Rapport de projet De Stijl 2.0

version 5 mars 2019

Mettez ici vos noms et les parties sur lesquelles vous avez travaillées : conception, robot, vidéo, mission, intégration, rédaction du compte-rendu
Exemple : P.-E. Hladik (conception, robot, rédaction du compte-rendu)

— Ce qu'il faut faire —

Remplacez tous les textes en bleu et supprimer les textes en rouge

Le rapport est à rendre en pdf et à déposer sur moodle avant le 31 mars 2019 23h59 : `https://moodle.insa-toulouse.fr/mod/assign/view.php?id=33071`

Vous devez aussi rendre les fichiers `tasks.h` et `tasks.cpp`.

Vous pouvez utiliser word ou un autre logiciel d'édition pour rédiger ce rapport, par contre vous devez **impérativement respecter la structure ci-après et le rendre au format pdf**.

Critères d'évaluation :
- Qualité rédactionnelle,
- Exhaustivité et justesse des règles de codage,
- Qualité de la conception (clarté, respect de la syntaxe, exhaustivité, justesse),
- Qualité des explications,
- Respect des règles dans la production du code.

Compétences évaluées :
- rédaction et communication sur un dossier de conception,
- concevoir une application concurrente temps réel,
- analyser une conception,
- passer d'un modèle de conception à une implémentation,
- écriture de code et utilisation de primitives au niveau système.

1 Conception

Mettez dans cette partie tous les éléments de votre conception en particulier vos diagrammes AADL (vue globale du système) et les diagrammes d'activité (détails des threads). Cette partie doit être auto-suffisante pour comprendre votre application.

Pour faciliter la lecture des schémas, vous allez présenter votre conception en trois parties, l'une focalisée sur la communication entre le moniteur et le superviseur, la seconde consacrée au traitement vidéo et la troisième au contrôle du robot.

Si vous le souhaitez, au lieu de dessiner vos diagrammes sous un éditeur, vous pouvez joindre un scan de vos schémas — ils doivent être lisibles et propres.

1.1 Diagramme fonctionnel général

Mettez ici un diagramme fonctionnel qui présente les principaux blocs de votre conception. Pour cela, inspirez vous du diagramme ci-dessous (fig. 1.1) en indiquant pour chaque groupe de threads les données et ports partagés. La figure 1.1 a été réalisée à partir du document de conception. **Vous devez absolument conserver le découpage en trois groupes de threads (`th_group_gestion_moniteur`, `th_group_vision`, `th_group_gestion_robot`).**

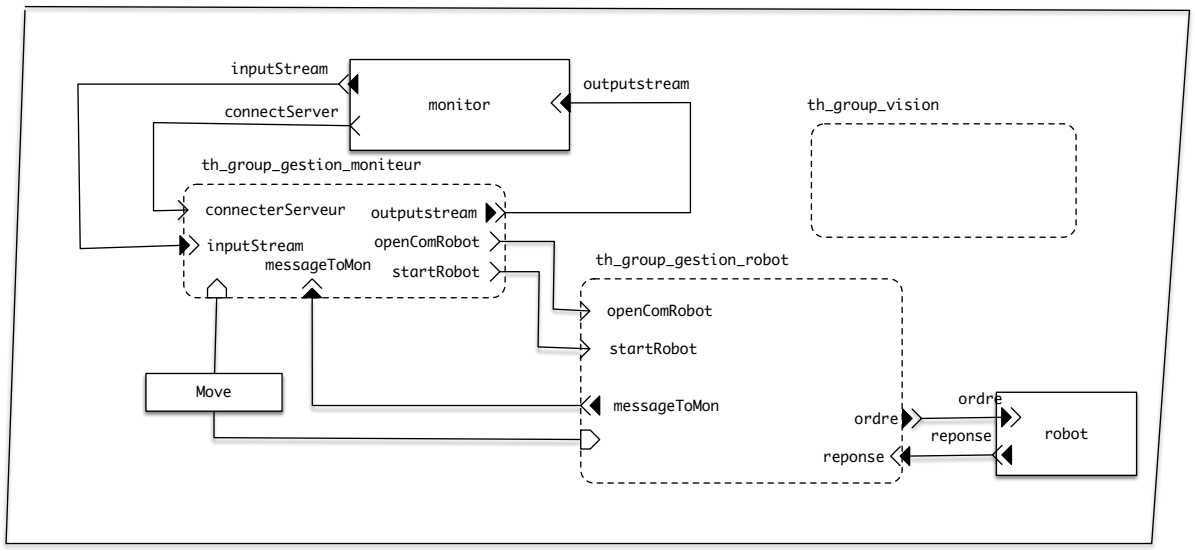

Fig. 1: Diagramme fonctionnel du système

1.2 Groupe de threads gestion du moniteur

Placez ici :

