Dossier de conception

***Auteurs***

François EVEN

Julien FRANCHINEAU

Version 1.0

Sommaire

[1. Conception générale 3](#_Toc315295298)

[1.1 Architecture candidate 3](#_Toc315295299)

[1.2 Principe de fonctionnement 4](#_Toc315295300)

[2. Conception détaillée 8](#_Toc315295301)

[2.1 Architecture logicielle 8](#_Toc315295302)

[2.2 Communication entre processus 10](#_Toc315295303)

[2.3 Les taches 11](#_Toc315295304)

[2.3.1 Processus simulateur 11](#_Toc315295305)

[2.3.2 Processus des codes à tester 11](#_Toc315295306)

[2.4 Utilisation des ressources externes 12](#_Toc315295307)

[2.4.1 Outil de création d’interface 12](#_Toc315295308)

[2.4.2 Compilation du code à tester 13](#_Toc315295309)

[Table des versions et révisions 14](#_Toc315295310)

**Table des illustrations**

[Figure 1*: Description de l’architecture candidate par un diagramme de collaboration UML* 3](#_Toc315295311)

[Figure 2 : *Diagramme séquence de la visualisation des messages sur l’UART* 4](#_Toc315295312)

[Figure 3 : *Diagramme séquence représentant la visualisation et l’envoi de messages CAN* 5](#_Toc315295313)

[Figure 4 : *Diagramme séquence représentant l’activité de la carte électronique interactive* 6](#_Toc315295314)

[Figure 5 : *Diagramme séquence permettant le contrôle des données de simulation* 6](#_Toc315295315)

[Figure 6 : *Diagramme de séquence représentant le scénario de pilotage de la simulation* 7](#_Toc315295316)

[Figure 7 : *Diagramme des classes détaillées du simulateur EVE* 9](#_Toc315295317)

[Figure 8 : *Tableau récapitulatif des abréviations de zones de l’IHM* 12](#_Toc315295318)

[*Figure 9 : Tableau récapitulatif des abréviations de type des zones de l’IHM* 12](#_Toc315295319)

# Conception générale

## Architecture candidate

Dans cette partie nous présenterons l’architecture candidate qui est composée des différents acteurs présents dans notre système. Nous décrirons les différentes méthodes et les différents attributs de ceux-ci.

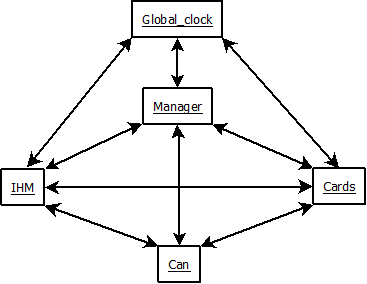


Figure 1*: Description de l’architecture candidate par un diagramme de collaboration UML*

Manager : Il lance les différents composants de notre système de simulation.

IHM : Il permet le contrôle de la simulation par l’utilisateur et permet l’affichage des différentes informations de la simulation.

Can : Il gère la communication entre les cartes par le protocole de communication CAN et permet l’affichage sur l’IHM des différents messages échangés.

Cards : cet élément contient le code à tester des cartes qui dialoguent entre elles par messages CAN. Cet élément dialogue également avec l’IHM afin de permettre à l’utilisateur de contrôler la simulation.

Global\_clock : ce composant a pour rôle de gérer les options temporelles de la simulation et de les fournir à l’ensemble des threads et des processus.

## Principe de fonctionnement

Les diagrammes suivants permettent de visualiser le comportement des différents composants de notre système lors de la réalisation des différents scénarios d’utilisation.

