# Dossier De Conception (DDC)

du projet

# Robot Mini-Sumo

## Responsabilité documentaire

| Action | NOM Prénom | Fonction | Date | Signature |
|---|---|---|---|---|
| Rédigé par | AVIGNI Matteo (équipe 13) | Technicien | 13/11/2026 | |
| Approuvé par | <Chef de Projet> (IUT GEII Bdx) | Chef de projet | JJ/MM/AAAA | |
| Approuvé par | <Client> (IUT GEII Bdx) | Client | JJ/MM/AAAA | |

## Suivi des révisions documentaires

| Indice | Date | Nature de la révision |
|---|---|---|
| 1 | 01/09/2022 | Publication préliminaire du DDC<br>document à compléter par le Technicien. |
| 2 | 15/10/2026 | Première publication (13_DDC_EQ13_V1) |
| 3 | 13/11/2026 | V2 : FAD par solution, coûts, conformité de conception,<br>corrections CAO et code du 13/11/2026 |
| | | |
| | | |

## Documents de références

| Sigle | Référence | Titre | Rév. | Origine |
|---|---|---|---|---|
| [CDC] | RMS-CDC v5-ESE | Cahier des charges | 5.0 | IUT GEII Bdx |
| [CAO] | robot sumo (KiCad 9) | Projet KiCad : schéma et PCB | 13/11/26 | Équipe 13 |
| [DDF] | 13_DDF_EQ13_V1 | Dossier de fabrication | 1 | Équipe 13 |
| [IR] | DDF télécommande | Récepteur de départ IR (M. Blanchard) | — | IUT GEII Bdx |
| [CDP] | RMS-CDP | Coût de développement : catalogue de prix | 1.9 | IUT GEII Bdx |
| [PDP] | RMS_PDP | Planning de développement du projet | 4.1 | IUT GEII Bdx |

# Table des matières

**1. Nature du document....5**

**2. Conception préliminaire du produit....5**
**2.1 Architecture électronique....6**
2.1.1 Choix des capteurs adversaire....8
2.1.2 Choix des capteurs de sol....9
2.1.3 Récepteur de l'ordre de départ....10
2.1.4 Choix du pont en H....11
2.1.5 Choix des régulateurs de tension....11
2.1.6 Batterie, protection et interrupteur....13
2.1.7 Choix de la centrale inertielle....15
2.1.8 Choix du microcontrôleur....16
**2.2 Architecture informatique....17**
2.2.1 Fonctions d'acquisition....18
2.2.2 Fonctions d'action....18
2.2.3 Fonctions d'énergie....18
2.2.4 Fonctions de traitement....19
**2.3 Coûts prévisionnels du produit....21**
**2.4 Réponses préliminaires aux exigences du CDC....23**
Mécanique et implantation....24
Alimentation et sûreté électrique....24
Perception, départ et comportement....24
Carte électronique et schéma....25
Implantation et fabrication PCB....25
Logiciel....26
Projet, coût et livrables....26
Sources techniques....27
**2.5 Conclusion de la conception préliminaire du produit....27**

**3. Conception détaillée du produit....29**
**3.1 Conception détaillée du robot mini-sumo....29**
**3.2 Conception détaillée de la partie acquisition....29**
3.2.1 Capteurs adversaires et centrale inertielle....29
3.2.1.1 Schéma électrique des capteurs adversaires....29
3.2.1.2 Code informatique des capteurs adversaires....32
3.2.2 Capteurs de sol....34
3.2.2.1 Schéma électrique des capteurs de sol....34
3.2.2.2 Code informatique des capteurs de sol....36
3.2.3 Capteur de télécommande....37
3.2.3.1 Schéma électrique du capteur de télécommande....37
3.2.3.2 Code informatique du capteur de télécommande....38
**3.3 Conception détaillée de la partie action....39**
3.3.1 Pont en H....39
3.3.1.1 Schéma électrique du pont en H....39
3.3.1.2 Code informatique du pont en H....41
**3.4 Conception détaillée de la partie énergie....42**
3.4.1 Pont diviseur de tension....42
3.4.1.1 Schéma électrique du pont diviseur de tension....42
3.4.1.2 Code informatique du pont diviseur de tension....43
3.4.2 Régulateurs de tension....45
3.4.2.1 Schéma électrique des régulateurs de tension....45

3.4.3 Protection, marche/arrêt et coupure du servo....47
3.4.3.1 Schéma électrique de la protection et des coupures....47
3.4.3.2 Code informatique de la coupure et du voyant....48
**3.5 Conception détaillée de la partie traitement....49**
3.5.1 Le microcontrôleur ESP32-S3-WROOM-1-N4....49
3.5.1.1 Schéma électrique du microcontrôleur....49
3.5.1.2 Code informatique du microcontrôleur....51
**4. Dérisquage des solutions techniques retenues....54**
**4.1 Mesure de la tension batterie (simulation)....54**
**4.2 Commande des motoréducteurs par le DRV8835 (prototypage rapide)....56**
**4.3 Détection de la bordure par CNY70 (prototypage rapide)....57**
**4.4 Entraînement et validation de la politique de décision (simulation)....58**
**4.5 Simulation des combats avec le contrôleur de référence....61**
**4.6 Suivi de cible de la tourelle (simulation du code réel)....62**
**4.7 Réception de l'ordre de départ (simulation du code réel)....63**
**4.8 Contrôles ERC et DRC de la CAO....64**
**4.9 Conclusion de la simulation / prototypage rapide du produit....65**
**5. Conclusion de la conception du produit....66**
**6. Matrice de conformité du produit....67**

# 1. Nature du document

Ce document est le dossier de conception (DDC) du robot mini-sumo de l'équipe 13. Il détaille la conception du produit développé et apporte, pour chaque exigence du cahier des charges [CDC], les éléments de preuve de la conformité : choix argumentés par des fiches d'aide à la décision, dimensionnements, schémas, code, simulations et prototypages rapides. Le paragraphe 3 du [CDC] précise la nature et le positionnement de ce document dans l'arborescence documentaire du projet.

Il décrit le projet KiCad « robot sumo » (schéma hiérarchique à quatre feuilles, carte routée) fabriqué selon le dossier de fabrication [DDF], dans sa révision du 13/11/2026. Le chapitre 2 présente l'architecture, la sélection de chaque solution technique et le coût prévisionnel ; le chapitre 3 détaille chaque bloc fonctionnel ; le chapitre 4 regroupe les simulations et prototypages ; la matrice du chapitre 6 synthétise la conformité de la conception. La vérification sur le robot assemblé fait l'objet du dossier de vérification (DDV).

# 2. Conception préliminaire du produit

Ce chapitre décrit l'architecture fonctionnelle du produit. Il apporte les premiers éléments de preuve de la faisabilité du produit vis-à-vis des exigences client.

**Démarche de sélection des solutions techniques**

Chaque solution technique à choisir est sélectionnée par une fiche d'aide à la décision (FAD) construite sur le modèle du gabarit :

- la colonne **Conformité** vaut 1 si la solution peut satisfaire toutes les exigences du [CDC], 0 sinon : elle est éliminatoire ;
- chaque **critère de sélection** reçoit une note de 1 (défavorable) à 5 (favorable), établie à partir de la caractéristique chiffrée écrite au-dessus de la note, tirée des fiches techniques [S…] ou de nos essais ;
- le **total** est le produit de la conformité et des notes : une seule note très faible pénalise fortement une solution ; la solution de plus grand total est retenue (ligne surlignée) ;
- une **synthèse** explique le choix et ses limites ; la **fonctionnalité** de la solution retenue et les **caractéristiques de l'information** à son entrée et à sa sortie sont ensuite décrites.

Les composants imposés par le [CDC] ou par le client (motoréducteurs GM9, batterie LiPo 2S de 500 mAh, interrupteur à bascule de la base, récepteur de départ du dossier [IR]) ne font pas l'objet d'une FAD ; leur fonctionnalité est décrite de la même façon.

## 2.1 Architecture électronique

**Référence de pré-conception :** CPR_ARCHI_ELEC

**Exigences client vérifiées par préconception :** EXIG_CARTE, EXIG_SECUR_BATT, EXIG_PROT_INV_POL, EXIG_INTER, EXIG_ADVERSAIRE, EXIG_LUMINOSITE, EXIG_DEPART, EXIG_DEPLACEMENT, EXIG_COMPORTEMENT, EXIG_CARTE_VISU

L'analyse fonctionnelle du robot conduit à quatre blocs : acquisition, traitement, action et énergie (figure 1).

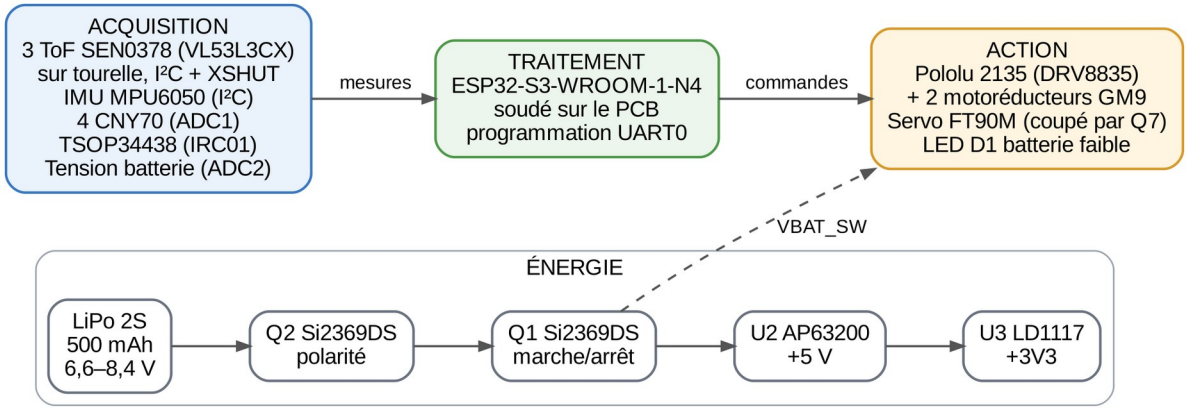



**Figure 1 :** Synoptique de l'architecture électronique du robot mini-sumo

- **Énergie** : la LiPo 2S traverse Q2 (protection contre l'inversion de polarité) puis Q1 (interrupteur électronique commandé par l'interrupteur à bascule relié à J2). La tension commutée VBAT_SW alimente directement le pont en H et le convertisseur abaisseur U2 (AP63200), qui fournit le +5 V au servomoteur et au régulateur linéaire U3 (LD1117S33), lequel crée le +3V3 de la logique et des capteurs.
- **Acquisition** : trois télémètres ToF et la centrale inertielle partagent le bus I²C ; chaque télémètre a sa ligne XSHUT. Les quatre CNY70 rejoignent quatre entrées ADC1, le récepteur de départ une entrée numérique et le pont diviseur de la batterie une entrée ADC2.
- **Traitement** : le module ESP32-S3-WROOM-1-N4, soudé sur la carte, exécute le suivi de cible de la tourelle et la politique de décision des roues (§ 2.2).
- **Action** : le module Pololu 2135 (DRV8835) commande les deux motoréducteurs GM9 ; le servomoteur FT90M oriente la tourelle ; la LED D1 signale la batterie faible.

Le tableau 1 donne les caractéristiques de l'information à l'entrée et à la sortie de chaque bloc.

**Tableau 1 :** Caractéristiques de l'information à l'entrée et à la sortie de chaque bloc

| Bloc | Information en entrée | Information en sortie |
|---|---|---|
| **Acquisition** | distance de l'adversaire (25 à 3000 mm, lumière IR 940 nm réfléchie) ; réflectance du sol (noir ou bande blanche de 25 mm) ; trame IR modulée à 38 kHz de l'IRC01 ; rotation (±500 °/s) et accélération (±4 g) du châssis ; tension batterie 6,6 à 8,4 V | I²C 400 kHz, niveaux 3,3 V : jusqu'à 4 cibles par télémètre (mm, sigma, signal), 6 mesures IMU sur 16 bits ; 4 tensions analogiques 0,3 à 3,3 V (blanc = tension basse) ; signal logique actif bas 3,3 V (trame démodulée) ; VBAT_ADC analogique 1,64 à 2,08 V |
| **Traitement** | signaux de l'acquisition ci-contre ; ordres de programmation par UART0 (115 200 bit/s) | 2 × (PHASE logique 0/3,3 V + ENABLE MLI 20 kHz, 8 bits) ; SERVO_PWM 50 Hz, 1000 à 2000 µs ; ACT_ENABLE 0/3,3 V ; LED_LOW_BAT 0/3,3 V (clignotement 2 Hz) ; 3 × XSHUT 0/3,3 V |
| **Action** | consignes logiques 3,3 V du traitement ; puissance VBAT_SW 6,6 à 8,4 V et +5 V | tension moteur ±α·VBAT (α rapport cyclique), jusqu'à 1,2 A par voie ; rotation des roues (GM9, 78 tr/min sous 6 V) ; orientation de la tourelle ±140° ; état visuel (LED D1) |
| **Énergie** | LiPo 2S 6,6 à 8,4 V, 500 mAh ; contact de l'interrupteur à bascule (84 µA) | VBAT_SW 6,6 à 8,4 V (3 A max.) ; +5 V : 4,86 à 5,12 V (2 A max.) ; +3V3 : 3,16 à 3,44 V (0,8 A max.) ; masse commutée du servo GND_SERVO |

Le schéma KiCad est hiérarchique : une feuille racine relie les feuilles « microcontrolleur », « alimentation » et « capteurs » par des étiquettes nommées (figure 2). Quatre points de test (VBAT_SW, +5 V, +3V3, GND) facilitent la mise sous tension.

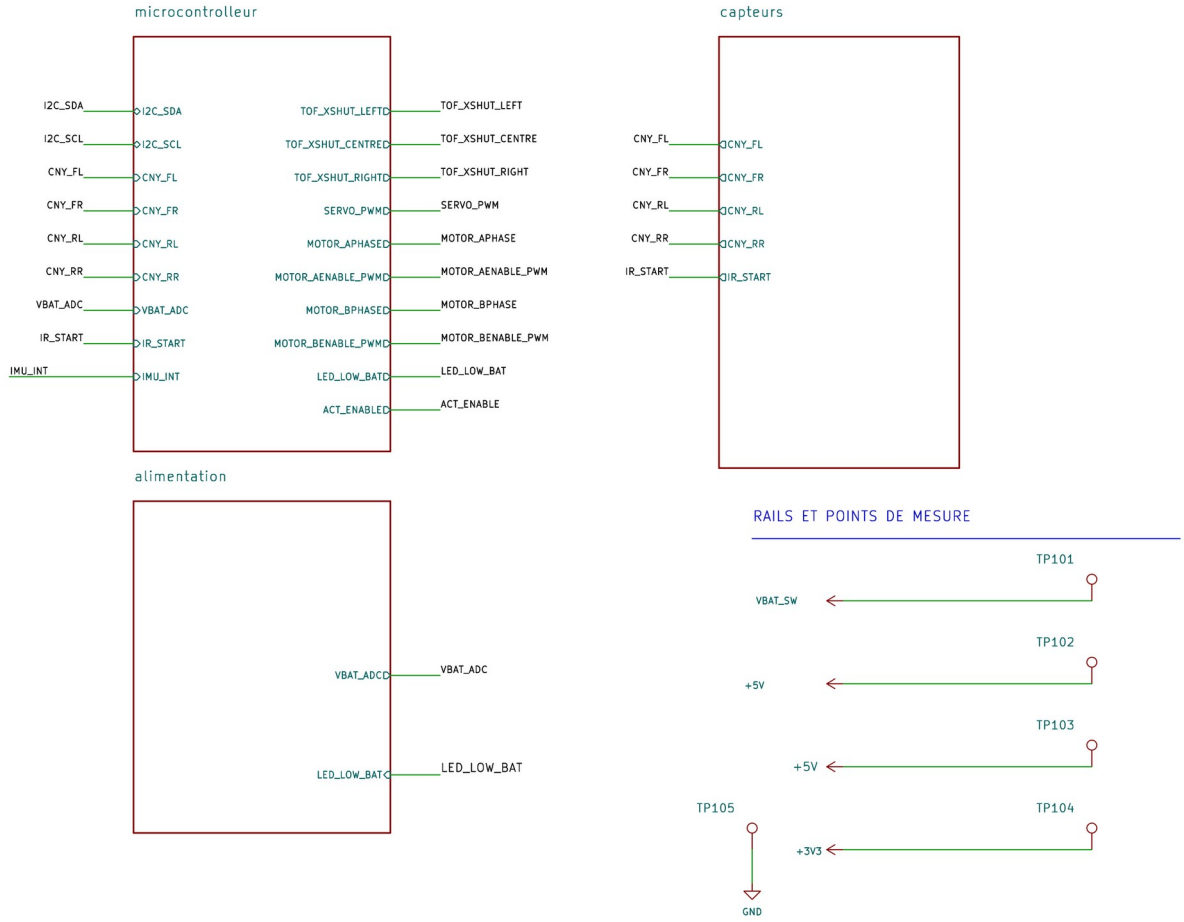



**Figure 2 :** Feuille racine du schéma : sous-feuilles et liaisons (à gauche), feuille capteurs et points de test (à droite)

À la mise sous tension, les rapports cycliques des moteurs sont nuls et ACT_ENABLE est maintenu à 0 par R6 = 100 kΩ : le servomoteur n'a pas de retour de masse tant que le programme ne l'a pas autorisé. L'arrêt des moteurs est logiciel (ENABLE = 0, freinage), choix validé par l'enseignant lors de la revue du 13/10/2026.

### 2.1.1 Choix des capteurs adversaire

**Référence de pré-conception :** CPR_ADVERSAIRE

**Exigences client vérifiées par préconception :** EXIG_ADVERSAIRE, EXIG_LUMINOSITE, EXIG_CARTE

Le robot doit savoir si l'adversaire est devant lui à 250 mm au moins (EXIG_ADVERSAIRE), sur un dohyo de 770 mm de diamètre, sous un éclairage qui peut être intense (EXIG_LUMINOSITE). Le [CDC] demande de justifier la référence, la quantité et la position des capteurs : deux FAD sont donc établies.

#### Technologie et référence du capteur

**Tableau 2 :** FAD pour le choix de la technologie des capteurs adversaire

| Solution technique proposée | Conf. (1/0) | Portée utile | Interface 3,3 V | Position de la cible | Éclairage | Achat | Total (produit) | Remarque |
|---|---|---|---|---|---|---|---|---|
| **Ultrasons HC-SR04** | 1 | 20 à 4000 mm **5** | 5 V, écho 5 V : adaptation **2** | cône ≈ 30° **2** | insensible **5** | hors liste **3** | **300** | encombrant (45 × 20 mm) |
| **Sharp GP2Y0A41SK0F (triangulation IR)** | 1 | 40 à 300 mm **2** | 4,5 à 5,5 V, sortie analogique **3** | faisceau étroit **4** | sensible au soleil **3** | liste CDP **5** | **360** | ne voit pas au-delà de 300 mm |
| **ST VL53L4CD (temps de vol)** | 1 | 1 à 1300 mm **5** | 2,6 à 3,5 V, I²C **5** | 18°, 1 cible **4** | VCSEL 940 nm filtré **4** | Farnell **4** | **1 600** | 17,75 € HT |
| **ST VL53L3CX, module DFRobot SEN0378** | 1 | 25 à 3000 mm **5** | 3,3 à 5 V, I²C + XSHUT **5** | 25°, jusqu'à 4 cibles **5** | VCSEL 940 nm filtré **4** | Farnell 3879709 **4** | **2 000** | 9,90 € HT |

**Synthèse de la sélection.** Le **VL53L3CX** sur module SEN0378 obtient le meilleur total (2 000 contre 1 600 pour le VL53L4CD). Sa portée de 25 à 3000 mm couvre les 250 mm exigés et tout le dohyo ; son histogramme multi-cibles distingue l'adversaire d'un objet d'arrière-plan situé à une autre distance ; il est alimenté en 3,3 V et dialogue en I²C sans adaptation de niveau [S11]. Les ultrasons sont écartés pour leur cône trop large et leur logique 5 V, le Sharp pour sa portée limitée à 300 mm, le VL53L4CD pour son prix presque double à performances voisines. La mesure par temps de vol d'une impulsion laser filtrée à 940 nm est peu sensible à la lumière ambiante continue ; l'effet des flashs sera vérifié lors de l'essai de réception.

### Nombre et position des capteurs

**Tableau 3 :** FAD pour le choix du nombre et de la position des capteurs adversaire

| Solution technique proposée | Conf. (1/0) | Couverture angulaire | Position de la cible | Broches et charge I²C | Coût | Taille | Total (produit) | Remarque |
|---|---|---|---|---|---|---|---|---|
| **1 ToF fixe à l'avant** | 1 | 25°<br>**1** | devant / absent<br>**1** | 1 XSHUT<br>**5** | 9,90 €<br>**5** | minimal<br>**5** | **125** | aveugle sur les côtés |
| **3 ToF fixes (−30°, 0°, +30°)** | 1 | ≈ 85°<br>**3** | 3 secteurs<br>**3** | 3 XSHUT<br>**4** | 29,70 €<br>**3** | 3 supports<br>**4** | **432** | |
| **5 ToF fixes en éventail** | 1 | ≈ 150°<br>**4** | 5 secteurs<br>**4** | 5 XSHUT<br>**2** | 49,50 €<br>**1** | 5 supports<br>**2** | **64** | budget dépassé |
| **3 ToF sur tourelle orientée par le FT90M** | 1 | 69° orientable sur ±140°<br>**5** | secteur + angle servo<br>**5** | 3 XSHUT + MLI + Q7<br>**3** | 37,03 €<br>**2** | tourelle imprimée<br>**3** | **450** | angle mort arrière 80° |

**Synthèse de la sélection.** La **tourelle de trois capteurs** l'emporte de peu sur trois capteurs fixes (450 contre 432) : pour un surcoût de 7,33 € (servomoteur), elle couvre tout le dohyo sauf un secteur arrière de 80°, que le châssis expose en pivotant, et donne la direction de la cible avec la résolution de l'angle du servo au lieu de trois secteurs fixes. Cinq capteurs fixes coûteraient 49,50 € et cinq lignes XSHUT pour une couverture moindre.

**Fonctionnalité de la solution retenue.** Les trois SEN0378 sont montés sur la tourelle à 89 mm de hauteur, inclinés de 10° vers le sol et orientés à −22°, 0° et +22° ; les trois cônes de 25° forment un éventail de 69°. La tourelle tourne de ±140° autour de l'axe avant (consigne servo 0 à 280°, 140° = avant). Chaque capteur mesure en mode longue distance (budget de 20 ms) et rend jusqu'à quatre cibles, avec leur distance, leur écart-type et leur taux de signal.

**Information en entrée :** lumière infrarouge à 940 nm réfléchie par l'adversaire situé entre 25 et 850 mm (au-delà, les retours sont hors du dohyo) ; commande XSHUT 0/3,3 V.

**Information en sortie :** trames I²C à 400 kHz : distance en mm (entier), écart-type et taux de signal (format 16.16) pour chacune des quatre cibles, horodatées par le programme.

## 2.1.2 Choix des capteurs de sol

**Référence de pré-conception :** CPR_SOL

**Exigences client vérifiées par préconception :** EXIG_DEPLACEMENT, EXIG_LUMINOSITE, EXIG_FOURNISSEUR

Le robot doit détecter la bordure blanche pour ne pas sortir de lui-même du dohyo (EXIG_DEPLACEMENT), y compris sous un fort éclairage (EXIG_LUMINOSITE).

### Référence du capteur

**Tableau 4 :** FAD pour le choix de la référence des capteurs de sol

| Solution technique proposée | Conf. (1/0) | Liste du CDP | Filtre de lumière du jour | Distance de détection | Empreinte KiCad standard | Essai sur prototype | Total (produit) | Remarque |
|---|---|---|---|---|---|---|---|---|
| **Vishay CNY70** | 1 | liste CDP<br>**5** | oui<br>**5** | 0 à 5 mm<br>**3** | Vishay_CNY 70<br>**5** | SIM03 réalisé<br>**5** | **1 875** | |
| **Vishay TCRT5000** | 1 | hors liste<br>**2** | oui<br>**5** | 0,2 à 15 mm<br>**5** | Vishay_TCR T5000<br>**5** | non essayé<br>**1** | **250** | |

**Synthèse de la sélection.** Le **CNY70** est retenu (1 875 contre 250) : il figure dans la liste préférentielle du [CDP], possède un filtre de lumière du jour [S9] et a été validé sur prototype (SIM03). Sa faible distance de détection impose de placer le capteur à quelques millimètres du sol, ce que permettent les supports imprimés des cartes capteurs.

**Nombre et disposition des capteurs**

**Tableau 5 :** FAD pour le choix de la configuration des capteurs de sol (notes de 1 à 5)

| Configuration | Conf. (1/0) | Couver-ture | Recul / poussée | Entrées ADC | Conso. | Encom-brement | Coût | Total |
|---|---|---|---|---|---|---|---|---|
| **1 CNY70 avant centre** | 0 | 1 | 1 | 5 | 5 | 5 | 5 | 0 |
| **2 CNY70 avant (G/D)** | 1 | 3 | 1 | 4 | 4 | 4 | 4 | 768 |
| **4 CNY70 aux quatre coins** | 1 | 5 | 5 | 3 | 3 | 3 | 3 | 2 025 |
| **6 CNY70 (coins + flancs)** | 1 | 5 | 5 | 1 | 2 | 2 | 2 | 200 |

**Synthèse de la sélection.** Un capteur unique ne voit pas la bordure lorsque le robot recule ou est poussé : il n'est pas conforme. La configuration à **quatre capteurs aux coins** (avant gauche, avant droit, arrière gauche, arrière droit) obtient le meilleur total (2 025) : elle surveille les quatre extrémités quel que soit le sens de déplacement et indique de quel côté se trouve la bordure. Six capteurs n'apportent rien de plus et consomment deux entrées ADC1 supplémentaires.

**Fonctionnalité de la solution retenue.** Chaque CNY70 associe une LED infrarouge (950 nm) qui éclaire le sol et un phototransistor monté en émetteur commun : la bande blanche réfléchit la lumière et fait chuter la tension de collecteur. Les capteurs sont regroupés par deux sur deux petites cartes (avant, arrière) reliées par les connecteurs Grove J8 et J7.

**Information en entrée :** réflectance du sol à 0,3 à 5 mm du capteur : noir du dohyo (faible) ou bande blanche de 25 mm (forte).

**Information en sortie :** quatre tensions analogiques de 0,3 V (blanc, transistor saturé) à environ 3,3 V (noir), lues par l'ADC1 sur 12 bits en millivolts étalonnés.

### 2.1.3 Récepteur de l'ordre de départ

**Référence de pré-conception :** CPR_DEPART

**Exigences client vérifiées par préconception :** EXIG_DEPART, EXIG_LUMINOSITE

La télécommande de référence IRC01 est imposée par le [CDC] ; le récepteur et son câblage reprennent le dossier [IR] de M. Blanchard : un récepteur intégré **TSOP34438** (photodiode, amplificateur à gain automatique, filtre passe-bande à 38 kHz et démodulateur) [S10]. Cette solution étant imposée, elle ne fait pas l'objet d'une FAD.

