Rapport de projet temps réel 4AE

Nom du binôme : JUMIN Noël, MARCÉ Clément

Enseignant de TP :

— Ce qu’il faut faire —

**Consignes**

Le rapport est à rendre en pdf et en word..

Vous devez :

1) l’envoyer par mail à votre encadrant de TP au plus tard une semaine après votre dernière séance de TP, et

2) le déposer sous moodle. Le nom du rapport portera les deux noms du binome.

Vous devez aussi rendre votre code (uniquement les fichiers que vous avez écrits ou modifiés) sous la forme d’une archive (zip ou tar). Le nom du zip portera les deux noms du binôme.

**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 et justifications.

**Compétences évaluées :**

— rédaction et communication sur un dossier de conception

— concevoir une application concurrente temps réel : analyse fonctionnelle et proposition d’une architecture en tâches

— passer d’un modèle de conception à une implémentation

— écriture de code et utilisation des services d’un système d’exploitation temps réel

— vérification et validation

NB : pour le rapport final, vous pourrez supprimer tout ce qui est écrit en gris dans le document

**Résumé**

Ce rapport a pour but d’expliquer le déroulement du bureau d’étude « Robot Mobile » du binôme Noël JUMIN et Clément MARCÉ. Il s’agit d’un projet s’inscrivant dans le cursus d’ingénieur Automatique Électronique de l’INSA de Toulouse et plus particulièrement de l’enseignement de Temps Réel en 4ème année. L’objectif consiste en la réalisation d’une entité de supervision des opérations d’un petit véhicule-robot.

Le travail que nous avons fournis à cette occasion est détaillé dans ce document. Cela commence par la découverte du sujet et des besoins, ainsi que la prise en main des ressources matérielles et logicielles à disposition. La partie spécification et conception de notre solution est ensuite évoquée et schématisée. Après cela, nous nous intéressons à la réalisation et à l’implémentation de cette solution sur le système. Ceci nous mène finalement vers une conclusion et une critique sur le travail effectué.

**Abstract**

The purpose of this report is to explain the progress of the « Mobile Robot » project carried out by Noël JUMIN and Clément MARCÉ. This project is part of the Automatic Electronics engineering cursus at INSA Toulouse, and more specifically the Real Time course in 4th year. The aim was to create an entity to supervise the operations of a small robotic vehicle.

The work we did on this occasion is detailed in this document. It starts with finding out about the subject and the requirements, as well as familiarising ourselves with the hardware and software resources available. The specification and design of our solution is then discussed and diagrammed. We then move on to the development and implementation of the solution on the system. This finally leads us to a conclusion and a critique of the work carried out.

**Sommaire**

[I. Introduction 5](#_Toc165737234)

[II. Architecture fonctionnelle 6](#_Toc165737235)

[III. Architectures physiques 7](#_Toc165737236)

[IV. Codage et livraisons incrémentales 10](#_Toc165737237)

[V. Analyse et validation du logiciel livré par rapport aux exigences 11](#_Toc165737238)

[VI. Commentaires et conclusion 12](#_Toc165737239)

# Introduction

Dans le cadre de la formation d’ingénieur Automatique Électronique de l’INSA de Toulouse et de son enseignement en systèmes temps réel, nous avons eu l’occasion de prendre part au bureau d’étude « Robot Mobile ».

L’objectif principal de ce projet consiste en la réalisation d’une unité de supervision des opérations d’un robot-véhicule. Cette solution technique doit d’abord être conçue à partir des besoins de supervision établis. Elle devra ensuite être programmée puis implémentée dans le système physique.

Les ressources à disposition sont :

* Un Rasperrby Pi 3 avec l’OS Xenomai basé sur Linux et un module de communication Xbee. Il s’agit du composant qui viendra accueillir le code de notre superviseur et assurera sa communication avec le robot. La carte dispose également d’un module caméra.
* Un robot véhicule avec un module de communication Xbee et un microcontrôleur. Les fonctions de déplacement du robot sont préalablement codées.
* Une arène dans laquelle circulera le robot.
* Un PC opérateur équipé d’un moniteur faisant office d’IHM pour l’envois de consignes au robot. C’est également l’ordinateur qui servira au développement du superviseur grâce à l’IDE Netbeans.
* Un cahier des charges des tâches à superviser.

Cette étude est réalisée en 4 séances de TP et couvre un vaste panel d’activités de l’ingénieur. En commençant par la prise en main d’un nouvel environnement de travail et l’établissement de besoins, il sollicite également les phases de conception et de développement d’une solution technique, jusqu’à sa livraison au client.

