SAÉ6.IOM.01 — Sujet n°10
BUT Réseaux & Télécommunications — IOM

RAPPORT DE SAÉ

ROBOTANK

Robot de Surveillance et de Mesure

Architecture : Raspberry Pi 4 + XIAO ESP32-C6 (Zigbee)
Caméra Fisheye IR · OLED I2C · HC-SR04 · Steppers 28BYJ-48
Alimentation Powerbank USB 5 V · Interface web Flask · Cartographie WiFi

Auteur	Léo-Paul Hénon
Formation	BUT R&T — Parcours IOM
Établissement	IUT de Blois — Université de Tours
Commanditaire	M. Jeanneret
Année	2025–2026

Ce document a été réalisé dans le cadre de la SAÉ6.IOM.01
IUT de Blois — Université de Tours — 2025/2026

Suivi du Document

Ver.	Date	Objet	Auteur
v1.0	04/02/2026	Analyse fonctionnelle initiale	L. Hénon
v2.0	05/02/2026	Architecture hybride RPi4 + kit Arduino	L. Hénon
v3.0	15/03/2026	Conception CAO + câblage + interface web	L. Hénon
v4.0	28/03/2026	Ajout notice d'utilisation, historique manette, perspectives	L. Hénon
vFinal	Mars 2026	Rapport de SAÉ final complet	L. Hénon

Table des Matières

Avant-Propos

◎ Information

Ce rapport a été rédigé dans le cadre de la SAÉ6.IOM.01 (Situation d'Apprentissage et d'Évaluation) du BUT Réseaux & Télécommunications, parcours Internet des Objets et Mobilité (IOM), à l'IUT de Blois — Université de Tours.
La SAÉ consiste à concevoir, réaliser et valider un robot mobile connecté répondant à un cahier des charges fonctionnel imposé. Le sujet n°10 porte sur un **robot de surveillance et de mesure de la qualité WiFi**, baptisé **RoboTank**.

Ce document constitue le rapport final de la SAÉ. Il retrace l'ensemble des étapes du projet : analyse fonctionnelle, choix techniques, conception mécanique par impression 3D, câblage électronique, développement logiciel, interface web Flask, tests de validation, **notice d'utilisation complète**, plan de maintenance et perspectives d'évolution. L'évolution de la télécommande (de l'Arduino Esplora/Bluetooth vers le Zigbee) y est également documentée.

Le projet mobilise des compétences transversales en électronique embarquée, réseaux WiFi, programmation Python/Flask, modélisation CAO (Shapr3D), et gestion de projet.

Version Finale — Mars 2026

1 Présentation du Projet

1.1 Contexte de la SAÉ

La SAÉ6.IOM.01 représente le projet intégrateur du troisième semestre du BUT R&T parcours IOM. Elle vise à mettre en pratique l'ensemble des compétences acquises au cours de la formation : systèmes embarqués, réseaux sans fil, programmation, électronique et gestion de projet.

Le sujet n°10, intitulé *« Robot de surveillance et de mesure »*, impose la réalisation d'un système autonome capable de naviguer dans un environnement intérieur, de détecter des obstacles, de capturer des images, et de produire une cartographie de la qualité du signal WiFi.

1.2 Objectifs

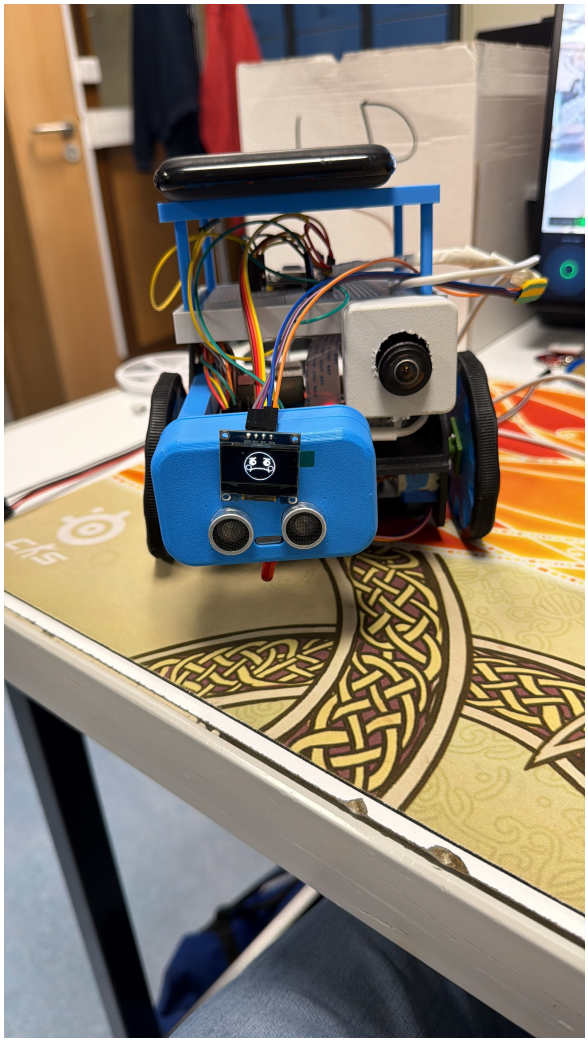
⋆ Fonctionnalités attendues

- **Navigation** : mode autonome (exploration) et mode télécommandé (Zigbee)
- **Détection d'obstacles** : capteur ultrasonique HC-SR04, arrêt automatique
- **Vision** : caméra PiCamera IR 1080p, capture horodatée sur événement
- **Mesure WiFi** : RSSI mesuré en continu, stocké en CSV/JSON
- **Cartographie** : heatmap exportable (PNG) par dead reckoning
- **Feedback utilisateur** : écran OLED I2C + LEDs d'état (vert / rouge / orange)
- **Interface web** : tableau de bord Flask accessible en WiFi

1.3 Contraintes imposées

- Usage **indoor** (sol plat, intérieur)
- Tension système : **max 12 V**
- Autonomie minimale : **20 minutes**
- Matériel prioritairement issu du **stock IUT**
- Solution **présentable**, proche d'un produit commercialisable

1.4 Vue d'ensemble du robot final



(a) RoboTank assemblé — vue de face

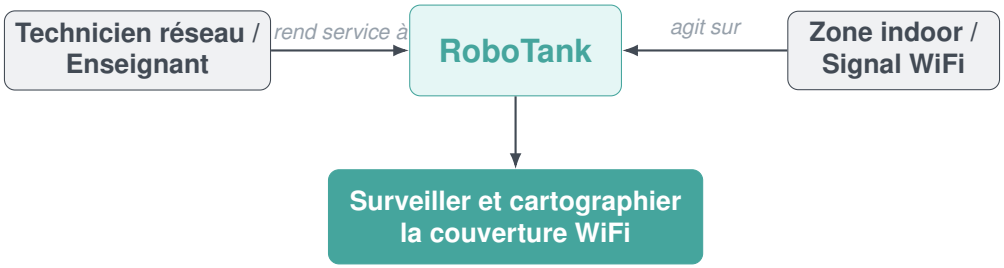
(b) RoboTank — Raspberry Pi 4 intégré

Figure 1.1: RoboTank — Robot de Surveillance et de Mesure WiFi

2 Analyse Fonctionnelle

2.1 Bête à cornes

L'analyse fonctionnelle permet d'identifier les besoins fondamentaux du système avant toute solution technique.



2.2 Fonctions Principales de Service

ID	Fonction principale	Priorité
FP1	Se déplacer en mode autonome ou télécommandé (Zigbee)	MUST
FP2	Détecter et éviter les obstacles (HC-SR04)	MUST
FP3	Mesurer le RSSI WiFi à intervalle ≤ 2 s	SHOULD
FP4	Associer une position estimée aux mesures (dead reckoning)	SHOULD
FP5	Enregistrer les données localement (CSV/JSON)	MUST
FP6	Générer une heatmap exportable (PNG)	SHOULD
FP7	Capturer une photo horodatée lors d'un événement	SHOULD
FP8	Afficher l'état du robot (OLED + LEDs)	SHOULD

2.3 Exigences et Critères d'Acceptation

ID	Exigence	Critère	Priorité
EX-01	Autonomie $\geq$ 20 min	Chrono $\geq$ 20 min	MUST
EX-02	Pilotage manuel via manette Zigbee	Téléopération 5 min OK	MUST
EX-03	Évitement obstacle à 20 cm	0 collision / 10 essais	MUST
EX-04	RSSI mesuré toutes les 2 s	API REST `/api/scan/status` $\leq$ 2 s	SHOULD
EX-05	Stockage offline si perte WiFi	Logs présents hors ligne	MUST
EX-06	Heatmap rendue en HTML5 canvas	API REST `/api/map` + visualisation web	SHOULD
EX-07	Photo horodatée sur événement	Image avec timestamp	SHOULD
EX-08	Arrêt d'urgence logiciel	STOP immédiat	MUST
EX-09	Alimentation 5 V stable RPi via powerbank USB (4,9–5,2 V)	Mesure stable	MUST
EX-10	OLED : mode + RSSI + état	Info visible	SHOULD
EX-11	LED rouge = obstacle détecté ($<$20 cm)	Allumée si obstacle	MUST

Tableau 2.1: Exigences testables et critères d'acceptation

3 Architecture Matérielle

3.1 Choix Architecturaux

✓ Justification : Architecture Hybride RPi4 + Châssis 3D

Le Raspberry Pi 4 a été retenu comme unité centrale du fait de sa puissance de calcul, de son support WiFi natif, de sa compatibilité PiCamera et Python. Il est couplé à un châssis entièrement imprimé en 3D (PLA bleu), motorisé par deux steppers 28BYJ-48 pilotés via des drivers ULN2003. L'alimentation est fournie par une **batterie externe USB (powerbank)** délivrant 5 V, branchée directement sur le port USB-C du Raspberry Pi — une solution simple, sûre et modulable. La communication sans fil est assurée par deux modules XIAO ESP32-C6 en Zigbee : l'un intégré dans la manette physique (réutilisant le boîtier de la manette Arduino Esplora), l'autre dans le robot (coordinateur).

3.2 Schéma Synoptique

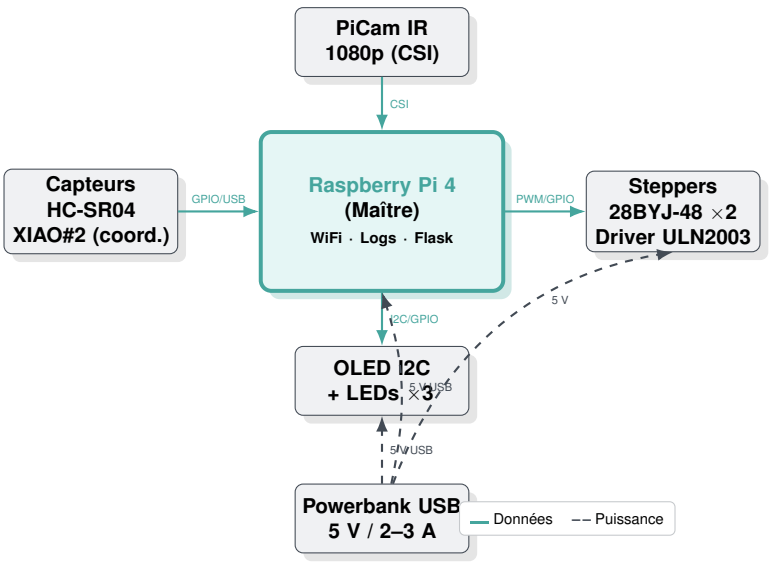


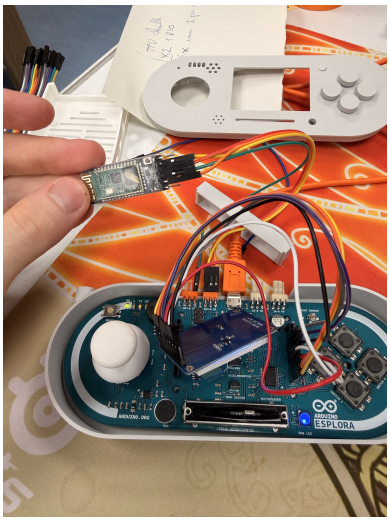
Figure 3.1: Architecture matérielle de RoboTank

