RAPPORT DE PROJET TEMPS REEL 4AE

DEPARTEMENT DE GENIE ÉLECTRIQUE ET INFORMATIQUE

Nom du binôme : Diani Badr et Housbani Soufiane

Enseignant de TP : Thibaud Lasguignes

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

1 Architecture fonctionnelle

Recensement des fonctions :

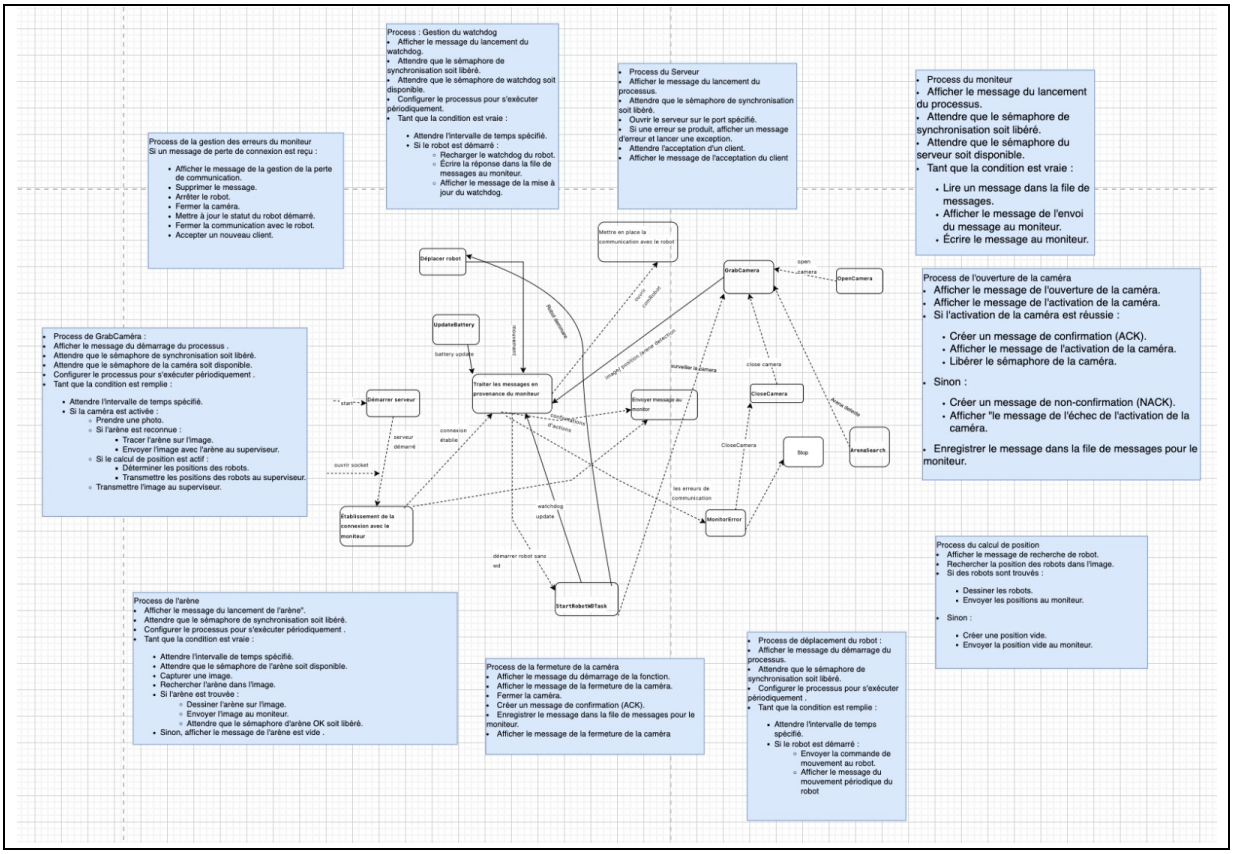
Nom de la fonction	Description du comportement	Entrées	Sorties
Envoyer_Message_au_moniteur	Envoie des messages au moniteur.	Message à envoyer	Confirmation de l'envoi ou erreur
Traiter_message_du_moniteur	Traite les messages reçus du moniteur.	Message reçu	Action en réponse au message traité
Démarrer_serveur	Lance le serveur pour établir la communication avec le moniteur.	Événement de démarrage	Serveur démarré ou message d'erreur
Établir_connexion_avec_moniteur	Établit la connexion avec le moniteur via un socket.	Demande de connexion	Connexion établie ou message d'erreur
Surveiller_communication_moniteur	Surveille la communication avec le moniteur pour détecter les pertes de connexion.	Statut de la communication	Message de perte de communication ou confirmation de surveillance
Démarrer_robot	Démarre le robot dans le mode demandé (avec ou sans watchdog).	Commande de démarrage	Statut du démarrage (succès/échec)
Transmettre_commande_robot	Transmet les commandes de mouvement au robot.	Commande de mouvement	Confirmation de la commande ou erreur
Mettre_a_jour_batterie_robot	Met à jour le niveau de batterie du robot et l'envoie au moniteur.	Niveau de batterie actuel	Niveau de batterie mis à jour
Démarrer_camera	Démarre la caméra suite à une demande du moniteur.	Demande de démarrage de la caméra	Confirmation de démarrage ou message d'erreur
Capturer_image	Capture une image à partir de la caméra et l'envoie au moniteur.	Fréquence de capture	Image capturée
Fermer_camera	Ferme la caméra suite à une demande du moniteur.	Demande de fermeture de la caméra	Confirmation de fermeture
Rechercher_arene	Recherche l'arène dans l'image capturée et envoie le résultat au moniteur.	Image à analyser	Résultat de la recherche (succès/échec)

