

BORDEAU Raphaël - DE LA FUENTE Léo - GOURJON Amélie  
MESSARA Errikos - PAGNY Louis

# Projet de fin d'étude : Guidage d'un bras robotisé par vision



<b>Introduction</b>	<b>1</b>
<b>Cahier des Charges</b>	<b>3</b>
<b>Présentation du robot</b>	<b>4</b>
<b>Architecture Logicielle</b>	<b>5</b>
Lors de ce projet, nous nous sommes répartis en trois équipes : Amélie et Raphaël ont travaillé sur le Robot, Louis sur la Computer Vision, et Errikos et Léo ont réalisé l'interface utilisateur.	5
Robot	5
Premier programme	5
Workshop	5
Machine Learning	6
Computer Vision	7
Détection de croisement	7
Remove_shadow	8
Interface utilisateur	8
<b>Gestion du projet : Management agile</b>	<b>10</b>
Organisation de l'équipe	10
Méthode Scrum	10
Organisation des tâches	10
Déroulement d'un sprint	11
Début de sprint	11
Durant le sprint	11
Fin du sprint	12
Avantages et inconvénients de la méthode :	12
Avantages	12
Inconvénients	12
Utilisation de Git	12
<b>Liste des rendus</b>	<b>12</b>
<b>Conclusion</b>	<b>13</b>

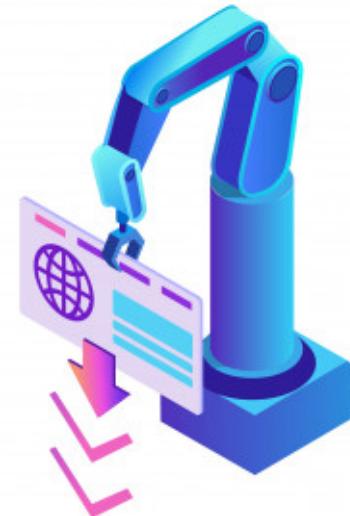
# Introduction

L'entreprise KP1, représentée par M. Vincent Vermorel a souhaité proposer un projet intitulé : « Guidage d'un bras robotisé par vision ». Cette entreprise du BTP est spécialisée dans les systèmes constructifs préfabriqués. Notre projet s'inscrit dans le processus de création de dalles précontraintes. Dans sa volonté d'automatisation et de faciliter le travail des ouvriers, l'entreprise cherche à montrer la faisabilité d'un projet de bras robotique permettant de réaliser les ligatures des fers dans ces dalles. Ce projet ayant pour but de jauger de la faisabilité du guidage du robot et de la détection de ligne ou d'objet, nous avons donc travaillé avec un robot éducatif Niryo One et son kit de vision.

Deux objectifs principaux étaient rattachés à ce projet. Tout d'abord, réussir à détecter des croisements et venir effectuer des opérations sur ceux-ci. Dans un second temps, M. Vermorel souhaitait mettre en place une méthode agile pour gérer ce projet et voir les résultats de cette gestion pour pouvoir la mettre en œuvre ultérieurement au sein de l'entreprise.

Afin de réaliser ce premier objectif, nous avons utilisé une partie de l'API fourni par le robot Niryo One sur Python, ainsi que les bibliothèques OpenCV pour le traitement d'image, Numpy pour le calcul matriciel, PyQt pour l'interface graphique et TensorFlow pour la partie d'intelligence artificielle.

En ce qui concerne la méthode agile, nous avons géré notre projet avec des éléments de la méthode SCRUM. Pour trier et visualiser les tâches nous avons réalisé un Tableau de post-it, qui s'est finalement transformé en digital avec l'outil en ligne Trello. Nous avons utilisé github pour échanger et versioner notre programme. Pour ce qui est de la communication nous n'avons pas eu besoin de logiciel spécifique puisque nous avons travaillé exclusivement en présentiel.