**Fonctionnalité de la solution retenue.** Le TSOP34438 restitue l'enveloppe de la trame IR modulée à 38 kHz ; le programme la décode avec la bibliothèque IRremote [S27]. Le filtre passe-bande et le gain automatique rejettent la lumière continue, les éclairages modulés à basse fréquence et les impulsions courtes des télémètres adverses, qui ne forment pas une trame décodable.

**Information en entrée :** rayonnement infrarouge 940 nm modulé à 38 kHz par l'IRC01.

**Information en sortie :** signal logique actif à l'état bas, aux niveaux 0/3,3 V puisque le récepteur est alimenté par le +3V3, sur GPIO21.

### 2.1.4 Choix du pont en H

**Référence de pré-conception :** CPR_DEPLACEMENT

**Exigences client vérifiées par préconception :** EXIG_DEPLACEMENT, EXIG_CARTE

Les deux motoréducteurs GM9 (moteurs RM3) sont imposés. Le pont en H doit les commander dans les deux sens à vitesse variable sous 6,6 à 8,4 V, avec un courant de blocage estimé à 0,94 A par moteur (§ 3.3.1).

**Tableau 6 :** FAD pour le choix du pont en H

| Solution technique proposée | Conf. (1/0) | Tensions | Courant de sortie | Pilotage | Prix | Taille | Dispo. | Total (produit) | Remarque |
|---|---|---|---|---|---|---|---|---|---|
| **Pololu 2135, DRV8835 2 × 1,2 A** | 1 | logique 2 à 7 V, moteurs 2 à 11 V **5** | 1,2 A (1,5 A en pointe) **5** | 2 GPIO + 2 MLI (MODE = 1) **5** | 7,70 € **5** | 19 × 11 × 3 mm **5** | oui **5** | **15 625** | essai pratique fait (SIM02) |
| **Pololu DRV8833 2 × 1,2 A** | 1 | 2,7 à 10,8 V (logique 3 ou 5 V) **3** | 1,2 A (2 A en pointe) **5** | 4 MLI **4** | 11,40 € **3** | 21 × 13 × 3 mm **4** | oui **5** | **3 600** | non essayé |

**Synthèse de la sélection.** Le **Pololu 2135 (DRV8835)** obtient le meilleur total (15 625 contre 3 600) : son mode PHASE/ENABLE (MODE au niveau haut) ne demande qu'une sortie de sens et une sortie MLI par moteur ; sa logique (VCC, 2 à 7 V) est alimentée en 3,3 V et sa puissance (VIN, 0 à 11 V) directement par la batterie ; il est moins cher (7,70 € contre 11,40 €) et plus compact (19 × 11 × 3 mm) ; il a été validé par prototypage rapide avec les GM9 (SIM02).

**Fonctionnalité de la solution retenue.** Pour chaque moteur, PHASE fixe le sens et ENABLE, modulé à 20 kHz, la tension moyenne ; ENABLE = 0 met les deux sorties à la masse et freine le moteur.

**Information en entrée :** 2 × (PHASE logique 0/3,3 V, ENABLE MLI 20 kHz 0/3,3 V) ; puissance VBAT_SW 6,6 à 8,4 V.

**Information en sortie :** tension moyenne ±α·VBAT aux bornes de chaque moteur (α rapport cyclique), 1,2 A continus et 1,5 A en pointe par voie [S5].

### 2.1.5 Choix des régulateurs de tension

**Référence de pré-conception :** CPR_REGULATION

**Exigences client vérifiées par préconception :** EXIG_SECUR_BATT, EXIG_AUTONOMIE, EXIG_CARTE, EXIG_PCB_CMS

Les moteurs sont alimentés directement par la batterie commutée. Deux tensions régulées restent à produire, avec des régulateurs CMS (EXIG_PCB_CMS) : 5 V pour le servomoteur (0,8 A en blocage [S14]) et la branche 3,3 V, soit environ 1 A au plus ; 3,3 V pour l'ESP32-S3, les capteurs et la logique du pont, soit 188 mA prévus (tableau 9) et 500 mA recommandés par Espressif pour absorber les pointes du module [S7].

### Régulateur 5 V

**Tableau 7 :** FAD pour le choix du régulateur 5 V

| Solution technique proposée | Conf. (1/0) | Courant disponible | Rendement à 1 A sous 7,4 V | Composants externes | Achat | Total (produit) | Remarque |
|---|---|---|---|---|---|---|---|
| **LP2985AIM5-5.0 (linéaire)** | 0 | 150 mA **1** | 68 % **2** | 2 condensateurs **5** | liste CDP **5** | **0** | courant insuffisant |
| **MC33269DT-5.0 (linéaire, DPAK)** | 1 | 800 mA **3** | 68 % : 2,4 W perdus **2** | 2 condensateurs **5** | liste CDP **5** | **150** | échauffement |
| **MAX1626 (contrôleur abaisseur)** | 1 | 2 A (selon MOSFET) **5** | ≈ 88 % **5** | MOSFET, diode, shunt, L **1** | Farnell **4** | **100** | ancienne version |
| **AP63200WU-7 (abaisseur synchrone)** | 1 | 2 A **5** | ≈ 88 % **5** | L, 2 R, 6 C **3** | RS **4** | **300** | TSOT-23-6 |

**Synthèse de la sélection.** Le **convertisseur abaisseur AP63200WU-7** est retenu (300) : il fournit 2 A à 500 kHz entre 3,8 et 32 V d'entrée et intègre ses deux transistors et sa limitation de courant [S1]. Un régulateur linéaire perdrait (7,4 − 5) × 1 A = 2,4 W au courant maximal, au détriment de l'autonomie et de la température ; le contrôleur MAX1626 de la version précédente demandait cinq composants de puissance externes.

### Régulateur 3,3 V

**Tableau 8 :** FAD pour le choix du régulateur 3,3 V

| Solution technique proposée | Conf. (1/0) | Courant disponible | Tension de déchet | Boîtier et dissipation | Achat | Total (produit) | Remarque |
|---|---|---|---|---|---|---|---|
| **LP2985AIM5-3.3 (SOT-23-5)** | 0 | 150 mA **1** | 0,28 V **5** | SOT-23-5 **2** | liste CDP **5** | **0** | courant insuffisant |
| **MC33269DT-3.3 (DPAK)** | 1 | 800 mA **5** | 1,0 V **3** | DPAK **3** | liste CDP **5** | **225** | |
| **AP7361C-33E (SOT-223)** | 1 | 1 A **5** | 0,36 V **5** | SOT-223 **4** | DigiKey seul **2** | **200** | hors fournisseurs du CDC |
| **LD1117S33CTR (SOT-223)** | 1 | 800 mA **5** | 1,1 V **3** | SOT-223 **4** | RS 686-9388 **5** | **300** | 0,18 € HT |

**Synthèse de la sélection.** Le **LD1117S33CTR** est retenu (300), choix confirmé à la revue du 13/10/2026 : il fournit 800 mA, sa sortie garantie de 3,16 à 3,44 V est dans la plage de l'ESP32-S3 (3,0 à 3,6 V) [S2, S7], il est disponible chez RS et son boîtier SOT-223 dissipe mieux qu'un SOT-23. Sa tension de déchet (1,1 V) reste compatible avec le +5 V minimal (4,86 − 3,44 = 1,42 V) ; l'AP7361C, meilleur sur ce point, n'est vendu par aucun fournisseur autorisé (EXIG_FOURNISSEUR). Un régulateur linéaire est préféré ici car sa sortie est peu bruitée pour les mesures analogiques.

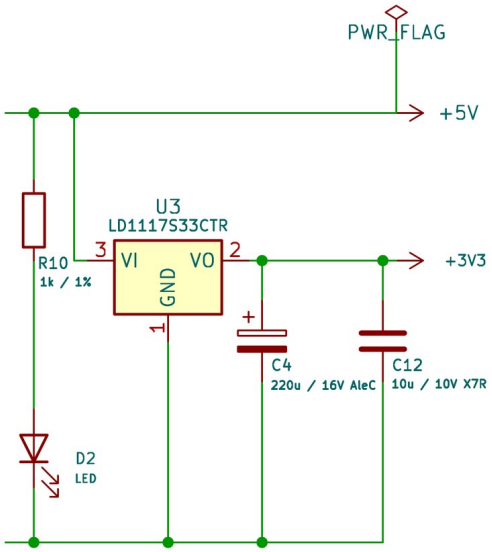



**Figure 3 :** Régulateur 3,3 V U3 LD1117S33CTR, condensateurs C4 et C12, LED d'alimentation D2

**Fonctionnalité de la solution retenue.** U2 abaisse VBAT_SW à 4,99 V par découpage à 500 kHz ; U3 abaisse ce +5 V à 3,3 V par régulation linéaire. Les deux régulateurs sont des composants CMS.

**Information en entrée :** VBAT_SW 6,6 à 8,4 V (U2) ; +5 V (U3).

**Information en sortie :** +5 V : 4,86 à 5,12 V, 2 A au plus ; +3V3 : 3,16 à 3,44 V, 800 mA au plus.

### 2.1.6 Batterie, protection et interrupteur

**Référence de pré-conception :** CPR_BATTERIE

**Exigences client vérifiées par préconception :** EXIG_AUTONOMIE, EXIG_SECUR_BATT, EXIG_PROT_INV_POL, EXIG_INTER

**Batterie imposée et bilan de consommation**

Le [CDC] impose une **LiPo 2S de 500 mAh**. Sa tension varie de 8,4 V à pleine charge à 6,6 V au seuil de protection (3,3 V par élément). L'autonomie est jugée par un déplacement continu de 65 minutes (EXIG_AUTONOMIE) : le courant moyen ne doit donc pas dépasser

$$I_{moy\ max} = Q / t = 0{,}500\ Ah / (65/60)\ h = \mathbf{0{,}462\ A}$$

Ce majorant ne tient pas compte de la capacité inutilisable sous 6,6 V. Le bilan prévisionnel (tableau 9) montre que le budget est tenu si le rapport cyclique moyen des moteurs reste limité.

**Tableau 9 :** Bilan de consommation prévisionnel

| Consommateur | Hypothèse | Courant sur le rail | Courant batterie |
|---|---|---|---|
| ESP32-S3 (radio coupée) | hypothèse haute | 80 mA sur 3,3 V | |
| 3 × SEN0378 (VL53L3CX) | ≈ 20 mA chacun en mesure | 60 mA sur 3,3 V | |
| 4 × CNY70 | 9,3 mA (LED) + 0,33 mA (tirage) | 39 mA sur 3,3 V | |
| MPU6050, TSOP34438, D1, logique DRV8835 | fiches techniques | 9 mA sur 3,3 V | |
| Total rail 3,3 V (via U3 sur le 5 V) | | 188 mA sur 5 V | |
| D2 et servomoteur FT90M en balayage | 3 mA + moyenne estimée 150 mA | 153 mA sur 5 V | |
| Total rail 5 V (U2, η ≈ 85 %) | P = 5 V × 0,34 A / 0,85 | 2,0 W | 0,27 A à 7,4 V |
| 2 moteurs GM9 à 100 % de MLI | mesure du prototype (SIM02) | | 0,30 A |
| **Total à 100 % de MLI** | | | **0,57 A** |
| **Total à 47 % de MLI moyen** | $I_{bat\ moteurs} \approx \alpha \times I_{moteurs}$ | | **0,41 A** |

Avec un pont en H, le courant moyen prélevé sur la batterie par les moteurs vaut environ $\alpha \times I_{moteurs}$, où α est le rapport cyclique. La décision limite donc le rapport cyclique moyen en déplacement continu à 47 % : le total est alors de 0,41 A, soit 11 % de marge sur 0,462 A.

### Protection contre l'inversion de polarité

**Tableau 10 :** FAD pour le choix de la protection contre l'inversion de polarité

| Solution technique proposée | Conf. (1/0) | Chute de tension à 2 A | Puissance perdue à 2 A | Courant en inversion (< 1 mA) | Nombre de composants | Total (produit) | Remarque |
|---|---|---|---|---|---|---|---|
| **Diode et fusible en parallèle (« crowbar »)** | 0 | 0 V<br>**5** | 0 W<br>**5** | court-circuit<br>**1** | 2<br>**4** | **0** | courant inverse > 1 mA |
| **Diode Schottky série (MBRS540)** | 1 | ≈ 0,5 V<br>**2** | ≈ 1 W<br>**2** | fuite < 1 mA<br>**4** | 1<br>**5** | **80** | |
| **MOSFET canal P Si2369DS, grille à la masse** | 1 | 0,08 V<br>**5** | 0,16 W<br>**5** | 0 (canal bloqué)<br>**5** | 1<br>**5** | **625** | 40 mΩ max. |

**Synthèse de la sélection.** Le **MOSFET canal P Si2369DS** (Q2) est retenu (625) : placé juste après le connecteur J101 comme l'exige le [CDC], il ne laisse passer aucun courant en polarité inversée ($V_{GS} \approx 0$) et ne perd que 0,16 W à 2 A, contre environ 1 W et 0,5 V de tension perdue pour une diode Schottky [S3].

**Interrupteur marche/arrêt**

**Tableau 11 :** FAD pour le choix de l'interrupteur électronique

| Solution technique proposée | Conf. (1/0) | Courant dans l'interrupteur | Consommation de commande | Masse commune | Taille | Total (produit) | Remarque |
|---|---|---|---|---|---|---|---|
| **Interrupteur en série (courant total)** | 0 | jusqu'à 3 A<br>**1** | 0<br>**5** | oui<br>**5** | aucun<br>**5** | **0** | interdit par EXIG_INTER |
| **Relais commandé par l'interrupteur** | 1 | bobine ≈ 30 mA<br>**4** | ≈ 30 mA permanents<br>**2** | oui<br>**5** | relais<br>**2** | **80** | |
| **MOSFET canal N côté masse** | 1 | < 0,1 mA<br>**5** | ≈ 0<br>**5** | masse coupée<br>**2** | SOT-23<br>**5** | **250** | niveaux faussés |
| **MOSFET canal P côté batterie (Si2369DS)** | 1 | 84 µA<br>**5** | 84 µA<br>**5** | oui<br>**5** | SOT-23<br>**5** | **625** | |

**Synthèse de la sélection.** Le **MOSFET canal P Si2369DS** (Q1), dont la grille est tirée à la masse par l'interrupteur à bascule, est retenu (625) : l'interrupteur ne conduit que 84 µA (EXIG_INTER) et la masse reste commune à toute la carte, contrairement à une coupure côté masse ; un relais consommerait en permanence le courant de sa bobine.

**Fonctionnalité de la solution retenue.** Q2 protège la carte contre l'inversion de polarité, Q1 commute toute l'alimentation sur ordre de l'interrupteur ; leurs sources sont communes. La tension VBAT_SW qui en sort alimente le pont en H et le convertisseur U2 ; elle est aussi mesurée par le pont R104/R105 (EXIG_SECUR_BATT, § 3.4.1).

**Information en entrée :** tension batterie 6,6 à 8,4 V (ou −8,4 V en cas d'inversion) ; contact de l'interrupteur.

**Information en sortie :** VBAT_SW = VBAT − 2 × $R_{DS(on)}$ × I, soit 0,16 V de moins à 2 A ; 0 V et aucun courant en inversion ou interrupteur ouvert.

## 2.1.7 Choix de la centrale inertielle

**Référence de pré-conception :** CPR_IMU

**Exigences client vérifiées par préconception :** EXIG_COMPORTEMENT, EXIG_DEPLACEMENT

Sans codeur sur les roues, le robot ne connaît ni sa rotation réelle ni les poussées qu'il subit. Une centrale inertielle apporte ces mesures à la décision.

**Tableau 12 :** FAD pour le choix de la centrale inertielle

| Solution technique proposée | Conf. (1/0) | Mesures | Intégration sur la carte | Logique 3,3 V | Coût | Total (produit) | Remarque |
|---|---|---|---|---|---|---|---|
| **Pas de centrale** | 1 | aucune<br>**1** | —<br>**5** | —<br>**5** | 0 €<br>**5** | **125** | poussées non détectées |
| **ST LSM6DSO en boîtier nu** | 1 | 6 axes, ±2000 °/s<br>**5** | LGA-14 pas 0,5 mm : non soudable au fer<br>**1** | oui<br>**5** | 4,80 €<br>**5** | **125** | version précédente |
| **MPU6050 sur module Joy-IT SEN-MPU6050** | 1 | 6 axes, ±500 °/s, ±4 g<br>**4** | module sur connecteur J1<br>**5** | oui<br>**5** | 4,12 €<br>**5** | **500** | |

**Synthèse de la sélection.** Le **module MPU6050** est retenu (500) : il se branche sur le connecteur J1, alors que le LSM6DSO en boîtier LGA-14 ne peut pas être soudé à la main au

département [S12, S24]. Ses plages ±500 °/s et ±4 g (65,5 LSB/(°/s) et 8192 LSB/g) couvrent les rotations et chocs d'un combat.

**Fonctionnalité de la solution retenue.** Le gyromètre mesure la vitesse de lacet, qui corrige le cap entre deux mesures ToF ; l'écart entre l'accélération mesurée et celle prévue par la commande des moteurs révèle un contact ou une poussée. Le module partage le bus I²C (adresse 0x68, distincte de celles des télémètres) et signale ses mesures prêtes sur GPIO40.

**Information en entrée :** rotation et accélération du châssis.

**Information en sortie :** six mots de 16 bits (accélérations X, Y, Z et vitesses angulaires X, Y, Z) lus en rafale sur l'I²C à 400 kHz ; interruption 3,3 V sur GPIO40.

### 2.1.8 Choix du microcontrôleur

**Référence de pré-conception :** CPR_MCU

**Exigences client vérifiées par préconception :** EXIG_CARTE, EXIG_CODE_LANGAGE

Le microcontrôleur doit être implanté sur la carte (EXIG_CARTE), lire quatre entrées analogiques, piloter trois sorties MLI et un bus I²C à 3,3 V, et exécuter le suivi de cible et l'inférence du réseau de neurones (17 922 poids, 72 ko).

**Tableau 13 :** FAD pour le choix du microcontrôleur

| Solution technique proposée | Conf. (1/0) | Logique 3,3 V (ToF, IMU) | Entrées ADC | Sorties MLI | Mémoire (réseau 72 ko) | Fréquence de calcul | Expérience de l'équipe | Total (produit) | Remarque |
|---|---|---|---|---|---|---|---|---|---|
| **ATMEGA328P** | 1 | non (5 V)<br>**2** | 6 × 10 bits<br>**3** | 6<br>**3** | 2 ko de RAM<br>**1** | 16 MHz<br>**2** | cours GEII<br>**5** | **180** | |
| **ESP32-WROOM-32E** | 1 | oui<br>**5** | ADC1 8 voies, non linéaire<br>**3** | 16 (LEDC)<br>**5** | 520 ko<br>**5** | 240 MHz<br>**5** | prototype tourelle<br>**4** | **7 500** | |
| **ESP32-S3-WROOM-1-N4** | 1 | oui<br>**5** | ADC1 10 voies, étalonné<br>**5** | 8 (LEDC)<br>**5** | 512 ko + 4 Mo flash<br>**5** | 240 MHz, 2 cœurs<br>**5** | nouveau<br>**3** | **9 375** | |

**Synthèse de la sélection.** L'**ESP32-S3-WROOM-1-N4** est retenu (9 375 contre 7 500) : logique 3,3 V commune avec les capteurs, ADC1 étalonné en usine pour les quatre CNY70, assez de mémoire et de calcul pour les trois pilotes VL53L3CX et l'inférence du réseau [S7, S8]. La version N4 (flash Quad SPI, sans PSRAM) laisse GPIO35 à GPIO37 libres. Le module est soudé sur la carte, conformément à EXIG_CARTE.

**Fonctionnalité de la solution retenue.** Le module acquiert les capteurs, exécute le suivi de cible de la tourelle et la politique de décision, et commande le pont en H, le servomoteur, Q7 et la LED D1 (§ 2.2).

**Information en entrée :** bus I²C, 4 tensions ADC1, VBAT_ADC (ADC2), IR_START, IMU_INT, UART0.

**Information en sortie :** signaux logiques et MLI 3,3 V décrits au tableau 1.

## 2.2 Architecture informatique

**Référence de pré-conception :** CPR_ARCHI_INFO

**Exigences client vérifiées par préconception :** EXIG_CODE_LANGAGE, EXIG_CODE_FONCTION, EXIG_CODE_IMBRICATION, EXIG_CODE_VARIABLE, EXIG_CODE_TEST, EXIG_DEPART, EXIG_DEPLACEMENT, EXIG_ADVERSAIRE, EXIG_SECUR_BATT, EXIG_COMPORTEMENT

Le logiciel embarqué est écrit en C++ avec le cœur Arduino-ESP32 (EXIG_CODE_LANGAGE). Il distingue ce qui s'exécute sur le robot de ce qui sert seulement à la conception sur PC (tableau 14).

**Tableau 14 :** Périmètre du logiciel embarqué et des outils de conception

| Partie | Rôle | Emplacement dans le dépôt |
|---|---|---|
| **Embarqué** : couche d'entrées/sorties | lecture des capteurs, commande des actionneurs, essais unitaires de chaque unité matérielle | firmware/robot_sumo_s3_io |
| **Embarqué** : suivi de cible | association des retours des trois ToF, apprentissage du dohyo vide, secteur et distance de la cible, consigne de la tourelle | Tracker, Observation (repris de outputs/esp32_turret_tracker) |
| **Embarqué** : décision | réseau de neurones (poids figés du champion de l'entraînement, itération 1 795) et supervision : départ, batterie, défaut | ml/runtime/actor_runtime ; firmware/robot_sumo_s3_io |
| Bibliothèques tierces | pilote ST VL53LX, IRremote, ESP32Servo | src/VL53L3CX, bibliothèques Arduino |
| PC : simulation et apprentissage | simulateurs MATLAB et C++, contrôleur de référence (enseignant de l'apprentissage), entraînement PyTorch | Matlab simulations, native_sim, firmware/shared, ml |
| PC et banc : prototype de tourelle | ESP32-WROOM, télémétrie Wi-Fi vers l'application SumoScope, utilisé pour le dérisquage SIM06 | outputs/esp32_turret_tracker |

Les règles de codage du [CDC] (EXIG_CODE_FONCTION, EXIG_CODE_IMBRICATION, EXIG_CODE_VARIABLE) s'appliquent au code embarqué écrit par l'équipe ; les bibliothèques tierces en sont exclues par le [CDC].

**Cycle de fonctionnement**

- **Initialisation** : ACT_ENABLE = 0 et moteurs freinés avant toute autre configuration ; puis mesure batterie, CNY70, récepteur IR, télémètres (adressage par XSHUT) et centrale inertielle.
- **Attente du départ** : rapports cycliques nuls tant qu'aucune trame IRC01 valide n'est reçue (EXIG_DEPART).
- **Boucle de commande** (pas de 5 ms) : surveillance de la batterie en premier ; lecture des quatre CNY70 ; lecture d'un télémètre par passage et de l'IMU ; mise à jour du suivi et de la consigne de tourelle ; décision des roues toutes les 100 ms, et immédiatement dès qu'un CNY70 voit le blanc ; application des rapports cycliques au pont en H.
- **Défaut** : batterie sous le seuil (6,7 V mesurés), erreur d'un capteur ou entrée non finie : moteurs freinés, servo coupé par Q7, LED D1 clignotante ; le défaut est mémorisé.

### 2.2.1 Fonctions d'acquisition

**Référence de pré-conception :** CPR_INFO_ACQUISITION

**Exigences client vérifiées par préconception :** EXIG_ADVERSAIRE, EXIG_LUMINOSITE, EXIG_DEPART, EXIG_SECUR_BATT

**Tableau 15 :** Fonctions d'acquisition

| Fonction | Entrée matérielle | Sortie logicielle |
|---|---|---|
| initialiserTelemetres() / lireTelemetre() | I²C GPIO41/42, XSHUT GPIO37/38/39 | trame de 1 à 4 cibles par télémètre : distance (mm), écart-type, signal, horodatage |
| Tracker::ingest() | trames des trois télémètres + angle du servo | secteur (gauche, centre, droite), distance, âge et identifiant de la cible |
| initialiserImu() / lireImu() | I²C, interruption GPIO40 | vitesse de lacet (°/s) et accélérations avant/latérale (g) |
| lireCapteursSol() | ADC1 GPIO4, 5, 6, 7 | 4 tensions en mV ; 4 indicateurs de bordure |
| ordreDepartRecu() | GPIO21 (TSOP34438) | vrai à la première trame IRC01 valide |
| surveillerBatterie() | ADC2 GPIO20 (R104/R105) | tension batterie en mV ; défaut mémorisé |

Les mesures sont horodatées. La décision ne reçoit jamais la position vraie de l'adversaire : seulement ce que mesurent les capteurs, exactement comme dans les simulateurs, ce qui permet d'embarquer sans modification le réseau entraîné en simulation.

### 2.2.2 Fonctions d'action

**Référence de pré-conception :** CPR_INFO_ACTION

**Exigences client vérifiées par préconception :** EXIG_DEPLACEMENT, EXIG_CARTE, EXIG_SECUR_BATT

**Tableau 16 :** Fonctions d'action

| Fonction | Sortie matérielle | Rôle |
|---|---|---|
| phaseEnable() / commanderMoteur() | PHASE GPIO17/8, ENABLE (MLI 20 kHz) GPIO18/19 | sens et vitesse de chaque moteur ; ENABLE = 0 : freinage |
| orienterTourelle() | GPIO12 (MLI 50 Hz) | orientation de la tourelle, 1000 à 2000 µs |
| autoriserActionneurs() | GPIO11 (ACT_ENABLE → Q7) | retour de masse du servomoteur |
| signalerDefautBatterie() | GPIO15 (LED D1) | clignotement à 2 Hz en cas de batterie faible |

### 2.2.3 Fonctions d'énergie

**Référence de pré-conception :** CPR_GESTION_ENERGIE

**Exigences client vérifiées par préconception :** EXIG_SECUR_BATT, EXIG_INTER, EXIG_PROT_INV_POL

La surveillance de la batterie est la première action de chaque cycle. Le pont R104/R105 (100 kΩ / 33 kΩ, 1 %) ramène VBAT_SW sur GPIO20 : 6,6 V deviennent 1,638 V. Dès que la tension mesurée passe sous 6,7 V (6,6 V plus la marge de mesure après étalonnage), le programme met les

MLI des moteurs à zéro (coupure de la partie puissance), arrête le signal du servo, met ACT_ENABLE à 0 (Q7 coupe la masse du servo) et fait clignoter D1. Le défaut est mémorisé : il ne disparaît pas quand la tension remonte après suppression de la charge. La protection de polarité et l'interrupteur marche/arrêt sont purement matériels (Q2, Q1).

### 2.2.4 Fonctions de traitement

**Référence de pré-conception :** CPR_INFO_TRA

**Exigences client vérifiées par préconception :** EXIG_COMPORTEMENT, EXIG_DEPLACEMENT, EXIG_ADVERSAIRE

La décision transforme, à chaque pas, les mesures des capteurs en deux rapports cycliques signés (roue gauche, roue droite). Deux solutions ont été développées et comparées sur les mêmes combats simulés : un contrôleur à états écrit à la main, puis un réseau de neurones entraîné par apprentissage par renforcement, qui est la solution embarquée.

