Dossier de spécification

***Auteurs***

François EVEN

Julien FRANCHINEAU

Version 1.2

Sommaire

[Avant-propos 4](#_Toc315285302)

[1. Grandes fonctionnalités 5](#_Toc315285303)

[1.1 L’expression du besoin 5](#_Toc315285304)

[1.2 Frontières du système 5](#_Toc315285305)

[2. Fonctionnalités du système 8](#_Toc315285306)

[2.1 Fonctionnalités générales 8](#_Toc315285307)

[2.2 Cas d’utilisations 8](#_Toc315285308)

[2.2.1 Tester les codes 9](#_Toc315285309)

[2.2.2 Visualiser/Envoyer des messages CAN 9](#_Toc315285310)

[2.2.3 Visualiser les messages sur l’UART 10](#_Toc315285311)

[2.2.4 Contrôler les données de simulation 11](#_Toc315285312)

[2.2.5 Utiliser la carte électronique interactive 12](#_Toc315285313)

[2.2.6 Piloter la simulation 12](#_Toc315285314)

[2.2.7 Visualiser le déplacement du robot 13](#_Toc315285315)

[2.2.8 Démarrer la simulation 13](#_Toc315285316)

[3. IHM 15](#_Toc315285317)

[3.1 Terminal CAN 15](#_Toc315285318)

[3.2 Options du CAN 16](#_Toc315285319)

[3.3 Terminal UART 18](#_Toc315285320)

[3.4 Carte électronique interactive 19](#_Toc315285321)

[3.5 Données de simulation matérielle 20](#_Toc315285322)

[3.6 Options de simulation 21](#_Toc315285323)

[3.7 Affichage dynamique 21](#_Toc315285324)

[3.8 Terminal utilisateur 22](#_Toc315285325)

[3. Livrables 23](#_Toc315285326)

[3.1 Code source et exécutable 23](#_Toc315285327)

[3.2 Dossier de spécification 23](#_Toc315285328)

[3.3 Dossier de conception 23](#_Toc315285329)

[3.4 Documentation 23](#_Toc315285330)

[Lexique 24](#_Toc315285331)

[Table des versions et révisions 25](#_Toc315285332)

**Table des illustrations**

[Figure 1*: Représentation des différentes actions réalisées par le système* 5](#_Toc315285287)

[Figure 2 : *Diagramme de l’architecture matérielle utilisant le procédé* 6](#_Toc315285288)

[Figure 3 : *Diagramme de l’architecture matérielle s’affranchissant du procédé* 6](#_Toc315285289)

[Figure 4 : *Diagramme des fonctionnalités principales du système* 8](#_Toc315285290)

[Figure 5 : *Interface complète du logiciel* 15](#_Toc315285291)

[Figure 6 : *Interface de l’affichage des messages CAN* 16](#_Toc315285292)

[Figure 7 : *Interface d’envoi des messages CAN prédéfinis* 17](#_Toc315285293)

[Figure 8 : *Interface d’envoi manuel des messages CAN* 17](#_Toc315285294)

[Figure 9 : *Interface des options de filtrage des messages CAN* 18](#_Toc315285295)

[Figure 10 : *Interface des terminaux UART* 19](#_Toc315285296)

[Figure 11 : *Interface de la carte électronique interactive* 20](#_Toc315285297)

[Figure 12 : *Interface de données de simulation matérielle* 20](#_Toc315285298)

[Figure 13 : *Interface des options de simulation* 21](#_Toc315285299)

[Figure 14 : *Affichage dynamique du robot* 22](#_Toc315285300)

[Figure 15 : *Interface du terminal utilisateur* 22](#_Toc315285301)

# Avant-propos

Dans ce document, nous allons répondre précisément aux demandes réalisées par notre client sur le produit livrable. Le Club Robotique de l’ESEO sera notre client final, et représenté ici par Monsieur Samuel Poiraud.

Pour ce faire, notre équipe, composée de François Even et Julien Franchineau, a spécifié dans ce dossier les fonctionnalités qu’elle réalisera au cours de ce projet. Elle a précisé également, dans ce document, les éléments qu’elle ne sera pas en mesure de réaliser.

Nous détaillerons donc le contexte de fonctionnement de cette application, les fonctions qu’elle implémentera. Par ailleurs, nous fournirons une ébauche de l’IHM afin que le client puisse avoir une idée du produit final.

Ce document s’adresse à l’équipe de développement ainsi qu’à notre client qui pourra valider ces spécifications.

