



Livrable 2 - Robot Kinocto

Design III

présenté à

M. Dominique Grenier, M. Luc Lamontagne et M. Abdelhakim Bendada

<i>matricule</i>	<i>nom</i>
910 058 073	Émile Arsenault
908 190 985	Philippe Bourdages
910 098 468	Pierre-Luc Buhler
998 107 355	Diane Fournier
908 159 170	Imane Mouhtij
908 318 388	Olivier Sylvain
910 055 897	Daniel Thibodeau
910 097 879	Francis Valois

Université Laval
7 mars 2013

Table des matières

Table des figures	iv
Liste des tableaux	v
1 Diagramme des fonctionnalités	1
2 Diagramme physique	2
3 Diagrammes de classes	6
4 Diagrammes de séquences	7
5 Registre de risques	11
6 Plan de tests	16
7 Matrice de vérification des exigences	20
8 Plan d'intégration	23
9 Description des propriétés fonctionnelles	25
10 Avancements pratiques	29
10.1 Alimentation des périphériques 5V	29
10.2 Alimentation 24V du Mac mini et du préhenseur	29
10.3 Réception du signal Manchester	31
10.4 Préhenseur	31
10.5 DEL verte de fin de tâche	33
10.6 Circuit de protection	33
10.7 Source d'énergie	33
10.8 Asservissement des moteurs	33
10.8.1 Modélisation	33
10.8.2 Implantation pratique	34
10.8.3 Implantation électronique	36
10.8.3.1 Les prises de mesures	36

TABLE DES MATIÈRES

ii

10.8.3.2 PIDF	36
10.8.3.3 Ajustement de la vitesse selon la positon	37
10.9 Servomoteurs de la Webcam	37
10.10 Extraction des sudocubes	37
10.10.1 Composantes du système	37
10.10.2 Les solutions retenues/considérées	39
10.10.3 Les moyens utilisés pour configurer	39
10.11 Solveur de Sudocubes	39
10.11.1 Composantes du solveur	39
10.11.2 Stratégie de test	40
10.12 Recherche de chemin	40
10.13 Communication microcontrôleur - Mac mini	40
10.14 Décodage du signal Manchester - partie logicielle	41
10.15 Orientation du robot	42
10.16 Vision par Kinect	42
10.16.1 Transformation des distances	42
10.16.2 Détection des obstacles	43
10.16.3 Détection du robot	44
10.17 Communication entre le Mac mini et la station de base	44
10.17.1 Composantes du système	44
10.17.2 Les solutions retenues et considérées	45
A Annexes	47
A.1 Asservissement des moteurs	47
A.1.1 Fonction agissant comme PID	47

Table des figures

1.1	Figure présentant le diagramme des fonctionnalités	1
2.1	Diagramme physique de l'implantation du robot Kinoclo	3
3.1	Figure présentant le diagramme de classes du projet	6
4.1	Diagramme des séquences présentant les liens entre l'utilisateur et la kinoclo	7
4.2	Diagramme des séquences présentant les liens entre la kinoclo et son environnement afin de déterminer sa position ainsi que le décodage de l'antenne avec les mouvements associés pour arriver	8
4.3	Diagramme des séquences présentant les liens entre la kinoclo et la station de base pour la transmission et l'affichage ainsi que la lecture et le décodage du sudocube avec les mouvements nécessaires pour y arriver	9
4.4	Diagramme des séquences présentant les liens entre la kinoclo et la table de jeu dans l'optique du déplacement de celui-ci pour effectuer le dessin du chiffre contenu dans la case rouge du sudocube	10
8.1	Figure présentant le plan d'intégration	24
10.1	Figure présentant les plans de l'alimentation 5V pour les périphériques	29
10.2	Figure présentant une photo de l'alimentation 5V pour les périphériques	30
10.3	Figure présentant une photo de l'alimentation 24V pour les périphériques	30
10.4	Figure présentant les plans du récepteur Manchester	31
10.5	Figure présentant les plans du circuit de commande du préhenseur	32
10.6	Figure présentant la réalisation mécanique du préhenseur	32
10.7	Figure présentant les plans du circuit de commande du préhenseur	33
10.8	Figure présentant la réponse à un échelon de consigne de $6400[tours^{-1}s^{-1}]$ du système régulé au moyen du régulateur optimisé dans Simulink	34
10.9	Figure présentant la réponse à un échelon de consigne de $6400[tours^{-1}s^{-1}]$ du système régulé au moyen du régulateur réel et de la réponse en position associée	35
10.10	Figure présentant la réponse à un échelon de consigne de $6400[tours^{-1}s^{-1}]$ du système régulé au moyen du régulateur réel et de la réponse en position associée	35

10.11Machine à états qui sert d'interface d'encodeur en quadrature (Cytron. Quadrature Encoder. http://tutorial.cytron.com.my/2012/01/17/quadrature-encoder/ , consulté le 3 mars.)	36
10.12Figure présentant un sudocube traité afin d'extraire les informations des cases	38
10.13Figure présentant un exemple d'images extraites des cases du sudocube et normalisées avant d'être passées au lecteur de chiffres	38
10.14Schéma représentant la table et les différentes mesures obtenues avec la Kinect pour un point quelconque	43
10.15Diagramme représentant la communication entre les nodes et les différents appareils	45

Liste des tableaux

2.1	Matrice de liaisons entre les composantes physiques et les fonctionnalités effectuées : Kinect	2
2.2	Matrice de liaisons entre les composantes physiques et les fonctionnalités effectuées : Station de base	4
2.3	Matrice de liaisons entre les composantes physiques et les fonctionnalités effectuées : Robot	5
5.1	Registre de risques partie 1	12
5.2	Registre de risques partie 2	13
5.3	Registre de risques partie 3	14
5.4	Registre de risques partie 4	15
6.1	Plan de tests côté matériel (partie 1)	17
6.2	Plan de tests côté matériel (partie 2)	18
6.3	Plan de tests côté logiciel	19
7.1	Matrice de vérification des exigences (partie 1)	21
7.2	Matrice de vérification des exigences (partie 2)	22
9.1	Description des propriétés fonctionnelles : section "Vision et Traitement Numérique"	26
9.2	Description des propriétés fonctionnelles : section "Communication et Déplacement"	27
9.3	Description des propriétés fonctionnelles : section "Alimentation et affichage"	28

Chapitre 1

Diagramme des fonctionnalités

Ce chapitre présente le diagramme des fonctionnalités.

