RAPPORT DE STAGE

1er juin 2025 - 27 juin 2025

DÉVELOPPEMENT SUR LE ROBOT UNITREE GO2

Étudiant : Alexis Xueref

Encadrant : Pascal Masson

Représentant : Erwin Franquet

Ecole Polytechnique Universitaire de Nice Sophia-Antipolis, Robotique Systèmes Autonomes
930 route des Colles, Parc de Sophia Antipolis, 06410 BIOT

Remerciements :

Je tiens à remercier en premier lieu Pascal Masson, mon encadrant de stage, pour son accompagnement technique rigoureux, ses conseils précis et sa disponibilité constante tout au long du stage.

Je remercie également Erwin Franquet, représentant du laboratoire, pour m'avoir accueilli dans l'environnement de travail et permis d'évoluer sur une plateforme robotique avancée comme le Unitree Go2.

Je souhaite aussi exprimer ma reconnaissance à Marco et Christian Masson, qui ont accepté de tester mes documents techniques. Leurs retours m'ont permis d'en améliorer la clarté et la pédagogie de manière concrète.

Je remercie Adrien Waeles et Rémi Guzzi pour le partage de leurs travaux antérieurs, qui m'ont fourni une base solide pour comprendre le fonctionnement du robot et structurer mes développements.

Enfin, merci à l'équipe pédagogique de Polytech Nice Sophia pour avoir rendu ce stage possible.

Résumé :

Ce stage a été réalisé dans le cadre de ma formation en robotique à Polytech Nice Sophia, au sein d'un laboratoire spécialisé dans les systèmes autonomes. L'objectif principal était de prendre en main le robot quadrupède Unitree Go2, de comprendre son environnement logiciel, et de produire des ressources pédagogiques permettant son utilisation par d'autres étudiants.

Le premier livrable a consisté dans la rédaction d'un document décrivant les prérequis nécessaires à l'installation et à la configuration de l'environnement de développement sous Ubuntu. Ce guide a été testé et validé pour garantir une reproductibilité totale.

Le second volet du stage a porté sur la conception d'un premier TP fonctionnel, permettant au robot d'exécuter des mouvements simples (avancer et reculer) et de mesurer l'erreur entre la commande et le déplacement réel. Une deuxième version de ce TP intègre un filtre de correction (type Kalman) afin de réduire l'erreur mesurée.

L'ensemble du travail s'inscrit dans une logique de structuration pédagogique, en vue de proposer une base stable pour les futurs enseignements pratiques sur le robot Go2.

Abstract :

This internship was conducted as part of my robotics engineering program at Polytech Nice Sophia, within a laboratory focused on autonomous systems. The main objective was to get familiar with the Unitree Go2 quadruped robot, understand its software environment, and produce pedagogical resources for future student use.

The first deliverable was the creation of a comprehensive guide detailing the prerequisites and setup procedures for the development environment on Ubuntu. This document was tested to ensure full reproducibility.

The second part of the work involved designing a first lab exercise, allowing the robot to perform basic movements (forward and backward) while measuring the discrepancy between commanded and actual displacement. A second version of the lab integrates a correction filter (e.g., Kalman) to reduce the observed error.

All contributions were structured with a pedagogical focus, aiming to provide a reliable foundation for future practical courses involving the Go2 robot.

Sommaire

Remerciements : 2
Résumé : 3
Abstract : 3
Sommaire 4
I. Introduction 5
I.1. Contexte du stage 5
I.2. Objectifs du stage 5
II. Présentation de l'environnement de travail 6
II.1. L'entreprise / le laboratoire d'accueil 6
II.2. Le robot Unitree Go2 6
II.2.a Capacités mécaniques et capteurs 6
II.2.b. Systèmes embarqués et interface 7
II.3. Environnement de développement 7
II.3.a. Outils logiciels 7
II.3.b. Dépôt GitHub de référence 7
III. Travail réalisé 8
III.1. Compréhension des algorithmes existants 8
III.1.a. Architecture logicielle 8
III.1.b. Algorithmes de localisation 8
III.2. Analyse critique 8
III.2.a. Limites observées 8
III.2.b. Propositions d'améliorations 9
III.3. Nouvelles fonctionnalités développées 9
III.3.a. Fichier d'explication des prérequis de l'environnement Ubuntu 9
III.3.b. TP1 – Mouvement rectiligne et mesure d'erreur 10
IV. Procédures pour l'ajout de nouvelles fonctions 11
IV .1. Étapes d'intégration 11
IV.2. Conseils et bonnes pratiques 12
V. Conclusion et perspectives 13
VI. Annexes 14