**Contrôleur de référence : automate à priorités**

Le contrôleur ChassisPolicy (firmware/shared) applique d'abord un arbitrage de priorité, puis une machine d'états de combat (figure 4). Il a été mis au point sur des dizaines de milliers de combats simulés (SIM05). Il sert d'enseignant pour l'apprentissage et de référence de comparaison ; il n'est pas embarqué dans la version finale.

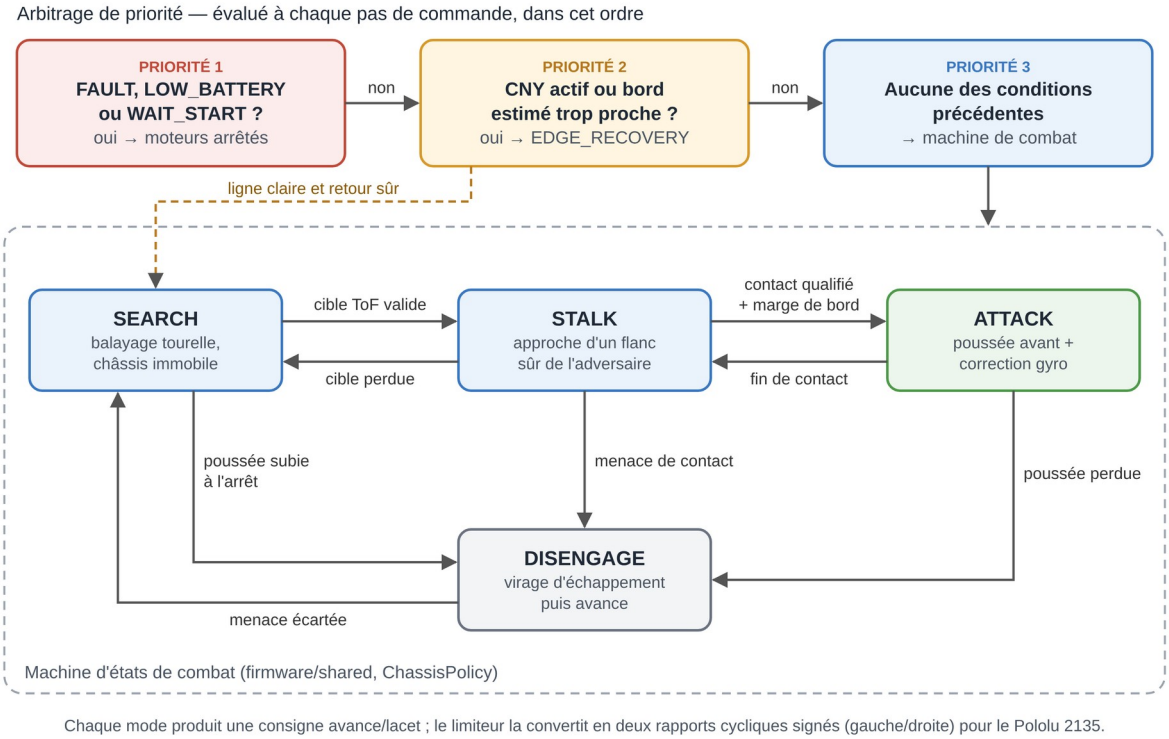



**Figure 4 :** Contrôleur de référence : arbitrage de priorité et machine d'états de combat

- **Priorité 1** : en défaut, batterie faible ou avant le départ, les moteurs sont arrêtés.
- **Priorité 2** : dès qu'un CNY70 voit le blanc, ou si le bord estimé est trop proche, EDGE_RECOVERY mémorise la direction extérieure, fait tourner le robot puis le ramène vers l'intérieur ; ce mode préempte tout combat.

- **SEARCH** : la tourelle balaie ; le châssis ne translate pas et pivote de 80° après un balayage complet sans cible.
- **STALK** : la cible est suivie ; deux points d'interception perpendiculaires sont comparés selon leur marge au bord, pour aborder l'adversaire par le flanc.
- **ATTACK** : poussée vers l'avant avec correction de lacet par le gyromètre, sur contact qualifié et marge de bord suffisante.
- **DISENGAGE** : virage d'échappement face à une approche menaçante ou à une poussée défavorable détectée par l'IMU.

**Solution embarquée : politique apprise par renforcement**

Écrire à la main une stratégie efficace contre tous les styles d'adversaires s'est révélé difficile : en simulation, le contrôleur de référence ne gagne que 42 % des combats en moyenne (SIM05). Nous entraînons donc un **réseau de neurones par apprentissage par renforcement**, qui décide des rapports cycliques des roues (figure 5). L'apprentissage se fait sur PC en simulation ; seul le réseau figé est exécuté par l'ESP32-S3.

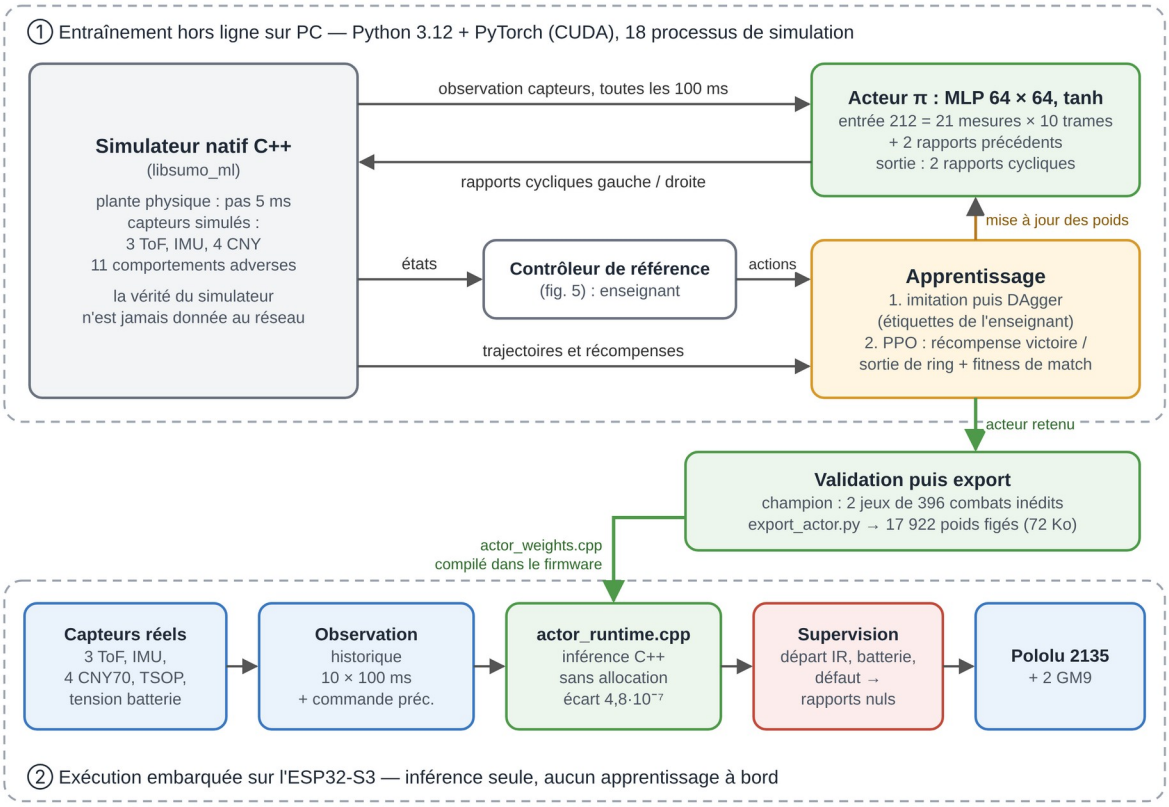



**Figure 5 :** Entraînement de la politique en simulation (PC) et exécution embarquée sur l'ESP32-S3

- **Environnement simulé** : simulateur natif C++ porté du modèle MATLAB : plante physique intégrée toutes les 5 ms, contact des carrosseries, sortie complète du dohyo, tourelle, cônes ToF en 3D, positions des CNY70, modèle couple-vitesse des GM9 et adhérence des pneus. Il exécute le vrai code C++ du suivi de cible. Onze comportements adverses (agressif, passif, flanqueur, etc.) sont tirés à parts égales.

- **Observation** : le paquet de 21 mesures que recevra le robot (départ, santé, cible ToF, IMU, angle de tourelle, quatre CNY70), sur les 10 dernières trames de 100 ms, plus la commande précédente, soit 212 entrées. L'historique compense ce qu'une mesure isolée ne montre pas (cible derrière la tourelle, mouvement de l'adversaire).
- **Réseau (acteur)** : perceptron à deux couches cachées de 64 neurones à tangente hyperbolique, 17 922 poids ; ses deux sorties tanh, entre −1 et 1, sont directement les rapports cycliques gauche et droit. Il décide toutes les 100 ms, et immédiatement si un CNY70 s'active.
- **Apprentissage** : imitation du contrôleur de référence avec correction sur les trajectoires propres du réseau (DAgger [S26]), qui évite de partir d'un comportement aléatoire, puis optimisation par PPO [S25], dont la mise à jour bornée garde l'apprentissage stable. La récompense est terminale (victoire, sortie du dohyo) avec un terme de performance du combat et des termes bornés d'approche et d'évitement du bord. Le réseau apprend aussi le dégagement de la bordure.
- **Embarquement** : export_actor.py écrit les poids en tableaux C++ constants ; actor_runtime.cpp calcule l'inférence sans allocation dynamique. Il reproduit PyTorch avec un écart maximal de $4{,}8 \times 10^{-7}$ sur 300 pas. La supervision (départ, batterie, défaut) impose des rapports cycliques nuls en aval du réseau.

Un réseau n'est retenu que s'il passe une porte d'acceptation sur deux jeux de combats jamais vus, l'un fixe et l'autre renouvelé à chaque évaluation : fitness supérieure, au moins autant de victoires, et pas plus de sorties du dohyo ni de sorties sans contact que le contrôleur de référence. Le réseau embarqué est le champion de l'entraînement du 12 au 13/11/2026 (itération 1 795), qui gagne 73,7 % des combats de validation contre 55,3 % au départ (SIM04).

## 2.3 Coûts prévisionnels du produit

**Référence de pré-conception :** CPR_COUT

**Exigences client vérifiées par préconception :** EXIG_COUT, EXIG_DELAI, EXIG_FOURNISSEUR

Le [CDC] limite le coût de l'ensemble des composants mécaniques et électroniques d'un prototype à 120 € HT (EXIG_COUT), avec une allocation bloc par bloc et une nomenclature chiffrée, et le temps de développement à 84 h (EXIG_DELAI). Ce paragraphe présente le coût matériel prévisionnel, puis la main-d'œuvre prévue au planning [PDP].

### Coût matériel prévisionnel

La nomenclature FAB02 du [DDF] (53 composants de la carte principale et 28 lignes d'ensembles externes) a été chiffrée ligne par ligne (tableau 17). Les prix sont, par ordre de préférence, des prix catalogue relevés le 02/10/2026 chez les fournisseurs autorisés, des prix publics relevés le 13/11/2026, ou des estimations pour les composants passifs dont la référence fabricant reste à fixer.

**Tableau 17 :** Nomenclature chiffrée prévisionnelle (EUR HT)

| Bloc | Désignation | Repères | Qté | PU HT (€) | Coût HT (€) | Base de prix |
|---|---|---|---|---|---|---|
| **Acquisition** | Module ToF DFRobot SEN0378 (VL53L3CX) | sur J3, J4, J5 | 3 | 9,90 | 29,70 | DDC V1 (équipe) |

| Bloc | Désignation | Repères | Qté | PU HT (€) | Coût HT (€) | Base de prix |
|---|---|---|---|---|---|---|
| | Module IMU Joy-IT SEN-MPU6050 | sur J1 | 1 | 4,12 | 4,12 | Prix public 13/11/2026 |
| | CNY70 (RS 708-5348) | U6 à U9 | 4 | 0,784 | 3,14 | Catalogue 02/10/2026 |
| | TSOP34438 (Farnell 2251342) | U5 | 1 | 1,11 | 1,11 | Catalogue 02/10/2026 |
| | Résistances 1206 des cartes capteurs | R16 à R23 | 8 | 0,10 | 0,80 | Estimation |
| | Connecteurs 1×5 et 1×3, Grove coudés | J1, J3–J6, J7, J8 | 7 | 0,40 | 2,80 | Estimation |
| | Câbles Grove (lot de 5) | J7, J8 | 1 | 2,00 | 2,00 | Estimation |
| **Traitement** | ESP32-S3-WROOM-1-N4 | U1 | 1 | 4,85 | 4,85 | Catalogue 02/10/2026 |
| | Condensateurs C1 (22 µF), C2, C3 | C1 à C3 | 1 | 1,90 | 1,90 | Estimation |
| | Résistances R2 à R5, boutons SW1/SW2, connecteur J13 | R2–R5, SW1, SW2, J13 | 1 | 1,85 | 1,85 | Estimation |
| **Action** | Pololu 2135 DRV8835 (GoTronic 32706) | U4 | 1 | 6,42 | 6,42 | Catalogue 02/10/2026 |
| | Motoréducteur GM9 (GoTronic 25343) | sur J10, J11 | 2 | 5,75 | 11,50 | Catalogue 02/10/2026 |
| | Servomoteur FT90M | sur J9 | 1 | 7,33 | 7,33 | Catalogue 02/10/2026 |
| | Coupure servo : Q7, R6, R11, C15, C16 | Q7, R6, R11, C15, C16 | 1 | 1,10 | 1,10 | Estimation |
| | Connecteurs J9, J10, J11 | J9 à J11 | 3 | 0,27 | 0,81 | Estimation |
| **Énergie** | Batterie LiPo 2S (CDP, Conrad 239029) | sur J101 | 1 | 9,49 | 9,49 | Catalogue 02/10/2026 |
| | Convertisseur AP63200WU-7 | U2 | 1 | 1,00 | 1,00 | Estimation |
| | Inductance XAL6060-103MEC (Farnell) | L1 | 1 | 3,41 | 3,41 | Catalogue 02/10/2026 |
| | Régulateur LD1117S33CTR (RS 686-9388) | U3 | 1 | 0,18 | 0,18 | Prix public 13/11/2026 |
| | MOSFET Si2369DS | Q1, Q2 | 2 | 0,40 | 0,80 | Estimation |
| | Condensateurs C4–C14 (céramiques 1206, électrolytiques) | C4–C12, C14 | 1 | 7,20 | 7,20 | Estimation |
| | Résistances, LED D1/D2, connecteurs J2/J101 | R1, R7–R10, R104, R105, D1, D2, J2, J101 | 1 | 1,70 | 1,70 | Estimation |
| | Interrupteur à bascule (fourni avec la base) | sur J2 | 1 | 1,00 | 1,00 | Estimation |
| **Mécanique et fabrication** | Circuits imprimés FR4 (carte principale, 2 cartes capteurs) | — | 1 | 4,00 | 4,00 | Estimation |
| | Tourelle et supports imprimés (PLA, ≤ 3 h) | — | 1 | 0,60 | 0,60 | Estimation |
| | Vis M3 ≤ 13 mm et entretoises | H1 à H4 | 4 | 0,20 | 0,80 | Estimation |
| | Base mécanique (châssis, roues GMPW) | — | 1 | 0,00 | 0,00 | fournie (CDP) |
| Total | | | | | **109,61** | |

**Tableau 18 :** Allocation du coût matériel par bloc

| Bloc | Coût prévisionnel HT | Part du total |
|---|---|---|
| Acquisition | 43,67 € | 40 % |
| Traitement | 8,60 € | 8 % |
| Action | 27,16 € | 25 % |
| Énergie | 24,78 € | 23 % |
| Mécanique et fabrication | 5,40 € | 5 % |
| **Total prévisionnel** | **109,61 €** | 100 % |
| Plafond EXIG_COUT | 120,00 € | |
| **Marge** | **10,39 €** | 8.7 % |

Le coût prévisionnel est de **109,61 € HT**, soit une marge de 10,39 € (8.7 %) sous le plafond de 120 € HT : le chiffrage est cohérent avec EXIG_COUT. Les modules achetés (télémètres, pont en H, motoréducteurs, servomoteur, batterie) représentent l'essentiel du coût ; les estimations ne portent que sur 28,36 € de petits composants, dont une erreur de 50 % ne ferait pas dépasser le plafond. Le budget initial du 02/10/2026, établi pour l'architecture précédente, était de 109,41 € HT ; l'écart de 0,20 € vient du remplacement des télémètres nus par des modules et de l'ajout de la centrale inertielle, compensés par la simplification de l'alimentation. Le coût réel sera relevé sur les factures dans le DDV.

### Coût de main-d'œuvre prévu au planning

**Tableau 19 :** Main-d'œuvre prévue par équipier au planning RMS_PDP v4.1

| Phase du planning | Heures encadrées | Heures en autonomie | Total (h) | Coût (35 € HT/h) |
|---|---|---|---|---|
| Conception (FAD, schémas, programmation, dérisquage, routage, DDC) | 32,2 | 18 | 50,2 | 1757 € |
| Fabrication (gravure, étamage, assemblage, programmation) | 8,1 | 4 | 12,1 | 424 € |
| Vérification (débogage, essais, DDV) | 7,75 | 4 | 11,75 | 411 € |
| Certification, maintenance et compétition | 7,75 | 2 | 9,75 | 341 € |
| **Total par équipier** | **55,8** | **28** | **83,8** | **2933 €** |

Le planning [PDP] affecte à chaque équipier 55,8 h de séances encadrées et 28 h de travail en autonomie, soit **83,8 h pour 84 h autorisées** (EXIG_DELAI). Les tâches de conception sont réparties par binôme et par bloc (acquisition, traitement et énergie, action), et le planning est mis à jour à chaque séance. Avec un taux horaire chargé de 35 € HT, retenu par l'équipe pour un technicien, la main-d'œuvre représente 2933 € HT par équipier. Le coût de développement d'un prototype est donc dominé par la main-d'œuvre ; seul le coût matériel est plafonné par le [CDC].

## 2.4 Réponses préliminaires aux exigences du CDC

Ce paragraphe associe à chacune des 36 exigences du [CDC] la solution retenue, sa justification et la preuve apportée au stade de la conception. « Conforme (conception) » signifie que la conception satisfait l'exigence et sa tolérance par le choix, le calcul, la simulation ou le prototypage ; l'essai de réception sur le robot assemblé est consigné dans le DDV.

## Mécanique et implantation

**EXIG_CHASSIS_DIMENSIONS** — Le PCB mesure 72,0 × 85,0 mm et les supports imprimés restent dans l'emprise de la base : l'ensemble tient dans 100 × 100 mm. Le DDV mesure le robot monté avec un gabarit de 100 × 100 mm. **Statut : Conforme (conception).**

**EXIG_MASSE** — La masse est maîtrisée par des modules légers (FT90M, modules ToF, carte double face) et la batterie imposée de 500 mAh ; la simulation des combats utilise 500 g comme hypothèse haute. Le DDV pèse le robot prêt au combat. **Statut : Conforme (conception).**

**EXIG_INTEGRITE_MECA** — Les platines fournies ne sont ni découpées ni percées : la carte et les capteurs sont fixés par des supports imprimés sur les ouvertures existantes. **Statut : Conforme (conception).**

**EXIG_INNOVATION_MECA** — Les supports de la tourelle et des cartes CNY70 sont conçus pour une impression courte ; la somme des durées du trancheur Bambu P1S (buse 0,4 mm, supports compris) est limitée à 180 min. **Statut : Conforme (conception).**

**EXIG_FIXATION_CARTE** — La carte comporte quatre trous M3 (H1 à H4) pour un minimum de trois ; les vis retenues mesurent au plus 13 mm. **Statut : Conforme (conception).**

## Alimentation et sûreté électrique

**EXIG_AUTONOMIE** — Le budget de 0,462 A pour 65 min de déplacement continu est tenu avec un rapport cyclique moteur moyen limité à 47 % (0,41 A, tableau 9). **Statut : Conforme (conception).**

**EXIG_SECUR_BATT** — Le pont R104/R105 donne 1,638 V à 6,6 V (1,613 à 1,662 V avec les tolérances, SIM01) ; le gain est étalonné. Dès 6,7 V mesurés, donc avant que la batterie n'atteigne 6,6 V, le programme coupe la partie puissance (MLI moteurs à zéro, servo coupé par Q7) et fait clignoter D1, sans réarmement automatique (§ 3.4.1). **Statut : Conforme (conception).**

**EXIG_PROT_INV_POL** — Q2 (Si2369DS) est le premier composant après J101 ; en polarité inversée, $V_{GS} \approx 0$ et sa diode interne est bloquée : le courant est celui de fuite du MOSFET, 1 µA au plus d'après [S3], pour une limite de 1 mA (§ 3.4.3). **Statut : Conforme (conception).**

**EXIG_INTER** — L'interrupteur relié à J2 ne commande que la grille de Q1 : il conduit 84 µA (89 µA au pire), alors que Q1 transporte tout le courant du robot. **Statut : Conforme (conception).**

## Perception, départ et comportement

**EXIG_ADVERSAIRE** — Trois VL53L3CX (25 à 3000 mm) sur une tourelle de ±140° couvrent le dohyo ; le suivi a été validé sur 55 scénarios (SIM06) et la décision embarquée gagne 73,7 % des combats de validation contre onze comportements adverses (SIM04). **Statut : Conforme (conception).**

**EXIG_LUMINOSITE** — Le CNY70 possède un filtre de lumière du jour, le TSOP34438 ne réagit qu'à une trame modulée à 38 kHz et le VL53L3CX mesure le temps de vol d'une impulsion laser filtrée à 940 nm ; le seuil des CNY70 (1,5 V) est placé à mi-chemin des niveaux noir et blanc estimés sous 3,3 V, avec 0,39 V de marge de chaque côté (§ 3.2.2). **Statut : Conforme (conception).**

**EXIG_DEPART** — Les rapports cycliques restent nuls jusqu'à la réception d'une trame IRC01 valide (touche quelconque, hors répétition) ; le bruit et les impulsions des télémètres adverses ne forment pas de trame décodable et sont ignorés (§ 3.2.3, SIM07 : 6 cas sur 6). **Statut : Conforme (conception).**

**EXIG_DEPLACEMENT** — Les quatre CNY70 aux coins et le pont en H permettent de parcourir le dohyo et d'en revenir dès qu'une bordure est vue. La décision embarquée ne fait pas plus de sorties sans contact que le contrôleur de référence (porte d'acceptation, SIM04), lequel n'en fait aucune sur 76 800 combats simulés (SIM05). **Statut : Conforme (conception).**

**EXIG_COMPORTEMENT** — L'ESP32-S3 décide seul des rapports cycliques des roues et de la consigne de la tourelle à partir des capteurs (§ 2.2.4) ; aucune commande humaine n'est utilisée après le départ. **Statut : Conforme (conception).**

## Carte électronique et schéma

**EXIG_CARTE** — La carte est un PCB FR4 double face de 1,6 mm sur lequel le module ESP32-S3-WROOM-1-N4 est soudé directement. Le microcontrôleur, les capteurs adversaire (référence, nombre, position), les capteurs de sol, le pont en H et les régulateurs sont justifiés par les FAD du § 2.1. **Statut : Conforme (conception).**

**EXIG_CARTE_VISU** — D2 (R10 = 1 kΩ) est reliée au rail +5 V et témoigne de l'alimentation ; D1 (R1 = 1 kΩ), commandée par GPIO15, affiche l'état batterie faible. **Statut : Conforme (conception).**

**EXIG_SCH_LISIBILITE** — Le schéma est hiérarchique (racine, microcontrolleur, alimentation, capteurs), avec des cadres titrés par fonction et des étiquettes nommées. Le 13/11/2026, les titres des cadres de la feuille capteurs (« ARRIERE – RL et RR », « AVANT – FL et FR ») et son numéro de page (4/4) ont été corrigés. **Statut : Conforme (conception).**

**EXIG_SCH_SYMBOLES** — Les symboles des bibliothèques KiCad sont utilisés pour les composants usuels (résistances, condensateurs, TSOP348xx, LD1117, AP63200, IRLML6244, connecteurs). Les symboles locaux (module Pololu 2135, Si2369DS, CNY70) sont regroupés dans la bibliothèque du projet ; leur brochage correspond aux empreintes standard. Le symbole CNY70 porte depuis le 13/11/2026 le nom « CNY70_EmetteurCommun », conforme au câblage réel. **Statut : Conforme (conception).**

**EXIG_SCH_VALEURS** — Toutes les résistances, capacités et inductances portent leur valeur sur le schéma, et la plupart leur tolérance ou leur tension de service (R8 = 157 kΩ / 1 %, C15 = 470 µF / 10 V, L1 = 10 µH / 20 %). **Statut : Conforme (conception).**

## Implantation et fabrication PCB

**EXIG_PCB_EMPREINTES** — Les empreintes KiCad standard sont utilisées (SOT-23, TSOT-23-6, SOT-223, 1206). Les variantes locales répondent à une absence d'empreinte standard (module Pololu, boutons MC32882) ou à une demande de l'enseignant (pastilles oblongues, connecteurs Grove coudés, dégagement d'antenne de l'ESP32-S3). **Statut : Conforme (conception).**

**EXIG_PCB_PISTES** — Sur 279 segments, les 163 segments de puissance mesurent 0,8 mm et les 116 segments de signal 0,4 mm ou 0,6 mm (retour FB de U2). Les deux segments du nœud de commutation de U2 ont été élargis de 0,5 à 0,8 mm le 13/11/2026. 10 mm de piste de 0,8 mm présentent 6,2 mΩ, soit 18,5 mV de chute à 3 A. **Statut : Conforme (conception).**

**EXIG_PCB_VIAS** — Quatre vias de 1,3 / 0,7 mm : trois sur la masse et un seul hors masse (ESP_EN), pour une limite de douze. **Statut : Conforme (conception).**

**EXIG_PCB_GND** — Les liaisons de masse ont été tracées en pistes de 0,8 mm (20 segments) et en vias avant le remplissage des zones ; le DRC ne relève aucune connexion manquante. **Statut : Conforme (conception).**

**EXIG_PCB_CMS** — La carte porte 13 résistances et 12 condensateurs non polarisés, tous en 1206 ; les régulateurs U2 et U3 sont CMS. Seuls les trois électrolytiques C4, C14 et C15 sont traversants, ce que le [CDC] autorise. **Statut : Conforme (conception).**

**EXIG_PCB_FIXATION** — Quatre trous non métallisés de 3,2 mm pour vis M3, sans piste ni composant sous la tête des vis sur les deux faces. **Statut : Conforme (conception).**

### Logiciel

**EXIG_CODE_LANGAGE** — Le code embarqué est écrit en C++ (Arduino-ESP32) ; l'entraînement du réseau, en Python, s'exécute uniquement sur PC. **Statut : Conforme (conception).**

**EXIG_CODE_FONCTION** — Le décompte automatique des 97 fonctions du code embarqué écrit par l'équipe donne 48 lignes au plus (Tracker::makeCluster) pour une limite de 60 ; les deux fonctions de suivi qui dépassaient la limite ont été découpées le 13/11/2026 sans changement de comportement (§ 3.5.1.2). **Statut : Conforme (conception).**

