Université Euromed de Fès

École d'Ingénierie Digitale et Intelligence Artificielle (EIDIA)

Cahier des Charges du Projet

SerbAI

Thématique :

Robot serveur intelligent utilisant ROS2, SLAM, Java et conception 3D.

Présenté par :

Hasnae AHOUZI

Étudiante en 2ème année du cycle ingénieur en Robotique et Cobotique

Encadrant :

Dr. Mostafa MRABTI

Dr. Hamza MOUNCIF

Prof. Nilam EL AMRANI

Prof. Zineb TADLAOUI

Année académique : 2025 – 2026

Table des matières

1 Problématique

Dans les environnements tels que les restaurants, hôpitaux, campus universitaires ou hôtels, le manque de main-d'œuvre pour des tâches répétitives (livraison, transport d'objets, service à table) se fait ressentir. Ces activités mobilisent le personnel pour des missions simples, au détriment de leur productivité et parfois de leur santé.

SerbAI vise à répondre à cette problématique en développant un robot serveur autonome, capable de :

— Naviguer de façon autonome dans un espace intérieur complexe.
— Identifier et atteindre différentes zones de service.
— Communiquer avec une application mobile pour recevoir des ordres.

2 Benchmark technologique

— **BellaBot (Pudu Robotics)** : design ergonomique, interaction vocale, navigation laser.
— **Servi (Bear Robotics)** : robot collaboratif de service avec SLAM 2D/3D.
— **Segway DeliveryBot S2** : navigation intelligente et multi-plateaux.
— **Dasher (Kody Robots)** : modèle compact, navigation par capteurs de profondeur.

SerbAI (AHOUZI Hasnae) s'inspire de ces modèles tout en adoptant une approche locale, éducative et économique :

— Utilisation d'un matériel open-source.
— Développement ROS2 compatible Raspberry Pi.
— Application mobile simple et intuitive en Java.

3 Solution proposée

Le projet combine quatre modules clés :

1. **Mécanique et conception 3D :** modélisation du châssis, supports moteurs, plateaux et boîtier via CATIA.
2. **Système embarqué et ROS2 :** gestion de la navigation, perception et communication entre nœuds.
3. **Navigation SLAM :** cartographie et localisation en temps réel avec RTAB-Map ou Hector SLAM.
4. **Application mobile :** contrôle manuel, visualisation de la carte, retour d'état (batterie, position, état du service).

Plan de travail sur 7 semaines

— **S1–S2 :** Étude du besoin + formation ROS2.
— **S3 :** Formation Navigation SLAM + conception 3D.
— **S4 :** Simulation Gazebo + test des nœuds ROS2.
— **S5 :** Assemblage matériel et intégration capteurs.
— **S6 :** Développement application mobile Android (Java).
— **S7 :** Tests, validation et démonstration finale.

4 Liste du matériel (capteurs et actionneurs)

Électronique principale

— **Raspberry Pi 4 (4 Go RAM)** — cœur du système ROS2, gère les nœuds, la communication et l'interface réseau.
— **L298N** — pont en H pour contrôler deux moteurs DC indépendants.
— **Deux moteurs DC avec encodeurs** — assurent un déplacement précis et mesurable pour le contrôle en boucle fermée.
— **LIDAR RPLidar A1** — fournit une cartographie 2D de l'environnement et permet l'évitement d'obstacles.
— **IMU MPU6050** — capteur inertiel 6 axes pour mesurer l'inclinaison, l'accélération et la vitesse angulaire.
— **Capteurs ultrason HC-SR04** — détection d'obstacles proches et mesure de distance.
— **Caméra USB 720p** — vision frontale (ajoutée dans la version avancée avec IA).
— **LEDs et Buzzer** — signaux visuels et sonores d'état et d'interaction.

Structure et alimentation

— **Châssis imprimé en 3D (PLA)** ou aluminium léger.
— **Roues motrices + roue libre avant** — stabilité mécanique.
— **Batterie Lithium 12V/7Ah** — alimentation principale du système embarqué.
— **Câblage et interrupteur principal** — sécurité et maintenance aisée.

Logiciels utilisés

— **Ubuntu 22.04 + ROS2 Humble** — environnement d'exécution principal.
— **Gazebo + RViz2** — simulation et visualisation du robot.
— **CATIA V5** — modélisation 3D du châssis et des composants.
— **Android Studio (Java)** — développement de l'application mobile de contrôle.

5 Justification du choix du matériel

Le choix des capteurs et actionneurs du projet **SerbAI** repose sur trois critères principaux :

— **Pertinence fonctionnelle** : adéquation du composant avec les besoins du scénario (navigation, détection, contrôle).
— **Fiabilité et compatibilité ROS2** : disponibilité de drivers et de packages ROS2 prêts à l'emploi.
— **Évolutivité et réutilisabilité** : possibilité d'étendre le système vers des versions plus complexes sans changer l'architecture de base.

