Rapport de projet temps réel 4AE

Nom du binôme : POUYANNE Thibault, BOGEN Haakon

Enseignant de TP : Benjamin Lesage

3 Architecture fonctionnelle

Recensement des fonctions :

Nom de la fonction	Description du comportement	Entrées	Sorties
Lancer le server	S'exécute eu lancement du superviseur. Envoie un message d'erreur à la console au cas d'échec.	Lancer serveur	Message d'acquittement ou d'échec
Etablir socket	Appelée par l'utilisateur qu'après le lancement du serveur. Envoie un message de réussite.	Serveur démarré	Connexion établie
Traiter et envoyer les messages	Reçoit des messages et les envoie au moniteur en moins de 10ms	Inputstream	Outputstream
Gérer perte com. mon.-superviseur	Détecte la perte de communication entre le superviseur et le moniteur. Affiche un message d'erreur sur la console. Arrêter le robot, fermer la caméra et le serveur.		Messages pour arrêter le robot, fermer la caméra et le serveur.
Ouvrir communication robot-superviseur	Lance dès que la connexion entre le moniteur et le superviseur a été faite. Renvoie un message d'acquittement ou d'erreur.	Connexion établie	Messages d'acquittement
Gérer perte de communication avec le robot	Détecter la perte de communication avec le robot. Envoyer un message spécifique au monitor, arrêter la communication avec le robot et se remettre dans un état permettant de rétablir la communication.		Message connexion perdue robot
Démarrer le robot sans watchdog	Le superviseur démarre le robot et renvoie un message au monitor selon la réussite du démarrage.	Demande démarrage	Messages d'acquittement ou d'erreur.
Commander robot	Reçoit des ordres de l'utilisateur et commande le robot à faire, - Le mouvement donné, en moins de 100ms.	Message de commande, robot démarré	Etat de batterie (si commandé)