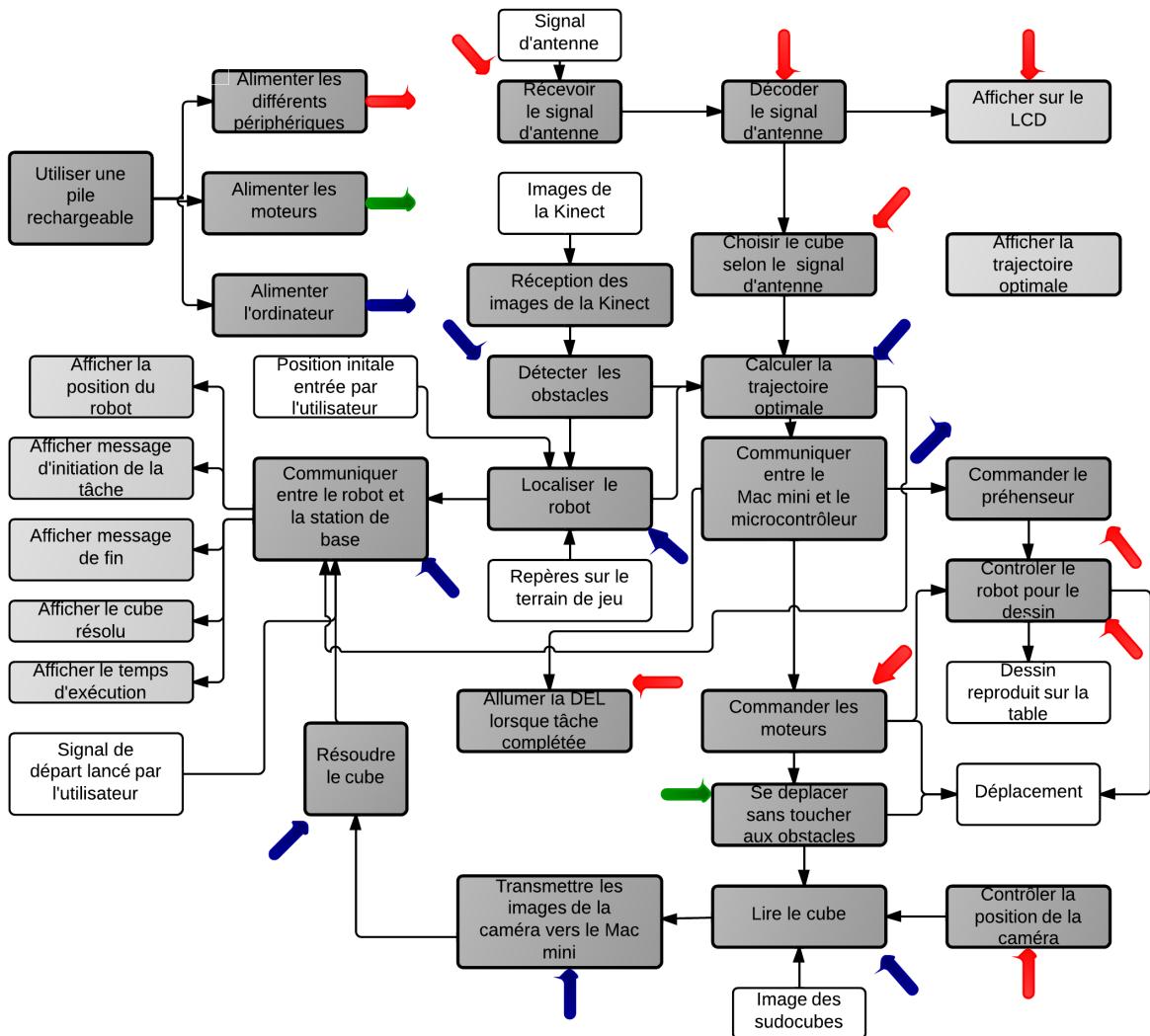


Figure 1.1 – Figure présentant le diagramme des fonctionnalités

Chapitre 2

Diagramme physique

Le diagramme physique est présenté à la figure 2.1. Il est séparé en trois sous-composantes qui sont respectivement : la kinect, la station de base et le robot. Chaque flèche correspond à une interaction entre composantes et peut être soit unidirectionnelle ou bidirectionnelle. De plus, chacun des protocoles ou types d'information sont écrits sur l'interaction pour aider à la compréhension du flux de données dans le projet. Finalement, chacune des boîtes à l'extérieur du rectangle principal correspond à des entrées-sorties nécessaires pour la réussite du projet. Pour permettre une meilleure compréhension du diagramme et éviter une surcharge de celui-ci, les fonctionnalités effectuées par chacune des composantes sont énumérées dans les tableaux 2.1, 2.2 et 2.3.

