

## RAPPORT DE PROJET DE FIN D'ÉTUDES

Présenté en vue de l'obtention du

Diplôme National de Licence Fondamentale en Sciences de l'Informatique  
Spécialité : Sciences de l'Informatique

Par

**Marouen KADRI**

---

# CONCEPTION, DÉVELOPPEMENT ET RÉALISATION D'UNE SOLUTION DE TEST AUTOMATIQUE POUR LES PLATEFORMES LINUX EMBARQUÉS DE TÉLÉMATIQUE AUTOMOBILES

---

Encadrant professionnel : **Monsieur Wajdi ZAIRI**

Ingénieur R&D

Encadrante académique : **Madame Ines GAM**

Maître Assistante

Réalisé au sein de ACTIA





## RAPPORT DE PROJET DE FIN D'ÉTUDES

Présenté en vue de l'obtention du

Diplôme National de Licence Fondamentale en Sciences de l'Informatique  
Spécialité : Sciences de l'Informatique

Par

**Marouen KADRI**

---

# CONCEPTION, DÉVELOPPEMENT ET RÉALISATION D'UNE SOLUTION DE TEST AUTOMATIQUE POUR LES PLATEFORMES LINUX EMBARQUÉS DE TÉLÉMATIQUE AUTOMOBILES

---

Encadrant professionnel : **Monsieur Wajdi ZAIRI**

Ingénieur R&D

Encadrant académique : **Madame Ines GAM**

Maître Assistante

Réalisé au sein de ACTIA



J'autorise l'étudiant à faire le dépôt de son rapport de stage en vue d'une soutenance.

Encadrant professionnel, **Monsieur Wajdi ZAIRI**

**Signature et cachet**

J'autorise l'étudiant à faire le dépôt de son rapport de stage en vue d'une soutenance.

Encadrant académique, **Madame Ines GAM**

**Signature**

# Dédicace

Je dédie ce travail :

À ma chère mère et à mon cher père qui n'ont jamais cessé de me supporter, me soutenir et  
m'encourager durant mes années d'études.

Qu'ils trouvent ici le témoignage de ma profonde gratitude et reconnaissance.

À mes frères, mes grands-parents et ma famille qui me donnent de l'amour et de la vivacité.

À tous ceux qui m'ont aidé - de près ou de loin - et ceux qui ont partagé avec moi les  
moments d'émotion lors de la réalisation de ce travail et qui m'ont chaleureusement  
supporté et encouragé tout au long de mon parcours.

À tous mes amis qui m'ont toujours encouragé, et à qui je souhaite plus de succès.

Merci !

- *Marouen*

# Remerciements

Je tiens, avant de présenter mon travail, à exprimer ma grande reconnaissance envers les personnes qui m'ont - de près ou de loin - apporté leurs soutiens. Qu'ils trouvent ici collectivement et individuellement l'expression de toute ma gratitude et ma reconnaissance.

Je tiens à remercier tout particulièrement et à témoigner toute ma reconnaissance à M. **ZAIRI Wajdi**, pour l'expérience enrichissante et pleine d'intérêt qu'ils m'a fait vivre durant la période du stage, pour tous les conseils et les informations qu'il m'a prodigués et pour le temps qu'il a consacré à l'encadrement et le suivi de ce travail.

Je tiens à remercier également mon professeur encadrant M. **GAM Ines**, en dépit de leur multiples charges, leur aides et les renseignements précieux que'elle m'a fournis ainsi que pour tous les conseils et les informations qu'elle m'a prodigués avec un degré de patience et de professionnalisme sans égal.

Je tiens aussi à adresser mes plus sincères remerciements à l'ensemble du corps administratif et enseignant de l'ISI ARIANA, pour avoir porté un vif intérêt à notre formation, et pour avoir accordé de l'attention et de l'énergie, et ce, dans un cadre agréable de respect.

Que les membres de jury trouvent, ici, l'expression de mes remerciements pour l'honneur qu'ils me font en prenant le temps de lire et d'évaluer ce travail.

# Table des matières

<b>Introduction générale</b>	<b>1</b>
<b>1 Contexte général</b>	<b>3</b>
1.1 Présentation de l'organisme d'accueil . . . . .	4
1.1.1 Organisme d'accueil . . . . .	4
1.1.2 ACTIA ES historique . . . . .	4
1.1.3 L'innovation . . . . .	5
1.1.4 Département d'accueil : Linux embarqué . . . . .	5
1.2 Contexte du projet . . . . .	5
1.2.1 Cadre du projet . . . . .	5
1.2.2 Problématique . . . . .	5
1.2.3 Solution proposée . . . . .	6
1.3 Choix méthodologique . . . . .	6
<b>2 Etude préalable</b>	<b>8</b>
2.1 Test manuel . . . . .	9
2.1.1 Types de tests manuels . . . . .	9
2.2 Test automatisé . . . . .	10
2.2.1 L'intégration continue . . . . .	10
2.3 Robot-framework . . . . .	11
2.3.1 Pourquoi Robot Framework ? . . . . .	11
2.3.2 High-level architecture . . . . .	12
2.3.3 Keywords . . . . .	12
2.3.4 Variables . . . . .	13
2.4 La technologie BUS CAN . . . . .	13
2.4.1 Définition . . . . .	13
2.4.2 Les avantages du BUS CAN . . . . .	14
2.4.3 Les trames CAN . . . . .	14
2.4.4 Adaptateur PCAN-USB . . . . .	16

---

2.4.5	CAN sous Linux . . . . .	16
<b>3</b>	<b>Analyse et spécification des besoins</b>	<b>19</b>
3.1	Recensement des besoins . . . . .	20
3.1.1	Besoins fonctionnels . . . . .	20
3.1.2	Besoins non fonctionnels . . . . .	21
3.2	Identification des acteurs du système . . . . .	21
3.3	Spécification des besoins . . . . .	22
3.3.1	Diagramme de cas d'utilisation «Ingénieur test» . . . . .	22
3.3.2	Diagramme de cas d'utilisation «Jenkins» . . . . .	24
3.3.3	Diagramme de cas d'utilisation «Robotframework» . . . . .	25
<b>4</b>	<b>Architecture et conception du système</b>	<b>26</b>
4.1	Architecture globale du système . . . . .	27
4.2	Conception détaillée . . . . .	27
4.2.1	Diagramme de déploiement . . . . .	27
4.2.2	Aspect dynamique . . . . .	28
4.2.3	diagramme de séquence . . . . .	28
<b>5</b>	<b>Mise en oeuvre de la solution</b>	<b>30</b>
5.1	Outils et technologies utilisées : . . . . .	31
5.1.1	Équipements . . . . .	31
5.1.2	Outils de développement et de collaboration . . . . .	32
5.2	le montage des différents composants . . . . .	32
5.3	Quelques vues de la réalisation . . . . .	33
5.3.1	Interaction entre les interfaces du CAN BUS . . . . .	33
5.4	Configuration . . . . .	34
5.4.1	Configuration "Minicom" . . . . .	34
5.4.2	Configuration "jenkins" . . . . .	35
5.5	Exécution du test . . . . .	38
5.5.1	Exécution manuel du test . . . . .	38
	<b>Conclusion générale</b>	<b>39</b>

---

<b>Annexes</b>	<b>40</b>
Annexe 1. Exemple d'annexe . . . . .	40

# Table des figures

1.1	Quelque secteur d'activité. . . . .	4
1.2	Implémentation de la solution. . . . .	6
2.1	. . . . .	10
2.2	.... . . . .	12
2.3	la communication <b>ECU</b> dans les voitures . . . . .	14
2.4	la . . . . .	14
2.5	. . . . .	15
2.6	trame CAN détaillé . . . . .	15
2.7	. . . . .	16
3.1	.... . . . .	20
3.2	Les cas d'utilisation «Ingénieur test» . . . . .	22
3.3	Les cas d'utilisation «Jenkins» . . . . .	24
3.4	Les cas d'utilisation «Robotframework» . . . . .	25
4.1	. . . . .	27
4.2	. . . . .	28
4.3	. . . . .	29
5.1	Boite d'alimentation . . . . .	31
5.2	TGU . . . . .	32
5.3	Montage des composants. . . . .	33
5.4	Récapitulatif du processus de Synchronisation. . . . .	33
5.5	interface minicom . . . . .	35
5.6	Interface principale de Jenkins . . . . .	36
5.7	Ajouter des plugins . . . . .	36
5.8	Création d'un nouveau projet jenkins . . . . .	37
5.9	Interface principale de jenkins . . . . .	37
5.10	nterface principale de jenkins . . . . .	38
5.11	commande de test . . . . .	38

Table des figures

---

5.12 test passé . . . . .	38
---------------------------	----

# Liste des tableaux

Annexe 1.1 Exemple tableau dans l'annexe . . . . .	40
--	----

# Liste des abréviations

- **CAN** = Controller Area Network
- **ECU** = l'unité de commande électronique
- **TGU** = Telematic Gateway Unit

# Introduction générale

Les applications automobiles représentent actuellement la plus grande utilisation des systèmes embarqués et resteront probablement la plus grande partie dans les années à venir. Dans l'automobile, les systèmes embarqués sont utilisés pour l'infodivertissement, la sécurité, la sensibilisation du conducteur, la maintenance et le contrôle global du système du véhicule. L'augmentation des exigences pour les véhicules dotés de fonctions avancées de navigation, d'aide à la conduite et de communication véhicule-rue ne fera qu'augmenter la demande des systèmes intégrés. Pour tester les logiciels embarqués de manière efficace et efficiente, un grand nombre de techniques de test, d'approches, d'outils et de cadres ont été proposés par les praticiens et les chercheurs au cours des dernières décennies.

Les tests intégrés se réfèrent à la vérification et à la validation du comportement du logiciel et du matériel utilisant ce logiciel. Il garantit que le système embarqué dans son ensemble fonctionne parfaitement sans aucun bug / défaut. Le test intégré est effectué sur le matériel. Il aide également à documenter le développement du système et répond aux exigences des clients.