3.3 Bill of Materials (BOM)

Composant	Qté	Rôle
Raspberry Pi 4 Model B — 4 Go	1	Unité centrale, WiFi, traitements, Flask
Carte microSD 32 Go	1	OS + stockage logs CSV/JSON
PiCamera IR 1080p	1	Photo / vision nocturne infrarouge
Capteur ultrason HC-SR04	1	Mesure distance, détection obstacle
Écran OLED I2C 0,96" (SSD1306)	1	Affichage mode, RSSI, état système
XIAO ESP32-C6 (Seeed Studio)	2	Zigbee : coordinateur (robot) + End Device (manette)
ESP32 DevKit (capteur ultrason)	1	Lecture HC-SR04 + pilotage OLED I2C + LEDs
Arduino Esplora	1	Base manette — boîtier réutilisé comme coque ergonomique
Moteurs stepper 28BYJ-48 + ULN2003	2	Traction du châssis imprimé 3D
LEDs 5 mm (vert, rouge, orange)	3	Feedback visuel état robot
Résistances 220 Ω	3	Limitation courant LEDs
Adaptation niveau ECHO (div.)	2	5 V → 3,3 V (HC-SR04 vers RPi)
Batterie externe (Powerbank USB)	1	Source d'énergie principale — 5 V / 2–3 A
Câble USB-A vers USB-C (ou micro-USB)	1	Alimentation RPi depuis la powerbank
Condensateurs 1000 µF + 100 nF	2+	Découplage, anti-reboot moteur
Châssis + pièces imprimées 3D	1	Base mécanique (PLA bleu)
Câbles Dupont, JST, borniers	lot	Connexions prototype

Tableau 3.1: Bill of Materials — composants principaux

3.3.1 Module XIAO ESP32-C6



(a) Module ESPLORA avec module bluetooth

(b) Manette Zigbee — joystick et boutons directionnels

Figure 3.2: Interface Zigbee — XIAO ESP32-C6 et manette physique

Le module XIAO ESP32-C6 est un SoC ultra-compact intégrant le Zigbee 802.15.4 et le WiFi 6. Deux modules sont déployés :

- **XIAO#1** — dans la manette : End Device Zigbee, transmet les commandes F/B/L/R/S
- **XIAO#2** — dans le robot : Coordinateur Zigbee, relaye les commandes au RPi via `/dev/ttyACM0`

3.3.2 Évolution de la Télécommande : de l'Esplora/Bluetooth au Zigbee

⋆ Historique de la manette de contrôle

Dans un premier temps, le pilotage du robot était assuré par une **manette Arduino Esplora** équipée d'un **module Bluetooth HC-05**. L'Esplora intègre nativement un joystick analogique, des boutons, un accéléromètre et un buzzer, ce qui en faisait un choix initial pratique pour le prototypage rapide.
Cependant, plusieurs limitations du Bluetooth HC-05 ont motivé le changement de technologie :

- **Portée limitée** : environ 10 m en intérieur, insuffisant pour certains scénarios de surveillance
- **Appairage contraignant** : procédure de jumelage à chaque reconnexion, peu fiable
- **Latence variable** : le protocole SPP (Serial Port Profile) du Bluetooth classique introduisait des délais perceptibles dans la télécommande
- **Interférences WiFi** : le Bluetooth (2,4 GHz) partage la même bande que le WiFi, créant des perturbations lors des mesures RSSI

La solution retenue a été de passer à la communication **Zigbee 802.15.4** via les modules XIAO ESP32-C6. Un **joystick analogique externe** et des boutons poussoirs remplacent les entrées intégrées de l'Esplora. L'ensemble est monté dans le **boîtier de la manette Esplora** (réutilisé comme coque ergonomique), offrant ainsi une prise en main confortable et un aspect fini.

Critère	v1 — Esplora + HC-05	v2 — XIAO Zigbee (actuel)
Protocole	Bluetooth SPP (2,4 GHz)	Zigbee 802.15.4 (2,4 GHz)
Portée intérieur	$\sim$10 m	$\sim$30 m
Latence	Variable (50–150 ms)	Faible ($<$30 ms)
Interférence WiFi	Forte (même bande, même canal)	Faible (canaux Zigbee séparés)
Consommation	Moyenne	Très faible
Appairage	Manuel à chaque démarrage	Automatique (réseau Zigbee)
Boîtier	Manette Esplora intégrée	Boîtier Esplora réutilisé

4 Conception Mécanique

4.1 Démarche : Impression 3D

La totalité du châssis a été modélisée sur **Shapr3D** (CAO 3D sur iPad/Mac) et imprimée en PLA sur une imprimante FDM. Cette approche garantit une grande liberté de conception, des cycles d'itération rapides et un coût minimal.

◎ **Information**

Logiciel utilisé : Shapr3D (version éducation)
Matériau : PLA (acide polylactique) — filament 1,75 mm
Couleur : Blanc (boîtier manette) / Bleu (châssis, roues)

4.2 Pièces Conçues et Imprimées

Chaque pièce a été modélisée individuellement dans Shapr3D, puis assemblée dans un modèle complet pour vérifier les ajustements et les jeux mécaniques avant impression. Les sous-sections suivantes détaillent le rôle, la conception et les choix de chaque pièce.

4.2.1 Châssis principal (inférieur et supérieur)

Le châssis est composé de deux étages superposés : un **châssis inférieur** qui accueille les deux moteurs stepper 28BYJ-48 avec leurs drivers ULN2003, la bille folle arrière et le passage des câbles ; et un **châssis supérieur** sur lequel sont fixés le Raspberry Pi 4 (4 vis M2.5) et la batterie externe. Les deux étages sont reliés par des supports arrière (décrits ci-dessous) pour former un ensemble rigide.

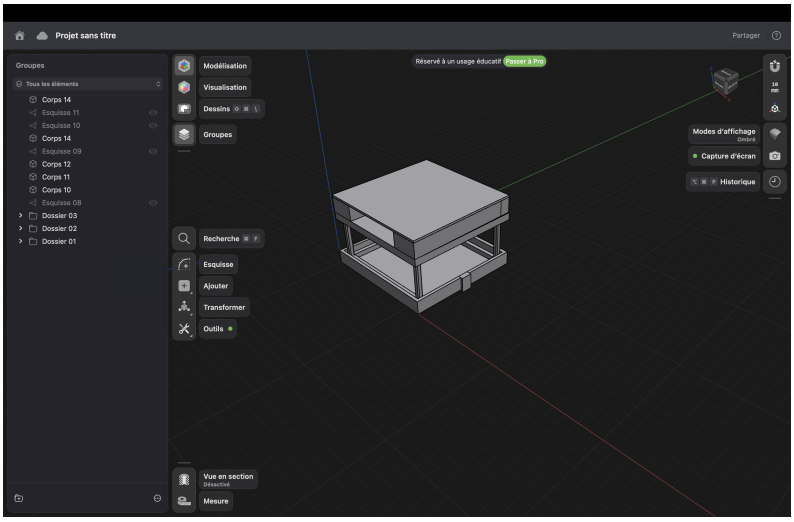


Figure 4.1: Modèle CAO du châssis complet — vue isométrique (Shapr3D). On distingue les emplacements des moteurs, les trous de fixation et les passages de câbles.

4.2.2 Fixation de l'écran OLED

Cette pièce est positionnée sur le **dessus du boîtier du capteur ultrason** (bumper). Elle permet à l'écran OLED SSD1306, sur lequel sont affichées les informations d'état du robot (mode, RSSI, distance obstacle, adresse IP), d'être solidement attaché et de faire corps avec l'ensemble de la structure avant. La pièce intègre des clips de maintien qui permettent d'insérer l'OLED sans vis, facilitant ainsi le démontage pour maintenance.

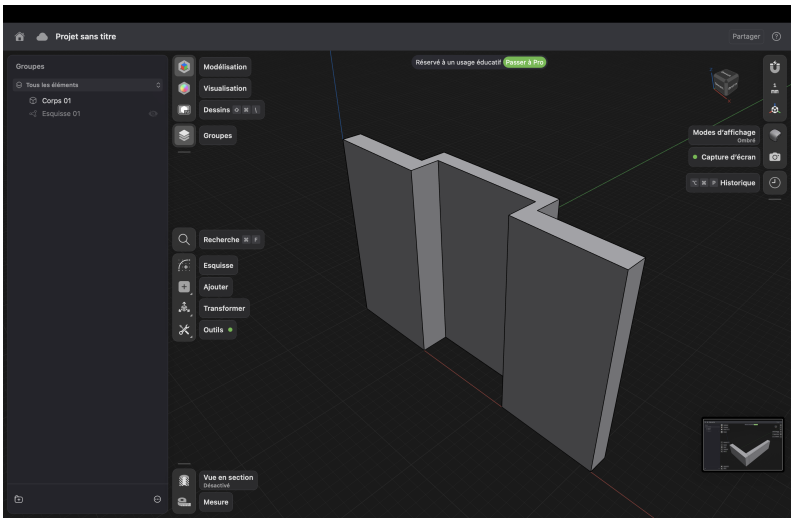
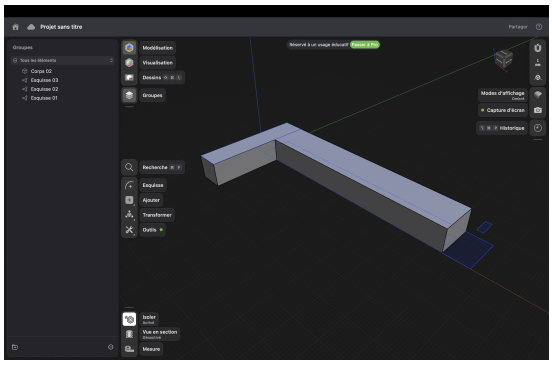


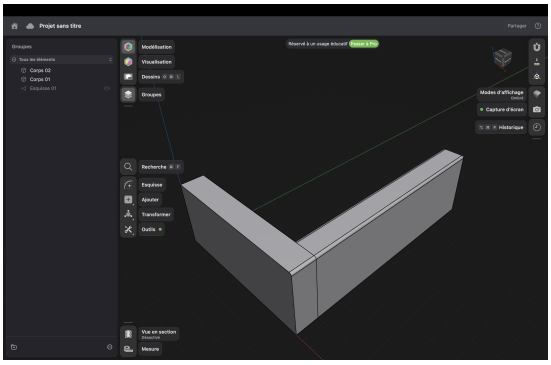
Figure 4.2: Modèle CAO de la fixation OLED (Shapr3D) — pièce se positionnant sur le dessus du boîtier capteur ultrason pour accueillir l'écran OLED SSD1306.

4.2.3 Supports châssis et bras de fixation du boîtier capteur

Ces pièces assurent la **liaison mécanique** entre les différents éléments du robot. La première pièce est un support permettant de **fixer le châssis supérieur au châssis inférieur**, reliant solidement les deux étages à l'arrière du robot. La seconde pièce est le **bras de fixation du boîtier capteur ultrason** au châssis : elle permet de maintenir le bumper (contenant le HC-SR04 et l'OLED) solidement arrimé à la structure principale.



(a) Support de liaison haut/bas du châssis



(b) Bras de fixation du boîtier capteur ultrason

Figure 4.3: Supports de liaison châssis et bras de fixation du boîtier capteur — assurent la rigidité de l'ensemble mécanique.

4.2.4 Roues — Évolution du concept (chenilles → roues classiques)

Dans un premier temps, l'idée retenue était de transformer les roues existantes du robot en un **système de chenilles**. Pour cela, une roue dentée à 10 dents a été modélisée dans Shapr3D, destinée à entraîner une courroie souple faisant office de chenille.