I. Introduction

I.1. Contexte du stage

Dans le cadre de ma formation d'ingénieur en robotique à Polytech Nice Sophia, ce stage a été réalisé au sein même de l'école. Ce robot, récemment acquis, constitue une plateforme avancée pour l'expérimentation en robotique mobile à jambes.

L'objectif du stage était double : d'une part, approfondir la compréhension de l'environnement logiciel du Go2 et, d'autre part, produire des supports pédagogiques destinés aux futurs travaux pratiques. Le stage visait à structurer les premières activités d'apprentissage autour du robot, en commençant par des manipulations simples mais fondamentales.

Ce travail s'inscrit dans une volonté de l'équipe enseignante de rendre accessible les outils robotiques complexes à travers une démarche progressive et reproductible, directement intégrée dans les cursus de Robotique et systèmes autonomes à Polytech Nice-Sophia.

I.2. Objectifs du stage

Les objectifs du stage étaient centrés sur la mise en place d'une base pédagogique fonctionnelle pour l'utilisation du robot Unitree Go2 dans le cadre des enseignements à Polytech Nice Sophia. Plus précisément, il s'agissait de :

- Documenter les prérequis techniques et les étapes nécessaires à l'installation de l'environnement de développement sous Ubuntu, en s'appuyant sur le SDK fourni par Unitree.
- Comprendre l'architecture logicielle du robot afin de pouvoir modifier ou ajouter des fonctions à la boucle de commande.
- Concevoir un premier TP exploitant les capacités de locomotion du Go2 pour effectuer des mouvements simples (avancer et reculer), en intégrant une procédure de mesure de l'erreur.
- Implémenter un filtre (type Kalman) dans le TP pour comparer les performances avec et sans correction.
- Structurer l'ensemble de ces travaux dans une logique pédagogique claire, testée et reproductible par d'autres étudiants.

Ce stage devait à la fois renforcer mes compétences techniques sur les systèmes robotiques réels et fournir des livrables exploitables directement dans un cadre pédagogique.

II. Présentation de l'environnement de travail

II.1. L'entreprise / le laboratoire d'accueil

Le stage a été réalisé au sein de Polytech Nice Sophia, école d'ingénieurs publique intégrée à l'Université Côte d'Azur, sur le campus de Sophia Antipolis. Spécialisée notamment en robotique et systèmes autonomes, l'école offre un environnement propice à l'expérimentation et à l'innovation technologique.

L'accueil s'est fait plus précisément au sein du Polytech Lab, un laboratoire interne dédié aux travaux pratiques, projets étudiants, et expérimentations sur plateformes robotiques réelles. Ce laboratoire dispose d'équipements variés, dont plusieurs robots mobiles, bras manipulateurs, capteurs embarqués, et stations de développement.

Le Polytech Lab joue un rôle central dans la mise en œuvre des enseignements orientés projet. Il permet aux étudiants d'évoluer dans des conditions proches de celles rencontrées en recherche ou en entreprise, tout en bénéficiant d'un encadrement pédagogique adapté.

II.2. Le robot Unitree Go2

Le Unitree Go2 est un robot quadrupède de dernière génération conçu pour la locomotion dynamique et la recherche en robotique mobile. Il se distingue par sa stabilité, sa capacité à se déplacer sur divers terrains, et son ouverture logicielle permettant des expérimentations en temps réel.

II.2.a Capacités mécaniques et capteurs

Le Go2 dispose de quatre jambes articulées à trois degrés de liberté chacune, pilotées par des moteurs brushless haute performance avec retour couple intégré. Il peut marcher, trotter, effectuer des virages serrés, et conserver son équilibre en présence de perturbations.