Les tests sont essentiels au succès de tout produit Si votre système ne fonctionne pas correctement, il est probable que la plupart des gens n'achèteront ou n'utiliseront même pas votre produit, du moins pas longtemps. qui peut nécessiter beaucoup de temps et de travail. Les tests manuels coûtent cher, il peut être difficile de les reproduire. Les outils qui peuvent automatiser les tests peuvent augmenter l'efficacité du processus et réduire les coûts. L'automatisation des outils permet également d'exécuter des tests dans un environnement d'intégration continue, réduisant ainsi les efforts nécessaires pour identifier et corriger les bugs lors du développement logiciel.

Le présent rapport synthétise ainsi le déroulement de mon travail sur ce projet. Il est structuré en cinq chapitres

Le premier chapitre comporte une brève présentation de l'organisme d'accueil ACTIA ES et du cadre général de ce projet. Il expose en effet, la problématique et met l'accent sur la solution proposée. Et il aborde à la fin la méthodologie appliquée pour assurer le bon déroulement de mon travail.

Le deuxième chapitre "Étude Préalable" permet de clarifier les concepts de base de notre projet

## Introduction générale

---

Dans le troisième chapitre "Analyse des besoins", nous présentons une analyse des besoins fonctionnels et non fonctionnels de la solution proposée ainsi que la modélisation de ces besoins par le recours aux diagrammes de cas d'utilisation.

Dans le quatrième chapitre "Architecture et conception du système", nous abordons l'architecture générale de l'application et offrons un aperçu des diagrammes de conception utilisés. Cet aperçu mènera à la conception générale des différentes fonctionnalités offertes. Cette conception est réalisée en utilisant le langage de modélisation UML en présentant le comportement statique et dynamique de la solution proposée.

Dans le Cinquième chapitre, il y a une présentation de la phase de réalisation du projet. Il présente les différents outils et les techniques utilisés ainsi que le résultat d'implantation

En fin je clôture par une conclusion générale qui présente une analyse du travail réalisé au sein de ACTIA ES. Cette partie met en évidence non seulement un résumé du travail effectué, mais aussi des propositions et des diverses perspectives.

## CONTEXTE GÉNÉRAL

---

### Plan

1	Présentation de l'organisme d'accueil . . . . .	4
2	Contexte du projet . . . . .	5
3	Choix méthodologique . . . . .	6

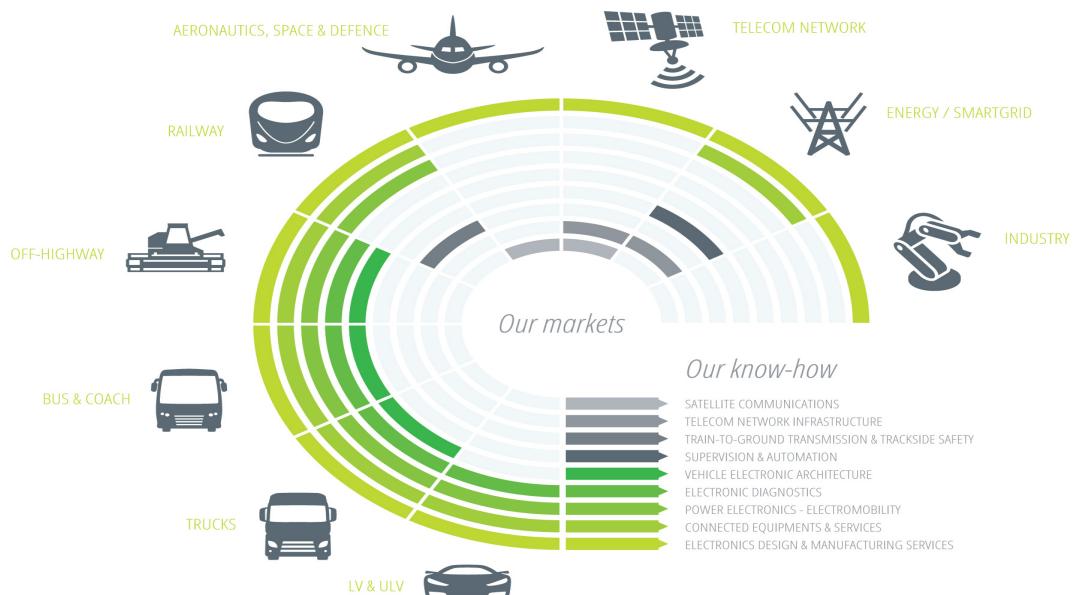
## Introduction

## 1.1 Présentation de l'organisme d'accueil

### 1.1.1 Organisme d'accueil

ACTIA GROUP est un groupe spécialisé dans la fabrication de composants électroniques pour la gestion des systèmes dans les domaines de l'automobile, des télécommunications et de l'énergie. Le groupe compte vingt-deux implantations dans seize pays, et son siège se situe à Toulouse dans le sud de la France.

ACTIA s'engage sur une politique d'innovation et une stratégie industrielle qui lui permet de faire la différence sur les marchés automotive comme le montre la figure 1.2.



**Figure 1.1:** Quelque secteur d'activité.

### 1.1.2 ACTIA ES historique

En Tunisie, ACTIA Group a créé en 1976 CIPI ACTIA qui est une SA franco-tunisienne totalement exportatrice. Son activité principale est la sous-traitance électronique : assemblage, test et maintenance de cartes électroniques pour plusieurs secteurs (automobile, télécom, électronique

grand public). En janvier 2009, le groupe a également créé la société ACTIA TUNISIE spécialisée en équipement de garage, elle emploie actuellement une trentaine de personnes.

### **1.1.3 L'innovation**

**ACTIA ES** s'associe à différents laboratoires et à d'autres industriels partout dans le monde pour développer des systèmes innovants et participer à l'élaboration des produits de demain. Membre fondateur de l'IRT St Exupéry, partenaire de la plateforme d'innovation thermique Fahrenheit Toulouse, partenaire de diverses écoles ou universités (France, Suède, Allemagne,...) ACTIA est engagé dans divers programmes de RD avancées dans une logique d'Open Innovation

### **1.1.4 Département d'accueil : Linux embarqué**

Le groupe est également présent au sein de diverses structures de normalisation, au plus près de ses métiers afin de contribuer à l'évolution des standards.

## **1.2 Contexte du projet**

### **1.2.1 Cadre du projet**

Notre projet intitulé :"Conception,développement et réalisation d'une solution de test automatique pour les plate-formes Linux embarqué de télématique automobiles **ACTIA ES**" a été proposé dans le cadre de l'élaboration d'un stage de fin d'études au sein de la société **ACTIA ES** pour répondre au besoin de l'un de ses équipes de développement.

### **1.2.2 Problématique**

Après Développement d'un système Linux embarqué pour les cartes **TGU** lance des tests manuelles sur la carte pour assurer le bon déroulement de ses fonctions logicielles durant de cette phase de test rapide on a :

- les risques d'erreurs humains
- la rendre le test très lourd après chaque mise a jour d'où la perte du temps

### 1.2.3 Solution proposée

Nous allons introduire une méthode qui nous permettra d'automatiser les tests et les exécuter dans un environnement d'intégration continue ,réduisant ainsi les efforts nécessaire pour identifier et corriger les bugs et réduire le coût et le temps nécessaire.

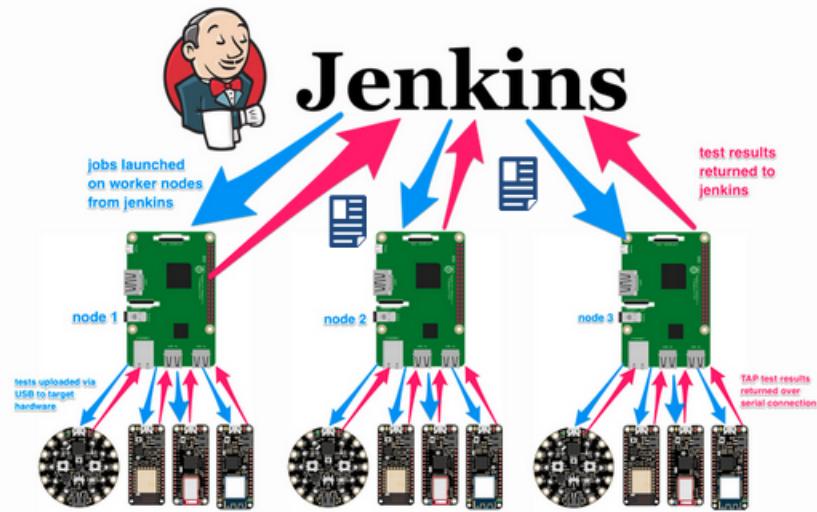


Figure 1.2: Implémentation de la solution.

### 1.3 Choix méthodologique

## Conclusion

Conclusion partielle ayant pour objectif de synthétiser le chapitre et d'annoncer le chapitre suivant.

---

*Chapitre 2*

---

## ETUDE PRÉALABLE

---

### Plan

<b>1</b>	<b>Test manuel . . . . .</b>	<b>9</b>
<b>2</b>	<b>Test automatisé . . . . .</b>	<b>10</b>
<b>3</b>	<b>Robot-framework . . . . .</b>	<b>11</b>
<b>4</b>	<b>La technologie BUS CAN . . . . .</b>	<b>13</b>

## Introduction

Introduction partielle, qui annonce le contenu.

### 2.1 Test manuel

Le test manuel est un test du logiciel où les tests sont exécutés manuellement . Il est effectué pour découvrir les bugs dans les logiciels en cours de développement. En test manuel, le testeur vérifie toutes les fonctionnalités essentielles de l'application ou du logiciel donné. Dans ce processus, les testeurs de logiciels exécutent les cas de test et génèrent les rapports sans l'aide d'outils de test de logiciels d'automatisation.