Tableau 2.1 – Matrice de liaisons entre les composantes physiques et les fonctionnalités effectuées : Kinect

Composantes	Fonctionnalités
Caméra RGB	Déetecter les obstacles
	Localiser le robot
Caméra infra-rouge	DéTECTER les obstacles
	Localiser le robot
Ordinateur embarqué	Transfert des images de la Kinect vers le Mac mini

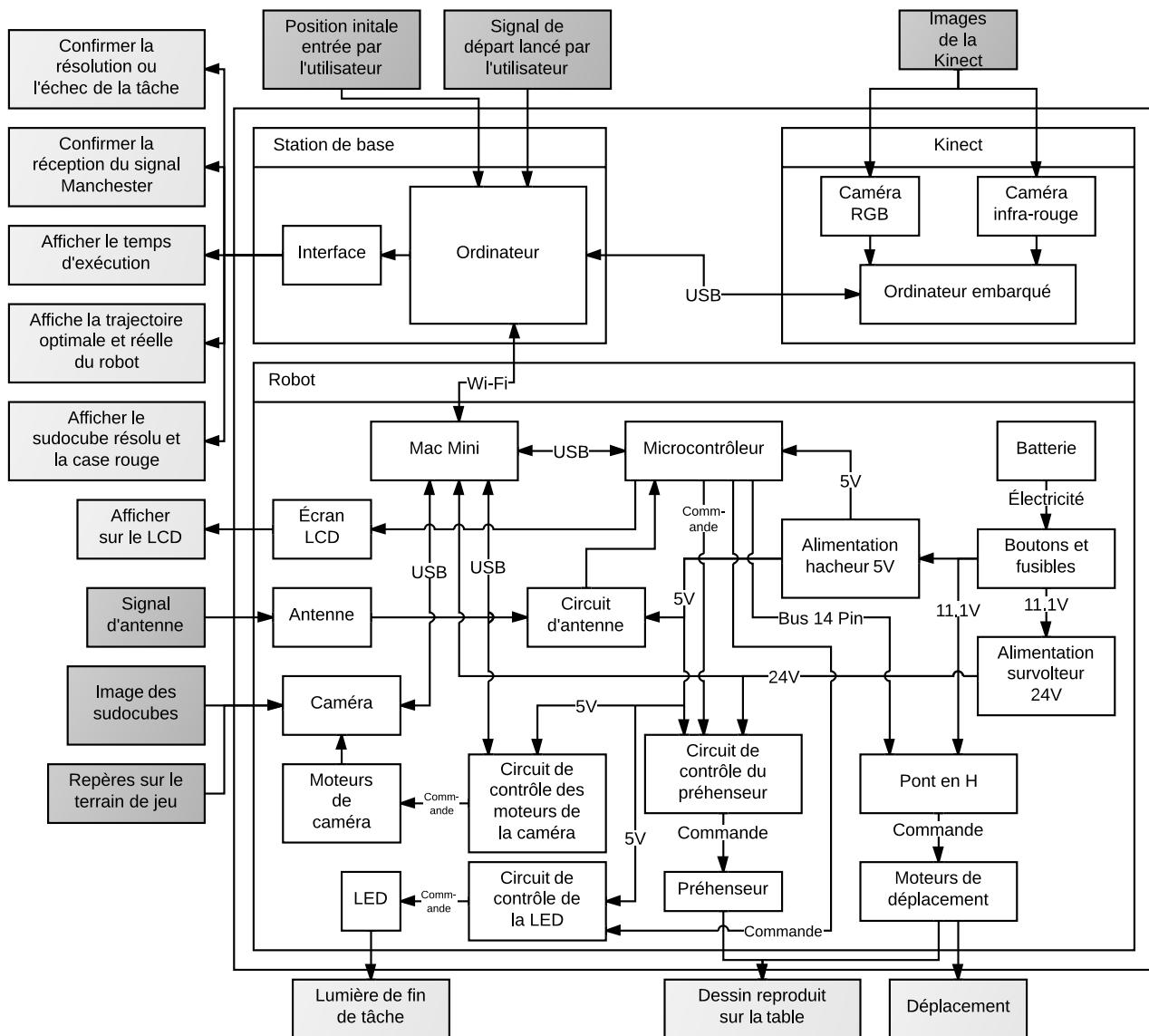


Figure 2.1 – Diagramme physique de l'implantation du robot Kinocto

Tableau 2.2 – Matrice de liaisons entre les composantes physiques et les fonctionnalités effectuées : Station de base

Composantes	Fonctionnalités
Ordinateur	Déetecter les obstacles
	Communiquer entre le robot et la station de base
	Localiser le robot
	Transfert des images de la Kinect vers le Mac mini
Interface	Afficher la trajectoire optimale
	Afficher la position du robot
	Afficher le temps d'exécution
	Afficher message d'initiation de la tâche
	Afficher message de fin
	Afficher le cube résolu

Tableau 2.3 – Matrice de liaisons entre les composantes physiques et les fonctionnalités effectuées : Robot

Composantes	Fonctionnalités
Mac mini	Déetecter les obstacles Choisir le cube selon le signal d'antenne Calcul de la trajectoire optimale Transmettre la trajectoire optimale Contrôler le robot pour le dessin Commander le préhenseur Contrôler la position de la caméra Résoudre le cube Commander les moteurs Allumer la DEL lorsque la tâche est complétée
Écran LCD	Afficher sur le LCD
Micro-contrôleur	Décodage du signal d'antenne Contrôler le robot pour le dessin Commander le préhenseur Commander les moteurs Allumer la DEL lorsque la tâche est complétée
Batterie	Utiliser une pile rechargeable Alimenter les moteurs
Alimentation hacheur 5V	Alimenter les différents périphériques
Alimentation survolteur 24V	Alimenter l'ordinateur et les différents périphériques
Pont en H	Commander les moteurs
Moteurs de déplacement	Se déplacer sans toucher aux obstacles
Circuit d'antenne	Réception du signal d'antenne
Antenne	Réception du signal d'antenne
Moteurs de la caméra	Contrôler la position de la caméra
Circuit de contrôle des moteur de la caméra	Contrôler la position de la caméra
Circuit de contrôle de la LED	Allumer la DEL lorsque la tâche est complétée
Circuit de contrôle du préhenseur	Commander le préhenseur
Caméra	DéTECTER les obstacles Transmettre les images de la caméra vers le Mac mini Lire le cube