Il est équipé de plusieurs capteurs embarqués, dont :

- Une centrale inertielle (IMU 9 axes)
- Des codeurs absolus sur chaque articulation
- Des capteurs de courant et température dans chaque moteur
- Un module de perception (caméra et LIDAR)

Ces capteurs permettent une estimation en temps réel de l'état du robot (posture, vitesse articulaire, orientation) et sont utilisés dans la boucle de contrôle embarquée.

II.2.b. Systèmes embarqués et interface

Le robot embarque un mini-ordinateur exécutant un système Linux allégé, sur lequel tourne le Unitree SDK, une interface logicielle propriétaire permettant de communiquer avec les actionneurs, lire les capteurs, et envoyer des commandes.

La communication avec un poste de développement externe (PC sous Ubuntu) se fait via un réseau local (Ethernet ou WiFi), en utilisant des messages structurés envoyés à travers le SDK (notamment via gRPC).

L'utilisateur peut ainsi coder et envoyer des commandes en C++ ou Python, récupérer les états internes du robot, et intégrer ses propres modules de traitement ou de filtrage en temps réel.

II.3. Environnement de développement

II.3.a. Outils logiciels

Le développement s'est effectué principalement sous Linux, avec une configuration adaptée à la robotique embarquée. L'environnement intègre l'IDE Visual Studio Code couplé à des extensions pour C++ et ROS. Le SDK Unitree Go2 utilisé est basé sur C++17, avec une compilation via CMake. Le contrôle de version s'est fait avec Git en ligne de commande. Le traitement des données et la visualisation de trajectoires ont été réalisés via des scripts Python, notamment pour le filtrage de Kalman et l'analyse des résultats.

II.3.b. Dépôt GitHub de référence

Le projet s'appuie sur le dépôt officiel Unitree Robotics, accessible à l'adresse : https://github.com/unitreerobotics/unitree_sdk2. Ce dépôt fournit les bibliothèques et exemples nécessaires à la communication et au contrôle du robot Go2.

III. Travail réalisé

III.1. Compréhension des algorithmes existants

III.1.a. Architecture logicielle

Le logiciel embarqué du robot Unitree Go2 repose sur une architecture modulaire en C++, structurée autour d'un système de publication/souscription utilisant les interfaces réseau fournies par le SDK. Le programme principal initialise le contrôleur, configure les capteurs (IMU, odométrie, moteurs) et lance les threads de traitement. L'ensemble du pipeline est organisé pour permettre l'acquisition des données en temps réel, leur enregistrement et leur traitement différé ou en ligne via des algorithmes de filtrage. Cette structure permet une séparation claire entre les couches matériel, acquisition, traitement, et commande.

III.1.b. Algorithmes de localisation

Les algorithmes de localisation utilisés s'appuient principalement sur la fusion de données provenant de l'IMU interne et de l'odométrie issue des moteurs des pattes. Un filtre de Kalman non linéaire (Unscented Kalman Filter - UKF) est utilisé pour estimer la position du robot dans un plan 2D. Le modèle dynamique intègre les erreurs systématiques des capteurs et prend en compte les incertitudes sur les mesures. L'UKF permet ainsi une meilleure robustesse aux bruits et dérives que les approches naïves d'intégration. Les algorithmes ont été testés avec des jeux de données simulés ou enregistrés sur le terrain, et comparés à une trajectoire de référence.

III.2. Analyse critique

III.2.a. Limites observées

Les principales limites rencontrées au cours du développement ont été liées à la précision de la localisation en présence de terrains irréguliers et à la gestion des dérives sur de longues distances. Bien que le filtre de Kalman non linéaire (UKF) ait montré une robustesse satisfaisante sur des surfaces planes et uniformes, il reste sensible aux erreurs d'odométrie accumulées, notamment lors de changements abrupts de direction ou de vitesse. De plus, la capacité du robot à compenser les instabilités dues à des imperfections dans la calibration des capteurs a été mise en évidence lors de tests sur des terrains complexes, ce qui a conduit à des erreurs de localisation significatives. La latence dans le traitement des données et le manque d'adaptation du système aux environnements dynamiques ont également constitué des obstacles à une performance optimale.

