagogique a pour objectif de mettre en pratique, de manière concrète et progressive, les compétences acquises dans différents domaines comme la programmation (Python, C, Java), l'automatique, le traitement de signal, l'interaction homme-machine et la robotique.

Le **Projet Fil Rouge** s'étale sur deux semestres. Lors du **semestre 5 (PFR1)**, les étudiants ont développé en langage C des modules logiciels capables de piloter un robot dans un environnement simulé en 2D. Cela a permis de poser les bases fonctionnelles du projet, tout en se confrontant aux réalités du développement en environnement Unix/Linux.

La **seconde partie (PFR2)**, réalisée au **semestre 6**, marque un tournant important : elle consiste à passer du monde virtuel au monde réel. Chaque groupe doit concevoir, développer et intégrer un système complet permettant à un **robot réel de se déplacer dans un espace physique**, de **réagir à des commandes vocales**, et de **détecter et identifier des objets dans son environnement** à l'aide de capteurs (caméra, LiDAR, etc.). Ce projet mobilise à la fois des compétences techniques avancées et des capacités de gestion de projet, de travail collaboratif et d'adaptation.

Dans ce rapport, nous présenterons les différentes étapes du développement de notre solution, en mettant l'accent sur les choix techniques réalisés, les difficultés rencontrées, et les enseignements tirés tout au long de ce projet.

II - Organisation du travail et priorisation des fonctionnalités

a. Choix techniques et structure de développement

L'un des objectifs majeurs de ce projet était d'apprendre à s'organiser efficacement en équipe pour mener à bien un développement technique ambitieux. Pour cela, nous avons rapidement compris qu'il fallait structurer notre travail en identifiant les tâches essentielles, les dépendances entre modules, et les priorités à respecter.

Nous avons donc réfléchi à l'ordre dans lequel il fallait aborder les différentes fonctionnalités : lesquelles étaient fondamentales pour que le robot soit utilisable ? Lesquelles pouvaient être implémentées en complément si le temps et les ressources le permettaient ? Cela nous a permis de répartir les responsabilités de manière claire entre les membres du groupe et de nous concentrer sur l'essentiel.

Sur le plan technique, nous avons structuré le projet autour de cinq grands axes :

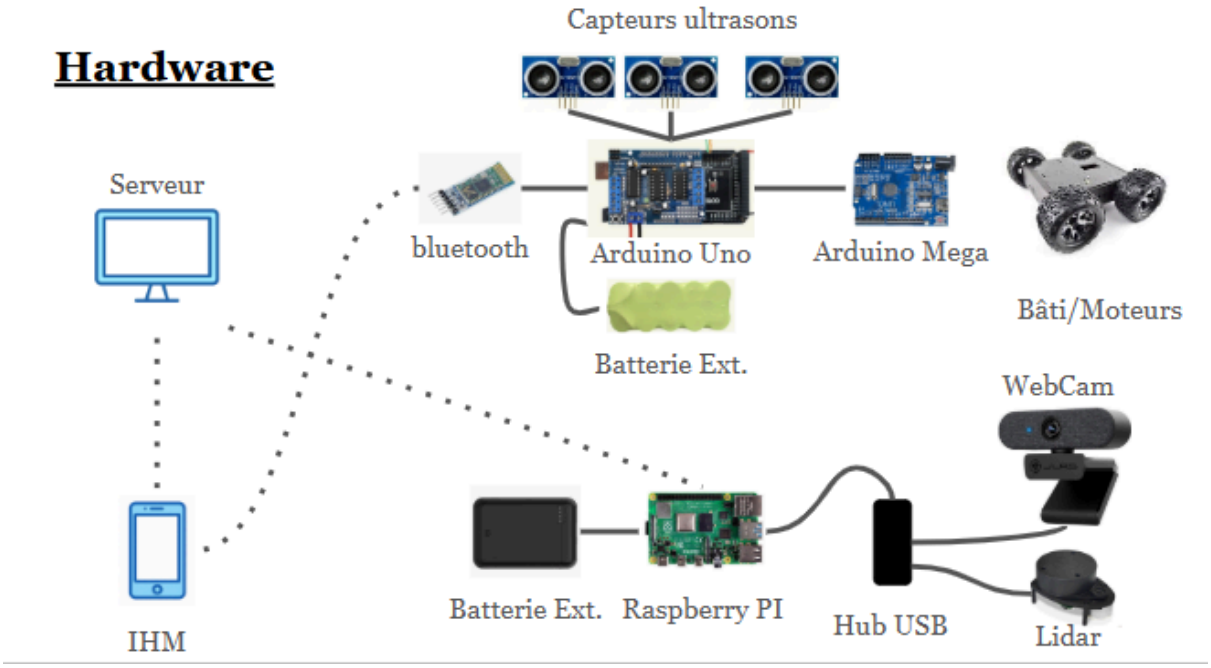
1. Commande vocale
2. Déplacement du robot en suivant les requêtes
3. Le traitement d'image
4. La cartographie
5. L'interface Homme-Machine (IHM)