# Cahier des Charges

Au début du projet, notre client nous a dicté un cahier des charges assez strict qui regroupe une partie plus technique des réalisations qu'il souhaitait voir fonctionner sur le robot et une partie plus orientée gestion de projet. Voici le cahier des charges fixés au tout début du projet :

- Prise en main du robot, choix d'un langage de programmation
- Simuler les croisements des fers dans la dalle de béton sur l'espace de travail du robot
- Déetecter les croisements des fers dans l'espace de travail et réaliser un action avec le robot
- Créer une interface homme-machine, facile d'utilisation, afin de simuler le travail qu'aurait à faire un opérateur de la chaîne de production
- Mettre en place une routine de gestion de projet suivant la méthode Scrum et les principes du management agile

Cependant, ayant une bonne efficacité dans les premières semaines du projet et la méthode scrum portant bien ses fruits. La nécessité de complexifier le programme et de le rendre plus robuste s'est alors faite assez naturellement. Le projet étant large, il n'a pas été difficile de rajouter de nouveaux points dans le cahier des charges. Nous l'avons alors fait évoluer au milieu du projet en y ajoutant les point suivants :

- Rendre la détection de croisement plus robuste en étant capable de détecter plus de deux droit qui se croisent, détection des droites ni horizontale ni verticale
- Crédation d'un magasin d'objet avec une détection de la position des objets et détection du type d'objet avec l'intelligence artificielle

# Présentation du robot

Dans le cadre de ce projet, KP1 nous a fourni un bras robotisé éducatif. Ce bras est un bras développé par la jeune entreprise Nyrio, il est rapide à prendre en main et facile d'utilisation. Le Nyrio one est un bras robot 6 axes (cf Figure Description du bras). Il est équipé d'une raspberry pi 3 à laquelle on peut se connecter pour programmer des routines. Pour assurer la communication avec le robot, nous avons la possibilité de nous connecter au robot soit par wifi à l'adresse IP locale "10.10.10.10", soit par câble ethernet à l'adresse "169.254.200.200".



Le robot est fourni avec 3 pinces qui sont plus ou moins adaptatives à la forme des objets à attraper. Pour ce projet nous avons utilisé uniquement la deuxième pince qui est la pince large.

Pour l'application de notre projet nous avons aussi eu accès au kit vision fourni par Nyrio avec : un espace de travail, la caméra et quelques objets de formes simple et de différentes couleurs.



L'entreprise propose l'application Niryo One Studio pour prendre en main ce robot. Cette application permet alors de contrôler facilement toutes les fonctionnalités du robot, réaliser des workspaces (espace de travail pour le robot), ainsi que coder des algorithmes avec un interface en forme de bloc. Le robot peut aussi être programmé avec des langages différents tels que python,C++, MATLAB,... grâce à des API fournies. Nous avons choisi d'utiliser le langage python car il permet de réaliser des interfaces utilisateurs, intègre facilement les bibliothèques OpenCV et Tensor Flow, et ne nécessite pas de compilation avant son exécution. De plus, ce langage est facile à prendre en main pour les personnes qui reprendront notre travail par la suite.

# Architecture Logicielle

Lors de ce projet, nous nous sommes répartis en trois équipes : Amélie et Raphaël ont travaillé sur le Robot, Louis sur la Computer Vision, et Errikos et Léo ont réalisé l'interface utilisateur.

## Robot

Dans un premier temps, nous avons eu une phase de découverte du robot et de son API python. Cette phase a été très importante car elle nous a permis d'explorer les fonctions que nous pourrions réutiliser par la suite et aussi de créer des routines très simples comme celle d'applaudissement qui s'exécute à l'appui sur le bouton du robot.

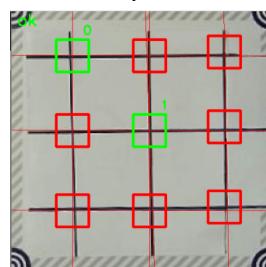
### Premier programme

La première routine que nous avons développée a été l'implémentation d'une première routine qui utilisait la détection de croisement des lignes mise en place dans la partie Computer Vision.

Dans cette routine nous avons cherché en premier lieu à détecter le "default\_workspace" puis à l'afficher.