— le diagramme fonctionnel en AADL décrivant le groupe de threads de gestion du moniteur (voir exemple de la figure 1.2.1 réalisée à partir du dossier de conception),

— remplir le tableau 3 pour expliquer le rôle de chacun des threads,

— les diagrammes d'activité de chaque thread de ce groupe.

Décrivez tous les éléments (paramètres, variables, etc.) qui vous semblent pertinents pour comprendre les diagrammes.

1.2.1 Diagramme fonctionnel du groupe gestion du moniteur

Exemple de diagramme fonctionnel pour le groupe de thread de gestion du moniteur. Mettez à jour ce diagramme avec votre conception.

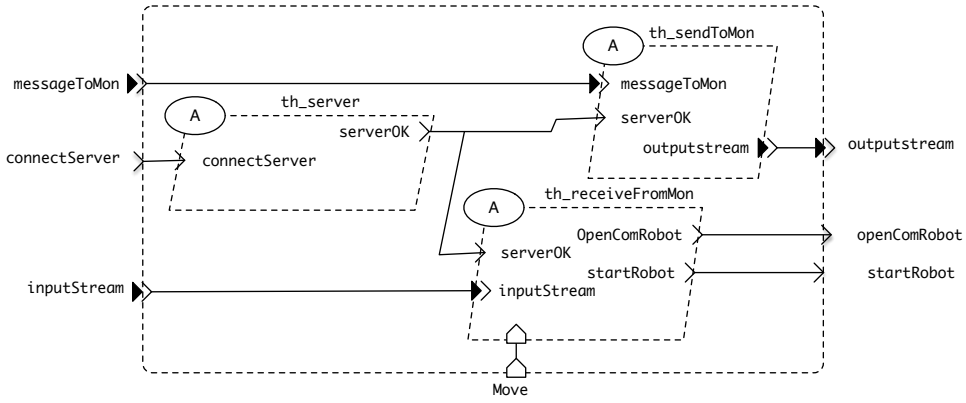

Fig. 2: Diagramme fonctionnel du groupe de threads gestion du moniteur

1.2.2 Description des threads du groupe gestion du moniteur

Remplissez le tableau ci-dessous pour expliquer le rôle de chaque thread et donner son niveau de priorité.

Tab. 1: Description des threads du groupe `th_group_gestion_moniteur`

Nom du thread	Rôle	Priorité
tCommuniquer	Prend en charge les messages entrants depuis le moniteur	25
tEnvoyer	Envoi l'ensemble des messages du superviseur au moniteur	30
...	...	...

1.2.3 Diagrammes d'activité du groupe gestion du moniteur

Décrivez le comportement de chacun de vos threads avec des diagrammes d'activité. Apportez les explications qui vous semblent nécessaires pour comprendre votre conception. A titre d'exemple les diagrammes fonctionnels tirés du document de conception sont remis.