Chapitre 3

Diagrammes de classes

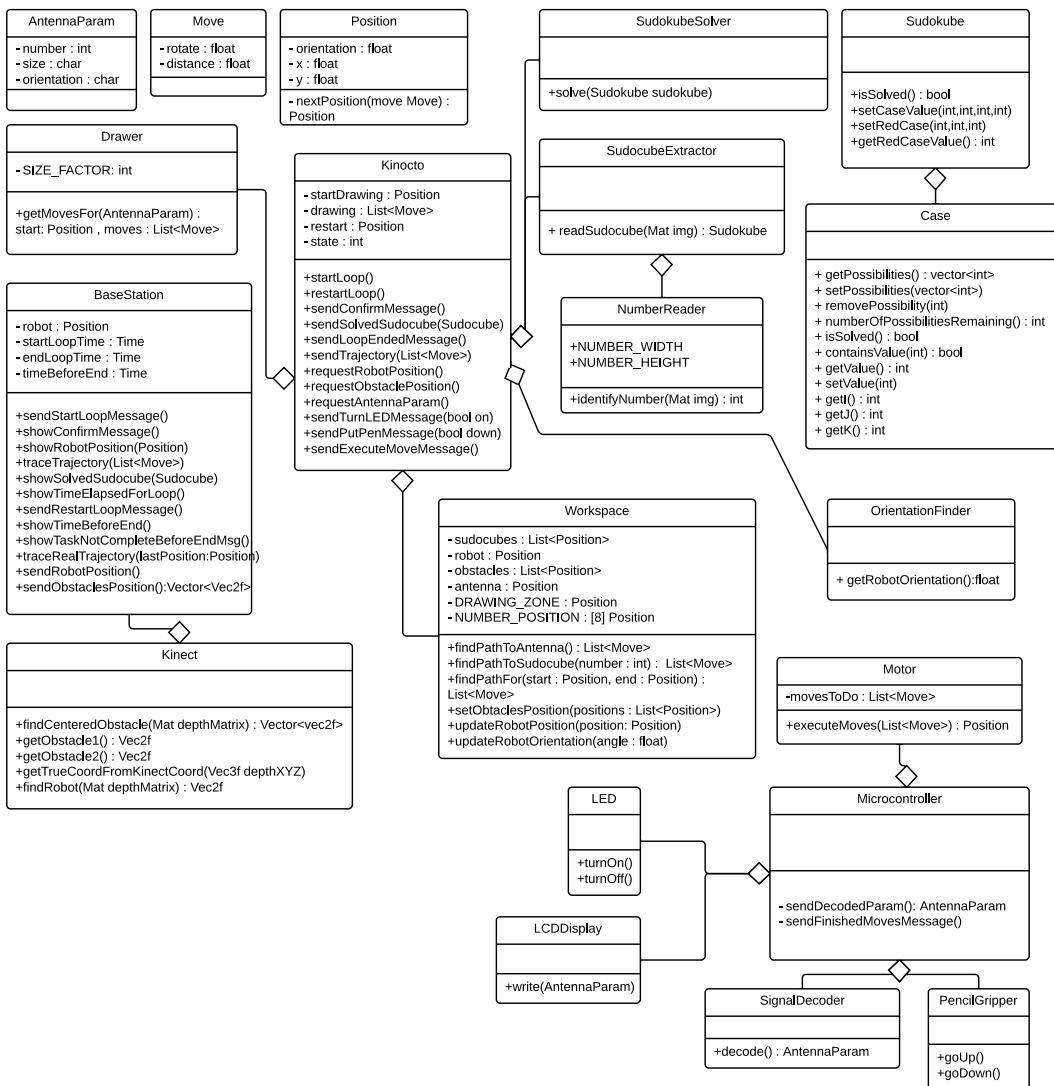


Figure 3.1 – Figure présentant le diagramme de classes du projet

Chapitre 4

Diagrammes de séquences

Ce chapitre présente les différentes figures associées au diagramme des séquences. Ce diagramme est séparé selon cinq portions relatives aux fonctions particulières du robot. La figure 4.1 présente les liens entre l'utilisateur et la kinocto. La figure 4.2 présente les liens entre la kinocto et son environnement afin de déterminer sa position ainsi que le décodage de l'antenne avec les mouvements associés pour arriver. La figure 4.3 présente les liens entre la kinocto et la station de base pour la transmission et l'affichage ainsi que la lecture et le décodage du sudocube avec les mouvements nécessaires pour y arriver. La figure 4.4 présente les liens entre la kinocto et la table de jeu dans l'optique du déplacement de celui-ci pour effectuer le dessin du chiffre contenu dans la case rouge du sudocube.