Calculer_position_robot	Calcule la position du robot dans l'arène et l'envoie au moniteur.	Image à analyser	Position du robot ou (-1, -1) si non trouvé
Stopper_calcul_position	Stoppe le calcul de la position du robot suite à une demande du moniteur.	Demande d'arrêt du calcul	Confirmation de l'arrêt
Surveiller_communication_robot	Surveille la communication avec le robot pour détecter les pertes de connexion.	Statut de la communication	Message de perte de communication ou confirmation de surveillance

Table des exigences sur chaque fonction :

Numéro exigence système	Concerne la fonction	Commentaire
1	Démarrer_serveur	Serveur doit être démarré au lancement
2	Établir_connexion_avec_moniteur	Connexion doit être établie suite à une demande
3	Réception_message_moniteur	Tous les messages doivent être reçus
4	Traiter_message_du_moniteur	Messages doivent être traités en moins de 10 ms
5	Surveiller_communication_moniteur	Détecter et signaler la perte de communication
6	Surveiller_communication_moniteur	Stopper le robot et fermer les communications en cas de perte
7	Ouvrir_communication_robot	Ouvrir la communication avec le robot
9	Surveiller_communication_robot	Surveiller la communication avec le robot
10	Démarrer_robot	Démarrer le robot sans watchdog
11	Transmettre_commande_robot	Transmettre les commandes de mouvement en moins de 100 ms
12	Mettre_a_jour_batterie_robot	Mettre à jour le niveau de batterie toutes les 500 ms
13	Démarrer_camera	Démarrer la caméra suite à une demande
14	Capturer_image	Envoyer une image toutes les 100 ms
15	Fermer_camera	Fermer la caméra suite à une demande
16	Rechercher_arene	Rechercher l'arène et envoyer le résultat
17	Calculer_position_robot	Calculer et envoyer la position du robot
18	Stopper_calcul_position	Stopper le calcul de la position

Architecture fonctionnelle statique :



Définition de la séquence d'exécution des fonctions :

Comportement de chaque fonction

Envoyer_Message_au_moniteur: Cette fonction envoie un message au moniteur via un socket. Si l'envoi réussit, elle renvoie une confirmation. En cas d'échec, elle renvoie un message d'erreur.
Traiter_message_du_moniteur : fonction reçoit un message du moniteur et le traite en fonction du contenu. Les actions peuvent inclure le démarrage du robot, l'envoi d'une commande de mouvement, etc. Le résultat du traitement est ensuite envoyé au moniteur.
Démarrer_serveur : Cette fonction lance le serveur au démarrage du superviseur. Si le serveur démarre correctement, elle produit un événement "Serveur démarré". En cas d'échec, elle affiche un message d'erreur et arrête le superviseur.
Établir_connexion_avec_moniteur : Cette fonction établit une connexion avec le moniteur après la demande de connexion. Elle attend l'événement "Serveur démarré", puis ouvre un socket pour la communication.
Surveiller_communication_moniteur : fonction surveille la communication avec le moniteur. En cas de perte de communication, elle affiche un message d'erreur et initie des procédures de récupération, telles que l'arrêt du robot et la fermeture des connexions.
Démarrer_robot : Cette fonction démarre le robot en mode demandé (avec ou sans watchdog). Elle envoie une commande de démarrage au robot et attend une confirmation. En cas d'échec, elle envoie un message d'erreur au moniteur.
Transmettre_commande_robot : Cette fonction transmet des commandes de mouvement au robot. Elle lit la commande de mouvement et l'envoie au robot. La confirmation ou l'erreur est ensuite retournée.

Mettre_a_jour_batterie_robot : Cette fonction lit le niveau de batterie actuel du robot et met à jour cette information sur le moniteur toutes les 500 ms.
Démarrer_camera : Cette fonction démarre la caméra après réception d'une demande du moniteur. Elle envoie une confirmation de démarrage ou un message d'erreur en cas d'échec.
Capturer_image : Cette fonction capture des images à partir de la caméra et les envoie au moniteur toutes les 100 ms.
Fermer_camera : Cette fonction ferme la caméra après réception d'une demande du moniteur et envoie une confirmation de fermeture.
Rechercher_arene : Cette fonction recherche l'arène dans l'image capturée et envoie le résultat de la recherche au moniteur. En cas d'échec, un message d'erreur est envoyé.
Calculer_position_robot : Cette fonction calcule la position du robot dans l'arène en analysant l'image et envoie cette position au moniteur toutes les 100 ms. Si le robot n'est pas trouvé, elle envoie (-1, -1).
Stopper_calcul_position : Cette fonction stoppe le calcul de la position du robot après réception d'une demande du moniteur et envoie une confirmation d'arrêt.
Surveiller_communication_robot : Cette fonction surveille la communication avec le robot. En cas de perte de communication, elle envoie un message d'erreur et initie des procédures de récupération, telles que la fermeture des connexions.