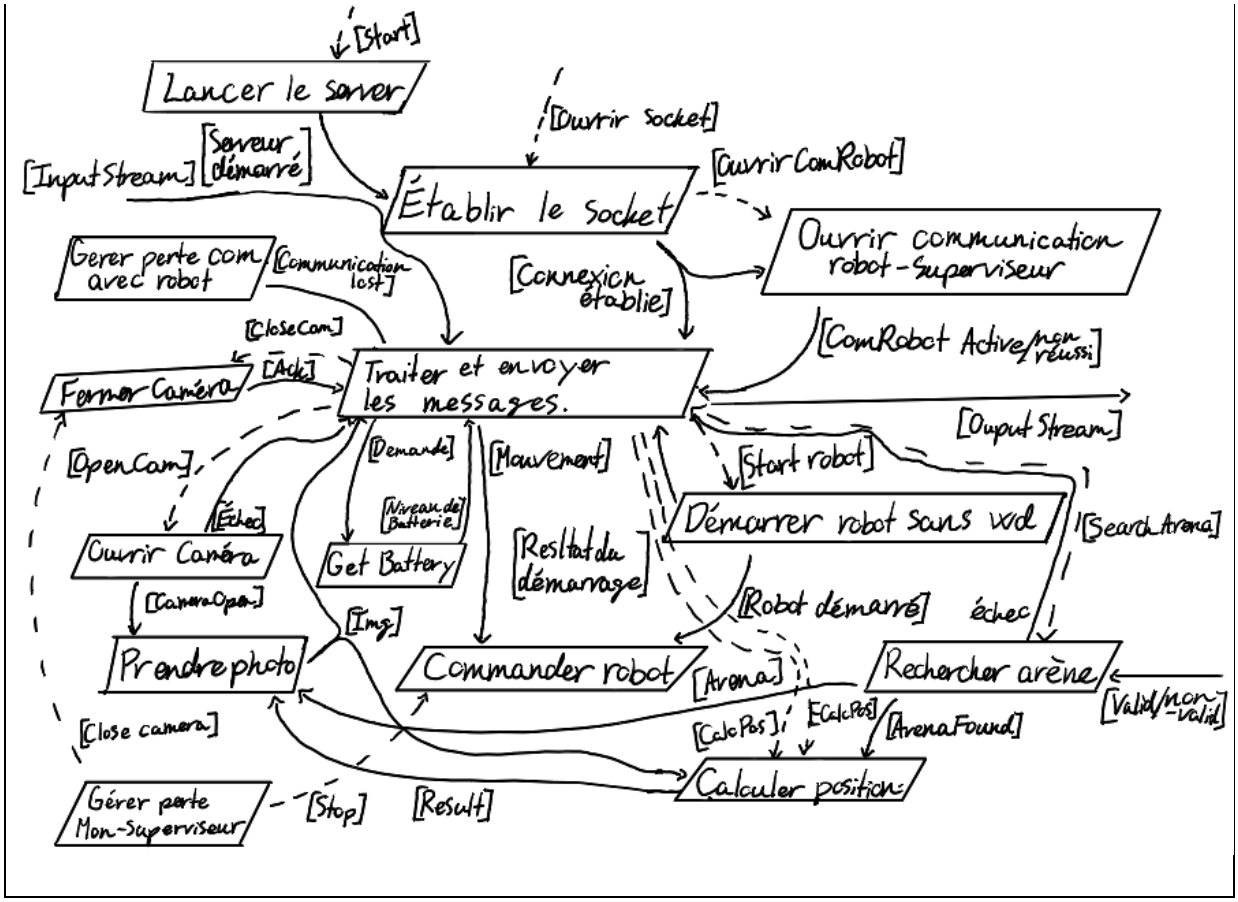
Get battery	Envoi toute les 500ms le niveau de la batterie du robot	Demande de niveau	Niveau de batterie
Ouvrir caméra	Ouvrir la caméra après une demande par l'utilisateur. Renvoyer un message d'acquittement ou d'échec.	Demande d'ouverture	Message d'acquittment ou d'échec.
Prendre photo	Prendre et envoyer une image au monitor. Dessiner l'arène et la position sur l'image en si le calcul de position est actif et si l'arène est trouvée.	Caméra ouverte, position calculée, arène trouvée	Message contenant une image.
Fermer caméra	Fermer la caméra, si elle est ouverte et envoyer un message d'acquittement.	Caméra ouverte, demande de fermeture	Message d'acquittement ou d'échec.
Rechercher arène	Arrêter l'envoie périodique des messages et trouver l'arène. Une fois l'arène trouvé, demande de vérification par l'utilisateur.	Img, Validé	Arena found, arena,
Calculer position	Calcule la position du robot à la suite de la demande de l'utilisateur toutes les 100ms. Si le calcul n'est pas réussi, il renvoie (-1,-1). Si l'utilisateur demande d'arrêter de calculer la position, le superviseur revient à ne qu'envoyer l'image sans la position.	Calculer position, Arena found, Img	Position

Table des exigences sur chaque fonction : (optionnel)

Numéro exigence système	**Concerne la fonction**	**Commentaire**
1	Lancer le serveur	
2	Etablir le socket	
3, 4	Traiter et envoyer les messages	La fonction créée gère la réception et l'envoie de message entre le moniteur et le superviseur.
5, 6	Gérer perte de communication entre le superviseur et le monitor	Cette fonction s'occupe de la détection de la perte de la communication et la reprise de communication entre le superviseur et le monitor.

7	Ouvrir communication robot superviseur	
9	Gérer perte de communication avec le robot	
10	Démarrer sans watchdog	
11	Commander robot	
12	Get Battery	
13	Ouvrir caméra	
14	Prendre photo	Cette fonction prend et envoi les images captées par la caméra au monitor. Elle ajoute aussi la position calculée et l'arène si cela a été demandé.
15	Fermer caméra	
16	Rechercher arène	
17, 18	Calculer position	

Architecture fonctionnelle statique :



4 Architectures physique

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
Th_server	Démarrer le serveur et créé la connexion entre le moniteur et le superviseur	Depuis le moniteur	Vers th_sendToMon	Apériodique	30
Th_sendToMon	Récupère les messages à envoyer du superviseur au moniteur et les envois	Depuis les threads – 5 entrées	Vers le moniteur	Apériodique	22
Th_receiveFromMon	Gère le flux de messages depuis le moniteur	Depuis le moniteur	Libère des sémaphores ou écrit dans des variables globales	Apériodique	25
Th_open_ComRobot	Etablir la connexion Xbee avec le robot	En attente d'un sémaphore, et réponse du robot	Vers th_sendToMon et vers robot	Apériodique	20
Th_startRobot	Démarre le robot sans watchdog	En attente d'un sémaphore,	Vers th_sendToMon et vers robot	Apériodique	20