Ce document est structuré autour de plusieurs parties. La première traite des fonctions qui seront réalisées et du matériel qui est proposé. Les deux autres parties développeront les aspects du système et l’IHM de l’application.

Attention, toutes les images présentées dans ce document sont non contractuelles.

# Grandes fonctionnalités

## L’expression du besoin

Les membres du club robotique de l’ESEO disposent d’un temps limité pour concevoir et créer le robot qui se présentera à la Coupe de France de Robotique. Le club améliore et fiabilise toujours plus les phases de développement, mais actuellement, les tests nécessitent l’utilisation importante du robot.

Le développement d’un logiciel permettant de tester le code éviterait de trop utiliser le robot, et surtout de valider les différentes parties du programme développées tout au long de l’année sans avoir à attendre l’arrivée des pièces du robot.

## Frontières du système

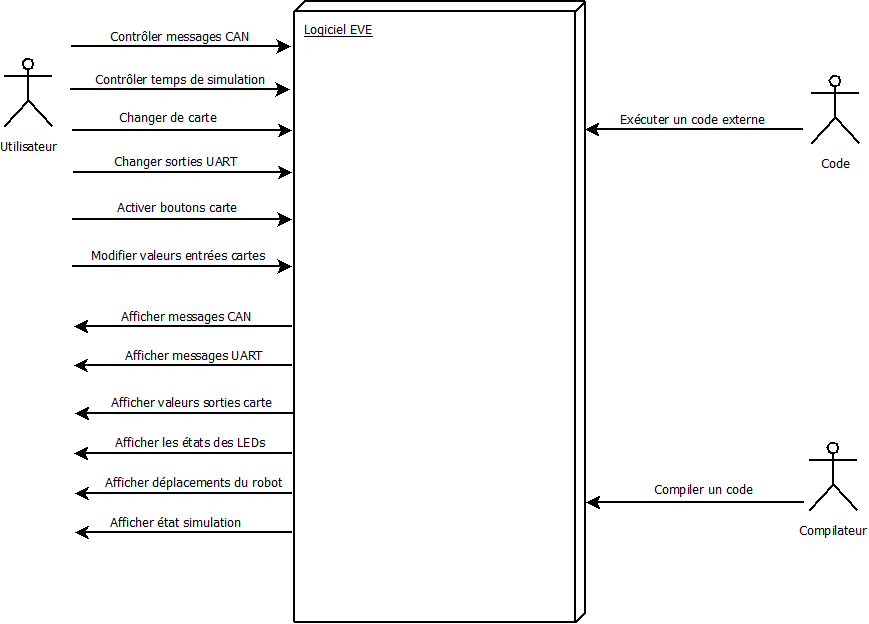


Figure 1*: Représentation des différentes actions réalisées par le système*

Le système fournit permettra de tester le bon fonctionnement des codes développés pour le robot. L’objectif est de fournir un logiciel permettant de s’affranchir du procédé (le robot physique) qui n’arrive que tard dans l’année pour pouvoir commencer à tester rapidement les codes et supprimer un maximum d’erreurs.

Le logiciel EVE qui sera développé utilisera les codes de chaque carte du robot (qui devront être déposés dans un dossier spécifique du logiciel) et les compilera avec un compilateur extérieur tel que MinGW pour ensuite exécuter les codes directement sur l’ordinateur tout en offrant une interface graphique permettant une interaction facilité avec la simulation.

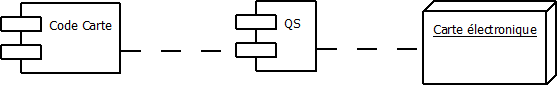


Figure 2 : *Diagramme de l’architecture matérielle utilisant le procédé*

Le diagramme ci-dessus décrit l’environnement matériel réel d’exécution des codes que nous souhaitons tester. L’objectif est de se passer du procédé, et donc ici de la carte électronique reliée au reste du robot. Le « code carte » à tester ne pouvant changer, nous devons développer une nouvelle QS permettant de s’affranchir du matériel électronique embarqué sur le robot.

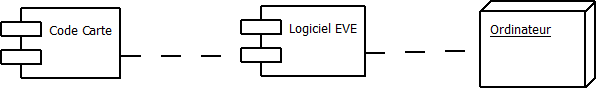


Figure 3 : *Diagramme de l’architecture matérielle s’affranchissant du procédé*

Le diagramme ci-dessus représente l’environnement matériel dans lequel nous allons développer notre système. Le « code carte » à tester reste inchangé, et la plateforme d’exécution devient un ordinateur, utilisable partout pour tout développeur du code. Le logiciel doit donc s’adapter à un maximum de formats d’ordinateurs.

