

## Rapport de Projet de SFE

Préparé par

**ANAS SAFI  
AKRAM KADRI  
YOUNES AMHIL  
MOHAMED ABDELBAR  
SOUFIANE EL FATHI LALAOUI**

Pour l'obtention du diplôme

Diplôme Universitaire de Technologie (D.U.T)

Option : Génie Informatique

Projet intitulé

**Réalisation et intégration d'une solution de  
*Smart-Logistique* pour la traçabilité de  
produits durant le cycle d'entreposage**

Encadré par :      **PR. Hicham BELHEDDAOUI**  
                         **PR. Nadia AFIFI**  
                         **PR. Mounir RIFI**



# Remerciements

En préambule à ce rapport, nous tenons à présenter nos sincères remerciements et gratitude à tous ceux qui ont contribué à sa mise en place et rendu possible sa réalisation.

Nous avons l'honneur, en marge de ce travail, d'exprimer notre profonde gratitude ainsi que toute notre reconnaissance à *M. Hicham BELHEDDAOUI* pour le temps qu'il nous a accordé, l'intérêt avec lequel il a suivi la progression de notre travail, ses conseils judicieux, ses directives claires et pour tous les moyens qu'il a mis à notre disposition.

Nos remerciements vont aussi particulièrement à nos encadrants *Mme. Nadia AFIFI* et *M. Mounir RIFI* qui nous ont sans cesse encouragés, conseillé et motivé tout au long des différentes phases de développement et face à l'accroissement de leur difficulté. Nous remercions aussi *M. Charif MAHMOUDI* pour le savoir-faire nous ayant été transmis de sa part.

Notre gratitude va ensuite à tout le corps professoral de l'ESTC pour la formation de qualité qu'ils nous ont prodiguée durant ces deux années, et spécialement *M. Mohamed OUZZIF* et *M. Khalid BOURAGBA* pour tous les efforts extrascolaires et administratifs qu'il ont veillé à prodiguer sans retenue.

Nous tenons finalement à remercier le laboratoire RITM pour l'honneur qu'il nous a fait en acceptant de nous confier la réalisation de ce travail. Qu'il accepte l'assurance de notre estime et de notre profond respect, et que toute l'équipe trouve ici l'expression de notre entière gratitude.

# Résumé

Le présent rapport synthétise le travail effectué dans le cadre du projet de fin d'études au sein du labo RITM. Ce projet a pour but final le développement et l'intégration d'une solution de smart-logistique visant à faciliter la gestion des entrepôts et la traçabilité des produits entreposés.

Les bonnes pratiques en logistique, et plus précisément lors de la création d'une solution, exigent un ou plusieurs environnements de travail satisfaisant l'ensemble des exigences de la chaîne décisionnelle. Il est convenu et admis qu'une solution Smart-logistique spécifique à un système préexistant demande encore plus de spécificités, à l'image de l'expertise RITM, nécessitant un appareillage MindStorm simple et efficace, une lecture RFID, une modélisation dimensionnelle légère mais globale, ainsi qu'une panoplie d'interfaces intuitives et épurées.

Il est de plus à noter que l'obtention d'une finalité dotée d'une précision accrue passe par une traçabilité non seulement précise quant à la position mais aussi quant au temps. L'établissement d'une géolocalisation précise pour chaque produit et le fait d'assurer une traçabilité grâce à la technique de block Chain, demeure la tâche la plus délicate et la plus longue, et détermine en soi les retombées finales.

Ce projet intervient principalement pour faciliter toutes les facettes de l'entreposage moderne, ceci après avoir entamé la découverte des processus déjà existants, la faisabilité du projet et ses spécifications, ainsi que la définition du cahier de charges. La suite du projet relève de la lecture RFID assurée par un robot autonome, le transfert de données vers le serveur principal et le traitement en block Chain pour assurer la traçabilité.

Mots-clés : Logistique, RFID, Bonnes pratiques, MindStorm, Block Chain, Traçabilité.

# Abstract

This report summarizes the work done as part of the end of studies project in the RITM lab. The ultimate goal of this project is the development and integration of a smart-logistics solution aimed at facilitating the management of warehouses and the traceability of stored products.

Good logistics practices, and more specifically when creating a solution, require one or more work environments that meet all the requirements of the business chain. It is agreed and admitted that a Smart-Logistics solution specific to a pre-existing system requires even more specificities, like the RITM expertise, requiring a simple and effective MindStorm equipment, an RFID reading, a light but global dimensional modeling, as well as a host of intuitive and uncluttered interfaces.

It should also be noted that obtaining a finality with increased precision requires not only precise traceability, but also precise time. Establishing accurate geolocation for each product and ensuring traceability through the block chain technique, remains the most delicate task and the longest, and determines in itself the final fallout.

This project is mainly used to facilitate all facets of modern storage, after having initiated the discovery of existing processes, the feasibility of the project and its specifications, as well as the definition of the specifications. The rest of the project is RFID reading provided by an autonomous robot, the transfer of data to the main server and blocking processing to ensure traceability.

Keywords: Logistics, RFID, Best practices, MindStorm, Block Chain, Traceability

# Liste des abréviations

2TUP	2 Tracks Unified Process
SW	Smart warehousing
Vcn	Virtual network computing
RFID	Radio frequency identification
RITM	Réseau informatique télécommunication et multimédia
BAP	Battery-Assisted Passive tags

# Table des figures

Figure 1 Table des Sprints.....	16
Figure 2 Cycle de vie en "Y" .....	17
Figure 3 Cycle de vie dimensionnel.....	19
Figure 4Diagramme de GANTT .....	21
Figure 5 Simulation de robots de smart-warehousing .....	30
Figure 6 Bête à cornes .....	33
Figure 7 Diagramme de pieuvre.....	34
Figure 8 Diagramme de F.A.S.T .....	36
Figure 9 Caméra LogiTech .....	51
Figure 10 Caméra Havic .....	51
Figure 11 Diagramme de séquence principal.....	51
Figure 12 Diagramme de séquence mappage.....	51
Figure 13 Diagramme de déploiement.....	51
Figure 14 Principe de communication RFID avec une radio-étiquette passive..	51
Figure 15 Bandes autorisées et possibles de fréquences pour les puces RFID communément commercialisées .....	51
Figure 16 Gammes de fréquences les plus utilisées et quelques applications RFID .....	51
Figure 17 Genuino UNO .....	53
Figure 18 Proteus 8.....	54
Figure 19 Python 3 .....	54
Figure 20 Simulation en Proteus 8 .....	56
Figure 21 Intérface Raspberry.....	57
Figure 22:Paramètre de configuration.....	57
Figure 23:test_script .....	58
Figure 24:memory_leak .....	59
Figure 25 NXT .....	76
Figure 26 JAVA .....	77
Figure 27 PUTTY .....	78
Figure 28 VNC servers.....	79

# Table des matières

<b>Liste des abréviations.....</b>	<b>7</b>
<b>Table des figures.....</b>	<b>8</b>
<b>Table des matières.....</b>	<b>9</b>
<b>Chapitre 1 : Contexte général du projet.....</b>	<b>14</b>
<b>1.1 Présentation de l'organisme d'accueil.....</b>	<b>14</b>
<b>1.2 Description du projet .....</b>	<b>15</b>
<b>1.2.1         Problématique .....</b>	<b>15</b>
<b>1.2.2         Objectifs du projet.....</b>	<b>15</b>
<b>1.2.3         Étalage des sprints de développement .....</b>	<b>16</b>
<b>1.2.4         Conduite du projet.....</b>	<b>17</b>
<b>Conclusion.....</b>	<b>26</b>
<b>Chapitre 2 : Analyse et spécification des besoins .....</b>	<b>28</b>
<b>2.1             Étude du marché et contextualisation du smart-warehousing.....</b>	<b>28</b>
<b>2.2             Qu'est-ce qui compose un entrepôt intelligent?.....</b>	<b>30</b>
<b>2.3             Etude de faisabilité.....</b>	<b>33</b>
<b>2.3.1         Bête à corne .....</b>	<b>33</b>
<b>2.3.2         Diagramme de pieuvre .....</b>	<b>34</b>
<b>2.3.3         Diagramme de F.A.S.T.....</b>	<b>35</b>
<b>Conclusion.....</b>	<b>37</b>
<b>Chapitre 3 : Conception et modélisation de la solution .....</b>	<b>51</b>
<b>3.1             Partie lecture RFID.....</b>	<b>51</b>
<b>3.1.1         Solution idéale épurée.....</b>	<b>51</b>
<b>3.1.2         Solution Prototypique.....</b>	<b>51</b>
<b>3.2             Partie lecture Code-Barres.....</b>	<b>51</b>
<b>3.2.1         Solution Idéale .....</b>	<b>51</b>
<b>3.2.2         Solution prototypique .....</b>	<b>51</b>

<b>3.3</b>	<b>Partie Robotique .....</b>	51
<b>3.4</b>	<b>Diagrammes de contexte.....</b>	51
<b>3.4.1</b>	<b>Diagramme de séquence principal .....</b>	51
<b>3.4.2</b>	<b>Diagramme de séquence mappage .....</b>	51
<b>3.4.3</b>	<b>Diagramme de déploiement .....</b>	51
<b>Conclusion.....</b>		51
<b>Chapitre 4 : Réalisation de la phase de lecture .....</b>		<b>51</b>
<b>4.1</b>	<b>Description de la technologie utilisée RFID.....</b>	51
<b>4.2</b>	<b>Outils utilisés .....</b>	53
<b>4.2.1</b>	<b>Genuino UNO .....</b>	53
<b>4.2.2</b>	<b>Proteus 8 .....</b>	54
<b>4.2.3</b>	<b>Python 3 .....</b>	54
<b>4.3</b>	<b>Codes et programmation effectuée .....</b>	55
<b>4.3.1</b>	<b>Solution RFID idéale.....</b>	55
<b>4.3.2</b>	<b>Solution RFID prototypique .....</b>	57
<b>Conclusion.....</b>		71
<b>Chapitre 5 : Réalisation de la partie robotique .....</b>		<b>73</b>
<b>5.1</b>	<b>Présentation de la carte NXT (MindStorm) .....</b>	73
<b>5.2</b>	<b>Logiciels utilisés.....</b>	76
<b>5.2.1</b>	<b>MindStorm NXT .....</b>	76
<b>5.2.2</b>	<b>JAVA.....</b>	77
<b>5.2.3</b>	<b>Python 3 .....</b>	78
<b>5.2.4</b>	<b>Putty .....</b>	78
<b>5.2.5</b>	<b>VcnServer /VcnViewer .....</b>	79
<b>5.3</b>	<b>Programmation des différents axes d'utilisation du robot.....</b>	80
<b>5.3.1</b>	<b>Programmation NXT .....</b>	80
<b>5.3.2</b>	<b>Programmation de la carte Raspberry Pi III.....</b>	82
<b>5.3.3</b>	<b>Programmation du mappage .....</b>	87
<b>Conclusion.....</b>		90
<b>Conclusion générale .....</b>		<b>91</b>
<b>Webographie.....</b>		<b>92</b>

## Introduction générale

Toutes les entreprises de productions s'appuient sur le principe du stockage physique, évidemment avec l'objectif d'assurer un répondant fort à la demande toujours aussi grandes de nos sociétés de surconsommation et ainsi générer un chiffre d'affaires et des bénéfices des plus élevés possibles. Le concept du smart-warehousing est une réponse aux limites du modèle d'entreposage traditionnel, beaucoup trop chaotique et ne permettant pas une traçabilité et un suivi sur le long terme. Ce concept vise à suivre et à faire évoluer la relation avec le client dans le temps, l'objectif de cette relation étant principalement de fidéliser les clients existants grâce aux différentes prouesses atteintes en termes d'optimisation temporelle et de cultiver un vivier de prospects.

Dans cette optique, le labo RITM spécialisé dans les solutions informatiques a décidé de s'investir dans le développement d'une solution de smart-warehousing en faisant appel à deux équipes s'occupant chacune d'une des facettes de la plate-forme. Cette plate-forme a pour objectif de faciliter, auprès des entrepôts, des opérations auparavant très couteuses en terme de ressources tel que l'inventaire le suivi des produits par exemple.

Dans sa globalité, le smart-warehousing devient un standard dans les entreprises de production, garantissant une solution uniformisée pour tous les services de gestion de produits ainsi que des données, ce qui permet dans un premier temps une traçabilité précise et une meilleure cohérence des données. Il est donc prévu que l'aspect logistique renforce la stratégie de suivi et d'accompagnement des solutions proposées par nos équipes.

Identifier le besoin, maîtriser les risques associés et proposer des alternatives aux solutions préalablement utilisées, sortir avec une solution de lecture RFID rapide et efficace et augmenter la mobilité grâce à des robots autonomes, ce sont là les enjeux majeurs dont traitera notre équipe.

En somme, la solution que nous présentons est primordial dans la mesure où la concurrence s'intensifie entre les différents acteurs, il est bon de rappeler que seuls ceux qui sont capables de fournir des services novateurs, grâce à leur incessante quête d'optimisation et d'automatisation,

peuvent espérer conserver longtemps leur position dominante. En d'autres termes, ne pas investir dans une solution optimisée telle que la nôtre, c'est accepter de perdre des parts de marché, et pour regagner le terrain perdu face à la concurrence, c'est accepter de payer plus cher la barrière à l'entrée pour les nouvelles innovations.

Ce stage de fin d'étude a donc pour objectif de mettre en œuvre une solution mettant encore plus l'accent sur le suivi à long terme et la traçabilité des produits entreposés grâce à notre procédé de smart-warehousing, de la conception à la réalisation de tout l'appareillage de la chaîne logistique.

Le présent mémoire expose les résultats du travail durant le stage. Il comporte cinq chapitres organisés suivant le processus de développement choisi :

- L'objet du premier chapitre est de cerner le projet dans son contexte général, à travers la présentation de l'organisme d'accueil, pour décrire ensuite le marché et la motivation du projet ainsi que les objectifs visés. La dernière section du chapitre sera consacrée à la conduite et planification du projet.
- Le deuxième chapitre englobe l'analyse et la spécification des besoins, à savoir l'étude fonctionnelle, dont l'objectif est de capturer les besoins fonctionnels et opérationnels du système futur, aussi bien que l'étude technique, qui englobe l'architecture physique et logicielle ainsi que les outils les plus appropriés pour ce type d'application.
- Au niveau du troisième chapitre sera explicitée la conception de la solution. Nous utilisons les différents diagrammes pour modéliser les fonctionnalités de la lecture RFID ainsi que celles du robot autonome.
- Le quatrième chapitre décrit d'une façon détaillée, la phase de mise en œuvre de la solution de lecture RFID
- Le cinquième et dernier chapitre plonge dans la mise en œuvre de la solution robotique de smart-warehousing.

Le mémoire se termine par une conclusion générale qui présente le bilan du travail réalisé et ses principales perceptives.

# **Chapitre 1**

---

## **Contexte général du projet**

La présentation du cadre général du projet a pour but de situer le projet dans son environnement organisationnel et contextuel. Ce chapitre introduit donc l'organisme d'accueil, décrit la problématique à traiter et présente la démarche suivie pour la réalisation du projet.

# 1. Contexte général du projet

## 1.1 Présentation de l'organisme d'accueil

L'école supérieure de technologie de Casablanca compte parmi les grands établissements publics d'enseignement supérieur à finalité professionnalisant, elle a été créée en 1986 par le ministre de l'enseignement supérieur de la formation des cadres et de la recherche scientifique.

L'école supérieure de technologie de Casablanca et une composante de l'Université Hassan sur sa vocation et de former des techniciens supérieurs polyvalent hautement qualifié et immédiatement opérationnels après leur sortie de l'école en tant que collaborateurs d'ingénieurs et de managers.

L'école supérieure de technologie Casablanca dispose de plusieurs départements qui pilotent et gèrent les formations de leurs étudiants, chaque département se charge de plusieurs disciplines.

Ce stage a été réalisé au sein du laboratoire de recherche RITM (réseau informatique télécommunication et multimédia) dont font partie les équipes de recherche du département génie informatique.

Ce Laboratoire existe depuis 2007 rattaché à l'École supérieure de technologie Casablanca dirigé par Monsieur Mounir RIFI professeurs de l'enseignement supérieur à l'ESTC même et qui travaille avec l'équipe de recherche réseau et télécommunication (R&T).

Le laboratoire est intégré dans le CED (Centre d'Etudes Doctoral) science de l'ingénieur à l'ENSEM. Il comporte plus d'une trentaine de membres composés des enseignants chercheurs et des doctorants rattachés à quatre équipes de recherche :

- ETIC : Technologie de l'Information et de la Communication
- R&T : Réseaux et Télécommunications
- I&R : Informatique et Réseaux
- SBIS : Software engineering Business Intelligence and Security

Parmi les membres du conseil exécutif on compte trois professeurs de l'enseignement supérieur : M. RIFI Mounir, M. ABCHIR Hamid et M. OUZZIF Mohamed.

## 1.2 Description du projet

### 1.2.1 Problématique

La gestion du stock d'un entrepôt n'est pas chose aisée. Avec les techniques traditionnelles cela peut prendre jusqu'à plusieurs semaines pour parcourir manuellement des surfaces extrêmement grandes jonchées produit attendant une distribution vers d'autres lieux de stockage tout autant grands. Et malgré les nombreuses avancés en terme de logistique il reste encore beaucoup à réaliser pour palier à de nombreux problèmes tels que :

- L'utilisation de techniques extrêmement lentes le plus souvent de façon manuelle ce qui est susceptibles d'induire beaucoup d'erreurs.
- Le temps précieux perdu à parcourir les embouchures des lieux d'entreposage afin de réaliser des comptes rendus ou un inventaire complet.
- L'égarement de données en cours de route durant les transactions entre les entrepôts ce qui crée dans la plupart du temps des soucis d'intégrité.

### 1.2.2 Objectifs du projet

Les objectifs du projet, tels qu'ils ont été établis lors des premières phases, ont été fixés autour de l'objectif principal qui est la mise en place d'une solution de smart-warehousing en utilisant le procédé de lecture RFID ainsi qu'un robot autonome.

Ce processus consiste à munir un robot d'une self-awareness géographique le rendant capable de s'adapter aux environnements d'entreposage grâce à une technique de mappage d'où l'aspect autonome, ensuite il est question de le programmer pour qu'il puisse lire des tags RFID qui est l'innovation apporté à l'ancien procédé du code à barres. Procédé qui ne sera pas pour autant abandonné puisque le robot devra lire les codes à barres aussi grâce au zoom caméra. Il faudra finalement lier toutes ces technologies pour pouvoir donner un géolocalisation précise et propre à chaque produit. Les données seront ensuite envoyé vers une autre équipe chargée de répertorier le tout dans une base données flexible usant de la technologie de block Chain pour assurer la traçabilité de chaque produit.

Pour plus de visibilité sur les enjeux du projet, nous les résumons par ordre décroissant de priorité dans les points suivants :

- Permettre une gestion de stock plus rapide
- Assurer la traçabilité des produits
- Diminuer le nombre de litiges et d'erreur, et garder une trace de tout disfonctionnement.
- Assurer une intégrité sans faille

### 1.2.3 Étalage des sprints de développement

Les résultats attendus de notre équipe se répartissent sur trois sprints de développement, chacun d'eux répondant à un besoin de l'entreprise, en sachant que leur charge est directement proportionnelle à l'investissement en ressources humaines et logicielles dans le but d'accomplir chaque sprint.

	Description	Tâches	Charge
1 <sup>er</sup> Sprint : Robot	Phase consistant en l'assamblage du robot et la programmation des différentes fonctionnalités dont on aura besoin pour mener à bien la réussite des challenges	Assamblage du robot. Programmation de la reconnaissance video. Programmation des mouvements. Mappage. Tests.	50%
2 <sup>ème</sup> Sprint : Lecture	Phase de solutionnement du problème de lecture (RFID et Code barre)	Choix du matériel. Intégration des programmes. Programmation de la caméra. Tests.	40%
3 <sup>ème</sup> Sprint : Lien	Lien entre les deux sprints précédents.	Intégration du programme RFID. Intégration du programme caméra.	10%

Figure 1 Table des Sprints

## 1.2.4 Conduite du projet

### 1.2.4.1 Cycle de vie en « Y »

La gestion de projet est un facteur déterminant dans la réussite d'un projet, du fait qu'elle orchestre ses différentes phases et trace les principaux traits de sa conduite. Pour cela, le choix d'une méthode de développement, qui soit adéquate aux particularités et exigences d'un projet, doit être élaboré au préalable afin d'obtenir un produit de qualité qui répond aux besoins et aux attentes des utilisateurs tout en respectant les délais et les exigences fixés.

Le processus de développement que nous avons adopté, afin de mener dans les meilleures conditions le projet, émane du processus 2TUP « 2 Track Unified Process » qui suit un cycle de vie en Y.

Il s'agit d'un processus itératif, incrémental et centré sur l'architecture appartenant à la famille des processus unifiés qui constitue une trame commune pour intégrer les meilleures pratiques de développement. Le processus 2TUP insiste sur la non-corrélation initiale des aspects fonctionnel et technique jusqu'aux phases avancées. Les deux branches d'étude fusionnent ensuite pour la conception du système, ce qui donne la forme d'un processus de développement en Y. La dichotomie initiale permet à la fois de capitaliser la connaissance métier sur la branche gauche et de réutiliser un savoir-faire technique sur la branche droite.

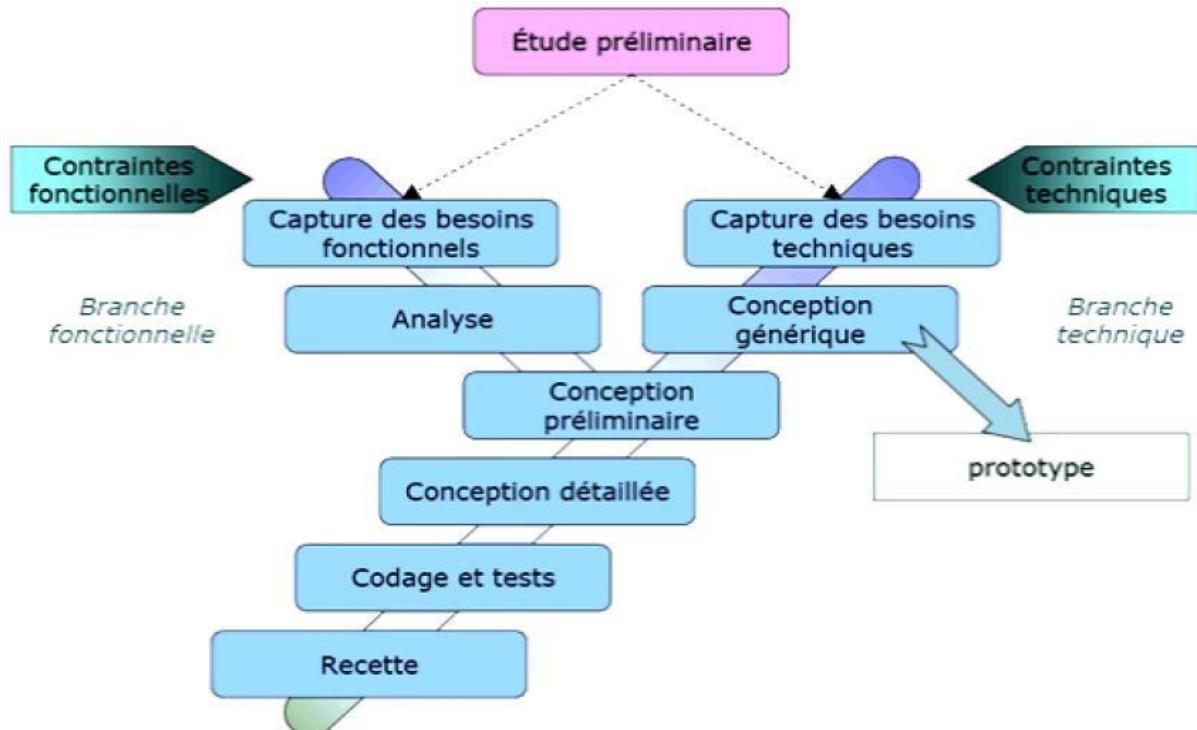


Figure 2 Cycle de vie en "Y"

Ce cycle s'articule selon 3 branches :

- **Branche fonctionnelle** comportant les deux phases suivantes :
  - La capture des besoins fonctionnels de la partie Robot et la partie Lecture.
  - L'analyse, qui consiste à étudier précisément la spécification fonctionnelle de manière à obtenir un lien entre les deux parties.
- **Branche architecture technique** qui quant à elle comporte les deux phases suivantes :
  - La capture des besoins techniques, qui recense toutes les contraintes sur les choix techniques du système. Les outils et le matériel sélectionné ainsi que la prise en compte des contraintes d'intégration avec l'existant (prérequis d'architecture technique).
  - La conception générique, qui définit ensuite les composants nécessaires à la construction de l'architecture technique. Cette conception est complètement indépendante des aspects fonctionnels. Elle a dans notre cas pour objectif d'uniformiser et de réutiliser les mêmes mécanismes pour tout système.
  - L'architecture technique construit le squelette du système. Lors de ce projet, son importance est telle qu'il est conseillé de réaliser un prototype.
- **Branche conception** dont les phases sont les suivantes :
  - La conception préliminaire, qui représente une étape délicate, car elle intègre le modèle d'analyse fonctionnelle dans l'architecture technique de manière à tracer la cartographie des composants du système à développer.
  - La conception détaillée, qui étudie ensuite comment réaliser chaque composant.
  - L'étape de codage et de construction, qui produit les composants et teste au fur et à mesure les unités réalisées.
  - L'étape de recette, qui consiste enfin à valider les fonctionnalités du système développé.