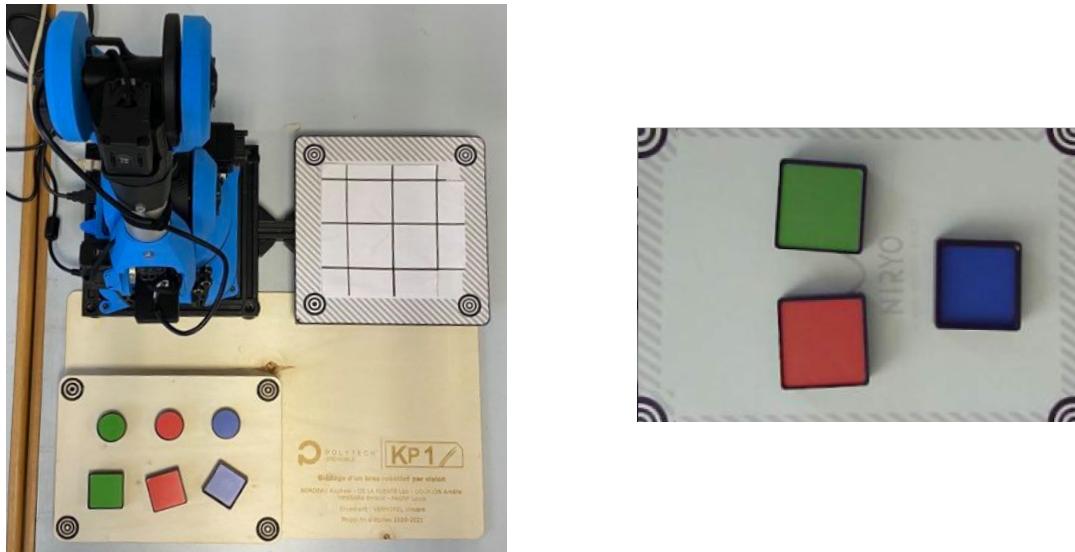
Ensuite en intégrant le code de détection de croisement (cf. [Computer vision](#)), nous avons programmé une routine qui demandait à l'utilisateur quels points il souhaitait sélectionner puis lorsqu'il appuyait sur la touche entrée le robot se déplaçait alors sur chaque point de l'image sélectionnée avec la pince fermée.



### Workshop

Ensuite nous avons travaillé sur la mise en place d'une seconde routine plus complète qui intégrera un magasin d'objet et la routine précédente. Cette routine a été utilisée comme version finale pour le programme.

Tout d'abord nous avons créé un nouvel espace de travail pour le robot que l'on a nommé "workshop". Nous l'avons créé via l'application Niryo one studio. Pour cela il a fallu réimprimer les marqueurs présents sur le "default\_workspace" pour que le robot puisse se repérer et bien détecter le workspace dans l'image capturée par la caméra.



Une fois que nous avions cette image du magasin où l'on voyait très bien les objets, nous avons travaillé sur la détection de la position des ces objets. Il est essentiel d'avoir une position correcte pour chaque objet pour éviter que le robot se détériore en voulant aller trop bas.

Le robot a besoin d'avoir la position du barycentre de l'objet qu'il souhaite attraper. Pour détecter les objets nous avons utilisé des fonctions nous permettant de détecter tous les contours présents dans l'image puis une fonction nous permettant de trouver les plus gros contours de cette image. Cela nous permettait alors d'obtenir des barycentres d'objets. A mesure où nous testions ce programme de détection d'objet nous nous sommes aperçus que notre calcul de barycentre n'était pas aussi bon que ça. En effet il n'était pas réellement au milieu des objets et cela décalait complètement le robot créant alors un fort risque de détérioration. En observant plus en détail les images sur lesquelles nous calculions les barycentre nous nous sommes aperçus que selon les conditions d'éclairage une ombre était plus ou moins présente pour chaque objet du workspace. Comme expliqué dans la partie computer vision (cf. [Computer vision](#)) nous avons utilisé un algorithme permettant de retirer les ombres de l'image afin de calculer plus justement les barycentres des objets.

## Machine Learning

Dans cette dernière partie, nous avons voulu mettre en pratique les connaissances acquises au cours de cette année en ISA, dans le domaine de l'intelligence artificielle. Nous nous sommes donc renseignés sur un modèle de deep learning de reconnaissance d'image.

Nous avons tout d'abord commencé par récupérer nos données, nous avons pris plusieurs photos des objets que nous utilisons (25 photos en moyenne par objets). Nous avons ensuite entraîné notre modèle avec ces données.

Notre modèle est composé de 3 couches de 64 neurones. Nous avons entraîné ce modèle sur 100 époques pour obtenir au finale une précision de 95% sur nos données test.

```
error 111 [1. 0. 0. 0. 0.] 3 0 square_blue circle_blue
0_1614606714.4514332.jpg
100.0 % training data (sample size 92)
95.65217391304348 % new data (sample size 23)
```

Pour conclure, notre modèle fonctionne assez bien pour déterminer les objets en temps réel malgré quelques petites erreurs. Il est alors aisément de rajouter des nouveaux objets pour que cette intelligence artificielle les reconnaissent.