Les logiciels :

Le **Code Carte** représente les cinq codes du robot (balises, actionneurs, propulsion, supervision, stratégie) qui sont les codes à tester. Ils sont indépendants de la plateforme d’exécution.

La **QS** représente le package qualité software. Il s’agit d’une couche de drivers permettant d’exécuter le **Code Carte** sur la **Carte électronique**. C’est la partie bas niveau du logiciel qui fait le lien entre le software et le hardware.

Le **Logiciel EVE** est notre logiciel présent sur l’**Ordinateur**.

Les matériels :

La **Carte électronique** est la carte embarquée sur le robot qui contient le programme spécifique pour une tâche donnée (stratégie, supervision…). Elle utilise un dsPIC30F6010A pour exécuter le code.

L’**Ordinateur** est fourni par l’utilisateur de l’application. Il héberge le **Logiciel EVE** et fonctionne sous GNU/Linux (Ubuntu).

# Fonctionnalités du système

## Fonctionnalités générales

Le système va récupérer un code en développement, le compiler et l’exécuter. L’utilisateur pour alors, grâce à l’interface offerte par le système, interagir avec le code, et analyser son exécution. De nombreux affichages des sorties et des messages des cartes permettront d’observer le bon déroulement du programme en cours de test. L’utilisateur pour agir sur certaines valeurs d’entrées du programme en cours de test pour analyser ses réactions, simuler les appuis sur les boutons de la carte, et contrôler temporellement la simulation. Bien entendu, il pourra aussi afficher les messages qu’il souhaitera grâce à un système de filtres prévus.

## Cas d’utilisations

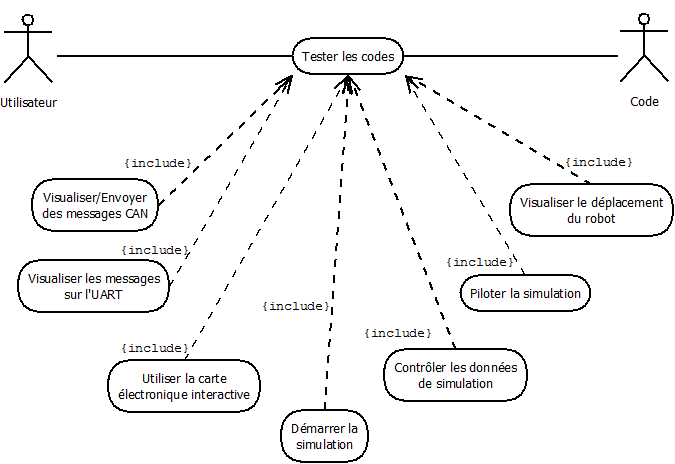


Figure 4 : *Diagramme des fonctionnalités principales du système*

### Tester les codes

Résumé : L’utilisateur visualise le fonctionnement du code afin de déterminer les erreurs de codage.

Portée : Logicielle

Niveau : Stratégique

Acteur principal : Utilisateur

Acteur secondaire : Robot

Acteur tertiaire : l’ESEO

Pré-conditions : Codes compilable

Garantie en cas de succès : Détermination du comportement total du code testé

Garantie minimale : Message d’erreur

Scénario nominal :

1. L’utilisateur charge les codes à tester
2. L’utilisateur compile les codes à tester
3. L’utilisateur lance la simulation
4. L’utilisateur visualise/envoi des messages CAN
5. L’utilisateur visualise les messages sur l’UART
6. L’utilisateur utilise la carte électronique interactive
7. L’utilisateur visualise le déplacement du robot sur l’affichage dynamique

### Visualiser/Envoyer des messages CAN

Résumé : L’utilisateur observe et contrôle les différents messages échangés entre les cartes. Il filtre et envoi des messages en fonction de ses besoins.

Portée : Logicielle

Niveau : Utilisateur

Acteur principal : Utilisateur

Acteur secondaire : Robot

Acteur tertiaire : l’ESEO

Pré-conditions : Code(s) lancé(s)

Garantie en cas de succès : Messages correctement affichés, envoyés et filtrés.