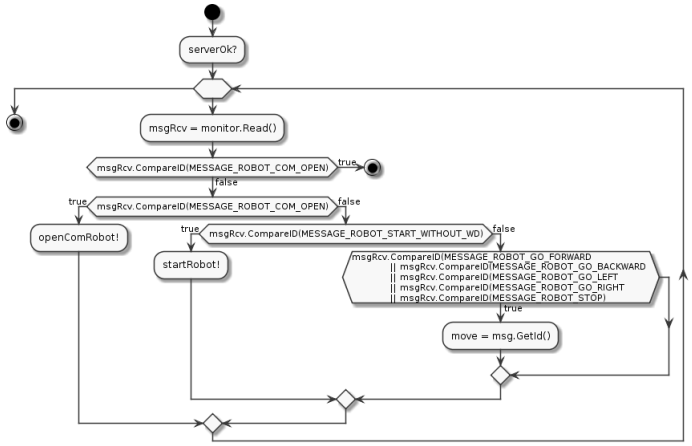

Fig. 3: Diagramme d'activité du thread `th_receiveFromMon`

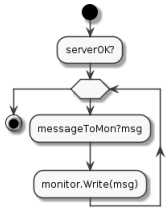

Fig. 4: Diagramme d'activité du thread `th_sendToMon`

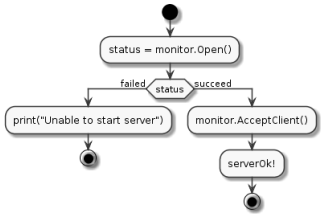

Fig. 5: Diagramme d'activité du thread `th_server`

1.3 Groupe de threads vision

1.3.1 Diagramme fonctionnel du groupe vision

Ajoutez le diagramme fonctionnel du groupe de threads de vision.

1.3.2 Description des threads du groupe vision

Remplissez le tableau ci-dessous pour expliquer le rôle de chaque thread et donner son niveau de priorité.

Tab. 2: Description des threads du groupe `th_group_vision`

Nom du thread	Rôle	Priorité
...	...	...

1.3.3 Diagrammes d'activité du groupe vision

Décrivez le comportement de chacun de vos threads avec des diagrammes d'activité. Apportez les explications qui vous semblent nécessaires pour comprendre votre conception.

1.4 Groupe de threads gestion du robot

1.4.1 Diagramme fonctionnel du groupe gestion robot

Ajoutez le diagramme fonctionnel du groupe de threads de gestion du robot.

1.4.2 Description des threads du groupe gestion robot

Remplissez le tableau ci-dessous pour expliquer le rôle de chaque thread et donner son niveau de priorité.

Tab. 3: Description des threads du groupe `th_group_gestion_robot`

Nom du thread	Rôle	Priorité
...	...	...

1.4.3 Diagrammes d'activité du groupe robot

Décrivez le comportement de chacun de vos threads avec des diagrammes d'activité. Apportez les explications qui vous semblent nécessaires pour comprendre votre conception.

2 Analyse et validation de la conception

Pour chacune des fonctionnalités attendues, indiquez si elle a été réalisée. Ajoutez d'éventuelles remarques de conception si le fonctionnement n'est pas exactement celui attendu et

justifiez le respect des propriétés temporelles. Vous n'êtes pas obligé de répondre à ces éléments pour les fonctionnalités réalisées dans la conception préliminaire (ces fonctionnalités sont indiquées dans la suite par un astérisque). Si vous ne souhaitez pas apporter de détails supplémentaires pour ces fonctionnalités, supprimez les de la liste ci-dessous.

2.1 Fonctionnalité 1*

Description : Le lancement du serveur doit être réalisé au démarrage du superviseur. En cas d'échec du démarrage du serveur, un message textuel doit être affiché sur la console de lancement de l'application. Ce message doit signaler le problème et le superviseur doit s'arrêter.

Réalisation : Fait en partie. La fonctionnalité est impléméntée sauf pour l'arrêt suite à l'échec de démarrage du serveur. L'implémentation actuelle stoppe la tâche en charge du démarrage du serveur, mais pas l'ensemble de l'application. Après discussion avec le client, la version actuelle est suffisante.

2.2 Fonctionnalité 2*

Description : La connexion entre le moniteur et le superviseur (via le socket) doit être établie suite à la demande de connexion de l'utilisateur.

Réalisation : indiquez si la fonctionnalité est traitée et s'il y a des remarques particulières.

2.3 Fonctionnalité 3*

Description : Tous les messages envoyés depuis le moniteur doivent être réceptionnés par le superviseur.

Réalisation : indiquez si la fonctionnalité est traitée et s'il y a des remarques particulières.

2.4 Fonctionnalité 4*

Description : L'application superviseur doit être capable d'envoyer les messages au moniteur (via le serveur) avec un délai d'au plus 10 ms.

Réalisation : indiquez si la fonctionnalité est traitée et s'il y a des remarques particulières.

2.5 Fonctionnalité 5

Description : Le superviseur doit détecter la perte de communication avec le moniteur. En cas de perte de la communication un message doit être affiché sur la console de lancement du superviseur.

Réalisation : indiquez si la fonctionnalité est traitée et s'il y a des remarques particulières.

2.6 Fonctionnalité 6

Description : En cas de perte de communication entre le superviseur et moniteur, il faut stopper le robot, la communication avec le robot, fermer le serveur et déconnecter la caméra afin de revenir dans le même état qu'au démarrage du superviseur.

