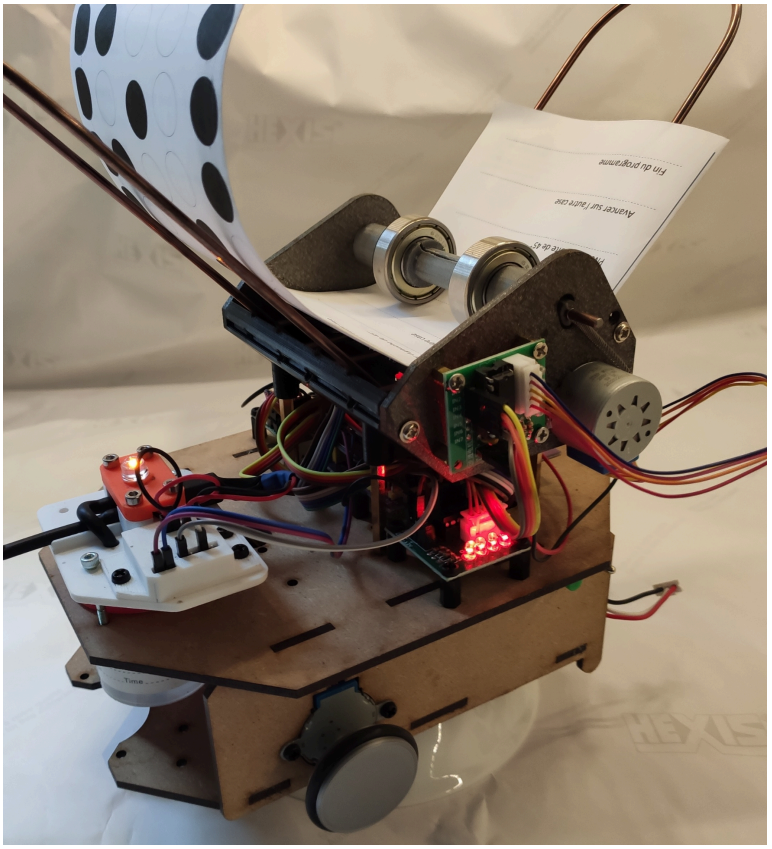
Rapport Technique : Rétro-ingénierie et Fiabilisation de l'Architecture Électronique du "Robot Gommettes"

Étudiants : **BÉNARD, CHABAUD**

Tuteur : **H. HALLIL-ABBAS**

Date de rendu : **19 janvier 2026**



Année universitaire 2025–2026

SOMMAIRE :

1. Introduction et Contexte du Projet

Le projet "Robot Gommettes", initié au sein du FabLab, vise à concevoir un outil pédagogique tangible permettant d'initier les enfants de 5 à 6 ans à la logique algorithmique sans recours aux écrans. Le principe repose sur la lecture optique d'une séquence binaire représentée par des gommettes, interprétée par le robot pour exécuter des déplacements ou des actions spécifiques.

Notre intervention s'inscrit dans une phase de fiabilisation du prototype existant afin de pouvoir le manufacturer le plus simplement possible sans défauts. Bien que le concept pédagogique et la mécanique aient été validés lors d'ateliers précédents, la solution électronique "filaire" utilisée jusqu'alors présentait des instabilités majeures et des risques électriques incompatibles avec un usage en milieu scolaire. Notre mission a consisté à réaliser la rétro-ingénierie de l'existant, à corriger les erreurs de conception critiques et à produire les fichiers de fabrication d'une carte électronique dédiée, centralisant l'intelligence et la puissance du robot.

2. Analyse de l'Existant et Rétro-ingénierie

La première phase de notre travail s'est concentrée sur l'audit technique des prototypes fournis et de la documentation laissée par les précédents stagiaires. L'analyse des schémas sous KiCad et l'inspection visuelle des cartes physiques ont révélé plusieurs incohérences majeures nécessitant une correction immédiate avant toute nouvelle fabrication.

Nous avons identifié une erreur critique de sécurité au niveau de l'étage d'alimentation : le schéma initial présentait un court-circuit franc au niveau de l'interrupteur général, reliant directement le pôle positif de la batterie (7.5V) à la masse (GND). De plus, le dimensionnement des composants passifs autour du régulateur de tension comportait des aberrations, notamment une résistance de rétroaction ("feedback") notée à 100 MΩ au lieu des 100 kΩ nécessaires pour fixer la tension de sortie, rendant la régulation inopérante.