Garantie minimale : Message d’erreur

Scénario nominal :

1. L’utilisateur visualise et contrôle les messages échangés
2. L’utilisateur active un ou plusieurs filtres
3. L’utilisateur visualise les messages filtrés
4. L’utilisateur envoie un message CAN prédéfini.
5. L’utilisateur contrôle les réactions relatives au message précédemment envoyé

Alternative :

A1. Le CU commence au point 3

1. L’utilisateur choisit de rentrer manuellement un message
2. L’utilisateur envoie un message

Le CU reprend au point 5

Extension :

E1. Le CU commence au point 5

1. L’utilisateur nettoie l’affichage du message
2. L’utilisateur visualise les nouveaux messages échangés

### Visualiser les messages sur l’UART

Résumé : L’utilisateur observe les messages des sorties UART

Portée : Logicielle

Niveau : Utilisateur

Acteur principal : Utilisateur

Acteur secondaire : Robot

Acteur tertiaire : l’ESEO

Pré-conditions : Code(s) lancé(s)

Garantie en cas de succès : Messages correctement affichés

Garantie minimale : Message d’erreur

Scénario nominal :

1. L’utilisateur visualise les messages des UART de la carte en cours d’affichage
2. L’utilisateur modifie la sortie de l’UART affiché sur un des terminaux UART
3. L’utilisateur observe les messages de la nouvelle sortie UART
4. L’utilisateur nettoie l’affichage
5. L’utilisateur modifie la carte courante
6. L’utilisateur visualise les UART de la nouvelle carte courante

Alternative :

E1. Le CU commence au point 5

1. L’utilisateur verrouille la fenêtre d’UART avec le bouton « garder »
2. L’utilisateur modifie la carte courante
3. L’utilisateur visualise un UART de la nouvelle carte courante et l’UART verrouillé

### Contrôler les données de simulation

Résumé : L’utilisateur impose des valeurs sur les entrées et contrôle les valeurs des sorties de la carte simulé

Portée : Logicielle

Niveau : Utilisateur

Acteur principal : Utilisateur

Acteur secondaire : Robot

Acteur tertiaire : l’ESEO

Pré-conditions : Simulation lancée

Garantie en cas de succès : Messages correctement affichés

Garantie minimale : Message d’erreur

Scénario nominal :

1. L’utilisateur observe les données de simulation matérielle
2. L’utilisateur modifie la valeur d’une ADC sur l’écran des données de simulation

### Utiliser la carte électronique interactive

Résumé : L’utilisateur appuie sur les boutons et visualise les LED de la carte électronique interactive

Portée : Logicielle

Niveau : Utilisateur

Acteur principal : Utilisateur

Acteur secondaire : Robot

Acteur tertiaire : l’ESEO

Pré-conditions : Simulation lancée

Garantie en cas de succès : Messages correctement affichés

Garantie minimale : Message d’erreur

Scénario nominal :

1. L’utilisateur appuie sur un bouton de la carte électronique
2. L’utilisateur observe les différentes LED présente sur la carte simulé

### Piloter la simulation

Résumé : L’utilisateur contrôle le déroulement temporel de la simulation

Portée : Logicielle

Niveau : Utilisateur

Acteur principal : Utilisateur

Acteur secondaire : Robot

Acteur tertiaire : l’ESEO

Pré-conditions : Code(s) chargé(s)

Garantie en cas de succès : Messages correctement affichés

Garantie minimale : Message d’erreur

Scénario nominal :

1. L’utilisateur lance la simulation en appuyant sur le bouton  « lecture » des options de simulation
2. L’utilisateur visualise la progression du temps de simulation dans la zone « temps actuel »
3. L’utilisateur met en pause la simulation en activant le bouton  « pause »
4. L’utilisateur choisit le temps à visualiser dans le « saut de temps »
5. L’utilisateur visualise les valeurs des données correspondant au temps choisi
6. L’utilisateur arrête la simulation en utilisant le bouton « arrêt »

### Visualiser le déplacement du robot

Résumé : L’utilisateur observe les déplacements du robot simulé

Portée : Logicielle

Niveau : Utilisateur

Acteur principal : Utilisateur

Acteur secondaire : Robot

Acteur tertiaire : l’ESEO

Pré-conditions : Simulation lancée

Garantie en cas de succès : Messages correctement affichés

Garantie minimale : Message d’erreur

Scénario nominal :

1. L’utilisateur observe le robot simulé se déplacer sur l’affichage dynamique
2. L’utilisateur visualise les coordonnées x, y et téta
3. L’utilisateur fait apparaitre l’affichage dynamique dans une nouvelle fenêtre en appuyant sur le bouton d’agrandissement
4. L’utilisateur visualise les déplacements du robot simulé sur la nouvelle fenêtre
5. L’utilisateur ferme la fenêtre dédié à l’affichage dynamique

### Démarrer la simulation

Résumé : L’utilisateur choisit quels codes il va utiliser et démarre la simulation