#### 1.2.4.2 Cycle de vie dimensionnel

Ce projet étant à caractère décisionnel, il a été nécessaire de suivre en plus, pour la démarche de sa réalisation, le cycle de vie dimensionnel. Ce cycle commence par la

planification du projet puis passe à la définition des besoins qui constituent le point de départ de trois trajectoires parallèles qui sont la technologie, les données et l'interface utilisateur. Ces trajectoires se rejoignent en phase de déploiement qui est suivie par la phase de maintenance et croissance.

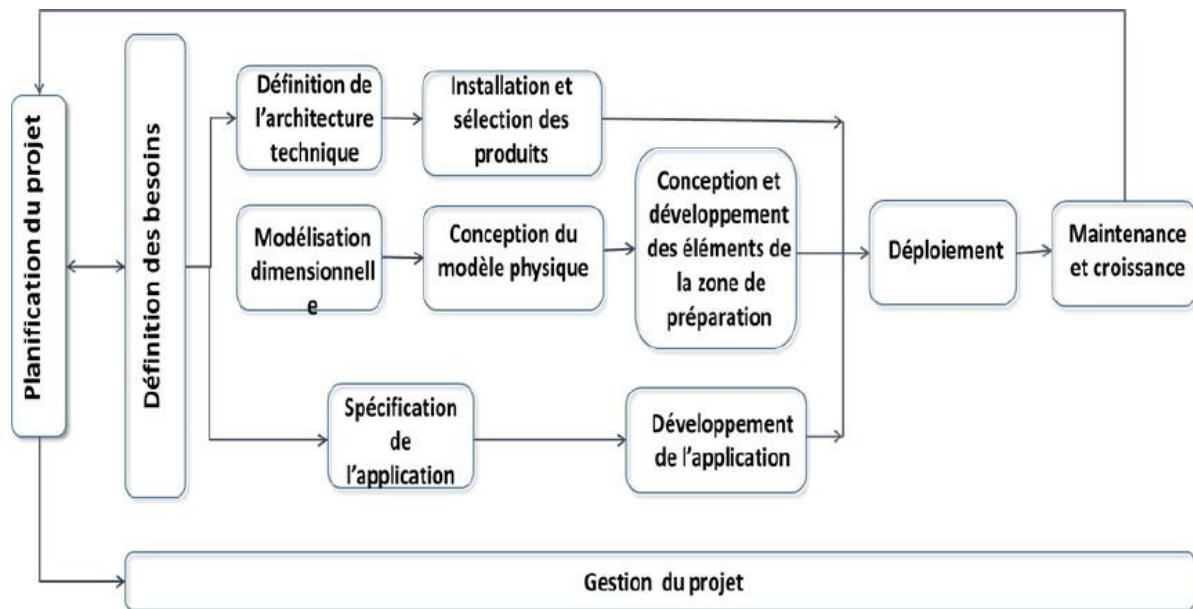


Figure 3 Cycle de vie dimensionnel

Les étapes du cycle de vie illustré dans la figure précédente sont :

- **La planification du projet :** Elle aborde la définition et l'étendue du projet. Elle permet l'affectation des tâches à leurs durées et à leur séquencement.
- **La définition des besoins :** Elle permet d'identifier les besoins et détermine les données requises pour répondre à ces derniers.
- **La modélisation dimensionnelle :** Elle commence par la construction d'une matrice qui représente les processus métiers clé et leurs dimensionnalités. A partir de cette matrice, on élabore une analyse plus détaillée des données des systèmes sources, puis on développe le modèle dimensionnel.
- **La conception du modèle physique :** Elle définit les structures nécessaires pour l'implémentation du modèle dimensionnel. Cette étape nécessite la détermination des règles de nommage des objets et la mise en place de l'environnement de la base de données.
- **La conception et le développement de la zone de préparation des données :** Elle se déroule en trois phases majeures : Extraction, Transformation et Chargement.

- **La définition de l'architecture technique :** Elle permet d'avoir une vision globale de l'architecture technique à mettre en œuvre. Dans cette phase, trois facteurs doivent être pris en compte : les besoins, l'environnement existant et les orientations techniques stratégiques planifiées.
- **La sélection des produits et l'installation :** Elle permet la sélection des composants spécifiques tels que la plateforme matérielle, le SGBD (Système de Gestion de Bases de Données), et les outils de préparation et d'accès aux données. Une fois évalués et sélectionnés, ceux-ci devront être installés et testés minutieusement.
- **Les spécifications de l'application utilisateur :** Cette étape décrit les maquettes d'états, les critères de sélection laissés à l'utilisateur et les calculs nécessaires.
- **Le développement de l'application utilisateur :** Il commence par une compréhension commune par l'équipe de développement et les utilisateurs finaux des spécifications établies.
- **Le déploiement :** Il est le point de convergence de la technologie, des données et des applications utilisateur. Une formation des utilisateurs est nécessaire. Elle intègre tous les aspects de cette convergence avant de permettre à ceux-ci d'accéder à l'application. La prise en compte des demandes d'évolution et de correction est également indispensable.
- **La maintenance et croissance :** Elle permet de s'occuper des utilisateurs en leur procurant un service de support et une formation continue.
- **La gestion du projet :** Elle garantit que les activités du cycle de vie dimensionnel restent sur la bonne voie et sont bien synchronisées.

### 1.2.4.3 Planning prévisionnel

La planification est l'une des phases d'avant-projet. Elle consiste à prévoir le déroulement du projet tout au long des phases constituant son cycle de développement. Grâce aux réunions tenues avec les encadrants, au sein du labo RITM, les différentes étapes du projet ainsi que leur déroulement ont été clarifiées et plusieurs détails fonctionnels pour chaque étape ont été émis.

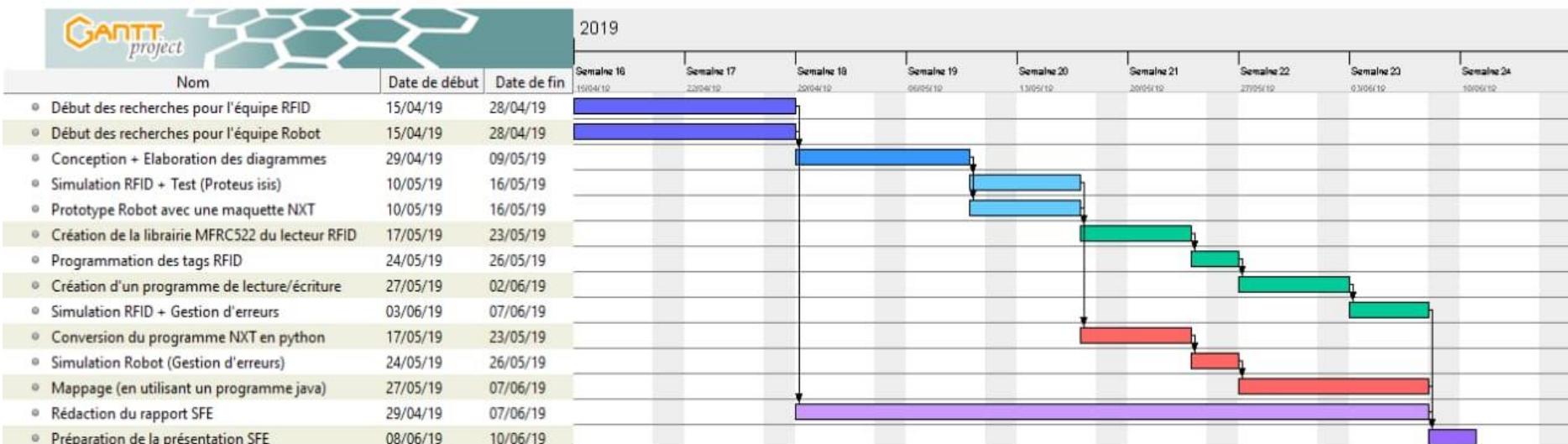


Figure 4Diagramme de GANTT

## Conclusion

Ce chapitre avait pour objet la présentation du cadre général de notre projet de fin d'études qui consiste en l'intégration d'une solution de smart-warehousing. Ainsi, nous avons présenté le contexte du projet, l'étude du marché cible et la problématique à laquelle il vient répondre ainsi que ses objectifs. Il permet de mettre le projet dans un cadre d'étude avant de pouvoir l'analyser et spécifier les besoins fonctionnels et techniques en jeu.

La problématique et les objectifs du projet relevés jusqu'ici sont donc la cible du prochain chapitre, celui de l'analyse transversale du cahier de charges et de la spécification des besoins fonctionnels et techniques.

# Chapitre 2

---

## **Analyse des besoins et étude de faisabilité.**

Dans ce chapitre seront exposés les résultats de la phase d'étude, d'analyse et de cadrage du projet. La spécification des besoins se fera à l'aide de la description des différentes sources de données, des indicateurs retenus et leurs axes d'analyse. Quant à l'étude de faisabilité elle sera représentée grâce à différents diagrammes.

## 2 Analyse et spécification des besoins

L'analyse d'un tel projet passe d'abord par une étude sectorielle visant à contextualiser son impact. Une étude de marché est donc nécessaire afin de comprendre l'envergure de l'activité ciblée et avoir une idée sur la forme et la taille des sources de données à exploiter.

### 2.1 Étude du marché et contextualisation du smart-warehousing

Avec la croissance inouïe que les entreprises de production ont enregistré durant la dernière décennie, ces dernières se sont vues très vite face à un besoin urgent de techniques de stockages plus évolués et optimisées. C'est ici qu'est intervenu le smart-warehousing.

Les systèmes d'entrepôts intelligents sont donc le résultat de la collaboration entre diverses technologies d'entreposage interconnectées. Ils forment un écosystème technologique où les marchandises sont reçues, identifiées, triées, organisées et extraites pour expédition automatique. Les meilleures solutions d'entrepôt intelligent automatisent la quasi-totalité des opérations, des fournisseurs aux clients, en minimisant les erreurs.

Ces systèmes de stockage pour être performants et éligibles aux demandes du marché doivent se soumettre à trois principaux axes :

#### Agilité

Ce n'est un secret pour personne que les opérations d'entreposage deviennent de plus en plus complexes.

Pour répondre à ces besoins complexes, les entrepôts intelligents doivent être en mesure de passer à la vitesse supérieure et de prendre en compte les variables qui changent rapidement dans le monde actuel.

Votre système d'entrepôt intelligent doit constamment essayer d'optimiser le transport des produits, du stockage à l'expédition.

De plus, les opérations d'entreposage intelligentes devraient également être à la recherche d'opportunités de stockage temporaire et de fournisseurs de services logistiques tiers (3PL) en cas de dépassement temporaire des stocks.

L'agilité intervient également en ce qui concerne les divers logiciels utilisés dans votre entrepôt intelligent. Tous vos systèmes doivent pouvoir fonctionner avec de nombreuses plates-formes, systèmes de vente et autres fonctionnalités d'entreposage.

## L'évolutivité

Alors que la complexité des chaînes d'entreposage et d'approvisionnement ne cesse de croître, votre entrepôt intelligent doit toujours garder un œil sur l'avenir. Les entrepôts intelligents doivent être prêts à accepter un afflux important de produits et de nouvelles versions de produits à tout moment.

Cela nous ramène à une autre qualité attrayante des solutions de smart-warehousing. L'ajout de nouvelles fonctionnalités à vos systèmes d'entreposage intelligents peut s'avérer difficile si vous disposez d'une solution sur site. Dans certains cas, il peut être nécessaire d'arrêter l'ensemble du système pendant que les mises à jour sont terminées, ce qui peut entraîner un ralentissement important de l'entrepôt.

Une solution de smart-warehousing ne lutte pas avec cette limitation. Les mises à jour peuvent être effectuées à la volée et les nouvelles inclusions peuvent être déployées sans compromettre l'efficacité. Implémenter une nouvelle fonctionnalité uniquement pour obliger le service informatique à résoudre des problèmes imprévus est un casse-tête qu'aucun exploitant d'entrepôt intelligent ne souhaite gérer.

## Visibilité des données

À mesure que la technologie progresse, la visibilité des données est devenue un impératif pour la plupart des logiciels liés aux fonctions de la chaîne logistique. Cela est très important aujourd'hui, alors que les chaînes d'approvisionnement et les opérations d'entreposage deviennent de plus en plus complexes, les solutions logicielles doivent être prêtes à mettre à jour et à stocker les données immédiatement. Les clients et les parties prenantes souhaitent pouvoir accéder aux données en temps réel fournies par votre système d'entreposage intelligent et voir où en est leur produit.

Votre entrepôt intelligent devrait permettre aux parties intéressées de trouver immédiatement ce qu'elles veulent. Si un client a le choix entre s'associer à une opération qui met à jour ses données du jour au lendemain ou à une opération qui reste constamment à jour, vous pouvez parier sur celle qu'il choisira.

Cet aspect est non seulement important pour la conservation des clients, mais la visibilité des données fournit également une foule d'informations utiles à l'opération elle-même. Les données en temps réel donnent aux opérateurs une vue à vol d'oiseau de tout l'entrepôt et de la performance de ses différentes pièces. Si des inexactitudes dans les stocks ou les délais de livraison se présentent, vous saurez immédiatement d'où elles proviennent. De cette façon, les exploitants d'entrepôts intelligents peuvent agir avant que les choses ne deviennent incontrôlables et ne jettent une plus grande clé dans les travaux.

## 2.2 Qu'est-ce qui compose un entrepôt intelligent?

À la base, un entrepôt intelligent se compose d'une variété de technologies interconnectées qui visent toutes les mêmes objectifs. Chaque pièce de ce casse-tête a un travail à faire qui permet à votre entrepôt de fonctionner de manière optimale. Voici quelques exemples des nombreux composants que vous trouverez dans un entrepôt intelligent :

### Robotique

Les robots humanoïdes qui habitent nos émissions de science-fiction et nos histoires sont encore loin, mais il existe d'autres types de robots presque aussi cool. Les robots d'entreposage que vous verriez aujourd'hui s'occupent principalement de la préparation et de l'emballage des marchandises.



Figure 5 Simulation de robots de smart-warehousing

Ressemblant généralement à Roombas, vos robots de stockage habituels automatisent le processus de prélèvement en apportant physiquement les rayons des produits aux employés chargés des commandes.

Essentiellement, ils sont automatisés et plus maniables. Ils bougent plus rapidement que les gens et peuvent même identifier le meilleur itinéraire pour récupérer les produits nécessaires.

## Identification radiofréquence

L'identification par radiofréquence (RFID) aide à organiser et à contrôler l'inventaire. La RFID supprime les anciennes méthodes de suivi du papier analogique au profit du suivi des colis contenant des étiquettes numériques. Les ondes radio sont ensuite utilisées pour transférer des données vers ou entre l'étiquette numérique et un système de balayage automatisé, enregistrant les informations du produit.

La RFID remplace les anciens scanners de codes à barres, où le code à barres doit être aligné avec précision sur le scanner pour l'identifier. Au lieu de cela, les scanners RFID peuvent simplement être orientés dans la direction générale du colis afin de l'identifier.

Comment cela aide-t-il avec la gestion des stocks? Tout d'abord, étant donné que les scanners n'ont pas besoin d'être alignés avec précision, des machines automatisées peuvent être utilisées pour numériser les colis à l'arrivée, en identifiant et en comptant le nombre de biens de chaque type reçus. En outre, ces scanners peuvent détecter les marchandises au fur et à mesure qu'elles quittent l'entrepôt lors de l'exécution de la commande, garantissant ainsi la précision de votre inventaire.

## Intelligence artificielle

L'utilisation de l'intelligence artificielle (IA) explose dans tous les secteurs, et pas seulement dans le stockage. La raison principale? L'intelligence artificielle contribue à augmenter la productivité tout en minimisant les erreurs.

Par exemple, AI aide les robots d'entreposage à trouver la voie la plus efficace pour choisir leurs produits. Il peut également être utilisé pour déterminer le meilleur type de boîte pour une expédition en fonction du type, du nombre, de la taille et du poids des produits. Certains entrepôts ont même été en mesure d'implémenter des machines capables d'emballer des produits, en utilisant l'IA pour les emballer de la manière la moins encombrante possible.

De telles capacités aident les opérations d'entrepôt à réduire considérablement leurs coûts d'exploitation. Le nombre de travailleurs humains est l'un des coûts les plus importants. Selon Cerasis, 30% des emplois au Royaume-Uni seront automatisés d'ici 2030, en grande partie grâce à l'utilisation de l'IA.

## Internet des objets

Vous avez probablement déjà entendu parler de l'Internet des objets ou de l'IdO. Si vous voulez que votre entrepôt intelligent fonctionne correctement, vous comptez sur l'IdO. Si vous avez besoin d'un rafraîchissement, l'IoT implique la connexion de plusieurs appareils compatibles avec Internet et le partage de données. Dans les systèmes d'entrepôt intelligents, cela signifie que les robots peuvent communiquer avec toutes les technologies dont il a besoin, y compris un système de gestion d'entrepôt (WMS).

Un exemple d'IoT au travail dans un système de gestion d'entrepôt intelligent commence à partir d'un entrepôt recevant un produit. Lors de la réception de l'envoi, un scanner RFID scanne les étiquettes et indique à un WMS quelles marchandises et combien de marchandises ont été reçues. Le système WMS communique ensuite avec les robots, les informant du lieu où ces marchandises doivent être stockées dans l'entrepôt.

Tout cela se fait automatiquement et de manière transparente, sans perdre aucune information cruciale en cours de route. Sans l'IdO, un travailleur humain devrait compléter manuellement chaque étape du processus. Cela est sujet aux erreurs, en particulier avec la masse d'informations sur chaque produit passant par ces systèmes. Mais grâce à l'IoT, les humains sont quasiment éliminés de l'équation, ce qui accélère le processus tout en réduisant considérablement les erreurs.

À l'avenir, certains commencent même à se préparer à un «Internet de tout».

## Systèmes de gestion d'entrepôt

L'utilisation d'un WMS est la cerise sur le gâteau de votre technologie intelligente. Une solution WMS a de nombreuses utilisations, allant de la collecte de données précieuses à l'aide des utilisateurs à gérer les processus d'entreposage. Cela vous permet de suivre l'efficacité des opérations quotidiennes de votre entrepôt et de déterminer si vous pouvez améliorer quelque chose de spécifique.

### Optimisation WMS

Un WMS ouvre un tout nouveau niveau d'optimisation des entrepôts.

Étant donné que la plupart des solutions WMS peuvent collecter des données en temps réel et créer des rapports visuels, elles permettent de révéler les lacunes de vos processus. Après avoir consulté un rapport WMS, vous pouvez prendre les mesures appropriées pour résoudre les problèmes et redresser vos opérations.

## 2.3 Etude de faisabilité

### 2.3.1 Bête à corne

La **bête à corne** est un outil d'**analyse fonctionnelle du besoin**. En matière d'innovation, il est tout d'abord nécessaire de formuler le besoin sous forme de fonctions simples (dans le sens de « fonctions de bases ») que devra remplir le produit ou le service innovant.

Dès le lancement d'un projet d'innovation, il est nécessaire d'expliciter simplement le besoin primaire, c'est-à-dire l'exigence principale. Son but doit être de satisfaire un besoin exprimé ou non par l'utilisateur.

L'usage d'un nouveau produit ou service doit générer des fonctions de services que la bête à cornes permet d'identifier et de caractériser.

Pour établir la bête à cornes d'un produit, il est nécessaire de se poser les questions suivantes :

- « **A qui mon produit rend-il service ?** » : C'est la cible-utilisateur du futur produit.
- « **Sur quoi agit mon produit ?** » : C'est la matière d'œuvre que va transformer mon produit ou sur laquelle mon produit va agir.
- « **Quel est le but de mon produit ?** » : C'est la fonction principale de mon produit, son intérêt. A quoi sert l'innovation ?

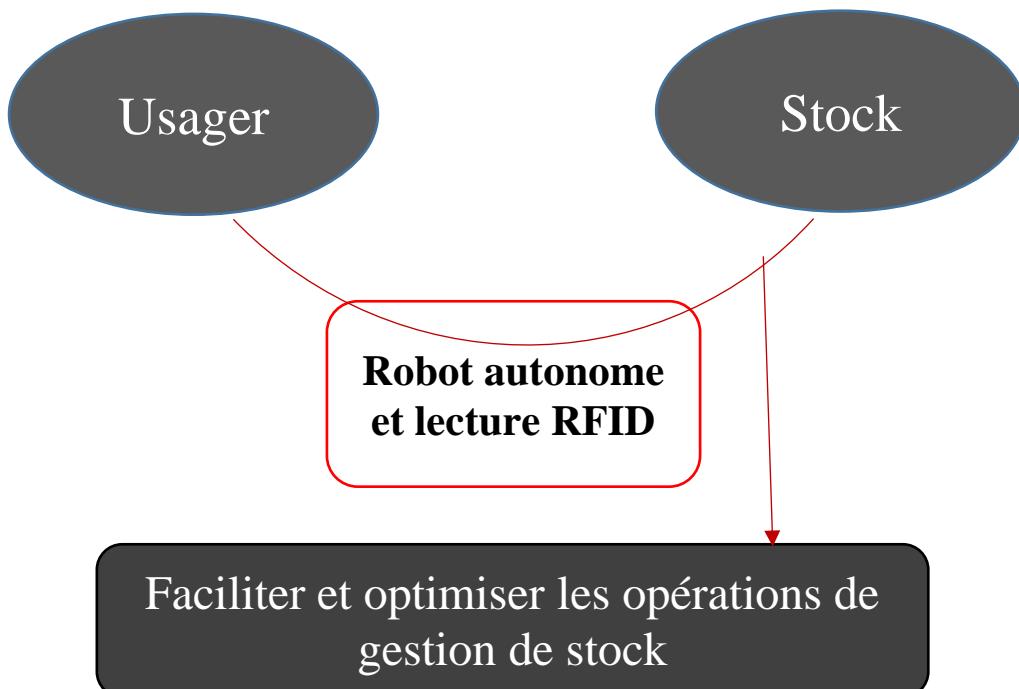


Figure 6 Bête à cornes

### 2.3.2 Diagramme de pieuvre

L'outil "diagramme pieuvre" est utilisé pour analyser les besoins et identifier les fonctions de service de produit.