#### 2.1.1 Types de tests manuels

1. Black Box Testing : méthode de test de logiciel dans laquelle les testeurs évaluent la fonctionnalité du logiciel testé sans regarder la structure de code interne.
  
2. White Box Testing :Il est basé sur la structure de code interne des applications. Dans les tests en boîte blanche, une perspective interne du système, ainsi que des compétences en programmation, sont utilisées pour concevoir des cas de test. Ces tests sont généralement effectués au niveau de l'unité.
  
3. Unit Testing : Les tests unitaires sont également appelés tests de modules ou tests de composants.  
Il s'agit de vérifier si l'unité ou le module individuel du code source fonctionne correctement.
  
4. System Testing : Le test de l'application entièrement intégrée pour évaluer la conformité des systèmes avec ses exigences spécifiées est appelé Test de bout en bout du système AKA. Vérification du système terminé pour s'assurer que l'application fonctionne comme prévu ou non.
  
5. Integration Testing :Le test d'intégration est le processus de test de l'interface entre les deux unités logicielles. Les tests d'intégration se font de trois manières. Approche Big Bang, approche descendante, approche ascendante

6. Acceptance Testing :Il est également connu sous le nom de tests de pré-production. Cette opération est effectuée par les utilisateurs finaux ainsi que les testeurs pour valider la fonctionnalité de l'application. Après un test d'acceptation réussi. Des tests formels ont été effectués pour déterminer si une application est développée conformément à l'exigence. Il permet au client d'accepter ou de rejeter la demande. Les types de tests d'acceptation sont Alpha, Beta et Gamma.

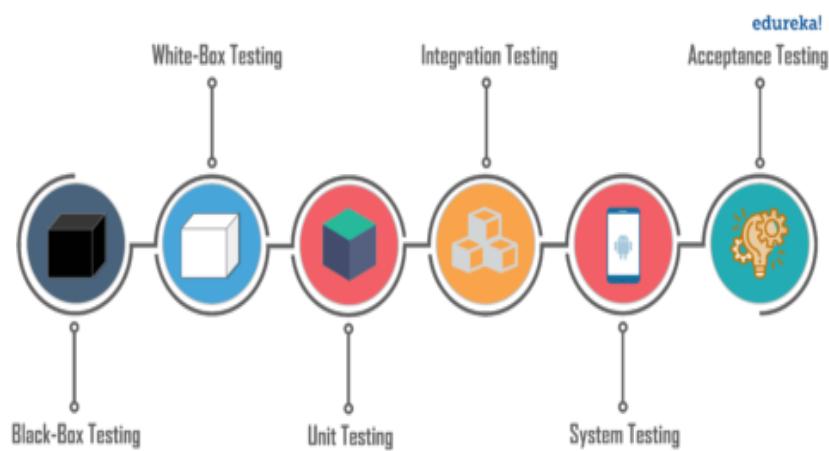


Figure 2.1: .

## 2.2 Test automatisé

Dans les tests de logiciels automatisés, les testeurs écrivent du code / des scripts de test pour automatiser l'exécution des tests. Les testeurs utilisent des outils d'automatisation appropriés pour développer les scripts de test et valider le logiciel. L'objectif est de terminer l'exécution du test en moins de temps. Les tests automatisés reposent entièrement sur le test pré-scripté qui s'exécute automatiquement pour comparer les résultats réels avec les résultats attendus. Cela permet au testeur de déterminer si une application fonctionne ou non comme prévu.

Les tests automatisés vous permettent d'exécuter des tâches répétitives et des tests de régression sans l'intervention d'un testeur manuel. Même si tous les processus sont exécutés automatiquement, l'automatisation nécessite un certain effort manuel pour créer des scripts de test initiaux.

### 2.2.1 L'intégration continue

L'intégration continue est une pratique de développement de logiciels **DevOps** où les développeurs fusionnent régulièrement leurs modifications de code dans une « branche » partagée, ou

un « tronc », parfois même tous les jours.Une fois que les modifications apportées par un développeur sont fusionnées, elles sont validées par la création automatique de l'application et l'exécution de différents niveaux de test automatisés (généralement des tests unitaires et d'intégration) qui permettent de vérifier que les modifications n'entraînent pas de dysfonctionnement au sein de l'application. En cas de détection d'un conflit entre le code existant et le nouveau code, le processus d'intégration continue permet de résoudre les dysfonctionnements plus facilement, plus rapidement et plus fréquemment.

Les principaux objectifs de l'intégration continue sont de trouver et de résoudre les bugs plus rapidement, d'améliorer la qualité des logiciels et de réduire le temps nécessaire pour valider et publier de nouvelles mises à jour logicielles.

## 2.3 Robot-framework

Les idées de base pour Robot Framework ont été formulées dans la thèse de master de **Pekka Klärck** en 2005. La première version a été développée chez **Nokia Networks** la même année. La version 2.0 est sortie en tant que logiciel open source le 24 juin 2008.Il s'agit d'un outil d'automatisation des tests IHM,BDD, serveurs et même des environnements mobiles open source .Il est basé sur python peut être utilisé dans des environnements distribués et hétérogènes, où l'automatisation nécessite l'utilisation de différentes technologies et interfaces.Il a un écosystème riche autour de lui composé de diverses bibliothèques génériques et d'outils qui sont développés en tant que projets séparés.

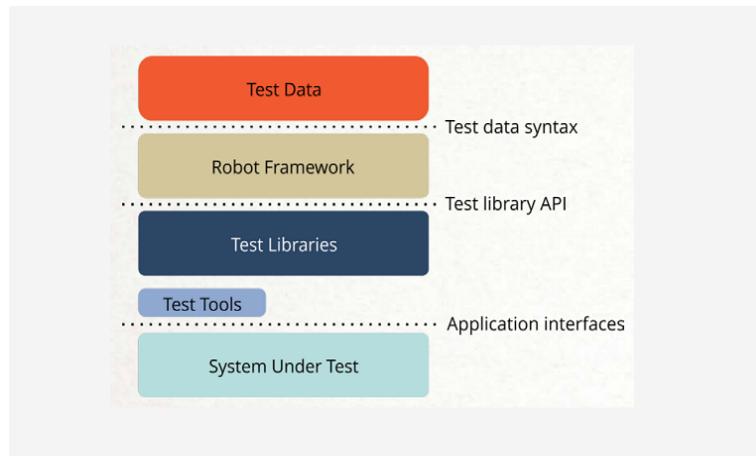
### 2.3.1 Pourquoi Robot Framework ?

- High-Level Architecture
- Simple Tabular Syntax
- Remote test libraries and other plugins for Jenkins/Hudson, Maven
- Data-driven Test Cases
- Clear Reports
- Detailed logs

### 2.3.2 High-level architecture

Robot Framework est un cadre générique, indépendant de l'application et de la technologie.

Il a une architecture hautement modulaire illustrée dans le schéma ci-dessous



**Figure 2.2:** ....

### 2.3.3 Keywords

Les cas de test sont créés avec des mots clés provenant de deux sources :

#### 2.3.3.1 Library Keywords

les mots clés de niveau le plus bas sont définis dans les bibliothèques standard qui peuvent être implémentées à l'aide de langages de programmation comme Python, Java, etc.

```
1 | *** Settings ***
2 | Library          OperatingSystem
3 | Library          lib/LoginLibrary.py
```

#### 2.3.3.2 User Keywords

L'une des puissantes fonctionnalités du robotframework est que nous pouvons créer des mots clés personnalisés de haut niveau à l'aide d'autres mots clés. Voyons un exemple pour comprendre comment cela fonctionne

```
Check The Reception Of ${MSG NAME} TimeOut ${TIME OUT} Seconds
    Check Msg      ${MSG NAME}      ${TIME OUT}      False      None

Check Message ${MSG NAME} TimeOut ${TIME OUT} Seconds Is Not Received
    Check Msg      ${MSG NAME}      ${TIME OUT}      True      None

Check The Reception Of ${MSG NAME} TimeOut ${TIME OUT} Seconds In Node ${NODE}
    Check Msg      ${MSG NAME}      ${TIME OUT}      False      ${NODE}

Check Message ${MSG NAME} TimeOut ${TIME OUT} Seconds Is Not Received In Node ${NODE}
    Check Msg      ${MSG NAME}      ${TIME OUT}      True      ${NODE}
```

### 2.3.4 Variables

Les variables sont une partie très importante de tout cas de test.

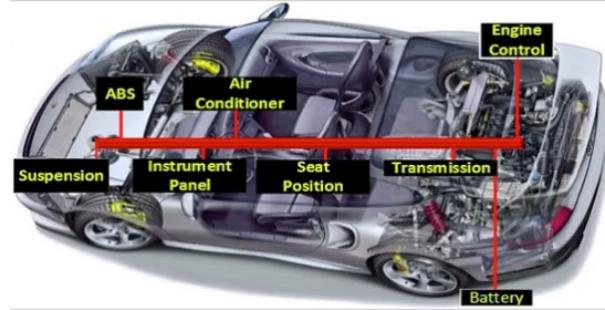
Voyons comment définir des variables dans un cas de test :

```
*** Variables ***
${USERNAME}          janedoe
${PASSWORD}          J4n3D0e
${NEW PASSWORD}      e0D3n4J
${DATABASE FILE}     ${TEMPDIR}${/}robotframework-quickstart-
db.txt
${PWD INVALID LENGTH}  Password must be 7-12 characters long
${PWD INVALID CONTENT}  Password must be a combination of lowercase
and uppercase letters and numbers
```

## 2.4 La technologie BUS CAN

### 2.4.1 Définition

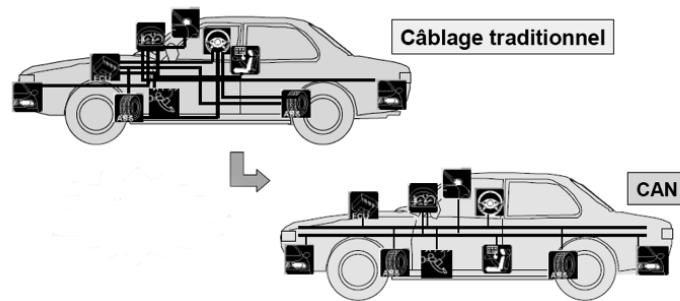
Le bus CAN est né en 1984 dans les ateliers BOSCH, initialement prévu pour des applications automobiles mais est également devenue un bus populaire dans l'automatisation industrielle ainsi que dans d'autres applications. Le bus CAN est principalement utilisé dans les systèmes embarqués et, comme son nom l'indique, est une technologie de réseau qui permet une communication rapide entre les micro-contrôleurs.