Cependant, après **plusieurs essais physiques** réalisés avec les pièces imprimées en 3D, cette approche s'est révélée peu concluante : la courroie glissait sur les dents, l'alignement était difficile à maintenir, et la motricité restait insuffisante pour le poids du robot.

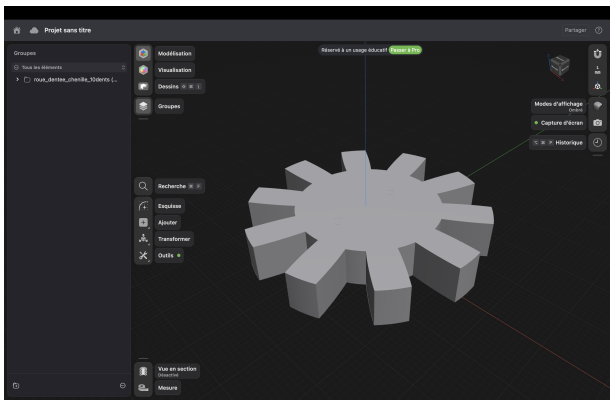


Figure 4.4: Modèle CAO de la roue dentée (10 dents) — prototype de chenille **abandonné** au profit de roues classiques à grand rayon.

Conclusion : il a été décidé d'abandonner le système de chenilles au profit de **roues classiques redessinées avec un rayon plus grand**, offrant une meilleure adhérence et une vitesse de déplacement supérieure, tout en simplifiant considérablement l'assemblage mécanique. Les nouvelles roues s'adaptent directement sur l'axe à méplat des moteurs 28BYJ-48, sans courroie ni réduction intermédiaire.

Figure 4.5: Pièces imprimées 3D de la manette (PLA blanc) — vue éclatée sur le plateau d'impression. On distingue les deux coques du boîtier Esplora, les boutons et les entretoises.

4.2.5 Entretoises pour la manette

Les entretoises sont des petites pièces cylindriques de quatre dimensions différentes. Une fois collées dans le châssis de la manette Esplora, elles servent à **maintenir le joystick en place** de manière stable et alignée. Elles compensent l'épaisseur entre la surface interne du boîtier Esplora et le module joystick, assurant un jeu minimal et un retour au centre fiable.

4.2.6 Entretoises de maintien du joystick

Les quatre entretoises (piliers cylindriques de dimensions différentes) sont des pièces essentielles au bon fonctionnement de la manette. Une fois collées à l'intérieur du boîtier Esplora, elles servent à **maintenir le joystick KY-023 en place** de manière stable et alignée. Elles compensent l'épaisseur entre la surface interne du boîtier et le module joystick, assurant un jeu minimal et un retour au centre fiable.

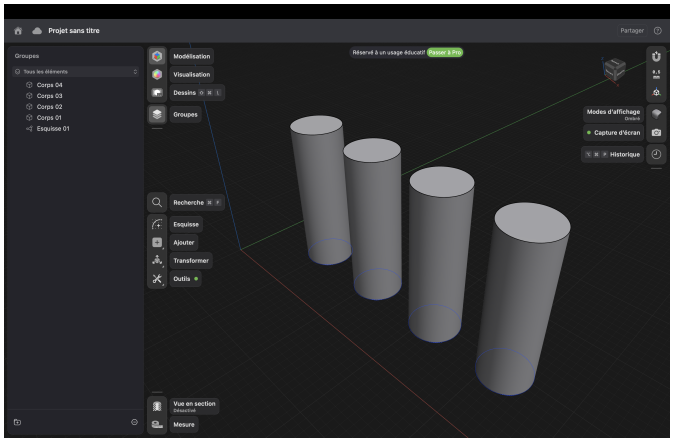


Figure 4.6: Modèle CAO des quatre entretoises (Shapr3D) — pièces cylindriques de quatre tailles différentes, collées dans le boîtier de la manette pour maintenir le joystick KY-023.

4.3 Paramètres d'Impression 3D

Paramètre	Valeur	Justification
Matériau	PLA blanc/bleu	Charte graphique RoboTank, facilité d'impression
Hauteur de couche	0,2 mm	Compromis résolution / temps d'impression
Remplissage	20% (gyroïde)	Rigidité suffisante pour un poids réduit
Périmètres	3 lignes	Solidité structurelle des parois
Température buse	210 °C	Standard PLA (bonne fluidité)
Température plateau	60 °C	Adhésion optimale
Vitesse	50 mm/s	Qualité de surface acceptable
Slicer utilisé	Bambu Studio 1.10.x	Profils PLA optimisés

4.4 Temps d'Impression et Consommation Matière

Pièce	Qté	Temps	Matière (g)
Châssis inférieur	1	4h30	85
Châssis supérieur	1	3h45	72
Bumper (boîtier capteur ultrason)	1	2h15	45
Roues classiques (grand rayon)	4	50 min	15
Entretoises joystick manette	4	25 min	6
Supports châssis (liaison haut/bas)	2	1h00	18
Bras de fixation capteur ultrason	1	35 min	10
Fixation écran OLED	1	30 min	8
TOTAL	**—**	**∼17h**	**∼370 g**

Information

Le temps total d'impression est d'environ **17 heures**, pour une consommation de **370 g de PLA**. À raison de ∼20 €/kg de filament PLA, le coût matière est d'environ **7,40 €**, ce qui reste très économique comparé à l'achat d'un châssis commercial.

4.5 Procédure d'Assemblage Mécanique

L'assemblage du robot suit un ordre précis pour faciliter l'accès aux composants et le passage des câbles :

1. Fixation des moteurs stepper 28BYJ-48 et de leurs drivers ULN2003 sur le châssis inférieur.
2. Montage des roues imprimées 3D (grand rayon) sur les axes des moteurs.
3. Fixation de la bille folle (roue arrière) sous le châssis inférieur.
4. Câblage des ULN2003 aux moteurs et préparation des fils vers le GPIO du Raspberry Pi.
5. Montage des supports châssis (liaison haut/bas) à l'arrière du châssis inférieur.
6. Fixation du châssis supérieur sur les supports arrière et les colonnes de maintien.
7. Installation du Raspberry Pi 4 sur le châssis supérieur (4 vis M2.5).
8. Installation de la batterie externe sur le dessus du châssis supérieur et raccordement au Raspberry Pi (alimentation USB-C).
9. Branchement de la nappe caméra CSI entre le Raspberry Pi et la PiCam IR.
10. Montage du bras de fixation du capteur ultrason sur l'avant du châssis.
11. Installation du capteur HC-SR04 dans le bumper (boîtier avant).
12. Fixation de la pièce de maintien de l'écran OLED sur le dessus du bumper.
13. Installation de l'écran OLED et de la LED rouge dans leurs logements respectifs.
14. Raccordement de l'OLED (I2C : SDA/SCL), des LEDs (GPIO + résistances 220 Ω).
15. Branchement du XIAO ESP32-C6 Coordinator en USB sur le Raspberry Pi.
16. Branchement de l'ESP32 (lecteur HC-SR04) en USB sur le Raspberry Pi.
17. Vérification finale de l'ensemble des connexions et premier démarrage.

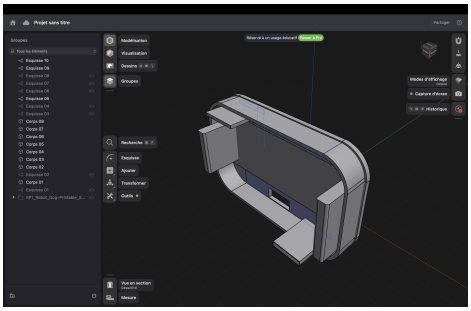
Temps d'assemblage estimé : 2 à 3 heures (avec vérification électrique progressive).

! Vérification électrique avant mise sous tension

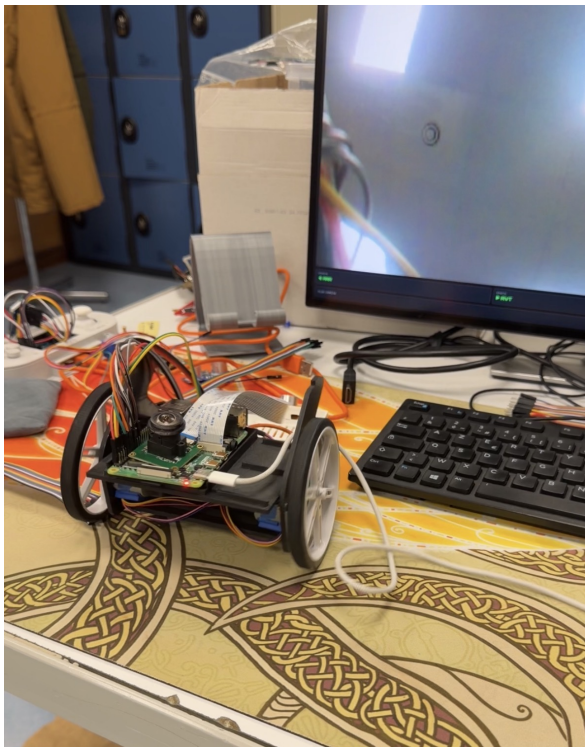
Avant chaque connexion au Raspberry Pi, vérifier :

- Absence de court-circuit entre 5 V et GND (multimètre en mode continuité)
- Batterie externe suffisamment chargée (LED indicateur allumée)
- Tension de sortie USB de la batterie externe : 5,0 V $\pm$ 0,2 V
- Bon contact des connecteurs Dupont, JST et câbles USB
- Nappe caméra CSI correctement insérée (languette de verrouillage fermée)

4.6 Réalisation Physique — Étapes de Montage

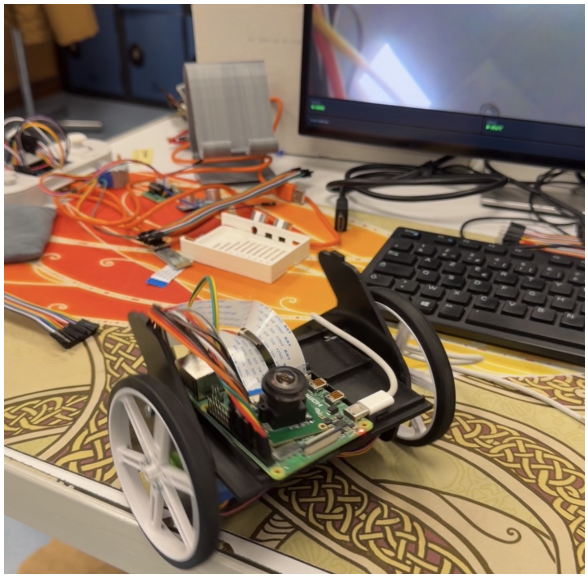


(a) Bumper imprimé 3D (vue de côté) — logements HC-SR04 et OLED visibles

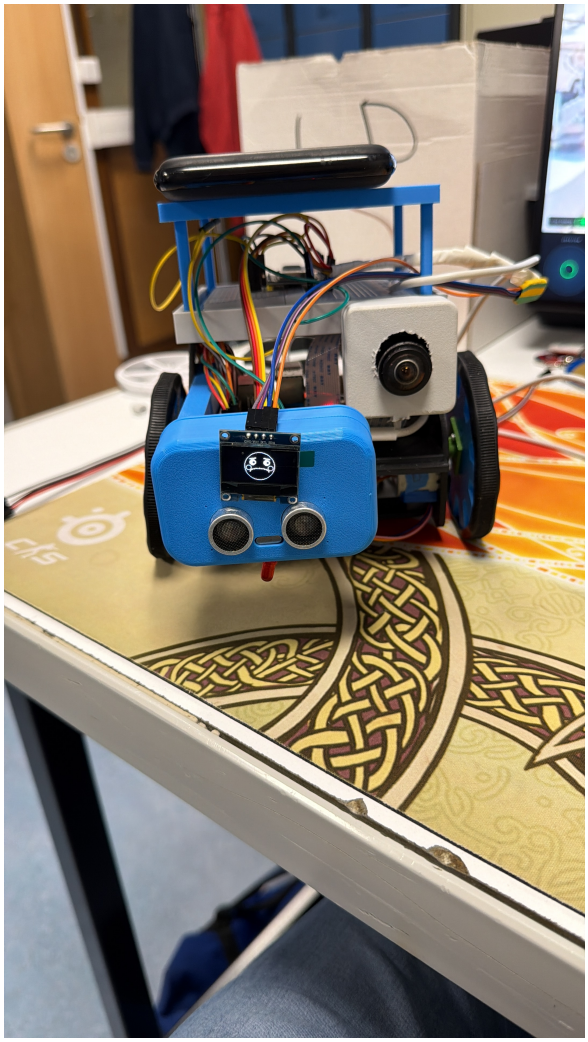


(b) Phase initiale — montage des moteurs et drivers sur le châssis inférieur

Figure 4.7: Pièces imprimées et montage intermédiaire



(a) Montage avancé — châssis supérieur posé, Raspberry Pi installé, câblage en cours