Réalisation : indiquez si la fonctionnalité est traitée et s'il y a des remarques particulières.

2.7 Fonctionnalité 7*

Description : 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. En cas d'échec, un message le signalant est renvoyé au moniteur.

Réalisation : indiquez si la fonctionnalité est traitée et s'il y a des remarques particulières.

2.8 Fonctionnalité 8

Description : La communication entre le robot et le superviseur doit être surveillée par un mécanisme de compteur afin de détecter une perte du médium de communication.

Réalisation : indiquez si la fonctionnalité est traitée et s'il y a des remarques particulières.

2.9 Fonctionnalité 9

Description : Lorsque la communication entre le robot et le superviseur est perdue, un message spécifique doit être envoyé au moniteur. Le système doit fermer la communication entre le robot et le superviseur et se remettre dans un état initial permettant de relancer la communication.

Réalisation : indiquez si la fonctionnalité est traitée et s'il y a des remarques particulières.

2.10 Fonctionnalité 10*

Description : Lorsque l'utilisateur demande, via le moniteur, le démarrage sans watchdog, le robot doit démarrer dans ce mode. En cas de succès, un message d'acquittement est retourné au moniteur. En cas d'échec, un message indiquant l'échec est transmis au moniteur.

Réalisation : indiquez si la fonctionnalité est traitée et s'il y a des remarques particulières.

2.11 Fonctionnalité 11

Description : Lorsque l'utilisateur demande, via le moniteur, le démarrage avec watchdog, le robot doit démarrer dans ce mode. Un message d'acquittement est retourné au moniteur. En cas d'échec, un message indiquant l'échec est transmis au moniteur.

Une fois le démarrage effectué, le robot doit rester vivant en envoyant régulièrement le message de rechargement du watchdog.

Réalisation : indiquez si la fonctionnalité est traitée et s'il y a des remarques particulières.

2.12 Fonctionnalité 12*

Description : Lorsque qu'un ordre de mouvement est reçu par le superviseur, le robot doit réaliser ce déplacement en moins de 100 ms.

Réalisation : Cette fonctionnalité a été implémentée à l'aide d'une tâche qui envoie toutes les 100 ms un ordre de mouvement au robot une fois que celui-ci est démarré (tâche th_move). Cette implémentation ne garantit pas que le temps soit inférieur à 100 ms entre la réception du message et sa prise en compte par le robot. En effet, le temps de traitement de la réception par la tâche th_receiveFromMon, le temps de traitement de la tâche th_move et celui de l'envoi de l'ordre via le Xbee ne sont pas considérés. Afin de réduire ces délais, les priorités de th_receiveFromMon et de th_move sont élevés mais ne permettent pas de garantir l'exigence de temps. Augmenter la fréquence de la tâche th_move permettrait de tenir cette contrainte, mais risque de surcharger la communication avec le robot (une version avec l'envoi de l'ordre que s'il a changé serait souhaitable). Finalement, une version asynchrone (attente d'un événement-donnée entre th_receiveFromMon et de th_move) aurait été préférable. Cependant, après discussion avec le client, la version périodique à 100 ms est cependant validée.

2.13 Fonctionnalité 13

Description : Le niveau de la batterie du robot doit être mis à jour toutes les 500 ms sur le moniteur.

Réalisation : indiquez si la fonctionnalité est traitée et s'il y a des remarques particulières.

2.14 Fonctionnalité 14

Description : La caméra doit être démarrée suite à une demande provenant du moniteur. Si l'ouverture de la caméra a échoué, il faut envoyer un message au moniteur.

Réalisation : indiquez si la fonctionnalité est traitée et s'il y a des remarques particulières.

2.15 Fonctionnalité 15

Description : Dès que la caméra est ouverte, une image doit être envoyée au moniteur toutes les 100 ms.

Réalisation : indiquez si la fonctionnalité est traitée et s'il y a des remarques particulières.

2.16 Fonctionnalité 16

Description : La caméra doit être fermée suite à une demande provenant du moniteur. Un message doit être envoyé au moniteur pour signifier l'acquittement de la demande. L'envoi périodique des images doit alors être stoppé.

Réalisation : indiquez si la fonctionnalité est traitée et s'il y a des remarques particulières.