**Figure 2.3:** la communication ECU dans les voitures .

#### 2.4.2 Les avantages du BUS CAN

- Vitesse : une vitesse pouvant aller jusqu'à 1 Mbits/s
- Coût : les coûts de matériel réduits et les exigences minimales de traitement du signal font de CAN une solution idéale pour les applications intégrées nécessitant une communication multiprocesseur avec un budget limité.
- Fiabilité : la communication CAN nécessite également moins de câbles et de connecteurs, réduisant considérablement les points de défaillance.
- Flexibilité : Les systèmes connectés CAN contiennent non seulement beaucoup moins de fils, ce qui les rend plus faciles à installer, mais l'ajout de nouveaux composants à un système nécessite beaucoup moins de développement tout en réduisant considérablement les complications de diagnostic et de résolution des problèmes de signal



**Figure 2.4:** la

#### 2.4.3 Les trames CAN

Les communications sur le réseau CAN sont réalisées grâce à différentes trames, chaque trame permettant de transmettre une information spécifique (données, requêtes, surcharges ou erreurs).

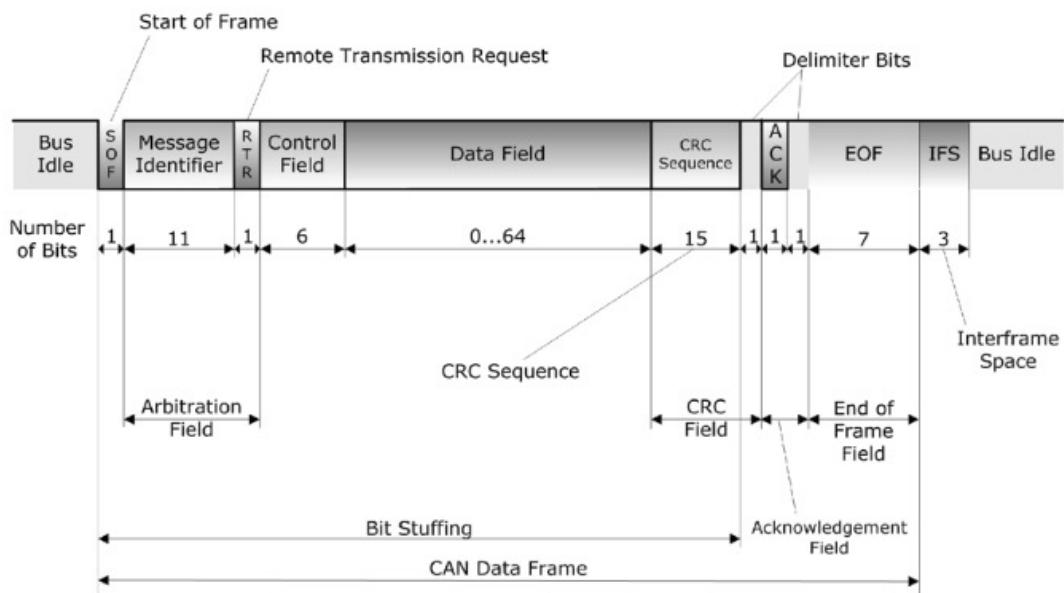
Ces trames sont composées de plusieurs champs permettant de définir tous les paramètres de la transmission

- SOF (Start of Frame) : Champ de départ de la trame toujours égal à 0.
- Arbitration Field : constitué de l'ID et du bit RTR(caractérise les Remote Frames)
- Control Field : Utilisé pour déterminer la taille des données et la longueur de l'ID du message
- Data Field : Ce sont les données transmises par la Data frame.
- CRC Field : permet de vérifier l'intégrité des données transmises
- ACK field : Le ACK field est composé de 2 bits, l'ACK Slot et le ACK Delimiter (1 bit récessif).  
Le noeud en train de transmettre envoie un bit récessif pour le ACK Slot. Un noeud ayant reçu correctement le message en informe le transmetteur en envoyant un bit dominant pendant le ACK Slot : il acquitte le message
- EOF (End of Frame) : Indique la fin de la transmission du message



**Figure 2.5:** .

L'image suivante montre la trame CAN plus détaillée :



**Figure 2.6:** trame CAN détaillé

#### 2.4.4 Adaptateur PCAN-USB

L'adaptateur PCAN-USB permet une connexion facile aux réseaux CAN. Grâce à son boîtier compact en matière plastique, il est non seulement très bien approprié aux applications mobiles



#### 2.4.5 CAN sous Linux

SocketCAN est un ensemble de pilotes CAN open source et une pile de mise en réseau(Networking stack) apportés par Volkswagen Research au noyau Linux. Anciennement connu sous le nom de Low Level CAN Framework (LLCF)

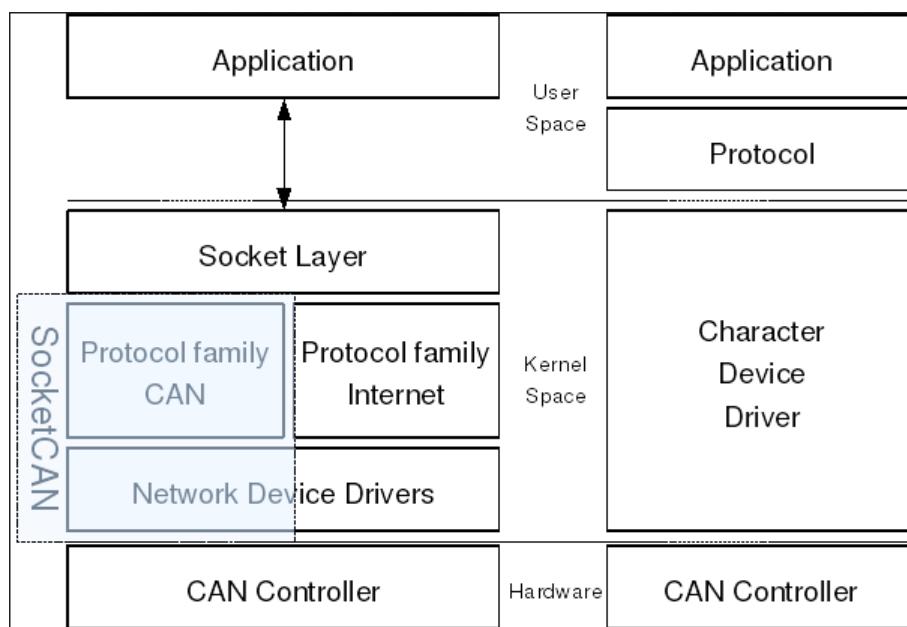


Figure 2.7: .

Sur Ubuntu, les modules de Kernel requis sont inclus, mais vous devrez installer **can-utils** :

- **sudo apt-get install can-utils**

#### **2.4.5.1 CAN-utils**

Les utilitaires CAN sont des outils permettant de travailler avec les communications CAN dans le véhicule à partir du système d'exploitation Linux. Ces outils peuvent être divisés en plusieurs groupes fonctionnels :

1. Basic tools to display, record, generate and play can traffic
2. CAN access via IP sockets
3. CAN in-kernel gateway configuration
4. Can Bus measurement
5. SO-TP tools
6. Log file converters
7. Serial line discipline (slc) configuration

Dans notre projet on a utilisé des Outils de base pour afficher, enregistrer, générer et rejouer le trafic CAN :

- **candump** :afficher, filtrer et enregistrer les données CAN dans des fichiers
- **canplayer** :rejouer les fichiers journaux CAN
- **cansend** :envoyer une seule trame
- **cangen** :générer du trafic CAN (aléatoire)
- **cansniffer** :afficher les différences de contenu des données CAN

#### **2.4.5.2 Configuration d'un réseau CAN virtuel**

1. chargez le module Vcan (CAN virtuel) :
  - **sudo modprobe vcan**
2. configurer l'interface virtuelle :
  - **ip link add dev can0 type vcan**
  - **ip link set up vcan0**

## Conclusion

Conclusion partielle ayant pour objectif de synthétiser le chapitre et d'annoncer le chapitre suivant.

## **ANALYSE ET SPÉCIFICATION DES BESOINS**

---

### **Plan**

<b>1</b>	<b>Recensement des besoins . . . . .</b>	<b>20</b>
<b>2</b>	<b>Identification des acteurs du système . . . . .</b>	<b>21</b>
<b>3</b>	<b>Spécification des besoins . . . . .</b>	<b>22</b>

## Introduction

Le présent chapitre est consacré à l'identification et l'analyse des besoins qui représentent une phase importante dans le cycle de développement d'un logiciel. Pour ce faire, Nous présentons, en premier lieu, les acteurs de notre système, puis nous spécifions les besoins fonctionnels et non fonctionnels. Ensuite, nous modélisons ces besoins avec les diagrammes des cas d'utilisation en décrivant et analysant leurs fonctionnalités majeures.