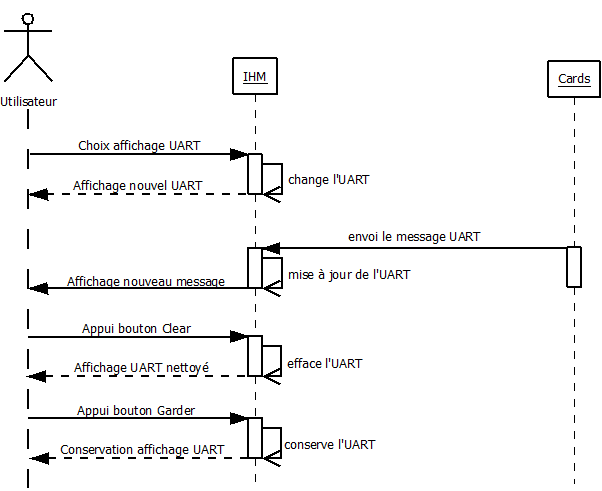


Figure 2 : *Diagramme séquence de la visualisation des messages sur l’UART*

Dans ce diagramme séquence, nous représentons le choix de l’UART par l’utilisateur, l’affichage des messages sortant de l’UART des cartes, le nettoyage de la fenêtre d’affichage de l’UART et également le choix de l’utilisateur de conserver la visualisation de l’UART courante.

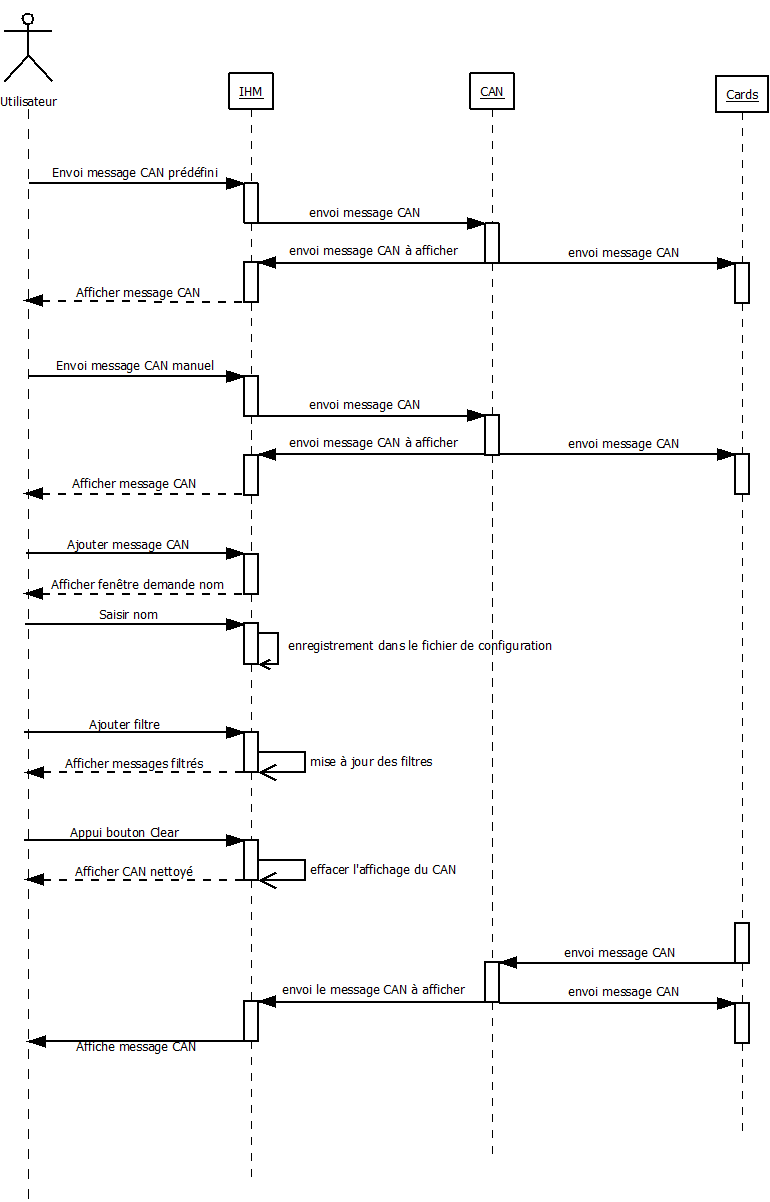


Figure 3 : *Diagramme séquence représentant la visualisation et l’envoi de messages CAN*

Ce diagramme séquence représente l’envoi depuis l’IHM des différents types de message CAN, le filtrage des messages CAN sur l’IHM et la réception des messages provenant du code des cartes à tester.