		et réponse du robot			
Th_move	Analyse le mouvement reçu du moniteur et envoie l'ordre de mouvement au robot	Reçoit du moniteur le mouvement à envoyer au robot	Envoie le mouvement au robot	Périodique	20
Th_battery	Récupère auprès du robot et envoie à th_sendToMon l'état de la batterie	Reçoit le message GetBattery et la réponse du robot	Envoie un message au robot d'acquisition de la batterie. Envoie au moniteur l'état de la batterie	Périodique	23
Th_camera_p	Récupère les images périodiquement, trouve l'arène, et la position du robot et l'envoie au moniteur via th_sendToMon	Depuis th_receiveFromMon, récupère les images du robot	Envoie des ordres pour acquérir l'image au robot	Périodique	26

Allocation des fonctions aux composants :

Nom de la fonction	**Tâche associée**	
Lancer le server	Th_server	
Etablir socket	Th_server	
Traiter et envoyer les messages	Th_receiveFromMon Th_sendToMon	Nous avons créé une fonction qui est gérée dans 2 threads.
Gérer perte com. mon.-superviseur	Non implémentée. Voir partie *Stratégie de codage et d'intégration*	
Ouvrir communication robot-superviseur	Th_open_ComRobot	
Gérer perte de communication avec le robot	Non implémentée. Voir partie *Stratégie de codage et d'intégration*	
Démarrer le robot sans wd	Th_startRobot	
Commander robot	Th_move	-
Ouvrir caméra	Th_receiveFromMon	
Prendre photo	Th_camera_p	
Fermer caméra	Th_receiveFromMon	
Rechercher arène	Th_camera_p	
Calculer position	Th_camera_p	

Nous sommes passés par cette étape au début du projet sans en laisser une trace. Dans ce rapport, nous montrons seulement les tâches liées entre elles.

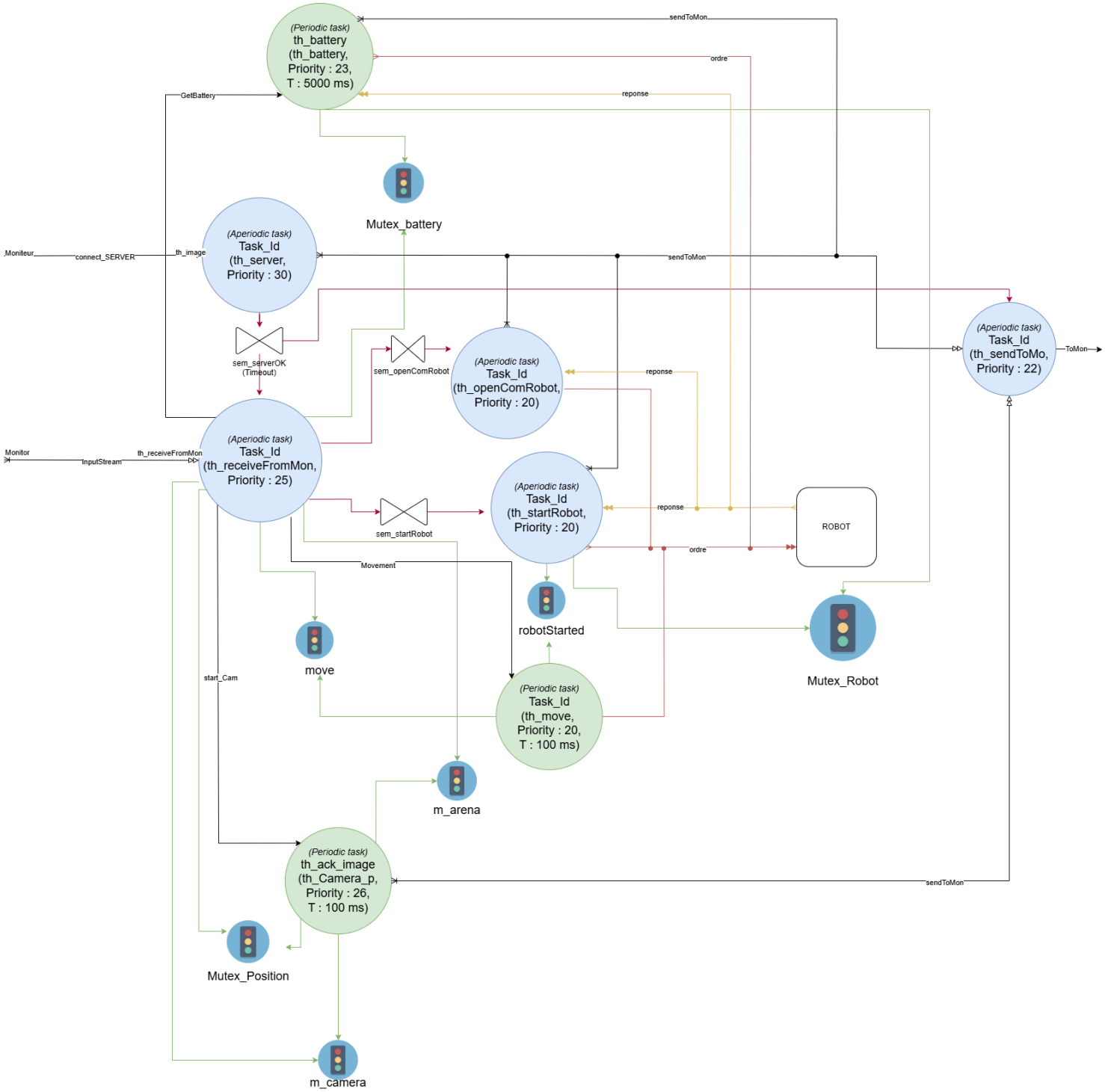
Table des exigences sur chaque tâche : (optionnel)