### 3.1 Recensement des besoins

Suite à l'étude de la problématique, nous avons dégagé l'ensemble des exigences que notre solution doit satisfaire. Ces exigences sont divisées en besoins fonctionnels, besoins non fonctionnels

#### 3.1.1 Besoins fonctionnels

Notre solution doit tester les fonctionnalités de ces composants

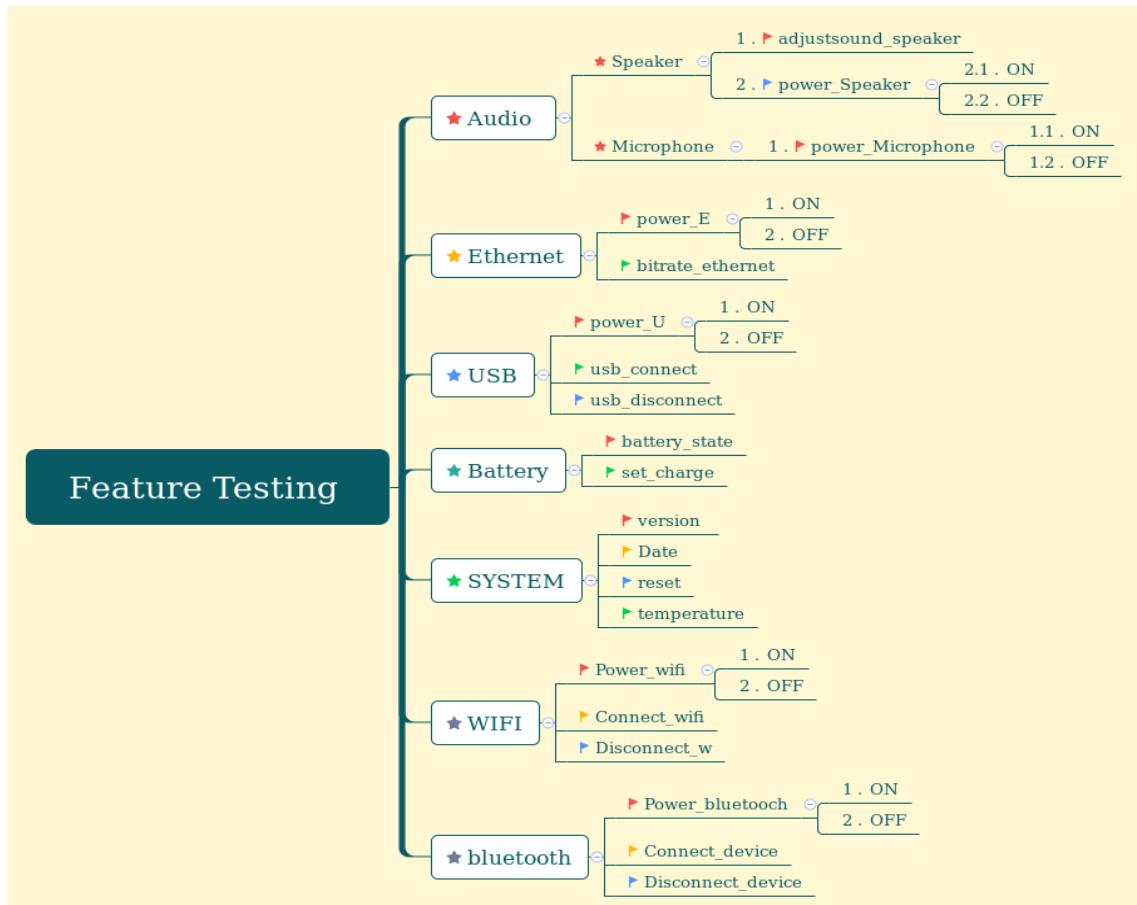


Figure 3.1: .....

### 3.1.2 Besoins non fonctionnels

la solution à réaliser doit répondre à un certain nombre d'exigences non fonctionnelles. Elle se caractérise par :

- **Maintenabilité et Évolutive** : Notre application doit permettre une maintenance facile et être susceptible de subir des évolutions fonctionnelles rapidement, au moindre coût, de manière rapide et fiable
- **Fiabilité** : Le système doit garantir l'intégrité et la cohérence des données à chaque mise à jour
- **Performance** : Le temps des décideurs est précieux. Les réponses doivent être par conséquent fournies dans un délai très réduit.

## 3.2 Identification des acteurs du système

Un acteur est une personne, un matériel ou un logiciel qui interagit avec le système. L'analyse du présent projet commence par une identification des acteurs agissants sur les différentes parties du système.

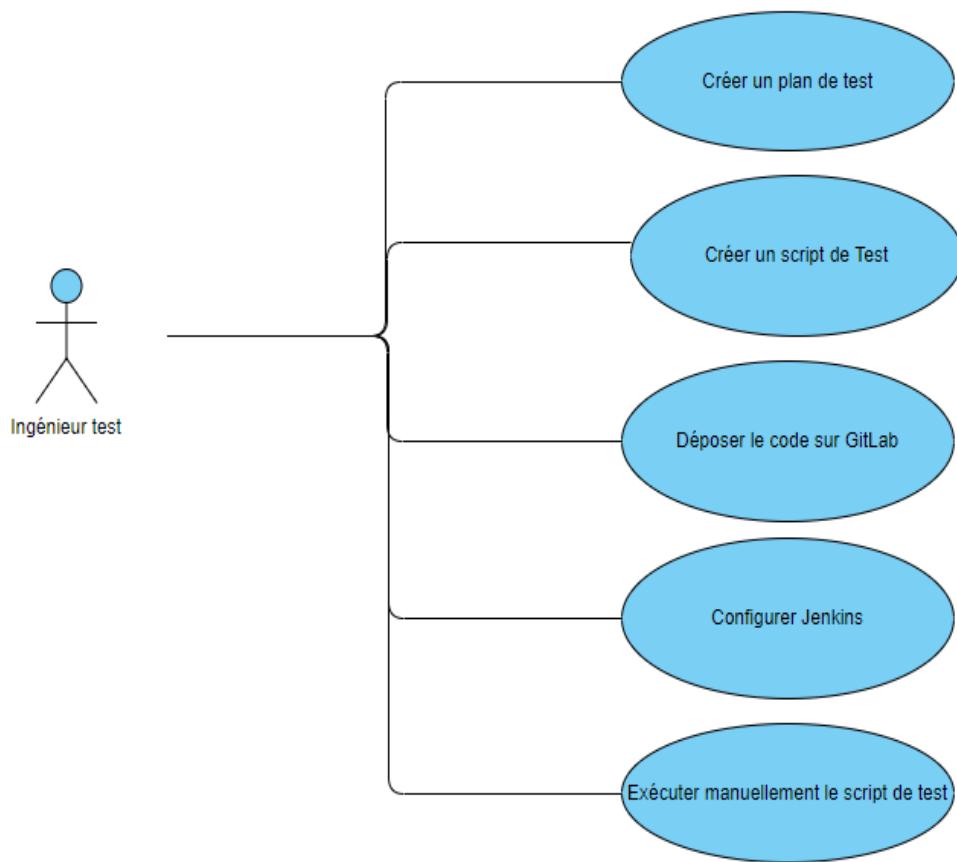
Le tableau 2.1 récapitule les acteurs en interaction avec le système en spécifiant le rôle de chacun avant de définir plus précisément leurs interactions avec le système en utilisant des diagrammes de cas d'utilisation.

Acteur	Fonction
Ingénieur test	Créer un plan de test Créer un script de test Déposer le code sur GitLab Configurer Jenkins Exécuter manuellement le script de test
Robotframework	Communiquer avec la carte "TGU"
Jenkins	Envoyer un build au Robotframework et récupérer les sorties

### 3.3 Spécification des besoins

L'analyse des besoins est une étape très importante dans le processus de l'étude et le développement des systèmes d'informations. Cette partie identifie l'ensemble des acteurs qui interagissent avec le système et définit l'ensemble des cas d'utilisation de ce dernier en se basant sur les diagrammes UML.

#### 3.3.1 Diagramme de cas d'utilisation «Ingénieur test»



**Figure 3.2:** Les cas d'utilisation «Ingénieur test».

- **Cas d'utilisation "Crée un plan de test" :**

- **Objectif :** Cette fonctionnalité permet aux testeurs de connaître le comportement de chaque cas de test
- **Acteur :** Ingénieur test
- **Scénario nominal :**

1. L'acteur crée une table qui contient une description détaillé de chaque cas de test.

• **Cas d'utilisation "Créer un script de test"** :

— *Objectif* :Cette fonctionnalité permet de créer un script de test

— *Acteur* :Ingénieur test

— *Scénario nominal* :

1. L'acteur crée un script de test

• **Cas d'utilisation "Déposer le code sur GitLab"** :

— *Objectif* :Cette fonctionnalité permet :

— viter des pertes de données

— Une modularisation aisée de son projet

— *Acteur* :Ingénieur test

— *Scénario nominal* :

1. L'acteur dépose son code sur GitLab.

• **Cas d'utilisation "Configurer Jenkins"** :

— *Objectif* :Ingénieur test

— *Acteur* :Ingénieur test.

— *Scénario nominal* :

1. L'acteur positionne les

• **Cas d'utilisation "Exécuter manuellement le script de test"** :

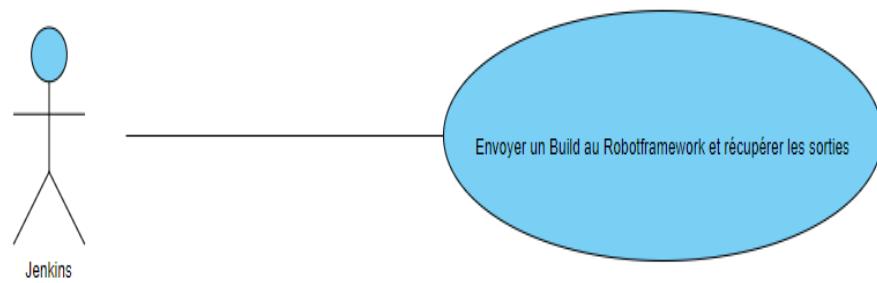
— *Objectif* :

— *Acteur* :Ingénieur test.