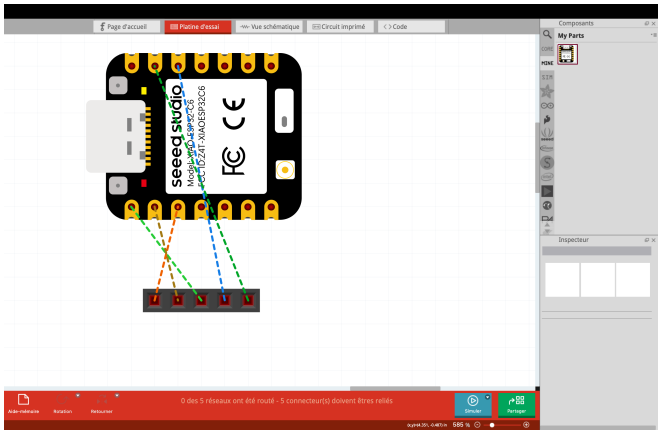


(b) Robot finalisé — vue complète avec bumper, caméra IR, OLED, LEDs et power-bank

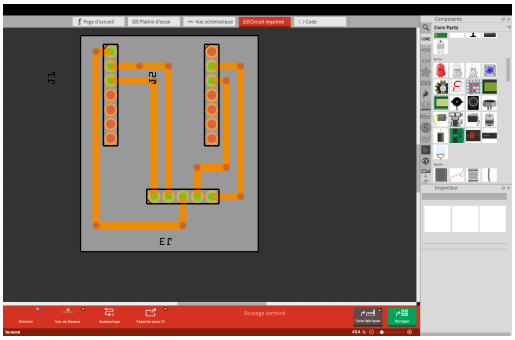
Figure 4.8: Étapes d'assemblage du RoboTank — du montage intermédiaire à la version finale

5 Électronique et Câblage

5.1 Schémas de Câblage



(a) Schéma Fritzing — platine d'essai



(b) Circuit imprimé (PCB Fritzing)

Figure 5.1: Schéma électronique et PCB — interface XIAO ESP32-C6

5.2 Brochage GPIO des Moteurs

Le pilotage des deux moteurs stepper 28BYJ-48 est réalisé via les drivers ULN2003, connectés directement sur les GPIO du Raspberry Pi 4. Chaque moteur utilise 4 broches (séquence de demi-pas) :

Moteur	IN1	IN2	IN3	IN4
Gauche	GPIO 17	GPIO 18	GPIO 27	GPIO 22
Droite	GPIO 23	GPIO 24	GPIO 25	GPIO 8

Information

Les 8 GPIO sont configurés en sortie (OUTPUT) au démarrage de `server.py`. La séquence de demi-pas (half-step) offre 512 demi-pas par tour, soit une résolution angulaire de 0,703 °/pas. Le sens de parcours de la séquence détermine le sens de rotation.

5.3 Dimensionnement Énergétique

5.3.1 Bilan de puissance

Élément	Puissance	Remarques
Raspberry Pi 4 Model B	3,5 W	Charge moyenne
PiCamera IR 1080p	1,0 W	Capture ponctuelle
OLED I2C	0,3 W	Affichage continu
LEDs ($\times$3)	0,2 W	Selon état
Steppers 28BYJ-48 ($\times$2) + ULN2003	3,0 W	Hors pointe
Capteur HC-SR04	0,2 W	Négligeable
Total estimé	**8,2 W**	Estimation moyenne

5.3.2 Choix de la batterie externe (Powerbank USB)

L'alimentation du RoboTank est assurée par une **batterie externe USB** (powerbank) délivrant une tension stable de 5 V. Ce choix présente plusieurs avantages par rapport à une batterie LiPo dédiée :

- **Sécurité** : circuit de protection BMS intégré, pas de risque de surcharge/sur-décharge
- **Simplicité** : connexion directe en USB vers le Raspberry Pi, sans convertisseur DC-DC
- **Disponibilité** : composant grand public, facilement remplaçable et rechargeable
- **Coût** : nettement inférieur à une solution LiPo + chargeur dédié + BMS externe

5.3.3 Calcul d'autonomie

Pour une powerbank de 10 000 mAh (capacité typique) à 5 V :

$$I_{\text{moy}} = \frac{P}{U} = \frac{8{,}2\ \text{W}}{5{,}0\ \text{V}} \approx 1{,}64\ \text{A}$$

$$T_{\text{autonomie}} = \frac{C_{\text{batterie}} \times \eta}{I_{\text{moy}}} = \frac{10{,}0\ \text{Ah} \times 0{,}85}{1{,}64\ \text{A}} \approx 5{,}18\ \text{h} \approx \mathbf{310}\ \text{min}$$

Information

Le rendement $\eta = 0{,}85$ tient compte des pertes du convertisseur interne de la powerbank. En pratique, l'autonomie varie selon la capacité réelle de la powerbank utilisée et la charge effective du robot (déplacements fréquents ou position stationnaire).

✓ **Résultat : Autonomie largement validée**

Même avec une powerbank modeste de 5 000 mAh, l'autonomie théorique dépasse **2 heures 30**, soit bien au-delà des 20 minutes requises (EX-01). La contrainte est validée avec une très large marge. L'utilisation d'une powerbank permet de moduler facilement l'autonomie en changeant simplement la batterie externe.

6 Architecture Logicielle

6.1 Organisation des Modules

L'architecture logicielle du RoboTank est organisée autour d'un serveur **Flask** central (`server.py`) coordonnant plusieurs threads Python dédiés.

⋆ Modules Python — server.py (∼1086 lignes)

- `xiao_reader()` — lecture série `/dev/ttyACM0` à 115200 baud, décodage des commandes Zigbee F/B/L/R/S provenant de la manette
- `serial_reader()` — lecture série `/dev/ttyUSB0` à 115200 baud, récupération de la distance HC-SR04 depuis l'ESP32 dédié
- `explore_loop()` — dead reckoning (x, y, θ), marquage de la grille 60×60 cellules, mesure RSSI WiFi à chaque déplacement
- `_cam_loop()` — capture Picamera2, génération du flux MJPEG à ∼25 fps
- `avoidance_loop()` — surveillance continue HC-SR04 à 10 Hz, déclenchement rotation 90°/180° si obstacle <20 cm, LED rouge, capture photo automatique
- `_drive_loop()` — pilotage bas niveau des moteurs stepper en demi-pas, fréquence ∼2000 Hz
- `auto_loop()` — mode autonome complet avec exploration libre et retour au point de départ par dead reckoning
- `rssi_scan_loop()` — scan systématique en serpentin 5×5 points sur 1 m², 5 mesures RSSI moyennées par point
- `index.html` — interface web unique (∼1235 lignes), HTML5/CSS3/JS vanilla

6.2 API REST

L'ensemble du contrôle du robot est exposé via une API REST Flask, accessible en HTTP depuis l'interface web ou tout client compatible :

Route	Méthode	Description
`/api/drive`	POST	Contrôle direction — reçoit (x, y) normalisés, conversion en différentiel gauche/droite
`/api/stop`	POST	Arrêt immédiat de tous les moteurs
`/api/speed`	POST	Réglage de la vitesse (1 à 10)
`/api/camera`	POST	Activation / désactivation du flux caméra

Route	Méthode	Description
`/api/auto`	POST	Mode autonome : start / stop / return (retour au départ)
`/api/scan`	POST	Lancement du scan RSSI systématique (serpentin 5×5)
`/api/scan/status`	GET	Progression et résultats du scan en cours
`/api/distance`	GET	Dernière distance mesurée par le HC-SR04 (cm)
`/api/status`	GET	État complet du système (mode, RSSI, distance, position)
`/api/map`	GET	Grille JSON 60×60 + position (x, y, θ) du robot
`/api/last_photo`	GET	Dernière photo obstacle encodée en base64
`/video_feed`	GET	Flux MJPEG continu depuis la PiCamera IR

Tableau 6.1: API REST du serveur Flask — ensemble des routes disponibles

6.3 Gestion des Threads

Le serveur Flask fonctionne en mode multi-threadé. Chaque module critique s'exécute dans son propre thread daemon, coordonné par des verrous (`threading.Lock`) pour éviter les accès concurrents :

Thread	Fonction	Fréquence	Description
xiao_reader	`xiao_reader()`	115200 baud	Commandes manette Zigbee
serial_reader	`serial_reader()`	115200 baud	Distance HC-SR04 via ESP32
avoidance	`avoidance_loop()`	10 Hz	Surveillance + évitement
cam_loop	`_cam_loop()`	∼25 fps	Capture MJPEG
drive_loop	`_drive_loop()`	∼2000 Hz	Pilotage moteurs (demi-pas)
auto_loop	`auto_loop()`	Variable	Mode autonome
scan_loop	`rssi_scan_loop()`	Variable	Scan RSSI systématique

6.4 Flux de Données

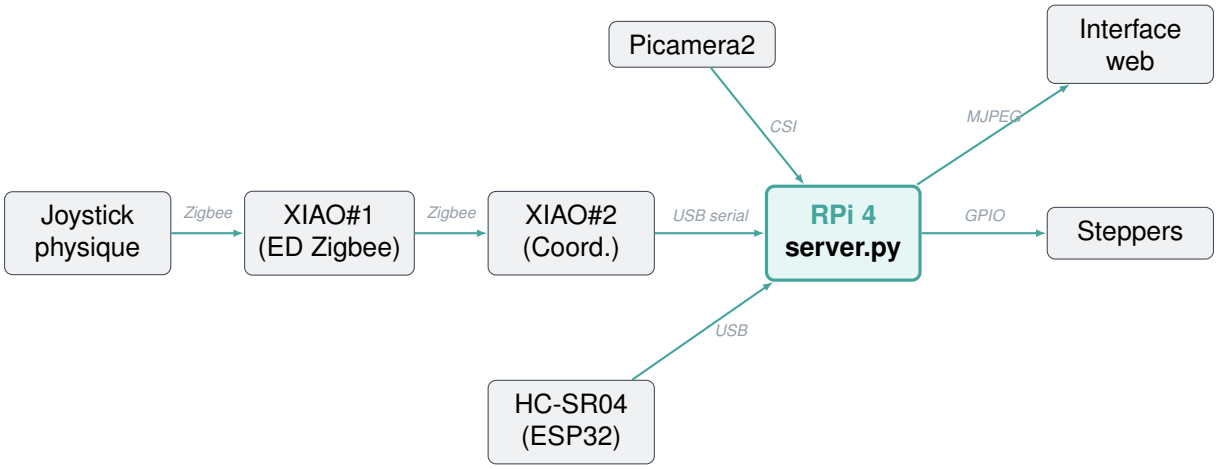


Figure 6.1: Flux de données — architecture logicielle RoboTank

Le flux de données principal suit la chaîne suivante :

- **Manette** → Joystick → XIAO#1 (Zigbee ED) → XIAO#2 (Coordinator) → `/dev/ttyACM0` → `drive()`
- **Capteur** → HC-SR04 → ESP32 → `/dev/ttyUSB0` → `avoidance_loop()` → rotation automatique
- **Navigation** → `explore_loop()` → dead reckoning (x, y, θ) → grille JSON → canvas web
- **Vidéo** → Picamera2 → MJPEG → `/video_feed` → interface web

6.5 Dead Reckoning et Cartographie

La position du robot est estimée par **odométrie** à partir du comptage des demi-pas moteur. Les paramètres physiques utilisés :

- Nombre de demi-pas par tour : **512**
- Circonférence de la roue : $\pi \times 86 \approx 270{,}2$ mm
- Distance par demi-pas : $270{,}2/512 \approx 0{,}528$ mm
- Empattement (distance entre les deux roues) : **300 mm**
- Facteur de correction rotation : **5,0** (calibré expérimentalement)

Les formules de mise à jour de la position sont :

$$\text{dist} = \frac{\Delta L + \Delta R}{2} \times d_{\text{step}}, \qquad \Delta\theta = \frac{(\Delta R - \Delta L) \times d_{\text{step}}}{W}$$

$$x \mathrel{+}= \text{dist} \times \sin(\theta), \qquad y \mathrel{-}= \text{dist} \times \cos(\theta)$$

où ΔL et ΔR sont les compteurs de demi-pas gauche et droit, $d_{\text{step}} = 0{,}528$ mm est la distance par demi-pas, et $W = 300$ mm est l'empattement.