Il met finalement en place différents aspects théoriques et pratiques, tant sur des bases acquises comme la programmation C que sur de nouveaux concepts fondamentaux des systèmes temps réel.

Nous allons par la suite explorer ces axes en entrant plus en détails dans le cœur du projet.

# Architecture fonctionnelle

Dans un premier temps, il nous a fallu nous intéresser au cahier des charges, c’est-à-dire la description des besoins du client. Il s’agit concrètement de la définition des tâches que devra réaliser notre superviseur.

Afin de prendre en main ce cahier des charges et débuter une démarche d’organisation de notre travail, nous avons regroupé les besoins du projet en fonctions.

Recensement des fonctions :

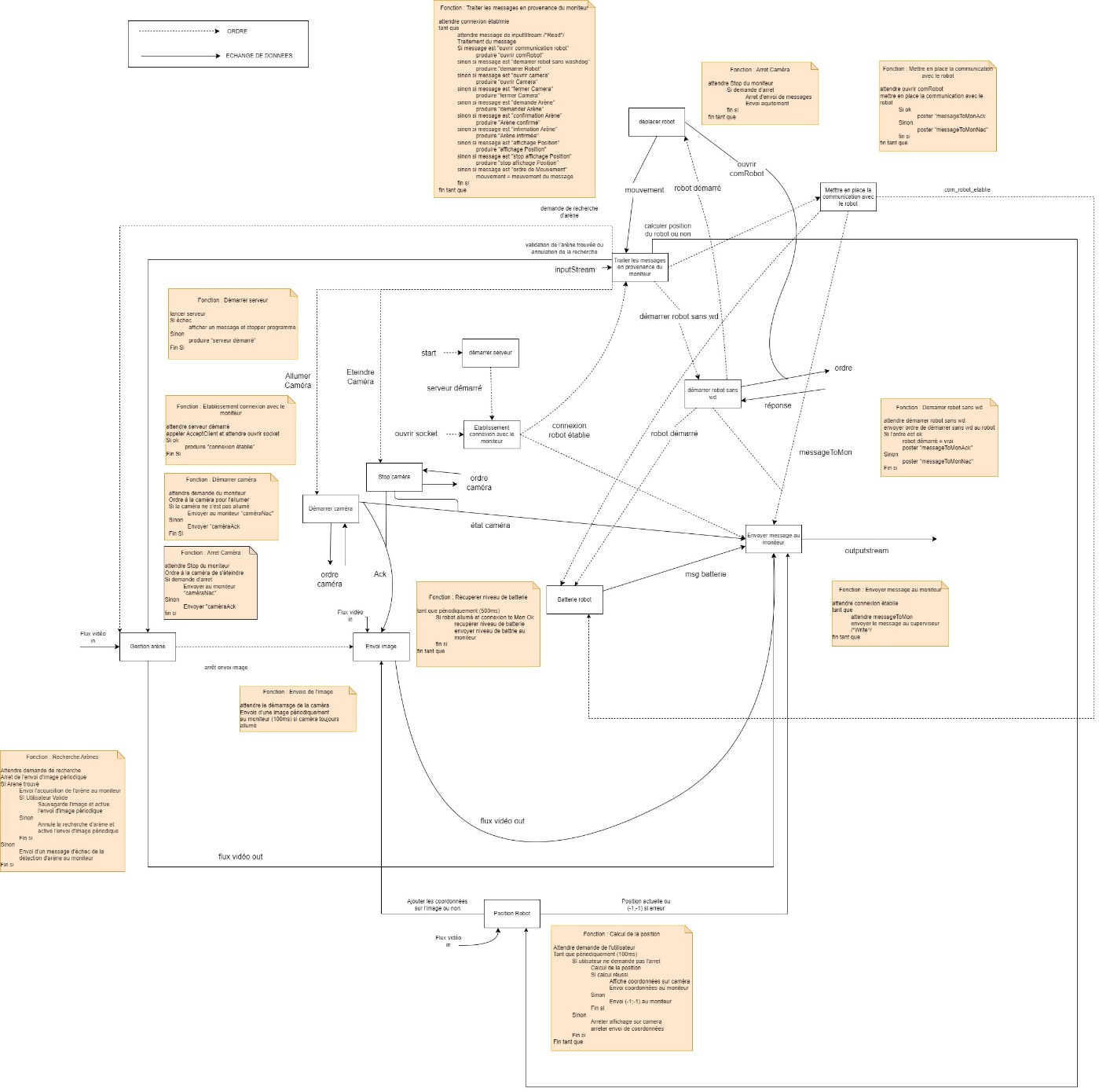
|  |  |  |  |
| --- | --- | --- | --- |
| Nom de la fonction | Description du comportement | Entrées | Sorties |
|  |  |  |  |
| Envoyer\_Message\_au\_moniteur |  |  |  |
| Traiter\_message\_du\_moniteur |  |  |  |
| … |  |  |  |
|  |  |  |  |

Ce travail préliminaire nous permet de mettre en lumière des blocs fonctionnels que nous devrons réaliser pour remplir notre mission.