— *Scénario nominal* :

1. L'acteur positionne

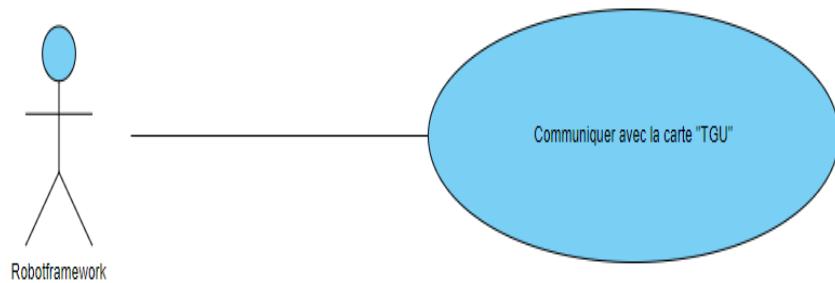
### 3.3.2 Diagramme de cas d'utilisation «Jenkins»



**Figure 3.3:** Les cas d'utilisation «Jenkins»

- Cas d'utilisation "Envoyer un Build au Robotframework et récupérer les sorties" :
  - *Objectif* :
  - *Acteur* : Jenkins
  - *Scénario nominal* : Jenkins lance un build au Robotframework et récupère les sorties

### 3.3.3 Diagramme de cas d'utilisation «Robotframework»



**Figure 3.4:** Les cas d'utilisation «Robotframework»

- **Cas d'utilisation "Communiquer avec la carte "TGU" " :**

- **Objectif :** tester les fonctionnalités des composants
- **Acteur :** Robotframework
- **Scénario nominal :**
  - Robotframework lance un test qui est basé sur l'envoie et la réception des trames CAN

## Conclusion

Conclusion partielle ayant pour objectif de synthétiser le chapitre et d'annoncer le chapitre suivant.

---

*Chapitre 4*

---

# ARCHITECTURE ET CONCEPTION DU SYSTÈME

---

## Plan

1	Architecture globale du système . . . . .	27
2	Conception détaillée . . . . .	27

## Introduction

Introduction partielle, qui annonce le contenu.

### 4.1 Architecture globale du système

### 4.2 Conception détaillée

#### 4.2.1 Diagramme de déploiement

Le diagramme de déploiement décrit la disposition physique des matériels qui composent le système et la répartition des entités logiques identifiées dans la conception architecturale sur ces matériels. Chaque machine est représentée par un nœud

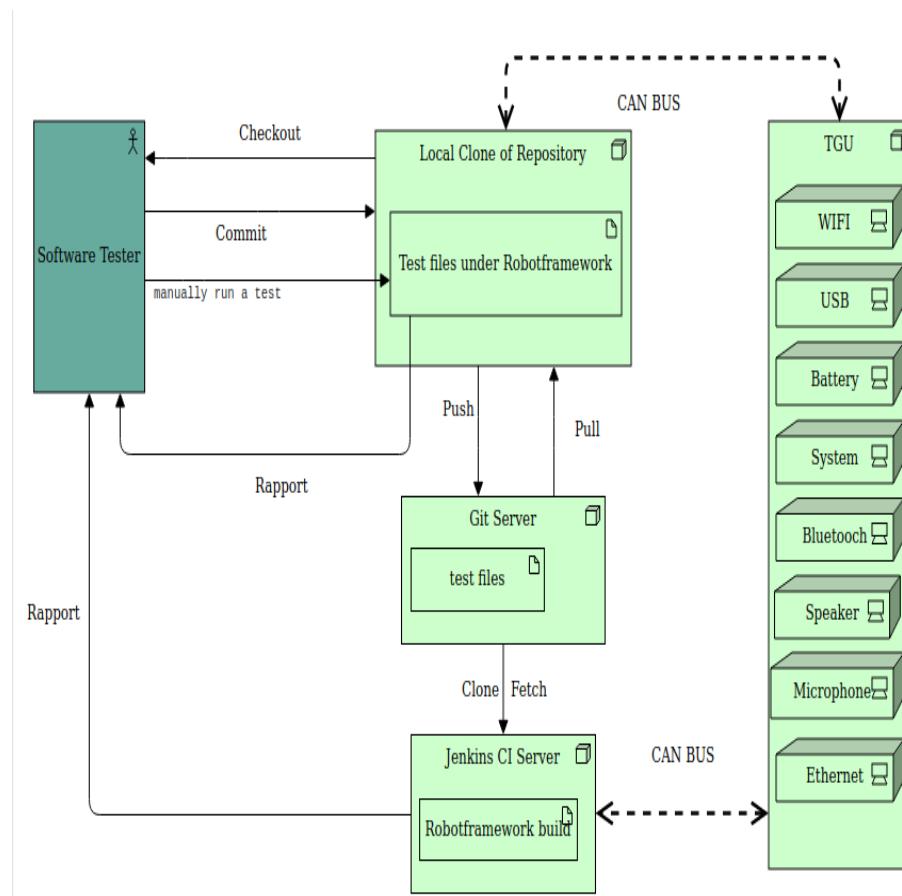


Figure 4.1: .

#### 4.2.2 Aspect dynamique

La vue dynamique permet de modéliser le comportement dynamique de notre système en indiquant comment ses objets interagissent au moment de l'exécution. Pour se faire, nous utilisons les diagrammes de séquence.

#### 4.2.3 diagramme de séquence

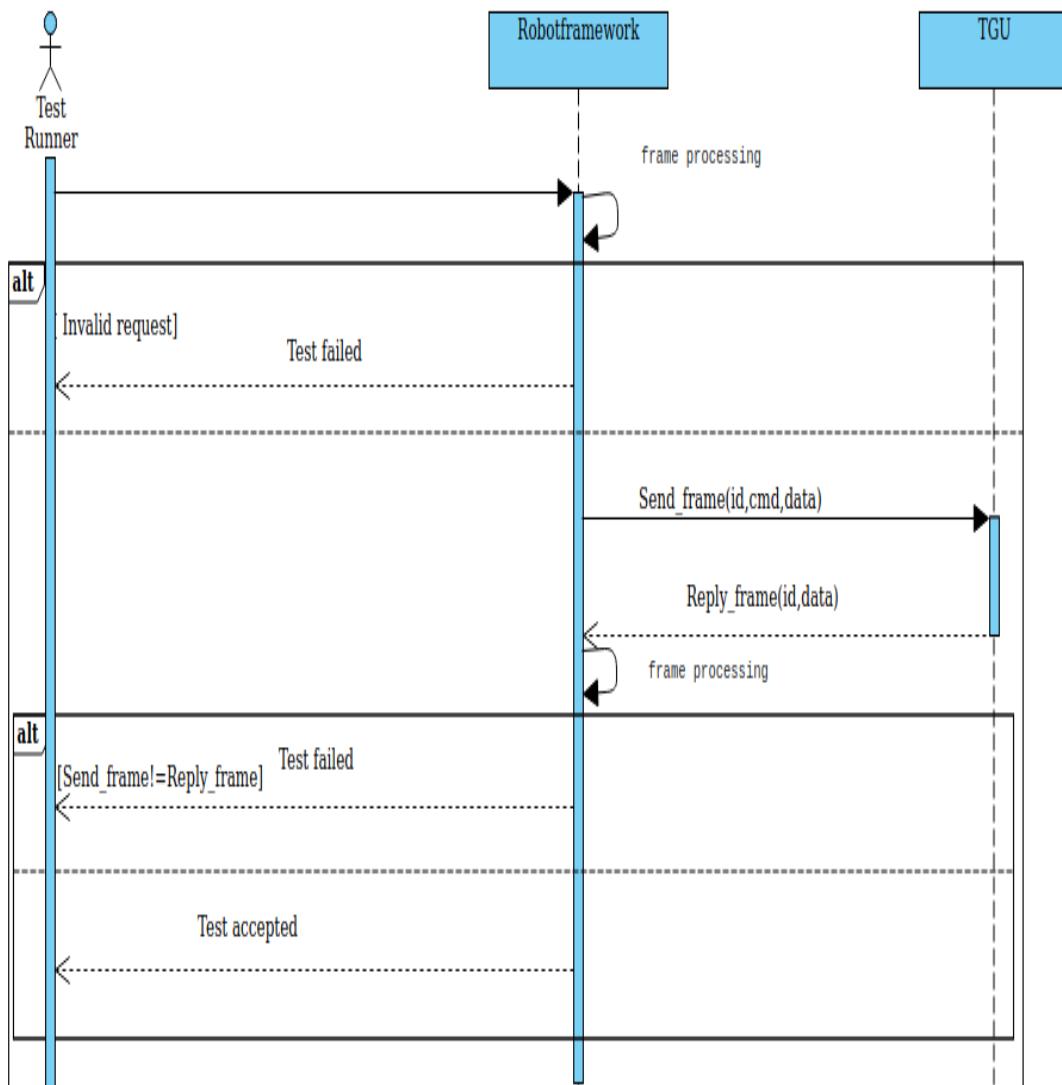


Figure 4.2: .