5.1 Tableau comparatif des capteurs et actionneurs

Composant	Modèle choisi	Justification technique	Alternatives étudiées
LIDAR	RPLIDAR A1M8	Portée 12 m, 360°, compatible ROS2 (package `rplidar_ros2`) ; idéal pour la détection d'obstacles et la navigation autonome.	YDLIDAR X4 (moins stable, portée plus courte)
IMU	MPU-6050	Capteur 6 axes (accéléromètre + gyroscope) ; permet la détection de l'inclinaison et la stabilisation du robot.	BNO055 (plus précis mais plus coûteux)
Capteurs ultrasoniques	HC-SR04 (x3)	Mesure de distance entre 2–400 cm ; complément idéal du LIDAR pour éviter les obstacles rapprochés.	Capteurs IR Sharp (moins précis selon la lumière)
Caméra	Caméra USB 720p	Fournit une vision frontale pour la reconnaissance d'objets ou d'humains dans la version IA.	Raspberry Pi Camera V2
Microcontrôleur	Raspberry Pi 4 (4 Go)	Exécute ROS2, contrôle les capteurs, gère les nœuds et la communication réseau.	Jetson Nano (plus puissant mais non essentiel ici)
Moteurs DC	2 × Moteurs 12V 200 RPM + encodeurs	Offrent un contrôle précis de la vitesse et de la position pour la navigation différenciée.	NEMA17 (nécessite drivers supplémentaires)
Contrôleur de moteur	L298N Dual H-Bridge	Gère la direction et la vitesse des moteurs via signaux PWM.	L293D (puissance limitée)
Alimentation	Batterie Li-Ion 12V 3Ah	Fournit une bonne autonomie et stabilité électrique pour les tests prolongés.	Batterie plomb-gel (trop lourde pour un petit robot)

5.2 Analyse technique et pédagogique

1. Évolutivité du système L'architecture du robot est conçue pour être **progressive** :

— Phase 1 : navigation dans un espace connu sans SLAM.
— Phase 2 : ajout du SLAM pour cartographier de nouveaux environnements.
— Phase 3 : intégration IA pour la reconnaissance d'objets et planification intelligente.

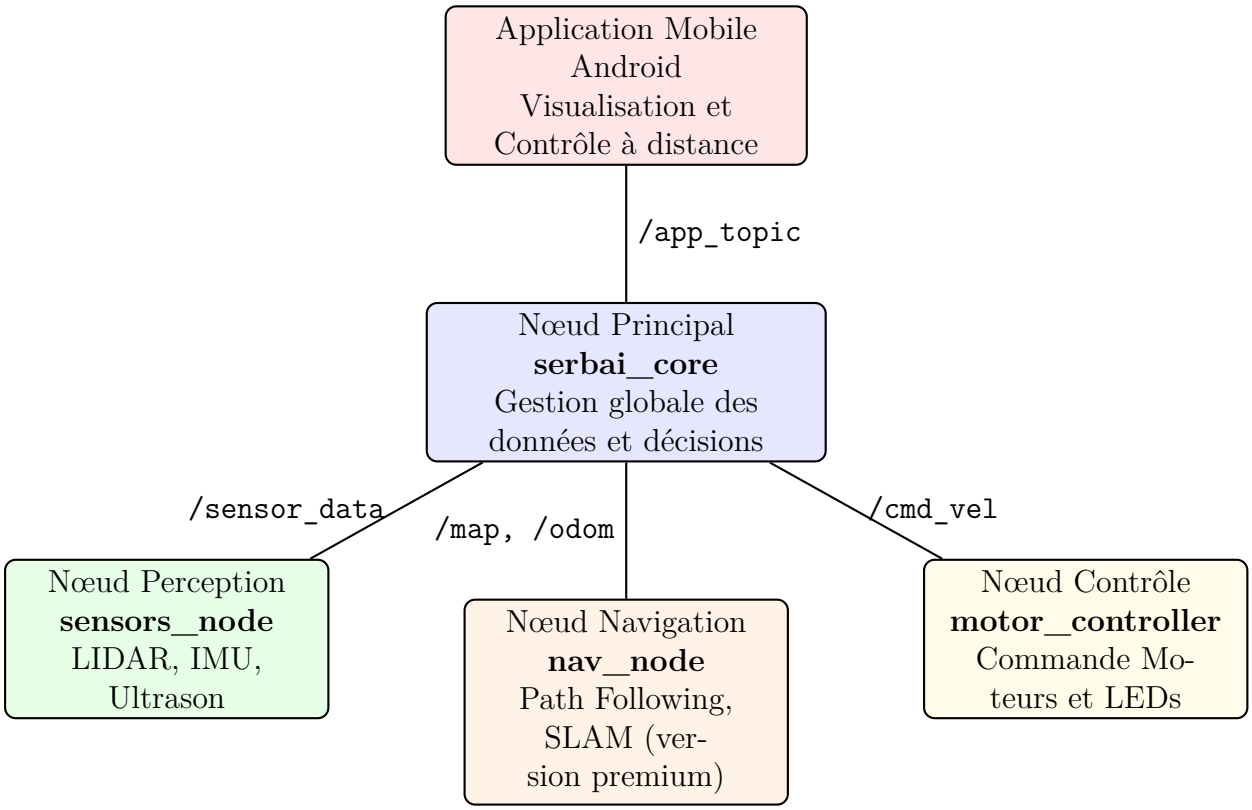
2. Compatibilité ROS2 et intégration pratique Tous les capteurs sont compatibles avec ROS2 via des packages officiels ou communautaires :