2.17 Fonctionnalité 17

Description : Suite à une demande de recherche de l'arène, le superviseur doit stopper l'envoi périodique des images, faire la recherche de l'arène et renvoyer une image sur laquelle est dessinée cette arène. Si aucune arène n'est trouvée un message d'échec est envoyé.

L'utilisateur doit ensuite valider visuellement via le moniteur si l'arène a bien été trouvée. L'utilisateur peut :

- valider l'arène : dans ce cas, le superviseur doit sauvegarder l'arène trouvée (pour l'utiliser ultérieurement) puis retourner dans son mode d'envoi périodique des images en ajoutant à l'image l'arène dessinée.
- annuler la recherche : dans ce cas, le superviseur doit simplement retourner dans son mode d'envoi périodique des images et invalider la recherche.

Réalisation : indiquez si la fonctionnalité est traitée et s'il y a des remarques particulières.

2.18 Fonctionnalité 18

Description : Suite à une demande de l'utilisateur de calculer la position du robot, le superviseur doit calculer cette position, dessiner sur l'image le résultat et envoyer un message au moniteur avec la position toutes les 100 ms. Si le robot n'a pas été trouvé, un message de position est envoyé avec une position (-1,-1).

Réalisation : indiquez si la fonctionnalité est traitée et s'il y a des remarques particulières.

2.19 Fonctionnalité 19

Description : Suite à une demande de l'utilisateur de stopper le calcul de la position du robot, le superviseur doit rebasculer dans un mode d'envoi de l'image sans le calcul de la position.

Réalisation : indiquez si la fonctionnalité est traitée et s'il y a des remarques particulières.

3 Transformation AADL vers Xenomai

Cette section est consacrée à la méthode pour passer d'un modèle AADL à un code sous Xenomai. Pour chacun des éléments AADL, vous expliquerez **comment vous l'avez traduit en code** et quels **services de Xenomai** vous avez utilisés **en expliquant ce qu'ils font**. Chaque élément devra être illustré avec des **extraits de code de votre projet**.

3.1 Thread

3.1.1 Instanciation et démarrage

Expliquer comment vous implémentez sous Xenomai l'instanciation et le démarrage d'un thread AADL.

Exemple de réponse : Chaque thread a été implémenté par un `RT_TASK` déclarés dans le fichier `tasks.h`. La création de la tâche se fait à l'aide du service `rt_task_create` et son démarrage à l'aide de `rt_task_start`. Toutes les tâches sont crées dans la méthode `init` de `tasks.cpp` et démarrées dans la méthode `run`.

Par exemple, pour la tâche `th_server`, sa déclaration est faite ligne 73 dans le fichier `tasks.h`

```
RT_TASK th_server;
```

sa création ligne 102 de `tasks.cpp` lors de l'appel de

```
rt_task_create(&th_server, "th_server", 0, PRIORITY_TSERVER, 0)
```

et son démarrage ligne 146 avec

```
rt_task_start(&th_server, (void(*)(void*)) & Tasks::ServerTask, this)
```

3.1.2 Code à exécuter

Comment se fait le lien sous Xenomai entre le thread et le traitement à exécuter.

3.1.3 Niveau de priorités

Expliquer comment vous fixez sous Xenomai le niveau de priorité d'un thread AADL.

3.1.4 Activation périodique

Expliquer comment vous rendez périodique l'activation d'un thread AADL sous Xenomai.

3.1.5 Activation événementielle

Expliquer les moyens mis en œuvre dans l'implémentation sous Xenomai pour gérer les activations événementielles d'un thread AADL.

3.2 Port d'événement

3.2.1 Instanciation

Comment avez-vous instancié un port d'événement ?

3.2.2 Envoi d'un événement

Quels services ont été employés pour signaler un événement ?

3.2.3 Réception d'un événement

Comment se fait l'attente d'un événement ?

3.3 Donnée partagée

3.3.1 Instanciation

Quelle structure instancie une donnée partagée ?

3.3.2 Accès en lecture et écriture

Comment garantissez-vous sous Xenomai l'accès à une donnée partagée ?

3.4 Ports d'événement-données

3.4.1 Instanciation

Donnez la solution retenue pour implémenter un port d'événement-données avec Xenomai.

3.4.2 Envoi d'une donnée

Quels services avez-vous employés pour envoyer des données ?

3.4.3 Réception d'une donnée

Quels services avez-vous employés pour recevoir des données ?