Portée : Logicielle

Niveau : Utilisateur

Acteur principal : Utilisateur

Acteur secondaire : Robot

Acteur tertiaire : l’ESEO

Pré-conditions : Aucune

Garantie en cas de succès : Le système est démarré avec les codes en cours d’exécution

Garantie minimale : Message d’erreur

Scénario nominal :

1. L’utilisateur choisit quels codes il souhaite exécuter
2. L’utilisateur lance la compilation des codes choisis et démarre la simulation
3. L’utilisateur contrôle que les codes souhaités sont correctement en fonctionnement

# IHM

L’interface se décompose en plusieurs parties répondant à une fonction spécifique pour permettre de tester les codes. Elle se présente sous la forme d’une seule fenêtre regroupant toutes les options nécessaire au test.

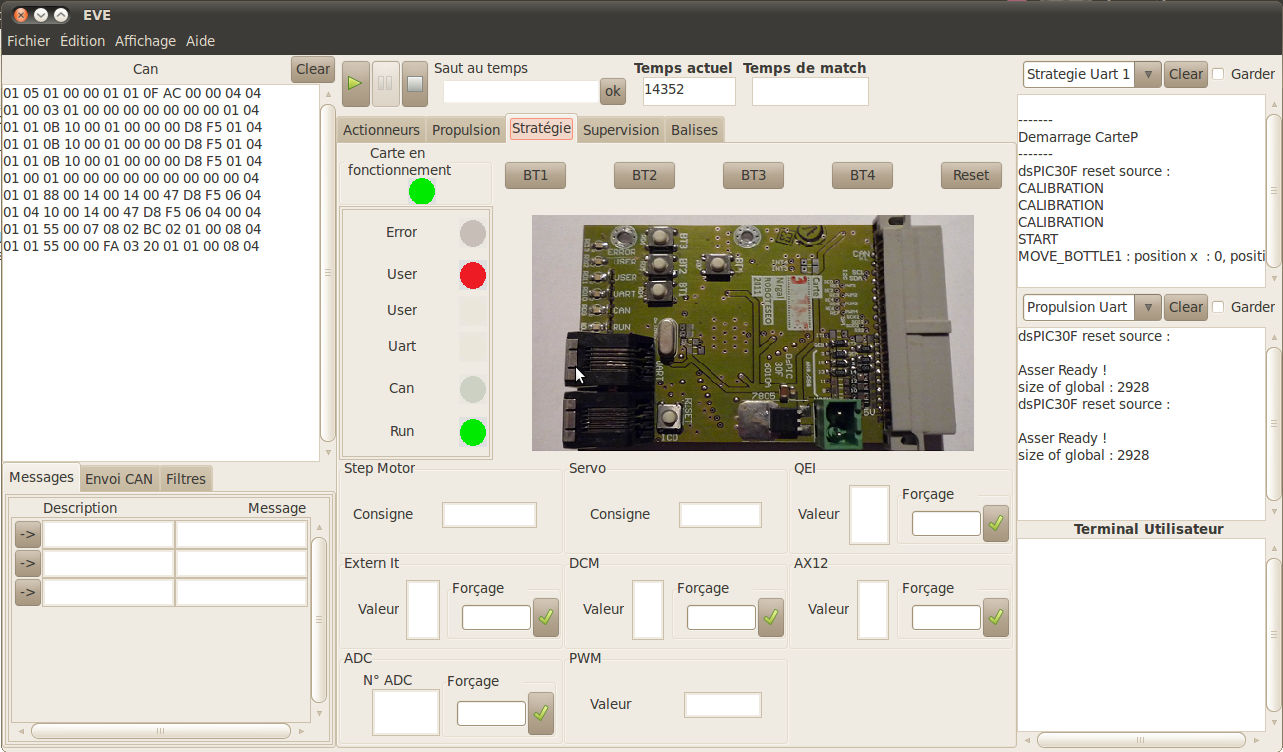


Figure 5 : *Interface complète du logiciel*

## Terminal CAN

Cette zone de l’interface est dédiée aux messages CAN qui transitent entre les différentes cartes du robot. Elle est composée de deux parties principales :

* Affichage des messages

Les messages seront affichés ici, sous format hexadécimal, et autant que possible de manière lisible avec un retour à la ligne à chaque nouveau message. L’affichage dispose d’une barre de scroll permettant de naviguer facilement entre les messages et ainsi d’afficher les messages précédemment reçus. Le bouton « Clear » permettra d’effacer l’ensemble des messages affichés pour redémarrer une session à partir de zéro.