En analysant le produit, on peut en déduire le diagramme "pieuvre", graphique circulaire qui met en évidence les relations entre les différents éléments de l'environnement du produit. Ces différentes relations sont appelées les fonctions de services qui conduisent à la satisfaction du besoin.

Parmis les fonctions retenues, il y a les fonctions principales, qui sont notées FP, qui représentent l'action d'un élément du milieu extérieur (EME) sur un autre EME, par l'intermédiaire du système. Ensuite, il y a les fonctions contraintes, qui sont notées FC, et qui représentent l'action d'un EME sur le système ou réciproquement.

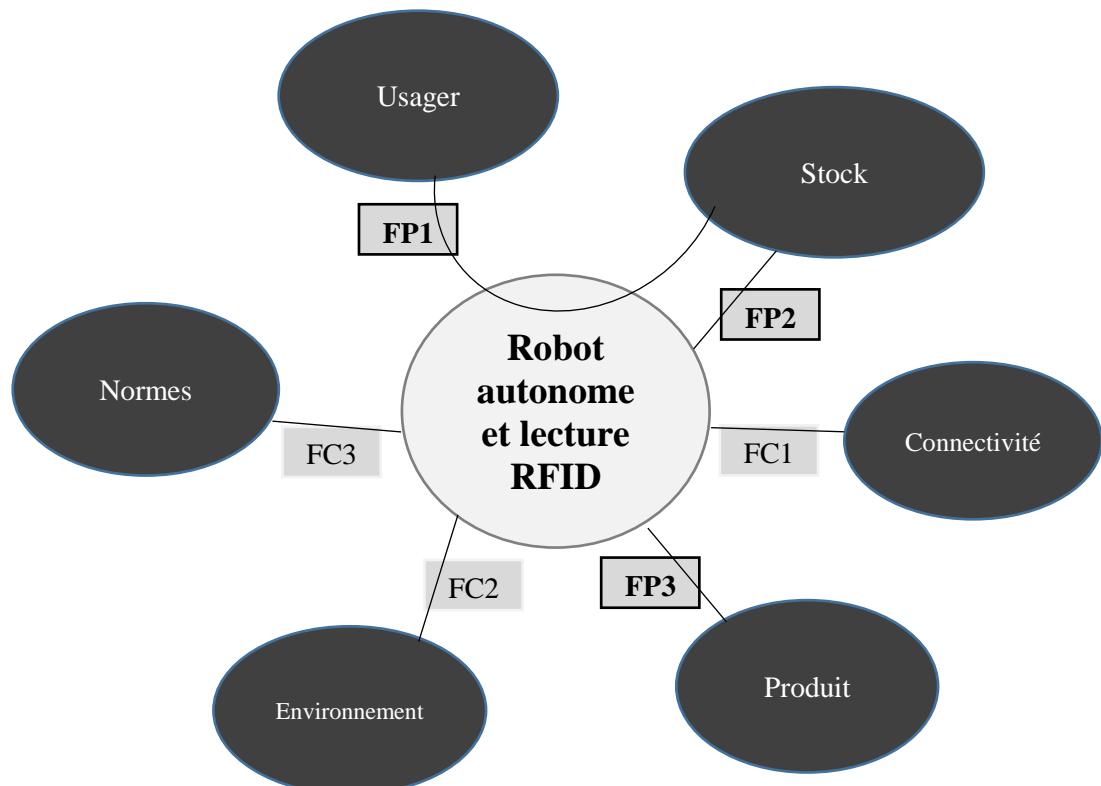


Figure 7 Diagramme de pieuvre

<b>Fonctions de service principales</b>	
<b>FP1</b>	Permet à l'usager de gérer son stock
<b>FP2</b>	Permet une lecture des tags appliqués sur le stock
<b>FP3</b>	Permet de géo localiser chaque produit

<b>Fonctions de service secondaires</b>	
<b>FC1</b>	Assurer une connectivité sans-fil
<b>FC2</b>	Assurer une symbiose avec l'environnement d'entreposage
<b>FC3</b>	Assurer un respect des normes

### 2.3.3 Diagramme de F.A.S.T

Le **FAST** est un outil qui permet de présenter les *fonctions techniques* et les *solutions techniques* retenues pour réaliser les différentes fonctions de service du produit.

Une fonction technique, à la différence d'une fonction de service, n'a pas de *finalité* : elle représente une tâche ou une opération nécessaire pour accomplir quelque chose de plus important pour l'utilisateur.

*Par exemple, sur une machine à laver, mettre en rotation la cuve permet de brasser le linge qui permet de laver le linge.*

Une solution technique est un organe utilisée et existante pour réaliser une fonction technique.

*Par exemple, pour mettre en rotation la cuve il est possible de retenir comme solution technique un moteur.*

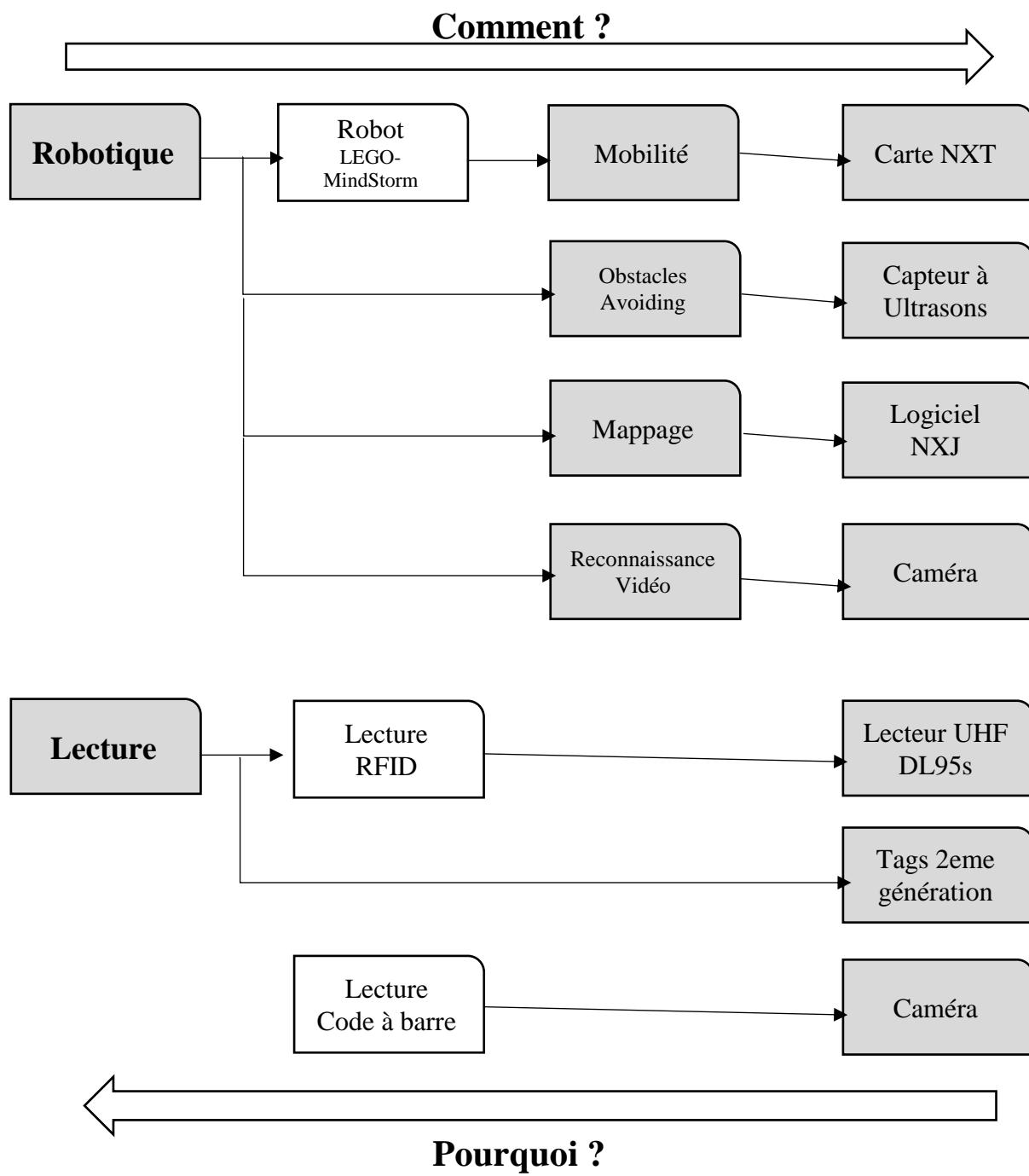
Le FAST peut être lu dans plusieurs sens...

- **Gauche vers la droite, COMMENT ?**  
*Décomposition du système*

En lisant le FAST de la gauche vers la droite il est possible de mettre en évidence la décomposition du système : répondre à la question "Comment a-t-on réalisé cette fonction ?"

- **Droite vers la gauche, POURQUOI ?**  
*Analyse des choix techniques*

En lisant le FAST de droite vers la gauche il est possible de mettre en évidence les choix retenus pour réaliser les différentes fonctions : répondre à la question "Pourquoi a-t-on cet élément ou cette fonction ?"



## Conclusion

Dans ce chapitre, nous avons procédé au recueil nécessaire des besoins fonctionnels en faisant une liste des indicateurs et des axes d'analyse ayant un enjeu décisionnel pour notre projet. Nous avons également recueilli les besoins techniques en définissant notamment l'architecture adoptée dans le cadre du projet.

Une étude de faisabilité du projet a aussi été traitée dans ce chapitre et est la source de l'exactitude des besoins relevés.

Telles étaient les différentes briques de ce deuxième chapitre représentant une étape primordiale afin de faciliter la phase de conception et d'assurer le bon déroulement du projet. À cette étape, nous sommes en mesure de créer les matrices dimensionnelles de croisement à partir du croisement des axes et indicateurs relevés.

# **Chapitre 3**

---

## **Conception et modélisation de la Solution**

Nous présentons dans ce chapitre la phase de conception de notre projet. En y incluant les deux axes majeurs de ce dernier, matérialisés en la partie robotique et la partie lecture. Il sera question de présenter la solution théoriquement idéale à chaque solution et une autre pratique. Puis nous présenterons différents diagrammes de consolidation du plan de travail.

## 3 Conception et modélisation de la solution

La conception que nous avons établi pour notre projet se subdivise en trois parties distinctes. La première traitant de la lecture des tags en RFID, qui fait office d'ajout et d'innovation. La deuxième va se focaliser sur la lecture des codes-barres qui est une technologie préalablement utilisée de la part des entrepôts. Finalement nous traiterons de la partie robotique, avec à chaque instance la solution théorique idéale et la solution pratique prototypique.

### 3.1 Partie lecture RFID

#### 3.1.1 Solution idéale épurée

Les recherches approfondies que nous avons menées ont conclu que les lecteurs de tags RFID ont tendance à être perturbés par des environnements contenant des composants métalliques, ce qui n'est pas pratique pour l'utilisation que nous aspirons y faire. C'est pour cela que nous avons opté pour la solution que nous vous présentons ci-suit :

Un lecteur « **UHF Ultra Long-Range Reader DL950S** » est un lecteur particulièrement remarquable en ce qui concerne la portée de lecture, qui peut même atteindre 30 m, avec une fréquence de fonctionnement de 860 MHz jusqu'à 960 MHz et une vitesse de lecture de 200 à 300 tags simultanément, le lecteur UHF RFID DL950S se comporte bien pour identifier et encoder les tags, c'est une excellente réalisation qui offre un service pour toutes les industries et sera étendu à un plus grand nombre de pays.

#### **UHF Ultra Long-Range Reader DL950S:**

Fréquence de fonctionnement : 860 MHz ~ 960 MHz

Plage de lecture : 20m ~ 30m (dépend du tag et de l'environnement)

Vitesse de lecture : Lecture simultanée de 200 à 300 tags

Distance d'écriture : 10m ~ 15m (dépend du tag et de l'environnement)

Vitesse d'écriture : 8bits moins de 30ms

Consommation d'énergie : 6W



### UHF Gen 2 tags :



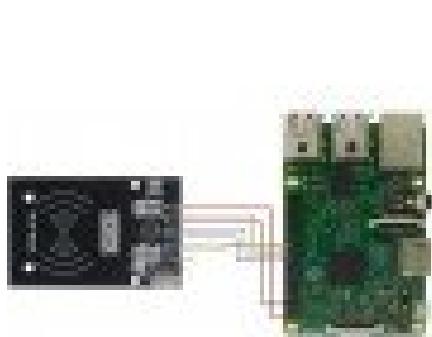
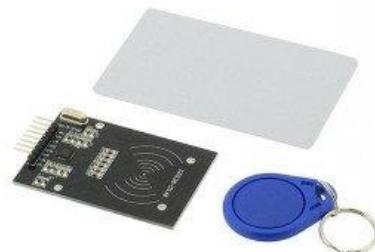
Les étiquettes **RFID UHF EPC Gen 2** sont conçues pour être montées sur du métal. Le protocole RFID UHF de code de produit électronique de classe 1, génération 2 (EPC, génération 2) a été établi en tant que norme pour les étiquettes RFID utilisées dans les applications de la chaîne logistique. Avec la conception ajoutée pour des capacités de lecture améliorées dans les environnements métalliques, ces étiquettes RFID garantissent une interrogation efficace même dans les environnements les plus difficiles.

### 3.1.2 Solution Prototypique

Le module RFID-RC522 de Joy-it est une carte d'interface compatible Arduino et Raspberry Pi, basée sur le circuit MFRC-522 de NXP et est utilisée pour lire et écrire sur des cartes ou badges RFID de type Mifare.

La carte microprocesseur (Arduino, Raspberry ou compatible) et le module RFID communiquent via le bus SPI permettant de laisser libres les autres ports de la carte pour d'autres applications.

Le module est livré avec une carte et un badge porte-clés utilisables en lecture et écriture.



Alimentation: 3,3 Vcc

Fréquence: 13,56 MHz

Protocole Mifare

Interface SPI

Dimensions (sans les broches): 61 x 40 mm

Hauteur avec les broches : 8 mm

## 3.2 Partie lecture Code-Barres

La lecture du code-barres se base sur une lecture visuelle, elle requiert donc l'utilisation d'une caméra liée à une carte programmable.

### 3.2.1 Solution Idéale

La caméra PTZ Pro 2 convient parfaitement aux salles d'entreposage avec un champ de vision diagonal à 90°, un contrôle flexible des fonctions de panoramique et d'inclinaison. Un zoom 10x avec mise au point automatique permet de cadrer parfaitement sur les codes-barres fixés sur les produits et leurs supports visuels, et offre aux systèmes d'enregistrement une bonne clarté.



La caméra Logitech PTZ Pro 2 fournit une optique de qualité supérieure ce qui est primordiale pour un traitement d'image se basant dans la plupart des cas sur le zoom. La partie logicielle qui devra intervenir est liée à la carte programmable Raspberry PI 3 qui va permettre de convertir les données photographiées en données binaires puis en données numériques présentes sur le code barre photographié.



Figure 9 Caméra LogiTech

### 3.2.2 Solution prototypique

HAVIC PC Webcam

Interface : USB 2.0

Capteur d'image : CMOS

Résolution : 640x288

Fréquence d'images : 30FPS/VGA

Format de sortie : YUY2/MJPEG

Sensibilité min : 2.0V/Lux

Focus distance : 20mm à extrêmement proche

Profondeur de vision : 50mm à l'infini

Contrôle du flash : Fréquence 50 Hz

Formats de stockage de l'image: BMP/JPG

Formats de stockage de la vidéo : AVI



Figure 10 Caméra Havic

### 3.3 Partie Robotique

**Raspberry Pi 3 Model B :**



Le Raspberry Pi 3 B est un ordinateur miniature de la taille d'une carte de crédit. Il repose sur un processeur à quadruple cœur ARM Cortex-A53, le BCM2837 de Broadcom, cadencé à 1,2 GHz. Cela signifie qu'il est 50 à 60% plus rapide que le Raspberry Pi 2 B. La communication par Wi-Fi 802.11n et du Bluetooth 4.1 est maintenant intégrée au RPi 3 ; ce nouveau modèle est toujours rétrocompatible avec les modèles précédents.

**Module RFID RC522 13,56 MHz :**

Ce module de lecteur RFID basé sur MFRC522 à faible coût est facile à utiliser et peut être utilisé dans une large gamme d'applications.

Le MFRC522 est un circuit intégré de lecture / écriture hautement intégré pour la communication sans contact à 13,56 MHz.

### Caractéristiques :

- Carte à puce MFRC522
- Fréquence de fonctionnement : 13,56 MHz
- Tension d'alimentation : 3.3V
- Courant : 13-26mA
- Portée de lecture : environ 3 cm avec la carte et le porte-clés fournis
- Interface SPI
- Taux de transfert de données maximum: 10 Mbit / s
- Dimensions : 60mm × 39m

### Mindstorm NXT 2.0 :



Lego Mindstorms NXT 2.0 est le deuxième ensemble de la série Lego Mindstorms de LEGO , lancé le 5 août 2009 à la boutique Lego aux États-Unis. Il contient 619 pièces, y compris un nouveau capteur capable de détecter les couleurs. Le prix est d'environ 280 USD, 350 USD, 230 GBP ou 500 AUD. LEGO Mindstorms NXT 2.0 a un successeur, appelé Lego Mindstorms EV3.

### 3.4 Diagrammes de contexte

Dans cette partie nous allons vous présenter des diagrammes pour pouvoir faciliter l'appréhension de tous les concepts et visions que nous essayons d'appliquer à ce projet. La modélisation de nos choix vous sera présentée sous forme de modèles dimensionnels.

### 3.4.1 Diagramme de séquence principal

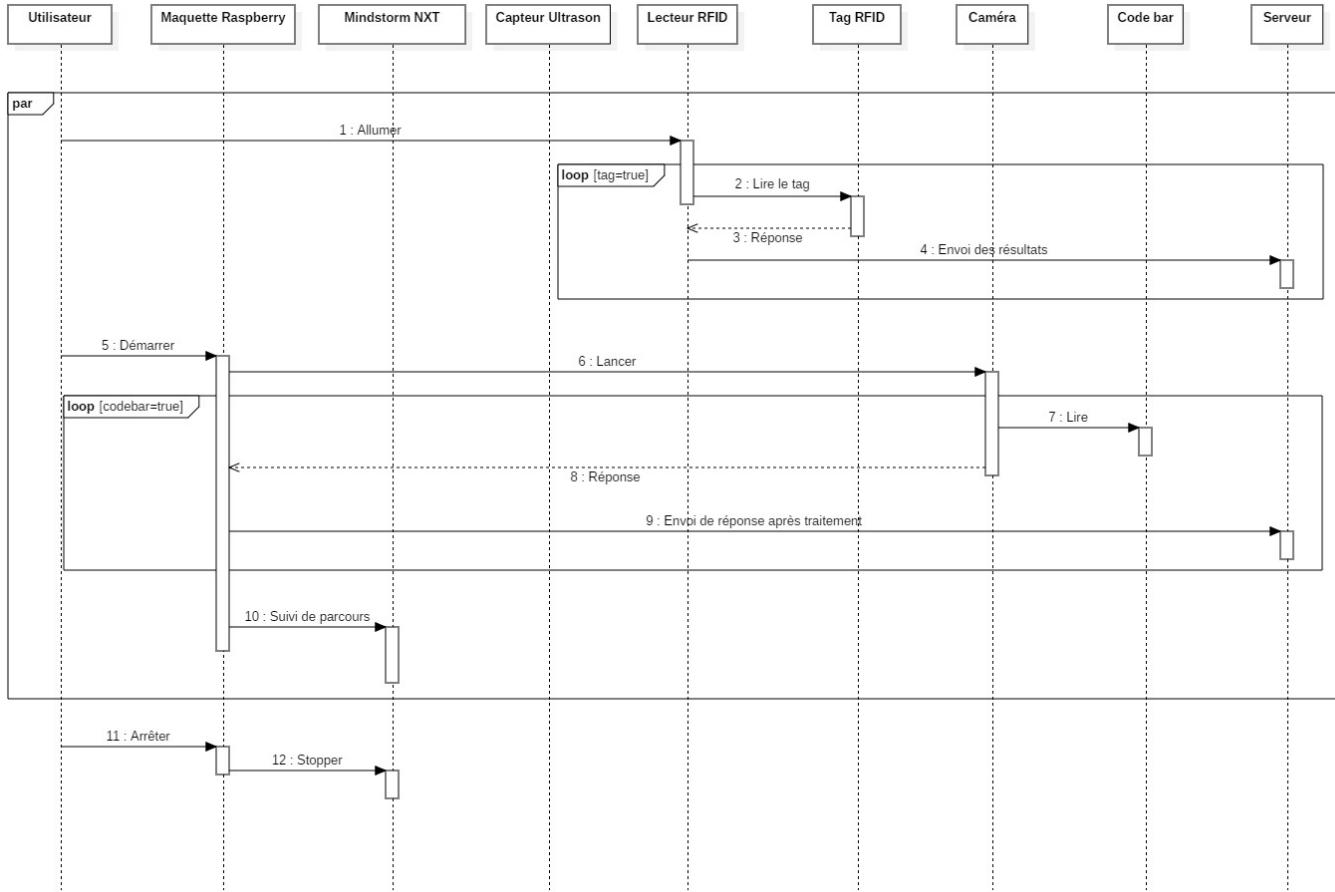


Figure 11 Diagramme de séquence principal

- Toutes les fonctions marchent en parallèle sauf l'arrêt qui est déclenché de façon unique.
- En premier lieu l'utilisateur allume le lecteur RFID et démarre la maquette Raspberry.
- La lecture des tags commence et l'opération se répète jusqu'à épuisement des tags.
- La maquette Raspberry se charge de démarrer la caméra et le robot MindStorm.
- Le robot démarre le suivi de parcours.
- La camera démarre l'acquisition visuelle.
- La caméra envoie des images à la maquette Raspberry qui s'occupe de les traiter et de les convertir en données numériques.

- Les données sont envoyées au serveur pour interfaçage et traitement.
- Le robot parcourt le chemin le plus optimisé.
- Toutes les étapes de lecture se répètent sous forme de boucle conditionnée à l'épuisement de tous les tags.
- Le robot s'arrête et l'utilisateur stoppe le processus.