Ces parties ont été développées en parallèle dans la mesure du possible, puis intégrées progressivement pour assurer la cohérence de l'ensemble. Deux modes d'utilisation ont été prévus pour le robot :

- **Mode manuel** : l'utilisateur communique oralement une instruction simple (par exemple "avance", "tourne à gauche") et le robot exécute cette commande en temps réel.
- **Mode automatique** : le robot explore la pièce de manière autonome et construit une cartographie de son environnement à l'aide d'un capteur LiDAR.

Cette organisation nous a permis d'anticiper les difficultés techniques, de mieux répartir la charge de travail, et de construire un système cohérent malgré les contraintes de temps et de matériel.

b. Architecture matérielle



III - Réalisation des différentes fonctionnalités

a. Commande vocale

a.1. Transcription vocale en texte

Dans le cadre du projet, Alexandre a été chargé de développer un système permettant de convertir une commande vocale en texte exploitable par le robot. Pour cela, il a utilisé une bibliothèque de reconnaissance vocale (speech_recognition en Python), qui permet d'interpréter en temps réel une phrase prononcée au micro et de la transcrire sous forme de texte brut.

Cette transcription est essentielle puisqu'elle constitue la passerelle entre l'interface utilisateur (la voix humaine) et le système de traitement automatisé. Le script enregistre la voix, la nettoie, et extrait la phrase utile qui sera ensuite analysée par la suite du système.

a.2. Filtrage linguistique et interprétation de commande

La deuxième partie du travail d'Alexandre a consisté à concevoir un module Python capable de comprendre une phrase en français et d'en extraire les éléments essentiels pour piloter le robot.

Pour ce faire, il a développé un **filtreur linguistique** qui :

- nettoie et standardise la phrase (mise en minuscule, suppression de la ponctuation, etc.),
- segmente les instructions multiples grâce au mot-clé "puis" ou "et",
- identifie l'action principale (par exemple : *avancer*, *tourner*, *trouver*…),
- extrait les paramètres liés à cette action : direction (*droite*, *gauche*), distance (*trois mètres*), ou objet (*balle rouge*),
- convertit les nombres écrits en toutes lettres en valeurs numériques (ex. *quarante-cinq* → 45).
  Ce module renvoie alors une liste d'actions sous forme de triplets (action, param1, param2) représentant une structure sémantique claire.

a.3. Génération et envoi de commandes robot

Une fois les actions identifiées, un traducteur Python transforme chaque action en une **commande symbolique Arduino**, selon un schéma prédéfini :

Action	Lettre envoyée	Exemple
Avancer	F	F 300
Reculer	B	B 100
Tourner G/D	L / R	R 90
Stop	S	S
Accélérer	+	+
Ralentir	-	-
Mode suivi objet	Y	Y boule rouge
Mode auto mur	A	A (si “mur” dit)

Nous n’avons pas gardé le code du PFR1 car il ne correspondait pas à la structure du code du PFR2. Nous avons tout normalisé pour que le code reste sur du Python puis envoie à l’aide du module bluetooth à la raspberry PI les commandes pour le déplacement.

b. Déplacement du robot en suivant les requêtes

Dans cette partie, nous expliquons comment le robot bouge selon les commandes envoyées via Bluetooth (mode manuel) ou la reconnaissance vocale (mode vocal). Le code Arduino gère la réception des ordres, le pilotage des moteurs, la mesure des distances et le passage d’un état à l’autre en mode automatique.