Sur le plan des niveaux logiques, nous avons constaté que le module WiFi de l'ESP32 était alimenté en 5V, alors que ses spécifications techniques imposent une tension de 3.3V. Cette surcharge exposait le microcontrôleur à un risque de destruction irréversible. Enfin, l'absence quasi-totale de condensateurs de découplage sur les lignes d'alimentation menaçait la stabilité du signal numérique et la durabilité des composants face aux appels de courant des moteurs.

3. Conception et Optimisation de la Nouvelle Architecture Électronique

Forts de ce diagnostic, nous avons procédé à la conception d'un nouveau circuit imprimé (PCB) sous le logiciel KiCad, avec pour objectif de regrouper l'intégralité des fonctions sur une carte unique ("All-in-One") afin d'éliminer le câblage volant source de pannes.

Gestion de l'alimentation et Puissance Nous avons redimensionné l'étage de conversion de puissance en intégrant correctement le régulateur abaisseur synchrone (Buck Converter) TPS563240. Ce composant assure désormais une conversion efficace de la tension batterie (7.4V) vers un 5V stable avec un rendement thermique optimisé. Nous avons calculé et

intégré les réseaux de résistances diviseurs de tension adéquats et ajouté les condensateurs de découplage nécessaires en entrée et sortie pour filtrer les parasites générés par la commutation.

Pour le pilotage des actionneurs, nous avons fait le choix d'intégrer directement trois drivers ULN2003 sur le PCB. Contrairement à la version précédente qui utilisait des modules externes encombrants, cette intégration sur la carte mère optimise l'espace. Nous avons ajouté un troisième driver par rapport au design initial afin de piloter indépendamment les deux moteurs de propulsion et le moteur d'entraînement du papier, garantissant ainsi une compatibilité totale avec le firmware existant sans nécessiter de refonte logicielle majeure.

Interface Logique et Sécurité L'interfaçage avec le microcontrôleur ESP32 a été entièrement revu. Nous avons corrigé les tensions d'alimentation des périphériques, notamment en assurant une ligne 3.3V dédiée pour le module WiFi et les capteurs sensibles. Le routage des pistes a été effectué en tenant compte des contraintes de courant pour les pistes de puissance et de l'intégrité du signal pour les pistes de données.

4. Intégration Mécanique et Logicielle

La conception du PCB a été dictée par des contraintes mécaniques strictes. Nous avons respecté les dimensions du châssis en bois existant, en positionnant les trous de fixation selon les entraxes relevés sur les prototypes (4,7 cm x 6,2 cm). L'utilisation de connecteurs JST en bord de carte a été privilégiée pour faciliter le montage et la maintenance future par les utilisateurs finaux.

Concernant la partie logicielle, notre architecture matérielle a été pensée pour être transparente vis-à-vis du code existant. Le microcontrôleur ESP32 conserve sa logique de fonctionnement ; seule une réaffectation mineure des broches (pinout) dans le fichier d'en-tête sera nécessaire lors du téléversement pour correspondre au nouveau routage optimisé.

5. État d'Avancement et Livrables

À ce jour, la phase de conception est terminée. Les schémas électroniques ont été validés et le routage du PCB est finalisé sous KiCad. Nous avons généré l'ensemble des fichiers de fabrication (Gerbers, fichiers de perçage) ainsi que la liste des composants (BOM) mise à jour, sans impact notable sur le budget initialement alloué de 30€ par robot.

La prochaine étape critique consiste en l'approvisionnement des composants et la fabrication des circuits imprimés. Les tests de validation électrique et fonctionnelle seront réalisés dès réception du matériel. L'intégralité de notre travail, incluant les fichiers sources KiCad corrigés et les explications techniques, a été documentée sur le Wiki du projet pour assurer la continuité et l'Open Source.

6. Conclusion et Perspectives

Cette mission a permis de faire passer le projet "Robot Gommettes" du stade de prototype artisanal à celui de **concept industriel pré-validé**. En réalisant la rétro-ingénierie et en corrigeant les failles critiques de sécurité, nous avons établi un dossier de fabrication (PCB et BOM) sain et documenté.

Si la conception théorique répond désormais aux exigences de sécurité, de coût et de reproductibilité, la **validation physique** reste l'étape charnière. La fabrication de la nouvelle carte électronique et les tests fonctionnels qui suivront sont indispensables pour confirmer la fiabilité de cette architecture "All-in-One".