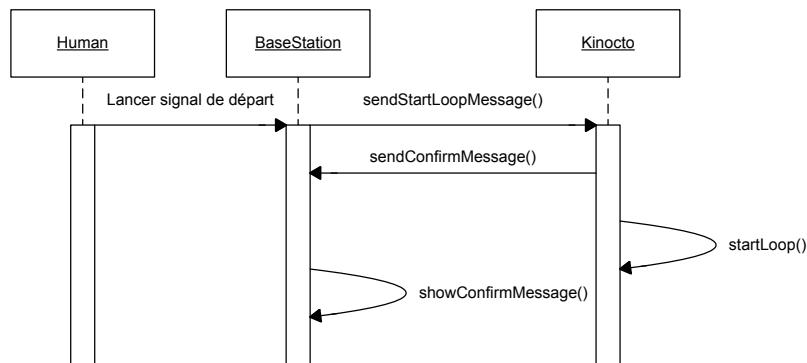


Figure 4.1 – Diagramme des séquences présentant les liens entre l'utilisateur et la kinocto

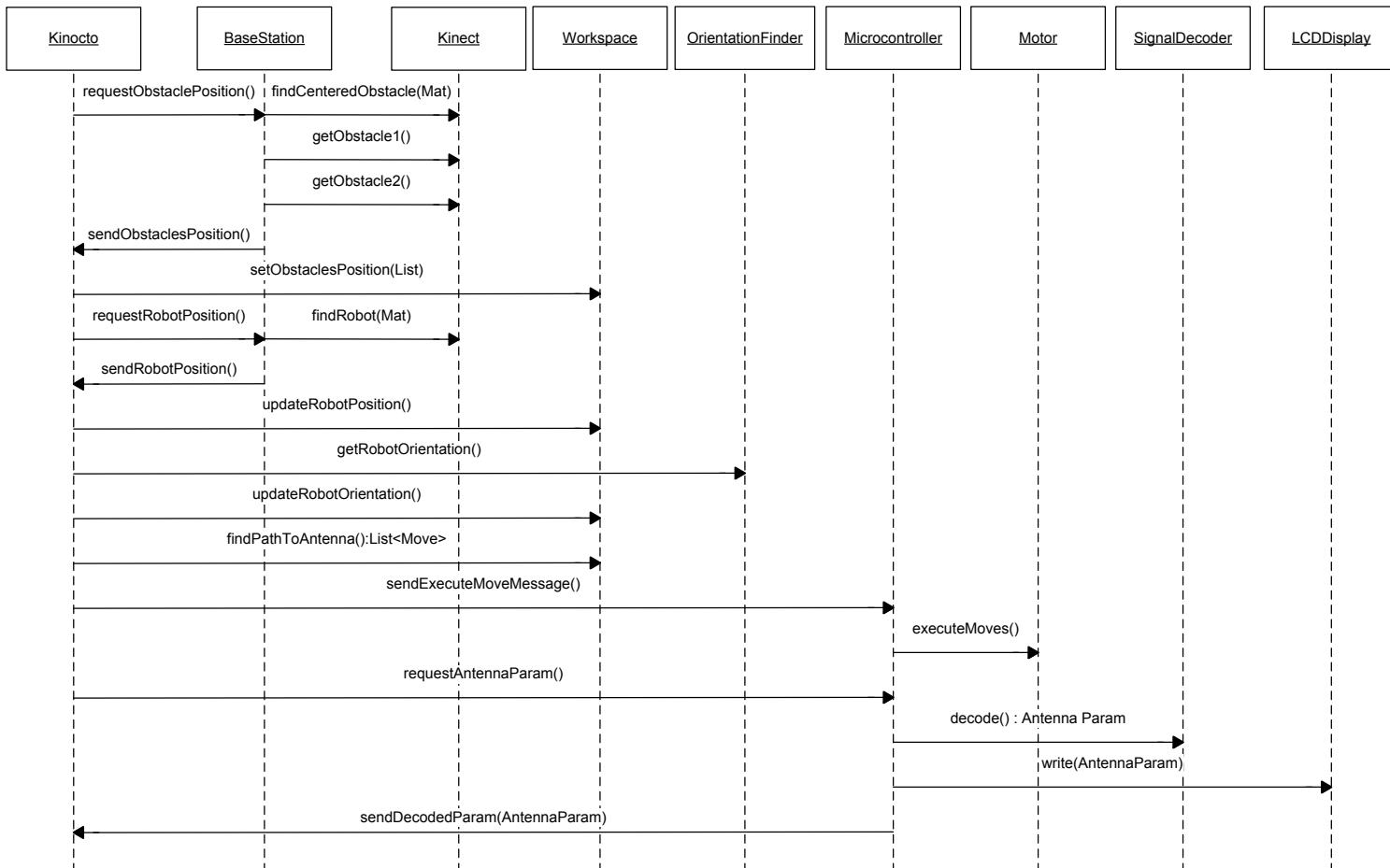


Figure 4.2 – Diagramme des séquences présentant les liens entre la kinecto et son environnement afin de déterminer sa position ainsi que le décodage de l'antenne avec les mouvements associés pour arriver