Séquence globale d'exécution des fonctions (Vue comportementale - dynamique)

1. **Démarrage du superviseur**:
   Au lancement du superviseur, la fonction **Démarrer_serveur** est appelée pour initialiser et démarrer le serveur. Si le serveur démarre correctement, il produit l'événement "Serveur démarré". En cas d'échec, le superviseur affiche un message d'erreur et s'arrête.
2. **Établissement de la connexion**:
   Après le démarrage du serveur, la fonction **Établir_connexion_avec_moniteur** attend l'événement "Serveur démarré". Sur réception de cet événement, elle établit une connexion avec le moniteur en ouvrant un socket. Si la connexion est réussie, elle renvoie une confirmation. Sinon, un message d'erreur est envoyé.
3. **Communication avec le moniteur**:
   La fonction **Surveiller_communication_moniteur** surveille en continu la connexion avec le moniteur pour détecter toute perte de communication. La fonction **Traiter_message_du_moniteur** reçoit et traite les messages du moniteur. En fonction du contenu des messages, elle peut appeler d'autres fonctions telles que **Démarrer_robot** ou **Transmettre_commande_robot** pour exécuter les actions nécessaires.
4. **Commandes et supervision du robot**:
   La fonction **Démarrer_robot** est appelée pour démarrer le robot selon la commande reçue du moniteur. La fonction **Transmettre_commande_robot** envoie les commandes de mouvement au robot. La fonction **Mettre_a_jour_batterie_robot** met à jour le niveau de batterie du robot régulièrement et envoie cette information au moniteur.
5. **Gestion de la caméra**:
   La fonction **Démarrer_camera** démarre la caméra sur demande du moniteur. La fonction **Capturer_image** capture et envoie des images au moniteur à des intervalles réguliers. Lorsque le moniteur demande l'arrêt de la caméra, la fonction **Fermer_camera** est appelée pour fermer la caméra et envoyer une confirmation.
6. **Recherche et suivi du robot**:
   La fonction **Rechercher_arene** recherche l'arène dans les images capturées et envoie les résultats au moniteur. La fonction **Calculer_position_robot** calcule et envoie la position du robot au moniteur à intervalles réguliers. Si le moniteur demande l'arrêt du calcul de la

position, la fonction **Stopper_calcul_position** est appelée pour arrêter ce calcul et envoyer une confirmation.

7. **Surveillance et gestion des connexions**:
   Les fonctions **Surveiller_communication_moniteur** et **Surveiller_communication_robot** surveillent continuellement les communications avec le moniteur et le robot, respectivement. En cas de perte de communication, des messages d'erreur sont envoyés et des procédures de récupération sont initiées, telles que l'arrêt du robot et la fermeture des connexions, pour remettre le système dans un état sûr.

Cette séquence assure que toutes les fonctions sont exécutées de manière coordonnée pour garantir le bon fonctionnement du système de supervision du robot.

Architecture fonctionnelle dynamique :

Scénario 1 : Démarrage et Connexion

Démarrage du superviseur: Lorsque le superviseur est lancé, la fonction **Démarrer_serveur** est immédiatement appelée pour initialiser et démarrer le serveur. Ce processus implique la vérification et la préparation des ressources nécessaires pour l'exécution du serveur. Si le serveur démarre correctement, il produit un événement "Serveur démarré" qui signale que le serveur est prêt à accepter des connexions. En cas d'échec du démarrage, la fonction affiche un message d'erreur sur la console et arrête le superviseur pour éviter tout comportement imprévisible.
Établissement de la connexion: Une fois que le serveur est démarré, la fonction **Établir_connexion_avec_moniteur** prend le relais. Cette fonction attend l'événement "Serveur démarré" et, une fois cet événement reçu, elle établit une connexion avec le moniteur en ouvrant un socket. Cette étape permet de créer un canal de communication entre le superviseur et le moniteur. Si la connexion est établie avec succès, une confirmation est renvoyée. Sinon, un message d'erreur est généré pour informer de l'échec de la connexion.
Surveillance de la communication: La fonction **Surveiller_communication_moniteur** entre alors en action pour surveiller en continu la communication avec le moniteur. Cette surveillance continue permet de détecter toute perte de connexion avec le moniteur. En cas de perte de communication, un message d'erreur est affiché et des procédures de récupération sont initiées, telles que l'arrêt du robot et la fermeture des connexions, pour assurer la sécurité et la stabilité du système.