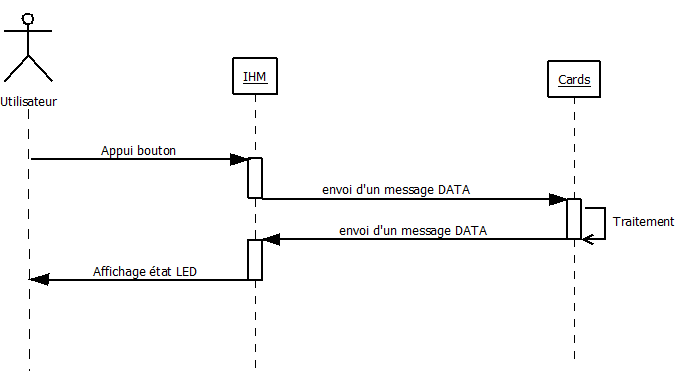


Figure 4 : *Diagramme séquence représentant l’activité de la carte électronique interactive*

Ce scénario prend en compte l’appui des différents boutons de la carte interactive et l’allumage des LEDs présentes sur la carte.

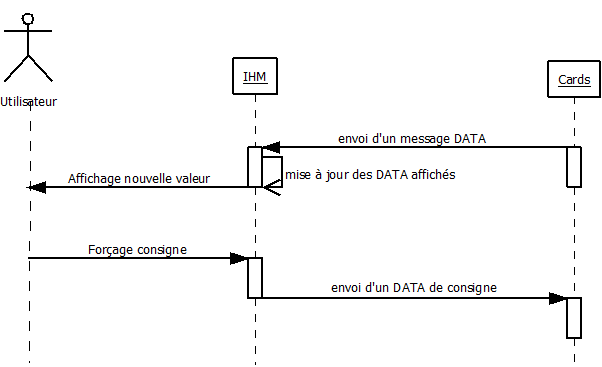


Figure 5 : *Diagramme séquence permettant le contrôle des données de simulation*

Ce diagramme représente le contrôle des données de simulation provenant des cartes ainsi que les valeurs de consigne que souhaite imposer l’utilisateur.