III.2.b. Propositions d'améliorations

Pour améliorer la précision de la localisation et la robustesse générale du système, plusieurs pistes peuvent être envisagées. Tout d'abord, l'intégration de capteurs supplémentaires, tels que des lidars ou des caméras stéréoscopiques, pourrait aider à réduire les incertitudes dues aux erreurs d'odométrie et fournir une meilleure compréhension de l'environnement. Une approche hybride combinant des méthodes de SLAM (Simultaneous Localization and Mapping) pourrait également permettre une mise à jour continue de la carte et une localisation plus fiable en terrain complexe. Par ailleurs, l'adaptation dynamique des paramètres du filtre de Kalman en fonction de l'environnement pourrait être une solution pour compenser les dérives dans des situations non idéales. Enfin, l'optimisation du temps de traitement des données, en particulier en réduisant la latence liée à la communication avec les capteurs, offrirait des gains significatifs en termes de réactivité et de stabilité du système.

III.3. Nouvelles fonctionnalités développées

III.3.a. Fichier d'explication des prérequis de l'environnement Ubuntu

Pour faciliter la prise en main du SDK Unitree et garantir une configuration homogène de l'environnement de développement, un document de mise en œuvre a été rédigé. Ce fichier détaille les prérequis techniques (Ubuntu ≥ 20.04, architecture Aarch64/x86_64, GCC 9.4.0) ainsi que les étapes d'installation des dépendances (build-essential, CMake, Eigen, etc.), l'installation de l'éditeur VS Code, et la compilation du SDK.
Il inclut également une section complète sur la configuration réseau nécessaire à la communication entre le PC et le robot Go2, en spécifiant les adresses IP valides, la vérification de la connexion via ping, et les précautions à prendre pour éviter les interférences entre groupes. Ce guide vise à standardiser l'environnement de travail pour tous les utilisateurs du robot, en particulier dans un cadre pédagogique.

III.3.b. TP1 – Mouvement rectiligne et mesure d'erreur

Le TP1 a pour objectif de faire effectuer au robot Unitree Go2 un mouvement rectiligne simple (avancer puis reculer), et de mesurer l'erreur entre la commande envoyée et la position réelle atteinte. Ce TP constitue une première manipulation fondamentale pour initier les étudiants à la commande cinématique du robot et à l'analyse des erreurs d'exécution.

Principe du TP

1. **Commande de mouvement**
   Le robot reçoit une commande de vitesse linéaire selon l'axe avant/arrière (souvent noté vx), avec une vitesse constante pendant un temps donné, lui permettant d'avancer en ligne droite. Après une phase d'arrêt, une commande inverse est envoyée pour le faire reculer sur la même distance.

2. **Mesure de la trajectoire réelle**
   Le robot fournit, à chaque pas de boucle, une estimation de sa position basée sur l'odométrie et l'IMU. Ces données sont collectées tout au long du mouvement pour tracer la trajectoire réelle effectuée par le robot.

3. **Calcul de l'erreur**
   À la fin de l'aller-retour, la position du robot est comparée à sa position initiale. L'écart obtenu (en norme Euclidienne) représente l'erreur cumulée due à l'imprécision de la commande, au glissement, à la dérive de l'estimation ou à d'autres imperfections physiques et logicielles.

Objectifs pédagogiques

- Apprendre à envoyer une commande de mouvement basique via le SDK.
- Comprendre comment récupérer les états internes du robot (position, orientation).
- Quantifier les erreurs de déplacement sur un aller-retour théoriquement symétrique.

Ce TP permet de mettre en évidence les limites naturelles de l'odométrie brute et justifie l'utilisation d'outils de fusion de capteurs pour fiabiliser la localisation.