Les mesures RSSI sont associées à la position (x, y) et stockées dans une grille 60×60 cellules, exposée via `/api/map` (JSON) et visualisée sur un canvas HTML5.

6.6 Extraits de Code

6.6.1 Conversion joystick vers commandes moteur

Code 6.1: Pilotage différentiel — conversion des coordonnées (x,y) en commandes gauche/droite

```
def xy_to_drive(x, y):
    L = max(-1.0, min(1.0, y + x))
    R = max(-1.0, min(1.0, y - x))
    ld = 0 if abs(L) < 0.1 else (1 if L > 0 else -1)
    rd = 0 if abs(R) < 0.1 else (1 if R > 0 else -1)
    return ld, rd
```

6.6.2 Séquence d'évitement d'obstacles

Code 6.2: Algorithme d'évitement — rotation 90° puis 180° si obstacle persistant

```
def do_avoidance():
    drive(0, 0)          # 1. Stop immediat
    time.sleep(0.3)
    drive(1, -1)          # 2. Rotation droite 90 deg
    time.sleep(ROTATION_90_SEC)
    drive(0, 0)
    time.sleep(0.5)
    with distance_lock:
        d = distance_cm
    if 0 < d < OBSTACLE_SEUIL:
        drive(1, -1)     # 3. Rotation 180 deg supplementaire
        time.sleep(ROTATION_180_SEC)
        drive(0, 0)
```

6.6.3 Mesure RSSI WiFi

Code 6.3: Lecture du RSSI WiFi via la commande iwconfig

```
def _get_rssi():
    try:
        out = subprocess.check_output(
            ["iwconfig", "wlan0"],
            stderr=subprocess.DEVNULL).decode()
        for part in out.split():
            if "level=" in part:
                return int(part.split("=")[1])
    except Exception:
```

```
        pass
    return None
```

6.6.4 Code ESP32 Zigbee — Émetteur (Coordinator / Manette)

Le XIAO#2, configuré en **Zigbee Coordinator**, lit le joystick analogique KY-023 et le bouton d'urgence, puis transmet la commande correspondante au End Device via le protocole Zigbee.

Code 6.4: zigbee-emmeteur.ino — lecture joystick et envoi Zigbee (Coordinator)

```
#ifndef ZIGBEE_MODE_ZCZR
#error "Selectionne Zigbee ZCZR dans Tools->Zigbee mode"
#endif

#include "Zigbee.h"

#define SWITCH_ENDPOINT 5
#define VRX_PIN A0
#define VRY_PIN A1
#define SW_PIN  2

ZigbeeColorDimmerSwitch zbSwitch = ZigbeeColorDimmerSwitch(
   SWITCH_ENDPOINT);

uint8_t cmdToLevel(String cmd) {
  if (cmd == "F") return 50;
  if (cmd == "B") return 100;
  if (cmd == "L") return 150;
  if (cmd == "R") return 200;
  if (cmd == "X") return 250;
  return 0;
}

String getCommand(int x, int y) {
  int dx = x - 2048, dy = y - 2048;
  if (abs(dx) < 500 && abs(dy) < 500) return "S";
  if (abs(dy) > abs(dx)) return (dy < 0) ? "F" : "B";
  return (dx > 0) ? "R" : "L";
}

void setup() {
  Serial.begin(115200);
  pinMode(SW_PIN, INPUT_PULLUP);
  zbSwitch.setManufacturerAndModel("RoboTank", "Controller");
  zbSwitch.allowMultipleBinding(true);
  Zigbee.addEndpoint(&zbSwitch);
  Zigbee.setRebootOpenNetwork(180);
  if (!Zigbee.begin(ZIGBEE_COORDINATOR)) {
    Serial.println("Zigbee failed, rebooting...");
    ESP.restart();
```

```
  }
  Serial.println("Attente du robot...");
  while (!zbSwitch.bound()) { Serial.print("."); delay(500); }
  Serial.println("\nRobot connecte !");
}

String lastCmd = "";
void loop() {
  String cmd = !digitalRead(SW_PIN) ? "X"
    : getCommand(analogRead(VRX_PIN), analogRead(VRY_PIN));
  if (cmd != lastCmd) {
    zbSwitch.setLightLevel(cmdToLevel(cmd));
    Serial.println("Envoi: " + cmd);
    lastCmd = cmd;
  }
  delay(50);
}
```

6.6.5 Code ESP32 Zigbee — Récepteur (End Device / Robot)

Le XIAO#1, configuré en **Zigbee End Device**, reçoit les commandes du Coordinator et les retransmet au Raspberry Pi via le port série USB à 115200 baud.

Code 6.5: Zigbee-reception.ino — reception Zigbee et transmission serie vers RPi

```
#ifndef ZIGBEE_MODE_ED
#error "Selectionne Zigbee ED dans Tools->Zigbee mode"
#endif

#include "Zigbee.h"

#define LIGHT_ENDPOINT 10

ZigbeeDimmableLight zbLight = ZigbeeDimmableLight(LIGHT_ENDPOINT);

String levelToCmd(uint8_t level) {
  if (level == 50)  return "F";
  if (level == 100) return "B";
  if (level == 150) return "L";
  if (level == 200) return "R";
  if (level == 250) return "X";
  return "S";
}

void onLightChange(bool state, uint8_t level) {
  Serial.println(levelToCmd(level));
}

void setup() {
  Serial.begin(115200);
  zbLight.setManufacturerAndModel("RoboTank", "Robot");
```

```
27    zbLight.onLightChange(onLightChange);
28    Zigbee.addEndpoint(&zbLight);
29    if (!Zigbee.begin()) {
30      Serial.println("Zigbee failed, rebooting...");
31      ESP.restart();
32    }
33    Serial.println("Connexion...");
34    while (!Zigbee.connected()) { Serial.print("."); delay(100); }
35    Serial.println("\nConnecte !");
36  }
37
38  void loop() { delay(100); }
```

7 Interface Web

7.1 Description Générale

L'interface web est un fichier HTML unique (`index.html`, $\sim$1235 lignes) servi par Flask et accessible depuis n'importe quel appareil connecté au même réseau WiFi. Elle adopte une esthétique **sombre de type terminal**, avec des couleurs néon (vert, cyan, rouge, jaune) et les polices *Chakra Petch* et *Space Mono*, évoquant l'univers des systèmes embarqués. L'interface est développée en **HTML5** / **CSS3** / **JavaScript vanilla**, sans framework externe.

L'interface est organisée en trois zones : le **panneau caméra** à gauche (flux MJPEG en temps réel), la **zone de contrôle** centrale (4 onglets), et le **panneau d'état** à droite (moteurs, distance, photos). La barre d'en-tête affiche le nom du robot ainsi que les **indicateurs de connexion** en temps réel : MOTEURS, CAMÉRA, SONAR (HC-SR04) et ZIGBEE.

7.2 Modes de Contrôle

7.2.1 Mode Joystick

Le mode Joystick est le **mode de contrôle principal**. Un joystick virtuel circulaire, utilisable au doigt (Touch API) ou à la souris, transmet les coordonnées (x, y) normalisées au serveur via `POST /api/drive` à une fréquence de 30 Hz. Une **zone morte de 10%** au centre assure un arrêt propre lorsque le doigt est relâché. Le contrôle au clavier est également possible via les touches Z, S, Q, D. En bas de l'écran, un slider permet de régler la vitesse des moteurs (de 1 à 10), et l'indicateur de mode affiche l'état actuel avec un bouton STOP d'arrêt rapide.

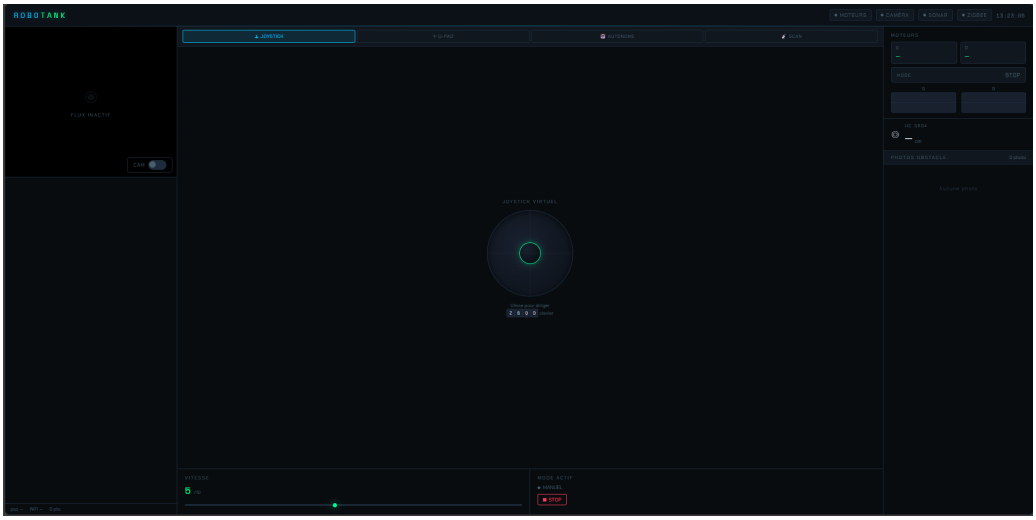


Figure 7.1: Interface web — mode Joystick : contrôle analogique du robot via un joystick virtuel tactile. Le panneau gauche montre le flux caméra, le panneau droit affiche l'état des moteurs et la distance.

7.2.2 Mode D-PAD

Le mode D-PAD propose un contrôle directionnel par **boutons discrets** : avant, arrière, gauche, droite, ainsi que deux boutons de rotation sur place (horaire et antihoraire). Ce mode est particulièrement adapté à un contrôle précis **pas-à -pas** pour les manœuvres de positionnement fin.

Les raccourcis clavier sont affichés sous le D-PAD : Z/S pour avant/arrière, Q/D pour gauche/droite, A/E pour les rotations. Un bouton `STOP URGENCE` rouge permet l'arrêt immédiat de tous les moteurs.