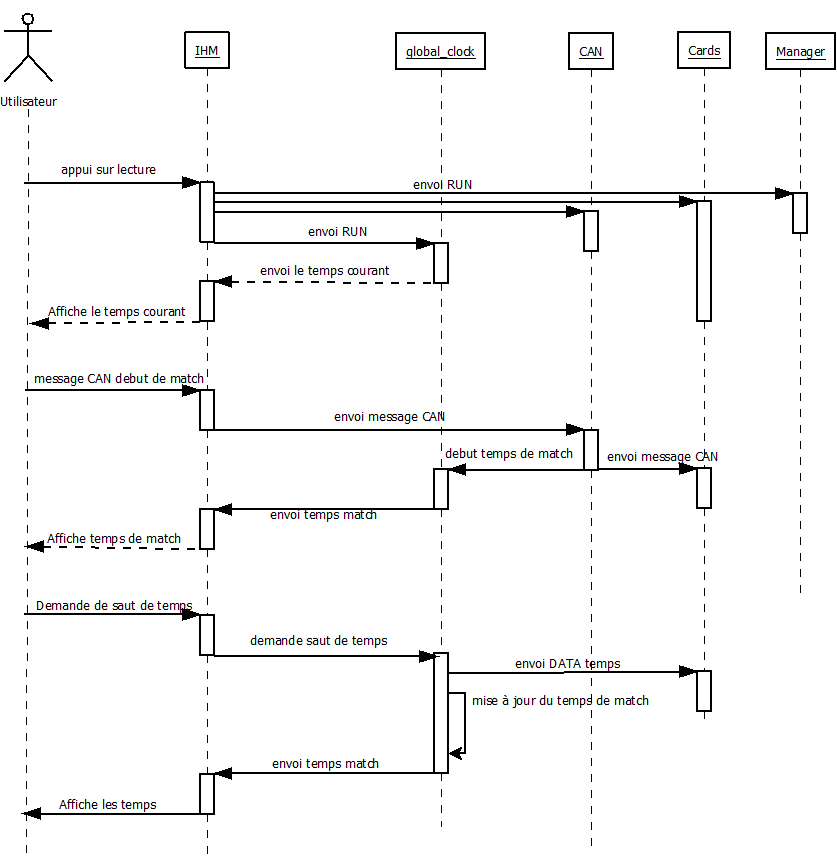


Figure 6 : *Diagramme de séquence représentant le scénario de pilotage de la simulation*

Ce diagramme représente le pilotage temporel de la simulation ainsi que l’affichage des différents temps de simulation.

# Conception détaillée

## Architecture logicielle

La réalisation de ce logiciel se fera en langage C sous le système d’exploitation Ubuntu. Il est donc nécessaire d’ajouter certaines classes de communication pour permettre aux différents processus de communiquer entre eux.

Les classes présentées dans la conception générale vont s’exécuter sur plusieurs processus différents. Un pour le simulateur et l’IHM et des processus séparés pour chaque code à tester. Il est donc nécessaire de rajouter des classes de communication permettant le dialogue entre plusieurs processus.

L’IHM étant très lourde à gérer, et comportant des contraintes dues à l’utilisation de GTK, deux threads seront créés pour la faire fonctionner. IHM\_display s’occupera d’afficher l’interface et de récupérer les demandes de l’utilisateur, lorsqu’il appuiera sur un bouton par exemple, et les transmettra aux threads ou processus correspondants. D’un autre côté, IHM\_receiver s’occupera de gérer les informations qui arrivent du reste du simulateur et des cartes pour mettre à jour l’IHM.