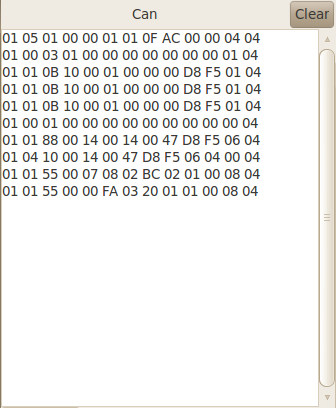


Figure 6 : *Interface de l’affichage des messages CAN*

## Options du CAN

Cette zone comporte 3 onglets qui permettent de contrôler les options du terminal CAN. Le premier onglet « Messages » contient plusieurs messages CAN prédéfinis qui peuvent être facilement envoyés sur le CAN via le bouton « Envoyer » prévu à cet effet. La liste comporte le nom du message (qui permet de plus facilement identifier un message) et les données du message en hexadécimal. Les messages sont stockés dans un fichier à part facilement modifiable avec un éditeur de texte permettant à l’utilisateur de choisir ses propres messages.

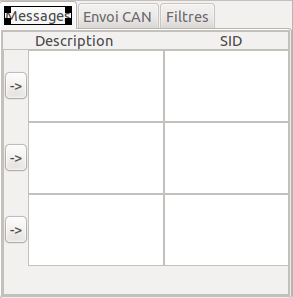


Figure 7 : *Interface d’envoi des messages CAN prédéfinis*

Le deuxième onglet « Envoi CAN » offre une interface de saisie de message CAN à l’utilisateur. Il peut ainsi écrire manuellement son message sous format hexadécimal et l’envoyer sur le bus CAN via le bouton « Envoyer ». Le bouton « Ajouter » permet de sauvegarder le message et de le stocker avec les messages CAN prédéfinis. Une fenêtre s’ouvrira pour que l’on puisse ajouter le nom du message qui sera affiché dans la liste des messages prédéfinis.

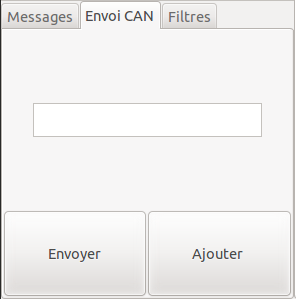


Figure 8 : *Interface d’envoi manuel des messages CAN*

Le troisième onglet « Filtres » permet à l’utilisateur de filtrer les messages qui sont affichés à l’écran. Différentes options sont présentes. Il est possible de filtrer les messages selon la carte de destination (actionneurs, stratégie, etc.) ou par des valeurs hexadécimales spécifiques que l’on pourra indiquer dans les zones de saisies appropriées.

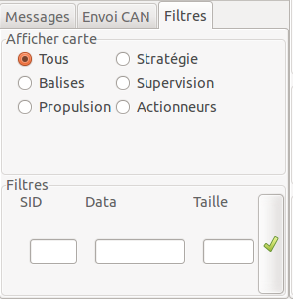


Figure 9 : *Interface des options de filtrage des messages CAN*

## Terminal UART

Cette zone de l’interface est dédiée à l’affichage des sorties UART des cartes. Ainsi, l’utilisateur pour facilement afficher les différents « printf » exécutés dans les codes de test. Le logiciel propose deux terminaux d’affichages identiques pour donner un maximum d’informations. Les terminaux proposent une option « Clear » permettant de vider l’affichage des messages pour démarrer une nouvelle session à partir de zéro. Un menu déroulant permet de choisir parmi une liste prédéfinie quel UART l’utilisateur souhaite afficher sur le terminal.

Les sorties UART correspondent aux sorties de la carte courante en cours d’utilisation. Lorsque la carte courante est changée par l’utilisateur, les sorties UART changent automatiquement pour se régler sur la nouvelle carte. Il est possible de conserver un affichage d’UART même en changeant de carte grâce à l’option « Garder » et ainsi ne pas à devoir modifier à chaque changement de carte la sortie UART que l’on souhaite afficher.

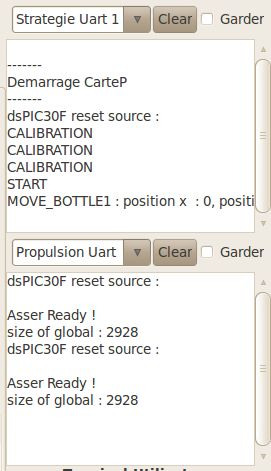


Figure 10 : *Interface des terminaux UART*

## Carte électronique interactive

Cette zone est consacrée a offrir une simulation interactive de la carte électronique. Elle comporte une photographie de la carte sur laquelle le code simulé devra s’exécuter, ainsi qu’un affichage dynamique des LED qui simuleront les véritables LED de la carte. L’utilisateur peut appuyer sur les boutons de la carte pour simuler au mieux la véritable carte.

