

Fiche Bilan et Scénario Pédagogique du TIPE

Théo

17 janvier 2026

1. DÉFINITION DU SYSTÈME

- **Type de robot** : Mobile Wheeled Inverted Pendulum (MWIP).
- **Spécificité** : Structure "Maison" (Breadboard) → Répartition de masse incertaine + Driver DBH-12V.
- **Objectif** : Stabiliser le robot à la verticale ($\theta = 0$) et maîtriser sa position (x).

2. LE SCÉNARIO "FIL ROUGE" : DE LA BOÎTE NOIRE À LA MÉCANIQUE

Ce TIPE suit une démarche d'ingénierie inverse : nous testons des solutions de commande "aveugles" et ne passons à la modélisation physique que lorsque la méthode l'exige impérativement.

Phase 1 : L'Approche "Boîte Noire" (Aucune Physique)

Nous considérons le robot comme un système inconnu. Nous ne posons **aucune équation mécanique**. Nous asservissons le système uniquement via le traitement du signal d'erreur $\varepsilon(t)$.

1. **Approche Linéaire Classique (PID)** :
 - On implémente un correcteur $U(t) = K_p\varepsilon + K_i \int \varepsilon + K_d \dot{\varepsilon}$.
 - *Constat* : Difficile à stabiliser car le système est instable en boucle ouverte. Le gain proportionnel K_p doit être très élevé, ce qui crée des oscillations.
2. **Approche Robuste (SMC "Signe")** : On définit une surface de glissement S . On applique une commande "Tout ou Rien" : $U = -K \cdot \text{sign}(S)$.
 - *Constat* : C'est très robuste (le robot ne tombe pas), mais c'est violent mécaniquement. Le phénomène de *chattering* (vibrations) apparaît.
3. **Approche Adoucie (SMC "Tanh")** : On remplace la fonction 'sign' par une tangente hyperbolique 'tanh' pour lisser la transition.
 - *Constat* : Moins de vibrations, mais on perd un peu en réactivité pure.
4. **Approche Hybride (PID Non-Linéaire / "P-SMC")** : On tente de combiner les deux mondes avec une commande "maison".
 - Le terme proportionnel du PID n'est plus un gain fixe K_p , mais une fonction non-linéaire inspirée du SMC (type 'tanh').
 - *Idée* : Avoir un gain fort quand l'erreur est grande (réactivité du SMC) et un comportement doux quand l'erreur est faible (précision du PID).
5. **Bilan de la Phase 1 (Le Mur)** : Toutes ces méthodes fonctionnent plus ou moins, mais elles sont **énergivores**. On ne maîtrise pas le couple moteur, on sature souvent le driver, et on risque la casse mécanique à terme. Il nous faut une méthode qui optimise l'énergie.

Phase 2 : Le Besoin d'Optimisation (LQR)

Nous nous tournons vers le régulateur **LQR (Linear Quadratic Regulator)**.

- *Pourquoi ?* C'est le seul algo qui minimise mathématiquement un coût J_{cout} prenant en compte l'erreur (tenir debout) ET l'énergie (ne pas saturer les moteurs).
- *Le Problème :* Le LQR ne fonctionne pas "à l'aveugle". Il exige de connaître la représentation d'état du système :

$$\dot{X} = AX + BU$$

Conclusion : C'est l'exigence du LQR qui nous **oblige** à faire de la mécanique.

Phase 3 : L'Ouverture de la Boîte (La Modélisation Mécanique)

C'est seulement maintenant que nous faisons de la physique pour trouver ces matrices A et B .

1. Modèle A : Masse Ponctuelle (Simplifié)

- *Hypothèse :* On néglige les inerties ($J \approx 0$). Tout est dans la batterie.
- *But :* Obtenir des matrices A et B simples pour un premier LQR dégrossi.

2. Modèle B : Modèle Inertiel (Complet)

- *Hypothèse :* On prend en compte $J_{chassis}$ et J_{roues} .
- *But :* Obtenir les matrices A et B exactes pour un LQR haute performance.

Phase 4 : Synthèse Finale

- Calcul des gains K_{LQR} via Python avec les matrices du Modèle B.
- Implémentation sur le robot.
- Validation : Le robot tient debout avec des mouvements beaucoup plus fluides et économies que le PID/SMC de la phase 1.

3. AXES D'AMÉLIORATION

A. Gestion des Frottements

Ajout d'une compensation statique pour contrer la zone morte des moteurs.

B. Observateur d'État (Kalman)

Utilisation d'un filtre de Kalman pour estimer proprement la vitesse \dot{x} (bruité par dérivation simple).

4. TO-DO LIST IMMÉDIATE

Actions Prioritaires

1. **Code "Boîte Noire"** : Implémenter et tester le PID, le SMC (Sign/Tanh) et l'Hybride sur Arduino.
2. **Modélisation** : Écrire les équations du mouvement (Lagrange/Newton) pour extraire A et B .
3. **Calcul LQR** : Utiliser le script Python pour générer les gains K optimaux.
4. **Comparaison** : Filmer le comportement du robot avec chaque loi de commande.