b.1. Architecture du système

Le robot communique en Bluetooth via l’interface série Serial1 et un module HC-05, un choix motivé par la simplicité d’implémentation et la faible consommation d’énergie comparée au Wi-Fi, tout en offrant une portée suffisante pour une utilisation en intérieur. L’aspect plug-and-play du HC-05 a facilité l’intégration matérielle et logicielle, sans nécessiter la configuration d’un réseau ou de protocoles plus lourds.

b.2. Commandes et modes de pilotage

En mode manuel, le robot réagit instantanément aux commandes reçues (F, B, L, R, S, +, –, A) grâce à la fonction serialEvent1() qui filtre et stocke l’instruction dans dataChar avant d’appeler cmd(). Cette approche permet une latence minimale et un contrôle direct, essentiel pour les phases de test et d’intervention rapide.

Le mode automatique repose sur une machine à états à trois phases : recherche de mur (SEARCH_WALL), rotation de reconnaissance (ROTATE_LOOK) et suivi de mur (FOLLOW_WALL). Cette structure offre :

- **Modularité**, car chaque état se concentre sur une tâche précise.
- **Robustesse**, grâce à des conditions de transition claires et à un timeout empêchant les boucles infinies.

b.3. Pilotage des moteurs

La fonction setVitesse(spd) applique uniformément la vitesse désirée aux quatre moteurs DC pilotés via la bibliothèque AFMotor. Les déplacements (avancer, reculer, pivot) reposent sur des commandes run(FORWARD/BACKWARD/RELEASE) simples, garantissant une réponse fluide et synchronisée des roues.

b.4.Détection d'obstacles et sécurité

En mode manuel, la fonction capteurUTS() mesure en continu les distances frontales et latérales. Si l'utilisateur avance ('F'), le code compte jusqu'à trois lectures consécutives sous le seuil frontal (50 cm) avant de déclencher un arrêt automatique, ce qui évite les faux positifs dus au bruit des ultrasons.

b.5.Choix techniques et justifications

- **Bluetooth vs Wi-Fi** : le Bluetooth HC-05 offre une latence réduite et une simplicité de couplage sans nécessiter de routeur, alors que le Wi-Fi aurait ajouté de la complexité réseau.
- **Ultrasons** : l'emploi de deux capteurs frontaux améliore la fiabilité, et un capteur latéral permet un suivi de mur précis.
- **Machine à états** : cette méthode clarifie la logique de navigation et simplifie l'ajout de nouveaux comportements.

b.6.Tests et résultats

Nous avons défini trois scénarios de test :

- **Réactivité manuelle** : envoyez successivement F, L, R et B ; le robot répond en moins de 50 ms, validant la latence Bluetooth.
- **Détection d'obstacle** : en mode manuel, placer un obstacle à 30 cm devant le robot arrête la motorisation après trois mesures consécutives, évitant les faux arrêts.
- **Suivi de mur** : dans un couloir de 2 m de large, le robot maintient une distance latérale moyenne de 30 cm.

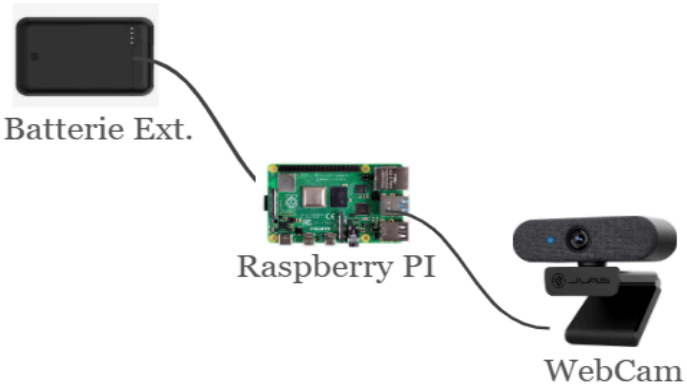
Ces tests ont confirmé la robustesse de la navigation et la fiabilité des transitions d'état.