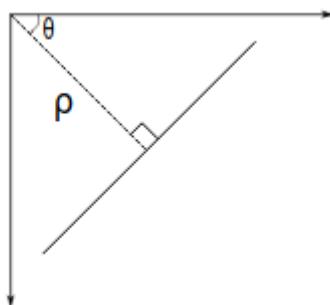
## Computer Vision

Le but de cette partie est d'une part aider la partie robot à fonctionner, et d'autre part détecter les intersections sur laquelle effectuer des actions.

### Détection de croisement

La premier gros travail de la partie "Computer Vision" a été de réaliser la détection d'intersection sur une image. Pour schématiser cette détection, cette fonction prends en entrée une image prise par le robot, et en sortie les coordonnées (en pixels) correspondant aux intersections.

Tout d'abord, le programme va chercher à détecter les lignes présentes sur l'image. Cette détection se base sur la fonction *Houghline* qui permet de détecter les lignes via une itération sur chaque couple de points important détecté par cette fonction. Houghlin permet de sortir 2 caractéristiques importantes  $\rho$  et  $\theta$ , qui sont respectivement la distance et l'angle dans des coordonnées polaires de chaque ligne trouvé.



En fonction du nombre d'apparition de ces couples la fonction *Houghline* sort un tableau comprenant plus ou moins de couple ( $\rho, \theta$ ) en fonction d'un seuil de sensibilité.

Dans la théorie, une fois l'ensemble des couples récupérés, nous utilisons plusieurs fonctions permettant de transformer des coordonnées polaires en coordonnées cartésiennes. Après avoir récupéré les coordonnées des droites dans un système cartésien, on résout un simple système pour trouver le point appartenant aux deux droites. Dans la pratique, un tas de problèmes peuvent survenir.

Par exemple les droites horizontales qui ont un coefficient directeur nul peuvent faire apparaître des divisions par zéro, les droites verticales ont un coefficient vertical de  $+\infty$ , ce que les programmes n'aime pas et qui finissent par être changer en NaN (Not a Number) et crée des erreurs. En ce qui concerne la détection de ligne, si une droite est difficilement discernable il faudra baisser la sensibilité ce qui fera apparaître d'autres droites supplémentaires qui peuvent être soit redondantes soit inutiles. Dans le premier cas il suffit d'assimiler les deux droites comme appartenant au même objet, dans le deuxième cas il est plus difficile de les traiter : il faudra que l'utilisateur n'en tienne pas compte.

Un autre cas difficile à traiter a été le cas des triple intersections. En effet lorsque 3 droites se rejoignent l'algorithme calcule l'intersection de la droite 1 avec la 2, la 2 avec la 3 et la 3 avec la 1, ce qui fait alors 3 intersections différentes. Pour pallier cela nous avons implémenté un seuil permettant de ne garder qu'un seul point lorsque la proximité entre 2 points est inférieur à ce seuil.

### Remove\_shadow

Pour aider le groupe Robot, il a fallu mettre au point une fonction capable de supprimer les ombres afin de mieux détecter le contour d'objets. Pour ce faire nous avons créé la fonction *remove\_shadow* qui reçoit en entré une image et qui lui applique successivement, une dilatation, un flou médian (de taille 11x11) puis fait la différence avec l'image d'origine.

On obtient donc les images suivantes, avec en vert le centre des silhouettes trouvé par les algorithmes:



## Interface utilisateur

Nous avons développé une interface simple qui permet à l'utilisateur d'interagir avec le robot et lui envoyer des commandes.

Cette interface permet à l'utilisateur de :

- Choisir l'adresse IP du robot et se connecter.
- Régler les paramètres de détection des intersections (Sensibilité, Espace entre deux lignes, Espace minimum entre deux intersections). Une explication détaillée de chaque paramètre est fournie dans le manuel d'application.

- Sélectionner des objets dans le workshop et des intersections dans l'espace de travail.
- Exécuter un ensemble d'actions dans un ordre prédéfini.
- Quitter l'application en mettant le robot à sa position de repos.

Nous utilisons deux outils pour l'affichage : OpenCv et PyQt :

- OpenCV nous permet de :
  - Traiter les images pour enlever les distorsions optiques
  - Afficher les images
  - Tracer des formes et afficher du texte
  - Capter la position du pointeur et déclencher des actions
- PyQt nous permet de :
  - Créer une interface qui intègre des sliders et des boutons
  - Créer des tâches séparées (threads)



Nous avons séparé l'exécution du programme en deux tâches. En effet, une tâche est réservée à l'affichage de l'interface et doit réaliser des actions simples pour ne pas paraître figée à l'utilisateur. La seconde nous permet de réaliser des tâches plus longues.