Scénario 2 : Commande et supervision du robot

Démarrage du robot: Lorsque le moniteur envoie une commande de démarrage du robot, la fonction **Traiter_message_du_moniteur** reçoit cette commande et appelle **Démarrer_robot**. Cette fonction envoie la commande de démarrage au robot, qui peut inclure des modes spécifiques tels que "avec watchdog" ou "sans watchdog". Après avoir envoyé la commande, la fonction attend une confirmation de démarrage du robot. Si le démarrage est réussi, une confirmation est envoyée au moniteur, sinon un message d'erreur est renvoyé.
Envoi de commandes de mouvement: Pour les commandes de mouvement, la fonction **Traiter_message_du_moniteur** reçoit des commandes spécifiques comme avancer, reculer, tourner à gauche ou à droite. Elle appelle alors **Transmettre_commande_robot** qui lit la commande de mouvement et l'envoie au robot. Après l'envoi de la commande, la fonction attend une confirmation du robot. Si la commande est exécutée avec succès, une confirmation est renvoyée au moniteur, sinon un message d'erreur est généré pour indiquer le problème.
Mise à jour du niveau de batterie: La surveillance continue de la batterie du robot est assurée par la fonction **Mettre_a_jour_batterie_robot**. Cette fonction est appelée périodiquement, toutes les 500 ms, pour lire le niveau de batterie actuel du robot et mettre à jour cette

information sur le moniteur. Cela permet à l'utilisateur de surveiller en temps réel l'état de la batterie du robot et de prendre des mesures préventives en cas de faible niveau de charge.

Scénario 3 : Gestion de la caméra

Démarrage de la caméra: Lorsqu'une demande de démarrage de la caméra est reçue du moniteur, la fonction **Traiter_message_du_moniteur** appelle **Démarrer_camera**. Cette fonction initialise et démarre la caméra, permettant ainsi au système de commencer la capture d'images. Une confirmation de démarrage est envoyée au moniteur pour indiquer que la caméra est opérationnelle. Si l'ouverture de la caméra échoue, un message d'erreur est envoyé au moniteur.
Capture et envoi d'images: Une fois la caméra démarrée, la fonction **Capturer_image** est appelée périodiquement pour capturer des images à intervalles réguliers, généralement toutes les 100 ms. Ces images sont ensuite envoyées au moniteur pour une surveillance visuelle en temps réel. Cela permet à l'utilisateur de voir ce que le robot voit et de prendre des décisions basées sur les données visuelles reçues.
Fermeture de la caméra: Lorsque le moniteur envoie une demande de fermeture de la caméra, la fonction **Traiter_message_du_moniteur** appelle **Fermer_camera**. Cette fonction ferme la caméra et envoie une confirmation de fermeture au moniteur pour indiquer que la caméra a bien été désactivée. Cela arrête la capture et l'envoi d'images, économisant ainsi des ressources système.

Scénario 4 : Recherche et suivi du robot

Recherche de l'arène: Pour la recherche de l'arène, la fonction **Traiter_message_du_moniteur** reçoit une demande de recherche et appelle **Rechercher_arene**. Cette fonction analyse les images capturées pour identifier les contours de l'arène dans laquelle le robot évolue. Une fois l'arène identifiée, le résultat de la recherche est envoyé au moniteur. En cas d'échec, un message d'erreur est envoyé pour signaler que l'arène n'a pas pu être trouvée.
Calcul de la position du robot: Pour le suivi du robot, la fonction **Traiter_message_du_moniteur** reçoit une demande de calcul de position et appelle **Calculer_position_robot**. Cette fonction analyse les images pour déterminer la position actuelle du robot dans l'arène. La position calculée est envoyée au moniteur toutes les 100 ms. Si le robot n'est pas trouvé, une position fictive (-1, -1) est envoyée pour indiquer l'échec de la localisation.
Arrêt du calcul de position: Si le moniteur envoie une demande pour arrêter le calcul de la position du robot, la fonction **Traiter_message_du_moniteur** appelle **Stopper_calcul_position**. Cette fonction arrête le calcul de position et envoie une confirmation au moniteur pour indiquer que le calcul a été stoppé. Cela permet de désactiver le suivi de position lorsque ce n'est plus nécessaire, économisant ainsi des ressources de traitement.
Ces scénarios et modèles dynamiques montrent comment les fonctions interagissent entre elles et comment les flux d'entrées/sorties permettent au système de remplir correctement sa mission et de gérer divers scénarios opérationnels.

2 Architectures physique

Le système d'intérêt est organisé en plusieurs composants abstraits, chacun ayant des rôles et responsabilités spécifiques. Le superviseur est le composant central, chargé d'orchestrer toutes les opérations, en gérant la communication entre le moniteur, le robot, et la caméra. Il est responsable de démarrer le serveur, d'établir et de maintenir les connexions, de surveiller les communications, et de traiter les commandes. Le moniteur, quant à lui, est l'interface