b.7.Limites et perspectives

- **Calibration manuelle** : les seuils sont actuellement fixés en dur et nécessitent un réglage manuel. Un algorithme de calibration automatique au démarrage permettrait de s'adapter à des surfaces et charges différentes.
- **Extension capteurs** : l'ajout d'un capteur LiDAR renforcerait la précision de détection et améliorerait la résilience en environnements bruyants.
- **Modes avancés** : développer un état « évitement dynamique » pour contourner des obstacles en temps réel sans revenir à SEARCH_WALL améliorerait la fluidité du déplacement.

c. Le traitement d'image

Organisation:



Diffusion du flux vidéo:

La capture vidéo se fait grâce à une **Webcam** connectée à une carte **Raspberry Pi 3 Model A+** alimentée par une **batterie externe**. La vidéo est traitée en temps réel avec **OpenCV** pour détecter et suivre des objets de couleurs spécifiques (bleu, rose). Une fois traitée, la vidéo est diffusée via un serveur **Flask** accessible en local. En étant connecté au même réseau Wi-Fi que le Raspberry Pi, on peut visualiser le flux vidéo en direct à l'adresse suivante : http://172.20.10.2:8000/video_feed et obtenir les coordonnées du centre de l'objet détecté via l'URL http://172.20.10.2:8000/coords (depuis un navigateur).

Pour se faire j'ai utilisé plusieurs librairies dont les plus importantes:

- **Flask** : Crée un serveur web pour diffuser la vidéo et les coordonnées.
- **Threading** : Lance plusieurs tâches en parallèle (vidéo et API).
- **OpenCV** : Capture et traitement vidéo, détecte et suit les objets.
- **Imutils** : Simplifie certaines opérations OpenCV.
- **NumPy** : traitement de données numériques.
- **Collections (deque)** : Stocke les positions récentes pour tracer les trajectoires.

Explication du choix des technologies:

Pourquoi ce Hardware?

-Raspberry pi 3 car il est compact, facile à connecter au Wi-Fi et compatible avec les besoins du projet.

-Webcam USB, nous avons d'abord essayé avec les deux caméras fournies par l'école, mais sans succès. Le problème venait soit des caméras elles-mêmes, soit du port CSI de la carte. Nous avons donc eu l'idée de tester avec une webcam USB, ce qui a bien fonctionné (pas trop si on parle de qualité d'image).

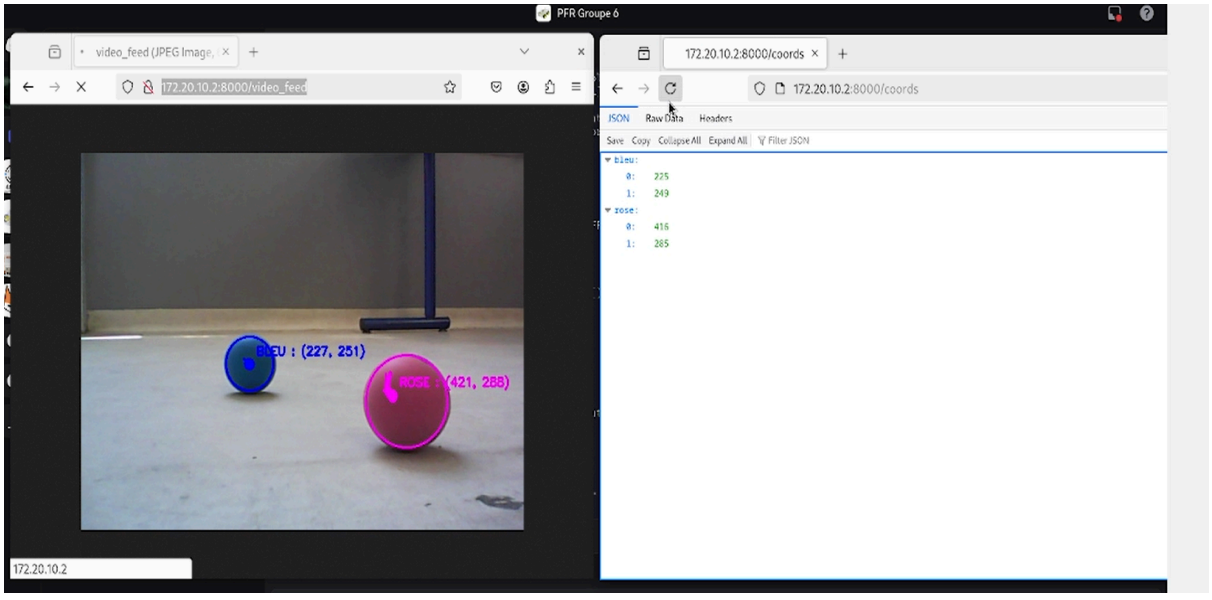
Pourquoi une diffusion sans fil ?