**EXIG_CODE_IMBRICATION** — L'imbrication maximale du code embarqué est de trois niveaux ; la couche d'entrées/sorties ne dépasse pas deux niveaux. **Statut : Conforme (conception).**

**EXIG_CODE_VARIABLE** — Les noms décrivent l'usage et l'unité (tensionMv, debutMesureMs, BUDGET_MESURE_US). Le code embarqué ne déclare aucune variable globale : l'état est porté par des structures locales passées en paramètre (Telemetres, Tourelle, EtatBatterie) ou par les objets Tracker et SensorPolicy. **Statut : Conforme (conception).**

**EXIG_CODE_TEST** — Une fonction d'essai existe pour chaque unité matérielle : batterie, LED D1, CNY70, chaque moteur, servo et Q7, télémètres, centrale inertielle et récepteur IR, appelées depuis un menu sur la liaison série (tests.cpp). **Statut : Conforme (conception).**

### Projet, coût et livrables

**EXIG_DELAI** — Le planning RMS_PDP v4.1 prévoit 83,8 h par équipier pour 84 h autorisées, réparties par binôme et par bloc (§ 2.3) ; il est mis à jour à chaque séance. **Statut : Conforme (conception).**

**EXIG_COUT** — La nomenclature chiffrée donne 109,61 € HT pour un prototype, soit 10,39 € de marge sous 120 € HT, avec une allocation bloc par bloc (§ 2.3). **Statut : Conforme (conception).**

**EXIG_FOURNISSEUR** — Les composants viennent d'abord de la liste préférentielle et du stock du département (CNY70, batterie, Pololu, GM9), puis des fournisseurs autorisés RS, Gotronic et Farnell ; la nomenclature FAB02 indique la provenance de chaque ligne. **Statut : Conforme (conception).**

**EXIG_DRIVE** — Schémas, code, nomenclature et dossiers sont déposés dans le Google Drive partagé de l'équipe, accessible aux enseignants ; le dépôt Git conserve l'historique. **Statut : Conforme (conception).**

**EXIG_FORMAT_DOC** — Le DDC, le DDF et le DDV sont remis au format PDF ; les FAD sont aussi disponibles au format tableur. **Statut : Conforme (conception).**

**EXIG_NOM_DOC** — Les livrables portent le nom normalisé %groupe%_DDC_EQ13_V%version% : 13_DDC_EQ13_V2 et 13_DDF_EQ13_V1. **Statut : Conforme (conception).**

### Sources techniques

[S1] Diodes Incorporated, AP63200/AP63201/AP63203/AP63205, DS41326 rév. 3-2, p. 1, 5 et 13, tableau 1.
[S2] STMicroelectronics, LD1117, DocID2572 rév. 38, p. 1, 6–7 et 18, tableau 12.
[S3] Vishay, Si2369DS, P-Channel 30 V (D-S) MOSFET, p. 1–2.
[S4] Infineon, IRLML6244, PD-97535A, p. 1–2.
[S5] Pololu, DRV8835 Dual Motor Driver Carrier #2135 ; Texas Instruments, DRV8835, tableau 2.
[S6] Solarbotics, GM9 Motor Data, tableau Default RM3.
[S7] Espressif, ESP32-S3-WROOM-1 / 1U Datasheet, version 1.8, p. 2–3 et 10–12.
[S8] Espressif, ESP32-S3 Series Datasheet, version 2.2, p. 24–26 ; Arduino-ESP32, API LEDC et ADC.
[S9] Vishay, CNY70, document 83751.
[S10] Vishay, TSOP322/324/344/348, document 82489.
[S11] DFRobot, SEN0378 (VL53L3CX) ; STMicroelectronics, VL53L3CX, DS13008.
[S12] Joy-IT, SEN-MPU6050 ; TDK InvenSense, MPU-6000/6050 Register Map, rév. 4.2 (registres 0x1B, 0x1C, 0x3B, 0x6B).
[S13] Coilcraft, XAL6060-103MEC.
[S14] Feetech, FT90M.
[S15] NXP, UM10204, I²C-bus specification and user manual, rév. 7, tableau 10 et § 7.1.
[S16] Vishay, TCRT5000, document 83760.
[S24] STMicroelectronics, LSM6DSO, DS12140.
[S25] J. Schulman et al., « Proximal Policy Optimization Algorithms », arXiv:1707.06347, 2017.
[S26] S. Ross, G. Gordon, J. A. Bagnell, « A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning » (DAgger), AISTATS 2011.
[S27] Arduino-IRremote, version 4.x, documentation de l'API IrReceiver (decode, resume, IRDATA_FLAGS_IS_REPEAT).
[C1] Cours et TD d'électronique du BUT GEII : loi d'Ohm, pont diviseur, transistor en commutation (émetteur commun), MOSFET, hacheur série, régulateur linéaire et dissipation thermique.

## 2.5 Conclusion de la conception préliminaire du produit

Chaque solution technique est sélectionnée par une fiche d'aide à la décision notée de 1 à 5 et justifiée par une synthèse : télémètres VL53L3CX sur tourelle, quatre CNY70 aux coins, récepteur TSOP34438, pont en H DRV8835, convertisseur AP63200 et régulateur LD1117, protection et interrupteur à MOSFET canal P, centrale MPU6050 et microcontrôleur ESP32-S3. Les caractéristiques des informations échangées entre les quatre blocs sont définies. La décision est confiée à un réseau de neurones entraîné par renforcement en simulation, qui n'a été retenu qu'après validation sur des combats inédits.

Le coût matériel prévisionnel (109,61 € HT) respecte le plafond de 120 € HT et le planning (83,8 h par équipier) respecte les 84 h. Les 36 exigences du [CDC] reçoivent une réponse conforme au

stade de la conception ; le chapitre 3 en détaille les dimensionnements et le chapitre 4 les dérisquages.

# 3. Conception détaillée du produit

Ce chapitre détaille la conception du produit développé. Il constitue une preuve de la conformité du produit. Pour chaque solution technique, la démarche suit le même ordre : exigences du [CDC] concernées, sources du montage et des caractéristiques (fiches techniques [S…], cours [C1]), dimensionnement détaillé, schéma retenu, puis vérification du respect de l'exigence et de sa tolérance ; pour le logiciel : source et justification de l'algorithme, code retenu et respect des règles de codage. Les figures sont extraites de l'export PDF du schéma KiCad (FAB01 du [DDF]).

## 3.1 Conception détaillée du robot mini-sumo

**Référence de conception :** CDT_CARTE

**Exigences client vérifiées :** EXIG_CARTE, EXIG_SCH_LISIBILITE, EXIG_SCH_SYMBOLES, EXIG_SCH_VALEURS, EXIG_PCB_CMS

À la suite de la conception préliminaire, la conception détaillée a été menée feuille par feuille (tableau 20). La feuille racine est donnée en figure 2.

**Tableau 20 :** Organisation du schéma KiCad

| Feuille | Contenu | Composants principaux | Paragraphe |
|---|---|---|---|
| 1 – racine | Liaisons entre feuilles, points de test | TP101 à TP105 | § 3.1 |
| 2 – microcontrolleur | ESP32-S3, démarrage, UART, servo, moteurs, ToF, IMU, CNY70 | U1, U4, Q7, J1, J3 à J5, J7 à J11, J13 | § 3.2, 3.3, 3.5 |
| 3 – alimentation | Batterie, marche/arrêt, mesure, 5 V, 3,3 V, LED | Q1, Q2, U2, U3, L1, R104/R105, D1, D2 | § 3.4 |
| 4 – capteurs | Quatre CNY70, récepteur de départ (cartes externes) | U5 à U9, R16 à R23, J6 | § 3.2.2, 3.2.3 |

**Respect de l'exigence et de sa tolérance.** Toutes les valeurs sont portées sur le schéma (EXIG_SCH_VALEURS) ; les résistances et condensateurs non polarisés de la carte sont en 1206 et les régulateurs en CMS (EXIG_PCB_CMS) ; les cadres et le cartouche de la feuille capteurs ont été corrigés le 13/11/2026 (EXIG_SCH_LISIBILITE) ; le contrôle ERC ne relève aucune erreur et le DRC du 13/11/2026 aucune erreur (SIM08). **Conforme (conception).**

## 3.2 Conception détaillée de la partie acquisition

### 3.2.1 Capteurs adversaires et centrale inertielle

**Référence de conception :** CDT_ACQUISITION

**Exigences client vérifiées :** EXIG_ADVERSAIRE, EXIG_LUMINOSITE, EXIG_COMPORTEMENT

#### *3.2.1.1 Schéma électrique des capteurs adversaires*

Les trois modules SEN0378 (gauche, centre, droite) se branchent sur J3, J4 et J5 ; la centrale inertielle sur J1 (figure 1, figure 7).

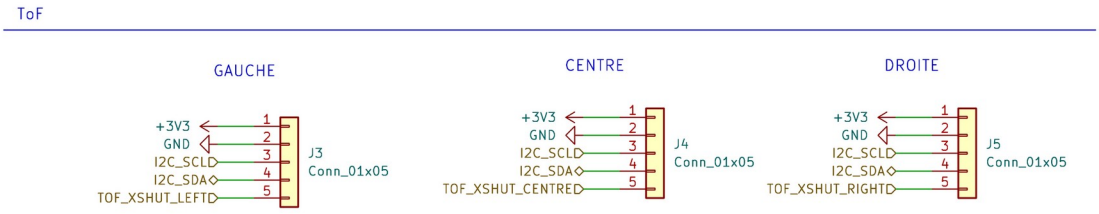



**Figure 6 :** Feuille 2 : connecteurs J3, J4 et J5 des trois télémètres

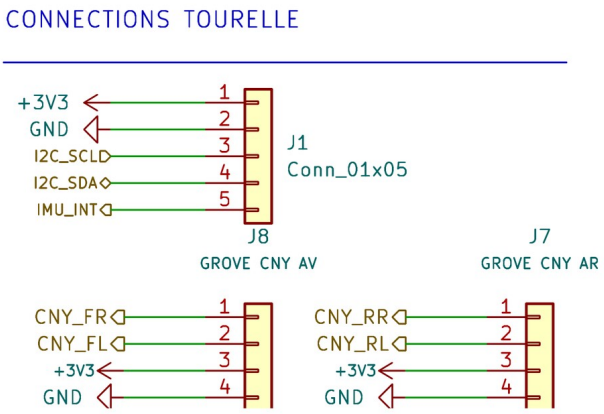



**Figure 7 :** Feuille 2 : connecteur J1 de la centrale inertielle et connecteurs Grove des CNY70

### Alimentation

Le module SEN0378 accepte 3,3 à 5 V [S11] ; il est alimenté par le +3V3, ce qui rend ses niveaux logiques directement compatibles avec l'ESP32-S3 sans adaptation. Le module MPU6050 accepte 3,3 à 5 V avec une logique 3,3 V [S12] et reçoit aussi le +3V3.

### Bus I²C

Les quatre modules partagent SDA (GPIO41) et SCL (GPIO42) en mode rapide à 400 kHz. Le bus étant à drain ouvert, les résistances de tirage intégrées aux modules se mettent en parallèle. La résistance équivalente $R_p$ doit respecter le courant d'état bas de 3 mA et le temps de montée de 300 ns du mode rapide [S15] :

$$R_{p\,min} = (V_{DD} - V_{OL}) / I_{OL} = (3{,}3 - 0{,}4) / 3\ \text{mA} = 967\ \Omega$$

$$R_{p\,max} = t_r / (0{,}847 \times C_b) = 300\ \text{ns} / (0{,}847 \times 100\ \text{pF}) = 3{,}54\ \text{k}\Omega$$

avec $C_b \approx 100$ pF pour quatre modules et environ 30 cm de câble vers la tourelle. La carte ne porte pas de résistance de tirage : celles des modules SEN0378 et SEN-MPU6050, en parallèle, sont utilisées. Leur valeur équivalente est mesurée à l'ohmmètre entre SDA et +3V3, modules branchés et hors tension, lors de l'assemblage. Elle doit être comprise entre 967 Ω et 3,54 kΩ ; au-delà, la fréquence du bus serait réduite (FREQUENCE_I2C_HZ) jusqu'à obtenir des fronts conformes à l'oscilloscope.

### Adressage par les lignes XSHUT

Les trois VL53L3CX démarrent tous à l'adresse 0x29 (7 bits). Chacun possède une ligne XSHUT (GPIO37, GPIO38, GPIO39) : le programme les maintient à l'arrêt, libère un seul capteur, lui donne une nouvelle adresse (0x2A, 0x2B puis 0x2C), puis passe au suivant. La version N4 du module

ESP32-S3 laisse GPIO37 libre (pas de PSRAM Octal) [S7]. L'interruption de l'IMU arrive sur GPIO40 ; son adresse 0x68 ne chevauche pas celles des télémètres.

**Mesure**

Chaque capteur mesure en mode longue distance avec un budget de 20 ms. Le suivi accepte les retours de 5 à 850 mm : 850 mm couvre le diamètre de 770 mm du dohyo avec une marge pour la position des capteurs, et élimine l'arrière-plan lointain. Avant le combat, une calibration balaie le dohyo vide et apprend, par capteur et par angle de tourelle, les retours fixes (sol, pièces du robot), qui sont ensuite ignorés.

**Servomoteur de tourelle**

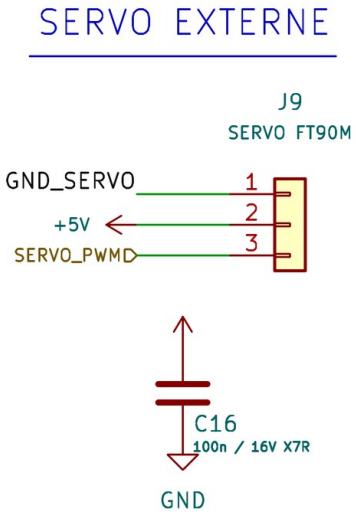



**Figure 8 :** Feuille 2 : connecteur J9 du servomoteur FT90M et découplage C16

Le FT90M accepte 3 à 8,4 V ; son courant de blocage est de 800 mA sous 4,8 V et son courant au repos de 5 mA [S14]. Il reçoit le +5 V sur J9.2, le signal SERVO_PWM (GPIO12, 50 Hz) sur J9.3 et sa masse GND_SERVO sur J9.1, coupée par Q7 (§ 3.4.3). La consigne 0 à 280° correspond à 1000 à 2000 µs ; 140° (1500 µs) aligne le capteur central sur l'avant du robot.

Le démarrage du servo provoque un appel de courant bref. Le condensateur de réserve C15 = 470 µF / 10 V placé sur le +5 V fournit ce courant localement. D'après $I = C \times dV/dt$, avec I = 1 A (majorant du blocage), $\Delta V$ = 0,25 V admissible et $\Delta t$ = 100 µs non compensées par la régulation :

$$C \geq I \times \Delta t / \Delta V = 1\ A \times 100\ \mu s / 0{,}25\ V = \mathbf{400\ \mu F} \quad \rightarrow \quad \text{valeur normalisée } \mathbf{470\ \mu F}$$

Au courant de blocage réel de 0,8 A, la chute est de 0,8 × 100 µs / 470 µF = 0,17 V. La tension de service de 10 V offre 95 % de marge sur 5,12 V. C16 = 100 nF, au plus près de J9, filtre les perturbations haute fréquence que l'électrolytique ne peut pas absorber.

**Respect de l'exigence et de sa tolérance.** EXIG_ADVERSAIRE demande de détecter l'adversaire à 250 mm au moins : la plage garantie du VL53L3CX (25 à 3000 mm [S11]) et la fenêtre logicielle de 5 à 850 mm incluent 250 mm avec une marge de 600 mm ; l'angle mort avant est nul puisque le capteur central regarde l'avant quand la tourelle est à 140°. Le bus I²C est dimensionné selon les limites du mode rapide de la norme [S15], et le +5 V du servo reste au-dessus de 4,85 V lors d'un appel de 0,8 A. **Conforme (conception).**

#### *3.2.1.2 Code informatique des capteurs adversaires*

**Source et justification de l'algorithme.** La séquence d'adressage (capteurs maintenus à l'arrêt par XSHUT, puis réveil et changement d'adresse un par un) est celle préconisée par ST pour plusieurs VL53LX sur un même bus [S11] ; elle évite d'ajouter un multiplexeur I²C. Les mesures sont lues par scrutation de l'indicateur « donnée prête » plutôt que par interruption, pour ne pas mobiliser trois entrées supplémentaires ; un seul télémètre est servi par passage dans la boucle, pour borner sa durée. Le suivi de cible (Tracker) est repris du prototype de tourelle validé en SIM06 : il associe les retours des trois capteurs, élimine ceux appris sur le dohyo vide pendant la calibration et produit le secteur et la distance de la cible.

```
// telemetres.cpp - Démarrage séquentiel par XSHUT et lecture multi-cibles des VL53L3CX
#include "telemetres.h"

static VL53LX &capteur(Telemetres &telemetres, uint8_t indice) {
  if (indice == 0) return telemetres.gauche;
  if (indice == 1) return telemetres.centre;
  return telemetres.droite;
}

static bool demarrerTelemetre(Telemetres &telemetres, uint8_t indice) {
  const uint8_t adresses[NOMBRE_TELEMETRES] = {ADRESSE_TOF_GAUCHE, ADRESSE_TOF_CENTRE,
                                               ADRESSE_TOF_DROITE};
  VL53LX &tof = capteur(telemetres, indice);
  tof.begin();  // XSHUT en sortie
  // InitSensor libère XSHUT, attend le démarrage puis change l'adresse 0x29.
  if (tof.InitSensor(adresses[indice]) != 0) return false;
  if (tof.VL53LX_SetDistanceMode(VL53LX_DISTANCEMODE_LONG) != 0) return false;
  if (tof.VL53LX_SetMeasurementTimingBudgetMicroSeconds(BUDGET_MESURE_US) != 0) return false;
  if (tof.VL53LX_StartMeasurement() != 0) return false;
  telemetres.debutMesureMs[indice] = millis();
  return true;
}

bool initialiserTelemetres(Telemetres &telemetres) {
  const uint8_t xshut[NOMBRE_TELEMETRES] = {PIN_TOF_XSHUT_GAUCHE, PIN_TOF_XSHUT_CENTRE,
                                            PIN_TOF_XSHUT_DROITE};
  for (uint8_t i = 0; i < NOMBRE_TELEMETRES; i++) {
    pinMode(xshut[i], OUTPUT);
    digitalWrite(xshut[i], LOW);  // les trois capteurs à l'arrêt : adresses libres
  }
  if (!Wire.begin(PIN_I2C_SDA, PIN_I2C_SCL, FREQUENCE_I2C_HZ)) return false;
  for (uint8_t i = 0; i < NOMBRE_TELEMETRES; i++) {
    if (!demarrerTelemetre(telemetres, i)) return false;
  }
  return true;
}

static void copierTrame(const VL53LX_MultiRangingData_t &donnees, uint32_t debutMs,
                        uint32_t finMs, sumo::Frame &trame) {
  trame = sumo::Frame{};
  trame.startedMs = debutMs;
  trame.capturedMs = finMs;
  trame.sequence = donnees.StreamCount;
  trame.count = donnees.NumberOfObjectsFound > sumo::kSlots ? sumo::kSlots
                                                            : donnees.NumberOfObjectsFound;
  for (uint8_t i = 0; i < trame.count; i++) {
    const VL53LX_TargetRangeData_t &cible = donnees.RangeData[i];
    sumo::Return &retour = trame.targets[i];
    retour.mm = cible.RangeMilliMeter;
    retour.status = cible.RangeStatus;
    retour.sigmaMm = cible.SigmaMilliMeter / 65536.0F;      // format 16.16
    retour.signalMcps = cible.SignalRateRtnMegaCps / 65536.0F;
    retour.ambientMcps = cible.AmbientRateRtnMegaCps / 65536.0F;
    retour.verdict = sumo::basicVerdict(retour);
  }
}

// Renvoie vrai quand une nouvelle mesure complète est disponible dans trame.
bool lireTelemetre(Telemetres &telemetres, uint8_t indice, sumo::Frame &trame) {
```

```
  VL53LX &tof = capteur(telemetres, indice);
  uint8_t prete = 0;
  if (tof.VL53LX_GetMeasurementDataReady(&prete) != 0 || prete == 0) return false;
  VL53LX_MultiRangingData_t donnees{};
  if (tof.VL53LX_GetMultiRangingData(&donnees) != 0) return false;
  const uint32_t maintenant = millis();
  copierTrame(donnees, telemetres.debutMesureMs[indice], maintenant, trame);
  telemetres.debutMesureMs[indice] = maintenant;
  return tof.VL53LX_ClearInterruptAndStartMeasurement() == 0;
}
```

**Programme 1 :** Module telemetres.cpp : adressage et lecture des trois VL53L3CX

- **initialiserTelemetres()** met les trois XSHUT à 0, démarre l'I²C à 400 kHz sur GPIO41/42, puis réveille chaque capteur et lui donne son adresse (0x2A, 0x2B, 0x2C sur 7 bits).
- **lireTelemetre()** renvoie vrai lorsqu'une nouvelle mesure complète est disponible ; elle copie au plus quatre cibles dans une trame horodatée au format attendu par le Tracker, puis relance la mesure.
- Chaque appel du pilote est vérifié : une erreur arrête l'initialisation et est signalée, au lieu de laisser un capteur muet.

La centrale inertielle est lue par le module imu.cpp (programme 2) : sortie de veille, plages ±500 °/s et ±4 g, puis lecture en rafale des 14 octets de mesure à partir du registre 0x3B [S12].

```
// imu.cpp - Configuration ±500 °/s, ±4 g et lecture en rafale des registres 0x3B à 0x48
#include "imu.h"
#include <Wire.h>

constexpr uint8_t REG_PWR_MGMT_1 = 0x6B;
constexpr uint8_t REG_GYRO_CONFIG = 0x1B;
constexpr uint8_t REG_ACCEL_CONFIG = 0x1C;
constexpr uint8_t REG_ACCEL_XOUT_H = 0x3B;
constexpr uint8_t NOMBRE_OCTETS_MESURE = 14;    // accéléro, température, gyro
constexpr float SENSIBILITE_GYRO_LSB = 65.5F;   // LSB/(°/s) à ±500 °/s
constexpr float SENSIBILITE_ACCEL_LSB = 8192.0F;  // LSB/g à ±4 g

static bool ecrireRegistre(uint8_t registre, uint8_t valeur) {
  Wire.beginTransmission(ADRESSE_IMU);
  Wire.write(registre);
  Wire.write(valeur);
  return Wire.endTransmission() == 0;
}

static int16_t motSigne(const uint8_t octets[], uint8_t indice) {
  return static_cast<int16_t>((octets[indice] << 8) | octets[indice + 1]);  // poids fort
d'abord
}

bool initialiserImu() {
  pinMode(PIN_IMU_INT, INPUT);
  if (!ecrireRegistre(REG_PWR_MGMT_1, 0x00)) return false;   // sortie de veille
  if (!ecrireRegistre(REG_GYRO_CONFIG, 0x08)) return false;  // FS_SEL = 1 : ±500 °/s
  return ecrireRegistre(REG_ACCEL_CONFIG, 0x08);             // AFS_SEL = 1 : ±4 g
}

bool lireImu(MesureImu &mesure) {
  uint8_t octets[NOMBRE_OCTETS_MESURE] = {};
  Wire.beginTransmission(ADRESSE_IMU);
  Wire.write(REG_ACCEL_XOUT_H);
  if (Wire.endTransmission(false) != 0) return false;  // redémarrage sans stop
  if (Wire.requestFrom(ADRESSE_IMU, NOMBRE_OCTETS_MESURE) != NOMBRE_OCTETS_MESURE) return
false;
  for (uint8_t i = 0; i < NOMBRE_OCTETS_MESURE; i++) octets[i] = Wire.read();
  mesure.accelAvantG = motSigne(octets, 0) / SENSIBILITE_ACCEL_LSB;
  mesure.accelLateraleG = motSigne(octets, 2) / SENSIBILITE_ACCEL_LSB;
  mesure.vitesseLacetDegS = motSigne(octets, 12) / SENSIBILITE_GYRO_LSB;
  return true;
}
```

**Programme 2 :** Module imu.cpp : configuration et lecture du MPU6050

La tourelle est commandée par **orienterTourelle()** de la couche d'entrées/sorties (programme 3), qui convertit un angle par rapport à l'avant en largeur d'impulsion puis en rapport cyclique sur 14 bits.

```
// tourelle.cpp - Commande du servo FT90M de la tourelle (PWM 50 Hz sur IO12)
// Même convention que le prototype : consigne 0..280, 140 = axe avant (1500 µs).
#include "tourelle.h"

constexpr uint32_t FREQUENCE_SERVO_HZ = 50;
constexpr uint8_t RESOLUTION_SERVO_BITS = 14;
constexpr int32_t PERIODE_SERVO_US = 20000;
constexpr int32_t IMPULSION_MIN_US = 1000;   // consigne 0   : butée droite
constexpr int32_t IMPULSION_MAX_US = 2000;   // consigne 280 : butée gauche
constexpr int16_t CONSIGNE_MAX = 280;
constexpr int16_t CONSIGNE_AVANT = 140;

void initialiserTourelle(Tourelle &tourelle) {
  ledcAttach(PIN_SERVO_PWM, FREQUENCE_SERVO_HZ, RESOLUTION_SERVO_BITS);
  tourelle.sensBalayage = 1;
  orienterTourelle(tourelle, 0);
}

// angleDeg : lacet par rapport à l'avant du robot, positif à gauche (-140..+140)
void orienterTourelle(Tourelle &tourelle, int16_t angleDeg) {
  int32_t consigne = constrain(CONSIGNE_AVANT + angleDeg, 0, CONSIGNE_MAX);
  tourelle.angleDeg = static_cast<int16_t>(consigne - CONSIGNE_AVANT);
  int32_t impulsionUs = IMPULSION_MIN_US
      + consigne * (IMPULSION_MAX_US - IMPULSION_MIN_US) / CONSIGNE_MAX;
  uint32_t rapportCyclique = (impulsionUs << RESOLUTION_SERVO_BITS) / PERIODE_SERVO_US;
  ledcWrite(PIN_SERVO_PWM, rapportCyclique);
}

void suivreCible(Tourelle &tourelle, int8_t direction) {
  orienterTourelle(tourelle, tourelle.angleDeg + direction * 3);
}

void balayer(Tourelle &tourelle) {
  if (abs(tourelle.angleDeg) >= CONSIGNE_AVANT) {
    tourelle.sensBalayage = (tourelle.angleDeg > 0) ? -1 : 1;  // demi-tour en butée
  }
  orienterTourelle(tourelle, tourelle.angleDeg + tourelle.sensBalayage * 16);
}
```

**Programme 3 :** Module tourelle.cpp : commande du servomoteur FT90M

**Respect de l'exigence et de sa tolérance.** Les fonctions de ce bloc comptent au plus 13 lignes et deux niveaux d'imbrication, n'utilisent aucune variable globale (l'état est dans les structures Telemetres et Tourelle) et chaque unité a sa fonction d'essai (testTelemetres, testImu, testServoEtCoupure). **Conforme (conception).**

### 3.2.2 Capteurs de sol

**Référence de conception :** CDT_SOL

**Exigences client vérifiées :** EXIG_DEPLACEMENT, EXIG_LUMINOSITE

#### *3.2.2.1 Schéma électrique des capteurs de sol*

Les quatre CNY70 (U6 à U9) sont placés aux coins du robot, sur deux petites cartes reliées à la carte principale par les connecteurs Grove J8 (avant) et J7 (arrière) (figure 9). Chaque capteur associe une LED infrarouge qui éclaire le sol et un phototransistor qui convertit la lumière réfléchie en courant.