Ce n'est qu'à l'issue de cette phase de vérification technique que le dispositif pourra être qualifié de "prêt pour le déploiement" et produit en série pour les ateliers pédagogiques. Le projet dispose désormais de toutes les bases techniques solides pour aborder cette étape décisive.

Annexe 1 : Protocole de validation et de test matériel.

Objectif : Ce protocole définit les étapes de vérification électrique et fonctionnelle de la nouvelle carte électronique "Robot Gommettes" (Version 2.0 - PCB KiCad) avant son assemblage final dans le châssis. Il vise à prévenir tout risque de destruction des composants (notamment l'ESP32) dû à d'éventuels défauts de soudure ou de fabrication.

Matériel requis :

- Multimètre numérique (Voltmètre / Ohmmètre / Testeur de continuité).
- Alimentation de laboratoire (réglée sur 7.4V) ou batterie Li-Po chargée.
- Câble USB pour la programmation.
- La carte PCB assemblée (sans l'ESP32 si celui-ci est monté sur support, sinon avec).

Phase 1 : Inspection Visuelle (Hors Tension)

Avant toute mise sous tension, une inspection minutieuse est réalisée à la loupe ou à l'œil nu.

1. **Sens des composants polarisés :** Vérification de l'orientation de la puce **TPS563240** (repère pin 1), des condensateurs électrochimiques, des diodes et des circuits intégrés **ULN2003**.
2. **Qualité des soudures :** Recherche de "ponts de soudure" (courts-circuits involontaires) entre les pattes fines des composants CMS et traversants.
3. **Connecteurs :** Vérification de la polarité des connecteurs JST (Alimentation, Moteurs) pour éviter une inversion de polarité fatale.

Phase 2 : Tests de Continuité et d'Isolation (Hors Tension)

Utilisation du multimètre en mode "bip" (continuité).

1. **Test VCC / GND :** Vérifier l'absence de court-circuit entre les bornes `+` et `-` de l'entrée batterie (7.4V).
2. **Test 5V / GND :** Vérifier l'absence de court-circuit en sortie du régulateur TPS563240.
3. **Test 3.3V / GND :** Vérifier l'absence de court-circuit sur les rails d'alimentation du module WiFi/Capteurs.
    - *Critère de validation :* Le multimètre ne doit **pas** biper. Une résistance infinie ou croissante (charge des condensateurs) doit être observée.

Phase 3 : Validation de l'Alimentation (Sous Tension)

Mise sous tension progressive de la carte. Si l'ESP32 est sur support, il est retiré pour cette étape.

1. Brancher la batterie (ou l'alimentation labo).
2. **Mesure de la tension d'entrée :** Vérifier la présence de ~7.4V en entrée.
3. **Validation du Régulateur Buck :** Mesurer la tension aux bornes des condensateurs de sortie du **TPS563240**.
    - *Critère de validation :* La tension doit être stable à **5.0V ± 0.1V**.
    - *Note :* Si la tension dépasse 5.2V, couper immédiatement l'alimentation (risque pour l'ESP32).

Phase 4 : Tests Fonctionnels Unitaires

Une fois les tensions validées, l'ESP32 est inséré et le code de test (`test_hardware.ino`, voir Annexe 2) est téléversé.

1. **Test Moteurs (x3) :** Vérification de la rotation dans les deux sens pour :
    - Moteur Roue Gauche.
    - Moteur Roue Droite.
    - Moteur Lecteur Papier (nouveau driver intégré).
2. **Test Pompe :** Activation de la sortie transistor (MOSFET/BJT) commandant la pompe. Vérification du bruit de fonctionnement ou de la tension aux bornes du connecteur.
3. **Test Capteurs IR (x5) :** Passage d'une feuille blanche puis d'une zone noire devant chaque capteur. Vérification du changement d'état (0/1) via le moniteur série.

Annexe 2 : Code source de test (`test_hardware.ino`).

Ce programme minimaliste permet d'isoler les pannes matérielles en activant individuellement les entrées et sorties du microcontrôleur, sans interférence avec l'algorithme complexe du robot.