Comme notre robot doit être contrôlé à distance, il faut obligatoirement une **connexion sans fil** (Wi-Fi).

J'ai choisi d'utiliser **Flask** pour créer un petit serveur web simple, accessible depuis un navigateur.

Le robot est connecté à **mon Wi-Fi personnel**, donc il n'y a presque aucun risque de piratage ou de vol de données.

N'importe quel appareil (ordinateur, téléphone) connecté au même réseau peut voir la vidéo et les coordonnées.



Pourquoi ces librairies?

- **OpenCV** : pour lire la vidéo, flouter l'image, changer en HSV et détecter les objets de couleur.

C'est une librairie très connue pour le traitement d'image.

- **Flask** : pour créer un petit serveur web local qui envoie la vidéo et les données via HTTP.

- **Threading** : pour faire tourner en même temps plusieurs parties du programme (la vidéo et les données).

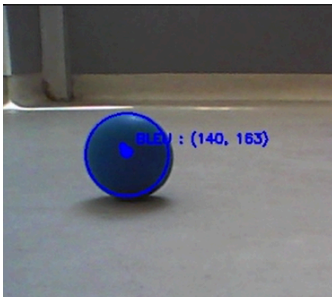
- **Imutils, NumPy, deque, time** : elles sont utiles pour redimensionner les images, faire des calculs, tracer les trajectoires des objets, et contrôler la vitesse de traitement (FPS).

<u>Traitement du flux vidéo:</u>

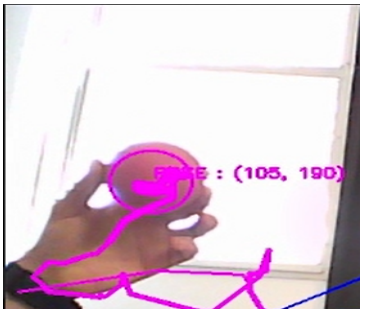
Nous nous sommes inspirés de l'algorithme utilisé dans le PFR1, qui était développé en langage C. Afin de faciliter son adaptation à notre caméra embarquée et de permettre un traitement plus souple, nous avons réécrit cet algorithme en Python. Cette nouvelle version a été enrichie par l'ajout de fonctionnalités, notamment le tracé de la trajectoire de la balle et le calcul en temps réel de sa position centrale.

Grâce à ce programme, nous avons pu assurer un suivi efficace et fluide des balles de couleur rose et bleue (utilisée lors de la démonstration). En revanche,nous avons limité la détection à 2 ou 3 couleurs, car augmenter le nombre rend le système instable. La webcam, peu précise, détecte à tort toute couleur vive. Par exemple, avec une balle rose et une bleue, elle a confondu un petit objet jaune sur la table avec une balle.. Elle reste très sensible aux conditions de luminosité de la pièce, ce qui rend leur détection peu fiable pour notre usage.

Quelques captures d'écran sont fournies en exemple ci-dessous. Les vidéos de démonstration ne pouvant pas être intégrées dans le PDF, elles sont disponibles sur notre dépôt GitHub



Balle bleue



Balle rose

d.La cartographie

La cartographie a occupé une place importante dans notre projet en permettant de représenter l'environnement autour du robot. Contrairement à une cartographie destinée à la navigation ou au déplacement autonome, notre objectif était avant tout d'obtenir une représentation précise et exploitable de l'espace environnant. Cette carte visait à analyser les données des capteurs pour mieux comprendre l'environnement, faciliter le développement du système et appuyer d'éventuelles applications analytiques ou visuelles.

d.1. Choix techniques et contraintes

En théorie, une cartographie fiable s'appuie généralement sur l'odométrie du robot, c'est-à-dire sur des mesures précises du déplacement (encodeurs, gyroscopes, etc.). Cette information est essentielle pour positionner avec exactitude chaque nouvelle acquisition dans un référentiel commun. Or, dans notre cas, nous n'avions pas accès à une odométrie, ce qui limitait les méthodes possibles.