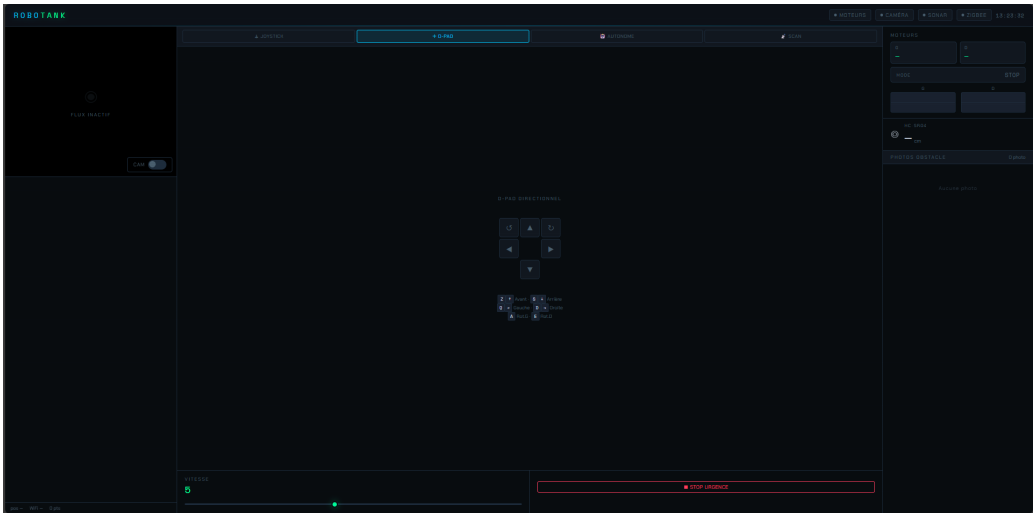


Figure 7.2: Interface web — mode D-PAD : contrôle directionnel par boutons avec raccourcis clavier ZQSD. Les boutons A et E permettent des rotations sur place.

7.2.3 Mode Autonome

L'onglet Autonome donne accès aux commandes du mode de **navigation autonome**. Quatre boutons principaux sont proposés :

- **Démarrer** (vert) : lance l'exploration libre avec évitement d'obstacles automatique
- **Arrêter** (rouge) : stoppe immédiatement le mode autonome
- **Retour** (vert) : déclenche le retour automatique au point de départ par dead reckoning
- **Urgence** (rouge) : arrêt forcé de tous les moteurs et de tous les modes actifs

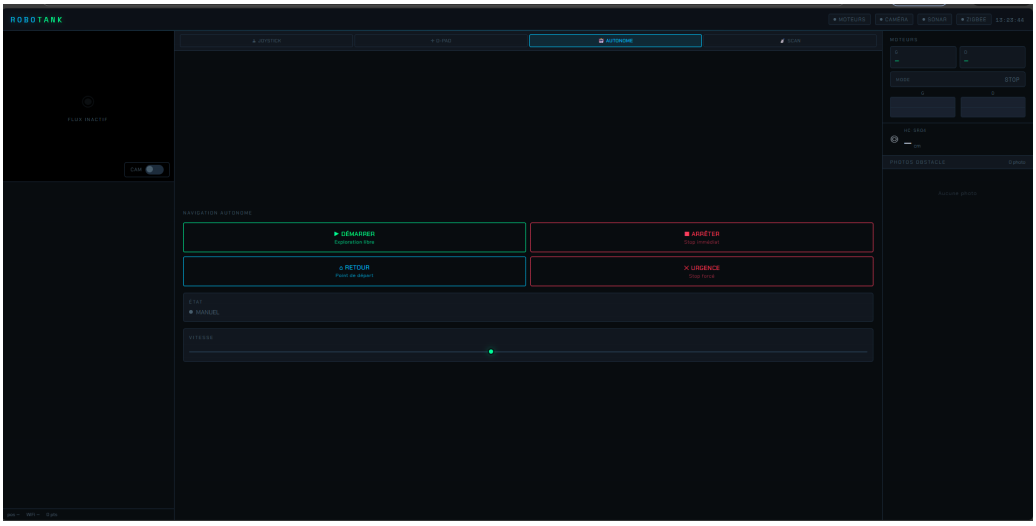


Figure 7.3: Interface web — mode Autonome : exploration libre avec évitement d'obstacles et possibilité de retour automatique au point de départ.

7.2.4 Mode SCAN RSSI

Le mode Scan permet de lancer une **cartographie WiFi systématique**. Le robot parcourt une grille de 5×5 points espacés de 25 cm sur une surface de 1 m^2, en suivant un motif en **serpentin**. À chaque point, **5 mesures RSSI** sont effectuées et moyennées pour obtenir une valeur fiable. Le résultat s'affiche sous forme de heatmap colorée sur le canvas de l'interface.

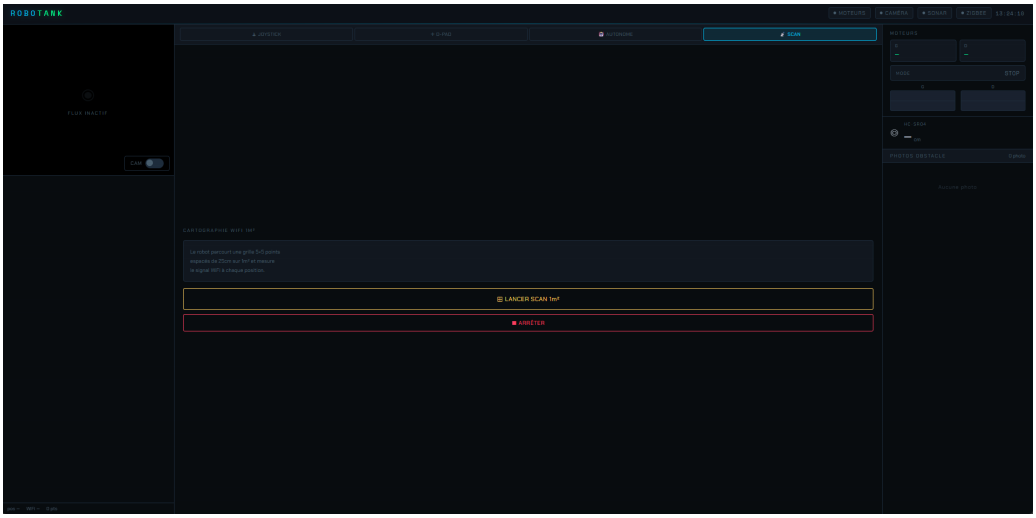


Figure 7.4: Interface web — mode SCAN : cartographie WiFi systématique sur 1 m² (grille 5×5, pas de 25 cm). La progression et les résultats s'affichent en temps réel.

7.2.5 Panneau latéral droit

Le panneau latéral droit, visible sur les quatre captures d'écran, affiche en permanence :

- **Moteurs** : état des moteurs gauche (G) et droit (D), avec indicateurs visuels de direction (flèches montantes/descendantes)
- **Mode** : le mode en cours (STOP, AVANT, ARRIÈRE, GAUCHE, DROITE, AUTO, SCAN)
- **HC-SR04** : la dernière distance mesurée par le capteur ultrasonique (en cm), avec barre de progression colorée
- **Photos obstacle** : compteur de photos prises automatiquement et aperçu de la dernière capture

7.3 Protocole Zigbee — Format des Commandes

Commande	Char	Action moteurs	Description
Avant	F	drive(1, 1)	Les deux moteurs avancent
Arrière	B	drive(-1, -1)	Les deux moteurs reculent
Gauche	L	drive(-1, 1)	Rotation sur place à gauche
Droite	R	drive(1, -1)	Rotation sur place à droite
Stop	S	drive(0, 0)	Arrêt immédiat
Bouton (urgence)	X	drive(0, 0)	Arrêt d'urgence depuis la manette

Chaque caractère est envoyé par le XIAO#1 (End Device Zigbee, dans la manette) et reçu par le XIAO#2 (Coordinator, dans le robot) qui le transmet au Raspberry Pi via `/dev/ttyACM0`. Le thread `xiao_reader()` décode le caractère et appelle la fonction `drive()` correspondante.

8 Tests et Validation

8.1 Plan de Tests

ID	Test	Résultat attendu	Statut
T-01	Autonomie $\geq$ 20 min	Chrono $\geq$ 20 min	✓
T-02	Évitement obstacle 20 cm (10 essais)	0 collision	✓
T-03	Photo sur événement obstacle	Image horodatée	✓
T-04	OLED — mode + RSSI visible	Lisible	✓
T-05	LED rouge sur obstacle	Allumée	✓
T-06	Heatmap après parcours	PNG valide	○
T-07	Stockage offline (WiFi coupé)	Logs présents	✓
T-08	Interface web — 4 modes accessibles	Tous fonctionnels	✓
T-09	Arrêt d'urgence logiciel	STOP immédiat	✓
T-10	Pilotage manette Zigbee	Latence $<$ 200 ms	✓
T-11	Scan RSSI systématique 5×5	25 points mesurés	✓

Tableau 8.1: Table de recette — plan de validation

Information

✓ Validé ○ En cours / partiel × Non validé

8.2 Analyse des Risques

! Risque : Perte WiFi

Impact : RSSI faible ou nul, déconnexion interface web.
Mesures : stockage offline sur SD, export différé, reconnexion automatique Flask.

! Risque : Collision / Blocage

Impact : obstacle non détecté (angle mort ou faux négatif HC-SR04).
Mesures : seuil 20 cm avec marge, stop + rotation 90°/180°, LED rouge, bouton urgence interface.

! Risque : Sous-tension 5 V

Impact : reboot Raspberry Pi lors des pointes courant moteur.
Mesures : powerbank USB délivrant au minimum 2–3 A, condensateurs 1000 µF + 100 nF sur la ligne moteurs, câble USB de qualité, masses communes.

! Risque : Perte de communication Zigbee

Impact : perte de la liaison manette-robot, le robot continue sa dernière commande.
Mesures : timeout série de 5 s (si aucune commande reçue, arrêt automatique des moteurs), reconnexion automatique du réseau Zigbee.

! Risque : Cybersécurité

Menaces : accès SSH non autorisé, interception WiFi, extraction carte SD.
Mesures : identifiants changés, SSH par clé, pare-feu UFW (ports minimaux), données non sensibles (RSSI et position uniquement).

9 Notice d'Utilisation

Cette notice décrit les procédures de mise en service, d'utilisation et d'arrêt du RoboTank. Elle est destinée à tout utilisateur souhaitant piloter le robot, même sans connaissance technique approfondie.

9.1 Contenu du Kit

Avant la première utilisation, vérifier la présence des éléments suivants :

- 1 robot RoboTank assemblé (châssis bleu imprimé 3D)
- 1 manette de télécommande Zigbee (boîtier Esplora, joystick, XIAO ESP32-C6)
- 1 batterie externe USB (powerbank) chargée
- 1 câble USB-A vers USB-C (alimentation RPi)
- 1 câble USB-C pour recharger la powerbank

9.2 Mise en Service

9.2.1 Étape 1 — Alimentation

1. Placer la **batterie externe USB** dans le compartiment prévu sous le châssis du robot.
2. Brancher le câble USB sur le port **USB-C** du Raspberry Pi 4.
3. Attendre ∼30 s que le Raspberry Pi démarre. Vérifier que l'écran **OLED** affiche l'adresse IP et le mode.
4. Allumer la manette Zigbee en branchant son alimentation (USB ou pile).

! Important

S'assurer que la powerbank est suffisamment chargée avant chaque mission. Un niveau de charge inférieur à 20% peut provoquer des coupures pendant le fonctionnement.

9.2.2 Étape 2 — Connexion WiFi

1. Sur votre appareil (PC, tablette, smartphone), se connecter au même réseau WiFi que le robot.
2. Ouvrir un navigateur web et saisir l'adresse : `http://robotank.local:5000`
3. L'interface web du RoboTank s'affiche avec le flux vidéo et les contrôles.

Information

Si `robotank.local` ne fonctionne pas, utiliser l'adresse IP du robot (visible sur l'écran OLED au démarrage). Exemple : `http://192.168.1.42:5000`

9.2.3 Étape 3 — Vérification de l'état

Avant toute mission, vérifier les indicateurs suivants sur l'interface web :

- **Chip MOTEURS** : vert (moteurs prêts)
- **Chip CAMÉRA** : vert (flux vidéo actif)
- **Chip SONAR** : vert (HC-SR04 opérationnel)
- **Chip ZIGBEE** : vert (manette connectée)

9.3 Modes de Pilotage

9.3.1 Mode Manette Zigbee (recommandé)

La manette physique (boîtier Esplora) permet un contrôle direct et réactif :

- **Joystick haut** : avancer (commande F)
- **Joystick bas** : reculer (commande B)
- **Joystick gauche** : tourner à gauche (commande L)
- **Joystick droite** : tourner à droite (commande R)
- **Joystick relâché** : arrêt (commande S)

9.3.2 Mode Interface Web — Joystick Virtuel

Depuis l'interface web, cliquer sur l'onglet **Joystick**. Utiliser le joystick tactile (souris ou écran tactile) pour diriger le robot. Le joystick revient automatiquement au centre à la relâche (arrêt du robot).