La première tâche, que nous appelons tâche principale, s'occupe de l'affichage et du traitement des images via l'outil OpenCV, de la classification des objets en utilisant la librairie d'intelligence artificielle TensorFlow, et de l'envoi des commandes au robot à l'aide de l'API de Niryo One.

La deuxième tâche est uniquement chargée de récupérer les réglages demandés par l'utilisateur (Sensibilité, Espace entre deux lignes, Espace minimum entre deux intersections) et de les envoyer à la tâche principale en utilisant de variables globales.

Pour réaliser une deuxième tâche, nous avons alors utilisé un Qthread qui est une sorte de thread défini par la librairie PyQt, qui permet de réaliser des tâches s'exécutant en arrière-plan de l'interface graphique.

Le fait d'utiliser deux tâches différentes nous a alors obligé à réaliser une communication entre ces tâches pour qu'elles puissent travailler main dans la main". Ainsi nous avons définis toutes les variables partagées entre les threads comme des variables globales, tout en faisant attention à utiliser des mutex et verrous lors de leur lecture ou écriture pour ne pas que ces variables soient modifiées en même temps par les deux tâches.

# Gestion du projet : Management agile

Pour la gestion de projet, les contraintes étaient relativement souples. Nous étions autonomes dans la mise en place de la méthode agile. La seule exigence était de tenir informer notre client à la fin de chaque sprint, et d'essayer de mettre en place la méthode poker.

## Organisation de l'équipe

Etant donné la bonne entente au sein de notre groupe de projet, nous avons décidé de ne pas définir de leader. Cela forme alors une hiérarchie horizontale, et nous a permis de mettre en place une bonne communication. Bien entendu, si des tensions avaient été présentes au sein de l'équipe, cette organisation n'aurait pas été la plus adéquate, puisque personne n'aurait pu faire attention à détendre ces tensions.

## Méthode Scrum

Pour pouvoir organiser le projet, nous avons utilisé un management Agile basé sur la méthode Scrum.

Cette méthode fonctionne en différents "sprint". Ceux-ci sont des itérations de durées assez courtes durant lesquelles des objectifs précis sont fixés pour toute l'équipe. Chaque fin de sprint est marquée par une évaluation du travail réalisé, et une planification des objectifs du sprint suivant. La fin du sprint est aussi souvent lié à un rendu client pour que celui-ci puisse donner son avis sur le travail réalisé. Cette rencontre avec le client permet aussi de coller au mieux avec ses attentes.



Nous avons défini une durée d'environ 1 semaine pour chaque sprint avec une présentation au client à la fin de chacun d'eux.

## Organisation des tâches

Nous avons choisi d'utiliser l'outil Trello pour nous permettre de nous partager le travail et planifier celui-ci tout au long du sprint.

Celui-ci comporte ainsi plusieurs colonnes dans lesquelles nous pouvons déplacer des sortes de post-it décrivant chaque tâche. Nous avons ainsi organisé cela en 3 colonnes pour le sprint en cours : "A faire durant le sprint", "En cours de réalisation", "Réalisé". L'outils Trello permet de nombreuses fonctionnalités pour définir précisément chaque tâche. Voici les fonctionnalités principales que nous avons réellement trouvé bien utile :

- Associer des personnes à chaque tâches
- Des étiquettes de couleur peuvent être attaché à chaque tâche. Nous les utilisons pour définir les différentes parties du projet associé ( interface utilisateur, robot, computer vision), et aussi pour définir l'importance de la tâche ( Peu important, important, critique).

Une date limite peut être associée à chaque tâche avec des notifications pour les personnes concernées. Etant donné qu'il est difficile d'estimer précisément le temps nécessaire à la réalisation d'une tâche, nous avons rarement utilisé cette fonctionnalité, et pensons qu'elle est utile que pour certaines tâches précises nécessitant vraiment un rendu à temps. Nous avons préféré ajouter marquer le temps prévu en commentaire de chaque post-it pour bien voir le temps total nécessaire au sprint.