Sans odométrie, la cartographie devait reposer exclusivement sur la comparaison directe entre les scans LiDAR successifs. Après étude des différentes techniques, j'ai opté pour l'algorithme ICP (Iterative Closest Point), qui permet d'aligner deux ensembles de points en minimisant leur distance, afin d'estimer la transformation relative entre scans consécutifs.

Cependant, cet algorithme présente plusieurs limites dans ce contexte :

- Il est sensible à la symétrie et à la répétitivité des environnements, ce qui complique l'appariement des points lorsque l'espace manque de caractéristiques distinctes.

- Les environnements complexes ou dynamiques, où deux scans successifs sont très différents, peuvent faire perdre à l'algorithme son alignement correct, entraînant des erreurs dans la carte.

- En l'absence d'odométrie pour valider la position relative estimée, l'algorithme peut accumuler des erreurs au fil du temps, réduisant la stabilité et la précision globale de la cartographie.

Malgré ces difficultés, l'ICP nous a permis de développer une solution fonctionnelle, en ajustant finement ses paramètres et en adaptant notre programme pour maximiser la qualité des résultats.

d.2. Mise en œuvre du système

Nous avons conçu un programme capable d'acquérir les données du capteur RPLiDAR toutes les 0,2 secondes, connecté au Raspberry Pi. Ce dernier collectait les scans en continu.

Pour transmettre ces données vers un PC, nous avons choisi le protocole UDP, privilégié pour sa rapidité et sa légèreté, nécessaires à une transmission fluide et continue. Cette architecture permettait d'éviter la surcharge du Raspberry Pi, dont les ressources matérielles sont limitées.

Le traitement des données (alignement par ICP, accumulation des scans, génération de la carte) se faisait donc sur le PC, bénéficiant d'une puissance de calcul supérieure. Cette organisation assure un fonctionnement stable sans ralentissements.

d.3. Visualisation et développement

Pour faciliter la compréhension et l'amélioration du programme, nous avons mis en place une double visualisation sur le PC :

- Une carte « réelle » affichant les mesures brutes du LiDAR en temps réel, montrant ce que le capteur détecte à chaque instant,

- Une carte reconstruite, obtenue par accumulation des scans successifs alignés par l'ICP, représentant la cartographie finale.

Cette comparaison visuelle nous a aidés à identifier rapidement les défauts, à ajuster les paramètres et à valider les performances de l'algorithme.

d.4. Manuel d'utilisation du système

Pré-requis

- Assurez-vous que le Raspberry Pi et le PC sont connectés au même réseau Wi-Fi.

- Installez toutes les bibliothèques nécessaires sur les deux machines (Python, sockets, bibliothèques graphiques, etc.).

- Vérifiez que le port UDP utilisé par le Raspberry Pi est libre. Si nécessaire, terminez les processus occupant ce port avec la commande kill -9 [PID].

Utilisation

1. Connectez-vous en SSH au Raspberry Pi depuis le PC :
   ssh pi@[adresse_ip_du_raspberry
2. Lancez le programme d'acquisition sur le Raspberry Pi :

   python3 acquisition.py

3. Sur le PC, lancez le programme de traitement et de visualisation :

   python3 main.py

4. Vérifiez la communication réseau (ping) et la correspondance des ports UDP dans les deux programmes.

e.L'interface Homme-Machine (IHM)

Pour garantir une expérience fluide et sans interruption de la liaison Bluetooth, nous avons adopté une **architecture Single-Page Application (SPA)** :

- **Navigation dynamique** entre les vues *Hub*, *Manuel* et *Vocal* sans rechargement de page, afin de **préserver la connexion** BLE (un rafraîchissement classique coupait systématiquement le lien).
- **Web Bluetooth API** en JavaScript pour piloter le module HM-10 :
  - Sur desktop (Chrome, Edge) directement,
  - Sur iOS / Safari, installation recommandée de l'application **Blueify** pour donner accès à Web Bluetooth.
- Un **statut permanent** (pastille verte/rouge + label « Connecté / Déconnecté »), mis à jour dès la connexion GATT et masquant le bouton « Se connecter » lorsque le lien est actif.