Nous n’avons pas inclus les exigences non réalisées.

Numéro exigence système	Concerne la tâche	Commentaire
1, 2	Th_server	
4	Th_sendToMon	
3, 13, 15	Th_receiveFromMon	
7	Th_open_ComRobot	
10	Th_startRobot	Démarrage sans watchdog
11	Th_move	
12	Th_battery	
14, 16, 17, 18	Th_camera_p	

Choix et justification des moyens de communication et de synchronisation

Nous avons utilisé des variables globales pour activer certaines tâches. Nous aurions pu utiliser des sémaphores, mais nous avions besoin de certaines informations contenues dans ces variables globales afin de lancer les bonnes fonctions dans les threads. Il nous ait paru plus simple l’utilisation de ces variables globales protégées par des mutex.
Un sémaphore jouant le rôle de buffer d’attente est aussi mis en place, lançant l’exécution de toutes les tâches au même moment.



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
Image, arène, position	messageQueue	Non	On écrit les messages à envoyer dans la queue qui est géré par la tâche th_sendToMon.
Démarrage des fonctions	Variable globale	Mutex : Mutex_batterie	On ne veut pas que les variables globales ne

d'acquisition de batterie, d'image, et de recherche de position et d'arènes		Mutex_camera Mutex_arene Mutex_position	soient lues et écrites au même moment. On les protège par un Mutex. On aurait également pu utiliser des sémaphores à la place de certaines de ces variables.

Tâches à synchroniser	Événement à signaler	Id du sémaphore binaire qui représente l'événement	Justification
Début des tâches => fin initialisation		Non	On veut savoir l'état d'initialisation des tâches avant de les démarrer. Les tâches attendent donc la libération d'un sémaphore (barrier).
Th_sendToMon et th_receiveFromMon	Serveur mis en place et connecté		

5 Codage et livraisons incrémentales

Diagramme d'architecture physique revu (si nécessaire) en fonction des contraintes de plateforme :

Pour effectuer notre diagramme des tâches nous avons commencé par regarder les threads déjà réalisés dans le code. Nous avons construit notre architecture physique en fonction de celle déjà réalisée.

Stratégie de codage et d'intégration

Nous avons décidé de commencer notre codage par l'acquisition de l'état de la batterie. Une fois cela effectué, nous avons continué sur l'allumage et la fermeture de la caméra sur demande, et l'envoi des images sur le moniteur périodiquement.
Puis, nous avons effectué la recherche et le dessin de l'arène que nous affichons périodiquement sur le moniteur.
Enfin, nous avons implémenté la recherche de robot dans l'arène et nous envoyons seulement la position de notre robot.
Nous avons eu beaucoup de problèmes techniques (déconnexions intempestives du robot, non connexion au superviseur…) lors de ce BE ce qui nous a empêché de réaliser le code pour les autres exigences.