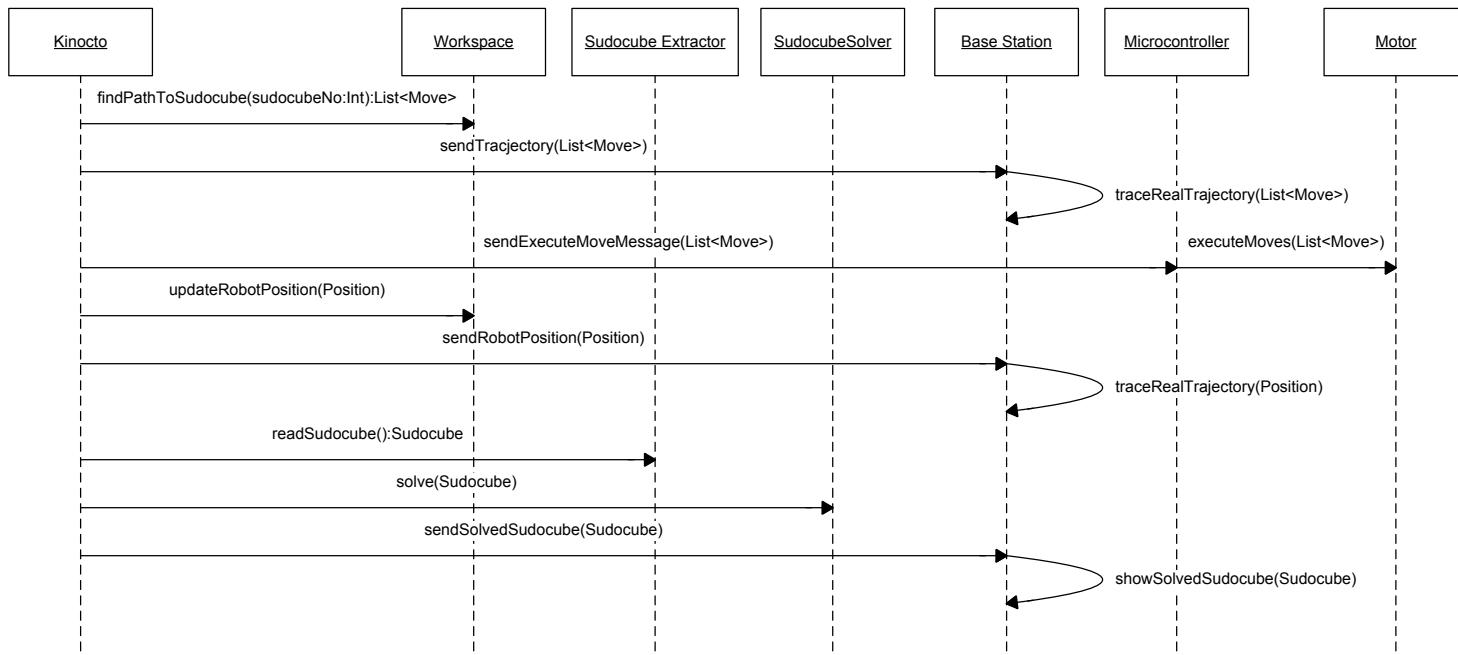


Figure 4.3 – Diagramme des séquences présentant les liens entre la kinecto et la station de base pour la transmission et l'affichage ainsi que la lecture et le décodage du sudocube avec les mouvements nécessaires pour y arriver

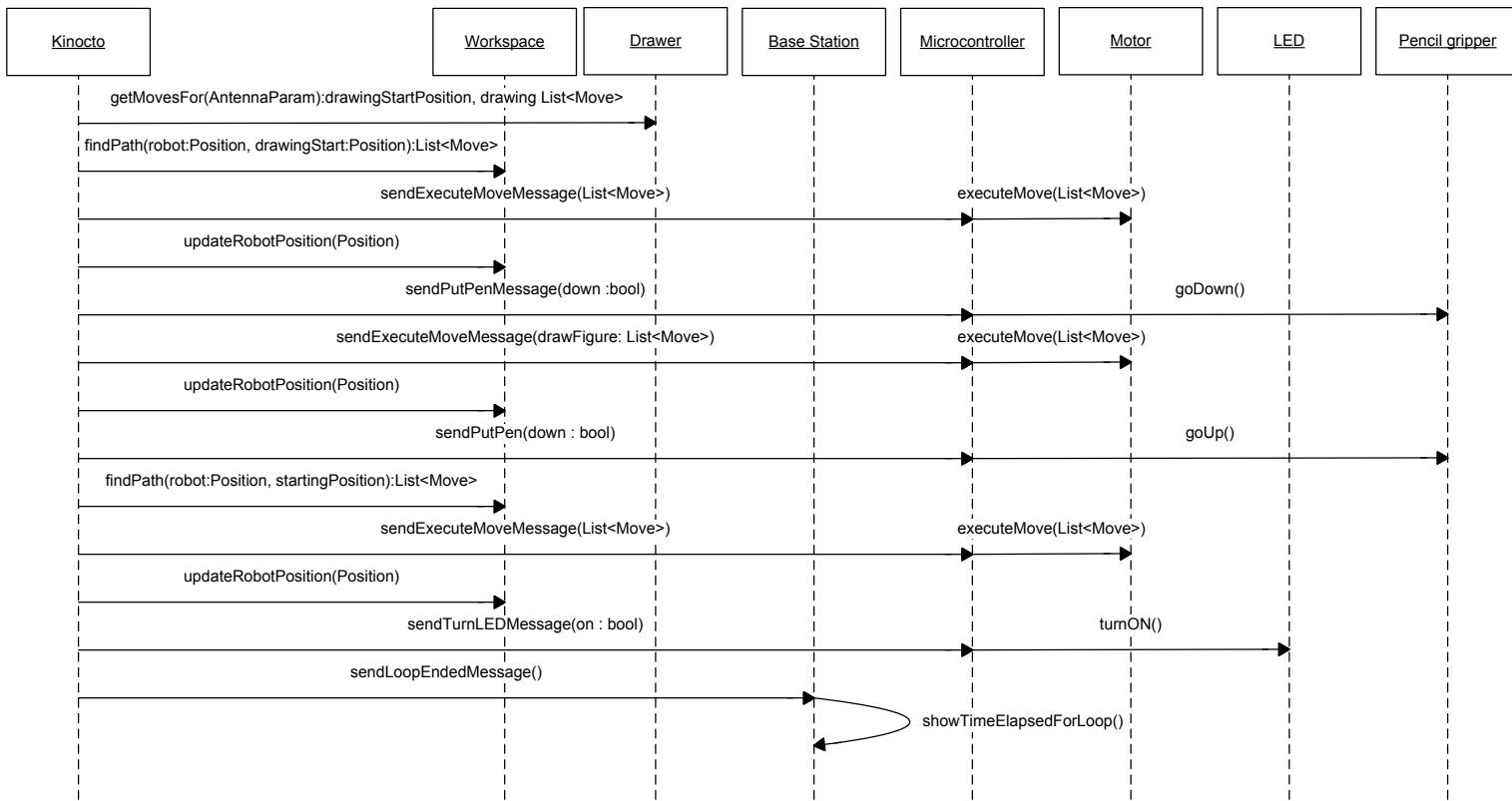


Figure 4.4 – Diagramme des séquences présentant les liens entre la kinecto et la table de jeu dans l'optique du déplacement de celui-ci pour effectuer le dessin du chiffre contenu dans la case rouge du sudocube