utilisateur qui permet de contrôler le robot et de surveiller son état en temps réel. Il envoie des commandes au superviseur et reçoit des mises à jour et des images. Le robot exécute les commandes de mouvement et rapporte son état, y compris le niveau de batterie et la position, au superviseur. Enfin, la caméra capture des images de l'environnement du robot pour la surveillance visuelle et la localisation. Ces composants abstraits interagissent de manière coordonnée pour assurer le bon fonctionnement du système et la réalisation des missions opérationnelles.
Les composants physiques du système incluent le Raspberry Pi avec Xenomai, le poste de travail (moniteur), le robot mobile, et la caméra fixée au-dessus de l'arène. Le Raspberry Pi, équipé de Xenomai, héberge le superviseur et assure des performances temps réel robustes. Il gère le serveur, la communication avec le moniteur et le robot, et la capture d'images. Le poste de travail sert d'interface utilisateur, où le moniteur est exécuté, permettant de coder, compiler, et exécuter le programme de supervision, tout en fournissant un point de commande pour les opérations de contrôle et de surveillance. Le robot mobile, équipé d'un microcontrôleur et d'une puce Xbee, exécute les commandes de mouvement et fournit des données de télémétrie. La caméra, montée au-dessus de l'arène, capture des images pour le suivi visuel et la localisation du robot. Ces composants physiques sont intégrés de manière à tirer parti des capacités temps réel de Xenomai, assurant une exécution déterministe et une coordination efficace des tâches critiques.

Choix et justification d'une organisation en constituants (découpage/regroupement des fonctions en tâches), caractérisation des tâches :

Nom de la tâche	**Rôle**	**Entrées**	**Sorties**	**Activation / Période**	**Priorité**
T_Envoyer_Message_au_moniteur	Envoie des messages au moniteur	Message à envoyer	Confirmation de l'envoi ou erreur	Sur demande	Haute
T_Traiter_message_du_moniteur	Traite les messages reçus du moniteur	Message reçu	Action en réponse au message traité	Sur réception	Haute
T_Démarrer_serveur	Lance le serveur pour établir la communication avec le moniteur	Événement de démarrage	Serveur démarré ou message d'erreur	Au démarrage	Très haute
T_Établir_connexion_avec_moniteur	Établit la connexion avec le moniteur via un socket	Demande de connexion	Connexion établie ou message d'erreur	Sur demande	Haute
T_Surveiller_communication_moniteur	Surveille la communication avec le moniteur pour détecter les pertes de connexion	Statut de la communication	Message de perte de communication ou confirmation de surveillance	Continu	Très haute

T_Démarrer_robot	Démarre le robot dans le mode demandé (avec ou sans watchdog)	Commande de démarrage	Statut du démarrage (succès/échec)	Sur demande	Haute
T_Transmettre_commande_robot	Transmet des commandes de mouvement au robot	Commande de mouvement	Confirmation de la commande ou erreur	Sur demande	Haute
T_Mettre_a_jour_batterie_robot	Met à jour le niveau de batterie du robot et l'envoie au moniteur	Niveau de batterie actuel	Niveau de batterie mis à jour	Périodique (500 ms)	Moyenne
T_Démarrer_camera	Démarre la caméra suite à une demande du moniteur	Demande de démarrage de la caméra	Confirmation de démarrage ou message d'erreur	Sur demande	Moyenne
T_Capturer_image	Capture une image à partir de la caméra et l'envoie au moniteur	Fréquence de capture	Image capturée	Périodique (100 ms)	Haute
T_Fermer_camera	Ferme la caméra suite à une demande du moniteur	Demande de fermeture de la caméra	Confirmation de fermeture	Sur demande	Moyenne
T_Rechercher_arene	Recherche l'arène dans l'image capturée et envoie le résultat au moniteur	Image à analyser	Résultat de la recherche (succès/échec)	Sur demande	Moyenne
T_Calculer_position_robot	Calcule la position du robot dans l'arène et l'envoie au moniteur	Image à analyser	Position du robot ou (-1, -1) si non trouvé	Périodique (100 ms)	Haute
T_Stopper_calcul_position	Stoppe le calcul de la position du robot suite à une demande du moniteur	Demande d'arrêt du calcul	Confirmation de l'arrêt	Sur demande	Moyenne
T_Surveiller_communication_robot	Surveille la communication avec le robot pour détecter les	Statut de la communication	Message de perte de communication ou confirmation	Continu	Très haute