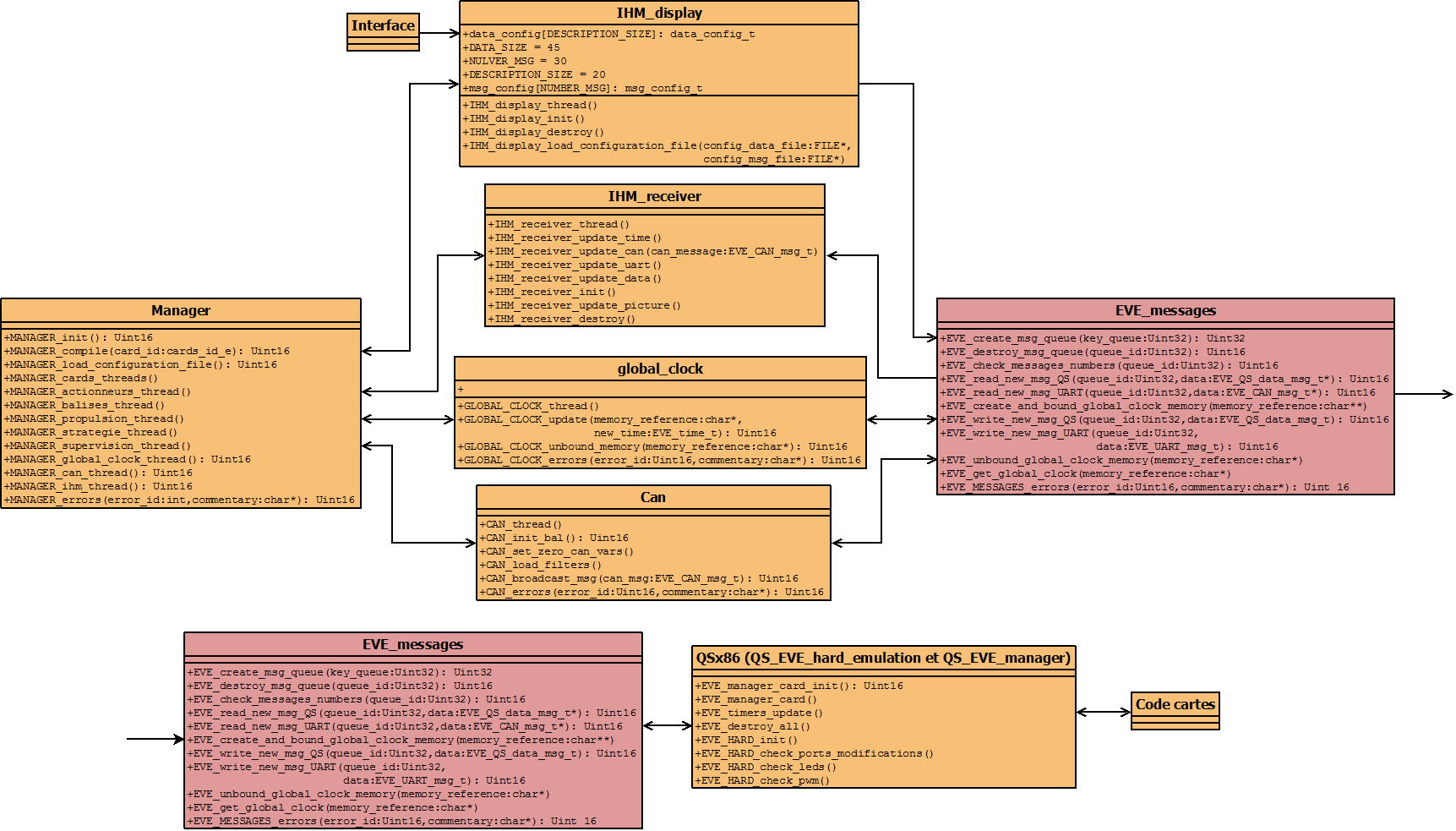


Figure  : *Diagramme des classes détaillées du simulateur EVE*

Les classes en rouge ci-dessus représentent les classes de communication entre processus. Ces classes s’échangent des messages CAN ainsi que des données de simulation.

## Communication entre processus

La communication entre processus se fait par boites aux lettres. Cette communication se fera grâce à 5 boites aux lettres par carte :

* Envoi de messages CAN du simulateur sur la carte
* Envoi des messages CAN de la carte au simulateur
* Envoi des données QS du simulateur à la carte
* Envoi des données QS de la carte au simulateur
* Envoi de printf par UART de la carte au simulateur

|  |  |  |  |
| --- | --- | --- | --- |
| Emetteur | Destinataire | Type de message | ID |
| Actionneurs | CAN | CAN | 2126 |
| Actionneurs | IHM | QS\_DATA | 2128 |
| Actionneurs | IHM | UART | 2129 |
| Balises | CAN | CAN | 2226 |
| Balises | IHM | QS\_DATA | 2228 |
| Balises | IHM | UART | 2229 |
| Propulsion | CAN | CAN | 2326 |
| Propulsion | IHM | QS\_DATA | 2328 |
| Propulsion | IHM | UART | 2329 |
| Stratégie | CAN | CAN | 2426 |
| Stratégie | IHM | QS\_DATA | 2428 |
| Stratégie | IHM | UART | 2429 |
| Supervision | CAN | CAN | 2526 |
| Supervision | IHM | QS\_DATA | 2528 |
| Supervision | IHM | UART | 2529 |
| CAN | Actionneurs | CAN | 2621 |
| CAN | Balises | CAN | 2622 |
| CAN | Propulsion | CAN | 2623 |
| CAN | Stratégie | CAN | 2624 |
| CAN | Supervision | CAN | 2625 |
| IHM | Actionneurs | QS\_DATA | 2821 |
| IHM | Balises | QS\_DATA | 2822 |
| IHM | Propulsion | QS\_DATA | 2823 |
| IHM | Stratégie | QS\_DATA | 2824 |
| IHM | Supervision | QS\_DATA | 2825 |