Fichier : `test_hardware.ino` **Plateforme :** Arduino IDE / PlatformIO pour ESP32

```
/*
 * ROBOT GOMMETTES - FIRMWARE DE TEST MATÉRIEL (V1.0)
 * Auteur : [Ton Nom]
 * Objectif : Valider le PCB (Alim, Drivers, Capteurs) sans logique complexe.
 */

// --- CONFIGURATION DU PINOUT (À ADAPTER SELON LE SCHÉMA KICAD RETENU)
---
// Moteurs (Pilotage via ULN2003)
#define PIN_MOTEUR_GAUCHE_1  16
#define PIN_MOTEUR_GAUCHE_2  17
#define PIN_MOTEUR_DROIT_1   18
#define PIN_MOTEUR_DROIT_2   19
#define PIN_MOTEUR_PAPIER    21

// Capteurs Optiques (Infrarouge)
#define PIN_CAPTEUR_1        34 // Input Only sur ESP32
#define PIN_CAPTEUR_2        35

void setup() {
  // Initialisation du port série pour le retour d'information
  Serial.begin(115200);
  while (!Serial) { delay(10); }

  Serial.println("\n--- DIAGNOSTIC PCB ROBOT GOMMETTES ---");
  Serial.println("Commandes disponibles :");
  Serial.println("[A] Activer Moteur Gauche (1 sec)");
  Serial.println("[Z] Activer Moteur Droit (1 sec)");
  Serial.println("[E] Activer Moteur Papier (1 sec)");
  Serial.println("[R] Lire les capteurs en continu (5 sec)");
  Serial.println("-------------------------------------");

  // Configuration des broches en Sortie
  pinMode(PIN_MOTEUR_GAUCHE_1, OUTPUT);
  pinMode(PIN_MOTEUR_GAUCHE_2, OUTPUT);
  pinMode(PIN_MOTEUR_DROIT_1, OUTPUT);
  pinMode(PIN_MOTEUR_DROIT_2, OUTPUT);
```

```
  pinMode(PIN_MOTEUR_PAPIER, OUTPUT);

  // Configuration des broches en Entrée
  pinMode(PIN_CAPTEUR_1, INPUT);
  pinMode(PIN_CAPTEUR_2, INPUT);

  stopAll(); // Sécurité au démarrage
}

void loop() {
  if (Serial.available() > 0) {
    char cmd = Serial.read();

    switch (cmd) {
      case 'a': case 'A':
        Serial.println("TEST: Moteur GAUCHE ON...");
        digitalWrite(PIN_MOTEUR_GAUCHE_1, HIGH);
        digitalWrite(PIN_MOTEUR_GAUCHE_2, HIGH);
        delay(1000);
        stopAll();
        Serial.println("TEST: Moteur GAUCHE OFF.");
        break;

      case 'z': case 'Z':
        Serial.println("TEST: Moteur DROIT ON...");
        digitalWrite(PIN_MOTEUR_DROIT_1, HIGH);
        digitalWrite(PIN_MOTEUR_DROIT_2, HIGH);
        delay(1000);
        stopAll();
        Serial.println("TEST: Moteur DROIT OFF.");
        break;

      case 'e': case 'E':
        Serial.println("TEST: Moteur PAPIER ON...");
        digitalWrite(PIN_MOTEUR_PAPIER, HIGH);
        delay(1000);
        stopAll();
        Serial.println("TEST: Moteur PAPIER OFF.");
        break;

      case 'r': case 'R':
        Serial.println("TEST: Lecture Capteurs (Passez une gommette)...");
        long startTime = millis();
        while (millis() - startTime < 5000) { // Lecture pendant 5 secondes
          int val1 = digitalRead(PIN_CAPTEUR_1);
          int val2 = digitalRead(PIN_CAPTEUR_2);
          Serial.print("Capteur 1: "); Serial.print(val1);
          Serial.print(" | Capteur 2: "); Serial.println(val2);
```

```
        delay(100);
      }
      Serial.println("Fin lecture capteurs.");
      break;
    }
  }
}

// Fonction de sécurité pour tout arrêter
void stopAll() {
  digitalWrite(PIN_MOTEUR_GAUCHE_1, LOW);
  digitalWrite(PIN_MOTEUR_GAUCHE_2, LOW);
  digitalWrite(PIN_MOTEUR_DROIT_1, LOW);
  digitalWrite(PIN_MOTEUR_DROIT_2, LOW);
  digitalWrite(PIN_MOTEUR_PAPIER, LOW);
}
```