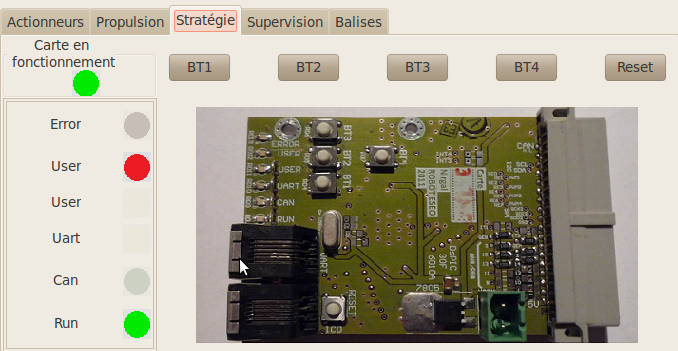


Figure 11 : *Interface de la carte électronique interactive*

## Données de simulation matérielle

Cet environnement est dédié aux entrées/sorties du microcontrôleur telles que les PWN ou DCM. Différentes options sont disponibles pour chaque sortie, comme la possibilité de changer de port et ainsi de visualiser plusieurs valeurs sur le même affichage. Il est possible d’imposer des valeurs sur les entrées pour, par exemple, simuler un capteur.

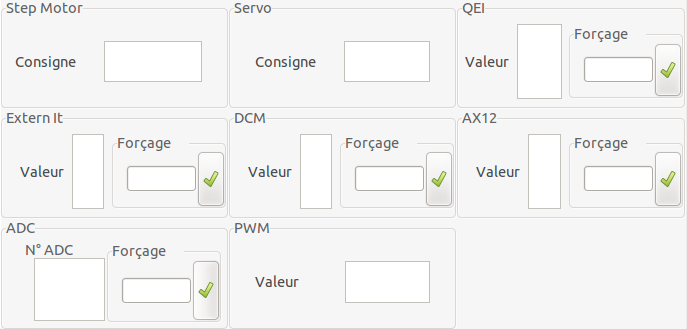


Figure 12 : *Interface de données de simulation matérielle*

## Options de simulation

Cette zone est permet de modifier la carte courante en choisissant la carte voulue parmi les onglets. De plus, l’utilisateur dispose ici des options permettant de contrôler temporellement la simulation. Il peut ainsi lancer, stopper temporairement et arrêter la simulation. Il peut aussi choisir un temps spécifique à partir duquel la simulation va être lancée. Cela lui permettra de faire des tests sur certaines parties du match comme il le souhaite. Un afficheur permet de connaître en permanence le temps courant de la simulation.

optionsDeSimulation.png

Figure 13 : *Interface des options de simulation*

## Affichage dynamique

Révision : dans une première version, l’espace qui était prévu pour cet affichage sera consacré au terminal utilisateur (paragraphe 3.8). L’affichage dynamique sera proposé sous la forme d’une fenêtre popup plus grande qui pourra s’ouvrir selon le choix de l’utilisateur qui pourra disposer d’un meilleur affichage.

Cet espace proposera par la suite un affichage dynamique offrant un rendu 2D temps réel des déplacements du robot pendant la simulation. L’objectif est ici de permettre à l’utilisateur de visualiser au mieux ce que le robot réalise comme action pendant la simulation. Un plan du terrain en image de fond offre une vue facile à utiliser pour suivre correctement le déroulement des déplacements. Les coordonnées du robot ainsi que son angle sur le terrain pourront être affichés pour une meilleure lisibilité des déplacements. Cet affichage sera amené à évoluer dans l’avenir pour proposer une interface plus grande pour une meilleure visualisation.

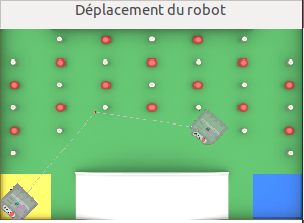


Figure 14 : *Affichage dynamique du robot*

## Terminal utilisateur

Cette zone d’affichage a pour objectif de permettre à EVE de donner des informations sur son état à l’utilisateur. Ainsi, il pourra être prévenu d’une erreur quelconque survenue pendant l’utilisation du logiciel. Cette zone sert uniquement d’affichage.