## Les taches

### Processus simulateur

Le processus principal du simulateur fonctionne en utilisant plusieurs taches.

Une tache est réservée à l’exécution de l’IHM. Une tache gère la gestion des messages CAN. Une tache est dédiée à la gestion de l’horloge générale. Le manager sera la tache de fond qui lancera et contrôlera le reste des taches. Les processus des cartes sont chacun lancés par un thread spécifique qui restera bloqué tant que la carte est active.

### Processus des codes à tester

Les processus qui utilisent le code à tester fonctionnent également en multithread, une tache permet le fonctionnement normal du code source à tester. Des fonctions de communication permettront le dialogue entre processus via les boites aux lettres.

## Utilisation des ressources externes

### Outil de création d’interface

L’ensemble de l’interface utilisateur a été développé sous Glade. Ce logiciel de création d’interface sous licence GNU GPL utilise les bibliothèques graphiques GTK+. Ce logiciel permet de manipuler les objets de GTK et de fournir un code en langage C définissant ces objets.

Cet outil permet la génération du fichier « IHM\_interface.c ». Le fichier « callbacks.c » s’occupe de gérer les signaux qui permettent de réaliser la communication entre les objets GTK et le reste du simulateur

L’ensemble des éléments, composants ou objets, de l’interface du simulateur respecte la convention de nommage suivante :

nomDeLaZone\_type\_descriptif

Exemple : can\_button\_clear

|  |  |
| --- | --- |
| Abréviation du nom de la zone | Description |
| can | Can |
| cei | Carte électronique interactive |
| data | Données de simulation |
| os | Options de simulation |
| uart | Uart |
| live | Affichage dynamique |

Figure 8 : *Tableau récapitulatif des abréviations de zones de l’IHM*

|  |  |
| --- | --- |
| Type des éléments | Description |
| button | Bouton |
| cb | Combo Box |
| chk | Check Button |
| zt | Zone de texte |
| st | Saisi de texte |
| ong | Onglet |

*Figure 9 : Tableau récapitulatif des abréviations de type des zones de l’IHM*

### Compilation du code à tester

Les codes à tester doivent se situer dans les dossiers SVN dédiés, il est nécessaire d’inclure la nouvelle QSx86 à la place de la QS initiale afin de réaliser la compilation.

Lors de l’exécution de EVE, le manager va compiler les codes grâce au makefile du projet qui contient des lignes de compilation spécifiques pour compiler chaque code sous Linux. Par la suite, il s’occupera de les exécuter en démarrant des threads qui chacun lancerons un processus pour une carte.

# Table des versions et révisions

|  |  |  |  |
| --- | --- | --- | --- |
| Version/Révision | Date | Modifications | Auteurs |
| 0.1 | 15/11/2011 | Création du document | François EVEN |
| 0.2 | 17/11/2011 | Ajout diagramme pour la conception générale | François EVEN |
| 0.3 | 22/11/2011 | Ajout des diagrammes de séquence et conception détaillée | François EVEN |
| 0.4 | 23/11/2011 | Modification de la conception détaillée | François EVEN |
| 0.5 | 24/01/2012 | Compléments et modifications après le développement | François EVEN  Julien FRANCHINEAU |
| 1.0 | 25/01/2012 | Mise à jour pour la release | Julien FRANCHINEAU |
|  |  |  |  |