	pertes de connexion		de surveillance		

Justification

T_Envoyer_Message_au_moniteur: Cette tâche est cruciale pour la communication bidirectionnelle entre le superviseur et le moniteur. Elle est activée sur demande lorsque le superviseur doit envoyer des mises à jour ou des confirmations au moniteur. La priorité élevée assure une réponse rapide aux commandes du moniteur.
T_Traiter_message_du_moniteur: Cette tâche traite les commandes reçues du moniteur. Elle doit répondre rapidement pour garantir une interaction fluide et réactive entre l'utilisateur et le robot, d'où sa haute priorité.
T_Démarrer_serveur: Essentielle pour initialiser le système, cette tâche doit être exécutée en premier avec une priorité très élevée pour établir la base de la communication.
T_Établir_connexion_avec_moniteur: Une fois le serveur démarré, cette tâche assure la connexion avec le moniteur, activée sur demande avec une haute priorité pour établir rapidement la communication.
T_Surveiller_communication_moniteur et **T_Surveiller_communication_robot**: Ces tâches de surveillance continue sont cruciales pour maintenir la fiabilité du système. Elles surveillent les connexions en continu avec une priorité très élevée pour détecter et réagir immédiatement en cas de perte de communication.
T_Démarrer_robot: Cette tâche démarre le robot en fonction des commandes reçues, avec une haute priorité pour assurer que le robot est prêt à exécuter les commandes rapidement.
T_Transmettre_commande_robot: La transmission des commandes de mouvement au robot doit être réactive pour garantir des mouvements fluides et précis, justifiant sa haute priorité.
T_Mettre_a_jour_batterie_robot: Mise à jour régulière du niveau de batterie pour le moniteur. La tâche est périodique avec une priorité moyenne car elle est moins critique mais toujours nécessaire pour une surveillance continue.
T_Démarrer_camera et **T_Fermer_camera**: Ces tâches gèrent l'état de la caméra, avec une priorité moyenne pour répondre aux commandes du moniteur de démarrer ou arrêter la capture d'images.
T_Capturer_image: Cette tâche capture les images de la caméra à une fréquence élevée, avec une priorité haute pour assurer que les images sont capturées et envoyées sans délai.
T_Rechercher_arene et **T_Calculer_position_robot**: Ces tâches analytiques sont essentielles pour la localisation et la navigation du robot. Elles sont activées sur demande ou périodiquement, avec des priorités adaptées pour garantir des mises à jour régulières et fiables de l'état de l'arène et de la position du robot.
T_Stopper_calcul_position: Cette tâche arrête le calcul de la position sur demande, avec une priorité moyenne car elle est nécessaire pour interrompre des opérations en cours de manière contrôlée.

Allocation des fonctions aux composants :

Tâche \ Fonction	Envoyer Message au moniteur	Traiter message du moniteur	Fonction i	Fonction j	Justification
T_Envoyer_Message_au_moniteur	X				Cette tâche est dédiée à l'envoi de messages au moniteur.
T_Traiter_message_du_moniteur		X			Cette tâche traite les messages reçus du moniteur.
T_Démarrer_serveur			X		Tâche nécessaire pour initialiser le serveur.
T_Établir_connexion_avec_moniteur			X		Établit la connexion avec

					le moniteur via un socket.
T_Surveiller_communication_moniteur			X		Surveille en continu la communication avec le moniteur.
T_Démarrer_robot				X	Démarre le robot en fonction des commandes reçues.
T_Transmettre_commande_robot				X	Transmet les commandes de mouvement au robot.
T_Mettre_a_jour_batterie_robot				X	Met à jour le niveau de batterie du robot.
T_Démarrer_camera				X	Démarre la caméra sur demande du moniteur.
T_Capturer_image				X	Capture des images à partir de la caméra périodiquement.
T_Fermer_camera				X	Ferme la caméra sur demande du moniteur.
T_Rechercher_arene				X	Recherche l'arène dans les images capturées.
T_Calculer_position_robot				X	Calcule la position du robot dans l'arène.
T_Stopper_calcul_position				X	Stoppe le calcul de la position sur demande.
T_Surveiller_communication_robot				X	Surveille la communication avec le robot en continu.

Table des exigences sur chaque tâche :

Numéro exigence système	Concerne la tâche	Commentaire
1	task_Lancer_Serveur	Le lancement du serveur doit être réalisé au démarrage du superviseur.
2	task_Connexion_Moniteur	La connexion entre le moniteur et le superviseur doit être établie suite à la demande.
3	task_Recevoir_Messages_Moniteur	Tous les messages envoyés depuis le moniteur doivent être réceptionnés par le superviseur.
4	task_Traiter_Messages_Moniteur	Le superviseur doit traiter les messages à envoyer au moniteur en moins de 10 ms.
5	task_Detecter_Perte_Communication	Le superviseur doit détecter la perte de communication avec le moniteur.
6	task_Gerer_Perte_Communication	En cas de perte de communication, diverses actions