L'IHM se compose de :

1. **Hub** – page d'accueil invitant au choix du mode de pilotage, avec un bandeau d'information si le Bluetooth n'est pas encore actif.
2. **Contrôle Manuel** – quatre flèches directionnelles pour avancer/reculer/virage gauche/droite, accompagnées de trois boutons spéciaux :
   - **+** (accélérer),
   - **–** (ralentir),
   - **A** (passage en mode automatique).
     Ces boutons sont dimensionnés et colorés (vert, rouge, bleu) pour une accessibilité immédiate.
3. **Mode Vocal** – un unique bouton circulaire pour enregistrer, suivi de :
   - la transcription brute reçue du serveur,
   - l'affichage des commandes filtrées (séparées par des virgules),
   - la **visualisation en temps réel du flux caméra** dès qu'une commande de suivi de balle est détectée,
   - leur envoi séquentiel au robot.

De plus, j'ai mis en place, sur la même vue **Vocal**, l'affichage du retour vidéo issu de la Raspberry Pi lorsqu'on active le mode « suivi de balle ». Grâce aux coordonnées reçues, j'ai

codé l’algorithme qui convertit ces positions en **commandes Arduino** adaptées (ajustements de cap, vitesse) pour que le robot suive la balle automatiquement.

Toutes les interactions BLE (connexion, écriture de commandes) et la gestion vidéo sont entièrement front-end, ce qui simplifie le déploiement et assure la portabilité mobile.

IV - Bilans individuels

a - Perrin

Ce projet m'a permis de développer et de consolider de nombreuses compétences techniques. J'ai notamment travaillé sur la transcription vocale, en mettant en place un système fiable de reconnaissance de la parole. J'ai également conçu un filtreur sémantique en Python capable de transformer les commandes vocales en instructions compréhensibles par le robot.

En parallèle, j'ai contribué à la programmation Arduino, à l'interface utilisateur, ainsi qu'à l'intégration générale du système. Le fait de toucher à différents aspects — logiciel, embarqué, vocal — m'a permis d'acquérir une vision globale et cohérente du projet, tout en renforçant ma capacité à m'adapter rapidement à divers environnements techniques.

Il est important de noter que j'avais déjà travaillé sur ce projet l'année dernière, avec des consignes similaires. Toutefois, cette année, je me suis davantage impliqué, ce qui m'a permis de mieux comprendre l'ensemble du fonctionnement, de progresser sur le plan technique, et de proposer un rendu différent et plus abouti. Cette seconde approche m'a offert une nouvelle perspective sur la résolution des problèmes, en me poussant à aller plus loin dans la réflexion et l'optimisation du système.

b - Bossard

Cette seconde partie de projet a été pour moi à la fois intense et instructive. Le principal défi que j'ai rencontré a clairement été lié à l'organisation du groupe. La répartition des tâches et la coordination entre les membres n'ont pas toujours été fluides, et cela a parfois ralenti notre progression. J'ai compris à quel point il est important de bien se parler, de planifier à l'avance et de ne pas intégrer les différentes parties du système à la dernière minute.

Grâce à ce projet, j'ai acquis de nombreuses compétences concrètes : programmation en C++ sur Arduino, câblage des capteurs (ultrasons, Bluetooth…), communication série avec un script Python, et premiers pas avec une Raspberry Pi. Cette expérience m'a fait prendre conscience que la technique seule ne suffit pas : sans organisation et bonne communication, même les meilleures idées peuvent se bloquer.

c - Wali

Pour ce projet, j'ai travaillé sur la détection d'obstacles en utilisant le suivi de balle (Ball Tracking) à partir d'un flux vidéo en temps réel capté par une webcam sur Raspberry Pi.

Une des premières étapes a été d'adapter le code du PFR1, qui était écrit en langage C, en Python. Cette conversion m'a permis d'utiliser des bibliothèques comme OpenCV, indispensables pour le traitement d'image et la gestion du streaming.