## Déroulement d'un sprint

### Début de sprint

Nous nous réunissons tous ensemble pour définir précisément l'objectif attendu pour la fin du sprint. Nous définissons alors les différentes tâches et sous-tâches à réaliser. Ces tâches doivent être les plus précises et les plus courtes possible. Nous attachons les étiquettes d'importance à chaque tâche, puis, nous définissons une estimation du temps de travail nécessaire à chaque tâche. En fonction du temps total des tâches, nous ré-ajustons les objectifs du sprint. Enfin, nous attachons à chaque tâche les personnes qui vont travailler dessus.

### Durant le sprint

Nous réalisons les différentes tâches. Chaque début de journée est marqué par un "stand up meeting" (= mêlée quotidienne), durant lequel nous expliquons aux autres où nous en sommes dans le projet, et ce que nous avons prévu de faire aujourd'hui. Cet échange est très utile pour connaître réellement l'avancement du projet global, ainsi que de travailler sur les bonnes choses au bon moment. De plus, étant donné que nous travaillons tous ensemble sur le robot dans le même bureau, il est alors très facile de demander l'attention de tout le monde pour préciser un point particulier qui a été mal défini, mal compris, ou non pris en compte, ou demander l'avis de toute l'équipe. Cette proximité de travail, nous a permis une meilleure communication au sein du groupe, et de moins facilement nous engager sur du travail inutile.

Tous les soirs, nous jetons un coup d'œil au Trello pour voir notre avancement et savoir sur quoi chaque personne travaillera le lendemain.

## Fin du sprint

L'avant dernier jour est un peu différent, nous sommes obligés de clarifier la situation, et de définir quel livrable sera présenté le lendemain pour la fin du sprint. En fonction de l'avancée du projet et du temps nécessaire, une ou plusieurs personnes travaillent alors à la finalisation de ce livrable.

Le jour de la fin du sprint, nous rencontrons le client pour lui présenter le travail, et le livrable. Nous rediscutons alors ensemble des fonctionnalités présentes, et des futures fonctionnalités attendues. Après cela nous pouvons alors définir le sprint suivant.

## Avantages et inconvénients de la méthode :

Voici la liste des avantages et inconvénients que nous avons relevé de cette méthode de travail.

### Avantages

- Améliore la communication au sein du projet avec par exemple les stand-up meetings qui permettent une communication efficace dès le début de la journée
- Réalise des livrables à chaque sprint, qui sont alors de étapes assez abouties
- Permet de redéfinir les tâche à effectuer en temps réel lorsqu'un problème se présente

### Inconvénients

- Difficulté de définir la durée que va prendre une tâche
- Augmente le nombre de réunions avec le client, et peut donc être considéré comme une perte de temps

Au sein de notre groupe, nous avons noté que la bonne entente et la bonne cohésion du groupe, et le fait de travailler en présentiel dans le même bureau nous a permis d'avoir une communication efficace et facile, ce qui est un point primordiale dans une méthode Agile.

## Utilisation de Git

Tout au long du projet, nous avons utilisé un Git comme logiciel de versionning, ainsi que Github pour nous permettre de nous transférer ce code. Bien qu'ayant eu un cours sur cet outil, nous n'avions pas tous eu l'occasion de bien utiliser celui-ci. Ce projet a alors été l'occasion pour nous de le prendre en main et comprendre son réel intérêt.

## Liste des rendus

Voici la liste des différents rendu que nous avons réalisé :

- Ce rapport permettant de comprendre notre travail et nos conclusions sur celui-ci
- L'application, écrite en python pour être exécuté sur Windows

- Un manuel d'utilisation de l'application permettant de prendre en main cette dernière (avec une vidéo "how to change-motor-4.mp4")
- Un support de présentation permettant de réaliser une soutenance de fin de projet
- Un poster de présentation du projet en Anglais

## Conclusion

Ce projet d'environ 7 semaines nous a permis de réaliser un démonstrateur fonctionnel pour l'entreprise KP1 avec le bras robotique 6 axes Niryo One. Ce projet était très complet puisque nous avons pu développer une application finale simple d'utilisation, mais nous avons aussi eu une relation réelle avec le client pour répondre au mieux au cahier des charges. En plus d'agrandir notre expérience sur un projet concret de groupe, ce travail nous a aussi apporté des connaissances bien utiles sur l'utilisation pratique d'un management Agile. En effet nous serons très vite confrontés lors de notre stage de fin d'étude ou notre futur vie d'ingénieur, à la mise en place de ce type de management.