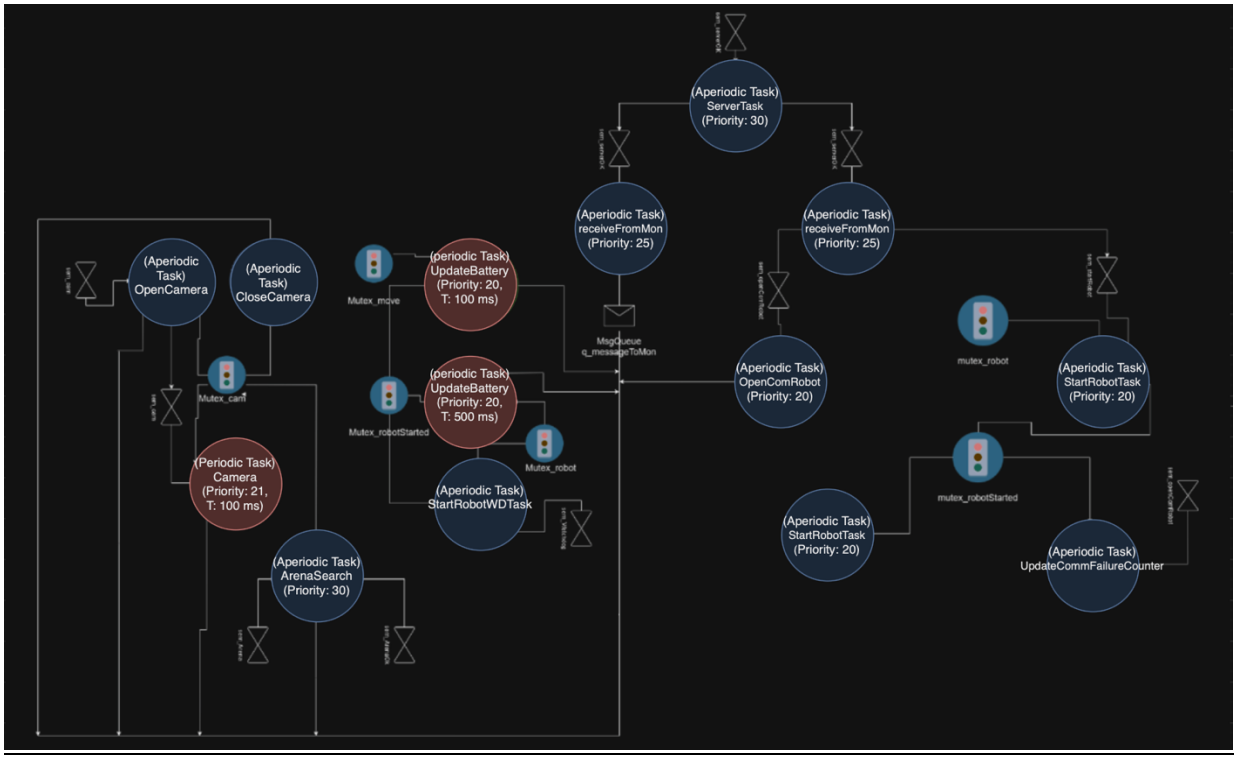
		doivent être prises pour remettre le système à l'état initial.
7	task_Ouvrir_Communication_Robot	La communication entre le superviseur et le robot doit être ouverte dès que la communication avec le moniteur est établie.
9	task_Surveiller_Communication_Robot	En cas de perte de communication avec le robot, le superviseur doit envoyer un message spécifique au moniteur.
10	task_Demarrer_Robot	Le superviseur doit démarrer le robot en mode sans watchdog à la demande de l'utilisateur.
11	task_Transmettre_Commandes_Robot	Le superviseur doit transmettre les commandes de mouvement au robot en moins de 100 ms.
12	task_Updater_Niveau_Batterie	Le superviseur doit mettre à jour le niveau de la batterie du robot sur le moniteur toutes les 500 ms.
13	task_Demarrer_Camera	Le superviseur doit démarrer la caméra suite à une demande provenant du moniteur.
14	task_Capturer_Image_Camera	Dès que la caméra est ouverte, le superviseur doit envoyer une image au moniteur toutes les 100 ms.
15	task_Fermer_Camera	Le superviseur doit fermer la caméra suite à une demande provenant du moniteur.
16	task_Rechercher_Arene	Le superviseur doit rechercher l'arène suite à une demande de l'utilisateur et renvoyer une image annotée.
17	task_Calculer_Position_Robot	Suite à une demande de l'utilisateur, le superviseur doit calculer la position du robot et envoyer la position au moniteur.
18	task_Stopper_Calcul_Position	Le superviseur doit stopper le calcul de la position du robot suite à une demande de l'utilisateur.

Choix et justification des moyens de communication et de synchronisation

Donnée échangée entre des tâches	Mode de communication (variable globale, messageQueue, …) et caractérisation (Id, nom, taille, timeout, ..)	Protection (ou pas) par Mutex et Id du Mutex	Justification
Communication avec le robot	Sémaphore sem_openComRobot	Protégé par Mutex, Id : mutex_robot	Protège l'accès concurrent par plusieurs tâches
Commandes de mouvement	Variable partagée, Id : move	Protégé par Mutex, Id : mutex_move	Synchronise les commandes de mouvement avec l'état du robot.
État du robot	Variable partagée, Id : robotStarted	Protégé par Mutex, Id : mutex_robotStarted	Protège l'accès concurrent à l'état du robot par les tâches.
État de la caméra	Variable partagée : OpenComRobot	Protégé par Mutex, Id: mutex_camera	Protège l'accès concurrent à l'état de la caméra par les tâches : OpenCameraTask, CloseCameraTask et GrabCameraTask
Etat de l'arène	Variable partagée, Id : arene_temp	Non protégé	Faible risque de conflit
Données de l'arène	Variable partagée: Arena	Protégé par Mutex, Id: mutex_camera	Protège l'accès concurrent par GrabCamera et ArenaTask
ID du Watchdog	Variable partagée, Id : watchdog_id	Non protégé	Faible risque de conflit.
Compteur des erreurs	Variable partagée: Counter	Non protégé	Pas de risque de conflit
Message à envoyer au moniteur	messageQueue q_MessageToMon	Non protégé	Pas de risque de conflit