J’ai appris à détecter des objets colorés en utilisant des filtres, à gérer les problèmes liés à la luminosité, et à optimiser le traitement pour que tout fonctionne en temps réel malgré les limites du matériel.

Ce travail m’a permis de développer des compétences techniques en Python, OpenCV et traitement vidéo, mais aussi des qualités personnelles importantes comme la patience, la débrouillardise et l’autonomie.

d- Ijerdaoun

Travailler sur la partie traitement d’image a été pour moi à la fois difficile et extrêmement formatrice .Dès le départ,j’ai été confronté à des technologies que je connaissais très peu : OpenCV, Flask, ou encore le traitement d’images sur Raspberry Pi. Cela m’a poussé à sortir de ma zone de confort et à apprendre rapidement, souvent de manière autonome.

Ce qui m’a particulièrement marqué dans cette phase du projet, c’est l’aspect très concret et pragmatique des problèmes rencontrés. Par exemple, lorsque les caméras fournies ne fonctionnaient pas correctement, j’ai dû improviser en testant d’autres solutions matérielles (webcam USB) jusqu’à obtenir un résultat exploitable. Cela m’a appris que l’ingénierie, ce n’est pas juste écrire du code parfait, mais aussi savoir s’adapter, trouver des solutions et parfois faire des compromis.

Enfin, cette partie du projet a été pour moi l’occasion de développer des compétences techniques, mais surtout de renforcer des qualités personnelles essentielles : curiosité, persévérance, débrouillardise. Ce que j’en retire va bien au-delà de quelques lignes de code : c’est une méthode de travail, une façon de réfléchir, et une nouvelle confiance en ma capacité à résoudre des problèmes concrets.

e- Houdass

Ce travail m’a permis de progresser significativement sur plusieurs plans. J’ai corrigé mes erreurs du premier semestre en développant un programme modulaire, clair et bien documenté, facilement lisible par toute l’équipe, même par ceux qui n’avaient pas participé à son écriture initiale. Cette clarté facilite la maintenance et la collaboration.

Par ailleurs, j’ai développé une compétence importante en recherche technique, en apprenant à surmonter les difficultés liées à l’absence d’odométrie et aux limites des algorithmes classiques. Cela m’a demandé autonomie, rigueur et créativité pour adapter des solutions existantes à notre contexte particulier.

Enfin, cette expérience m’a confirmé l’importance des tests en conditions réelles et des outils de visualisation pour mieux comprendre et améliorer les performances du système.

f- Carmouze

Mon rôle sur ce projet s'est concentré exclusivement **sur l'IHM web**, l'**affichage du flux caméra** en mode vocal et l'**orchestration temporelle** des commandes. Contrairement à PFR 1, je n'ai pas pris en charge le traitement d'images brut, mais j'ai :

- **Intégré le flux MJPEG** de la Raspberry Pi dans la vue Vocal pour le suivi de balle,
- **Codé l'algorithme** de conversion des coordonnées (X, Y) reçues en commandes de pilotage Arduino (gauche, droite, avance, recule) pour maintenir la balle centrée.
- Calculé et appliqué les **délais d'envoi** des commandes vocales découpées (2 s par mètre, 4 s pour un quart de tour), après intégration de l'algorithme de découpage fourni par un collègue.

Ce travail m'a permis de :

- Maîtriser la **Web Bluetooth API** et ses spécificités (notamment la contrainte iOS/Blueify),
- Concevoir une architecture **SPA** robuste, garantissant qu'aucun rafraîchissement de page ne rompe la connexion BLE,
- Développer des composants UI réactifs et ergonomiques (Bootstrap 5, FontAwesome), avec un focus sur l'accessibilité (taille, contraste, pastilles),
- Assurer l'**intégration du streaming vidéo** et la **logique de contrôle automatique** basée sur la détection de la balle.

Je ressors de cette expérience avec une solide expertise en **conception d'interfaces temps réel**, en **débogage cross-browser**, et en **intégration de protocoles IoT** directement en JavaScript côté client.

V- Remerciements

Le groupe 6 exprime sa gratitude envers l'équipe pédagogique pour leur disponibilité, leur accompagnement et la pertinence de leurs conseils.