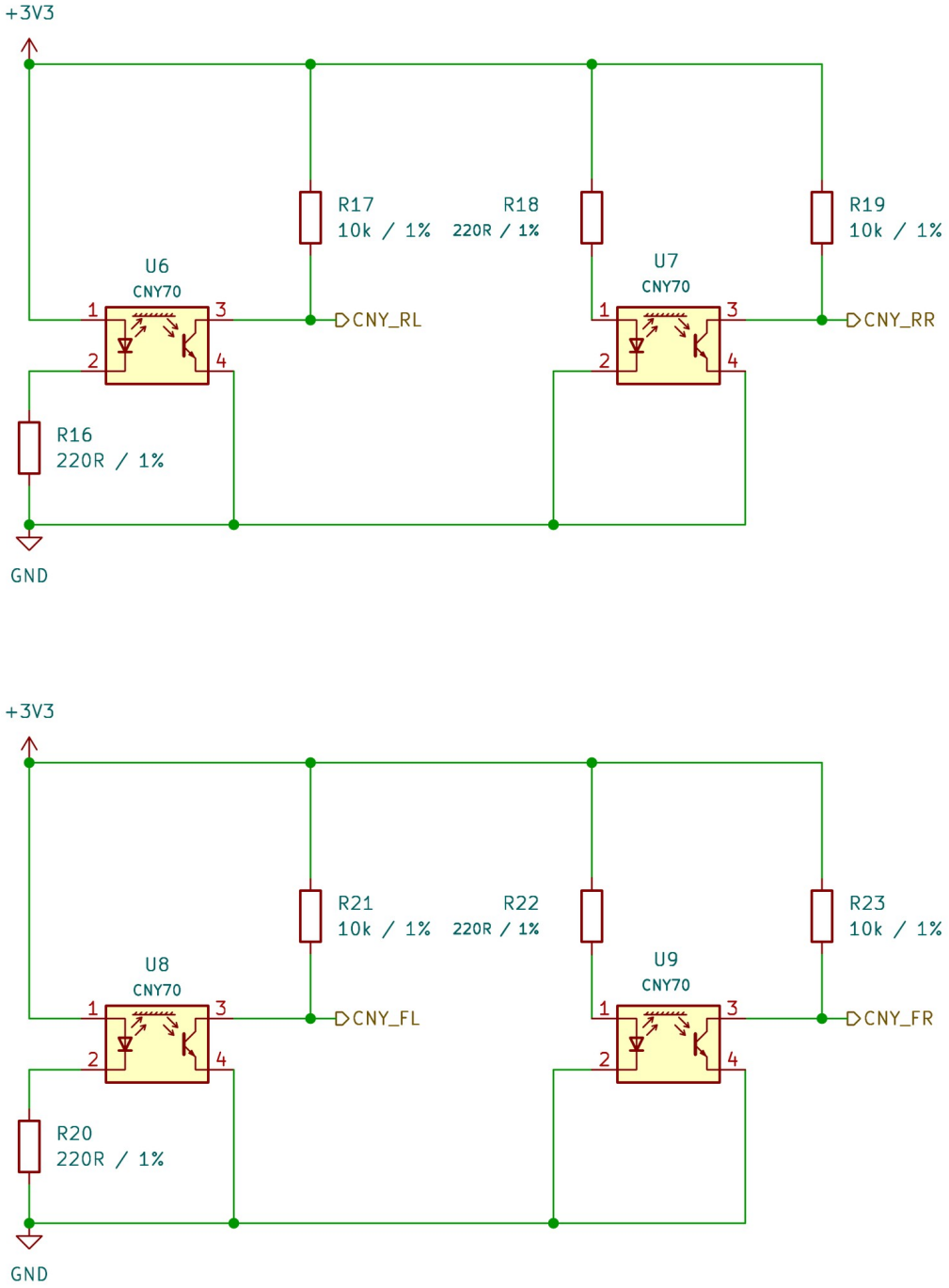



**Figure 9 :** Feuille 4 : les quatre CNY70 ; en haut la carte arrière (U6, U7), en bas la carte avant (U8, U9) ; résistances de LED R16, R18, R20, R22 et de collecteur R17, R19, R21, R23

### Résistance de limitation de la LED

La LED est alimentée par le +3V3 à travers une résistance de 220 Ω. Avec $V_F \approx 1{,}25$ V [S9] :

$$I_F = (V_{CC} - V_F) / R = (3{,}3 - 1{,}25) / 220\ \Omega = \mathbf{9{,}3\ mA}$$

Ce courant reste cinq fois inférieur au maximum absolu de 50 mA ; dans le cas le plus défavorable ($V_F$ = 0, 3,44 V, résistance à −1 %), il reste inférieur à 15,8 mA. La résistance dissipe 19 mW, très en dessous des 125 mW d'une résistance 1206. La valeur normalisée 220 Ω (E12) est prise à ±1 % pour que les quatre capteurs aient la même sensibilité.

**Résistance de collecteur (montage émetteur commun)**

Le collecteur du phototransistor est tiré au +3V3 par 10 kΩ et son émetteur est relié à la masse. La tension de sortie vaut :

$$V_{OUT} = V_{CC} - R \times I_C$$

Sur le noir, peu de lumière est réfléchie : $I_C$ est faible et la tension reste haute. Sur le blanc, $I_C$ augmente et le transistor se sature : la tension tombe vers $V_{CE\,sat} \approx 0{,}3$ V. Un courant de 0,33 mA suffit à saturer le transistor avec 10 kΩ ; la fiche donne 1 mA typique sur blanc à 0,3 mm [S9]. La résistance de 10 kΩ fixe donc un fort écart entre noir et blanc tout en limitant le courant de repos à 0,33 mA par capteur. **Le blanc donne une tension basse.**

**Mesures sur prototype**

Le même montage (220 Ω, 10 kΩ) a été essayé sur Arduino UNO sous 5 V (SIM03), avec V = N × 5 / 1023 (tableau 21).

**Tableau 21 :** Mesures d'un CNY70 sur prototype (Arduino UNO, 5 V)

| Surface | Valeur ADC (10 bits) | Tension correspondante |
|---|---|---|
| Noire (intérieur du dohyo) | 500 à 600 | 2,44 à 2,93 V |
| Blanche (bordure) | 0 à 200 | 0 à 0,98 V |

L'écart minimal entre les deux surfaces est de 2,44 − 0,98 = 1,46 V. Sur la carte, le CNY70 est alimenté en 3,3 V : le courant de LED passe de (5 − 1,25) / 220 = 17 mA à 9,3 mA et le courant du phototransistor, proportionnel au flux reçu, est multiplié par 9,3 / 17 = 0,55 [S9]. Les courants déduits des mesures ($I_C$ = (5 − $V_{OUT}$) / 10 kΩ) donnent alors :

$$\text{noir} : I_C \le 0{,}256\ \text{mA} \times 0{,}55 = 0{,}141\ \text{mA} \rightarrow V_{OUT} \ge 3{,}3 - 10\ \text{k}\Omega \times 0{,}141\ \text{mA} = \mathbf{1{,}89\ V}$$

$$\text{blanc} : I_C \ge 0{,}402\ \text{mA} \times 0{,}55 = 0{,}221\ \text{mA} \rightarrow V_{OUT} \le 3{,}3 - 10\ \text{k}\Omega \times 0{,}221\ \text{mA} = \mathbf{1{,}10\ V}$$

Le seuil SEUIL_BORDURE_MV est placé à mi-chemin, **1,5 V**, avec 0,39 V de marge de chaque côté. Les sorties rejoignent GPIO4 (avant droit), GPIO5 (avant gauche), GPIO6 (arrière droit) et GPIO7 (arrière gauche), toutes sur l'ADC1, indépendant de la radio, dont la pleine échelle (environ 3,1 V à 11 dB) couvre ces niveaux.

**Respect de l'exigence et de sa tolérance.** EXIG_DEPLACEMENT impose de détecter la bordure blanche : les niveaux estimés (blanc ≤ 1,10 V, noir ≥ 1,89 V) sont séparés par le seuil de 1,5 V avec 0,39 V de marge, supérieure à l'erreur de l'ADC étalonné (quelques dizaines de mV [S8]) ; le courant de LED (9,3 mA, 15,8 mA au pire) reste sous les 50 mA admis ; le filtre de lumière du jour du CNY70 répond à EXIG_LUMINOSITE. Le seuil sera ajusté sur la carte définitive, sous plusieurs éclairages, lors de l'essai de réception. **Conforme (conception).**

### *3.2.2.2 Code informatique des capteurs de sol*

La fonction **lireCapteursSol()** (programme 4) mesure les quatre tensions en millivolts avec analogReadMilliVolts(), qui applique l'étalonnage d'usine de l'ADC, et compare chacune au seuil : en dessous, le capteur voit la bordure. **bordureAvant()** et **bordureArriere()** regroupent les deux

capteurs d'un même côté. Ces quatre indicateurs forment le tableau edge[4] transmis au contrôleur du châssis, dont la récupération de bordure (EdgeRecovery) préempte tout combat.

**Source et justification de l'algorithme.** La comparaison à un seuil unique est l'algorithme le plus simple qui distingue deux niveaux bien séparés ; elle s'exécute en quelques microsecondes à chaque pas de 5 ms. analogReadMilliVolts() est préférée à analogRead() car elle applique la courbe d'étalonnage d'usine de l'ADC [S8] : le seuil s'exprime directement en millivolts, sans dépendre de la dispersion de l'ADC d'un module à l'autre.

```
// sol.cpp - Détection de la bordure blanche par quatre CNY70 (émetteur commun)
#include "sol.h"

constexpr uint8_t PINS_CNY70[NB_CNY70] = {
  PIN_CNY70_AVANT_GAUCHE, PIN_CNY70_AVANT_DROIT,
  PIN_CNY70_ARRIERE_GAUCHE, PIN_CNY70_ARRIERE_DROIT
};
// Seuil à mi-chemin des niveaux estimés sous 3,3 V (§ 3.2.2 du DDC) :
// blanc <= 1,10 V et noir >= 1,89 V, soit 0,39 V de marge de chaque côté.
constexpr uint16_t SEUIL_BORDURE_MV = 1500;

void initialiserCapteursSol() {
  for (uint8_t i = 0; i < NB_CNY70; i++) {
    analogSetPinAttenuation(PINS_CNY70[i], ADC_11db);
  }
}

void lireCapteursSol(CapteursSol &sol) {
  for (uint8_t i = 0; i < NB_CNY70; i++) {
    sol.tensionsMv[i] = static_cast<uint16_t>(analogReadMilliVolts(PINS_CNY70[i]));
    // Collecteur tiré à +3V3 par 10 k : surface blanche => transistor saturé => tension basse
    sol.bordureDetectee[i] = sol.tensionsMv[i] < SEUIL_BORDURE_MV;
  }
}

bool bordureAvant(const CapteursSol &sol) {
  return sol.bordureDetectee[AVANT_GAUCHE] || sol.bordureDetectee[AVANT_DROIT];
}

bool bordureArriere(const CapteursSol &sol) {
  return sol.bordureDetectee[ARRIERE_GAUCHE] || sol.bordureDetectee[ARRIERE_DROIT];
}
```

**Programme 4 :** Module sol.cpp : lecture des quatre CNY70

### 3.2.3 Capteur de télécommande

**Référence de conception :** CDT_DEPART

**Exigences client vérifiées :** EXIG_DEPART, EXIG_LUMINOSITE

#### *3.2.3.1 Schéma électrique du capteur de télécommande*

Ce bloc, repris du DDF de M. Blanchard, reçoit l'ordre de départ de la télécommande IRC01 (figure 10). Le TSOP34438 (U5) intègre photodiode, amplificateur, filtre passe-bande centré sur 38 kHz et démodulateur : sa sortie OUT, active à l'état bas, restitue directement la trame. Elle rejoint GPIO21 (IR_START) par J6.

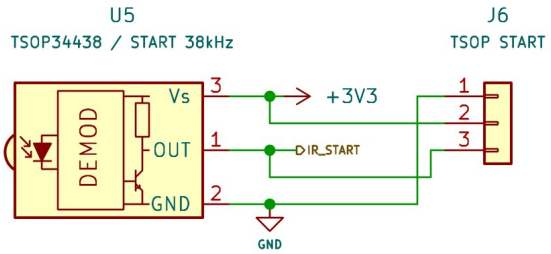



**Figure 10 :** Feuille 4 : récepteur U5 TSOP34438 et connecteur J6

Le TSOP34438 consomme moins de 1 mA et fonctionne sous 3,3 V [S10] : le +3V3 lui convient et sa sortie est directement compatible avec l'ESP32-S3. Le filtre passe-bande à 38 kHz rejette la lumière ambiante continue et les éclairages modulés à basse fréquence. Le brochage de J6 (1 GND, 2 +3V3, 3 OUT) diffère de celui du TSOP (1 OUT, 2 GND, 3 VS) : le câble doit assurer la correspondance. Le constructeur propose un filtre RC optionnel sur l'alimentation (100 Ω et 0,1 µF) ; il n'est pas monté sur la carte capteur, car le +3V3 est déjà filtré par U3, C4 et C12.

**Respect de l'exigence et de sa tolérance.** EXIG_DEPART impose que le robot reste immobile jusqu'à l'appui sur une touche de l'IRC01, même si les télémètres adverses visent le récepteur : le TSOP34438 ne transmet que les salves modulées à 38 kHz [S10], et le logiciel n'accepte qu'une trame complètement décodée (§ 3.2.3.2, SIM07). **Conforme (conception).**

### *3.2.3.2 Code informatique du capteur de télécommande*

**Source et justification de l'algorithme.** Le code d'essai du dossier [IR] reposait sur initIR() et lireIR() avec la bibliothèque IRremote [S27]. Nous en reprenons le principe sur l'ESP32-S3 (programme 5) avec quatre corrections : la broche est celle du schéma (GPIO21) ; l'API IrReceiver de la version 4 est utilisée seule ; les trames de répétition sont filtrées au lieu d'un anti-rebond de 200 ms ; la variable d'état non déclarée disparaît au profit d'une valeur de retour. Comme le [CDC] demande un départ sur l'appui d'**une** touche, toute trame décodée selon un protocole connu, hors répétition, donne le départ : le décodeur d'IRremote vérifie l'en-tête, les durées et le contrôle du protocole, ce qu'un bruit ou une impulsion de télémètre ne satisfait pas (protocole UNKNOWN).

```
// depart.cpp - Réception de l'ordre de départ IRC01 par le TSOP34438
// Reprise des fonctions initIR() et lireIR() du DDF de M. Blanchard, portées sur ESP32-S3
(IO21).
#include <IRremote.hpp>
#include "depart.h"

void initialiserRecepteurIr() {
  IrReceiver.begin(PIN_IR_DEPART, DISABLE_LED_FEEDBACK);
}

// EXIG_DEPART : le départ est donné par l'appui sur une touche quelconque de l'IRC01.
// Seule une trame complète, décodée selon un protocole connu (en-tête, durées et
// contrôle du protocole vérifiés par IRremote), autorise le départ. Le bruit, la
// lumière modulée et les impulsions des télémètres adverses donnent UNKNOWN.
bool ordreDepartRecu() {
  if (!IrReceiver.decode()) {
    return false;  // aucune trame complète reçue
  }
  const bool trameReconnue = IrReceiver.decodedIRData.protocol != UNKNOWN;
  const bool estRepetition = IrReceiver.decodedIRData.flags & IRDATA_FLAGS_IS_REPEAT;
  IrReceiver.resume();  // réarme la réception pour la trame suivante
  return trameReconnue && !estRepetition;
}
```

```
// Essai unitaire : affiche chaque trame reçue pour relever le code de la touche IRC01
void afficherTrameIr() {
  if (!IrReceiver.decode()) {
    return;
  }
  Serial.printf("Trame IR : protocole %s, commande 0x%02X\n",
                getProtocolString(IrReceiver.decodedIRData.protocol),
                IrReceiver.decodedIRData.command);
  IrReceiver.resume();
}
```

**Programme 5 :** Module depart.cpp : réception de l'ordre de départ

- **initialiserRecepteurIr()** active la réception sur GPIO21.
- **ordreDepartRecu()** renvoie vrai pour la première trame décodée d'un protocole connu, hors répétition, puis réarme la réception ; une trame non décodable est ignorée.
- **afficherTrameIr()** est la fonction d'essai qui affiche le protocole et le code de chaque trame reçue.

**Respect de l'exigence et de sa tolérance.** La simulation SIM07 rejoue six situations sur le code réel : les deux appuis de touche donnent le départ ; l'absence de trame, la répétition et les trames non décodables ne le donnent pas. Le module ne déclare aucune variable globale de l'équipe (IrReceiver est l'objet de la bibliothèque tierce). **Conforme (conception).**

## 3.3 Conception détaillée de la partie action

### 3.3.1 Pont en H

**Référence de conception :** CDT_ACTION
**Exigences client vérifiées :** EXIG_DEPLACEMENT, EXIG_CARTE

#### *3.3.1.1 Schéma électrique du pont en H*

Les deux motoréducteurs GM9 (moteurs RM3), imposés, sont raccordés par J10 et J11 au module Pololu 2135 (U4) (figure 11, figure 12).

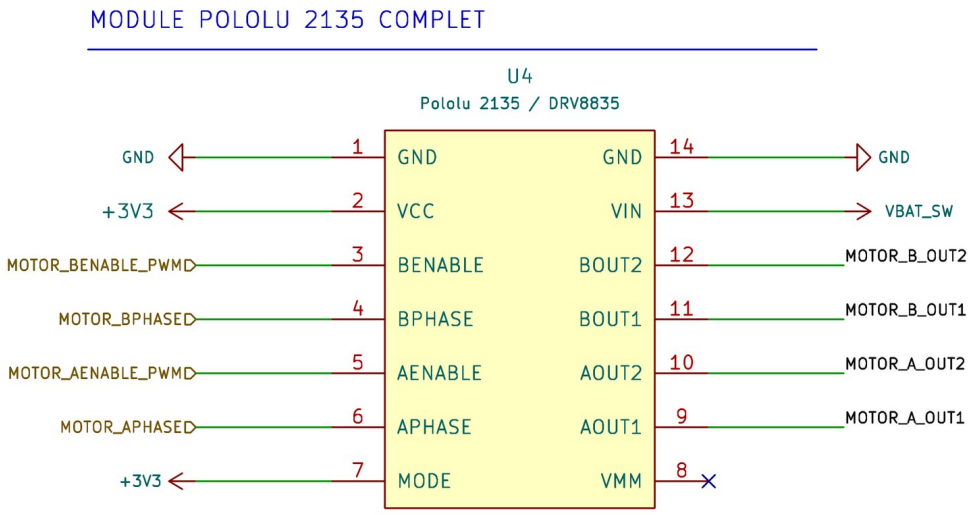



**Figure 11 :** Feuille 2 : module U4 Pololu 2135 (DRV8835)

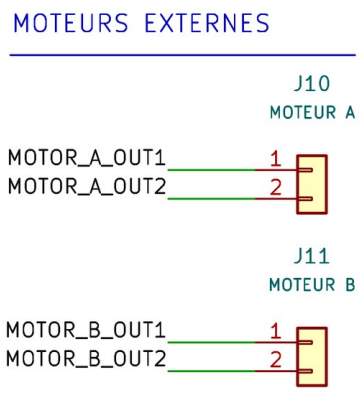



**Figure 12 :** Feuille 2 : connecteurs des moteurs J10 (moteur A) et J11 (moteur B)

### Alimentations et mode de commande

La logique du module (VCC, 2 à 7 V) est alimentée par le +3V3 ; ses entrées sont donc compatibles avec les sorties de l'ESP32-S3. La puissance (VIN, 0 à 11 V, protégée contre l'inversion sur le module) est reliée à VBAT_SW, 6,6 à 8,4 V. MODE est relié au +3V3, ce qui sélectionne le mode PHASE/ENABLE (tableau 22).

**Tableau 22 :** Fonctionnement du DRV8835 en mode PHASE/ENABLE (MODE = 1) [S5]

| xPHASE | xENABLE | xOUT1 | xOUT2 | Fonctionnement |
|---|---|---|---|---|
| 0 | PWM | PWM | L | marche avant, vitesse ∝ rapport cyclique |
| 1 | PWM | L | PWM | marche arrière, vitesse ∝ rapport cyclique |
| X | 0 | L | L | freinage (sorties à la masse) |

Chaque moteur n'utilise qu'une sortie de sens et une sortie PWM : moteur A sur GPIO17 (PHASE) et GPIO18 (ENABLE), moteur B sur GPIO8 (PHASE) et GPIO19 (ENABLE). GPIO19 est USB_D− au démarrage : la fonction USB n'est pas activée et la programmation passe par l'UART0.

### Courant

Le constructeur des GM9 (RM3) donne 0,7 A de blocage sous 6 V et 1,0 A sous 9 V [S6]. Par interpolation linéaire à 8,4 V :

$$I_{blocage} \approx 0{,}7 + (8{,}4 - 6) \times (1{,}0 - 0{,}7) / (9 - 6) = \mathbf{0{,}94\ A}\ \text{par moteur}$$

Le module admet 1,2 A en continu et 1,5 A en pointe par voie [S5] : la marge est de 0,26 A en continu, même moteur bloqué batterie pleine. Sur le prototype (SIM02), les deux moteurs consomment 300 mA en roulage et 670 mA au total en poussée contre un obstacle. Le courant consommé par la logique du module (2 mA au plus) est négligeable.

### Fréquence de découpage et arrêt

La MLI est réglée à 20 kHz, au-dessus de la bande audible et très en dessous des 250 kHz admis par le DRV8835. ENABLE = 0 met les deux sorties à la masse, ce qui freine le moteur : c'est l'arrêt logiciel retenu, les moteurs restant alimentés par VBAT_SW.

**Respect de l'exigence et de sa tolérance.** EXIG_DEPLACEMENT demande de parcourir le dohyo dans les deux sens : le mode PHASE/ENABLE commande chaque roue en marche avant, arrière et freinage (SIM02) ; le courant de blocage estimé à 8,4 V (0,94 A) reste sous la limite continue

de 1,2 A par voie [S5], et la tension VIN maximale (11 V) dépasse la tension batterie pleine (8,4 V) de 31 %. **Conforme (conception).**

#### *3.3.1.2 Code informatique du pont en H*

Le contrôleur du châssis rend un rapport cyclique signé par roue ; la fonction **phaseEnable()** de firmware/shared (programme 6) le convertit en signaux du DRV8835 : le signe donne PHASE, la valeur absolue bornée à 1 donne ENABLE. Une valeur non finie (erreur de calcul) produit un freinage.

**Source et justification de l'algorithme.** La conversion signe/valeur absolue découle directement de la table de vérité du DRV8835 en mode PHASE/ENABLE [S5] ; le passage par une valeur signée entre −1 et 1 est imposé par la sortie tanh du réseau de neurones (§ 2.2.4). Une valeur non finie est traitée comme un défaut et freine la roue.

```
// Extrait de firmware/shared/WheelCommand.h
struct PhaseEnable {
  bool phase;     // DRV8835 MODE=1: false drives forward, true reverse.
  double enable;  // 0..1 PWM; zero is electrical brake in PHASE/ENABLE mode.
};

inline PhaseEnable phaseEnable(double signedDuty) {
  if (!std::isfinite(signedDuty)) return {false, 0};
  return {signedDuty < 0, std::fmin(1.0, std::fabs(signedDuty))};
}
```

**Programme 6 :** Conversion d'un rapport cyclique signé en PHASE/ENABLE (WheelCommand.h)

La couche d'entrées/sorties applique ces signaux aux broches de la carte (programme 7).

```
// moteurs.cpp - Commande des deux GM9 par le Pololu 2135 (DRV8835) en PHASE/ENABLE
#include "moteurs.h"

constexpr uint32_t FREQUENCE_PWM_MOTEUR_HZ = 20000;  // hors bande audible
constexpr uint8_t RESOLUTION_PWM_MOTEUR_BITS = 8;    // rapport cyclique 0 à 255
constexpr bool INVERSER_MOTEUR_DROIT = true;  // moteurs tête-bêche : sens vérifié au montage

void initialiserMoteurs() {
  pinMode(PIN_MOTEUR_A_PHASE, OUTPUT);
  pinMode(PIN_MOTEUR_B_PHASE, OUTPUT);
  ledcAttach(PIN_MOTEUR_A_PWM, FREQUENCE_PWM_MOTEUR_HZ, RESOLUTION_PWM_MOTEUR_BITS);
  ledcAttach(PIN_MOTEUR_B_PWM, FREQUENCE_PWM_MOTEUR_HZ, RESOLUTION_PWM_MOTEUR_BITS);
  arreterMoteurs();
}

void commanderMoteur(CoteMoteur cote, int16_t vitesse) {
  vitesse = constrain(vitesse, -VITESSE_MAX, VITESSE_MAX);
  if (cote == MOTEUR_DROIT && INVERSER_MOTEUR_DROIT) {
    vitesse = -vitesse;
  }
  uint8_t pinPhase = (cote == MOTEUR_GAUCHE) ? PIN_MOTEUR_A_PHASE : PIN_MOTEUR_B_PHASE;
  uint8_t pinPwm = (cote == MOTEUR_GAUCHE) ? PIN_MOTEUR_A_PWM : PIN_MOTEUR_B_PWM;
  digitalWrite(pinPhase, vitesse < 0 ? HIGH : LOW);  // PHASE = 0 : marche avant
  ledcWrite(pinPwm, abs(vitesse));                   // ENABLE = PWM : vitesse
}

void commanderDeplacement(int16_t vitesseGauche, int16_t vitesseDroite) {
  commanderMoteur(MOTEUR_GAUCHE, vitesseGauche);
  commanderMoteur(MOTEUR_DROIT, vitesseDroite);
}

void arreterMoteurs() {
  commanderDeplacement(0, 0);  // ENABLE = 0 : sorties à la masse (freinage)
}
```

**Programme 7 :** Module moteurs.cpp : commande des deux moteurs sur la carte ESP32-S3

- **initialiserMoteurs()** configure les sorties de sens et associe les broches ENABLE au périphérique LEDC (20 kHz, 8 bits), puis freine les deux moteurs.

- **commanderMoteur()** borne la consigne entre −255 et +255, inverse le sens du moteur droit (monté tête-bêche), écrit PHASE et applique le rapport cyclique.
- **arreterMoteurs()** impose ENABLE = 0 sur les deux voies.