#### 4.2.3.1 diagramme d'activité

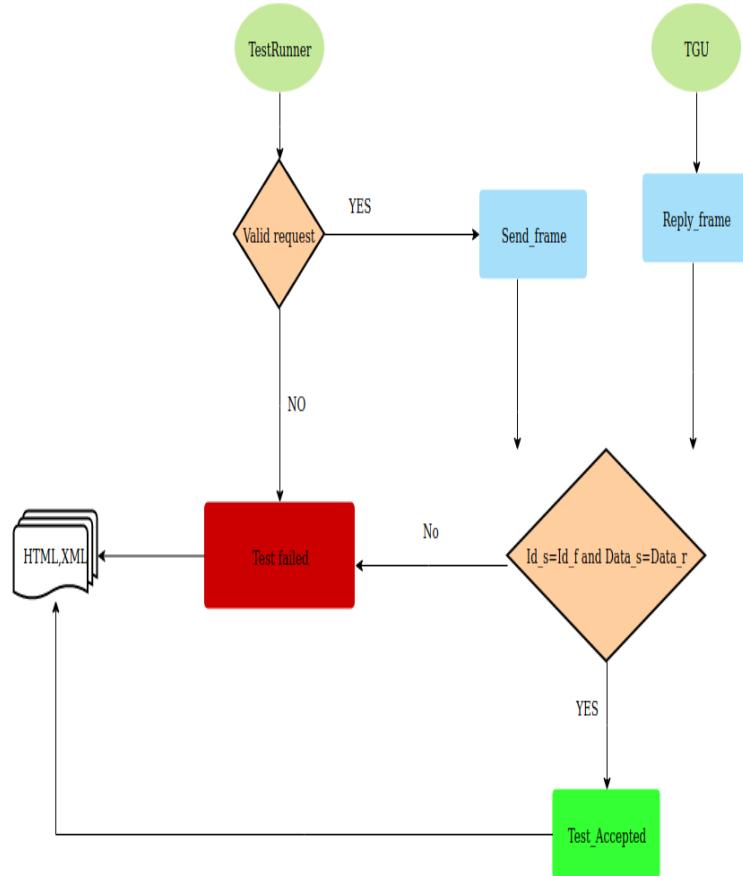


Figure 4.3: .

## Conclusion

Conclusion partielle ayant pour objectif de synthétiser le chapitre et d'annoncer le chapitre suivant.

# MISE EN OEUVRE DE LA SOLUTION

---

## Plan

<b>1</b>	<b>Outils et technologies utilisées :</b>	<b>31</b>
<b>2</b>	<b>le montage des différents composants</b>	<b>32</b>
<b>3</b>	<b>Quelques vues de la réalisation</b>	<b>33</b>
<b>4</b>	<b>Configuration</b>	<b>34</b>
<b>5</b>	<b>Exécution du test</b>	<b>38</b>

## Introduction

Dans ce chapitre, on va présenter les outils utilisés pour la mise en oeuvre de la solution test automatisé ainsi qu'un aperçu des différentes vues de cette solution.

Cette étape constitue la phase d'achèvement et d'aboutissement du projet. Pour accomplir cette tâche avec succès il faut savoir utiliser les outils adéquats et nécessaires. Ce choix d'outils peut influencer la qualité du produit obtenu et donc nécessite une attention particulière et doit se baser sur les besoins du projet et le résultat escomptés.

Ce chapitre présente l'environnement technique du travail ainsi que le choix pris en matière d'environnement logiciel.

### 5.1 Outils et technologies utilisées :

#### Python

Python est un langage de programmation puissant et facile à apprendre. Il dispose de structures de données de haut niveau et permet une approche simple mais efficace de la programmation orientée objet.

##### 5.1.1 Équipements

###### 5.1.1.1 Boite d'alimentation à courant continu

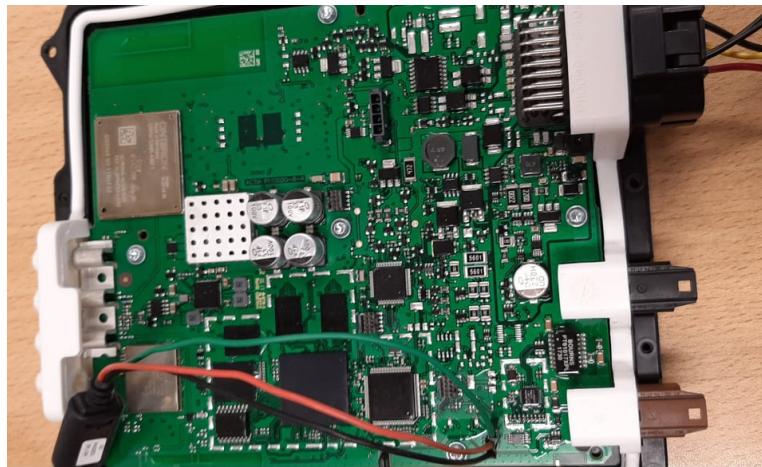
L'alimentation fournit un courant électrique (12V 24V)



**Figure 5.1:** Boite d'alimentation

### 5.1.1.2 TGU

c'est l'unité de contrôle de la connectivité et de la mise en réseau est au cœur du système télématique complexe présent dans les véhicules les plus récents. Elle doit fournir toute la connectivité intra-système à un certain nombre de modules ou sous-systèmes tels que les modules de connectivité (Bluetooth, Wi-Fi, etc.)



**Figure 5.2:** TGU

### 5.1.2 Outils de développement et de collaboration

#### Pycharm

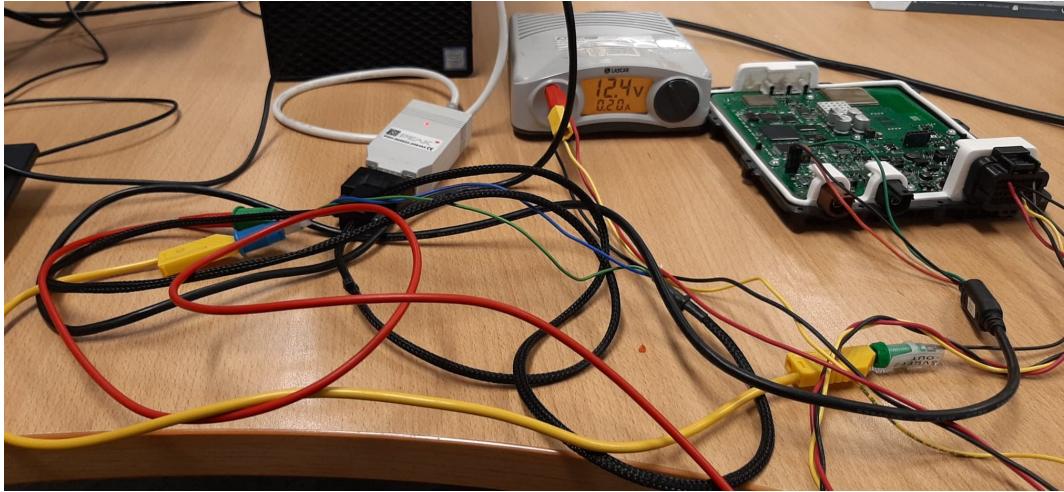
PyCharm est un environnement de développement intégré utilisé pour programmer en Python.

## 5.2 le montage des différents composants

Tout au long de la phase d'implantation et de test, nous avons réalisé le montage de la figure le montage est composé de :

- TGU
- CAN BUS
- PCAN-USB
- boîte d'alimentation à courant continu
- PC

Afin de donner une vision plus claire sur le montage utilisé la figure montre une photo réelle de ce dernier



**Figure 5.3:** Montage des composants.

## 5.3 Quelques vues de la réalisation

### 5.3.1 Interaction entre les interfaces du CAN BUS

notre solution basé sur l'échange des trames CAN entre les différentes interfaces du CAN BUS d'où des noeuds envoient et d'autre reçoivent. la figure montre les échanges des trames

```

Activités Terminator
jeu. 14:07 ● ard@pfe-lsw-02: ~ 8x22
ard@pfe-lsw-02:~$ cansend vcan1 01#1R
ard@pfe-lsw-02:~$ cansend vcan1 01#0a1b2
ard@pfe-lsw-02:~$ cansend vcan1 01#0a1b2
ard@pfe-lsw-02:~$ cansend vcan1 02#AAR
ard@pfe-lsw-02:~$ candsmp vcan1 any
vcan1 5B1 [3] 3B 3B 3B
vcan1 5A1 [3] 1A 2B 3C
vcan1 3A1 [3] 1A 2B 3A
vcan2 011 [0] remote request
vcan2 01A [3] A1 B2 C3
vcan2 01A [3] A1 2A 3B

jeu. 14:07 ● ard@pfe-lsw-02: ~ 8x22
ard@pfe-lsw-02:~$ candump vcan0
vcan1 011 [0] remote request
vcan1 011 [2] A1 B2
vcan2 5A1 [3] E1 D2 A3
vcan2 01B [3] A1 B2 C3
vcan3 01E [0] remote request
vcan3 01A [2] A1 B2
^Card@pfe-lsw-02:~$ cansend vcan0 01#1R
ard@pfe-lsw-02:~$ candsmp vcan0 01#4d4e
ard@pfe-lsw-02:~$ cansend vcan0 3A#1a2b3a
ard@pfe-lsw-02:~$ 

ard@pfe-lsw-02:~$ cansend vcan2 5A1#1d23
ard@pfe-lsw-02:~$ candsmp vcan2 01#0a1b2c3
ard@pfe-lsw-02:~$ candump vcan2 any
vcan0 011 [0] remote request
vcan0 011 [2] 4E 4E
vcan0 01B [3] A1 2B 3C
vcan1 01A [2] A1 2B
vcan1 02A [0] remote request
^Card@pfe-lsw-02:~$ cansend vcan2 01#1R
ard@pfe-lsw-02:~$ candsmp vcan2 01#1R
ard@pfe-lsw-02:~$ cansend vcan2 01#0a1b2c3
ard@pfe-lsw-02:~$ cansend vcan2 01#A#012a3b
ard@pfe-lsw-02:~$ 

```

**Figure 5.4:** Récapitulatif du processus de Synchronisation.

## 5.4 Configuration

### 5.4.1 Configuration "Minicom"

#### Install

Utilisez apt-get sous Debian / Ubuntu Linux, entrez :

- **sudo apt-get install minicom**

#### configuration

1. Tout d'abord vérifier que le système a bien détecté le port sur lequel vous voulez pointer. En faisant une petite visualisation de log :

- **sudo dmesg | grep tty**

2. Pour identifier le port que vous voulez :

- **sudo setserial -g /dev/tty\***

3. Une fois le port identifié on va pouvoir dialoguer avec grâce à minicom :

- **sudo minicom**

4. On peut préciser le périphérique comme cela :

- **sudo minicom -D /dev/ttyUSB0**

```
cmd: DIGIPEATER ON
?
?md:
cmd:+-----+
? | A - Serial Device      : /dev/ttyS0
cmd:| B - Lockfile Location : /var/lock
? | C - Callin Program    :
cmd:| D - Callout Program   :
OK | E - Bps/Par/Bits     : 4800 8N1
cmd:| F - Hardware Flow Control : No
ECHO| G - Software Flow Control : No
TXDE|
GPS | Change which setting? |
MONi+-----+
DIGIpeater 0| Screen and keyboard
BEACON On EV| Save setup as dfl
UNPROTO GPSC| Save setup as..
MYCALL TF2SU| Exit
MYALIAS +-----+
BTEXT >DiP-M
OK
cmd: PERM
OK
CTRL-A Z for help | 4800 8N1 | NOR | Minicom 2.4 | ANSI | Offline
```

Figure 5.5: interface minicom

#### 5.4.2 Configuration "jenkins"

Après avoir configuré la partie technique, nous entamons la partie fonctionnelle. Cette partie consiste à configurer l'orchestrateur jenkins pour communiquer avec les autres composants de notre architecture afin d'assurer le bon fonctionnement de l'intégration continue. Ci-dessous l'interface principale jenkins dès son installation.