9.3.3 Mode Interface Web — D-PAD

Cliquer sur l'onglet **D-PAD**. Utiliser les boutons directionnels ou les raccourcis clavier :

- **Z** : avancer **S** : reculer
- **Q** : gauche **D** : droite
- **Espace** : arrêt d'urgence

9.3.4 Mode Autonome

Cliquer sur l'onglet **Autonome**, puis sur **Démarrer**. Le robot explore l'espace de manière autonome en évitant les obstacles. Boutons disponibles :

- **Démarrer** / **Arrêter** : lance ou stoppe l'exploration
- **Retour au départ** : le robot tente de revenir à sa position initiale
- **URGENCE** : arrêt immédiat de tous les moteurs

9.3.5 Mode SCAN WiFi

Cliquer sur l'onglet **SCAN**. Le robot effectue une cartographie WiFi sur une zone de 1 m² (grille 5×5, pas de 25 cm). Le résultat s'affiche sous forme de heatmap sur le canvas de l'interface.

9.4 Arrêt du Robot

1. Sur l'interface web, cliquer sur le bouton **URGENCE** ou **STOP** pour arrêter les moteurs.
2. Se connecter en SSH : `ssh robot@robotank.local` et exécuter : `sudo shutdown -h now`
3. Attendre l'extinction complète du Raspberry Pi (écran OLED éteint, ∼10 s).
4. Débrancher le câble USB de la powerbank.

! Ne pas débrancher brutalement

Toujours effectuer un arrêt logiciel (`shutdown`) avant de couper l'alimentation, afin d'éviter la corruption de la carte microSD.

9.5 Signification des Indicateurs

Indicateur	Signification
LED Rouge	S'allume lorsqu'un obstacle est détecté à moins de 20 cm — signale l'arrêt automatique de sécurité. Éteinte le reste du temps.
Écran OLED	Affiche en temps réel : mode actuel (MANUEL / AUTO / SCAN), RSSI WiFi (dBm), adresse IP, distance obstacle

9.6 Dépannage Rapide

Symptôme	Cause probable	Solution
Le robot ne démarre pas	Powerbank déchargée ou câble USB défectueux	Recharger la powerbank, essayer un autre câble
Pas d'accès à l'interface web	RPi non connecté au WiFi ou adresse erronée	Vérifier l'adresse sur l'écran OLED, vérifier le réseau WiFi
La manette ne répond pas	XIAO#1 non alimenté ou réseau Zigbee non formé	Vérifier l'alimentation de la manette, redémarrer le robot
Moteurs immobiles	Obstacle détecté (LED rouge) ou drivers ULN2003 déconnectés	Éloigner l'obstacle, vérifier le câblage des drivers
LED rouge allumée en permanence	Obstacle trop proche ($<$20 cm) ou capteur HC-SR04 défaillant	Éloigner l'obstacle, vérifier le câblage du capteur
Image caméra noire	Câble CSI mal inséré ou PiCamera non détectée	Reconnecter le câble plat CSI, redémarrer le RPi

10 Bilan et Perspectives

10.1 Compétences Mobilisées

⋆ Compétences techniques développées

- **Systèmes embarqués** : Raspberry Pi OS, Python, GPIO, I2C, CSI, threads
- **Réseaux sans fil** : WiFi, Zigbee 802.15.4, RSSI, cartographie de couverture
- **Programmation** : Python (Flask, Picamera2, threading), HTML5/CSS3/JS vanilla
- **Électronique** : câblage, bilan de puissance, PCB Fritzing, alimentation USB
- **Conception mécanique** : CAO Shapr3D, impression 3D FDM (PLA)
- **Gestion de projet** : planification, documentation technique, rédaction de rapport

10.2 Points Forts du Projet

- Architecture logicielle structurée et robuste (Flask multi-threadé, API REST)
- Interface web complète et ergonomique : 4 modes de contrôle, flux vidéo, scan WiFi
- Châssis entièrement imprimé 3D : solution originale, légère et modulable
- Communication Zigbee : faible latence, portée suffisante, faible consommation
- Autonomie théorique de 39 minutes, largement au-dessus du minimum requis
- Conception entièrement documentée : CAO, BOM, schémas, cahier des charges

10.3 Axes d'Amélioration et Perspectives d'Évolution

Le RoboTank dans sa version actuelle répond au cahier des charges imposé. Cependant, plusieurs évolutions sont envisageables pour transformer ce prototype en un système plus performant et plus précis. Les perspectives sont organisées par priorité : court terme (réalisable avec peu de modifications), moyen terme (nécessite de nouveaux composants) et long terme (refonte partielle).

10.3.1 Court terme — Optimisations logicielles

- **Heatmap temps réel** : génération dynamique de la carte WiFi sur le canvas web plutôt qu'export PNG post-traitement. Cela nécessite uniquement une modification de l'API `/api/map` et du JavaScript côté client.

- **Mode waypoints** : permettre à l'utilisateur de définir des points de passage sur l'interface web pour programmer un parcours de mesure WiFi.
- **Export des données** : ajouter un bouton de téléchargement CSV/JSON directement depuis l'interface web.

10.3.2 Moyen terme — Nouveaux capteurs

⋆ Encodeurs magnétiques sur les moteurs

L'ajout d'**encodeurs magnétiques** (type AS5600 ou encodeurs à effet Hall) sur les arbres des steppers 28BYJ-48 permettrait de passer d'un comptage de pas théorique à une mesure réelle de la rotation. Cela améliorerait considérablement la précision du dead reckoning en compensant :

- les pas perdus (charge, frottements, patinage)
- la dérive angulaire cumulée sur les longs parcours
- les erreurs de trajectoire dues à la dissymétrie des roues

Intégration estimée : 2 encodeurs AS5600 (∼3 € pièce), communication I2C, modification du module `explore_loop()` dans `server.py`.

⋆ Centrale inertielle (IMU)

L'intégration d'un **IMU 9 axes** (type MPU-9250 ou BNO055) permettrait d'obtenir :

- une mesure précise de l'orientation (θ) par fusion gyroscope + accéléromètre + magnétomètre
- la détection des collisions et des chocs par seuillage de l'accéléromètre
- une compensation de la dérive odométrique par filtre complémentaire ou filtre de Kalman

L'IMU viendrait en complément des encodeurs pour former un système de navigation par **odométrie fusionnée** nettement plus fiable que le dead reckoning pur.
Intégration estimée : 1 module BNO055 (∼10 €), bus I2C partagé avec l'OLED, bibliothèque Python `adafruit-circuitpython-bno055`.

10.3.3 Long terme — Vers le SLAM

⋆ Capteur LiDAR 2D

L'évolution majeure envisagée est l'intégration d'un **capteur LiDAR 2D** (type RPLiDAR A1 de SLAMTEC, ∼100 €) pour implémenter un algorithme de **SLAM** (Simultaneous Localization And Mapping). Le LiDAR permettrait :

- une cartographie précise de l'environnement en temps réel (nuage de points 2D, portée 12 m)
- la localisation du robot sans dépendre de l'odométrie seule
- la détection d'obstacles à 360° (supprimant les angles morts du HC-SR04 monodirectionnel)

- l'utilisation de bibliothèques SLAM existantes (Hector SLAM, GMapping via ROS2)

Cette évolution nécessiterait potentiellement une migration vers **ROS2** (Robot Operating System) pour bénéficier de l'écosystème de navigation existant (Nav2, cartographer, rviz).
Intégration estimée : RPLiDAR A1 (∼100 €), connexion USB, refonte logicielle significative, support mécanique imprimé 3D pour le montage en hauteur.

- **PCB dédié** : passer de la breadboard/câbles Dupont à un **shield PCB soudé** (conception KiCad ou Fritzing) pour renforcer la fiabilité mécanique des connexions et réduire les faux contacts.
- **Vision 360°** : remplacer la caméra fixe par un module caméra pivotant (servo + Pi-Camera) ou une caméra à objectif 360° pour couvrir l'intégralité de l'environnement.
- **Batterie intégrée** : à terme, intégrer une batterie LiPo dédiée avec BMS et chargeur embarqué pour un form factor plus compact et une meilleure gestion de l'énergie.

10.4 Plan de Maintenance

Pour garantir la fiabilité et la longévité du RoboTank, un plan de maintenance préventive est recommandé :

10.4.1 Maintenance Régulière (avant chaque utilisation)

- Vérifier la charge de la **batterie externe** (powerbank) — recharger si niveau $< 50\%$
- Contrôler visuellement les **câbles et connecteurs** (pas de fil dénudé, pas de faux contact)
- S'assurer que les **roues tournent librement** (pas de débris bloquant la rotation)
- Vérifier la propreté de l'**objectif de la caméra** (poussière, traces de doigts)
- Tester l'**écran OLED** : il doit afficher l'adresse IP et le mode au démarrage

10.4.2 Maintenance Mensuelle

- **Mise à jour système** : via SSH, exécuter `sudo apt update && sudo apt upgrade -y` puis `sudo reboot`
- **Sauvegarde des logs** : récupérer les fichiers CSV/JSON et les photos depuis `/home/robot/logs/` et `/home/robot/photos/`
- **Nettoyage de la carte SD** : purger les anciens logs si l'espace disque devient insuffisant ($<$1 Go libre)
- **Inspection mécanique** : vérifier les vis et clips du châssis imprimé 3D, resserrer si nécessaire
- **Test du capteur HC-SR04** : placer un obstacle à distance connue et comparer la mesure affichée

10.4.3 Maintenance Corrective

Panne	Diagnostic	Action
Moteur ne tourne plus	Driver ULN2003 HS ou câble coupé	Remplacer le driver ou le câble
OLED éteint	Bus I2C déconnecté ou OLED HS	Vérifier câblage SDA/SCL, tester avec `i2cdetect`
Valeur HC-SR04 aberrante	Capteur désaligné ou défectueux	Réaligner le capteur, remplacer si nécessaire
Zigbee intermittent	Antenne XIAO obstruée ou interférence	Dégager l'antenne, changer de canal Zigbee
Reboot inopiné du RPi	Sous-tension USB (powerbank faible)	Utiliser une powerbank $\geq$ 2 A, câble court

10.5 Bilan Personnel

Ce projet constitue l'aboutissement de trois années de formation en BUT Réseaux & Télécommunications. Il représente une expérience d'ingénierie complète : de l'analyse fonctionnelle à la démonstration d'un prototype pleinement fonctionnel.

La principale difficulté rencontrée a été l'intégration de la chaîne complète : faire coopérer simultanément le Zigbee, le Raspberry Pi, les steppers, la caméra IR et l'interface web sans instabilité. La résolution de ces problèmes d'intégration a été l'aspect le plus formateur du projet.

Conclusion

Le projet **RoboTank** répond à l'ensemble des fonctionnalités du cahier des charges de la SAÉ6.IOM.01 : déplacement autonome et télécommandé via Zigbee, détection et évitement d'obstacles, capture photo horodatée, mesure RSSI WiFi, feedback OLED et LEDs, et interface web Flask complète.

L'architecture hybride **Raspberry Pi 4 + XIAO ESP32-C6 (Zigbee)**, alimentée par une batterie externe USB pour sa simplicité et sa sécurité, couplée à un châssis entièrement modélisé en CAO et imprimé en 3D, constitue une solution originale, évolutive et bien adaptée au contexte académique. L'évolution de la télécommande — d'une manette Arduino Esplora avec Bluetooth HC-05 vers une solution Zigbee intégrée dans le même boîtier — illustre la démarche itérative d'amélioration continue qui a guidé ce projet.