### 3.4.2 Diagramme de séquence mappage

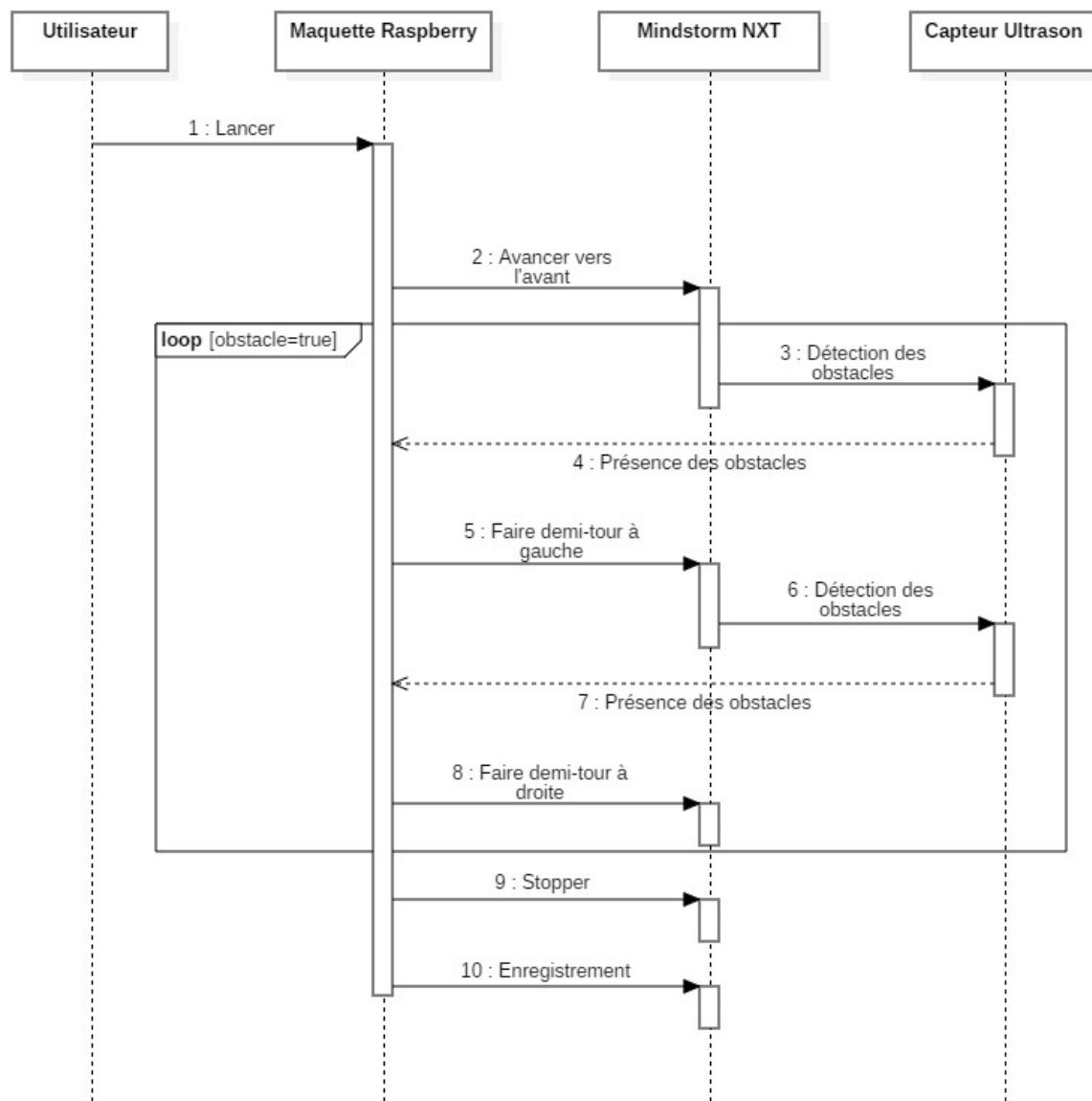


Figure 12 Diagramme de séquence mappage

- Ce diagramme met en démonstration les étapes de la toute première utilisation.
- L'utilisateur lance la commande qui démarre le robot.
- Le robot commence à parcourir un chemin.
- Le robot s'arrête à des intervalles réguliers pour prendre conscience des obstacles qui jonchent autour, et ceci grâce au pivotement du capteur ultrason à 180 degrés.
- Le robot s'arrête et l'enregistrement de la map s'effectue sur l'ordinateur.

### 3.4.3 Diagramme de déploiement

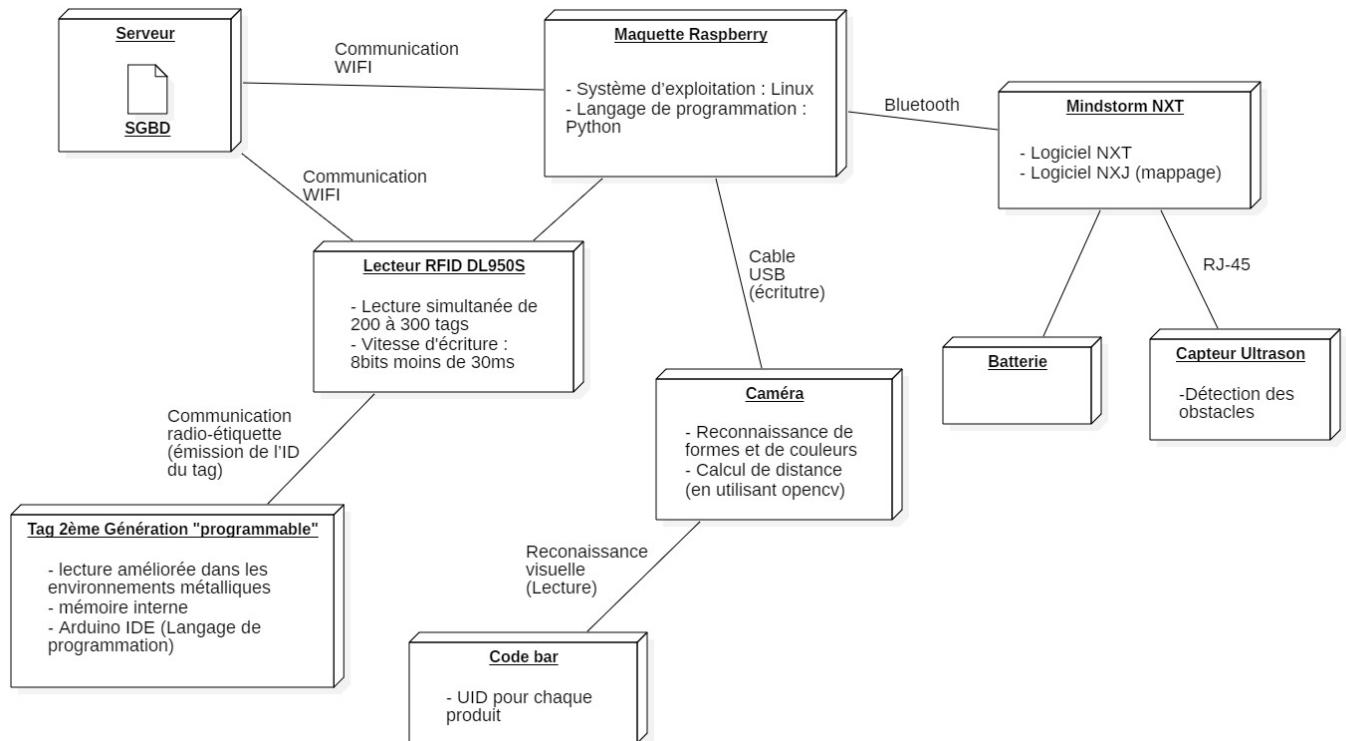


Figure 13 Diagramme de déploiement

## Conclusion

Nous avons présenté dans ce chapitre la conception des différents axes de notre projet. Cette conception a été faite sur la base de diagrammes séquentiels. En effet, ces dernières établissent un croisement entre les indicateurs relatifs à chacun des domaines et les différents axes d'analyses pouvant le ventiler.

Arrivé à ce point du projet, toutes les conditions sont rassemblées afin d'entamer la phase de réalisation qui rappelle souvent de constater la pertinence de la conception. Le prototypage est en effet le meilleur moyen de s'assurer d'une bonne conception et de fournir le résultat le plus fidèle et démonstratif possible.

# **Chapitre 4**

---

## **Réalisation de la phase de lecture**

Ce chapitre vient concrétiser les résultats des phases d'analyse et de conception. Il présente ainsi la démarche adoptée pour la réalisation du back end et du front end du projet, quelques exemples d'outils utilisés, ainsi que des exemples de traitements logiciels.

## 4 Réalisation de la phase de lecture

### 4.1 Description de la technologie utilisée RFID

#### Radio-identification

La **radio-identification**, le plus souvent désignée par le sigle **RFID** est une méthode pour mémoriser et récupérer des données à distance en utilisant des marqueurs appelés « radio-étiquettes » (« *RFID tag* » ou « *RFID transponder* » en anglais).

Les radio-étiquettes sont de petits objets, tels que des étiquettes autoadhésives, qui peuvent être collés ou incorporés dans des objets ou produits et même implantés dans des organismes vivants (animaux, corps humain). Les radio-étiquettes comprennent une antenne associée à une puce électronique qui leur permet de recevoir et de répondre aux requêtes radio émises depuis l'émetteur-récepteur.

Ces puces électroniques contiennent un identifiant et éventuellement des données complémentaires.

Cette technologie d'identification peut être utilisée pour identifier :

- les objets, comme avec un code-barres (on parle alors d'étiquette électronique) ;
- les personnes, en étant intégrée dans les passeports, carte de transport, carte de paiement (on parle alors de carte sans contact) ;
- les carnivores domestiques (chats, chiens et furets) dont l'identification RFID est obligatoire dans de nombreux pays, en étant implantée sous la peau. C'est également le cas de manière non obligatoire pour d'autres animaux de travail, de compagnie ou d'élevage de rente (on parle alors de puce sous-cutanée).

#### Historique

La première utilisation du RFID est militaire. Dès 1935, Robert Watson-Watt développe une application pour l'armée britannique, permettant de différencier les avions ennemis des alliés : c'est le système d'identification IFF « Identification friend or foe », qui reste le principe de base utilisé de nos jours pour le contrôle du trafic aérien.

Entre 1948 et 1952, H. Stockman et F. L. Vernon écrivent les premiers articles scientifiques sur la RFID. Leurs articles sont considérés comme les fondements de la technologie RFID. Harry Stockman a notamment prédit que « ...un travail de développement et de recherche considérable doit être fait avant que les problèmes fondamentaux de la communication par puissance réfléchie soient résolus et que le domaine des applications utiles soit exploré... ».

Dans les années 1960, les applications commerciales sont de plus en plus visées. Le premier tag fait son apparition en 1966. Cette première étiquette RFID (1-bit) est développée et commercialisée sous l'acronyme EAS (Electronic Article Surveillance), la seule information porte sur la détection ou non du tag. D'autres brevets sont déposés autour de la problématique du contrôle d'accès. La théorie fondamentale sur laquelle s'appuie la RFID est décrite précisément à travers plusieurs publications, dont celles de R. Harrington et de J. K. Schindler.

Le premier brevet associé à l'abréviation RFID a été accordé à Charles Walton en 1983.

Les années 1990 marquent le début de la normalisation pour une interopérabilité des équipements RFID.

En 1999, des industriels créent le centre d'identification automatique (Auto-ID Center) au MIT avec l'objectif de standardiser la technologie RFID. Ce centre a été fermé en 2003 lorsque les travaux sur le code produit électronique (EPC) ont été achevés, et les résultats ont été transférés au EPCglobal Inc. nouvellement fondée par le Uniform Code Council (UCC) et EAN International (dénommés maintenant GS1 US et GS1).

Depuis 2005, les technologies RFID sont largement répandues dans la majorité des secteurs industriels (aéronautique, automobile, logistique, transport, santé, vie quotidienne, etc.). L'ISO (International Standard Organisation) a largement contribué à la mise en place de normes tant techniques qu'applicatives permettant d'avoir un haut degré d'interopérabilité voire d'interchangeabilité.

## Principe

Un système de radio-identification est composé de deux entités qui communiquent entre elles :

- Un marqueur, nommé radio-étiquette, tag RFID, ou encore transpondeur (de l'anglais *transponder*, contraction des mots *transmitter* et *responder*), apposé sur l'élément à identifier et encodant des données numériques. Ces données peuvent être lues sans ligne de vue directe, contrairement aux code-barres, avec une détection automatique et avec des distances de lecture supérieures (de 10 à 200 m selon le type de puces).
- D'un ou plusieurs lecteurs RFID, appelés aussi interrogateurs, coupleurs, ou station de base.

À ces deux éléments s'ajoute généralement un intergiciel (middleware) ou application hôte, constitué d'un terminal (ordinateurs de supervision), connecté au lecteur, et permettant l'exploitation des données collectées.

Le système est activé par un transfert d'énergie électromagnétique. Le lecteur agit généralement en maître, il envoie une onde électromagnétique en direction de l'objet à identifier. Il active ainsi le marqueur, qui lui renvoie de l'information.

Le lecteur envoie des requêtes aux tags RFID pour récupérer des données stockées dans leur mémoire. Le tag, généralement télé-alimenté par le signal du lecteur, génère en premier lieu un code permettant d'identifier l'objet sur lequel il est déposé. La communication entre les deux entités s'engage. Le lecteur peut procéder à une écriture d'information dans le tag.

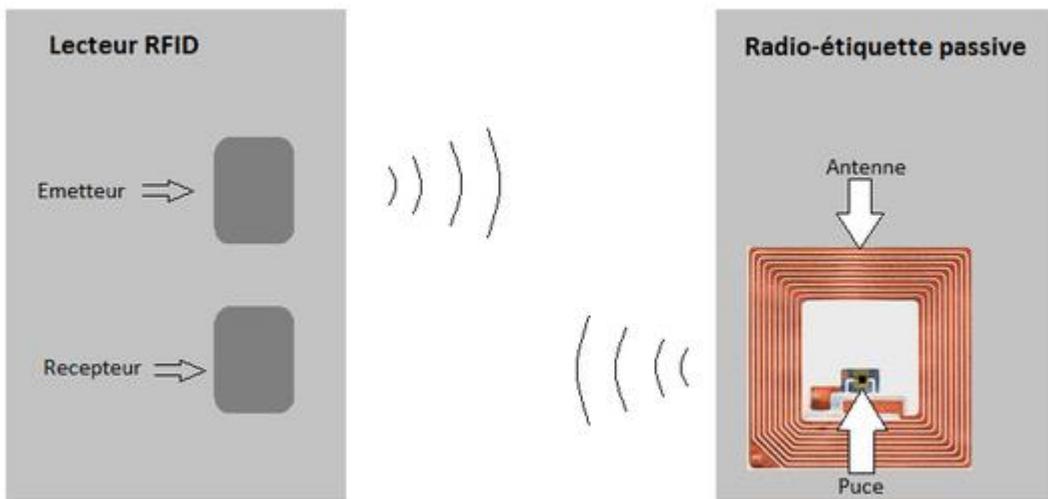


Figure 14 Principe de communication RFID avec une radio-étiquette passive

## Lecteur

Le lecteur est le composant qui coordonne la communication RFID et assure la télé-alimentation des tags dans le cas de la RFID passifs. Il est composé d'un module radio fréquence pour la transmission et la réception, d'une unité de contrôle, d'une antenne, et d'une interface pour transmettre les données vers un terminal.

Les lecteurs sont des dispositifs actifs, émetteurs de radiofréquences qui vont activer les marqueurs qui passent devant eux en leur fournissant à courte distance l'énergie dont ceux-ci ont besoin. Ainsi, le lecteur est constitué d'un circuit qui émet une énergie électromagnétique à travers une antenne, et une énergie électronique, qui reçoit et décide les informations envoyées par les marqueurs, puis les envoie au dispositif de collecte des données. Le lecteur est aussi à même d'écrire du contenu sur le tag RFID. Le lecteur RFID est l'élément responsable de la lecture des étiquettes radiofréquence, de l'écriture de contenu sur le tag RFID si besoin, et de la transmission des informations vers le middleware.

## Fréquence

La fréquence est la caractéristique qui permet d'établir la communication entre la puce et l'antenne. Cette fréquence utilisée est variable, selon le type d'application visé et les performances recherchées :

- Basse fréquence
  - 125 kHz ;
  - 134,2 kHz pour la charge du transpondeur ; 134,2 kHz pour un bit 0 et 123,2 kHz pour un bit 1 pour la réponse du transpondeur dans le cas d'une transmission FSK (Texas Instruments Séries 2000) ;

Les caractéristiques physiques de ces tags, d'un poids et une taille réduits, font d'eux des candidats idéals pour être d'une part intégrée dans tout type de matériaux (textiles, métaux, plastiques, etc.), et d'autre part pour l'identification du bétail. Les basses-fréquences permettent une lecture en tout milieu, mais à courte distance (quelques décimètres au maximum).

- Haute fréquence
  - 13,56 MHz (ISO 14 443 A 1-4, ISO 14443B 1-4, ISO 15693-3 et ISO 18000-3), la plus répandue actuellement dans l'industrie et le grand public pour des applications à portée limitée ;

Ces tags sont particulièrement fins, les antennes boucle pouvant être imprimées ou gravées. Ils sont utilisés pour des applications de logistique et de traçabilité, par exemple dans les applications de transport et d'identité : passeport, badge de transport comme le pass Navigo, badge de ski, cartes sans contact, contrôle d'accès des bâtiments, etc. Cette technologie est à la base des applications NFC (Near Field Communication), que l'on trouve dans de plus en plus de smartphones. Cette fréquence permet une lecture à une distance de l'ordre du mètre, mais est plus sensible à la proximité de métaux ou de liquides.

- Ultra haute fréquence
  - Ces fréquences ne sont pas harmonisées dans toutes les régions du monde, variant entre 860 et 960 MHz : 915 MHz aux États-Unis, de 865 MHz à 868 MHz dans l'Union européenne pour l'UHF (EPCglobal (en) et ISO 18000-6c). Les fréquences et les puissances d'émission dépendent des législations en vigueur. En conséquence, les tags doivent généralement présenter des bandes passantes importantes qui réduisent leurs performances.

Une application est par exemple le suivi des trains.

- Supra-haute fréquence
  - 2,45 GHz ou 5,8 GHz (micro-ondes), permet des portées de plusieurs mètres, utilisé pour le télépeage notamment.

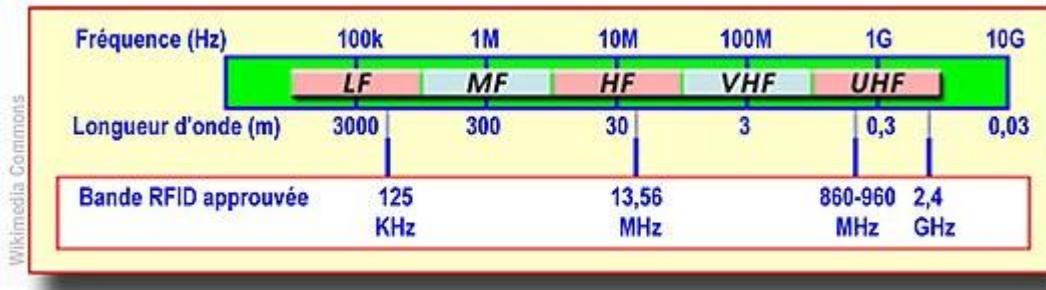


Figure 15 Bandes autorisées et possibles de fréquences pour les puces RFID communément commercialisées

Figure 16 Gammes de fréquences les plus utilisées et quelques applications RFID

Familles de fréquences	Bandes de fréquences	Régulations	Portée	Taux de transfert	Capacité de lecture près du métal ou des surfaces mouillées	Type de couplage	ISO/CEI 18000	Applications typiques
LF	120–150 kHz	Non régulé	10 cm–50 cm	Lent	Le meilleur	Couplage inductif	ISO/CEI 18000-Partie 2	Suivi des animaux, gestion des accès
HF	13.56 MHz	Band ISM	10 cm–1 m	Lent à moyen	Moyen (Susceptibilité au métal)	Couplage inductif	ISO/CEI 18000-Partie 3	Suivi des bagages, des livres dans les bibliothèques, surveillance électronique d'articles, porte-monnaie électronique, contrôle d'accès
UHF	433 MHz	Appareils de courte portée	1–100 m	Moyen à rapide	Mauvais	Couplage électrique	ISO/CEI 18000-Partie 7	Suivi dans la chaîne d'approvisionnement et gestion d'entrepôt, applications pour la défense
UHF	865–868 MHz (Europe) 902–928 MHz (Amérique du Nord)	Band ISM	1–12 m	Rapide	Mauvais	Couplage électrique	ISO/CEI 18000-Partie 6	Code-barres EAN, suivi de chemin de fer <sup>41</sup> , système de télécommande
SHF	2450–5800 MHz	Band ISM	1–2 m	Très rapide	Le pire	Couplage électrique	ISO/CEI 18000-Partie 4	Télémétrie, suivi de chemin de fer, 802.11 WLAN, standards Bluetooth
ULB	3.1–10 GHz	ULB	Supérieur à 200 m	Très rapide	-	Couplage électrique	Non défini	-

Une fréquence plus élevée présente l'avantage de permettre un échange d'informations (entre lecteur et marqueur) à des débits plus importants qu'en basse fréquence, et à une distance de lecture plus grande. Les débits importants permettent l'implémentation de nouvelles fonctionnalités au sein des marqueurs (cryptographie, mémoire plus importante, anticollision). Cependant une fréquence plus basse bénéficiera d'une meilleure pénétration dans la matière.

Le lecteur et le tag sont équipés d'antennes, qui doivent s'adapter à l'environnement. De plus, la RFID doit cohabiter d'un point de vue spectral avec d'autres technologies sans fil.

L'anticollision est la possibilité pour un lecteur de dialoguer avec un marqueur lorsque plus d'un marqueur se trouvent dans son champ de détection. Plusieurs algorithmes d'anticollision sont décrits par les normes (ISO 14443, ISO 15693 et ISO 18000).

## Principaux types de lecteurs

- les lecteurs mobiles sont usuellement montés sur les chariots élévateurs, et offrent une mobilité et une flexibilité accrues dans les applications de type gestion d'entrepôts ;
- les lecteurs fixes servent majoritairement dans les configurations de types portiques ou convoyeurs ;
- les lecteurs portatifs sont en général utilisés dans les applications de recherche et location de produits dans un entrepôt et dont les antennes intégrées sont incorporées directement dans le dispositif.

## Radio-étiquette

Le transpondeur RFID détient l'information (ex. prix du produit, nom du manufacturier, date de péremption, etc.) sur une puce électronique miniaturisée, associée une antenne qui assure la transmission de l'information vers le lecteur RFID via fréquence radios.

Le marqueur se compose :

- d'une antenne ;
- d'une puce de silicium ;
- d'un substrat et/ou d'une encapsulation.

Un tag RFID est composé d'une antenne conçue pour fonctionner dans une bande de fréquence donnée, connectée à une puce électronique, qui stocke les données. Un circuit d'adaptation est nécessaire dans certains cas pour adapter l'impédance de l'antenne à celle de la puce.

La capacité d'information d'une étiquette RFID est typiquement de 2 kB, mais la plupart ne contiennent qu'un numéro d'identification de 96 ou 128 bits.

Outre de l'énergie pour l'étiquette, le lecteur envoie un signal d'interrogation particulier auquel répond l'étiquette. L'une des réponses les plus simples possibles est le renvoi d'une identification numérique, par exemple celle du standard EPC-96 qui utilise 96 bits. Une table ou une base de données peut alors être consultée pour assurer un contrôle d'accès, un comptage ou un suivi donné sur une ligne de montage, ainsi que toute statistique souhaitable.

Le marqueur est extrêmement discret par sa finesse (parfois celle d'une feuille de rhodoïd), sa taille réduite (quelques millimètres), et sa masse négligeable. Son coût étant devenu minime, on peut envisager de le rendre jetable, bien que la réutilisation soit plus « écologiquement correcte ».

