

# JOURNAL D'AVANCEMENT

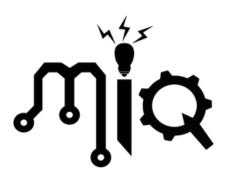
# Projets de Recherche Technologique

EN COLLABORATION AVEC LE LABORATOIRE ICUBE

# Système mécatronique d'aide à l'atterrissage de précision des drones

Auteur: GIRARDET Valentin

Encadrant: Mr. KIEFER Renaud



Mécatronique 5 - Semestre 9

21 novembre 2024

# Avancement des tâches

Date de mise à jour : 21 novembre 2024

# Objectifs généraux :

- 1. Développer un système de stabilisation de caméra avec la Raspberry Pi 4 et deux servomoteurs.
- 2. Intégrer la détection et l'analyse des marqueurs ArUco en temps réel.
- 3. Concevoir un prototype mécanique fonctionnel adapté au système.
- 4. Tester et améliorer les performances globales du système.
- 5. Préparer les livrables et la documentation du projet.

#### Avancement détaillé:

### Tâche 1: Préparation initiale

#### Description:

- ➤ Étude bibliographique sur les systèmes de stabilisation et la détection d'images.
- > Installation de la Raspberry Pi et des dépendances nécessaires (Picamera2, OpenCV).

```
sudo apt update
sudo apt install python3 python3-pip -y
sudo apt update
sudo apt install libopencv-dev python3-opencv -y
```

- ➤ Mise en place des outils de développement (environnements Python etc.).
- > Développement du script camera\_capture.py:
  - Capture de vidéos en utilisant la Pi Camera V2 avec un encodeur H264.
  - Prévisualisation en temps réel avec Preview.QTGL.
  - Option pour capturer des photos en mode preview.

#### Problèmes rencontrés:

- Configuration initiale de la caméra nécessitant des ajustements pour activer toutes les fonctionnalités.
- Difficultés avec les dépendances Python pour Picamera2.

#### Tâche 2 : Développement des programmes de détection

#### Description:

Développement et test de plusieurs scripts pour la gestion des marqueurs ArUco :

- > Compréhension des étapes de base du traitement d'image nécessaires pour lire et identifier un marqueur ArUco en suivant un tutoriel (script : lecture\_manuelle.py) :
  - Extraction des contours, élimination des bordures, et détection des motifs binaires du marqueur pour calculer son ID.
- > Utilisation d'un générateur de marqueurs ArUco (script : aruco\_generator.py) :

- Génération de marqueurs ArUco personnalisés avec des IDs spécifiques.
- Conversion en images RGBA et sauvegarde des marqueurs pour des tests futurs.
- ➤ Détection et identification des marqueurs ArUco en temps réel à partir du flux vidéo de la Pi Camera V2 (script : aruco\_id.py) :
  - Utilisation d'OpenCV pour détecter les marqueurs et afficher les IDs correspondants.
  - Intégration de fonctionnalités avancées, comme le dessin des contours détectés sur les images capturées.
- ➤ Mise en œuvre de la capture vidéo en temps réel avec la Pi Camera V2 pour tester la détection des marqueurs. Voir vidéo

#### Problèmes rencontrés:

- Non détection des marqueurs dûes à une confusion entre les dictionnaires d'ArUco.
- Erreurs dans l'alignement des marqueurs pendant les tests en direct.

#### Solutions apportées:

- Attention à porter sur le choix du dictionnaire ArUco (sélection de la bibliothèque 6x6, marker size = 100mm)
- Ajustement possible des seuils de détection dans OpenCV pour améliorer la robustesse.

# Tâche 3 : Étude et conception mécanique

#### Description:

- > Sélection des moteurs Smart Servo SCS15 après des calculs sur le couple minimal nécessaire :
  - Le couple minimal a été calculé en prenant en compte :  $\alpha = 2 \text{rad/s}^2$  (accéleration angulaire souhaitée) et l'effet de la gravité.
  - En considérant la caméra Pi Camera V2 de masse 3g, un bras d'environ 6 cm, et un support/moteur de 180 g, le couple gravitationnel a été estimé à 0,108 N·m, et le couple inertiel a été négligé en raison de sa faible contribution par rapport au couple gravitationnel, ce qui nous donne donc un total de 0,108 N·m pour le couple minimal du moteur.
  - Les 2 Smart Servo SCS15, avec un couple maximal de 1,47 N·m à 6 V, dépassent donc largement ces attendus, garantissant fiabilité et précision avec une grande marge de sécurité.
- ➤ Les Smart Servo SCS15 ont été choisis pour ces raisons :
  - Compacts, moyennement légèrs et économiquement abordables.
  - Disponibles chez un fournisseur INSA (GOTRONIC)
  - Compatibles avec la Raspberry Pi grâce à leur connectivité série et leur statut de servos intelligents offrant un retour de position.
- Développement du modèle CAO sur CREO pour concevoir un système mécanique aligné avec les exigences des moteurs et de la caméra :
  - Conception des supports en U des servos fournis avec les moteurs pour assurer leur intégration dans le stabilisateur.
  - Conception d'un support en L reliant les deux bras motorisés (pitch et roll) du stabilisateur.
  - Réflexion sur les tolérances et points de rotation pour permettre des ajustements rapides et garantir l'alignement précis entre les axes des moteurs et celui de la caméra.
  - Conception en cours d'un support spécifique pour finaliser l'alignement de la caméra.

#### Problèmes rencontrés:

- Difficultés initiales pour aligner précisément les axes de la caméra et des moteurs dans le modèle CAO.
- Réflexions sur la compacité du système

#### Solutions apportées:

- Ajustement des dimensions et des tolérances dans CREO pour optimiser les alignements.
- Intégration des supports en U et du L-Bracket dans le modèle pour une structure plus robuste.

# Tâche 4 : Documentation et présentation

#### Description:

- Production du livrable d'étude bibliographique
- Mise en place du journal d'avancement et d'un dépôt Github

## Semaine?: Stabilisation et prototypage (futur)

#### ${\bf Description:}$

— Connexion et tests initiaux des Smart Servo avec la Raspberry Pi.

# Bibliographie