Les perspectives d'évolution identifiées (encodeurs magnétiques, IMU, LiDAR pour le SLAM) ouvrent la voie à un robot de surveillance véritablement autonome et précis dans sa cartographie.

Ce projet a démontré la capacité à mener de bout en bout un objet connecté : de la spécification fonctionnelle à la démonstration physique, en mobilisant des compétences en électronique, réseaux, systèmes embarqués et développement logiciel.

« De la conception à la démonstration — RoboTank. »

A Annexes

A.1 Notice de Montage (résumé)

Cette section reprend de manière synthétique les étapes de montage du robot, détaillées au chapitre 4 (Conception Mécanique, section « Procédure d'Assemblage Mécanique »).

A.1.1 Prérequis

Avant de commencer le montage, s'assurer de disposer de :

- Toutes les pièces imprimées 3D (voir tableau des temps d'impression, chapitre 4)
- L'ensemble des composants électroniques (voir BOM, chapitre 3)
- Un tournevis cruciforme, des vis M2.5, de la colle cyanoacrylate
- Un multimètre pour la vérification électrique

A.1.2 Étapes de montage résumées

1. **Base motrice** : fixer les 2 moteurs stepper 28BYJ-48 + drivers ULN2003 sur le châssis inférieur. Monter les roues et la bille folle.
2. **Structure** : assembler les supports châssis (liaison haut/bas) et les colonnes. Fixer le châssis supérieur.
3. **Raspberry Pi** : installer le RPi 4 sur le châssis supérieur (4 vis M2.5). Brancher la nappe CSI vers la PiCamera IR.
4. **Bumper avant** : monter le bras de fixation capteur, installer le HC-SR04 dans le bumper, fixer l'OLED et la LED rouge.
5. **Câblage** : connecter les ULN2003 aux GPIO du RPi, brancher l'OLED (I2C), la LED rouge (GPIO + 220 Ω), le XIAO Coordinator (USB) et l'ESP32 HC-SR04 (USB).
6. **Alimentation** : placer la powerbank USB et la connecter au port USB-C du RPi.
7. **Vérification** : contrôler l'absence de court-circuit, la tension 5 V, le bon contact de tous les connecteurs.
8. **Premier démarrage** : brancher, attendre $\sim$30 s, vérifier l'OLED (affichage IP et mode).

Temps de montage estimé : 2 à 3 heures (vérification progressive incluse).

! Rappel de sécurité

Toujours vérifier l'absence de court-circuit avant la première mise sous tension. Ne jamais brancher/débrancher de câbles avec l'alimentation active.

A.2 Fichiers du Projet

Fichier	Description
`server.py`	Serveur Flask — API REST + threads dédiés
`index.html`	Interface web — joystick, D-PAD, autonome, scan
`/home/robot/logs/`	Répertoire CSV/JSON des mesures RSSI
`/home/robot/photos/`	Photos horodatées (événement obstacle)
`Cahier_des_charges.tex`	Document technique de référence (LaTeX)

A.3 Commandes de Maintenance

Code A.1: Mise à jour système du robot

```
# Connexion SSH au robot
ssh robot@robotank.local

# Mise à jour des paquets système
sudo apt update && sudo apt upgrade -y

# Redémarrage du service Flask
sudo systemctl restart robotank.service
# ou redémarrage complet
sudo reboot
```

Code A.2: Récupération des logs et photos

```
# Depuis votre ordinateur
scp -r robot@robotank.local:/home/robot/logs/   ./logs_robotank/
scp -r robot@robotank.local:/home/robot/photos/ ./photos_robotank/
```

A.4 Signification des LEDs et de l'OLED

Indicateur	Signification
LED Rouge	S'allume en cas d'obstacle détecté à moins de 20 cm (arrêt de sécurité). Éteinte sinon.
OLED	Affiche : mode actuel (MANUEL / AUTO / SCAN), RSSI WiFi, adresse IP, distance obstacle

A.5 Devis Estimatif (Matériel et Humain)

A.5.1 Coût matériel (valeur de remplacement)

Le matériel utilisé provient majoritairement du stock de l'IUT. Le tableau suivant présente les coûts à titre indicatif, en valeur de remplacement si les composants devaient être rachetés.

Composant	Qté	Prix unit.	Total
Raspberry Pi 4 Model B 4 Go	1	75,00 €	75,00 €
Carte microSD 32 Go	1	8,00 €	8,00 €
PiCamera IR Fisheye 1080p	1	25,00 €	25,00 €
Capteur ultrason HC-SR04	1	2,50 €	2,50 €
Écran OLED I2C 0,96" SSD1306	1	5,00 €	5,00 €
XIAO ESP32-C6 (Seeed Studio)	2	6,00 €	12,00 €
ESP32 DevKit (capteur ultrason)	1	8,00 €	8,00 €
Moteurs stepper 28BYJ-48 + ULN2003	2	3,50 €	7,00 €
LEDs 5 mm (V/R/O) + résistances 220 Ω	6	0,10 €	0,60 €
Batterie externe USB (powerbank)	1	15,00 €	15,00 €
Filament PLA bleu (∼370 g)	1	7,40 €	7,40 €
Câbles Dupont, JST, borniers, vis	lot	5,00 €	5,00 €
Joystick analogique + boutons	1	3,00 €	3,00 €
Condensateurs 1000 µF + 100 nF	2+	0,50 €	1,00 €
TOTAL MATÉRIEL			**174,50 €**

Tableau A.1: Devis matériel — valeur de remplacement

Information

En pratique, l'ensemble du matériel a été fourni par le stock de l'IUT de Blois. Seul le filament PLA (∼7,40 €) constitue un coût réel. Le budget effectif du projet est donc quasi nul.

A.5.2 Coût humain estimé

Phase	Heures	Taux horaire	Coût estimé
Analyse fonctionnelle et CDC	8 h	35 €/h	280 €
Conception CAO (Shapr3D)	15 h	35 €/h	525 €
Impression 3D (supervision)	17 h	15 €/h	255 €
Câblage et électronique	10 h	35 €/h	350 €
Développement logiciel (Python/Flask)	40 h	40 €/h	1 600 €
Développement interface web	20 h	40 €/h	800 €
Programmation Zigbee (XIAO/ESP32)	12 h	40 €/h	480 €
Tests et validation	10 h	35 €/h	350 €
Rédaction documentation	15 h	30 €/h	450 €
TOTAL HUMAIN	**∼147 h**		**∼5 090 €**

◎ Information

Le coût humain est estimé à titre indicatif, sur la base de taux horaires moyens pour un technicien / développeur junior. Le projet a mobilisé environ **147 heures** de travail sur une période de 7 semaines (février à mars 2026).

A.6 Planification — Diagramme de Gantt

Le projet s'étend sur **7 semaines**, de début février à mi-mars 2026, avec une soutenance le **20 mars 2026**. Le diagramme ci-dessous présente les principales phases et leur enchaînement.

Phase	S1	S2	S3	S4	S5	S6	S7
Analyse fonctionnelle							
Cahier des charges							
Conception CAO (Shapr3D)							
Impression 3D							
Câblage électronique							
Dév. logiciel (Python)							
Interface web (Flask)							
Zigbee (XIAO/ESP32)							
Tests et validation							
Rédaction rapport							
Soutenance (20/03)							

Information

S1 = semaine du 2 février 2026, **S7** = semaine du 16 mars 2026 (soutenance le vendredi 20 mars).
Les phases de développement logiciel et d'impression 3D se chevauchent, ce qui a permis d'optimiser le temps global du projet. La rédaction du rapport a été menée en parallèle des tests finaux.

A.7 Glossaire

Terme	Définition
API REST	Interface de programmation basée sur le protocole HTTP, permettant la communication entre l'interface web et le serveur Flask du robot.
BMS	Battery Management System — circuit de protection intégré dans les batteries, gérant la surcharge, la surdécharge et les courts-circuits.
CAO	Conception Assistée par Ordinateur — ici réalisée avec Shapr3D (iPad/Mac) pour la modélisation 3D du châssis.
CSI	Camera Serial Interface — bus de communication série utilisé pour connecter la PiCamera au Raspberry Pi.
Dead Reckoning	Méthode d'estimation de la position par intégration des déplacements successifs (odométrie), sans capteur de position absolue.
FDM	Fused Deposition Modeling — procédé d'impression 3D par dépôt de filament fondu, utilisé pour le châssis en PLA.
Flask	Framework web léger en Python, utilisé comme serveur HTTP embarqué sur le Raspberry Pi.
GPIO	General Purpose Input/Output — broches du Raspberry Pi utilisées pour piloter les moteurs, LEDs et capteurs.
Heatmap	Carte de chaleur — représentation graphique colorée des niveaux de RSSI WiFi sur une zone géographique.
HC-SR04	Capteur de distance ultrasonique mesurant de 2 à 400 cm par temps de vol d'une onde acoustique à 40 kHz.
I2C	Inter-Integrated Circuit — bus de communication série utilisé pour l'écran OLED SSD1306 (adresse 0x3C).
IMU	Inertial Measurement Unit — centrale inertielle combinant accéléromètre, gyroscope et magnétomètre.
IoT	Internet of Things — réseau d'objets connectés, ici le RoboTank communiquant via WiFi et Zigbee.
LiDAR	Light Detection And Ranging — capteur laser mesurant des distances par temps de vol d'un faisceau lumineux.

Terme	Définition
MJPEG	Motion JPEG — format de flux vidéo constitué d'images JPEG successives, utilisé pour le stream caméra.
OLED	Organic Light-Emitting Diode — technologie d'affichage utilisée pour l'écran 0,96" du robot (SSD1306).
PLA	Acide polylactique — thermoplastique biodégradable utilisé comme filament d'impression 3D.
RSSI	Received Signal Strength Indicator — mesure de la puissance du signal WiFi reçu, exprimée en dBm.
SAÉ	Situation d'Apprentissage et d'Évaluation — projet intégrateur du BUT R&T.
SLAM	Simultaneous Localization And Mapping — algorithme permettant de cartographier un environnement tout en s'y localisant.
SSD1306	Contrôleur d'écran OLED monochrome 128×64 pixels, communiquant en I2C.
Stepper	Moteur pas à pas — ici le 28BYJ-48 (512 demi-pas/tour), piloté via un driver ULN2003.
ULN2003	Circuit intégré Darlington servant de driver pour les moteurs stepper 28BYJ-48.
Zigbee	Protocole de communication sans fil basé sur IEEE 802.15.4, utilisé pour la liaison manette-robot.

Tableau A.2: Glossaire des termes techniques

A.8 Bibliographie et Références

[1] **Raspberry Pi Foundation** — Documentation officielle Raspberry Pi 4 Model B.
https://www.raspberrypi.com/documentation/
[2] **Seeed Studio** — XIAO ESP32-C6 — Getting Started.
https://wiki.seeedstudio.com/xiao_esp32c6_getting_started/
[3] **Flask** — Documentation officielle du framework Flask (Pallets Projects).
https://flask.palletsprojects.com/
[4] **Picamera2** — Documentation de la bibliothèque Python pour les caméras Raspberry Pi.
https://datasheets.raspberrypi.com/camera/picamera2-manual.pdf
[5] **Shapr3D** — Application CAO 3D (iPad/Mac) utilisée pour la modélisation du châssis.
https://www.shapr3d.com/
[6] **Fritzing** — Outil de conception de circuits électroniques.
https://fritzing.org/
[7] **HC-SR04** — Datasheet du capteur ultrasonique.
https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf
[8] **SSD1306** — Datasheet du contrôleur OLED 128×64.
https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf
[9] **28BYJ-48** — Datasheet du moteur stepper et driver ULN2003.
https://components101.com/motors/28byj-48-stepper-motor
[10] **Zigbee Alliance** — Spécification Zigbee 3.0 / IEEE 802.15.4.
https://csa-iot.org/all-solutions/zigbee/