Les tags RFID peuvent être classés en fonction de leur mode d'alimentation, leur fréquence d'utilisation, leur capacité cryptographique, leur protocole de communication, la présence ou non d'une puce électronique, leur performance de communication, leurs propriétés en lecture et/ou écriture, leur prix.

## Modes d'alimentation

### *Tag passif :*

Dénusés de piles, ces tags tirent leur énergie des ondes magnétique ou électromagnétiques émises par le lecteur au moment de leur interrogation. Ils rétromodulent l'onde issue de l'interrogateur pour transmettre des informations. Ils n'intègrent pas d'émetteurs RF. La rétention des données est estimée à 10 ans et 100 000 cycles d'écriture.

Ils sont peu coûteux à fabriquer : leur coût moyen de 2007 à 2016 se situe entre 0.10€ et 0.20€, et varie de 0,05€ au minimum à 1,5€. Ils sont généralement réservés à des productions en volume.

Auparavant, la lecture des puces passives était limitée à une distance d'environ 10 mètres, mais maintenant, grâce à la technologie utilisée dans les systèmes de communications avec l'espace bi lointain, cette distance peut s'étendre jusqu'à 200 mètres.

**Tag semi-actif :**

Les étiquettes semi-actives (aussi appelés semi-passives ou encore BAP, *Battery-Assisted Passive tags*, en français marqueurs passifs assistés par batterie) utilisent l'énergie du lecteur pour générer la réponse à une requête lecteur. Elles agissent comme des étiquettes passives au niveau communication. En revanche, les autres éléments de la puce tels que le microcontrôleur et la mémoire tirent leur énergie d'une pile. Cette batterie leur permet, par exemple, d'enregistrer des données lors du transport. Ces étiquettes sont utilisées dans les envois de produits sous température dirigée et enregistrent la température de la marchandise à intervalle régulier.

Ces tags sont plus robustes et plus rapides en lecture et en transmission que les tags passifs, mais ils sont aussi plus chers.

**Tag actif :**

Les étiquettes actives sont équipées d'une batterie leur permettant d'émettre un signal. De ce fait, ils peuvent être lus depuis de longues distances (100 m environ), contrairement aux marqueurs passifs. En général, les transpondeurs actifs ont une plus grande capacité mémoire pour stocker divers types d'information telles que le connaissance (128 Kb et plus). Ils sont principalement utilisés dans des applications de télématrice, pour communiquer un grand nombre d'informations sur de grandes distances.

Cependant, une émission active d'informations signale à tous la présence des marqueurs et pose des questions quant à la sécurité des marchandises. Autre limitation, leur durée de vie est de 5 ans maximum. Ces tags coûtent généralement plus chers (15 à 40 € en 2007). Le risque de collision entre la fréquence d'opération du transpondeur avec des ondes électromagnétiques usuelles est plus élevé, ce qui limite également la localisation très fine des produits.

Les étiquettes sans puce font aussi leur apparition. Comme leur nom l'indique, ils ne disposent pas de circuit électronique. C'est l'impression de l'étiquette, basée sur des principes physiques ou chimiques, qui engendre un identifiant unique. D'un coût très faible, ces dernières peuvent être une alternative aux codes-barres. Un exemple d'étiquette sans puce est le tag SAW (*surface acoustic wave*, onde acoustique de surface).

## Obstacles

**Environnements métalliques :**

La lecture de radio-étiquettes posées sur des objets situés dans un conteneur métallique est plus difficile. Du fait de la présence d'un plan de masse, l'accord de l'antenne du tag est modifié. Ceci peut réduire de manière drastique la distance de lecture. De nouvelles familles de tags intègrent la présence d'un plan métallique dans le design de l'antenne ce qui permet de garder des distances de lecture proches de celles observées sur des supports plus neutres. Dans tous les cas, un tag placé à l'intérieur d'une enceinte métallique ne pourra pas être lu par un lecteur situé à l'extérieur. C'est l'effet de cage de Faraday, qui réalise un blindage électromagnétique.

## Collisions

Lorsque plusieurs marqueurs se trouvent dans le champ d'un même lecteur, les communications sont brouillées par l'activité simultanée des marqueurs.

La détection de la collision est en fait une détection d'erreur de transmission, à l'aide d'un bit de parité, d'une somme de contrôle ou d'une fonction de hachage. Dès qu'une erreur est détectée, l'algorithme d'anticollision est appliqué.

Plusieurs méthodes d'anticollision ont été développées. Voici les quatre principales :

- La méthode fréquentielle : Chaque marqueur communique sur une plage de fréquences différente avec le lecteur. En pratique, c'est inutilisable à grande échelle.
- La méthode spatiale : Avec une antenne directionnelle et à puissance variable, le lecteur va couvrir petit à petit chaque partie de l'espace pour communiquer avec chaque marqueur et l'inhiber, en attendant de le réactiver pour ensuite communiquer avec. En pratique, la présence de deux marqueurs à faible distance l'un de l'autre rend cette méthode inefficace.
- La méthode temporelle : Le lecteur propose aux marqueurs une série de canaux de temps dans lesquels ils peuvent répondre. Les marqueurs choisissent de façon aléatoire le canal de temps dans lequel ils vont répondre. Si un marqueur est le seul à répondre dans ce canal de temps, il est détecté et inhibé par le lecteur. S'il y a plusieurs marqueurs qui répondent en même temps, il sera nécessaire d'effectuer à nouveau cette méthode. Petit à petit, tous les marqueurs sont connus et inhibés ; il suffit alors au lecteur de réactiver le marqueur avec lequel il souhaite communiquer. En pratique, le côté aléatoire fait que la durée de cette méthode est inconnue.
- La méthode systématique : Il existe de nombreux brevets décrivant des méthodes systématiques. Cette méthode consiste à détecter puis inhiber tour à tour tous les marqueurs en parcourant l'arbre de toutes les possibilités d'identifiants (par exemple, le lecteur envoie une requête du type « Tous les marqueurs dont le premier bit d'identification est 1 doivent se manifester. » Si un seul marqueur se manifeste, le lecteur l'inhibe, et s'intéresse ensuite aux marqueurs avec pour premier bit 0, et ainsi de suite). En pratique, cette méthode peut parfois s'avérer longue.

## Utilisations

### *Marquage d'objets :*

- Système implanté d'identification et mémorisation : de manière courante, des puces basse fréquence (125 à 135 kHz) sont utilisées pour la traçabilité d'objets. La traçabilité d'objets tels que des livres dans les librairies et les bibliothèques ou la localisation des bagages dans les aéroports utilise plutôt la classe haute fréquence (13,56 MHz).

- Une utilisation peu connue de la RFID et qui tend à se développer à l'avenir concerne la gestion rationnelle des déchets ménagers, afin de mettre en place une tarification incitative.
- Contrôle d'accès : il se fait par badge de « proximité » ou « mains-libres ». Certaines « clés électroniques » d'accès sont des marqueurs permettant la protection « sans serrures » de bâtiments ou portières automobiles. Les badges mains-libres, permettent une utilisation jusqu'à 150 cm (selon le type d'antenne utilisée). Ils peuvent contenir une Identité numérique ou un certificat électronique ou y réagir et permettent l'accès à un objet communicant ou son activation. Utilisé par exemple pour le contrôle d'accès à des systèmes de transports en commun (exemple Passe Navigo)
- Traçabilité distante d'objets (fixes ou mobiles) : Par exemple, des palettes et conteneurs peuvent être suivis dans des entrepôts ou sur les docks) via des marqueurs UHF (ultra haute fréquence).

#### *Transactions financières :*

Les systèmes de **paiement sans contact** tel que des cartes de crédit, des porte-clés, des cartes à puce ou d'autres dispositifs (téléphone mobile...) utilisent la technologie radio frequency identification et Near Field Communication pour effectuer des paiements sécurisés. Une puce intégrée et une antenne permettent aux consommateurs de payer avec leur carte (sans contact) sur un lecteur au point de vente.

Certains fournisseurs affirment que les transactions peuvent être presque deux fois plus rapides qu'une transaction classique. Il n'y a ni signature, ni saisie du code PIN requis pour les achats de moins de 25 \$ US aux États-Unis, moins de CHF 40 en Suisse et moins de 30 € pour la France.

#### *Marquage d'être vivants :*

- Identification de plantes (arbres de la ville de Paris), d'animaux d'élevage (vaches, cochons) ou d'animaux de compagnie comme les chats et les chiens (grâce à une puce installée sous la peau dans le cou), d'animaux sauvages (cigognes, manchots): ce sont généralement des puces basse fréquence (125 à 135 kHz).
- Relevés scientifiques : des marqueurs sont aussi des moyens de communication pour la collecte des données issues des relevés scientifiques (monitoring) produits dans un organisme ou par des stations de mesure isolées et autonomes (stations météorologiques, volcaniques ou polaires).
- Chez l'Homme : des radio-marqueurs sous-cutanées, originellement conçus pour la traçabilité des animaux, peuvent sans aucune contrainte technique être utilisés sur des humains. La société Applied Digital Solutions propose ainsi ses radio-marqueurs sous-cutanés (nom commercial : VeriChip) destinés à des humains, comme une solution pour identifier les fraudes, assurer l'accès protégé à des sites confidentiels, le stockage des données médicales et aussi comme un moyen de résoudre rapidement des enlèvements de personnalités importantes. Combinés à des capteurs sensibles aux fonctions principales du corps humain, ces systèmes sont aussi proposés comme solution intégrée de supervision de l'état de santé d'un patient.

## Applications existantes

- Accès aux transports publics
- Télpéages d'autoroutes.
- Suivis industriels en chaîne de montage.
- Inventaires : Une analyse académique effectuée chez Wal-Mart a démontré que la radio-identification peut réduire les ruptures d'inventaire de 30 % pour les produits ayant un taux de rotation entre 0,1 et 15 unités/jour.
- Saisie automatique d'une liste de produits achetés ou sortis du stock.
- Dans des universités comme Cornell, des cartes à radio-identification permettent aux étudiants de l'université d'accéder sans formalité à la bibliothèque vingt-quatre heures sur vingt-quatre et sept jours sur sept. Les livres sont munis eux aussi de radio-étiquettes, ce qui élimine toute perte de temps administrative lors des emprunts.
- Antivols utilisés dans les magasins, notamment pour la lutte contre la contrefaçon. Des étiquettes RFID antivol sont présentes directement sur les emballages ou sur les produits dans les étalages.
- Identification de livres pour enfants par le Nabaztag : tag pour téléchargement des livres audio correspondants.
- Identification de containers de substances chimiques, de médicaments.

## Applications potentielles

Les étiquettes « intelligentes » sont souvent envisagées comme un moyen de remplacer et d'améliorer les codes-barres de la norme UPC/EAN. Les radio-identifiants sont en effet assez longs et dénombrables pour envisager de donner à chaque objet un numéro unique, alors que les codes UPC utilisés actuellement ne permettent que de donner un numéro pour une classe de produits. Cette propriété de la radio-identification permet de tracer le déplacement des objets d'un endroit à un autre, depuis la chaîne de production jusqu'au consommateur final. C'est cette propriété qui fait que la technologie est considérée par de nombreux industriels de la chaîne logistique comme la solution technologique ultime à tous les problèmes de traçabilité, notion essentielle depuis les crises sanitaires liées aux filières alimentaires.

Cependant, les solutions de radio-identification, bien qu'opérationnelles, souffrent d'un manque de normalisation. La jungle des solutions proposées par les différents fabricants rend la traçabilité universelle difficile à réaliser.

## 4.2 Outils utilisés

### 4.2.1 Genuino UNO

L'Uno peut être programmé avec l'"Arduino Software (IDE). Sélectionnez "Arduino / Genuino Uno" dans le menu Outils> Conseil (selon le microcontrôleur sur votre carte). Pour plus de détails, voir la référence et des tutoriels. Les ATmega328 sur l'Uno sont préprogrammé avec un boot loader qui vous permet de télécharger le nouveau code à elle sans l'utilisation d'un programmeur de matériel externe. Il communique en utilisant le protocole original STK500.

Vous pouvez également contourner le bootloader et programmer le microcontrôleur à travers le (Programmation Serial In-Circuit) ICSP tête en utilisant Arduino ISP ou similaire ; voir ces instructions pour plus de détails. Le ATmega16U2 (ou 8U2 dans le rev1 et les conseils rev2) le code source du firmware est disponible dans le référentiel Arduino. Le ATmega16U2 / 8U2 est chargé avec un chargeur de démarrage DFU, qui peut être activé par :

Sur les cartes Rév1 : connecter le cavalier de soudure sur le dos de la carte (près de la carte de l'Italie), puis Rese ing l'8U2.

Sur les cartes Rev2 ou plus tard : il y a une résistance qui en tirant la ligne 8U2 / 16U2 HWB au sol, ce qui rend plus facile à mettre en mode DFU.



Figure 17 Genuino UNO

### 4.2.2 Proteus 8

**Proteus** est une suite logicielle permettant la CAO électronique éditée par la société Labcenter Electronics. Proteus est composé de deux logiciels principaux : *ISIS*, permettant entre autres la création de schémas et la simulation électrique, et *ARES*, destiné à la création de circuits imprimés.

Grâce à des modules additionnels, *ISIS* est également capable de simuler le comportement d'un microcontrôleur (PIC, Atmel, 8051, ARM, HC11...) et son interaction avec les composants qui l'entourent.

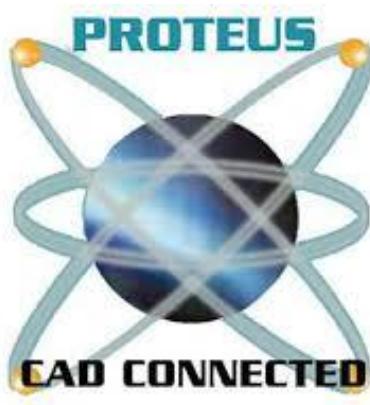


Figure 18 Proteus 8

### 4.2.3 Python 3

**Python** est un langage de programmation interprété, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions ; il est ainsi similaire à Perl, Ruby, Scheme, Smalltalk et Tcl.



Figure 19 Python 3

## 4.3 Codes et programmation effectuée

### 4.3.1 Solution RFID idéale

Le tag RFID de 2eme génération est un tag beaucoup plus élaboré que les tags des générations précédentes, ce qui explique les nombreux paramètres qui en découlent. Il est donc imperatif d'utiliser des logiciels plus complets tels que Genuino UNO :

```

// rfid_attendance_system | Arduino 1.8.8
Fichier Édition Croquis Outils Aide
rfid_attendance_system
#include <Wire.h>
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 9, 8, 7, 6); // declaring the lcd

char rfid_receive[13]; // buffer to store the reading card
char rfid_token1[13] = "1600660505AA"; // token to compare
char rfid_token2[13] = "8400811CECDE";
char rfid_token3[13] = "8500427942FC";
char rfid_token_master[13] = "6C00908D3B4A"; // master token to check the attendance

int std_vikas=0;
int std_ankit=0;
int std_monty=0;

void setup()
{
    Serial.begin(9600); // initializing the serial with 9600 baud rate
    lcd.begin(20, 4); // initializing the lcd as 20x4
    lcd.print("show your card"); // printing first statement on lcd first line
}

void loop()
{
    if(Serial.available() > 0) // checking if card is punched or not
    {
        Serial.readBytesUntil('\0', rfid_receive, 12); // storing the card number in rfid_receive
        if(!strcmp(rfid_receive, rfid_token1)) // comparing the two string if matched
        {
            lcd.clear();
            lcd.setCursor(0,1); // selecting column and line and printing data
            lcd.print(" Welcome ");
            lcd.setCursor(0,2);
            lcd.print(" Mr.vikas sharma ");
            lcd.setCursor(0,3);
            lcd.print(" attendance = ");
        }
        else if(!strcmp(rfid_receive, rfid_token2))
        {
            lcd.clear();
            lcd.setCursor(0,1);
            lcd.print(" Welcome ");
            lcd.setCursor(0,2);
            lcd.print(" Mr. ankit sahu");
            lcd.setCursor(0,3);
            lcd.print(" attendance = ");
            lcd.print(+std_vikas);
            delay(1000);
            lcd.clear();
            lcd.setCursor(0,0);
            lcd.print("show your card");
        }
        else if(!strcmp(rfid_receive, rfid_token3))
        {
            lcd.clear();
            lcd.setCursor(0,1);
            lcd.print(" Welcome ");
            lcd.setCursor(0,2);
            lcd.print(" Mr. monty ");
            lcd.setCursor(0,3);
            lcd.print(" attendance = ");
            lcd.print(+std_monty);
            delay(1000);
            lcd.clear();
            lcd.setCursor(0,0);
            lcd.print("show your card");
        }
        else if(!strcmp(rfid_receive, rfid_token_master)) // this sequence shows the attendance of all person
        {
            lcd.clear();
            lcd.setCursor(0,1);
            lcd.print(" Welcome ");
            lcd.setCursor(0,2);
            lcd.print(" Mr. vikas sharma ");
            lcd.setCursor(0,3);
            lcd.print(" attendance = ");
            lcd.print(+std_vikas);
            lcd.setCursor(0,4);
            lcd.print(" Mr. ankit sahu");
            lcd.setCursor(0,5);
            lcd.print(" attendance = ");
            lcd.print(+std_vikas);
            lcd.setCursor(0,6);
            lcd.print(" Mr. monty ");
            lcd.setCursor(0,7);
            lcd.print(" attendance = ");
            lcd.print(+std_monty);
            delay(1000);
            lcd.clear();
            lcd.setCursor(0,0);
            lcd.print("show your card");
        }
    }
}

```



```

// rfid_attendance_system | Arduino 1.8.8
Fichier Édition Croquis Outils Aide
rfid_attendance_system
lcd.setCursor(0,3);
lcd.print(" attendance = ");
lcd.print(+atd_monty);
delay(1000);
lcd.clear();
lcd.setCursor(0,0);
lcd.print("show your card");
}
else if(!strcmp(rfid_receive,rfid_token_master))) // this sequence shows the attendance of all person
{
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Mr. vikas = ");
  lcd.print(atd_vikas);
  lcd.setCursor(0,1);
  lcd.print("Mr. monty = ");
  lcd.print(atd_monty);
  lcd.setCursor(0,2);
  lcd.print("Mr. ankit = ");
  lcd.print(atd_ankit);
  delay(5000);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("show your card");
}
else
{
  lcd.setCursor(0,1); // printing sequence if none of them matched
  lcd.print("not recognized ");
  delay(1);
  lcd.setCursor(0,2);
  lcd.print("please enter again ");
  delay(1000);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("show your card");
}
}

```

La simulation du fonctionnement de ce tag programmable peut se faire grâce au logiciel Proteus 8 avec la seule contrainte que le tag matériel doit être présent et connecté à l'ordinateur

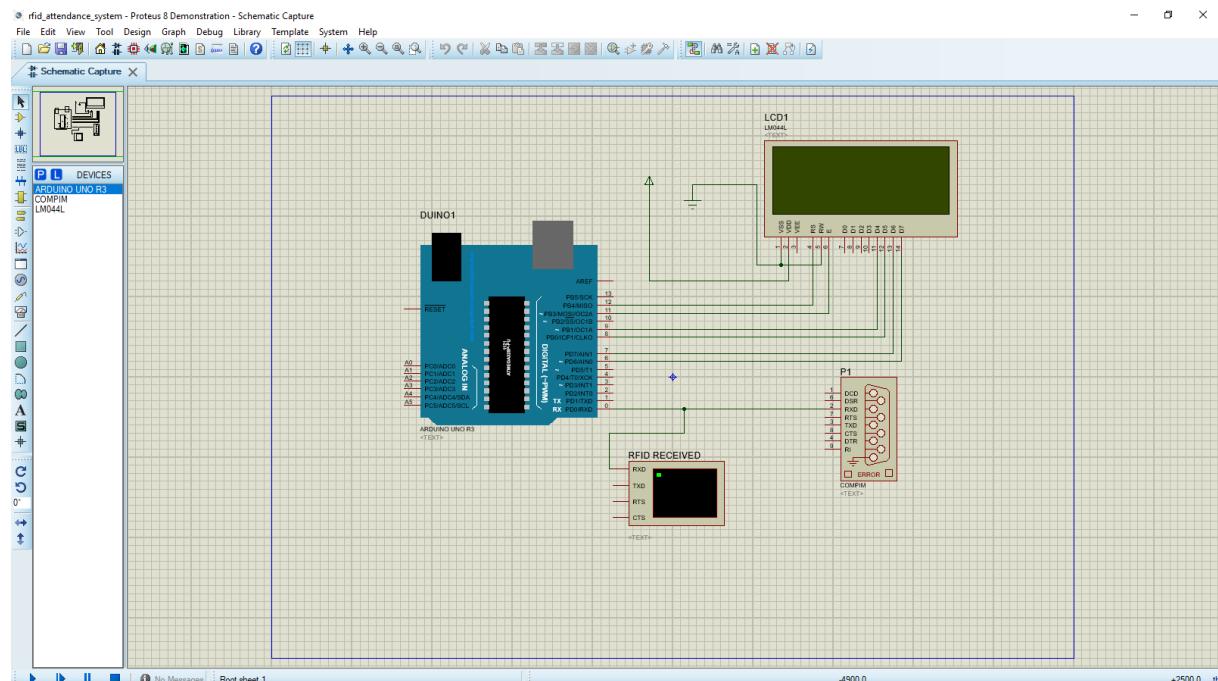


Figure 20 Simulation en Proteus 8

### 4.3.2 Solution RFID prototypique

La solution de lecture dont on a fait usage est accessible grâce au suivi de plusieurs étapes :

**1ère étape** : mettre à jour la Raspberry pi

- 1- sudo apt-get update
- 2- sudo apt-get upgrade
- 3- sudo raspi-config

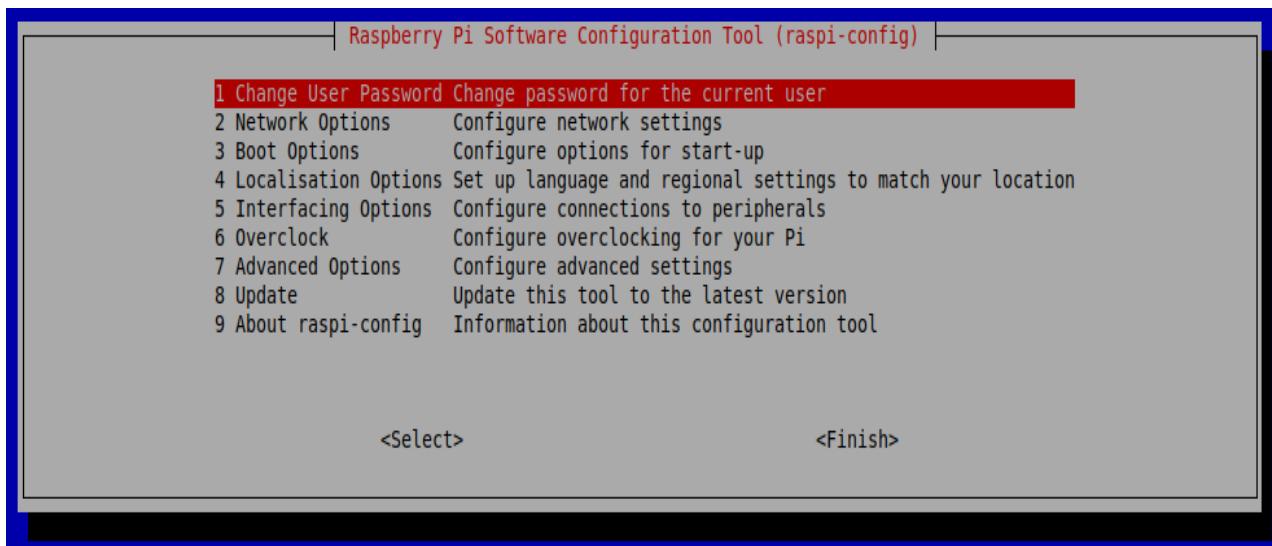


Figure 21 Intérface Raspberry

**2ème étape** : redémarrer la Raspberry

**3ème étape** : vérifier le démarrage du SPI (lsmod | grep spi)

**4ème étape** : installer la librairie de la carte

- 1- Passer au dossier SPI-PY (cd SPI-PY)
- 2- Lancer le setup (sudo python setup.py)

