Rapport détaillé sur les modèles de conception

Analyse des modèles de conception dans le projet

1. **Modèle de machine à états**
   - **Description**: Le projet repose fortement sur le modèle de la machine à états finis (FSM) pour gérer les états et les transitions du robot.
   - **Preuve**:
     - Le document mentionne le développement d'une bibliothèque FSM générique utilisée pour contrôler les opérations du robot (pages 6-17).
     - La FSM gère des états comme `robot_initialisation`, `robot_integrity`, `home`, et divers états de tâche.

2. **Modèle Singleton**
   - **Description**: Garantit qu'une classe n'a qu'une seule instance.
   - **Preuve**:
     - Le document implique probablement ce modèle pour la classe `Robot` afin d'assurer qu'une seule instance du contrôleur du robot existe, gérant l'état et les actions du robot (page 6).

3. **Modèle Stratégie**
   - **Description**: Définit une famille d'algorithmes et les rend interchangeables.
   - **Preuve**:
     - Le projet abstrait diverses tâches (par exemple, contrôle manuel, tâches automatisées) comme des stratégies interchangeables que l'application principale peut changer en fonction des entrées de l'utilisateur (page 20).

4. **Modèle Observateur**

- **Description**: Notifie automatiquement les objets dépendants lorsque l'état d'un autre objet change.

- **Preuve**:

- La FSM pourrait implémenter ce modèle pour notifier les différentes parties du système des changements d'état, assurant la synchronisation (page 6).

5. **Modèle Méthode Template**

- **Description**: Définit la structure d'un algorithme dans une méthode, en déléguant certaines étapes à des sous-classes.

- **Preuve**:

- La structure de la FSM utilise probablement ce modèle pour définir les étapes génériques des transitions d'état, permettant des comportements spécifiques à être implémentés dans des sous-classes (pages 10-17).

6. **Modèle Composite**

- **Description**: Compose des objets en structures arborescentes pour représenter des hiérarchies partie-tout.

- **Preuve**:

- Le système de gestion des tâches utilise probablement ce modèle pour regrouper diverses tâches et les gérer comme une seule entité, permettant une exécution hiérarchique des tâches (page 6).

Analyse détaillée avec preuves

Modèle de machine à états

- **Exemple**: La classe `FiniteStateMachine` encapsule la logique globale de la machine à états, gérant des états comme `UNINITIALIZED`, `IDLE`, `RUNNING`, et `TERMINAL_REACHED`.
- **Preuve**:
  - Détaillé dans les sections "Librairie FiniteStateMachine" (pages 10-17).
- **Avantage**: Fournit une approche structurée pour gérer les transitions d'état, améliorant la lisibilité et la maintenabilité du code.

Modèle Singleton
- **Exemple**: Garantit qu'il n'y a qu'une seule instance de la classe `Robot`.
- **Preuve**:
  - Impliqué dans la section "Classe Robot", qui mentionne des opérations de haut niveau et la gestion des états pour le robot GoPiGo3 (page 6).
- **Avantage**: Empêche la création de multiples instances du contrôleur du robot, assurant une gestion d'état cohérente.

Modèle Stratégie
- **Exemple**: Gestion de différentes tâches comme `manual_control` et d'autres tâches futures.
- **Preuve**:
  - Le projet abstrait les tâches comme des stratégies interchangeables que l'application peut changer en fonction des entrées de l'utilisateur (page 20).
- **Avantage**: Permet d'ajouter facilement de nouvelles tâches sans modifier la logique principale de l'application.

Modèle Observateur
- **Exemple**: Notifier les différentes parties du système des changements d'état.
- **Preuve**:

- La bibliothèque FSM pourrait utiliser ce modèle pour assurer la synchronisation entre les états de la FSM et les autres composants (page 6).
- **Avantage**: Assure que tous les composants restent mis à jour avec l'état actuel, améliorant la synchronisation.

Modèle Méthode Template
- **Exemple**: Définir un algorithme de transition d'état générique.
- **Preuve**:
  - La structure de la FSM utilise ce modèle pour les transitions d'état, avec des étapes spécifiques implémentées dans des sous-classes (pages 10-17).
- **Avantage**: Favorise la réutilisation du code et impose une structure d'algorithme cohérente tout en permettant des personnalisations spécifiques.

Modèle Composite
- **Exemple**: Regroupement de diverses tâches et gestion en tant qu'entité unique.
- **Preuve**:
  - Mentionné dans la section "Infrastructure de gestion des tâches", qui met en évidence la nécessité d'une dépendance minimale entre l'application principale et les tâches (page 6).
- **Avantage**: Améliore la modularité et la gestion des séquences de tâches complexes.

Conclusion
Le document du projet soutient l'utilisation de plusieurs modèles de conception, principalement les modèles de machine à états, Singleton, Stratégie, Observateur, Méthode Template, et Composite. Ces modèles contribuent collectivement à la flexibilité, la réutilisabilité et la maintenabilité de la solution logicielle pour contrôler le robot GoPiGo3.