**Respect de l'exigence et de sa tolérance.** Les fonctions de commande comptent au plus 10 lignes et un niveau d'imbrication ; le test unitaire wheel_command_test (dossier tests) vérifie la conversion, y compris pour une entrée non finie ; testMoteur() essaie chaque moteur séparément (EXIG_CODE_TEST). **Conforme (conception).**

# 3.4 Conception détaillée de la partie énergie

## 3.4.1 Pont diviseur de tension

**Référence de conception :** CDT_BATTERIE_ADC

**Exigences client vérifiées :** EXIG_SECUR_BATT, EXIG_AUTONOMIE

### *3.4.1.1 Schéma électrique du pont diviseur de tension*

La tension batterie (jusqu'à 8,4 V) dépasse la pleine échelle de l'ADC (environ 3,1 V avec une atténuation de 11 dB) : le pont R104/R105 la ramène dans cette plage (figure 8).

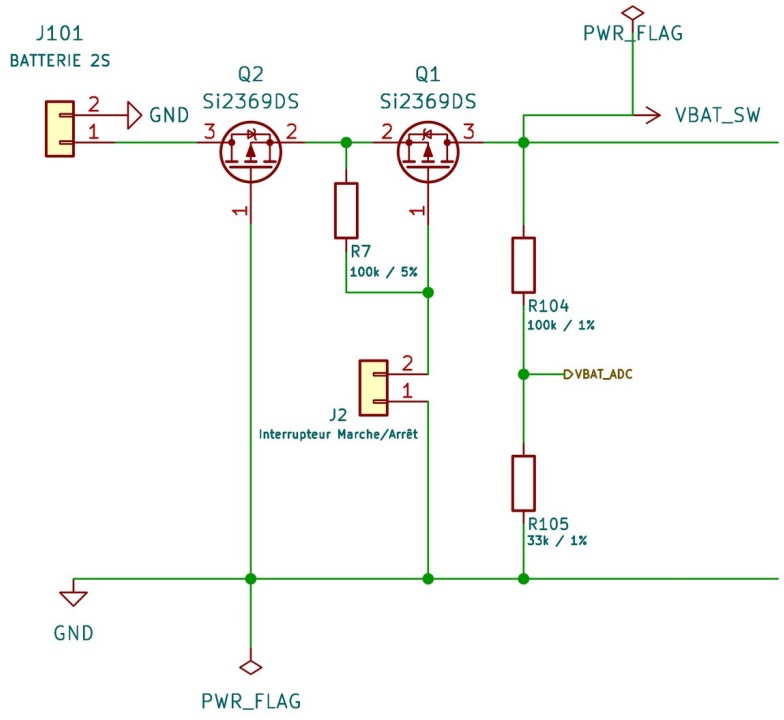



**Figure 13 :** Feuille 3 : batterie J101, protection Q2, interrupteur Q1, commande J2/R7 et pont R104/R105

$$V_{ADC} = V_{BAT} \times R_{105} / (R_{104} + R_{105}) = V_{BAT} \times 33 / 133 = 0{,}2481 \times V_{BAT}$$

8,4 V → **2,084 V** 6,6 V → **1,638 V**

Avec R104 = 100 kΩ et R105 = 33 kΩ à ±1 %, le rapport varie de 0,2444 à 0,2519 : à 6,6 V, l'entrée est comprise entre 1,613 et 1,662 V (SIM01). Un seuil fixe de 1,638 V sans étalonnage

correspondrait à une batterie entre 6,50 et 6,70 V ; le gain est donc étalonné sur une mesure faite à J101. Le courant du pont à 8,4 V vaut 8,4 / 133 kΩ = 63 µA (0,07 mAh en 65 min) et sa puissance 0,5 mW : l'effet sur l'autonomie est négligeable.

VBAT_ADC rejoint GPIO20, voie 9 de l'ADC2, qui est aussi USB_D+ au démarrage : la fonction USB n'est pas utilisée. L'ADC2 est partagé avec la radio : la mesure n'est fiable que Wi-Fi coupé, ce qui sera la règle en combat (la télémétrie Wi-Fi reste réservée aux essais). Aucun condensateur de filtrage n'est monté sur VBAT_ADC ; la moyenne logicielle de 16 mesures réduit le bruit, et la résistance de Thévenin du pont (100 kΩ ‖ 33 kΩ = 24,8 kΩ) reste compatible avec l'échantillonneur de l'ADC, dont la capacité se recharge entre deux conversions espacées de plusieurs microsecondes [S8].

**Respect de l'exigence et de sa tolérance.** EXIG_SECUR_BATT impose de couper la puissance sous 6,6 V : l'image de 6,6 V vaut 1,613 à 1,662 V avec les tolérances (SIM01) ; après étalonnage du gain sur une mesure à J101, l'erreur résiduelle est celle de l'ADC étalonné, de quelques dizaines de millivolts sur l'entrée [S8], soit environ ±0,1 V sur la batterie. Le seuil logiciel est donc fixé à 6 700 mV : la coupure intervient au plus tard quand la batterie atteint 6,6 V. La chute dans Q1 et Q2 (0,16 V à 2 A) fait lire une tension légèrement inférieure à celle de la batterie, ce qui anticipe encore l'arrêt. **Conforme (conception).**

#### *3.4.1.2 Code informatique du pont diviseur de tension*

**Source et justification de l'algorithme.** La moyenne de 16 échantillons est un filtre passe-bas simple qui divise par 4 l'écart-type du bruit de mesure ; le défaut est mémorisé (verrou) parce que la tension d'une LiPo remonte dès que la charge est coupée, ce qui relancerait sinon les moteurs en boucle. Les calculs sont faits en entiers 32 bits pour éviter le flottant dans la boucle de commande.

```
// batterie.cpp - Mesure de la tension LiPo 2S par le pont R104/R105
#include "batterie.h"

constexpr uint32_t RESISTANCE_HAUTE_OHM = 100000;  // R104
constexpr uint32_t RESISTANCE_BASSE_OHM = 33000;   // R105
constexpr uint32_t GAIN_ETALONNAGE_POUR_MILLE = 1000;  // ajusté après mesure à J101
// 6,6 V (EXIG_SECUR_BATT) + 0,1 V d'erreur de mesure après étalonnage : coupure garantie avant
6,6 V
constexpr uint16_t SEUIL_BATTERIE_FAIBLE_MV = 6700;
constexpr uint8_t NB_ECHANTILLONS_BATTERIE = 16;

void initialiserMesureBatterie(EtatBatterie &batterie) {
  analogSetPinAttenuation(PIN_VBAT_ADC, ADC_11db);  // pleine échelle ~3,1 V
  batterie.defautMemorise = false;
  batterie.tensionMv = lireTensionBatterieMv();
}

uint16_t lireTensionBatterieMv() {
  uint32_t sommeMv = 0;
  for (uint8_t i = 0; i < NB_ECHANTILLONS_BATTERIE; i++) {
    sommeMv += analogReadMilliVolts(PIN_VBAT_ADC);  // ADC2 : radio Wi-Fi coupée
  }
  uint32_t tensionAdcMv = sommeMv / NB_ECHANTILLONS_BATTERIE;
  uint32_t tensionBatterieMv = tensionAdcMv
      * (RESISTANCE_HAUTE_OHM + RESISTANCE_BASSE_OHM) / RESISTANCE_BASSE_OHM;
  return static_cast<uint16_t>(tensionBatterieMv * GAIN_ETALONNAGE_POUR_MILLE / 1000);
}

void surveillerBatterie(EtatBatterie &batterie) {
  batterie.tensionMv = lireTensionBatterieMv();
  if (batterie.tensionMv < SEUIL_BATTERIE_FAIBLE_MV) {
    batterie.defautMemorise = true;  // aucun réarmement si la tension remonte
  }
}
```

**Programme 8 :** Module batterie.cpp : mesure et surveillance de la batterie

- **initialiserMesureBatterie()** règle l'atténuation de GPIO20 à 11 dB, efface le défaut et fait une première mesure.
- **lireTensionBatterieMv()** moyenne 16 mesures étalonnées, multiplie par (R104 + R105) / R105 = 133/33 et applique le gain d'étalonnage. Les calculs en entiers 32 bits ne peuvent pas déborder ($16 \times 3100\ mV \times 133 < 2^{32}$).
- **surveillerBatterie()** mémorise le défaut dès le premier passage sous 6 700 mV (6,6 V plus la marge de mesure) et ne l'efface jamais seule : la tension d'une LiPo remonte quand les moteurs s'arrêtent, ce qui ne doit pas relancer le robot.

### 3.4.2 Régulateurs de tension

**Référence de conception :** CDT_REGULATION
**Exigences client vérifiées :** EXIG_CARTE, EXIG_AUTONOMIE, EXIG_PCB_CMS

#### *3.4.2.1 Schéma électrique des régulateurs de tension*

**Convertisseur 5 V AP63200**

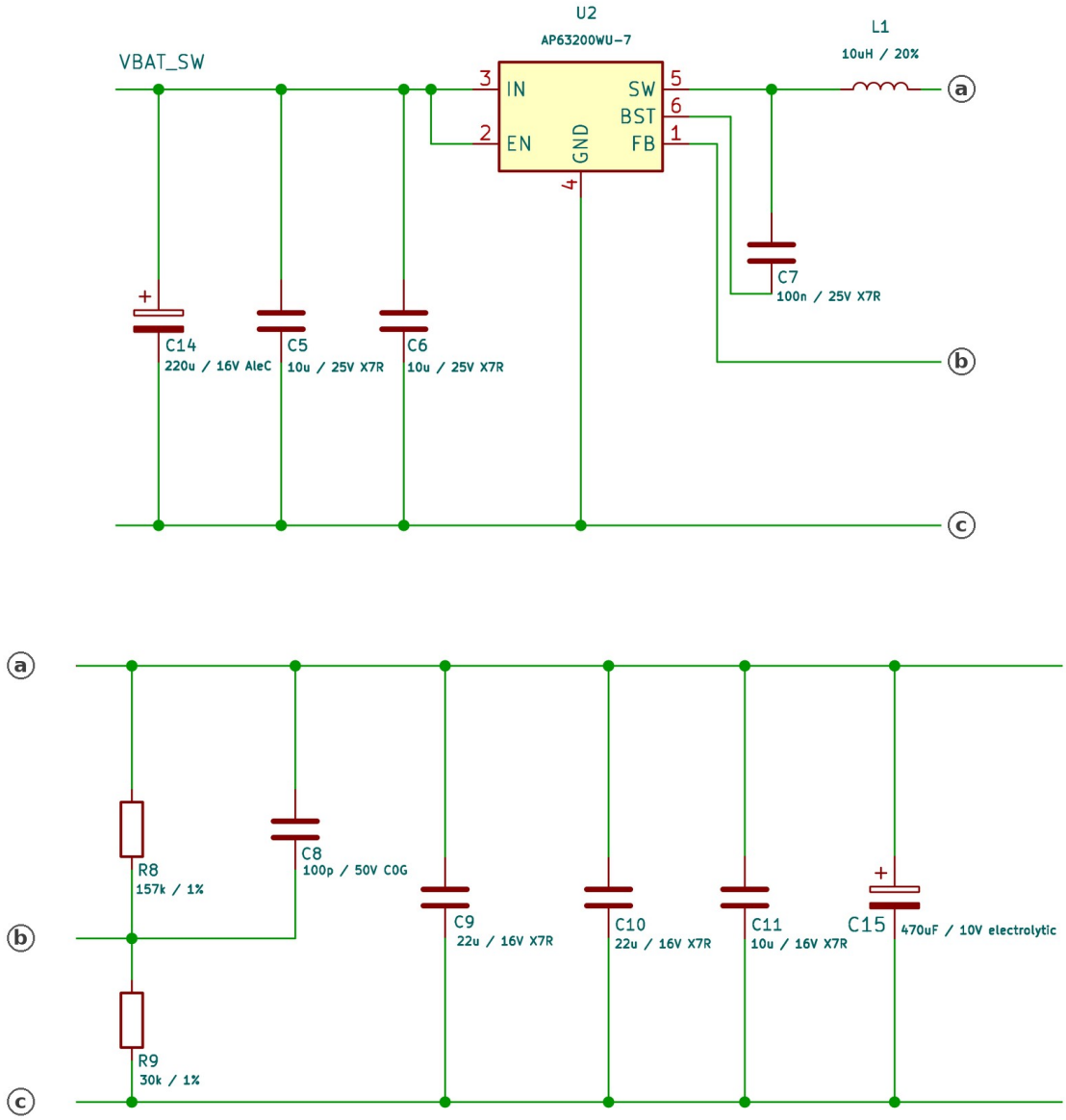



**Figure 14 :** Feuille 3 : convertisseur U2 AP63200, inductance L1, pont de retour R8/R9 et condensateurs, en deux parties reliées par les liaisons a (+5 V), b (retour FB) et c (masse)

U2 (AP63200WU-7) est alimenté par VBAT_SW (6,6 à 8,4 V, dans sa plage de 3,8 à 32 V) ; EN est relié à VIN. Les valeurs suivent la ligne « 5 V » du tableau 1 de la fiche [S1]. La tension de sortie est fixée par le pont de retour sur la référence de 0,8 V :

$$V_{OUT} = 0{,}8 \times (1 + R_8 / R_9) = 0{,}8 \times (1 + 157\ k\Omega / 30\ k\Omega) = \mathbf{4{,}987\ V}$$

Avec R8 et R9 à ±1 % et la référence à ±1 %, la sortie reste entre 4,855 et 5,122 V. L'ondulation du courant dans L1 = 10 µH à 500 kHz vaut, à 8,4 V :

$$\Delta I_L = V_{OUT} \times (1 - V_{OUT} / V_{IN}) / (L \times f) = 5 \times (1 - 5 / 8{,}4) / (10\ \mu H \times 500\ kHz) = 0{,}405\ A$$

À la charge maximale de 2 A, le courant crête est de 2,2 A, sous la limite de crête minimale de 2,5 A du convertisseur. L'inductance XAL6060-103MEC supporte 7,6 A en saturation et 5 A efficaces, avec une résistance de 29,8 mΩ au plus, soit 0,12 W de pertes à 2 A [S13]. C7 = 100 nF alimente le circuit d'amorçage (BST), C8 = 100 pF accélère la boucle de régulation, C5 et C6 (2 × 10 µF / 25 V) découplent l'entrée et C14 = 220 µF / 16 V sert de réserve côté batterie ; C9 et C10 (2 × 22 µF / 16 V) et C11 (10 µF) filtrent la sortie. Toutes les tensions de service dépassent d'au moins 90 % la tension appliquée.

Charge de travail : 1 A pour le servo et 0,5 A pour la branche 3,3 V donnent 1,5 A sur le 5 V, soit 0,5 A de réserve nominale.

**Régulateur 3,3 V LD1117S33**

U3 (LD1117S33CTR, SOT-223) est alimenté par le +5 V ; sa languette est la sortie, pas la masse (figure 3). Sa sortie garantie (3,16 à 3,44 V pour 4,75 à 10 V en entrée) est dans la plage de l'ESP32-S3 (3,0 à 3,6 V) [S2, S7]. C12 = 10 µF (minimum constructeur : 10 µF) et C4 = 220 µF stabilisent la sortie. La puissance dissipée et la température de jonction valent, avec $R_{\theta JA}$ = 110 °C/W et 40 °C dans le robot :

$$P_D = (V_{IN} - V_{OUT}) \times I = 1{,}7 \times I \qquad T_J = 40\ °C + 110\ °C/W \times P_D$$

**Tableau 23 :** Échauffement estimé de U3 selon le courant consommé

| Courant sur le 3,3 V | 200 mA | 300 mA | 400 mA | 500 mA |
|---|---|---|---|---|
| Puissance dissipée | 0,34 W | 0,51 W | 0,68 W | 0,85 W |
| Température de jonction | 77 °C | 96 °C | 115 °C | 134 °C |

La consommation prévue (188 mA, tableau 9) donne environ 75 °C, sous la limite de fonctionnement de 125 °C. Au-delà d'environ 450 mA continus, la marge disparaît : la radio Wi-Fi, qui peut appeler plusieurs centaines de milliampères, reste donc coupée en combat. Une surface de cuivre reliée à la languette réduirait encore la température.

**Respect de l'exigence et de sa tolérance.** Le +5 V reste entre 4,855 et 5,122 V, dans la plage du FT90M (3 à 8,4 V [S14]) et au-dessus de la tension minimale d'entrée de U3 (3,3 V + 1,1 V de déchet = 4,4 V) ; le +3V3 reste entre 3,16 et 3,44 V, dans la plage de l'ESP32-S3 (3,0 à 3,6 V [S7]) et des capteurs ; le courant crête de L1 (2,2 A) est sous la limite du convertisseur (2,5 A) et la jonction de U3 reste à 75 °C pour 188 mA, sous 125 °C. Les régulateurs sont CMS (EXIG_PCB_CMS). **Conforme (conception).**

### 3.4.3 Protection, marche/arrêt et coupure du servo

**Référence de conception :** CDT_PROTECTION

**Exigences client vérifiées :** EXIG_PROT_INV_POL, EXIG_INTER, EXIG_SECUR_BATT, EXIG_CARTE_VISU

#### *3.4.3.1 Schéma électrique de la protection et des coupures*

La chaîne batterie et le pont diviseur sont donnés en figure 8 (§ 3.4.1).

**Protection contre l'inversion de polarité (Q2) et marche/arrêt (Q1)**

Q1 et Q2 sont des MOSFET canal P Si2369DS (−30 V, ±20 V grille-source, 40 mΩ au plus sous $V_{GS}$ = −4,5 V) à sources communes [S3]. Q2 a son drain côté batterie et sa grille à la masse : en polarité correcte, sa diode interne conduit d'abord, la source monte à la tension batterie et $V_{GS} \approx -V_{BAT}$ rend le canal passant ; en polarité inversée, $V_{GS} \approx 0$ et la diode est bloquée. Q1 a sa grille rappelée à la source par R7 = 100 kΩ (bloqué) et tirée à la masse par l'interrupteur relié à J2 (passant).

Le courant dans l'interrupteur vaut 8,4 V / 100 kΩ = 84 µA (89 µA avec R7 à −5 %) : le courant des moteurs ne le traverse pas. Les deux transistors sont en série ; avec 40 mΩ chacun :

à 2 A : $\Delta V = 2 \times 0{,}040\ \Omega \times 2\ A = 0{,}16\ V$ $\quad P = 0{,}040 \times 2^2 = 0{,}16\ W$ par transistor

à 3 A : $\Delta V = 2 \times 0{,}040\ \Omega \times 3\ A = 0{,}24\ V$ $\quad P = 0{,}040 \times 3^2 = 0{,}36\ W$ par transistor

Le cas de 3 A correspond aux deux GM9 bloqués (1,88 A) et à 1,5 A sur le 5 V. Une diode Schottky de 0,5 V dissiperait 1,5 W à 3 A et retirerait 0,5 V à la batterie. La résistance du MOSFET augmente à chaud : l'échauffement des deux SOT-23 sera mesuré sous charge.

**Coupure du servomoteur (Q7)**

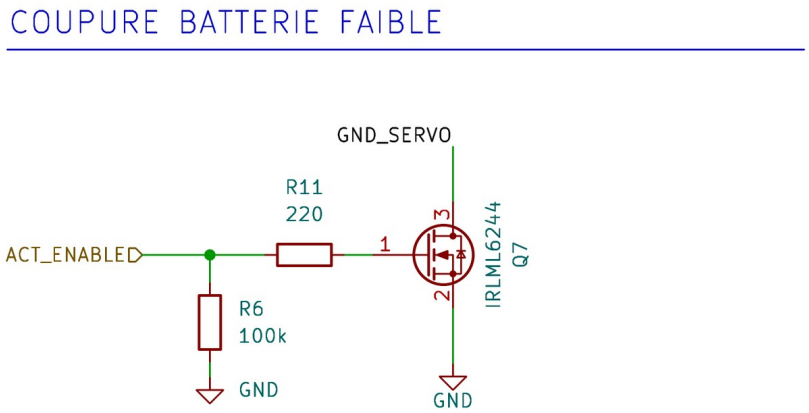



**Figure 15 :** Feuille 2 : coupure du retour de masse du servo par Q7

Q7 (IRLML6244, canal N, 27 mΩ dès $V_{GS}$ = 2,5 V [S4]) coupe GND_SERVO. Sa grille est commandée par ACT_ENABLE (GPIO11) à travers R11 = 220 Ω, qui limite le courant de charge de la grille (3,3 / 220 = 15 mA au démarrage), et rappelée à la masse par R6 = 100 kΩ : tant que la sortie est flottante (réinitialisation, démarrage), le servo n'a pas de masse. La dissipation au courant de blocage de 0,8 A vaut $0{,}8^2 \times 0{,}027 = 17$ mW. Avant de mettre ACT_ENABLE à 0, le programme arrête le signal MLI du servo, pour qu'il ne fournisse pas un retour de masse parasite.

**Voyants**

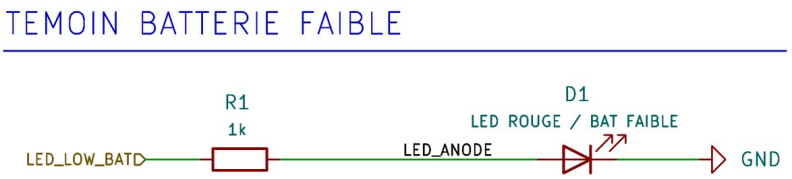



**Figure 16 :** Feuille 3 : LED de batterie faible D1 commandée par GPIO15

D1 (rouge) est commandée par GPIO15 à travers R1 = 1 kΩ : avec $V_F \approx 2$ V, I = (3,3 − 2) / 1 kΩ = 1,3 mA. D2 est reliée au +5 V par R10 = 1 kΩ : environ 3 mA, et 26,5 mW au plus dans R10, en dessous des 125 mW d'une résistance 1206.

**Respect de l'exigence et de sa tolérance.** EXIG_PROT_INV_POL : en inversion, Q2 est bloqué ($V_{GS} \approx 0$) et seul son courant de fuite circule, 1 µA au plus [S3], pour une limite de 1 mA. EXIG_INTER : l'interrupteur ne conduit que 84 µA (89 µA au pire). EXIG_SECUR_BATT : Q7 coupe la masse du servo et l'arrêt des MLI coupe les moteurs. Les deux MOSFET dissipent 0,16 W chacun à 2 A, sous les 1,25 W admis en boîtier SOT-23 à 25 °C [S3]. EXIG_CARTE_VISU : D2 sur le rail +5 V, D1 sur une sortie du microcontrôleur. **Conforme (conception).**

### *3.4.3.2 Code informatique de la coupure et du voyant*

```
// securite.cpp - Coupure de la masse du servo (Q7) et LED de batterie faible D1
#include "securite.h"

constexpr uint32_t DEMI_PERIODE_CLIGNOTEMENT_MS = 250;

void initialiserSecurite() {
  pinMode(PIN_ACT_ENABLE, OUTPUT);
  digitalWrite(PIN_ACT_ENABLE, LOW);  // Q7 bloqué : servo sans retour de masse
  pinMode(PIN_LED_BATTERIE, OUTPUT);
  digitalWrite(PIN_LED_BATTERIE, LOW);
}

void autoriserActionneurs(bool autorisation) {
  digitalWrite(PIN_ACT_ENABLE, autorisation ? HIGH : LOW);
}

void signalerDefautBatterie() {
  bool ledAllumee = (millis() / DEMI_PERIODE_CLIGNOTEMENT_MS) % 2;
  digitalWrite(PIN_LED_BATTERIE, ledAllumee ? HIGH : LOW);
}

void allumerLedEtat(bool allumee) {
  digitalWrite(PIN_LED_BATTERIE, allumee ? HIGH : LOW);
}
```

**Programme 9 :** Module securite.cpp : ACT_ENABLE et LED D1

**initialiserSecurite()** est appelée en premier : elle force ACT_ENABLE à 0 avant toute autre configuration. **autoriserActionneurs()** n'est appelée avec la valeur vraie qu'après l'ordre de départ. **signalerDefautBatterie()** fait clignoter D1 à 2 Hz à partir de millis(), sans bloquer le programme : un delay() bloquerait la surveillance des CNY70 pendant le clignotement.