Figure 23:test\_script

```

import spi

def transact():

    data_out = (0xFF,0x00,0xFA)
    # Open file descriptor for
    # spi device 0 using the CE0 pin for chip select
    device_0 = spi.openSPI(device="/dev/spidev0.0",
                           mode=0,
                           speed=1000000)

    print("**")
    print(device_0)

    # This is not necessary, just demonstrate loop-back
    data_in = (0x00, 0x00, 0x00)
    data_in = spi.transfer(device_0, data_out)
    print("Received from device 0:")
    print(data_in)

    spi.closeSPI(device_0)
    print(device_0)
    print("**")

def main():

    g_run = True

    while g_run:
        try:
            transact()

        except KeyboardInterrupt:
            g_run = False

    if __name__ == "__main__":
        main()

```

Figure 24:memory\_leak

```

import spi

def main():
    # Open file descriptor for
    # spi device 0 using the CE0 pin for chip select
    device_0 = spi.openSPI(device="/dev/spidev0.0",
                           mode=0,
                           speed=1000000)

    # Open file descriptor for
    # spi device 0 using the CE1 pin for chip select
    device_1 = spi.openSPI(device="/dev/spidev0.1",
                           mode=0,
                           speed=1000000)

    # Transact data
    data_out = (0xFF, 0x00, 0xFF)

    # This is not necessary, not just demonstrate loop-back
    data_in = (0x00, 0x00, 0x00)
    data_in = spi.transfer(device_0, data_out)
    print("Received from device 0:")
    print(data_in)

    data_in = (0x00, 0x00, 0x00)
    data_in = spi.transfer(device_1, data_out)
    print("Received from device 1:")
    print(data_in)

    # Close file descriptors
    spi.closeSPI(device_0)
    spi.closeSPI(device_1)

if __name__ == "__main__":
    main()

```

### **5ème étape** : initialiser les paramètres MFRC522

CommIEnReg	= 0x02
DivlEnReg	= 0x03
CommIrqReg	= 0x04
DivIrqReg	= 0x05
ErrorReg	= 0x06
Status1Reg	= 0x07
Status2Reg	= 0x08
FIFODataReg	= 0x09
FIFOLevelReg	= 0x0A
WaterLevelReg	= 0x0B
ControlReg	= 0x0C
BitFramingReg	= 0x0D
CollReg	= 0x0E
Reserved01	= 0x0F
Reserved10	= 0x10
ModeReg	= 0x11
TxModeReg	= 0x12
RxModeReg	= 0x13
TxControlReg	= 0x14
TxAutoReg	= 0x15
TxSelReg	= 0x16
RxSelReg	= 0x17
RxThresholdReg	= 0x18
DemodReg	= 0x19
Reserved11	= 0x1A
Reserved12	= 0x1B
MifareReg	= 0x1C
Reserved13	= 0x1D
Reserved14	= 0x1E
SerialSpeedReg	= 0x1F
Reserved20	= 0x20
CRCResultRegM	= 0x21
CRCResultRegL	= 0x22
Reserved21	= 0x23
ModWidthReg	= 0x24
Reserved22	= 0x25
RFCfgReg	= 0x26
GsNReg	= 0x27

```

GsNReg          = 0x27
CWGsPReg        = 0x28
ModGsPReg       = 0x29
TModeReg        = 0x2A
TPrescalerReg   = 0x2B
TReloadRegH     = 0x2C
TReloadRegL     = 0x2D
TCounterValueRegH = 0x2E
TCounterValueRegL = 0x2F

Reserved30      = 0x30
TestSel1Reg     = 0x31
TestSel2Reg     = 0x32
TestPinEnReg    = 0x33
TestPinValueReg = 0x34
TestBusReg      = 0x35
AutoTestReg     = 0x36
VersionReg      = 0x37
AnalogTestReg   = 0x38
TestDAC1Reg     = 0x39
TestDAC2Reg     = 0x3A
TestADCReg      = 0x3B
Reserved31      = 0x3C
Reserved32      = 0x3D
Reserved33      = 0x3E
Reserved34      = 0x3F

serNum = []

def __init__(self, dev='/dev/spidev0.0', spd=1000000):
    spi.openSPI(device=dev, speed=spd)
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(22, GPIO.OUT)
    GPIO.output(self.NRSTPD, 1)
    self.MFRC522_Init()

def MFRC522_Reset(self):
    self.Write_MFRC522(self.CommandReg, self.PCD_RESETPHASE)

def Write_MFRC522(self, addr, val):

```

```

def Write_MFRC522(self, addr, val):
    spi.transfer(((addr<<1)&0x7E,val))

def Read_MFRC522(self, addr):
    val = spi.transfer(((addr<<1)&0x7E) | 0x80,0))
    return val[1]

def SetBitMask(self, reg, mask):
    tmp = self.Read_MFRC522(reg)
    self.Write_MFRC522(reg, tmp | mask)

def ClearBitMask(self, reg, mask):
    tmp = self.Read_MFRC522(reg);
    self.Write_MFRC522(reg, tmp & (~mask))

def AntennaOn(self):
    temp = self.Read_MFRC522(self.TxControlReg)
    if(~(temp & 0x03)):
        self.SetBitMask(self.TxControlReg, 0x03)

def AntennaOff(self):
    self.ClearBitMask(self.TxControlReg, 0x03)

def MFRC522_ToCard(self,command sendData):
    backData = []
    backLen = 0
    status = self.MI_ERR
    irqEn = 0x00
    waitIRq = 0x00
    lastBits = None
    n = 0
    i = 0

    if command == self.PCD_AUTHENT:
        irqEn = 0x12
        waitIRq = 0x10
    if command == self.PCD_TRANSCEIVE:
        irqEn = 0x77
        waitIRq = 0x30

```

```

self.Write_MFRC522(self.CommEnReg, irqEn|0x80)
self.ClearBitMask(self.CommIrqReg, 0x80)
self.SetBitMask(self.FIFOLevelReg, 0x80)

self.Write_MFRC522(self.CommandReg, self.PCD_IDLE);

while(i<len(sendData)):
    self.Write_MFRC522(self.FIFODataReg, sendData[i])
    i = i+1

self.Write_MFRC522(self.CommandReg, command)

if command == self.PCD_TRANSCEIVE:
    self.SetBitMask(self.BitFramingReg, 0x80)

i = 2000
while True:
    n = self.Read_MFRC522(self.CommIrqReg)
    i = i - 1
    if ~((i!=0) and ~(n&0x01) and ~(n&waitIRq)):
        break

self.ClearBitMask(self.BitFramingReg, 0x80)

if i != 0:
    if (self.Read_MFRC522(self.ErrorReg) & 0x1B)==0x00:
        status = self.MI_OK

    if n & irqEn & 0x01:
        status = self.MI_NOTAGERR

    if command == self.PCD_TRANSCEIVE:
        n = self.Read_MFRC522(self.FIFOLevelReg)
        lastBits = self.Read_MFRC522(self.ControlReg) & 0x07
        if lastBits != 0:
            backLen = (n-1)*8 + lastBits
        else:
            backLen = n*8

```

```

    if n == 0:
        n = 1
    if n > self.MAX_LEN:
        n = self.MAX_LEN

    i = 0
    while i<n:
        backData.append(self.Read_MFRC522(self.FIFODataReg))
        i = i + 1;
    else:
        status = self.MI_ERR

    return (status,backData,backLen)

def MFRC522_Request(self, reqMode):
    status = None
    backBits = None
    TagType = []

    self.Write_MFRC522(self.BitFramingReg, 0x07)

    TagType.append(reqMode);
    (status,backData,backBits) = self.MFRC522_ToCard(self.PCD_TRANSCEIVE, TagType)

    if ((status != self.MI_OK) | (backBits != 0x10)):
        status = self.MI_ERR

    return (status,backBits)

def MFRC522_Anticoll(self):
    backData = []
    serNumCheck = 0

    serNum = []

    self.Write_MFRC522(self.BitFramingReg, 0x00)

```

```

serNum.append(self.PICC_ANTICOLL)
serNum.append(0x20)

(status,backData,backBits) = self.MFRC522_ToCard(self.PCD_TRANSCEIVE,serNum)

if(status == self.MI_OK):
    i = 0
    if len(backData)==5:
        while i<4:
            serNumCheck = serNumCheck ^ backData[i]
            i = i + 1
        if serNumCheck != backData[i]:
            status = self.MI_ERR
    else:
        status = self.MI_ERR

return (status,backData)

def CalulateCRC(self, pIndata):
    self.ClearBitMask(self.DivIrqReg, 0x04)
    self.SetBitMask(self.FIFOLevelReg, 0x80);
    i = 0
    while i<len(pIndata):
        self.Write_MFRC522(self.FIFODataReg, pIndata[i])
        i = i + 1
    self.Write_MFRC522(self.CommandReg, self.PCD_CALCCRC)
    i = 0xFF
    while True:
        n = self.Read_MFRC522(self.DivIrqReg)
        i = i - 1
        if not ((i != 0) and not (n&0x04)):
            break
    pOutData = []
    pOutData.append(self.Read_MFRC522(self.CRCResultRegL))
    pOutData.append(self.Read_MFRC522(self.CRCResultRegM))
    return pOutData

def MFRC522_SelectTag(self, serNum):
    backData = []

```

```

def MFRC522_SelectTag(self, serNum):
    backData = []
    buf = []
    buf.append(self.PICC_SELECTTAG)
    buf.append(0x70)
    i = 0
    while i<5:
        buf.append(serNum[i])
        i = i + 1
    pOut = self.CalculateCRC(buf)
    buf.append(pOut[0])
    buf.append(pOut[1])
    (status, backData, backLen) = self.MFRC522_ToCard(self.PCD_TRANSCEIVE, buf)

    if (status == self.MI_OK) and (backLen == 0x18):
        return backData[0]
    else:
        return 0

def MFRC522_Auth(self, authMode, BlockAddr, Sectorkey, serNum):
    buff = []

    # First byte should be the authMode (A or B)
    buff.append(authMode)

    # Second byte is the trailerBlock (usually 7)
    buff.append(BlockAddr)

    # Now we need to append the authKey which usually is 6 bytes of 0xFF
    i = 0
    while(i < len(Sectorkey)):
        buff.append(Sectorkey[i])
        i = i + 1
    i = 0

```

```

# Next we append the first 4 bytes of the UID
while(i < 4):
    buff.append(serNum[i])
    i = i +1

# Now we start the authentication itself
(status, backData, backLen) = self.MFRC522_ToCard(self.PCD_AUTHENT,buff)

# Check if an error occurred
if not(status == self.MI_OK):
    print ("ERREUR AUTHENTIFICATION")
if not (self.Read_MFRC522(self.Status2Reg) & 0x08) != 0:
    print ("AUTH ERROR (status2reg & 0x08) != 0")

# Return the status
return status

def MFRC522_StopCrypto1(self):
    self.ClearBitMask(self.Status2Reg, 0x08)

def MFRC522_Read(self, blockAddr):
    recvData = []
    recvData.append(self.PICC_READ)
    recvData.append(blockAddr)
    pOut = self.CalulateCRC(recvData)
    recvData.append(pOut[0])
    recvData.append(pOut[1])
    (status, backData, backLen) = self.MFRC522_ToCard(self.PCD_TRANSCEIVE, recvData)
    if not(status == self.MI_OK):
        print ("Erreur durant la lecture")
    i = 0
    if len(backData) == 16:
        print ("Secteur "+str(blockAddr)+" "+str(backData))
        print("Traduction en ASCII : ", end='')
        c=0
        while (c<16):
            if(backData[c]!=0) :
                try :
                    print (str(unichr(backData[c])),end="")

```

```

        except :
            print(" Contenu Illisible")
            c+=1
            print("\n")

def MFRC522_Write(self, blockAddr, writeData):
    buff = []
    buff.append(self.PICC_WRITE)
    buff.append(blockAddr)
    crc = self.CalculateCRC(buff)
    buff.append(crc[0])
    buff.append(crc[1])
    (status, backData, backLen) = self.MFRC522_ToCard(self.PCD_TRANSCEIVE, buff)
    if not(status == self.MI_OK) or not(backLen == 4) or not((backData[0] & 0x0F) == 0x0A):
        status = self.MI_ERR

    if status == self.MI_OK:
        i = 0
        buf = []
        while i < 16:
            buf.append(writeData[i])
            i = i + 1
        crc = self.CalculateCRC(buf)
        buf.append(crc[0])
        buf.append(crc[1])
        (status, backData, backLen) = self.MFRC522_ToCard(self.PCD_TRANSCEIVE,buf)
        if not(status == self.MI_OK) or not(backLen == 4) or not((backData[0] & 0x0F) == 0x0A):
            print ("Erreur durant l'\\'ecriture")
        if status == self.MI_OK:
            print ("Ecriture terminee")

def MFRC522_DumpClassic1K(self, key, uid):
    i = 0
    while i < 64:
        status = self.MFRC522_Auth(self.PICC_AUTHENT1A, i, key, uid)
        # Check if authenticated
        if status == self.MI_OK:
            self.MFRC522_Read(i)

```

```

    else:
        print ("Erreur d\\'Authentification")
        i = i+1

```

```

def MFRC522_Init(self):
    GPIO.output(self.NRSTPD, 1)

    self.MFRC522_Reset();

```

```

    self.Write_MFRC522(self.TModeReg, 0x8D)
    self.Write_MFRC522(self.TPrescalerReg, 0x3E)
    self.Write_MFRC522(self.TReloadRegL, 30)
    self.Write_MFRC522(self.TReloadRegH, 0)

    self.Write_MFRC522(self.TxAutoReg, 0x40)
    self.Write_MFRC522(self.ModeReg, 0x3D)
    self.AntennaOn()

```

**6ème étape** : écrire un programme de lecture de tag

```

#!/usr/bin/env python
# -*- coding: utf8 -*-
import RPi.GPIO as GPIO
import MFRC522
import signal
continue_reading = True
# Fonction qui arrete la lecture proprement
def end_read(signal,frame):
    global continue_reading
    print ("Lecture terminée")
    continue_reading = False
    GPIO.cleanup()
signal.signal(signal.SIGINT, end_read)
MIFAREReader = MFRC522.MFRC522()
print ("Passer le tag RFID a lire")
while continue_reading:
    # Detecter les tags
    (status,TagType) = MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)
    # Une carte est detectee
    if status == MIFAREReader.MI_OK:
        print ("Carte detectee")
    # Recuperation UID
    (status,uid) = MIFAREReader.MFRC522_Anticoll()
    if status == MIFAREReader.MI_OK:
        print ("UID de la carte : "+str(uid[0])+"."+str(uid[1])+"."+str(uid[2])+"."+str(uid[3]))
        # Cleee d authentification par defaut
        key = [0xFF,0xFF,0xFF,0xFF,0xFF,0xFF]
        # Selection du tag
        MIFAREReader.MFRC522_SelectTag(uid)
        # Authentification
        status = MIFAREReader.MFRC522_Auth(MIFAREReader.PICC_AUTHENT1A, 8, key, uid)
        if status == MIFAREReader.MI_OK:
            MIFAREReader.MFRC522_Read(8)
            MIFAREReader.MFRC522_StopCrypto1()
        else:
            print ("Erreur d'\Authentification")

```

**7ème étape** : écrire un programme d’écriture sur les tags

```

#!/usr/bin/env python
# -*- coding: utf8 -*-
import RPi.GPIO as GPIO
import MFRC522
import signal
continue_reading = True
# Fonction qui arrete la lecture proprement
def end_read(signal,frame):
    global continue_reading
    print ("Lecture terminée")
    continue_reading = False
    GPIO.cleanup()
signal.signal(signal.SIGINT, end_read)
MIFAREReader = MFRC522.MFRC522()
data = []
texte = raw_input("Entrez une chaîne de caractère :\n")
for c in texte:
    if (len(data)<16):
        data.append(int(ord(c)))
while(len(data)!=16):
    data.append(0)
print ("Placez votre carte RFID")
while continue_reading:
    # Detecter les tags
    (status,TagType) = MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)
    # Une carte est détectée
    if status == MIFAREReader.MI_OK:
        print ("Carte détectée")
    # Recuperation UID
    (status,uid) = MIFAREReader.MFRC522_Anticoll()
    if status == MIFAREReader.MI_OK:
        # Print UID
        print ("UID de la carte : "+str(uid[0])+". "+str(uid[1])+". "+str(uid[2])+". "+str(uid[3]))
        # Clé d'authentification par défaut
        key = [0xFF,0xFF,0xFF,0xFF,0xFF]
        # Sélection du tag
        MIFAREReader.MFRC522_SelectTag(uid)
        # Authentification
        status = MIFAREReader.MFRC522_Auth(MIFAREReader.PICC_AUTHENT1A, 8, key, uid)
        if status == MIFAREReader.MI_OK:
            print ("Le secteur 8 contient actuellement : ")
            MIFAREReader.MFRC522_Read(8)
            print ("Ecriture ...")
            MIFAREReader.MFRC522_Write(8, data)
            print ("Le secteur 8 contient maintenant : ")
            MIFAREReader.MFRC522_Read(8)
            # Stop
            MIFAREReader.MFRC522_StopCrypto1()
            continue_reading = False
        else:
            print ("Erreur d'authentification")

```

## Conclusion

Ce quatrième vient clore le cycle de développement de la solution de lecture. Les résultats auxquels a abouti le projet en termes de réalisation n'y sont pas tous répertorié pour cause de redondance, le quatrième chapitre de ce mémoire comporte donc une vue d'ensemble sur comment s'est opéré le choix de chaque utilitaire illustrant chaque type de technologies à disposition, tout cela dans le but de donner une perspective plus simplifié mais tout autant performante et optimisée du smart-warehousing.

# **Chapitre 5**

---

## **Réalisation de la partie robotique**

Ce chapitre vient concrétiser les résultats des phases d'analyse et de conception. Il présente ainsi la démarche adoptée pour la réalisation du back end et du front end du projet, quelques exemples d'outils utilisés, ainsi que des exemples de traitements logiciels.

## 5 Réalisation de la partie robotique

### 5.1 Présentation de la carte NXT (MindStorm)

La brique NXT est l'organe principal du kit est le « cerveau » du robot MINDSTROMTM.  
Elle est constituée :

D'un processeur ATMEL® ARM7 32 bits, fonctionnant à 48 Mhz (associé à 256 ko de mémoire flash et 64 ko de RAM) qui permet notamment l'exécution des programmes

D'un coprocesseur ATMEL® AVR ATmega48, fonctionnant à 8 Mhz (associé à 4 ko de mémoire flash et 512 o de RAM) qui a en charge la gestion des périphériques

De 4 ports d'entrée (1, 2, 3 et 4) destinés à la connexion des capteurs

De 3 ports de sortie (A, B et C) destinés au pilotage des servomoteurs

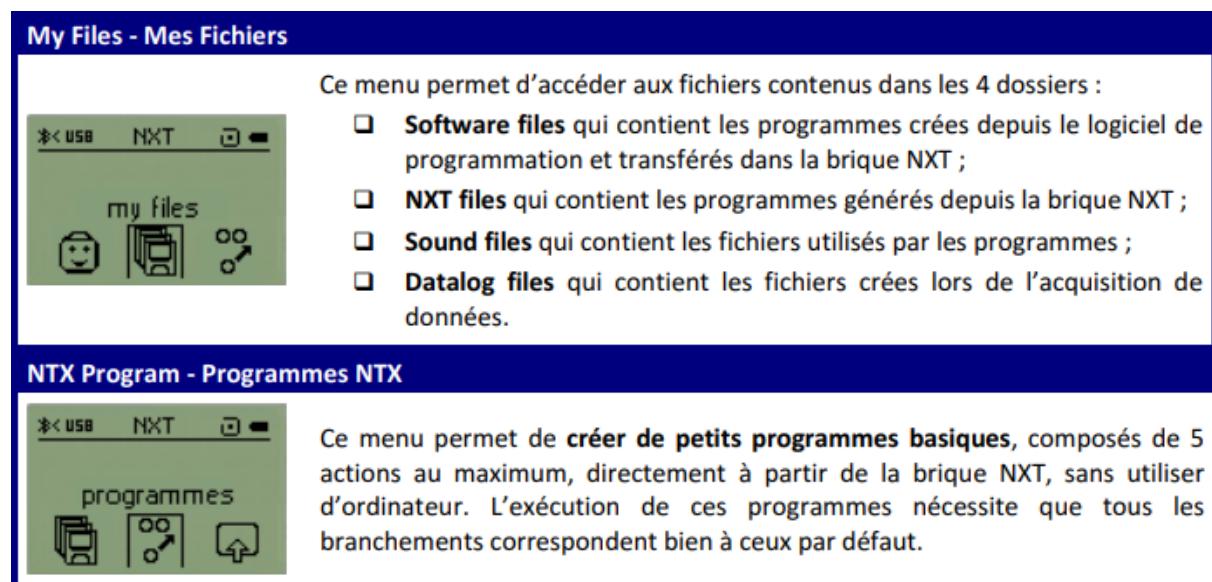
D'un écran LCD 100 × 64

D'un haut-parleur relié à contrôleur son 32 bits

D'un port USB permet la communication avec un PC

D'un contrôleur Bluetooth autorisant la communication avec un PC, une tablette numérique ou entre robots ainsi que la commande à distance depuis le logiciel de programmation ou d'une application compatible.

Une fois la brique NXT allumée, il est possible d'accéder au menu principal du NXT. En utilisant les flèches directionnelles, les catégories suivantes deviennent accessibles : « My Files », « NXT Program », « NXT Datalog », « View », « Bluetooth », « Settings » et « Try Me ».



### NTX Datalog - Journal de données NXT



Le menu du NXT Datalog (Journal des données du NXT) permet **l'acquisition de données** sans que le NXT soit connecté à l'ordinateur. L'exécution d'un programme d'acquisition de donnée du NXT crée un fichier journal, qui est enregistré dans le NXT et peut être importé sur l'ordinateur.

### View - Affichage



Ce menu permet de **visualiser les résultats de mesure** sur un des capteurs ou sur les codeurs optiques des moteurs sans avoir à créer de programme spécifique. Pour cela il suffit de sélectionner le capteur ou le moteur à tester ainsi que le port auquel il est connecté et d'observer les relevés effectués.

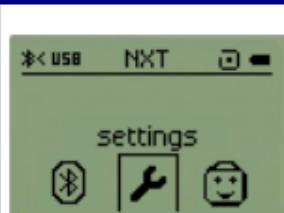
### Bluetooth



Le menu Bluetooth permet de **configurer une connexion Bluetooth** entre votre NXT et d'autres appareils, comme d'autres unités NXT, un téléphone mobile et un ordinateur.

Cette connexion peut être utilisée pour **télécharger les programmes** sans utiliser de câble USB ou pour **contrôler à distance** le robot NXT.