Tâches à synchroniser	Événement à signaler	Id du sémaphore binaire qui représente l'événement	Justification
Toutes les tâches	Synchronisation initiale	sem_barrier	Assurer que toutes les tâches sont démarrées avant d'exécuter des opérations critiques.
ServerTask	Connexion établie	sem_serverOK	Indique que le serveur est prêt à accepter la connexion
ReceiveFromMonTask	Connexion établie	sem_serverOK	Assure que les messages ne sont reçus qu'après la connexion.
SendToMonTask	Connexion établie	sem_serverOK	Assure que les messages ne sont envoyés qu'après la connexion.
OpenComRobot	Prêt pour ouvrir la communication avec le robot	sem_openComRobot	Permet de commencer l'ouverture de la communication série.
StartRobotTask	Robot démarré	sem_startRobot	Signal pour démarrer le robot avec ou sans Watchdog
UpdateBattery	Mise à jour du niveau de batterie	sem_battery	Envoie périodique du niveau de batterie au moniteur.
StartRobotWDTask	Mise à jour du Watchdog	sem_startRobotWD	Assure la recharge périodique du Watchdog.

GrabCamera	Caméra ouverte	sem_camera	Assure que les images sont capturées après l'ouverture de la caméra.
ArenaTask	Arena Validé	sem_ArenaDraw	Indique si l'arène est validée ou pas
ArenaTask	Prêt pour détection de l'arène	sem_ArenaSearch	Signal pour détecter l'arène dans les images capturés



3 Codage et livraisons incrémentales

Stratégie de codage et d'intégration

Nous avons mis en œuvre les fonctionnalités progressivement, en veillant à ce que chaque fonctionnalité soit opérationnelle avant d'en ajouter d'autres, en commençant par les tâches les plus simples. Voici la liste des tâches implémentées dans l'ordre chronologique : UpdateBattery, OpenCamera, GrabCamera, CloseCamera, ArenaTask, StartRobotWDTask.
Pour la tâche GrabCamera, nous avons commencé par son activation. Ensuite, pour calculer la position du robot, nous avons préféré le faire dans cette fonction car le calcul de la position utilise principalement les données de la caméra. En revanche, nous avons choisi d'ajouter une fonction ArenaTask pour plus de clarté, car l'arène est captée une fois et reste inchangée, contrairement au calcul de la position qui doit se faire en continu.

4 Analyse et validation du logiciel livré par rapport aux exigences

Numéro exigence	Description de l'exigence	État
1	Le lancement du serveur doit être réalisé au démarrage du superviseur. En cas d'échec dudémarrage du serveur, un message doit êtreaffiché sur la console de lancement de l'application qui doit signaler le problème et le superviseur doit s'arrêter.	Réalisée et fonctionnelle
2	La connexion entre le moniteur et le superviseur doit être établie suite à la demande de connexion de l'utilisateur	Réalisée et fonctionnelle
3	Tous les messages envoyés depuis le moniteur doivent être réceptionnés par lesuperviseur.	Réalisée et fonctionnelle
4	Le superviseur doit traiter les messages àenvoyer au moniteur en moins de 10 ms.	Réalisée et fonctionnelle
5	Le superviseur doit détecter la perte de communication avec le moniteur. En cas deperte de la communication un message doitêtre affiché sur la console de lancement du superviseur.	Réalisée et fonctionnelle
6	En cas de perte de communication entre le superviseur et le moniteur, le superviseur doit stopper le robot, la communication avec le robot, fermer le serveur et déconnecter lacaméra afin de revenir dans le même état qu'au démarrage du superviseur.	Non Réalisée
7	Dès que la communication avec le moniteur est en place, la communication entre le superviseur et le robot doit être ouverte. Si la communication est active, il faut envoyer un message d'acquittement au moniteur. Encas d'échec, un message le signalant est renvoyé au moniteur.	Réalisée et fonctionnelle
8	Lorsque la communication entre le robot et le superviseur est perdue, le superviseur doitenvoyer un message spécifique au moniteur.Le superviseur doit alors fermer la communication entre le robot et le superviseur et se remettre dans un état initial permettant de relancer la communication.	Non Réalisée
9	Lorsque l'utilisateur demande, via le moniteur, le démarrage sans watchdog, le superviseur doit démarrer le robot dans ce mode. En cas de succès, un message d'acquittement est retourné au moniteur. Encas d'échec, un message indiquant l'échec esttransmis au moniteur.	Réalisée et fonctionnelle

5 Commentaires

L'implémentation incrémentale des fonctionnalités nous a permis de tester et de valider chaque composant individuellement avant de les intégrer. Cependant, L'implémentation de certaines tâches, notamment ArenaTask, nous a causé quelques problèmes de détection du robot. En effet, la détection du robot n'est plus comme avant. Après avoir sélectionné l'arène, la tâche sur le robot commence à clignoter. Nous avons essayé de changer la fréquence de détection de l'image, mais cela n'a pas résolu le problème.