Figure 5.6: Interface principale de Jenkins

#### 5.4.2.1 Ajouter des plugings



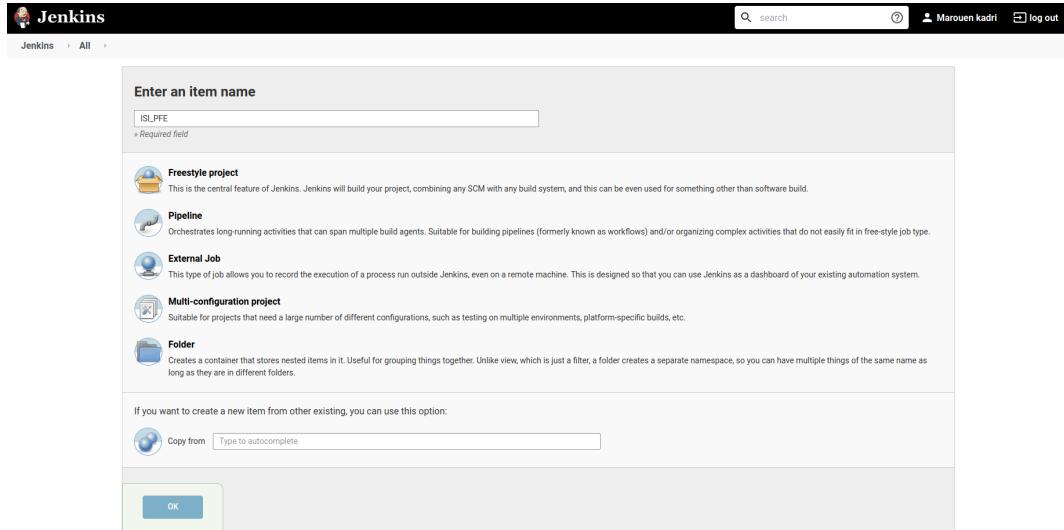
Figure 5.7: Ajouter des plugins

- nous avons suivons nous avons suivi la tache "Manage jenkins"

## Chapitre 5. Mise en oeuvre de la solution

- Dans la barre de recherche on a tapé Robotframework plugin ,Gitlab, Git
- En fin nous cliquons sur "Install now"

Création d'un nouveau job jenkins de type pipeline pour tourner le projet .

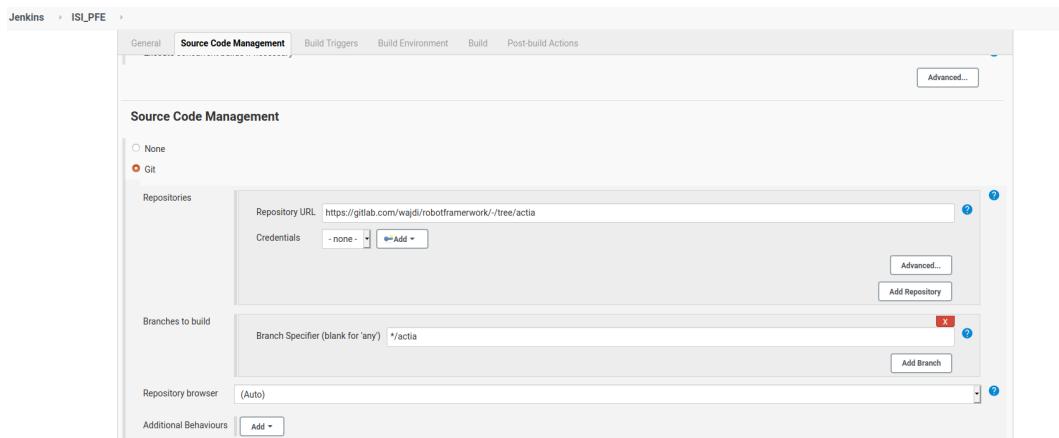


**Figure 5.8:** Crédation d'un nouveau projet jenkins

- Nous avons saisi le nom du job
- Nous cochons par la suite le type du job : Freestyle project
- En fin nous cliquons sur OK

### 5.4.2.2 Configuration du job

Ce qui nous concerne principalement est la partie source code management et la partie Build



**Figure 5.9:** Interface principale de jenkins

- Nous cochons l'option git

- on a entré l'URL du dépôt Git que nous utilisons
- on a ajouté le nom de la branche utilisé



Figure 5.10: nterface principale de jenkins

- on a choisie le script de commande : shell
- Nous avons écrit un script pour lancer notre test

## 5.5 Exécution du test

### 5.5.1 Exécution manuel du test

- lancer une commande de test

```
robot -t "Send a CAN frame" testsuite/canRaw.robot
```

Figure 5.11: commande de test

— on

```
=====
canRaw
=====
Send a CAN frame | PASS |
-----
canRaw | PASS |
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed
=====
Output: /home/arl/wajdi-workspace/Personal_Space/stages-robotframework/robotframework-can-uds-library/CURF/output.xml
Log:   /home/arl/wajdi-workspace/Personal_Space/stages-robotframework/robotframework-can-uds-library/CURF/log.html
Report: /home/arl/wajdi-workspace/Personal_Space/stages-robotframework/robotframework-can-uds-library/CURF/report.html
arl@arl-ThinkPad-E590:~/wajdi-workspace/Personal_Space/stages-robotframework/robotframework-can-uds-library/CURF$
```

Figure 5.12: test passé

## Conclusion

Conclusion partielle ayant pour objectif de synthétiser le chapitre et d'annoncer le chapitre suivant.

# Conclusion générale

La première itération du projet consistait en la définition du périmètre du projet. Les itérations qui suivaient ont été consacrées à la réalisation de plusieurs modules métiers de base comme le flux d'import et la gestion des profils. Lors de chaque itération, une étude fonctionnelle a été effectuée suivie d'une conception détaillée permettant par la suite d'enchaîner la phase de réalisation et de développement des modules en question.

Le présent projet m'a apporté de la valeur ajouté sur plusieurs niveaux. D'une part, j'ai appris à mieux gérer mon temps, mieux communiquer et collaborer avec les autres membres de l'équipe de développement. D'autre part, il m'a permis d'acquérir des compétences extrêmement importantes telle que le maintien et l'optimisation de mon code, l'intégration des bonnes pratiques, l'autonomie ainsi qu'une grande capacité de modélisation.

En terme de perspectives, le projet dans sa version actuelle n'est pas encore achevé, on est actuellement en cours du personnalisation de l'extension backoffice pour faciliter encore plus l'accès et la gestion des informations des produits. Les prochaines itération porteront toujours sur ce sujet en plus de l'évolution du processus d'export des produits pour qu'il soit beaucoup plus personnalisable par l'utilisateur.

# Annexes

## Annexe 1. Exemple d'annexe

Les chapitres doivent présenter l'essentiel du travail. Certaines informations-trop détaillées ou constituant un complément d'information pour toute personne qui désire mieux comprendre ou refaire une expérience décrite dans le document- peuvent être mises au niveau des annexes. Les annexes, **placées après la bibliographie**, doivent donc être numérotées avec des titres (Annexe1, Annexe2, etc.).

Le tableau annexe 1.1 présente un exemple d'un tableau dans l'annexe.

**Tableau annexe 1.1 : Exemple tableau dans l'annexe**

0	0
1	1
2	2
3	3
4	4

ملخص

يوضع الملخص باللغة العربية هنا... الرجاء أن يكون في حدود العشر أسطر... يوضع الملخص باللغة العربية هنا... الرجاء أن يكون في حدود العشر أسطر... يوضع الملخص باللغة العربية هنا... الرجاء أن يكون في حدود العشر أسطر... يوضع الملخص باللغة العربية هنا... الرجاء أن يكون في حدود العشر أسطر... يوضع الملخص باللغة العربية هنا... الرجاء أن يكون في حدود العشر أسطر... يوضع الملخص باللغة العربية هنا... الرجاء أن يكون في حدود العشر أسطر... يوضع الملخص باللغة العربية هنا... الرجاء أن يكون في حدود العشر أسطر... يوضع الملخص باللغة العربية هنا... الرجاء أن يكون في حدود العشر أسطر... يمكّن أن تكتب كلمات بحروف لاتينية في وسط الملخص مثال Exemple ici يوضع الملخص باللغة العربية هنا...

**كلمات مفاتيح :** الرجاء عدم تجاوز الخمس كلمات

## Résumé

**Mots clés :** Merci de ne pas dépasser les cinq mots

## Abstract

**Keywords :** Please don't use more than five keywords