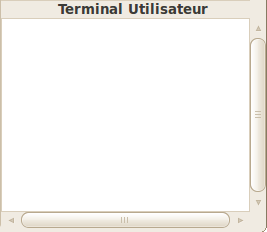


Figure 15 : *Interface du terminal utilisateur*

# Livrables

Ce paragraphe présente les différents documents et fichiers qui seront fournis à la fin du projet.

## Code source et exécutable

L’intégralité du code source sera remise à la fin du projet. Il permettra aux futurs développeurs de pouvoir faire évoluer le logiciel pour répondre encore plus aux attentes des testeurs. Dans la mesure du possible, le code sera commenté pour permettre une prise en main facilité. Un exécutable du programme, fonctionnel dans la mesure du possible, sera également fournis pour pouvoir utiliser immédiatement le logiciel.

## Dossier de spécification

Ce document sera également remis à la fin du projet. Il permettra de pouvoir facilement reprendre le projet en détaillant les besoins du client.

## Dossier de conception

Le dossier de conception sera aussi livré pour que les futurs développeurs puissent rapidement appréhender l’ensemble du code, son fonctionnement et son organisation afin de pouvoir, par la suite, le faire évoluer pour améliorer le logiciel.

## Documentation

Une documentation d’utilisation complète sera aussi proposée avec le reste des documents. Elle contiendra un manuel d’installation du logiciel pour les testeurs ainsi qu’un manuel d’utilisation. De plus, un manuel d’installation des différents logiciels pour le développement du code sera aussi proposé afin de fournir un environnement de qualité rapidement mis en place pour pouvoir continuer à développer le logiciel. Le code sera également commenté et documenté de manière à faciliter sa prise en main.

# Lexique

- Affichage dynamique : affichage d’un rendu 2D temps réel des déplacements du robot durant la simulation. Cet affichage pourra être amené à évoluer en fonction des possibilités du projet.

- Carte courante : carte actuellement à l’écran dont l’interface affiche les différentes sorties (PWN, DCM, etc.), et l’état des LED.

- Carte électronique interactive : affichage interactif de la carte avec l’état des LED de la carte courante et la possibilité d’utiliser les boutons via un clic pour interagir avec la carte courante.

- Données de simulation matérielle : affichage des différentes valeurs de sorties intéressantes de la carte (PWN, DCM, etc.). Il est aussi possible d’imposer des valeurs sur les entrées de la carte via l’interface.

- IHM : Interface Homme-Machine, interface graphique permettant une interaction facilitée de l’utilisateur avec l’application.

- Message CAN prédéfini : message CAN enregistré et prêt à être envoyé par l’utilisateur. Ces messages sont stockés dans un fichier spécifique et la liste pourra évoluer en fonction des besoins des utilisateurs.

- PWM : modulation de largeur d’impulsions (*Pulse Width Modulation*), technique permettant de synthétiser des signaux continus. En appliquant une succession d’états discrets pendant des durées bien choisies, on peut obtenir en moyenne sur une certaine durée n’importe quelle valeur intermédiaire.

- Robot/Robot simulé : représente le robot physique sur lequel doit s’exécuter le code en cours de test.

- Terminal CAN : ensemble de l’interface correspondant au fonctionnement du CAN. Il comprend la zone d’affichage des messages qui transitent sur le bus CAN, ainsi que les onglets d’options comprenant les filtres, les envois de messages prédéfinis et les envois de messages manuels.

- Terminal UART : ensemble de l’interface correspondant aux affichages des sorties UART. Il comprend les 2 zones d’affichages des messages UART, ainsi que les boutons d’options d’affichages qui permettent de choisir quel UART sera affiché, de le conserver lorsque l’on change de carte, et de nettoyer l’affichage.

# Table des versions et révisions

|  |  |  |  |
| --- | --- | --- | --- |
| Version/Révision | Date | Modifications | Auteurs |
| 0.1 | 14/11/2011 | Création du document | Julien FRANCHINEAU |
| 0.2 | 14/11/2011 | Ajout de la structure et des CU | François EVEN |
| 0.3 | 15/11/2011 | Ajout des frontières du système, lexique, IHM | Julien FRANCHINEAU |
| 0.4 | 17/11/2011 | Ajouts IHM, architecture matérielle, CU  Compléments du lexique  Ajout des livrables | Julien FRANCHINEAU |
| 1.0 | 17/11/2011 | Validation du document par Samuel | Samuel POIRAUD |
| 1.1 | 17/11/2011 | Ajouts compléments architecture matérielle | Julien FRANCHINEAU |
| 1.2 | 23/01/2012 | Mise à jour du document | François EVEN |