C’est un bon moyen de segmenter les tâches, ce qui permettra de coder et valider leur fonctionnement séparément, étape par étape (branches git différentes, répartition des tâches dans l’équipe).

Il s’agit également d’une base pour visualiser les différents liens entre ces tâches et les éléments du système qui entrent en jeux dans chacune de leur réalisation. C’est ce que nous avons fais par la suite avec le diagramme d’architecture fonctionnelle.

Architecture fonctionnelle statique :



Une fois ce travail réalisé, nous avons à présent une excellente vision des tâches de notre futur superviseur et de la façon dont elles s’articuleront entre les différents éléments du système.

Ce premier concept reste néanmoins théorique et ne considère que les besoins du client à proprement parler. Pour s’approcher d’une solution faisable, il est nécessaire d’intégrer cette première étude dans le cadre de nos outils et moyens de réalisation de la solution.

# Architectures physiques

À présent, nous allons faire le lien entre les tâches théoriques évoquées précédemment et nos outils de travail.

Comme évoqué en introduction, nous avons à disposition une Raspberry Pi dans laquelle nous allons implémenter notre programme superviseur. Le code en question est développé en C, tout en ayant recours à l’API Xenomai qui permet d’utiliser des fonctions typiquement orientées temps réel.

Entre autres, ces fonctions vont nous permettre de mettre en place la base de l’ordonnancement des tâches en temps réel ; les sémaphores et les mutex. Les sémaphores nous serviront à synchroniser l’accès à une tâches par les différents éléments du système, tandis que les mutex serviront à bloquer l’accès à une variable par plusieurs threads simultanément.

Pour mettre en place ce réseau de gestion d’accès aux tâches et de communication entre elles, il est important de définir pour chacune leur élément d’activation (sémaphore ou mutex) et leur priorité d’exécution.

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

À l’aide du tableau suivant, identifier et caractériser un ensemble de composants (tâches) et de leurs interfaces logiques (entrées et sorties)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nom de la tâche | Rôle | Entrées | Sorties | Activation / Période | Priorité |
|  |  |  |  |  |  |
| T\_Envoyer\_Message\_au\_moniteur |  |  |  |  |  |
| T\_Traiter\_message\_du\_moniteur |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Allocation des fonctions aux composants :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tâche \ Fonction | Envoyer\_Message\_au\_moniteur | Envoyer\_Message\_au\_moniteur | Fonction i | Fonction j | Justification |
|  |  |  |  |  |  |
| T\_Envoyer\_Message\_au\_moniteur |  |  |  |  |  |
| T\_Traiter\_message\_du\_moniteur |  |  |  |  |  |
| … |  |  |  |  |  |
|  |  |  |  |  |  |

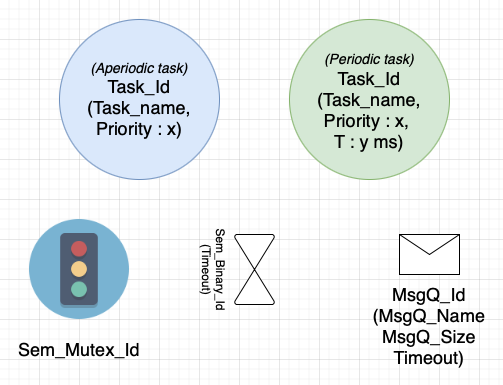
Exemple :

Une fois que vous avez vos fonctions, vous vous posez des questions :

- si par exemple vous effectuez deux traitements à la même fréquence, vous pouvez faire une seule et même tache périodique, sinon il en faudra deux ;

- si une tâche est activée dès qu’elle reçoit des informations, alors elle ne sera pas périodique, mais sera (par exemple) en attente active (avec un timeout bloquant) sur réception d’une message queue

Initier le diagramme d’architecture en positionnant les composants et leurs entrées/sorties.