— `rplidar_ros2` — acquisition de scan LIDAR.
— `imu_filter_madgwick` — estimation d'orientation à partir de l'IMU.
— `ros2_serial` — communication entre la Raspberry Pi et les capteurs/moteurs.
— `rosbridge_suite` — passerelle entre ROS2 et l'application mobile Android.

6 Architecture ROS2 et Communication Inter-nœuds

6.1 Description du diagramme ROS2

Le système **SerbAI** est conçu selon une architecture modulaire ROS2 afin d'assurer une communication fluide entre les différents nœuds fonctionnels : perception, décision, navigation, et exécution. Le cœur du système repose sur l'échange de messages à travers plusieurs *topics*, complétés par des *services* pour des requêtes ponctuelles et des *actions* pour les tâches longues comme la navigation.



6.2 Détails des Topics, Services et Actions

1. Scénario de base (sans SLAM)

Ce scénario correspond à la première version du projet, où le robot évolue dans un environnement connu et cartographié à l'avance.

Topics

— `/sensor_data` : regroupe les mesures du LIDAR, de l'IMU et des capteurs ultrasoniques.
Rôle : détection d'obstacles, estimation de l'orientation et correction d'erreurs de trajectoire.

— `/cmd_vel` : topic standard de ROS2 pour transmettre les commandes de vitesse linéaire et angulaire aux moteurs.
Rôle : pilotage direct du déplacement.

— `/app_topic` : canal bidirectionnel entre l'application mobile et le système ROS2.
Rôle : envoi d'ordres (démarrage, arrêt, vitesse) et réception du statut du robot.

Services
— `/reset_odom` : remet à zéro l'odométrie.
— `/set_speed_limit` : ajuste la vitesse maximale selon le mode (manuel, auto, sécurité).

Actions
— `/move_to_point` : permet au robot de se déplacer vers une position connue (x, y, θ).

2. Scénario avancé (avec SLAM et navigation autonome)

Dans la version premium, le système intègre un module de cartographie et de localisation simultanée (SLAM). Le robot devient capable d'explorer un environnement inconnu, de construire une carte en temps réel et de planifier ses trajets de manière autonome.

Topics additionnels
— `/scan` : topic issu du LIDAR, contenant les distances mesurées à 360°.
— `/map` : carte générée et mise à jour en continu.
— `/odom` : odométrie calculée à partir des roues et du gyroscope, pour fusionner les données de position.
— `/path` : trajectoire optimale calculée par l'algorithme de planification.
— `/goal_pose` : destination définie par l'utilisateur ou générée automatiquement.

Services additionnels
— `/save_map` : enregistre la carte dans le système de fichiers.
— `/load_map` : recharge une carte existante pour redémarrage rapide.

Actions avancées
— `/navigate_to_pose` : commande de navigation complète avec retour de statut (succès, échec, bloqué).
— `/explore_environment` : mode d'exploration automatique pour reconnaître de nouveaux espaces.

6.3 Justification des choix techniques

— **Séparation en nœuds ROS2** : facilite le débogage, la maintenance et la réutilisation des composants.
— **Utilisation de topics standardisés** : compatibilité avec les librairies ROS2 existantes (ex : `geometry_msgs`, `nav_msgs`).
— **Architecture évolutive** : passage simple d'un mode manuel (téléop) à un mode autonome (SLAM + Navigation).
— **Communication Android–ROS2 via MQTT ou WebSocket** : réduit la latence et permet une interface temps réel.

6.4 Synthèse comparative

Élément	Version Basique	Version Premium (avec SLAM)
Cartographie	Pré-définie	Dynamique, en temps réel
Navigation	Suivi de trajectoire fixe	Planification automatique
Topics principaux	/sensor_data, /cmd_vel	+ /map, /odom, /scan
Actions	/move_to_point	+ /navigate_to_pose, /explore_environment
Complexité logicielle	Moyenne	Élevée

Conclusion

Le projet **SerbAI** représente une intégration complète de la robotique moderne, alliant ROS2, navigation autonome, conception 3D et développement mobile. Grâce à une planification sur 7 semaines, un budget réduit et une approche open-source, ce robot servira non seulement à démontrer la faisabilité d'un robot serveur low-cost, mais aussi à valoriser les compétences acquises à l'Université Euromed de Fès.