Validation :

Nous n'avons pas eu le temps de vérifier le temps d'exécution des fonctions dans les threads. Par exemple, pour l'envoi d'une image de moins de 100ms, nous pensions mesurer le temps d'exécution de chaque fonction appelée dans le thread image. C'est-à-dire les fonctions :
- camera -> Grab()
- DrawArena()
- SearchArena()
- SearchRobot()
- DrawAllRobots()

Il faut être sur que l'enchainement de certaines de ces fonctions ne dépasse pas 100ms, afin que ce soit le timer placé au début du thread qui imposera la cadence de 100ms.

De même pour la validation des vérifications de l'état de connexion entre le superviseur, le moniteur et le robot.
Pour la vérification de l'état de la connexion entre le robot et le superviseur (exigence 9):
Nous pensions vérifier l'état de la connexion dans tous les threads utilisant le robot :
Th_move, th_startRobot, th_open_ComRobot, th_battery, th_camera_p (et th_watchdog si nous l'avions fait). Pour chacun des threads : si la connexion est perdue, c'est-à-dire l'échec de la fonction *ping* de la classe Robot 3 fois d'affilée, la tâche ferme la connexion avec le robot.

Pour la vérification de l'état de connexion entre le superviseur et le moniteur (exigence 5 et 6) :
On effectue une vérification similaire dans tous les threads utilisant l'accès au moniteur.

Pour le démarrage du watchdog (exigence 10) : on envoie le message *StartWithWatchdog* de la classe Robot.

Le partage du travail s'est fait de la manière suivante :
Nous avons tous les deux réfléchis à l'architecture fonctionnelle et statique. Puis Haakon a codé la partie d'acquisition de la batterie et l'ouverture de la caméra. Thibault a codé l'acquisition de l'image et la recherche de l'arène et de la position. Nous avons travaillé ensemble pour chaque partie.
Pour le rapport : Haakon a mis au propre les schémas et a écrit la partie sur l'architecture fonctionnelle. Thibault a mis au propre les schémas et a écrit la partie architecture physique.

Les livrables :
Nous fournissons notre code écrit dans les fichiers « *task.cpp* » et « *task.h* ».
Ainsi que ce rapport expliquant notre avancée dans le projet.

6 Analyse et validation du logiciel livré par rapport aux exigences

Exigence	Description de l’exigence	Etat
1	Déjà implémenté par les professeurs.	Implémenté et fonctionnelle
2	Déjà implémenté par les professeurs.	Implémenté et fonctionnelle
3	Déjà implémenté par les professeurs.	Implémenté et fonctionnelle
4	Déjà implémenté par les professeurs.	Implémenté et fonctionnelle
5		Implémentation réfléchis
6		Implémentation réfléchis
7	Déjà implémenté par les professeurs.	Implémenté et fonctionnelle
9		Implémentation réfléchis
10	Déjà implémenté par les professeurs.	Implémenté et fonctionnelle
11	Déjà implémenté par les professeurs.	Implémenté et fonctionnelle
12	Le niveau de la batterie s’effectue 5 sec après la demande depuis le moniteur.	Implémenté et fonctionnelle
13	Le démarrage de la caméra s’effectue.	Implémenté et fonctionnelle
14	Les images s’affichent rapidement.	Implémenté et fonctionnelle
15	On peut stopper l'envoi des images.	Implémenté et fonctionnelle
16	On recherche l’arène et on la dessine sur l’image avant de l’envoyer. Puis on attend la validation de l’utilisateur avant de retourner sur un mode d’envoi périodique des images avec le dessin de l’arène.	Implémenté et fonctionnelle
17	On arrive à envoyer dessiner et envoyer la position du robot choisi.	Implémenté et fonctionnelle

18	On peut arrêter l’envoi de la position périodique du robot.	Implémenté et fonctionnelle

8 Commentaires

Comme déjà décrit plus haut nous avons eu des problèmes de connexions avec les robots. Nous ne pouvions pas avancer et tester notre code et cela nous a fait perdre beaucoup de temps. Nous n’avons donc pas réussi à implémenter l’ensemble des exigences.