Choix et justification des moyens de communication et de synchronisation

Consiste à identifier les moyens de communication et de synchronisation entre les tâches.

|  |  |  |  |
| --- | --- | --- | --- |
| Donnée échangée entre des tâches  (voir diagramme d’architecture logique statique) | Mode de communication (variable globale, messageQueue, …) et caractérisation  (Id, nom, taille, timeout, ..) | Protection (ou pas) par Mutex et Id du Mutex | Justification |
|  |  |  |  |
| Niveau de batterie | Variable Gobale :  Niveau\_Batterie | Mutex SemM\_Batterie | Accédée par plusieurs tâches (envoi moniteur, monitoring batterie…) |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Par exemple, si le niveau de batterie est partagé en lecture ou en écriture entre plusieurs tâches, on peut choisir de le représenter par une variable globale Niveau\_Batterie, protégée par un mutex SemM\_Batterie si on redoute des accès concurrents à la variable.

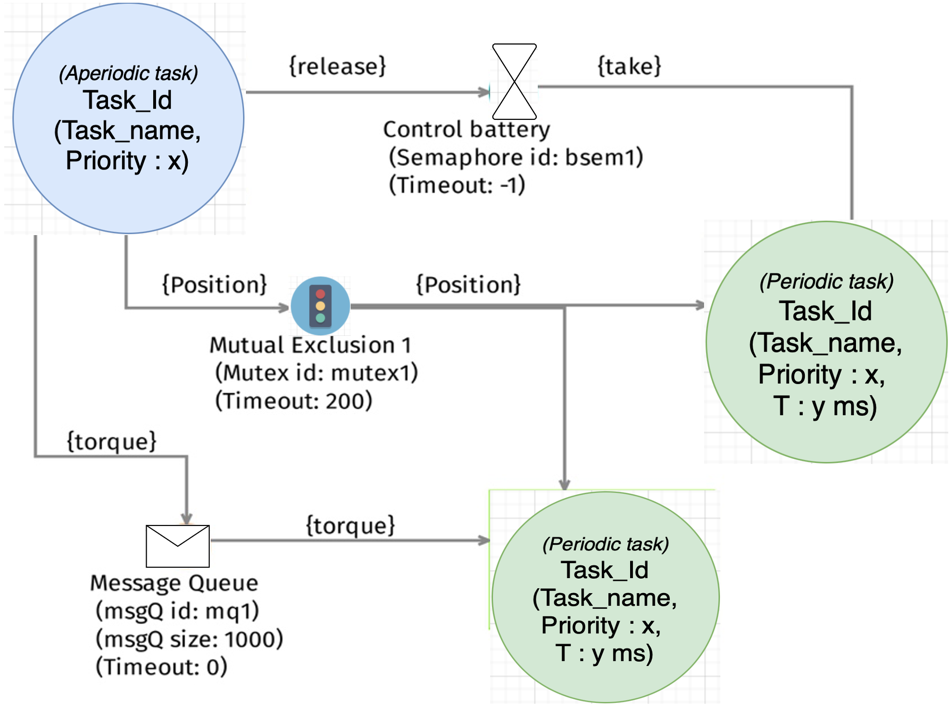
|  |  |  |  |
| --- | --- | --- | --- |
| Tâches à synchroniser | Événement à signaler | Id du sémaphore binaire qui représente l’événement | Justification |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Par exemple, si vous devez synchroniser l’exécution de tâches, vous pouvez utiliser un sémaphore binaire, qui est libéré par l’une des tâches et sur lequel une autre tâche est en attente.

Transformer le diagramme d’architecture logique en un **diagramme d’architecture physique**.

Architecture physique statique :

*Reprendre le diagramme précédent décrivant les constituants du système étudié (les tâches) et complétez-le pour préciser les choix d’implantation physique (dépendant de la plateforme d’exécution) pour supporter les liens d’interface internes et externes (entrées/sorties des tâches) au vu des choix d’implantation opérés pour les moyens de communication et de synchronisation.*

**

# Codage et livraisons incrémentales

Pour le développement et la validation de l’application, vous devez décrire :

* La stratégie de codage, vérification et intégration,
* Le partage du travail dans l’équipe,
* Les livrables.

Décrire la stratégie de codage et intégration :

Stratégie de codage et d’intégration

# Analyse et validation du logiciel livré par rapport aux exigences

Pour chacune des exigences attendues, indiquez si elle a été réalisée. Ajoutez d’éventuelles remarques si le fonctionnement n’est pas exactement celui attendu, ou expliquant les difficultés rencontrées, justifiez le respect des propriétés temporelles.

|  |  |  |
| --- | --- | --- |
| Numéro exigence | Description de l’exigence | État |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Commentaires et conclusion

Dans cette partie indiquer toute remarque relative au déroulement du projet.