### Settings - Régagements



Ce menu permet de :

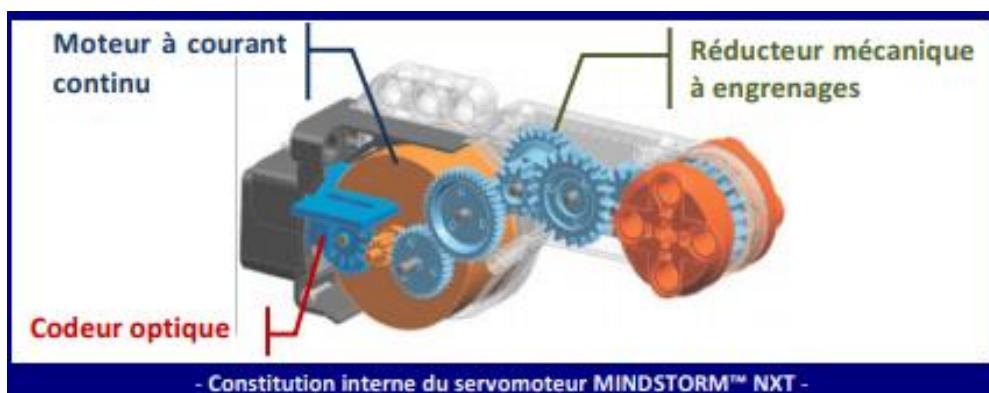
- paramétriser le volume** sonore du NXT ;
- régler la durée** avant la mise en veille ;
- afficher les informations** concernant la version de la brique NXT ;
- supprimer tous les fichiers** des 4 dossiers « Software files », « NXT files », « Sound files » et « Try me files ».

## Les servomoteurs du nxt

Le robot NXT possède trois servomoteurs. Ils permettent au robot de se déplacer ou bien de mettre en mouvement un autre élément du robot (bras, capteur ultrason mobile ...).



Chacun de ces servomoteurs est constitué d'un codeur optique qui permet de connaître la position angulaire de l'arbre moteur et donc d'en déduire le nombre de tours effectués, la vitesse de déplacement et la distance parcourue



## Présentation du capteur ultrason

Le capteur ultrason est constitué d'un émetteur et d'un récepteur ultrason. Il permet de calculer la distance le séparant d'un objet à partir de la mesure de l'intervalle de temps entre l'émission de l'onde ultrasonore et la réception de l'onde réfléchie par l'objet.



Ce capteur permet de mesurer des distances allant jusqu'à 2,5 m avec une précision de  $\pm$  3 cm. Compte tenu de cette précision, le capteur peut avoir des difficultés à détecter des objets trop proches de lui. D'autre part certains objets en tissu, en verre, incurvés, très minces ou très petits

## 5.2 Logiciels utilisés

### 5.2.1 MindStorm NXT

Il existe de nombreuses possibilités pour programmer le Mindstorms NXT : Dans le kit grand public, un logiciel de programmation graphique est fourni. Il se nomme NXT-G. Il est basé sur Labview mais ne nécessite aucune compétence en LabView. Il est très facile de démarrer à l'aide de NXT-G.

Le logiciel NXT-G n'est pas fourni dans la version éducation du kit Lego.

- Programmation textuelle :
  - Langages .NET, tels C sharp ou Visual Basic .NET, grâce à Microsoft Robotics Studio
  - Le NBC, un langage assembleur
  - Le NXC, qui est un langage proche du C. Il est gratuit et open-source.
  - RobotC est un autre langage de programmation basé sur C, développé par l'université Carnegie Mellon aux États-Unis
  - Lejos est une API open-source basée sur le langage Java
  - Urbiscript : langage de la plate-forme logicielle Urbi (*Universal Real-Time Behavior Interface*)
  - En Ada (complet ou profil Ravenscar)
  - Une bibliothèque ROS existe également pour le robot Lego
  - Une bibliothèque Matlab créée par l'université d'Aix-la-Chapelle<sup>1</sup> existe aussi
- Programmation graphique :
  - NXT-G, la version Lego de LabVIEW qui permet une prise en main aisée
  - Microsoft Robotics Studio
  - DialogOS permet de commander les robots par la voix
  - Robolab est une programmation possible du groupe National Instrument



Figure 25 NXT

## 5.2.2 JAVA

La première caractéristique, le caractère orienté objet (« OO ») et familier, fait référence à une méthode de programmation et de conception du langage et le fait qu'un programme écrit en Java ressemble assez fort à un programme écrit en C++.

Bien qu'il existe plusieurs interprétations de l'expression **orienté objet**, une idée phare dans ce type de développement est que les différents types de données doivent être directement associés avec les différentes opérations qu'on peut effectuer sur ces données. En conséquence, les données (appelées *Propriétés*) et le code les manipulant (appelé *Méthodes*) sont combinés dans une même entité appelée *Classe d'objet*. Le code devient logiquement découpé en petites entités cohérentes et devient ainsi plus simple à maintenir et plus facilement réutilisable, étant intrinsèquement modulaire.

D'autres mécanismes tels que l'*héritage* permettent d'exploiter toutes les caractéristiques d'une *Classe* précédemment écrite dans ses propres programmes sans même avoir à en connaître le fonctionnement interne — on n'en voit que l'*interface* (l'interface décrit les propriétés et les méthodes sans fournir le code associé). Java interdit la notion d'héritage depuis plusieurs classes parent sauf si elles sont des interfaces.

Dans la version 1.5 du langage ont été rajoutés les *génériques*, un mécanisme de polymorphisme semblable (mais différent) aux *templates* du langage C++ ou aux foncteurs d'OCaml. Les génériques permettent d'exprimer d'une façon plus simple et plus sûre les propriétés d'objets comme des conteneurs (listes, arbres...) : le type liste est alors considéré génériquement par rapport au type d'objet contenu dans la liste.

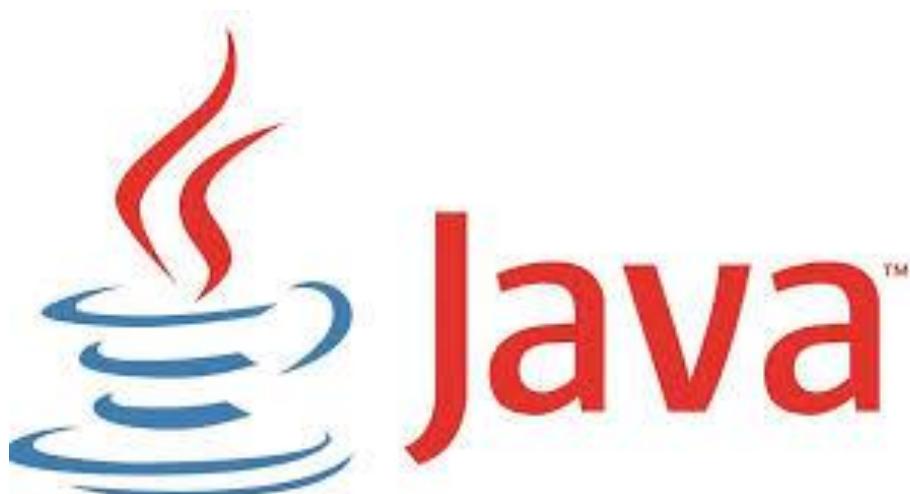


Figure 26 JAVA

### 5.2.3 Python 3

**Python** est un langage de programmation interprété, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions ; il est ainsi similaire à Perl, Ruby, Scheme, Smalltalk et Tcl.



### 5.2.4 Putty

**PuTTY** est un émulateur de terminal doublé d'un client pour les protocoles SSH, Telnet, rlogin, et TCP brut. Il permet également des connexions directes par liaison série RS-232. À l'origine disponible uniquement pour Windows, il est à présent porté sur diverses plates-formes Unix (et non-officiellement sur d'autres plates-formes).



[www.technikwiki.org](http://www.technikwiki.org)

Figure 27 PUTTY

### 5.2.5 VcnServer /VcnViewer

**VNC (Virtual Network Computing**, littéralement « informatique virtuelle en réseau ») est un système de visualisation et de contrôle de l'environnement de bureau d'un ordinateur distant. Il permet au logiciel client VNC de transmettre les informations de saisie du clavier et de la souris à l'ordinateur distant, possédant un logiciel serveur VNC à travers un réseau informatique. Il utilise le protocole RFB pour les communications.

VNC est indépendant du système d'exploitation : un client VNC installé sur n'importe quel système d'exploitation peut se connecter à un serveur VNC installé sur un système d'exploitation différent ou identique. Il existe des clients et des serveurs VNC pour la plupart des systèmes d'exploitation. Plusieurs clients peuvent se connecter en même temps à un unique serveur VNC.

Parmi les utilisations de ce protocole, on peut citer le support technique à distance, l'administration et la maintenance de systèmes ou logiciels ne permettant que des contrôles graphiques et demandant l'utilisation de la souris ou bien encore la visualisation distante d'applications diverses et variées, dans un but éducatif par exemple.

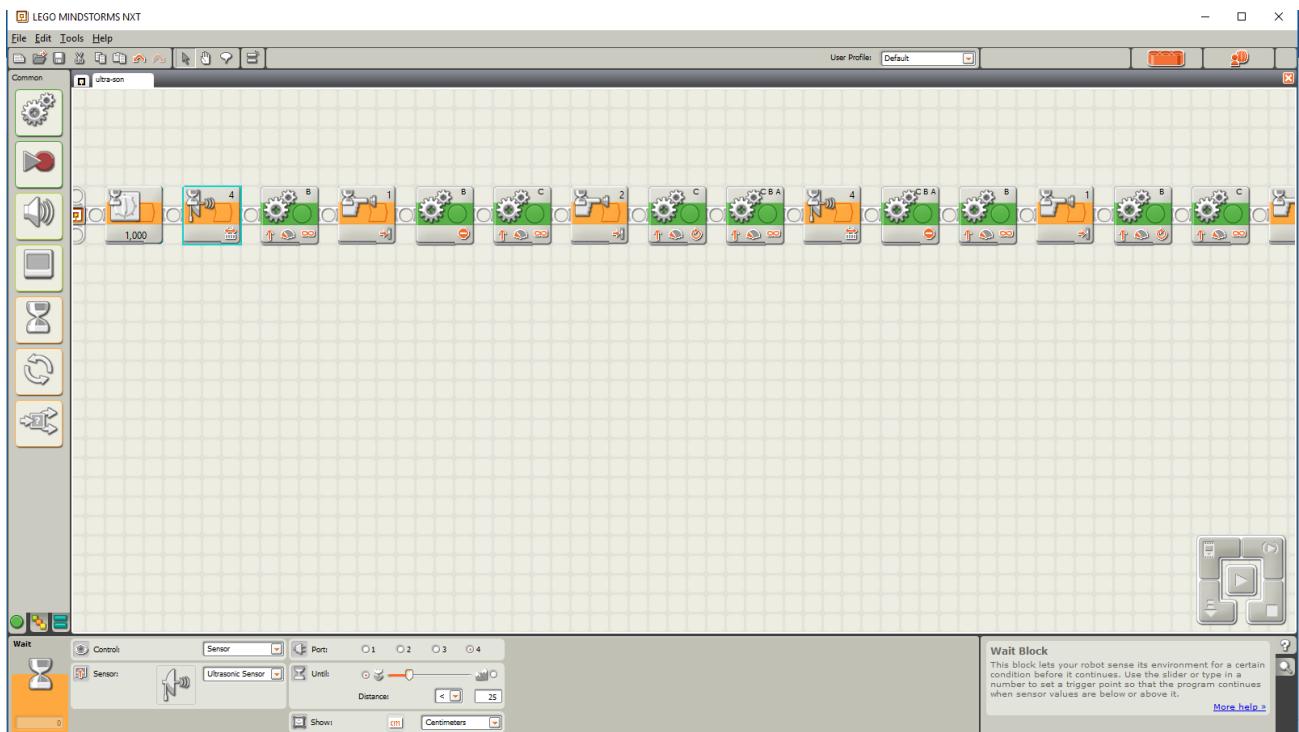


Figure 28 VNC servers

## 5.3 Programmation des différents axes d'utilisation du robot

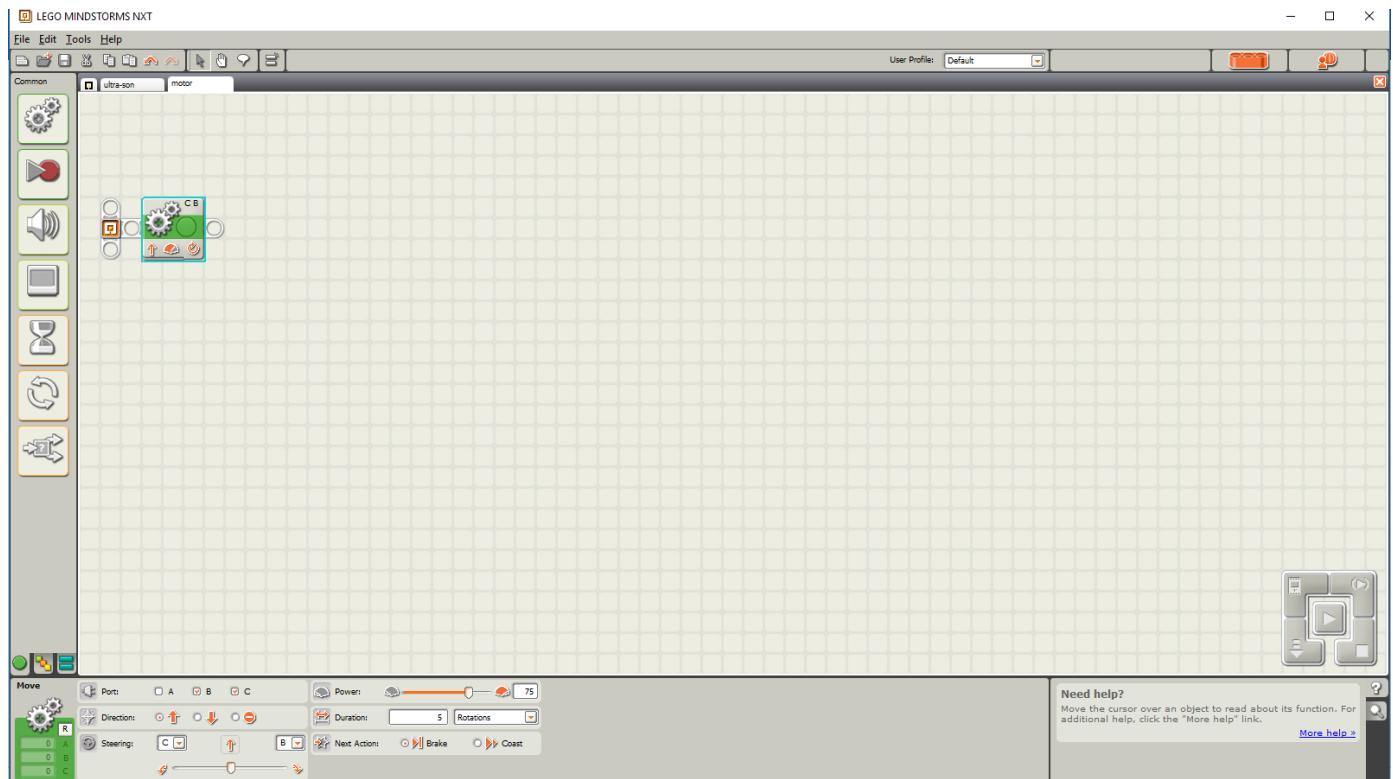
### 5.3.1 Programmation NXT

#### 1- Capteur ultra-son (NXT)



Cette séquence de commande permet au robot de détecter l'obstacle d'une distance au moins de 25 cm puis ce dernier fait une rotation de 90° à gauche avance une distance donnée puis fait une autre rotation de 90° à droite et avance jusqu'à le détection d'un autre obstacle.

## 2- Les moteurs



Notre Robot utilise deux moteurs donc suite à la distance que le robot doit parcourir on lui donne une puissance stable et la durée nécessaire les mêmes variables aussi pour programmer la rotation du robot en ajoutant l'arrêt d'un moteur et la marche d'un autre pour aider la rotation soit à gauche soit à droite.

Le 3ème moteur sera utilisé pour le mappage seulement.

## 5.3.2 Programmation de la carte Raspberry Pi III

### 1- Moteurs / Capteur de Vibration

Ecrire les mêmes commandes du NXT en utilisant le langage python3 pour guider le robot en ligne de commande python :

```

C:\Users\ThinkPad\Downloads\piratecarabv3.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
piratecarabv3.py
1 ID = '00:16:53:53:6D:C6'
2
3 # import the necessary packages
4 from shapendetector import ShapeDetector
5 import argparse
6 import imutils
7 import numpy as np
8 import cv
9 import sys
10 import tty, termios
11 import tty
12 import EV3BT
13 import nxt.locator
14 import thread
15 from nxt.motor import *
16 # This is all we need to import for the beep, but you'll need more for motors, sensors, etc.
17 from nxt.bluesock import BlueSock
18 from time import sleep
19 from nxt.sensor import *
20 import cv
21 import cv2.cv as cv
22 import time
23
24 KNOWN_WIDTH = {}
25
26 #KNOWN_DISTANCE = {}
27
28 focallengths = {}
29
30 ROTATION = 57
31 PUISSANCE = 80
32 NBESSAIS = 6
33 DD = 140
34 MAXDISTANCEVIEW = 50
35 MINSHAPCONSIDERE = 500
36 PRECISIONFORM = 0.015
37 DUREEATTENTECAM = 8
38 DV = 360
39
40 notrelacherpiece = True

```

Line 309, Column 18

Les imports nécessaires

```

C:\Users\ThinkPad\Downloads\piratecarabv3.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
piratecarabv3.py
41
42 IMAGE_PATHS = ["l80.png", "c80.png", "t80.png", "d80.png"]
43 d_shapes = ["pentagon", "circle", "triangle", "rectangle"]
44 d_colors = ["blue", "red", "green", "green"]
45 KNOWN_WIDTH["pentagon"] = 120
46 KNOWN_WIDTH["circle"] = 32
47 KNOWN_WIDTH["triangle"] = 48
48 KNOWN_WIDTH["rectangle"] = 110
49 d_cmpt = 0
50 def keeppiece(pick):
51     while notrelacherpiece:
52         pick.turn(1, 69, False)
53 def runinstruction(pick):
54     #keeppiece(pick)
55     thread.start_new_thread(keeppiece,(pick,))
56 def play(brick):
57     brick.play_tone_and_wait(440, 1000)
58 def getch():
59     print "Quelle action voulez vous faire?"
60     print "1 : Aller l'ile cerculaire rouge"
61     print "2 : Aller l'ile triangulaire rouge"
62     print "3 : Aller l'ile cerculaire jaune"
63     print "4 : Aller l'ile carree jaune"
64     print "5 : Aller l'ile polygone verte"
65     print "6 : Aller l'ile triangulaire verte"
66     print "7 : Aller l'ile carree verte"
67     print "8 : Aller s'approvisionner a l'ile polygone bleu"
68     print "9 : Lancer le calibrage"
69     print "e : Marcher Avant"
70     print "d : Reculer"
71     print "f : Tourner a droite"
72     print "s : Tourner a gauche"
73     print "q : Quitter"
74     print "c : Pour capturer la piece metalique"
75     print "r : Pour relacher la piece metalique"
76
77     fd = sys.stdin.fileno()
78     old_settings = termios.tcgetattr(fd)
79     try:
80         tty.setraw(fd)

```

Line 58, Column 1

Menu

```
C:\Users\ThinkPad\Downloads\piratecarabv3.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
piratecarabv3.py

201
202
203 sock = BlueSock(ID)
204 if sock :
205     brick = sock.connect()
206     #left = nxt.Motor(brick, PORT_A)
207     #right = nxt.Motor(brick, PORT_B)
208     #pick = nxt.Motor(brick, PORT_C)
209     #both = nxt.SynchronizedMotors(left, right, 0)
210     #leftboth = nxt.SynchronizedMotors(left, right, 100)
211     #rightboth = nxt.SynchronizedMotors(right, left, 100)
212     print "Im here"
213     s = EV3BT.encodeMessage(EV3BT.MessageType.Text, 'abc', 'Eat responsibly')
214     print(EV3BT.printMessage(s))
215     EV3.write(s)
216     time.sleep(1)
217     EV3.close()
218
219
220 while ch != 'q':
221     compteur = 0
222     ch = getch()
223     print "Execution de la tache demander"
224     if ch == 'e':
225         both.turn(PIUSSANCE, DD, False)
226     elif ch == 'd':
227         both.turn(-1*PIUSSANCE, DD, False)
228     elif ch == 's':
229         leftboth.turn(PIUSSANCE, ROTATION, False)
230     elif ch == 'f':
231         rightboth.turn(PIUSSANCE, ROTATION, False)
232     elif ch == 'r':
233         notrelacherpiece = True
234         runinstruction(pick)
235     elif ch == 'c':
236         notrelacherpiece = False
237     elif ch == '9':
238         frame = getImageFromCam()
239         shapes = ["circle", "pentagon", "rectangle", "triangle"]
240         colors = [ "green", "yellow"]

Line 58, Column 1
```

Rotation et puissance

```
C:\Users\ThinkPad\Downloads\piratecarabv3.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
piratecarabv3.py

241
242
243     for color in colors :
244         tmp = 0
245         for shape in shapes :
246             if tmp > 0 :
247                 break
248             val = getShapeColorFromImage(frame, shape, color)
249             if val is not None:
250                 tmp = 1
251                 print 'quel est la distance du %s %s' % (shape, color)
252                 data = input("Entrer un numero: ")
253                 #KNOWN_DISTANCE[shape] = data
254                 width = input("Quelle est la superficie de la forme: ")
255                 #KNOWN_WIDTH[shape] = width
256                 focallengths[shape] = (cv2.contourArea(val) * data) / KNOWN_WIDTH[shape]
257             print "Calibrage terminer"
258         else :
259             shape = "pentagon"
260             color = "blue"
261             while compteur < NBEAIS :
262                 if ch == '1' :
263                     shape = "circle"
264                     color = "red"
265                 elif ch == '2':
266                     shape = "triangle"
267                     color = "red"
268                 elif ch == '3':
269                     shape = "circle"
270                     color = "yellow"
271                 elif ch == '4':
272                     shape = "rectangle"
273                     color = "yellow"
274                 elif ch == '5' :
275                     shape = "pentagon"
276                     color = "green"
277                 elif ch == '6' :
278                     shape = "triangle"
279                     color = "green"
280                 elif ch == '7':
281                     shape = "rectangle"

Line 58, Column 1
```

Définition de paramètre



```
C:\Users\ThinkPad\Downloads\piratecarraigv3.py • - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
piratecarraigv3.py

271         shape = "rectangle"
272         color = "yellow"
273     elif ch == '5' :
274         shape = "pentagon"
275         color = "green"
276     elif ch == '6' :
277         shape = "triangle"
278         color = "green"
279     elif ch == '7':
280         shape = "rectangle"
281         color = "green"
282     elif ch == '8' and shape == "pentagon":
283         shape == "pentagon"
284         color = "blue"
285     # Capture frame-by-frame
286     frame = getImageFromCam()
287     val = getShapeColorFromImage(frame, shape, color)
288     if val is not None:
289         play(brick)
290         distance = getDistanceFromCam(val, shape)
291         print "distance = %d" % (distance)
292         if distance > MAXDISTANCEVIEW :
293             allera = distance - (MAXDISTANCEVIEW - 1)
294             both.turn(PIUSSANCE, 20*distance, False)
295             break #compteur = 0
296         else :
297             both.turn(PIUSSANCE, 20*distance , False)
298             break
299         else :
300             compteur = compteur + 1
301             leftboth.turn(PIUSSANCE, ROTATION, False)
302             time.sleep(DUREEATTENTECAM)
303     sock.close()

Line 58, Column 1
```