<u>Librairies externes à installer :</u>

- Eigen3
- Unitree Go2 SDK
- Cyclone DDS (Eclipse Cyclone DDS)
- cmake (si pas déjà installé)
- pkg-config (si pas déjà installé)

IV. Procédures pour l'ajout de nouvelles fonctions

IV .1. Étapes d'intégration

L'ajout de nouvelles fonctionnalités dans le projet suit une procédure structurée garantissant la stabilité du système et la compatibilité avec les composants existants :

1. **Définition claire de la fonctionnalité** : identification des entrées/sorties, fréquence d'exécution, impact sur les modules existants.
2. **Création d'un module indépendant** : développement dans un fichier source dédié, avec un header propre, pour favoriser l'encapsulation.
3. **Intégration au pipeline principal** : ajout du module dans le contrôleur central ou les threads de traitement en fonction du besoin (temps réel ou différé).
4. **Compilation et tests unitaires** : vérification de l'absence d'erreurs à la compilation et validation fonctionnelle isolée.
5. **Tests en conditions réelles** : exécution sur le robot avec enregistrement des données pour analyse postérieure.
6. **Documentation** : mise à jour des fichiers README et des commentaires dans le code pour assurer la maintenabilité.

Dans le cas du TP1 après avoir installé le dossier avec fichiers et librairies exécuter ce code (procédure similaire pour tous les programmes à envoyer) :

```
cd TP1_profs
mkdir build && cd build
cmake ..
make
```

Pour envoyer au Unitree Go2 :

```
sudo ./unitree_go2_localization enp0s25
```

Remplacer enp0s25 par votre nom de cartes réseau trouvé par la commande **ifconfig.**

IV.2. Conseils et bonnes pratiques

- **Respecter la structure du SDK** : éviter de modifier les fichiers natifs, préférer la création de modules dans des dossiers dédiés.
- **Gérer les threads proprement** : utiliser des mécanismes de synchronisation pour éviter les accès concurrents aux données capteurs.
- **Utiliser Git de façon régulière** : commits fréquents, messages clairs, branches pour chaque fonctionnalité.
- **Documenter chaque étape** : autant dans le code que dans les fichiers auxiliaires (fichiers .md ou .txt).
- **Conserver une compatibilité ascendante** : ne pas casser les interfaces déjà utilisées par d'autres modules ou utilisateurs.

V. Conclusion et perspectives

Ce stage m'a permis d'aborder concrètement la prise en main d'un robot quadrupède avancé, le Unitree Go2, dans un cadre pédagogique appliqué. À travers la rédaction d'un guide d'installation complet et le développement d'un TP structuré autour de mouvements simples et de la mesure d'erreur, j'ai pu explorer plusieurs aspects fondamentaux de la robotique mobile : commande de vitesse, estimation de position, limites de l'odométrie, et introduction au filtrage.

L'approche adoptée m'a permis de renforcer mes compétences techniques (programmation C++, compilation, utilisation de SDK propriétaires), tout en développant une rigueur dans la production de documents reproductibles et exploitables par d'autres étudiants.

Perspectives d'évolution :

- Concevoir un TP d'orientation acoustique, dans lequel le robot détecte une source sonore et s'en approche en s'appuyant sur ses microphones et la direction estimée du bruit.
- Développer un TP d'exploration et de cartographie 3D, en exploitant les capteurs du robot (ex. caméra ou LIDAR) pour générer une carte tridimensionnelle de son environnement.
- Intégrer une fonctionnalité de suivi visuel d'objet, où le robot détecte et suit une balle montrée devant sa caméra embarquée, combinant vision par ordinateur et commande dynamique.

Ces perspectives visent à construire une suite cohérente de TP à difficulté croissante, couvrant un large spectre de compétences (perception, localisation, commande, interaction), et faisant du Go2 une plateforme d'apprentissage complète pour les systèmes robotiques intelligents.

VI. Annexes

- Lien du Github de Adrien Waeles et Rémi Guzzi : https://github.com/AdrienWaeles/Localization_Of_Legged_Robots/tree/main
- Lien du Github du SDK de l'Unitree Go2 : https://github.com/unitreerobotics/unitree_sdk2
- Lien du Go2 SDK Development Guide : https://support.unitree.com/home/en/developer/about_Go2