# 3.5 Conception détaillée de la partie traitement

## 3.5.1 Le microcontrôleur ESP32-S3-WROOM-1-N4

**Référence de conception :** CDT_TRAITEMENT

**Exigences client vérifiées :** EXIG_CARTE, EXIG_COMPORTEMENT, EXIG_CODE_LANGAGE, EXIG_CODE_FONCTION, EXIG_CODE_IMBRICATION, EXIG_CODE_VARIABLE, EXIG_CODE_TEST

### *3.5.1.1 Schéma électrique du microcontrôleur*

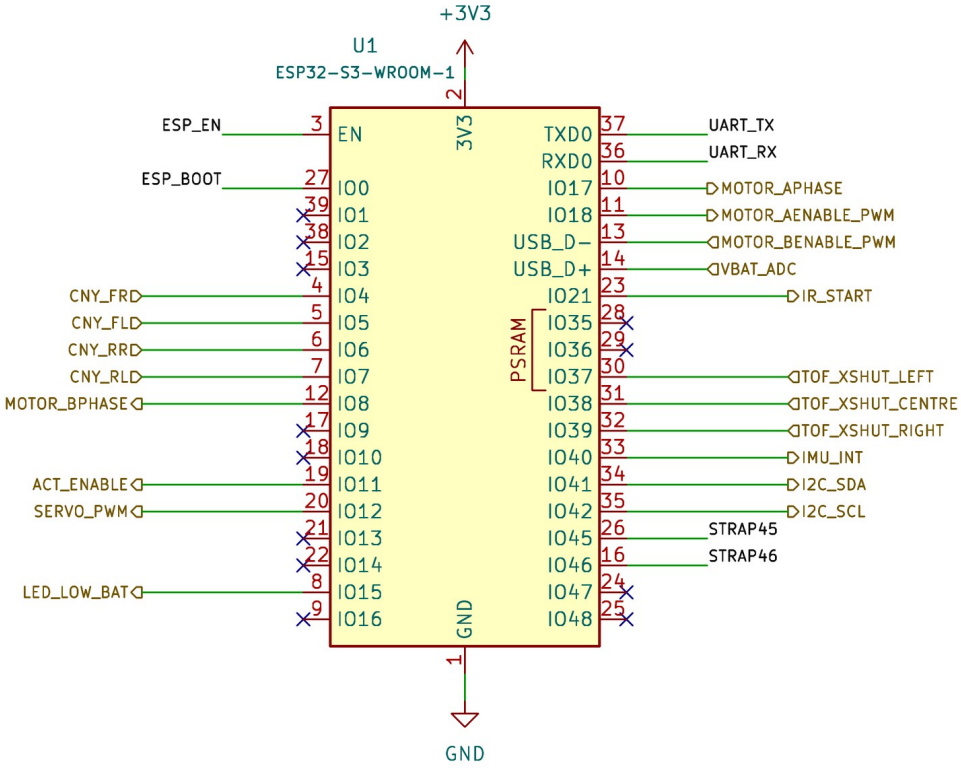



**Figure 17 :** Feuille 2 : module U1 ESP32-S3-WROOM-1 et affectation des broches

**Tableau 24 :** Affectation des broches de l'ESP32-S3

| Signal | GPIO | Type | Signal | GPIO | Type |
|---|---|---|---|---|---|
| CNY_FR / CNY_FL | 4 / 5 | ADC1 | MOTOR_APHASE / AENABLE | 17 / 18 | sortie / PWM |
| CNY_RR / CNY_RL | 6 / 7 | ADC1 | MOTOR_BPHASE / BENABLE | 8 / 19 | sortie / PWM |
| VBAT_ADC | 20 | ADC2 | SERVO_PWM | 12 | PWM 50 Hz |
| IR_START | 21 | entrée | ACT_ENABLE | 11 | sortie (Q7) |
| I2C_SDA / I2C_SCL | 41 / 42 | I²C | LED_LOW_BAT (D1) | 15 | sortie |
| XSHUT G / C / D | 37 / 38 / 39 | sorties | UART TX / RX (J13) | 43 / 44 | programmation |
| IMU_INT | 40 | entrée | STRAP45 / STRAP46 | 45 / 46 | configuration |

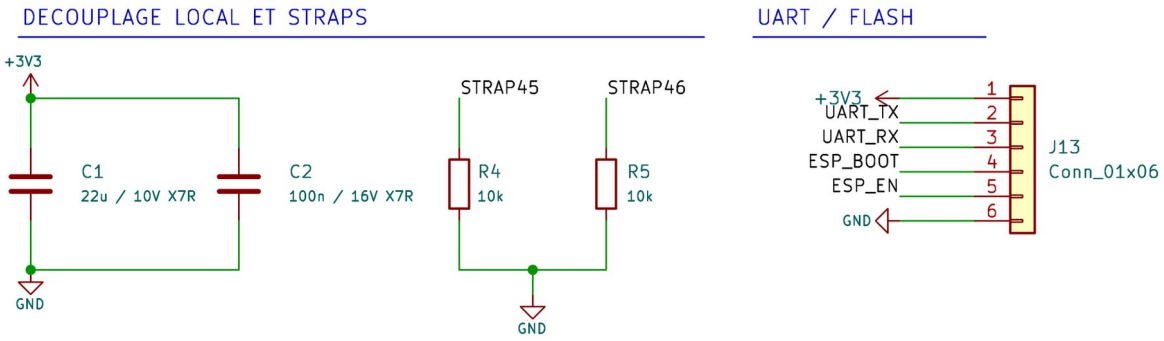



**Figure 18 :** Feuille 2 : découplage C1/C2, broches de configuration R4/R5 et connecteur de programmation UART J13

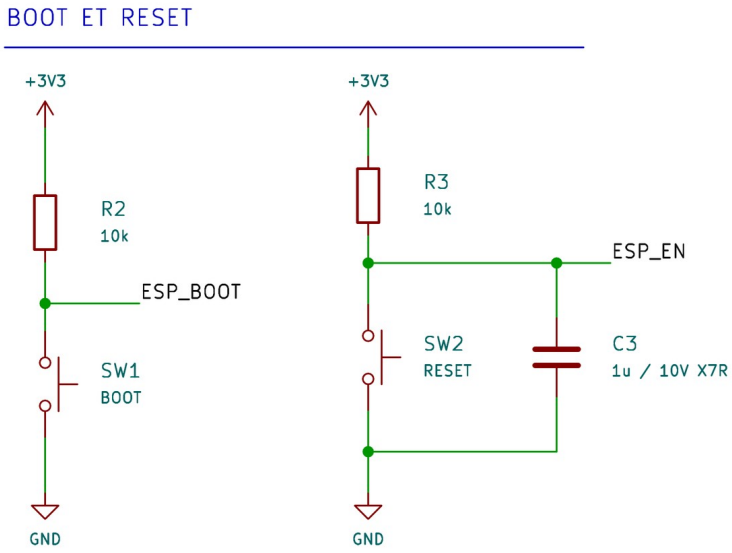



**Figure 19 :** Feuille 2 : circuits BOOT (R2, SW1) et RESET (R3, SW2, C3)

Le module est alimenté par le +3V3, découplé par C1 = 22 µF et C2 = 100 nF au plus près de sa broche 3V3. Le circuit de réinitialisation R3 = 10 kΩ / C3 = 1 µF retarde le démarrage de τ = R × C = 10 ms, le temps que l'alimentation se stabilise ; SW2 (RESET) force EN à 0. R2 = 10 kΩ tire IO0 au niveau haut (démarrage normal) ; SW1 (BOOT) le force à 0 pour la programmation. R4 et R5 = 10 kΩ fixent les broches de configuration 45 et 46. La programmation passe par J13 (UART0, adaptateur 3,3 V, TX/RX croisés) ; l'adaptateur ne doit pas alimenter le +3V3 lorsque la batterie est branchée.

Trois règles guident l'affectation : toutes les entrées des CNY70 sont sur l'ADC1, indépendant de la radio ; GPIO19 et GPIO20, broches USB au démarrage, sont réaffectées (B_ENABLE et VBAT_ADC) sans activer l'USB ; GPIO39 à GPIO42, broches JTAG, servent aux XSHUT, à l'interruption et à l'I²C sans débogage JTAG externe. Les sorties des moteurs sont reliées directement au module de puissance : l'absence d'impulsion pendant le démarrage est contrôlée roues levées lors de la mise sous tension.

**Respect de l'exigence et de sa tolérance.** EXIG_CARTE : le module ESP32-S3 est soudé sur la carte ; son alimentation reste entre 3,16 et 3,44 V, dans sa plage de 3,0 à 3,6 V ; τ = 10 ms dépasse

le temps d'établissement du +3V3 ; toutes les entrées analogiques utilisent l'ADC1 sauf VBAT_ADC, mesurée radio coupée [S7, S8]. **Conforme (conception).**

#### *3.5.1.2 Code informatique du microcontrôleur*

Le programme de la carte (programme 10) initialise d'abord la sécurité et les moteurs (ACT_ENABLE = 0, roues freinées), puis les mesures, et lance le menu des essais unitaires. L'état est local à setup() : aucune variable globale n'est déclarée.

```
// robot_sumo_s3_io.ino - Couche d'entrées/sorties ESP32-S3 et essais unitaires
// La décision (ChassisPolicy ou politique apprise) est appelée à partir de ces fonctions.
#include "batterie.h"
#include "depart.h"
#include "moteurs.h"
#include "securite.h"
#include "sol.h"
#include "tests.h"

void setup() {
  initialiserSecurite();   // ACT_ENABLE = 0 avant toute autre configuration
  initialiserMoteurs();    // ENABLE = 0 : roues freinées
  Serial.begin(VITESSE_LIAISON_SERIE);
  EtatBatterie batterie;
  initialiserMesureBatterie(batterie);
  initialiserCapteursSol();
  initialiserRecepteurIr();
  while (true) {
    executerMenuTests();
  }
}

void loop() {
  // Non utilisé : aucune variable globale, la boucle s'exécute dans setup().
}
```

**Programme 10 :** Programme principal de la carte ESP32-S3

**Inférence de la politique apprise**

**Source et justification de l'algorithme.** L'algorithme d'apprentissage est PPO (Proximal Policy Optimization [S25]), précédé d'une imitation par DAgger [S26]. PPO est retenu parce que sa mise à jour bornée (rapport de probabilités écrêté) reste stable avec des récompenses rares comme la victoire ; il s'applique à des actions continues, ici deux rapports cycliques. Le réseau embarqué est un perceptron multicouche : son calcul est une suite fixe de produits matrice-vecteur, sans allocation dynamique ni branchement dépendant des données, donc de durée constante et prévisible sur le microcontrôleur. La fonction tanh borne naturellement les sorties entre −1 et 1, le domaine des rapports cycliques signés.

L'inférence (programme 11) normalise les 21 mesures, les concatène aux neuf trames précédentes et à la dernière commande (212 entrées), puis les fait traverser deux couches de 64 neurones et une couche de sortie à tanh. Elle demande environ 18 000 multiplications-additions et 130 tangentes hyperboliques par décision : à quelques cycles par opération sur l'unité de calcul flottant de l'ESP32-S3 à 240 MHz, sa durée est de l'ordre de la milliseconde, soit environ 1 % de la période de décision de 100 ms ; elle sera mesurée sur la carte. Les 17 922 poids (72 ko, constantes en mémoire flash) occupent 1,8 % des 4 Mo de flash.

```
// Extrait de ml/runtime/actor_runtime.cpp (inférence embarquée, sans allocation)
std::array<float, 2> SensorPolicy::step(const double raw[kObservationSize]) {
  const auto current = normalize(raw);                  // 21 mesures normalisées
  std::array<float, kInputSize> input{};
  int offset = 0;
  for (float value : current) input[offset++] = value;
```

```
  for (const auto& frame : history_)                    // 9 trames précédentes
    for (float value : frame) input[offset++] = value;
  for (float value : lastAction_) input[offset++] = value;

  if (raw[0] - lastHistoryAt_ >= .095) {                // une trame toutes les 100 ms
    for (int index = kHistoryFrames - 2; index > 0; --index)
      history_[index] = history_[index - 1];
    history_[0] = current;
    lastHistoryAt_ = raw[0];
  }

  std::array<float, kHiddenSize> first{}, second{};
  std::array<float, 2> action{};
  denseTanh<kHiddenSize, kInputSize>(weights_.firstWeight,
      weights_.firstBias, input.data(), first.data());
  denseTanh<kHiddenSize, kHiddenSize>(weights_.secondWeight,
      weights_.secondBias, first.data(), second.data());
  denseTanh<2, kHiddenSize>(weights_.outputWeight,
      weights_.outputBias, second.data(), action.data());
  lastAction_ = action;
  return action;                                        // rapports cycliques gauche/droite
}
```

**Programme 11 :** Inférence de la politique apprise (actor_runtime.cpp)

**Respect de l'exigence et de sa tolérance.** EXIG_COMPORTEMENT : le réseau change la direction du robot à partir des seules mesures des capteurs et agit sur les deux moteurs ; l'inférence C++ reproduit PyTorch à $4{,}8 \times 10^{-7}$ près (SIM04). Les trois fonctions d'actor_runtime.cpp comptent au plus 27 lignes et un niveau d'imbrication. **Conforme (conception).**

### Essais unitaires et règles de codage

Le module tests regroupe une fonction d'essai par unité matérielle (EXIG_CODE_TEST), appelée depuis un menu sur la liaison série. Les règles de codage ont été contrôlées automatiquement sur tout le code embarqué écrit par l'équipe (tableau 25) : couche d'entrées/sorties, suivi de cible (Tracker, Observation) et inférence ; les bibliothèques tierces (pilote ST, IRremote) en sont exclues comme le permet le [CDC].

**Tableau 25 :** Contrôle automatique des règles de codage du code embarqué

| Règle du CDC | Limite | Code embarqué (97 fonctions) | Résultat |
|---|---|---|---|
| EXIG_CODE_FONCTION : lignes par fonction | 60 | 48 au plus (Tracker::makeCluster) | Conforme |
| EXIG_CODE_IMBRICATION : niveaux | 3 | 3 au plus ; 2 dans la couche E/S | Conforme |
| EXIG_CODE_VARIABLE : variables globales | interruptions seulement | aucune | Conforme |
| EXIG_CODE_TEST : une fonction par unité | toutes les unités | 8 fonctions d'essai (9 unités) | Conforme |
| Compilation | sans erreur | g++ -Wall -Wextra : 0 avertissement | Conforme |

Deux fonctions du suivi de cible dépassaient la limite de 60 lignes : Tracker::updateTracking (98 lignes) et ObservationLog::record (69 lignes). Elles ont été découpées le 13/11/2026 en sous-fonctions nommées par étape (noteTargetSector, turnTowardGuard, extendProbe, seekCentreOverlap ; copyMatch, detectCrossings, addCrossing). Le comportement est inchangé : les 55 scénarios du test du suivi passent, et 102 combats complets rejoués avant et après le découpage donnent des traces identiques à l'octet près.

**Respect de l'exigence et de sa tolérance.** Les quatre règles de codage du [CDC] sont respectées par l'ensemble du code embarqué ; chaque unité matérielle dispose de sa fonction d'essai. **Conforme (conception).**

# 4. Dérisquage des solutions techniques retenues

Ce chapitre détaille les activités de dérisquage des solutions techniques retenues : simulation et/ou prototypage rapide. Il constitue une preuve partielle de la conformité du produit. Chaque paragraphe fait clairement référence aux exigences client issues du [CDC].

Il permet également de confirmer les résultats théoriques des chapitres 2 et 3. Chaque fiche précise le périmètre et la technique du dérisquage, son protocole, le résultat attendu déduit de l'exigence et de sa tolérance, le résultat obtenu, la conformité et, pour chaque écart rencontré, sa cause-racine. Les outils sont ngspice pour les circuits, une Arduino UNO pour les prototypages rapides, MATLAB puis un simulateur natif C++ pour les combats, PyTorch pour l'apprentissage et le compilateur g++ pour rejouer le code embarqué réel sur PC.

L'ensemble des fichiers est disponible dans le dépôt Git du projet « robot sumo » (dossiers « simulations », « Matlab simulations », « native_sim », « ml », « tests » et « outputs »), copié dans le Google Drive partagé de l'équipe.

**Tableau 26 :** Activités de dérisquage

| Réf. | Solution dérisquée | Technique | Exigences |
|---|---|---|---|
| SIM01 | Pont diviseur de mesure batterie | simulation ngspice | EXIG_SECUR_BATT |
| SIM02 | Pont en H DRV8835 et GM9 | prototypage rapide | EXIG_DEPLACEMENT, EXIG_AUTONOMIE |
| SIM03 | Détection de bordure CNY70 | prototypage rapide | EXIG_DEPLACEMENT, EXIG_LUMINOSITE |
| SIM04 | Politique de décision apprise | simulation (369 600 combats) | EXIG_COMPORTEMENT, EXIG_DEPLACEMENT, EXIG_ADVERSAIRE |
| SIM05 | Contrôleur de référence | simulation (76 800 combats) | EXIG_DEPLACEMENT, EXIG_COMPORTEMENT |
| SIM06 | Suivi de cible de la tourelle | simulation du code réel | EXIG_ADVERSAIRE |
| SIM07 | Réception de l'ordre de départ | simulation du code réel | EXIG_DEPART |
| SIM08 | Schéma et routage | contrôles ERC/DRC KiCad | EXIG_SCH_*, EXIG_PCB_* |

## 4.1 Mesure de la tension batterie (simulation)

**Référence de la simulation :** SIM01

**Exigences client vérifiées :** EXIG_SECUR_BATT, EXIG_AUTONOMIE

**Périmètre et technique :** pont R104/R105 et entrée ADC de GPIO20 ; simulation électrique ngspice avec les cas extrêmes de tolérance.

**But de l'essai :** vérifier le rapport du pont R104/R105 et l'effet des tolérances de ±1 % sur l'image du seuil de 6,6 V.

**Fichiers :** simulations/sim01_pont_diviseur.cir (ngspice), simulations/plot.py.

**Protocole :**

Le pont de la feuille alimentation est saisi tel quel, l'entrée de l'ADC étant considérée sans courant. Une analyse continue fait varier VBAT de 6,0 à 8,4 V. Elle est répétée pour les valeurs nominales et pour les deux cas extrêmes de tolérance (R104 à +1 % et R105 à −1 %, puis l'inverse) ; la sonde est placée sur VBAT_ADC.

**Résultats attendus :**

| Grandeur | Valeur attendue (exigence) | Tolérance |
|---|---|---|
| $V_{ADC}$ pour VBAT = 8,4 V (pleine charge) | 2,084 V, sous la pleine échelle de 3,1 V | ± 1,5 % (résistances à 1 %) |
| $V_{ADC}$ pour VBAT = 6,6 V (seuil EXIG_SECUR_BATT) | 1,638 V | ± 1,5 % (résistances à 1 %) |

**Résultats obtenus :**

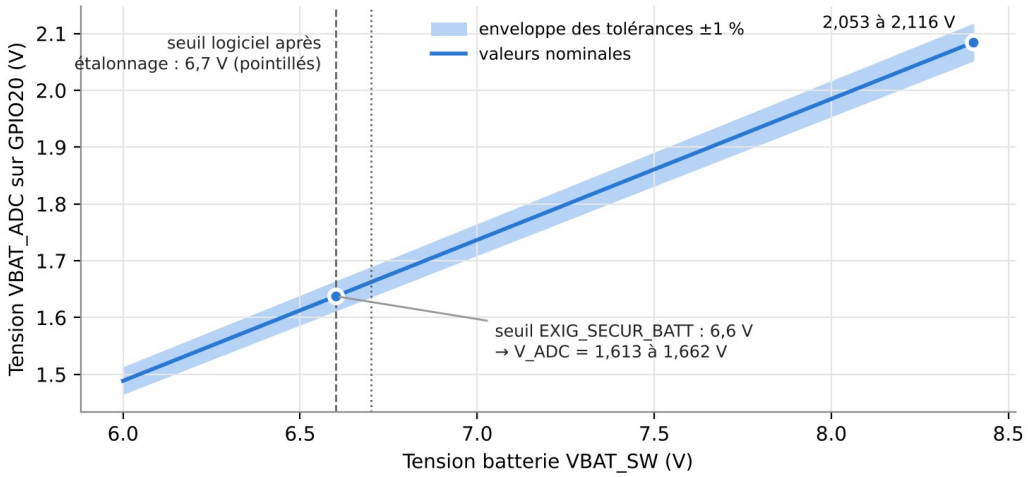



**Figure 20 :** SIM01 : tension VBAT_ADC en fonction de la tension batterie, avec l'enveloppe des tolérances

La caractéristique est une droite de pente 33/133 = 0,248 (figure 20). À 6,6 V, l'entrée varie de 1,613 à 1,662 V selon les tolérances, exactement comme le calcul du § 3.4.1.

| Grandeur | Valeur obtenue | Conformité |
|---|---|---|
| $V_{ADC}$ pour VBAT = 8,4 V | 2,053 à 2,116 V (−1,5 % / +1,5 %) | Conforme |
| $V_{ADC}$ pour VBAT = 6,6 V | 1,613 à 1,662 V (−1,5 % / +1,5 %) | Conforme |

**Statut de l'essai :** Conforme.

**Écarts rencontrés et causes-racines :**

Aucune non-conformité. La simulation ne contient pas l'erreur propre de l'ADC2 ni la chute dans Q1/Q2 : c'est pourquoi le gain est étalonné sur une mesure faite à J101 et le seuil logiciel fixé à 6,7 V, afin que la coupure intervienne au plus tard à 6,6 V (§ 3.4.1).

## 4.2 Commande des motoréducteurs par le DRV8835 (prototypage rapide)

**Référence de la simulation :** SIM02

**Exigences client vérifiées :** EXIG_DEPLACEMENT, EXIG_AUTONOMIE

**Périmètre et technique :** module Pololu 2135 et deux GM9 ; prototypage rapide sur Arduino UNO.

**But de l'essai :** vérifier que le module commande les deux GM9 dans les deux sens et à vitesse variable, et mesurer la consommation des moteurs.

**Fichiers :** programme d'essai des fonctions marche(), arriere(), gauche() et droite() (Arduino UNO) ; relevés de courant.

**Protocole :**

Le module est câblé en mode PHASE/ENABLE (MODE au niveau haut) sur une Arduino UNO, avec deux sorties MLI et deux sorties de sens, les moteurs étant alimentés sous 6,4 V. Les quatre mouvements sont enchaînés et le courant des moteurs est relevé à l'ampèremètre en roulage libre, puis robot bloqué contre un obstacle.

**Résultats attendus :**

| Grandeur | Valeur attendue (exigence) | Tolérance |
|---|---|---|
| Déplacements avant, arrière, rotation gauche et droite (EXIG_DEPLACEMENT) | réalisés | — |
| Courant par moteur en blocage | ≤ 1,2 A (limite continue du module) | — |

**Résultats obtenus :**

Les quatre déplacements sont réalisés et la vitesse varie avec le rapport cyclique.

| Grandeur | Valeur obtenue | Conformité |
|---|---|---|
| Déplacements avant, arrière, gauche, droite | réalisés | Conforme |
| Courant moyen des deux moteurs (roulage) | 300 mA | — (donnée du bilan) |
| Courant total en poussée contre un obstacle | 670 mA (335 mA par moteur) | Conforme |

**Statut de l'essai :** Conforme.

**Écarts rencontrés et causes-racines :**

Aucune non-conformité. L'essai a été fait sous 6,4 V : sous 8,4 V, le constructeur donne jusqu'à 0,94 A de blocage par moteur (§ 3.3.1), toujours sous la limite de 1,2 A. Le programme d'essai a été porté sur l'ESP32-S3 (moteurs.cpp, testMoteur()).

## 4.3 Détection de la bordure par CNY70 (prototypage rapide)

**Référence de la simulation :** SIM03

**Exigences client vérifiées :** EXIG_DEPLACEMENT, EXIG_LUMINOSITE

**Périmètre et technique :** un CNY70 câblé comme sur la carte (220 Ω, 10 kΩ) ; prototypage rapide sur Arduino UNO et échantillon de dohyo.

**But de l'essai :** vérifier qu'un CNY70 distingue nettement le noir du dohyo de sa bordure blanche et fixer un seuil.

**Fichiers :** programme Arduino de lecture des CNY70 ; relevés de mesures.

**Protocole :**

Un CNY70 est câblé sur plaque d'essai avec la LED à travers 220 Ω et le collecteur tiré par 10 kΩ (montage de la carte), relié à une entrée analogique d'une Arduino UNO (10 bits, 5 V). Le capteur est déplacé à hauteur constante au-dessus d'un échantillon de dohyo ; les valeurs sont relevées au-dessus du noir puis du blanc, sous l'éclairage de la salle.

**Résultats attendus :**

| Grandeur | Valeur attendue (exigence) | Tolérance |
|---|---|---|
| Écart de tension entre noir et blanc (EXIG_DEPLACEMENT) | > 1 V | — |
| Recouvrement des plages noir / blanc | aucun | — |

**Résultats obtenus :**

Les relevés sont donnés au tableau 21 : 2,44 à 2,93 V sur le noir et 0 à 0,98 V sur le blanc. Transposés sous 3,3 V (§ 3.2.2), ils donnent au moins 1,89 V sur le noir et au plus 1,10 V sur le blanc.

| Grandeur | Valeur obtenue | Conformité |
|---|---|---|
| Écart minimal entre noir et blanc (5 V) | 1,46 V | Conforme |
| Recouvrement des plages | aucun | Conforme |
| Marge autour du seuil de 1,5 V (3,3 V) | 0,39 V de chaque côté | Conforme |

**Statut de l'essai :** Conforme.

**Écarts rencontrés et causes-racines :**

Aucune non-conformité. Le prototype est alimenté en 5 V alors que la carte alimente les CNY70 en 3,3 V : les niveaux ont été transposés par le calcul (§ 3.2.2). La mesure des quatre capteurs à leur hauteur réelle, sous éclairage intense et avec flashs, fait partie des essais de réception (testCapteursSol()).

## 4.4 Entraînement et validation de la politique de décision (simulation)

**Référence de la simulation :** SIM04

**Exigences client vérifiées :** EXIG_COMPORTEMENT, EXIG_DEPLACEMENT, EXIG_ADVERSAIRE

**Périmètre et technique :** réseau de neurones de décision des roues ; apprentissage par renforcement (PPO) dans le simulateur de combat, validation sur des combats jamais vus à l'entraînement.

**But de l'essai :** entraîner le réseau qui décide des rapports cycliques des roues, suivre sa progression sur toute la durée de l'entraînement et retenir le meilleur réseau qui fait au moins aussi bien que le contrôleur de référence.

**Fichiers :** ml/train.py, ml/model.py, ml/export_actor.py, ml/runtime/actor_runtime.cpp ; statistiques de l'entraînement dans ml/stats/unified-20261112-until98 (summary.json, evaluation_progression.csv, behavior_summary.csv, figures).

**Protocole :**

L'entraînement « unified-20261112-until98 » a été lancé le 12/11/2026 sur un PC à carte graphique (processus de simulation en parallèle, mises à jour du réseau sur CUDA). Le simulateur oppose notre robot à une population de **onze comportements adverses** tirés à parts égales (agressif, passif, chargeur, fuyard, embusqué, tourneur, contre-attaquant, zigzagueur, avance intermittente, flanqueur, pelle) ; le comportement « pelle » modélise un adversaire à lame avant (contact « pelle »). Le réseau part de l'imitation du contrôleur de référence (DAgger), puis est optimisé par PPO : chaque itération joue 176 nouveaux combats de 60 s (16 graines par comportement).

Toutes les cinq itérations, le réseau est évalué avec des actions déterministes sur deux jeux de 396 combats (36 graines par comportement) : un jeu **fixe**, identique à chaque évaluation, et un jeu **renouvelé**, dont les graines changent à chaque fois. Chaque combat est rejoué par le contrôleur de référence sur la même graine. Le résultat retenu est le plus faible des deux taux de victoire ; le réseau qui obtient le meilleur score est conservé comme **champion**. Les valeurs ci-dessous sont celles de l'instantané du 13/11/2026 à 23 h 39, après 26,8 h d'entraînement.

**Résultats attendus :**

| Grandeur | Valeur attendue (exigence) | Tolérance |
|---|---|---|
| Porte d'acceptation, sur les jeux fixe et renouvelé : fitness, victoires, sorties du dohyo et sorties sans contact comparées au contrôleur de référence (EXIG_COMPORTEMENT, EXIG_DEPLACEMENT) | fitness supérieure ; victoires ≥ ; sorties ≤ ; sorties sans contact ≤ | aucune |
| Progression du taux de victoire sur la population (EXIG_ADVERSAIRE) | champion meilleur que le réseau de départ | écart > intervalle de mesure |
| Écart entre inférence C++ et PyTorch | $< 10^{-5}$ | — |

**Résultats obtenus :**

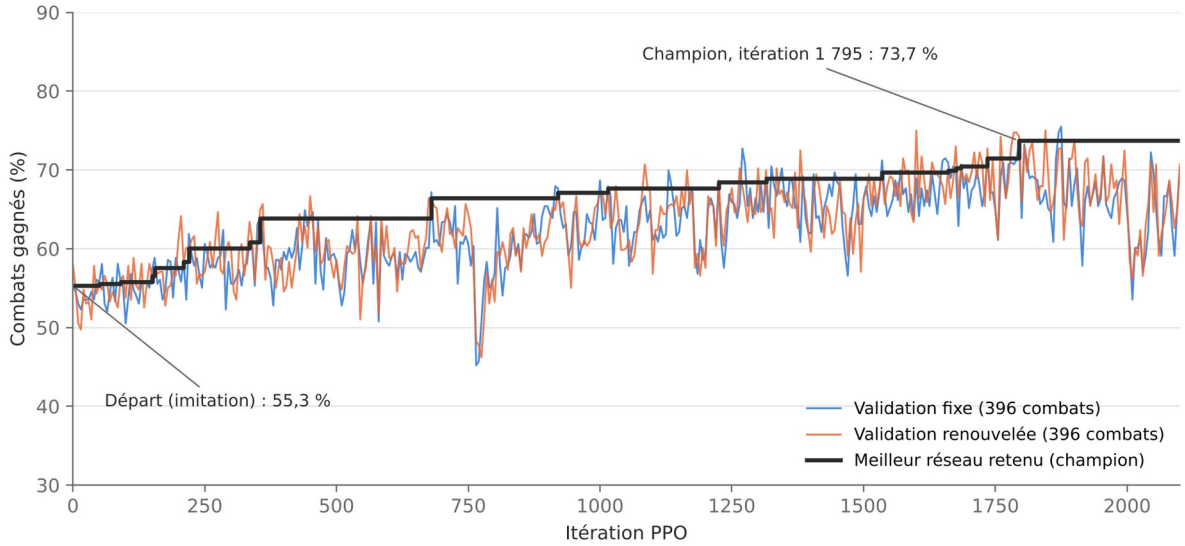



**Figure 21 :** SIM04 : progression du taux de victoire au cours de l'entraînement (jeux de validation fixe et renouvelé, champion retenu)

La figure 21 montre la progression sur 2 100 itérations, soit 369 600 combats d'entraînement et 421 évaluations. Le réseau de départ, issu de l'imitation, gagne 55,3 % des combats du jeu fixe. Le champion, obtenu à l'itération 1 795 après 23,1 h, en gagne **73,7 %** sur le jeu fixe et 74,2 % sur le jeu renouvelé, soit **+18,4 points**. Les deux courbes restent proches l'une de l'autre : le réseau ne s'est pas spécialisé sur les graines du jeu fixe. Elles fluctuent de quelques points d'une évaluation à l'autre, ce qui est attendu pour 396 combats (intervalle de confiance à 95 % d'environ ±4,3 points autour de 74 %). Un creux est visible vers l'itération 760, puis rattrapé : c'est pourquoi le réseau retenu est le champion des validations, et non le dernier réseau entraîné (70,2 % à l'itération 2 100).