La condition d'arrêt

## 2- Caméra

En plus de la lecture de tags RFID notre projet consiste à lire l'ancienne solution les codes-barres donc pour cela on utilise un camera qui prendra une photo de chaque code-barres détecté puis l'envoyé à la raspberry pour traitement en rendant l'image texte (python 3).

```

1 ID = '00:16:53:53:6D:C6'
2
3 # import the necessary packages
4 from shapendetector import ShapeDetector
5 import argparse
6 import imutils
7 import numpy as np
8 import cv2
9 import sys
10 import tty, termios
11 import tty
12 import EV3BT
13 import nxt.locator
14 import thread
15 from nxt.motor import *
16 # This is all we need to import for the beep, but you'll need more for motors, sensors, etc.
17 from nxt.bluesock import BlueSock
18 from time import sleep
19 from nxt.sensor import *
20 import cv
21 import cv2.cv as cv
22 import time
23
24 KNOWN_WIDTH = {}
25
26 #KNOWN_DISTANCE = {}
27
28 focalLengths = {}
29
30 ROTATION = 57
31 PUISSANCE = 80
32 NBEAIS = 6
33 DD = 140
34 MAXDISTANCEVIEW = 50
35 MINSHAPCONSIDERE = 500
36 PRECISIONFORM = 0.015
37 DUREEATTENTECAM = 8
38 DV = 360
39
40 notrelacherpiece = True

```

Les imports nécessaires

```

41
42 IMAGE_PATHS = ["l80.png", "c80.png", "t80.png", "d80.png"]
43 d_shapes = ["pentagon", "circle", "triangle", "rectangle"]
44 d_colors = ["blue", "red", "green", "green"]
45 KNOWN_WIDTH["pentagon"] = 120
46 KNOWN_WIDTH["circle"] = 32
47 KNOWN_WIDTH["triangle"] = 48
48 KNOWN_WIDTH["rectangle"] = 110
49 d_cmpt = 0
50 def keeppiece(pick):
51     while notrelacherpiece :
52         pick.turn(1, 69, False)
53 def runinstruction(pick):
54     #keeppiece(pick)
55     thread.start_new_thread(keeppiece,(pick,))
56 def play(brick):
57     brick.play_tone_and_wait(440, 1000)
58 def getch() :
59     print "Quelle action voulez vous faire?"
60     print "1 : Aller l'ile cerculaire rouge"
61     print "2 : Aller l'ile triangulaire rouge"
62     print "3 : Aller l'ile cerculaire jaune"
63     print "4 : Aller l'ile caree jaune"
64     print "5 : Aller l'ile polygone verte"
65     print "6 : Aller l'ile triangulaire verte"
66     print "7 : Aller l'ile carree verte"
67     print "8 : Aller s'approvisionner a l'ile polygone bleu"
68     print "9 : Lancer le calibrage"
69     print "e : Marcher Avant"
70     print "d : Reculer"
71     print "f : Tourner a droite"
72     print "s : Tourner a gauche"
73     print "q : Quitter"
74     print "c : Pour capturer la piece metalique"
75     print "r : Pour relacher la piece metalique"
76
77     fd = sys.stdin.fileno()
78     old_settings = termios.tcgetattr(fd)
79     try:
80         tty.setraw(fd)

```

Menu

```

C:\Users\ThinkPad\Downloads\piratecarabv3.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
piratecarabv3.py
81     ch = sys.stdin.read(1)
82     finally:
83         termios.tcsetattr(fd, termios.TCSADRAIN, old_settings)
84     return ch
85
86
87 ch = ''
88 print "Ready"
89
90 #fin all the 'yellow shapes in the image'
91 lowerRed2 = np.array([160, 100, 100],np.uint8)
92 upperRed2 = np.array([179, 255, 255],np.uint8)
93 lowerRed1 = np.array([0, 100, 100],np.uint8)
94 upperRed1 = np.array([10, 255, 255],np.uint8)
95
96 #fin all the 'blue shapes in the image'
97 lowerBlue1 = np.array([110,150,150],np.uint8)
98 upperBlue1 = np.array([130,255,255],np.uint8)
99 lowerBlue2 = np.array([100,150,0],np.uint8)
100 upperBlue2 = np.array([140,255,255],np.uint8)
101
102 #fin all the 'yellow shapes in the image'
103 lowerYellow1 = np.array([20, 100, 100],np.uint8)
104 upperYellow1 = np.array([32, 255, 255],np.uint8)
105 lowerYellow2 = np.array([15, 100, 100],np.uint8)
106 upperYellow2 = np.array([20, 255, 255],np.uint8)
107
108 #fin all the 'yellow shapes in the image'
109 lowerGreen1 = np.array([40, 100, 50],np.uint8)
110 upperGreen1 = np.array([80, 255, 255],np.uint8)
111 lowerGreen2 = np.array([45,100,100],np.uint8)
112 upperGreen2 = np.array([75,255,255],np.uint8)
113
114 def getShapeColorFromImage(image, form, color) :
115     #appliquer filtre pour enlever les parasites de l image
116     median = cv2.medianBlur(image,9)
117     hsv_img = cv2.cvtColor(median,cv2.COLOR_BGR2HSV)
118
119     shapeMask1 = cv2.inRange(hsv_img, lowerBlue1, upperBlue1)
120     shapeMask2 = cv2.inRange(hsv_img, lowerBlue2, upperBlue2)

```

Line 58, Column 1

## Les paramètres de couleur

```

C:\Users\ThinkPad\Downloads\piratecarabv3.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
piratecarabv3.py
121     if color == "red":
122         shapeMask1 = cv2.inRange(hsv_img, lowerRed1, upperRed1)
123         shapeMask2 = cv2.inRange(hsv_img, lowerRed2, upperRed2)
124     elif color == "yellow":
125         shapeMask1 = cv2.inRange(hsv_img, lowerYellow1, upperYellow1)
126         shapeMask2 = cv2.inRange(hsv_img, lowerYellow2, upperYellow2)
127     elif color == "green":
128         shapeMask1 = cv2.inRange(hsv_img, lowerGreen1, upperGreen1)
129         shapeMask2 = cv2.inRange(hsv_img, lowerGreen2, upperGreen2)
130     else:
131         shapeMask1 = cv2.inRange(hsv_img, lowerBlue1, upperBlue1)
132         shapeMask2 = cv2.inRange(hsv_img, lowerBlue2, upperBlue2)
133
134     shapeMask = cv2.addWeighted(shapeMask1, 1.0, shapeMask2, 1.0, 0.0)
135
136     # find contours in the thresholded image and initialize the
137     # shape detector
138     cnts,hierarchy = cv2.findContours(shapeMask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_TC89_KCOS)
139
140     print "I found %d shapes" % (len(cnts))
141
142     #cnts = cnts[0] if imutils.is_cv2() else cnts[1]
143     sd = ShapeDetector()
144
145     # loop over the contours
146     for c in cnts:
147         ca = cv2.contourArea(c)
148         if ca > MINSHAPECONSIDERE:
149             print "ca = %d" % (ca)
150
151             shape = sd.detect(c)
152             print shape
153             if form == shape :
154                 peri = cv2.arcLength(c, True)
155                 approx = cv2.approxPolyDP(c, PRECISIONFORM * peri, True)
156                 return approx
157
158     for shape in d_shapes :
159         image = cv2.imread(IMAGE_PATHS[d_cmpt])
160         val = getShapeColorFromImage(image, shape, d_colors[d_cmpt])

```

Line 58, Column 1

## Les paramètres de couleur (2)

### 5.3.3 Programmation du mappage

#### INSTALLATION

Pour compiler et lancer les programmes, il faut installer le logiciel NXJ leJOS : <http://www.lejos.org/nxt/nxj/tutorial/Preliminaries/GettingStarted.htm>

Puis installer le plugin Eclipse : <http://www.lejos.org/nxt/nxj/tutorial/Preliminaries/UsingEclipse.htm> (sur Windows, utiliser la version 32 bits d'Eclipse, la 64 bits peut poser problème)

Vous pouvez ensuite importer les programmes du dossier courant dans Eclipse.

Le sigle [PC] signifie que le programme fonctionne du coté PC. Le sigle [NXT] signifie que le programme fonctionne du coté robot.

Les projets principaux sont :

- Scanner Runner [NXT]
- Bluetooth Mapper [PC]

(Un mini Javadoc est disponible dans leur dossier doc)

L'adresse du robot dans main doit être changé par l'adresse de votre robot.

#### FORME DU ROBOT

Forme du robot pour "Scanner" :

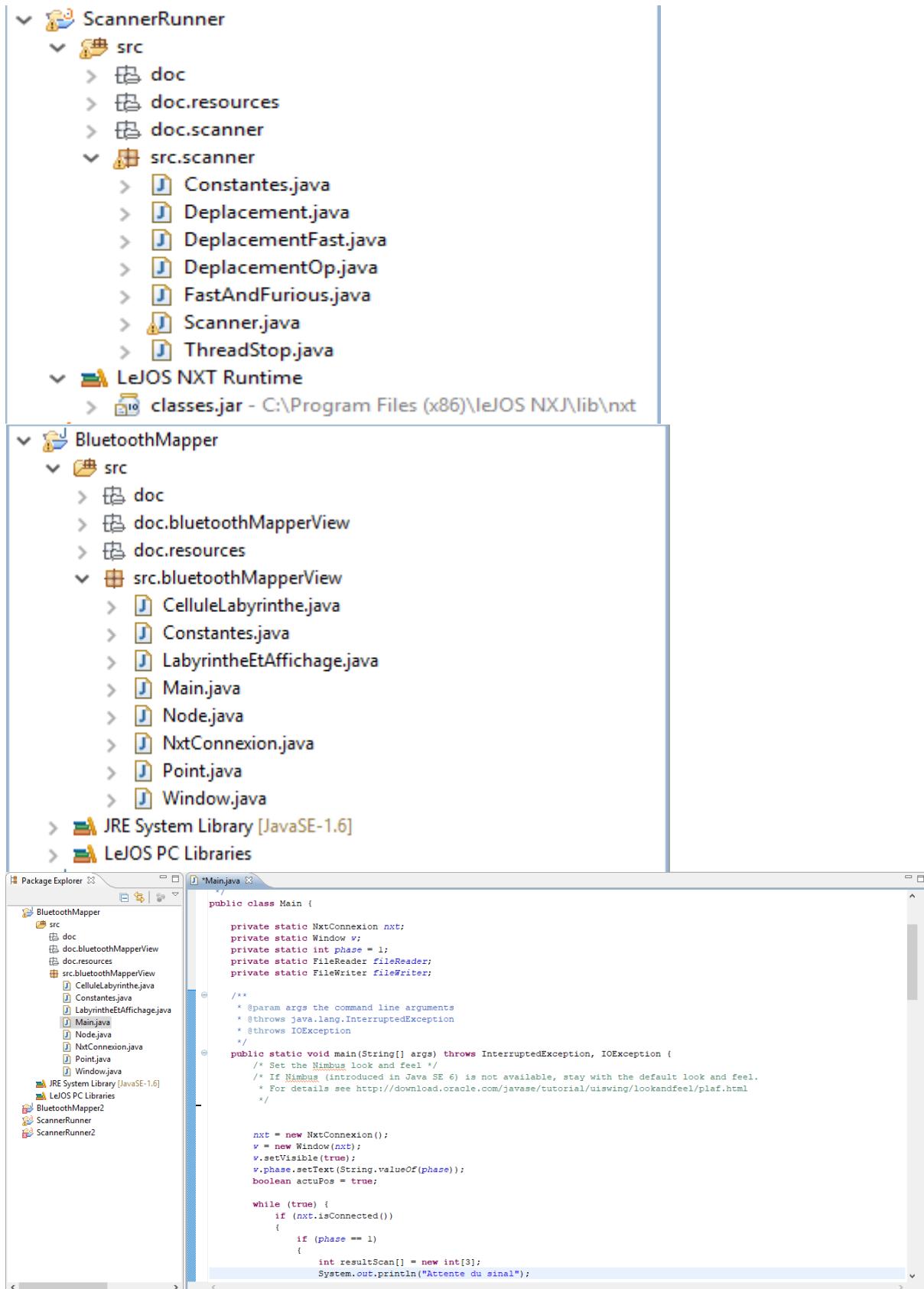
- Moteur droit / port A
- Moteur gauche / port B
- Moteur tête / port C
- Capteur ultra son / port 1

Peut être modifié dans la classe "Constantes".

#### BLUETOOTH

Vous devez disposer d'un dongle Bluetooth sur votre ordinateur !

Si vous avez des problèmes pour trouver vos NXT en Bluetooth, la commande suivante peut être très utile : nxjbrowse -b -n "\*" Elle va chercher tous les NXT présent et les ajouter dans le cache de NXJ, ça résout en général les problèmes. Le programme nxjbrowse se lancera également, il peut vous être utile si vous voulez uploader un fichier quelconque sur le robot.



The image consists of three vertically stacked screenshots of the Eclipse IDE interface, each showing a Java code editor and a Package Explorer view.

- NxtConnexion.java:** This screenshot shows the code for connecting to an NXT brick via Bluetooth. The code initializes a connection, handles log events, and includes a main method for testing.

```

package src.bluetoothMapperView;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;
import java.util.Scanner;

public class NxtConnexion {
    private static NXTConnector conn;
    private static DataOutputStream dos;
    private static InputStream dis;
    private boolean connexion = false;

    /**
     * Initialisation de la connexion à Hydra,
     * ouverture des streams d'écriture et de lecture
     */
    public void connexion() {
        conn = new NXTConnector();
        if (!connexion) {
            conn.addLogListener(new NXTCommLogListener());
            public void logEvent(String message) {
                System.out.println("BTISend Log.listener: "+message);
            }
        }
    }
}

```

- Scanner.java:** This screenshot shows the code for a scanner robot. It initializes ultrasonic and light sensors, manages movement, and handles communication over BT. It includes a main method for testing.

```

package src.scanner;

import static src.scanner.Constantes.*;

/**
 * Cartographie le labyrinthe
 */
public class Scanner {

    private static ThreadStop emergencyStop = new ThreadStop();

    // Initialisation des capteurs
    private static UltrasonicSensor sonic = new UltrasonicSensor(SensorPort.S1);
    private static LightSensor lightLeft = new LightSensor(SensorPort.S2, true);
    private static LightSensor lightRight = new LightSensor(SensorPort.S3, true);

    // Gestion des déplacements
    private static DeplacementOp placement = new DeplacementOp();

    // Pour la communication
    private static BTConnection btc;
    private static DataInputStream receive;
    private static DataOutputStream send;

    //private static File son = new File("cri_agonie.wav");
    private static boolean terminé = false; // Cartographie terminée ?
    private static boolean ligneArrive = false; // Est-ce qu'on a trouvé la ligne d'arrivée ?

    public static void main(String[] args) throws Exception {
    }
}

```

- FastAndFurious.java:** This screenshot shows the code for a fast and furious scanner. It initializes ultrasonic and light sensors, manages movement, and handles communication over BT. It includes a main method for testing.

```

package src.scanner;

import static src.scanner.Constantes.*;

/**
 * Réalise une séquence de déplacement et l'effectue
 * en utilisant des déplacements rapides
 */
public class FastAndFurious {

    private static ThreadStop emergencyStop = new ThreadStop();

    // Initialisation des capteurs
    private static UltrasonicSensor sonicRight = new UltrasonicSensor(SensorPort.S1);
    private static UltrasonicSensor sonicLeft = new UltrasonicSensor(SensorPort.S4);
    private static LightSensor lightLeft = new LightSensor(SensorPort.S2, true);
    private static LightSensor lightRight = new LightSensor(SensorPort.S3, true);

    // Gestion des déplacements
    private static DeplacementFast placement = new DeplacementFast();

    // Pour la communication
    private static BTConnection btc;
    private static DataInputStream receive;

    private static int distanceMin = 6;

    public static void main(String[] args) throws IOException, InterruptedException {
        emergencyStop.setDaemon(true);
        emergencyStop.start();
    }
}

```

The screenshot shows the Eclipse IDE interface. On the left, the 'Package Explorer' view displays a project structure with packages like 'BluetoothMapper', 'ScannerRunner', and 'src'. The 'src' package contains several Java files: 'Constantes.java', 'Deplacement.java', 'DeplacementOp.java', 'DeplacementFast.java' (which is currently selected), 'FastAndFurious.java', 'Scanner.java', and 'ThreadStop.java'. It also includes 'LeOS NXT Runtime' and 'ScannerRunner2'. On the right, the 'DeplacementFast.java' code editor window is open, showing the following Java code:

```

/*
 * TER Lego 2015 - Université Paul Sabatier
 */
package src.scanner;

import static src.scanner.Constantes.*;

/**
 * Fourni des déplacements et redressements optimisés pour
 * la vitesse. Pour plus de documentation sur les méthodes,
 * voir la classe DeplacementOp
 */
public class DeplacementFast extends DifferentialPilot {

    public DeplacementFast() {
        super(wheelDiameter, trackWidth, leftMotor, rightMotor);
        super(56, 120, Motor.B, Motor.A);
    }

    public void fastHairpinTurnTo(int cote) {
        setTravelSpeed(340);
        if (cote == LEFT) {
            arc(0, 90);
        } else if (cote == RIGHT) {
            arc(0, -90);
        }
    }

    public void fastRecoveryAngleWithLine() {
        LightSensor lightLeft = new LightSensor(SensorPort.S2, true);
        LightSensor lightRight = new LightSensor(SensorPort.S3, true);
        if (lightLeft.getNormalizedLightValue() > lightLimit) {
            Motor.B.setSpeed(180);
        }
    }
}

```

## Conclusion

Ce cinquième et dernier chapitre vient clore le cycle de développement de la solution robotique. Les résultats auxquels a abouti le projet en termes de réalisation ont été présentés durant les différentes subdivisions de ce chapitre.

Tous les axes majeurs et les objectifs présentés en début de mémoire ont été réalisés tout en restant dans la simplicité et l'optimisation.

## Conclusion générale

Ce projet de fin d'études réalisé au sein du labo RITM vient en réponse aux besoins croissants des entreprises de production en matière de smart-warehousing, et s'inscrit dans le cadre de la stratégie d'ouverture sur le domaine de l'informatique logistique aux horizons 2019. Il s'agit, en effet, d'amorcer cette dynamique en lançant le projet de création d'une solution de smart-warehousing dédiée aux gestionnaires de stocks. Cette solution a pour objectif la simplification fonctionnelle et technique des processus d'entreposage.

Le travail a donc consisté en la réalisation de solutions plus souples et expressives et leur réutilisation et restitution, dans le but d'afficher une nouvelle démarche visant à munir les entreprises d'outils de pilotage de leur activité et à faciliter la gestion de leurs stocks. Il s'agissait dans une vision finale de réaliser une tracabilité sous forme graphique et interactive.

Ainsi, nous avons été amenés durant ce projet à suivre les étapes de développement d'un projet logistique. En effet, nous avons commencé par effectuer une étude de marché afin de prendre connaissance des plateformes de smart-warehousing déjà existantes, ayant très vite aidé à spécifier quelques besoins techniques. L'étude suivante nous a également permis de recueillir les besoins techniques du projet, en parallèle avec l'établissement des exigences fonctionnelles concernant les données. Nous sommes ensuite passés à la modélisation des différents composants du système avant d'attaquer la réalisation de la structure de lecture puis de la partie robotique.

Au terme de ce travail, les objectifs fixés dans notre cahier de charges ont été atteints en mettant en place une solution résolvant les problèmes de manque de visibilité et de traçabilité. Cependant, et bien qu'il s'agisse d'une solution complète, elle reste soumise au processus de prototypage et peut être en proie à la nécessité de recourir à des améliorations. Ainsi, nous pouvons proposer en perspectives l'usage de la solution idéale préalablement présentée, le recourt à des éléments matériels plus évolués, ainsi que l'implémentation logicielle de la solution.

## Webographie

<https://lewebpedagogique.com/isneiffel/files/2017/05/Pr%C3%A9sentation-du-Robot-NXT.pdf>

[www.lego.com](http://www.lego.com)

<https://www.realvnc.com/fr/connect/docs/get-connected.html>

<http://benja135.xyz/blog/cartographie-et-sortie-labyrinthe>

<https://www.putty.org/>

<http://www.est-uh2c.ac.ma/laboritm/>

<https://github.com/benja135/bluetooth-mapper-v2>

<https://www.raspberrypi-spy.co.uk/2018/02/rc522-rfid-tag-read-raspberry-pi/>

<https://github.com/ondryaso/pi-rc522/blob/master/README.md>

<https://docs.google.com/document/d/1FjbQOsFSr8OsqqaLW5gKd0JqbsWJFrlltR2TXDNOta8/edit>

<https://stackoverflow.com/questions/54847012/raspberry-pi-3b-and-rfid-rc522-python-typeerror>

<https://github.com/Soufianelalaoui/gestion-d-entrep-t/edit/master/README.md>

<https://stackoverflow.com/questions/54847012/raspberry-pi-3b-and-rfid-rc522-python-typeerror>

<http://www.lejos.org/nxt/nxj/tutorial/Preliminaries/UsingEclipse.htm>

<https://les-enovateurs.com/rfid-raspberry-pi-code/>

<https://www.raspberrypi-spy.co.uk/2018/02/rc522-rfid-tag-read-raspberry-pi/>

<https://github.com/lthiery/SPI-Py>

<https://www.raspberrypi.org/documentation/remote-access/vnc/>

<https://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase7-521261.html>

<https://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase6-419409.html>

<https://sourceforge.net/projects/nxt.lejos.p/>

<https://sourceforge.net/projects/nxt.lejos.p/postdownload>

<http://www.lejos.org/nxj-downloads.php>

[https://fr.wikipedia.org/wiki/Lego\\_Mindstorms\\_NXT](https://fr.wikipedia.org/wiki/Lego_Mindstorms_NXT)

<https://www.lego.com/fr-fr/mindstorms/downloads/nxt-software-download>

<https://www.youtube.com/watch?v=x2AG79TMxI8>

[https://drive.google.com/file/d/1sBSFaRjBwaLaC-SQ9AbSIM4d\\_hmgdFvV/view](https://drive.google.com/file/d/1sBSFaRjBwaLaC-SQ9AbSIM4d_hmgdFvV/view)

<https://www.theengineeringprojects.com/2015/08/interfacing-rfid-rc522-arduino.html>

<https://forums.futura-sciences.com/electronique/688746-isis-rfid.html>

<https://www.dropbox.com/s/1esce6it8zy0ggq/rfid.zip?dl=0>

<https://www.robot-advance.com/art-capteur-ultrasons-lego-mindstorms-education-nxt-587.htm>

<https://www.jumia.ma/havic-pc-webcam-21234893.html>

[https://www.gotronic.fr/art-module-rfid-13-56-mhz-tag-rc522-25651.htm - complte\\_desc](https://www.gotronic.fr/art-module-rfid-13-56-mhz-tag-rc522-25651.htm - complte_desc)

<http://helloraspberrypi.blogspot.com/2013/11/remote-control-raspberry-pi-from.html>

<https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

<http://helloraspberrypi.blogspot.com/2013/11/install-tightvnc-server-on-raspberry-pi.html>

<https://www.logitech.fr/fr-fr/product/conferencecam-ptz-pro2>

<https://selecthub.com/warehouse-management/smart-warehouse-systems/>

<https://www.wikipedia.org/>

<https://github.com/pimylifeup/MFRC522-python>

<https://github.com/EspaceRaspberryFrancais/RFID-RC522>

<https://www.instructables.com/>

<https://github.com/mgxw/MFRC522-python/blob/master/MFRC522.py>