Chapitre 5

Registre de risques

Tableau 5.1 – Registre de risques partie 1

Risque	Type de risque	Niveau de priorité (1-faible, 5-élevé)	Conséquences de l'occurrence du risque	Coût en performance associé au risque	Prob. d'occurrence (%)	Coût estimé du risque (\$)	Plan de réduction du risque	Responsable du risque
1	Bris de la batterie Li-Po suite à une mauvaise utilisation	5	Plus d'autonomie énergétique pour le robot, opération impossible	L'ensemble du système sera inopérable	5	60	Formation des utilisateurs à l'endroit de l'utilisation. Apposition d'un capteur de tension. Dispositifs de protection (fusibles)	É. Arsenault
2	Bris d'un ou des système d'alimentation	5	Les systèmes auxiliaires, le Mac mini ou les moteurs ne seront pas alimentés. Le système ne sera pas fonctionnel	Une partie ou l'ensemble du système sera inopérable	5	25	Utilisateur de connecteurs protégés (d'un seul sens possible). Dispositifs de protection (fusibles), surdimensionnement des composantes d'alimentation. Achat de pièces de rechange.	D. Thibodeau
3	Bris du crayon lors du dessin	4	Le dessin ne pourra être complété	La portion dessin sera partiellement complète	20	2	Tests rigoureux et optimisation du choix de crayon avant la compétition	É. Arsenault
4	Bris du système de préhension	5	Le dessin ne pourra être complété et selon le moment du bris, la trajectoire du robot peut être affectée	La portion dessin sera partiellement complète et la trajectoire sera déviée	5	10	Tests rigoureux et essais multiples pour vérifier la stabilité en température lors du fonctionnement	É. Arsenault
5	Problèmes de réception ou de décodage du signal d'antenne	4	Si la réception est erronée, le robot peut exécuter sa séquence, mais l'exécution ne sera pas conforme au message. Si la réception est impossible, le système ne s'amorcera pas.	Accomplissement de la mauvaise tâche ou système non fonctionnel	5	15	Tests répétés pour un taux de succès de 100% lors de la réception et du décodage avec ajout de sources de bruit externes.	D.Fournier

Tableau 5.2 – Registre de risques partie 2

Risque	Type de risque	Niveau de priorité (1-faible, 5-élévé)	Conséquences de l'occurrence du risque	Coût en performance associé au risque	Prob. d'occurrence (%)	Coût estimé du risque (\$)	Plan de réduction du risque	Responsable du risque
6	Bris du pont en H	5	Les moteurs ne pourront être commandés correctement	Le robot ne peut pas se déplacer, le système n'est pas fonctionnel	5	130	Utilisation d'un régulateur de type PID afin d'éviter les appels brusques de courants, positionnement du pont de manière à limiter son exposition aux accrochages	F. Valois
7	Bris d'une portion ou de la totalité du microcontrôleur	4	Le bris d'une portion du microcontrôleur empêche l'exécution de la tâche dans son ensemble et peut occasionner des bris dans les systèmes reliés	Un robot qui ne peut pas se déplacer correctement, qui n'allume pas la DEL ou qui n'active pas le préhenseur	5	100	Isolation des entrées et sorties avec des dispositifs de protection (diode), limiteur de courant	D. Fournier
8	Bris du Mac mini	5	L'ensemble des auxiliaires ne fonctionnera pas correctement. Le système sera non fonctionnel	Robot incapable de se déplacer selon la trajectoire prévue et d'effectuer la tâche	5	600	Apposition de protections électriques (fusibles et interrupteurs) sur l'étage d'alimentation du Mac mini. Fixation robuste du Mac mini sur le robot. Protection du port USB du Mac mini en n'utilisant pas le fil d'alimentation.	F. Valois
9	Caméra web désaxée	3	Les prises de données du sudo cube seront affectées	L'acquisition des données du sudo cube pourrait être non fonctionnelle et causer une erreur dans la résolution du cube	10		Tests rigoureux d'asservissement de la caméra et de retour à l'axe désiré suivant une modification externe.	P. Buhler
10	Bris de la caméra web	4	Le bris de la caméra empêche la vision des cubes	Si la caméra ne peut voir le sudo cube, on ne peut trouver le chiffre dans la case rouge et effectuer le bon dessin	5	80	Storage adéquat de la caméra, protection d'alimentation (fusible), limiter les chocs contre les obstacles	D. Thibodeau