**Tableau 27 :** SIM04 : résultats de validation au départ, pour le champion et à la dernière itération

| Grandeur | Départ (it. 0) | Champion (it. 1 795) | Dernier (it. 2 100) |
|---|---|---|---|
| Victoires, jeu fixe | 219 / 396 (55,3 %) | 292 / 396 (73,7 %) | 278 / 396 (70,2 %) |
| Victoires, jeu renouvelé | 229 / 396 (57,8 %) | 294 / 396 (74,2 %) | 280 / 396 (70,7 %) |
| Sorties du dohyo, fixe / renouvelé | 126 / 106 | 80 / 78 | 100 / 93 |
| Fitness moyenne, fixe / renouvelé | −91,8 / −73,1 | −28,4 / −23,9 | −41,9 / −39,4 |
| Porte d'acceptation | passée | passée | passée |

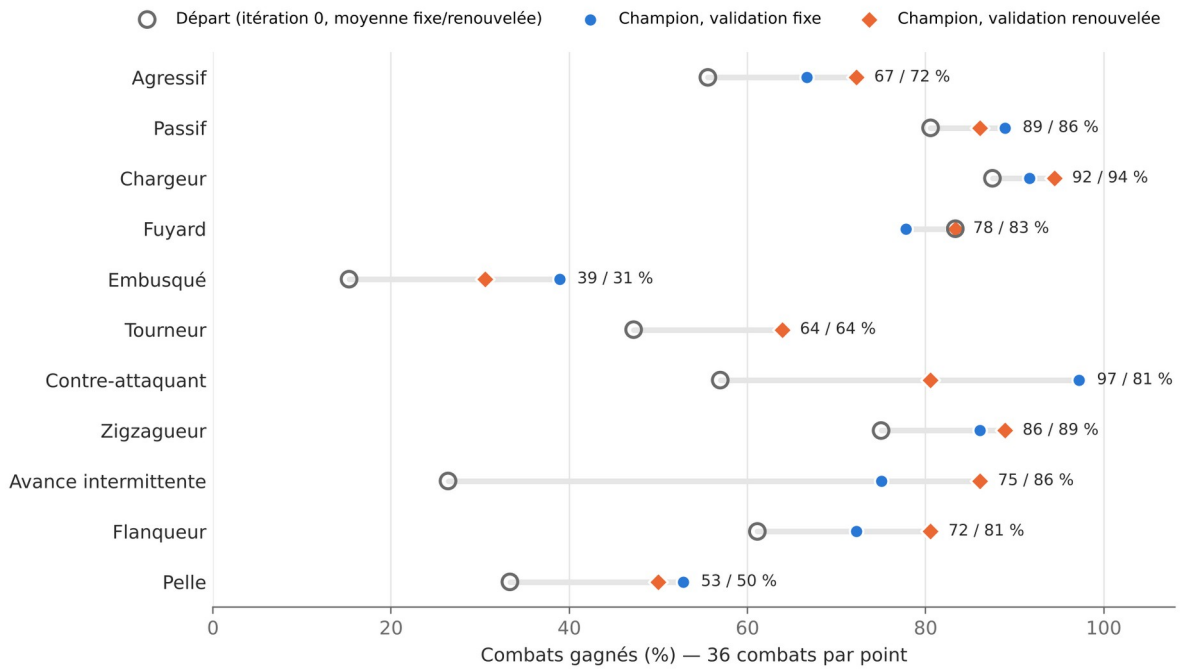



**Figure 22 :** SIM04 : victoires du champion par comportement adverse, comparées au réseau de départ

La figure 22 détaille le champion par comportement. Il progresse fortement contre l'avance intermittente (de 26 % à 75–86 %), le contre-attaquant (de 57 % à 81–97 %), l'embusqué (de 15 % à 31–39 %) et la pelle (de 33 % à 50–53 %). Contre le fuyard, il reste au niveau du départ (78–83 % contre 83 %). L'embusqué, qui attend avant de s'engager, et la pelle, dont la lame avant est favorisée dans les contacts de face, restent les deux adversaires les plus difficiles. Les noms de comportements ne servent qu'au diagnostic : le réseau ne les reçoit pas et la sélection porte sur l'ensemble de la population.

| Grandeur | Valeur obtenue | Conformité |
|---|---|---|
| Porte d'acceptation du champion (jeux fixe et renouvelé) | passée : fitness −28,4 et −23,9 ; victoires au moins égales, sorties et sorties sans contact au plus égales à celles du contrôleur de référence | Conforme |
| Progression du champion par rapport au départ | +18,4 points (55,3 → 73,7 %), supérieure à l'intervalle de ±4,3 points | Conforme |
| Écart C++ / PyTorch sur 300 pas (même architecture) | $4{,}8 \times 10^{-7}$ | Conforme |

**Statut de l'essai :** Conforme.

**Écarts rencontrés et causes-racines :**

**Régression d'un réseau à l'autre** (creux vers l'itération 760, baisse de 73,7 à 70,2 % en fin d'instantané). Cause-racine : PPO améliore la récompense moyenne des combats d'entraînement, pas directement le taux de victoire de validation ; une mise à jour peut dégrader un comportement minoritaire. La même cause avait été observée sur le premier simulateur. Traitement : le réseau embarqué est toujours le champion des validations, qui doit en plus passer la porte d'acceptation.

**Embusqué et pelle** : 31 à 53 % de victoires. Cause-racine : ces deux comportements donnent peu d'indices au télémètre avant le contact (cible immobile, puis engagement brusque ou lame basse), et le simulateur révisé du 05/11/2026, qui introduit la pelle, est plus exigeant que le premier (huit familles d'adversaires). Les taux du premier simulateur ne sont donc pas comparables à ceux-ci. L'entraînement continue ; un nouveau champion ne remplacera le réseau retenu que s'il passe la porte d'acceptation.

Le critère d'arrêt automatique de l'entraînement (98 % sur les deux jeux) est un objectif d'optimisation de l'équipe, non une exigence du [CDC] ; il n'est pas atteint. Ces résultats restent des résultats de simulation : la détection CNY70 et la dispersion des moteurs y sont idéalisées ; le comportement sur le robot est vérifié par les essais de réception. L'inférence C++ a été comparée à PyTorch sur un export précédent de la même architecture ; la comparaison sera rejouée à l'export du champion.

## 4.5 Simulation des combats avec le contrôleur de référence

**Référence de la simulation :** SIM05

**Exigences client vérifiées :** EXIG_DEPLACEMENT, EXIG_COMPORTEMENT

**Périmètre et technique :** contrôleur à états ChassisPolicy (enseignant de l'apprentissage) ; simulations MATLAB puis C++ de grands lots de combats appariés.

**But de l'essai :** mettre au point le contrôleur à états et mesurer l'effet de chaque règle sur de grands lots de combats appariés.

**Fichiers :** Matlab simulations/ (sumo_simulate.m, sumo_compare.m, batch_report_2026-10-28.md) ; native_sim/ (sim.cpp, main.cpp, RESULTS.md).

**Protocole :**

Le modèle MATLAB sépare la vérité du monde des observations : dohyo de 770 mm avec bande blanche de 25 mm, deux robots de 500 g, moteurs GM9 (78 tr/min et 0,7 A de blocage à 6 V, roues de 69 mm), cônes ToF en 3D, IMU, CNY70. Il compile par MEX le vrai code C++ du suivi de tourelle et de la commande des roues. Le simulateur natif C++ porte ce modèle pour jouer des dizaines de milliers de combats.

Chaque modification du contrôleur est comparée à la version précédente sur les mêmes graines (combats appariés de 60 s, 6 400 combats par groupe d'adversaires et par graine), une graine étant réservée à la validation.

**Résultats attendus :**

| Grandeur | Valeur attendue (exigence) | Tolérance |
|---|---|---|
| Sorties du dohyo sans contact (EXIG_DEPLACEMENT) | 0 | aucune |
| Effet d'une règle ajoutée | gain de victoires sans hausse des sorties | — |

**Résultats obtenus :**

Sur 76 800 combats appariés, la règle d'attaque rapprochée (cible à 8 cm, aucun CNY depuis 0,75 s) fait passer les victoires de 32 104 à 32 264 et les sorties de dohyo de 33 698 à 33 582, sans

aucune sortie sans contact. Une variante de poursuite plus large gagnait 1 745 combats mais ajoutait 1 360 sorties : elle a été rejetée.

| Grandeur | Valeur obtenue | Conformité |
|---|---|---|
| Sorties du dohyo sans contact (76 800 combats) | 0 | Conforme |
| Taux de victoire du contrôleur de référence | 42,0 % | — (référence de SIM04) |
| Règles retenues | gain net, sorties en baisse | Conforme |

**Statut de l'essai :** Conforme.

**Écarts rencontrés et causes-racines :**

Le lot MATLAB du 28/10/2026 a montré qu'un contact frontal adverse pouvait être pris pour un contact de flanc. Cause-racine : la direction de l'adversaire était estimée avec trop peu d'observations ; le seuil de confiance a été relevé. L'estimation de la position dans le dohyo par intégration du gyromètre dérive sur les longs combats (pas de codeur) : elle n'autorise jamais seule une poursuite vers le bord. La performance de combat limitée (42 %) a conduit à la politique apprise (SIM04).

## 4.6 Suivi de cible de la tourelle (simulation du code réel)

**Référence de la simulation :** SIM06

**Exigences client vérifiées :** EXIG_ADVERSAIRE, EXIG_LUMINOSITE

**Périmètre et technique :** classes Tracker et Observation embarquées ; exécution du code réel sur PC avec des trames de télémètres synthétiques.

**But de l'essai :** vérifier que le suivi détecte un adversaire à 250 mm, ignore l'arrière-plan et le dohyo vide, et oriente la tourelle vers une cible qui se déplace.

**Fichiers :** simulations/sim06_tourelle/ (sim06_adversaire_250mm.cpp, résultats) ; tests/esp32_turret_tracker_test.cpp (55 scénarios).

**Protocole :**

Le code réel de Tracker.cpp et Observation.cpp est compilé sur PC avec g++. Le suivi est d'abord calibré sur un dohyo vide (116 vues), puis reçoit des trames de trois télémètres décrivant chaque situation : adversaire à 250 mm vu par un ou trois capteurs, avec ou sans mur à 1500 mm hors du dohyo, adversaire qui glisse vers la gauche, dohyo vide pendant 7 s. La suite de 55 scénarios de l'équipe couvre en plus les pertes de cible, les ambiguïtés, les butées de la tourelle et le débordement de l'horloge.

**Résultats attendus :**

| Grandeur | Valeur attendue (exigence) | Tolérance |
|---|---|---|
| Acquisition d'un adversaire à 250 mm (EXIG_ADVERSAIRE) | en 4 vues au plus (≤ 0,4 s) | distance ± 1 mm |
| Retour hors dohyo (mur à 1500 mm) | ignoré | — |
| Dohyo vide | aucune fausse cible | — |
| Scénarios de la suite de tests | 55 / 55 | — |

**Résultats obtenus :**

Les cinq cas liés à l'exigence et les 55 scénarios passent. Le découpage du 13/11/2026 de deux fonctions (§ 3.5.1.2) a été vérifié en rejouant 102 combats complets avant et après : les traces sont identiques.

| Grandeur | Valeur obtenue | Conformité |
|---|---|---|
| Acquisition à 250 mm (3 capteurs, capteur central seul) | 2 vues, 250 mm | Conforme |
| Mur à 1500 mm | ignoré, cible à 250 mm retenue | Conforme |
| Dohyo vide pendant 7 s | aucune cible (mode SEARCH) | Conforme |
| Adversaire qui glisse vers la gauche | tourelle de 145° à 169° | Conforme |
| Suite de tests | 55 / 55 | Conforme |

**Statut de l'essai :** Conforme.

**Écarts rencontrés et causes-racines :**

Aucune non-conformité. Les trames sont synthétiques : la portée réelle des VL53L3CX sur la carrosserie d'un adversaire et l'effet des flashs sont mesurés lors des essais de réception avec la fonction testTelemetres().

## 4.7 Réception de l'ordre de départ (simulation du code réel)

**Référence de la simulation :** SIM07

**Exigences client vérifiées :** EXIG_DEPART, EXIG_LUMINOSITE

**Périmètre et technique :** fonction ordreDepartRecu() embarquée ; exécution du code réel sur PC avec un récepteur IRremote simulé.

**But de l'essai :** vérifier que le robot ne part que sur une trame valide de l'IRC01, et jamais sur une répétition, du bruit ou les impulsions de télémètres adverses.

**Fichiers :** simulations/sim07_depart/ (sim07_depart.cpp, IRremote.hpp simulé, sim07_resultats.txt).

**Protocole :**

Le fichier réel depart.cpp est compilé avec un récepteur simulé qui délivre, à chaque appel, une trame telle que la rendrait IRremote après décodage : aucune trame, trame NEC de deux touches différentes, trame de répétition, trames non décodables (protocole UNKNOWN) produites par le bruit, la lumière modulée ou les impulsions d'un télémètre. Pour chaque cas, la décision de départ et le réarmement du récepteur sont relevés.

**Résultats attendus :**

| Grandeur | Valeur attendue (exigence) | Tolérance |
|---|---|---|
| Départ sur l'appui d'une touche de l'IRC01 (EXIG_DEPART) | oui, pour toute touche | — |
| Départ sur répétition, bruit ou impulsions ToF | jamais | aucune |
| Réarmement du récepteur après chaque trame | oui | — |

**Résultats obtenus :**

Les six cas donnent le résultat attendu (tableau 28).

**Tableau 28 :** SIM07 : réponses de ordreDepartRecu() aux trames simulées

| Cas simulé | Départ | Attendu | Réarmement |
|---|---|---|---|
| Aucune trame reçue | non | non | — |
| Touche IRC01, 1er appui (NEC) | oui | oui | oui |
| Autre touche IRC01 (NEC) | oui | oui | oui |
| Répétition NEC (touche maintenue) | non | non | oui |
| Bruit / impulsions ToF (UNKNOWN) | non | non | oui |
| Lumière modulée (UNKNOWN) | non | non | oui |

| Grandeur | Valeur obtenue | Conformité |
|---|---|---|
| Départ sur appui de touche (2 cas) | départ | Conforme |
| Départ sur répétition, bruit, ToF (3 cas) | aucun départ | Conforme |
| Sans trame | aucun départ | Conforme |
| Réarmement | 6 / 6 | Conforme |

**Statut de l'essai :** Conforme.

**Écarts rencontrés et causes-racines :**

Aucune non-conformité. Une première version n'acceptait qu'un code de touche (0x45) qui restait à relever. Cause-racine : le code de la touche n'était pas connu lors de l'écriture ; la relecture du [CDC] (« dès que le juge appuie sur une touche ») a montré qu'il n'est pas nécessaire. La fonction accepte désormais toute trame décodée, ce qui supprime la donnée à relever. La portée et la tenue à l'éclairage intense du TSOP34438 sont mesurées lors des essais de réception.

## 4.8 Contrôles ERC et DRC de la CAO

**Référence de la simulation :** SIM08

**Exigences client vérifiées :** EXIG_SCH_LISIBILITE, EXIG_PCB_PISTES, EXIG_PCB_VIAS, EXIG_PCB_GND

**Périmètre et technique :** projet KiCad complet ; contrôles automatiques ERC et DRC de KiCad 9 et relevé des pistes et vias sur le fichier natif.

**But de l'essai :** détecter les incohérences du schéma, les connexions manquantes et les défauts géométriques du routage.

**Fichiers :** outputs/ddf/controle/ERC.json et DRC.json (15/10/2026) ; DRC du 13/11/2026 (kicad-cli 9).

**Protocole :**

L'ERC est exécuté sur la racine hiérarchique, la netlist exportée, puis le DRC lancé avec contrôle de parité schéma–PCB ; les largeurs de pistes et le nombre de vias sont relevés sur le fichier natif.

**Résultats attendus :**

| Grandeur | Valeur attendue (exigence) | Tolérance |
|---|---|---|
| Erreurs ERC | 0 | — |
| Connexions manquantes et courts-circuits (DRC) | 0 | — |
| Erreurs géométriques DRC (dont garde de 0,5 mm au bord) | 0 | — |
| Pistes de puissance / de signal (EXIG_PCB_PISTES) | ≥ 0,8 mm / ≥ 0,4 mm | — |
| Vias hors masse (EXIG_PCB_VIAS) | ≤ 12, Ø 1,3 / 0,7 mm | — |

**Résultats obtenus :**

Le contrôle du 15/10/2026 relevait quatre erreurs géométriques, corrigées le 13/11/2026 : C7 décalé de 0,1 mm et SW2 de 0,1 mm vers l'intérieur, pastille 1 de Q2 reliée en connexion pleine à la zone de masse. Les deux segments de 0,5 mm du nœud de commutation de U2 ont été élargis à 0,8 mm. Le DRC du 13/11/2026 ne relève plus aucune erreur ; les 14 écarts de parité correspondent aux 13 repères placés volontairement sur les cartes externes (R16 à R23, TP101 à TP105) et au logo en cuivre.

| Grandeur | Valeur obtenue | Conformité |
|---|---|---|
| Erreurs ERC | 0 | Conforme |
| Connexions manquantes et courts-circuits | 0 | Conforme |
| Erreurs géométriques DRC (13/11/2026) | 0 | Conforme |
| Pistes : puissance / signal | 163 à 0,8 mm / 116 à 0,4 ou 0,6 mm | Conforme |
| Vias hors masse | 1 sur 4, Ø 1,3 / 0,7 mm | Conforme |

**Statut de l'essai :** Conforme.

**Écarts rencontrés et causes-racines :**

Écarts du 15/10/2026, corrigés. Cause-racine des écarts au bord : placement final de C7 et SW2 à 0,48 mm du contour, sous la garde de 0,5 mm fixée dans les règles de la carte, non revu avant l'export du 15/10/2026. Cause-racine de la branche thermique unique de Q2 : la zone de masse n'atteignait sa pastille 1 que d'un seul côté ; une connexion pleine convient puisque cette pastille est reliée à la masse. Les 225 avertissements de sérigraphie concernent une couche non fabriquée.

## 4.9 Conclusion de la simulation / prototypage rapide du produit

Les huit dérisquages sont conformes à leurs résultats attendus. Ils confirment les dimensionnements électriques (seuil de batterie, commande et consommation des moteurs, détection de la bordure, connectivité et routage de la carte) et le logiciel embarqué : suivi de cible à 250 mm, réception de l'ordre de départ, et décision apprise qui gagne 73,7 % des combats de validation contre onze comportements adverses (+18,4 points pendant l'entraînement) en passant la porte d'acceptation face au contrôleur de référence. Les essais de réception du DDV vérifient ces résultats sur le robot assemblé.

# 5. Conclusion de la conception du produit

La conception répond aux 36 exigences du [CDC] : chaque solution technique est choisie par une fiche d'aide à la décision, dimensionnée à partir des fiches techniques et des cours, reliée à son schéma et à son code, et dérisquée par simulation ou prototypage. La sûreté repose d'abord sur le matériel (protection de polarité, interrupteur électronique, servo coupé par défaut), puis sur la priorité donnée au départ, à la batterie et à la bordure. La décision a été mise au point sur des centaines de milliers de combats simulés ; le réseau entraîné par renforcement dépasse nettement le contrôleur écrit à la main.

Les écarts relevés pendant la conception ont été corrigés le 13/11/2026 ; le tableau suivant les récapitule avec leur cause-racine, ainsi que les points de vigilance transmis aux essais de réception.

**Tableau 29 :** Écarts corrigés et points de vigilance

| Point | Cause-racine | Traitement |
|---|---|---|
| DRC : 4 écarts géométriques (15/10) | placement à 0,48 mm du bord ; zone de masse d'un seul côté de Q2 | corrigé (13/11) : DRC sans erreur |
| Feuille capteurs : titres de cadres et numéro de page | titres et numérotation repris d'une version à neuf feuilles | corrigé (13/11) |
| Deux fonctions de suivi de plus de 60 lignes | fonctions enrichies au fil des essais du prototype | découpées (13/11) ; comportement identique (SIM06) |
| Seuil CNY70 initial de 1,0 V | seuil choisi sur les mesures à 5 V | 1,5 V calculé pour 3,3 V (§ 3.2.2) |
| Départ sur un seul code de touche | lecture « une touche déterminée » du CDC | toute trame IRC01 valide (SIM07) |
| Autonomie : 0,57 A à pleine vitesse pour 0,462 A admis | moteurs à 100 % en permanence | rapport cyclique moyen limité à 47 % ; mesure de 65 min en réception |
| Mesure batterie sur l'ADC2 | ADC2 partagé avec le Wi-Fi | radio coupée en combat ; gain étalonné ; seuil à 6,7 V |
| Victoires contre l'embusqué et la pelle (31 à 53 %) | adversaires peu visibles avant le contact, simulateur révisé plus exigeant | entraînement poursuivi ; nouveau champion seulement s'il passe la porte |

Le robot assemblé est ensuite soumis aux essais de réception du dossier de vérification (DDV).

# 6. Matrice de conformité du produit

Ce chapitre synthétise la conformité de la conception du produit par rapport aux exigences du cahier des charges. « Conforme (conception) » signifie que la conception satisfait l'exigence et sa tolérance, démontré par le choix, le calcul, la simulation ou le prototypage indiqués ; l'essai de réception sur le robot assemblé est consigné dans le DDV.

**Tableau 30 :** Matrice de conformité du produit

| Exigence | Méthodes de vérification | Éléments vérifiant l'exigence | Statut |
|---|---|---|---|
| EXIG_CHASSIS_DIMENSIONS | Conception | CPR_ARCHI_ELEC, FAB04/FAB05 | Conforme (conception) |
| EXIG_MASSE | Conception | CPR_ARCHI_ELEC, SIM05 | Conforme (conception) |
| EXIG_INTEGRITE_MECA | Conception | CPR_ARCHI_ELEC | Conforme (conception) |
| EXIG_INNOVATION_MECA | Conception | CPR_ARCHI_ELEC | Conforme (conception) |
| EXIG_FIXATION_CARTE | Conception | FAB04/FAB05 | Conforme (conception) |
| EXIG_AUTONOMIE | Conception Prototypage | CPR_BATTERIE, SIM02 | Conforme (conception) |
| EXIG_SECUR_BATT | Conception Simulation | CDT_BATTERIE_ADC, CDT_PROTECTION, SIM01 | Conforme (conception) |
| EXIG_PROT_INV_POL | Conception | CPR_BATTERIE, CDT_PROTECTION | Conforme (conception) |
| EXIG_INTER | Conception | CPR_BATTERIE, CDT_PROTECTION | Conforme (conception) |
| EXIG_ADVERSAIRE | Conception Simulation | CPR_ADVERSAIRE, CDT_ACQUISITION, SIM04, SIM06 | Conforme (conception) |
| EXIG_LUMINOSITE | Conception Prototypage | CDT_ACQUISITION, CDT_SOL, CDT_DEPART, SIM03, SIM07 | Conforme (conception) |
| EXIG_DEPART | Conception Simulation | CDT_DEPART, SIM07 | Conforme (conception) |
| EXIG_DEPLACEMENT | Conception Simulation Prototypage | CDT_SOL, CDT_ACTION, SIM02, SIM03, SIM04 | Conforme (conception) |
| EXIG_COMPORTEMENT | Conception Simulation | CPR_INFO_TRA, CDT_TRAITEMENT, SIM04, SIM05 | Conforme (conception) |
| EXIG_CARTE | Conception | CPR_MCU, CDT_CARTE, CDT_TRAITEMENT | Conforme (conception) |
| EXIG_CARTE_VISU | Conception | CDT_PROTECTION | Conforme (conception) |
| EXIG_SCH_LISIBILITE | Conception | CDT_CARTE, SIM08 | Conforme (conception) |
| EXIG_SCH_SYMBOLES | Conception | CDT_CARTE | Conforme (conception) |
| EXIG_SCH_VALEURS | Conception | CDT_CARTE | Conforme (conception) |
| EXIG_PCB_EMPREINTES | Conception | FAB02/FAB05 | Conforme (conception) |

| Exigence | Méthodes de vérification | Éléments vérifiant l'exigence | Statut |
|---|---|---|---|
| EXIG_PCB_PISTES | Conception Simulation | SIM08, FAB03 | Conforme (conception) |
| EXIG_PCB_VIAS | Conception Simulation | SIM08, FAB04 | Conforme (conception) |
| EXIG_PCB_GND | Conception Simulation | SIM08, FAB03 | Conforme (conception) |
| EXIG_PCB_CMS | Conception | CPR_REGULATION, FAB02/FAB05 | Conforme (conception) |
| EXIG_PCB_FIXATION | Conception | FAB04/FAB05 | Conforme (conception) |
| EXIG_CODE_LANGAGE | Conception | CPR_ARCHI_INFO, CDT_TRAITEMENT | Conforme (conception) |
| EXIG_CODE_FONCTION | Conception Simulation | CDT_TRAITEMENT, SIM06 | Conforme (conception) |
| EXIG_CODE_IMBRICATION | Conception | CDT_TRAITEMENT | Conforme (conception) |
| EXIG_CODE_VARIABLE | Conception | CDT_TRAITEMENT | Conforme (conception) |
| EXIG_CODE_TEST | Conception | CDT_TRAITEMENT | Conforme (conception) |
| EXIG_DELAI | Conception | CPR_COUT, planning RMS_PDP | Conforme (conception) |
| EXIG_COUT | Conception | CPR_COUT, FAB02 | Conforme (conception) |
| EXIG_FOURNISSEUR | Conception | CPR_COUT, FAB02 | Conforme (conception) |
| EXIG_DRIVE | Conception | Google Drive de l'équipe | Conforme (conception) |
| EXIG_FORMAT_DOC | Conception | DDC, DDF (PDF) | Conforme (conception) |
| EXIG_NOM_DOC | Conception | DDC, DDF | Conforme (conception) |