Tableau 5.3 – Registre de risques partie 3

Risque	Type de risque	Niveau de priorité (1-faible, 5-élevé)	Conséquences de l'occurrence du risque	Coût en performance associé au risque	Prob. d'occurrence (%)	Coût estimé du risque (\$)	Plan de réduction du risque	Responsable du risque
11	Problème de communication entre le Mac mini et la station de base	5	Les informations requises ne pourront être transmises correctement, on perd l'information sur le comportement du robot ainsi que sa position. Le robot ne pourra pas se localiser initialement et en temps réel.	Le robot ne remplira pas les exigences d'affichage sur la station de base, le robot ne recevra aucune position de la Kinect	5		Tests répétés pour un taux de succès de près de 100% lors de la transmission et de la réception de l'information en temps réel entre le Mac mini et la station de base	P. Bourdages
12	Bris du système d'exploitation du Mac mini	4	La portion logicielle du robot et le traitement seront absents. Le système ne sera pas fonctionnel	Robot incapable d'accomplir un traitement de tâches	30		Clonage d'une version fonctionnelle et stable du système d'exploitation	P. Buhler
13	Problème de communication entre le Mac mini et le microcontrôleur	5	L'ensemble des auxiliaires ne fonctionnera pas correctement. Le système sera non fonctionnel	Robot incapable de se déplacer selon la trajectoire prévue et d'effectuer la tâche	5		Tests répétés pour un taux de succès de près de 100% lors de la transmission et de la réception de l'information entre le Mac mini et le microcontrôleur	D. Fournier
14	Contact avec un ou des obstacles	3	Bris du système et déviation de trajectoire possible	Un robot qui entre en contact avec les obstacles ne remplit pas les exigences du projet	20		Tests rigoureux sur les déplacements et marge de sécurité importante pour le contournement.	O. Sylvain

Tableau 5.4 – Registre de risques partie 4

Risque	Type de risque	Niveau de priorité (1-faible, 5-élevé)	Conséquences de l'occurrence du risque	Coût en performance associé au risque	Prob. d'occurrence (%)	Coût estimé du risque (\$)	Plan de réduction du risque	Responsable du risque
15	Problème d'identification du robot et de l'environnement (vision) par la Kinect	5	Correction de la trajectoire erronée, risque de rencontrer les obstacles, mauvaise trajectoire calculée	La trajectoire réelle ne sera pas optimale et le robot risque de rencontrer des obstacles	10		Tests rigoureux et taux de succès de l'identification proche de 100%	I. Mouhtij
16	Perturbations de l'environnement du robot (irrégularités sur la table ou dans l'éclairage)	2	La trajectoire du robot pourrait être déviée si la vision est entachée par une mauvaise luminosité ou une irrégularité dans la table	La trajectoire parcourue par le robot ne sera pas idéale	10		Tests rigoureux des algorithmes de vision et d'asservissement, essais avec ajout d'irrégularités	D. Thibodeau
17	Départ de l'un des membres de l'équipe	3	La quantité de tâches des membres restants de l'équipe devra être augmentée.	Des détails et des ajustements de pointes nécessitant plus de temps ne seront pas réalisés	5		S'assurer d'une bonne communication dans l'équipe et d'un bon transfert de connaissances	D. Thibodeau

Chapitre 6

Plan de tests

Tableau 6.1 – Plan de tests côté matériel (partie 1)

Fonctionnalité	Sous-fonctionnalité	Exigence	Méthode de vérification	Équipement requis	Méthode d'analyse
Traitement numérique	Contrôler le robot pour le dessin	Précision de $\pm 1\text{cm}$ de la ligne centrale et temps d'exécution inférieur à 60s	Tests pratiques	Robot fonctionnel Ruban à mesurer Gabarits des dessins Crayon	Effectuer l'ensemble des dessins 3 fois et mesurer l'écart maximal
Déplacement	Se déplacer sans toucher aux obstacles	Distance minimale de 1cm par rapport à l'axe de la trajectoire et vitesse supérieure à 3cm/s	Tests pratiques	Robot fonctionnel Ruban à mesurer Crayon Rapporteur d'angle	Vérifier la déviation maximale ainsi que la vitesse moyenne pour une dizaine de trajectoires différentes
Alimentation	Alimenter le mac-mini et le préhenseur à une tension de 24V	Fournir une tension de 24 volts CC avec des oscillations maximum de 200 mV et une puissance minimale de 75W	Tests de mesure	Mac mini Oscilloscope Sondes de mesure	Vérifier la tension moyenne dans une résistance connue à la sortie de l'alimentation et l'amplitude des oscillations qui y sont superposées sur l'oscilloscope
Alimentation	Alimenter le micro-contrôleur et les différents circuits à une tension de 5 volts	Fournir une tension de 5 volts CC avec des oscillations maximum de 200mV et une puissance minimale de 10W	Tests de mesure	Robot fonctionnel Oscilloscope Sondes de mesure	Vérifier la tension moyenne à la sortie de l'alimentation dans une résistance connue et l'amplitude des oscillations qui y sont superposées sur l'oscilloscope
Affichage	Allumer la DEL lorsque la tâche est complétée	Allumer la DEL lorsque 5V est appliqué à son circuit et l'éteindre lorsqu'on relâche la commande	Test électrique	Robot fonctionnel Source 5V CC	Vérifier que la DEL verte allume lorsqu'on applique une commande de 5V à son circuit

Tableau 6.2 – Plan de tests côté matériel (partie 2)

Fonctionnalité	Sous-fonctionnalité	Exigence	Méthode de vérification	Équipement requis	Méthode d'analyse
Alimentation	Utiliser une pile rechargeable	Avoir un temps de charge (pour 10 minutes) inférieur à 1 heure	Test de mesure	Pile Testeur de batterie Chargeur Chronomètre	Mesurer le temps de charge pour une opération de 10 minutes
Communication Microcontrôleur - Mac mini	Transmettre les commandes du Mac mini vers le microcontrôleur via un port série	Vitesse de transfert d'environ 57600 bit/s	Test pratique	Robot fonctionnel pour ses déplacements	Vérifier que les commandes passent du Mac mini au microcontrôleur et que les mécanismes de vérification et de gestion des erreurs de communication sont efficaces
Affichage des informations sur le robot	Afficher les paramètres décodé de l'antenne par le microcontrôleur sur un écran situé sur le robot	Afficher les paramètres dès qu'ils sont décodés à partir du signal d'antenne.	Test pratique	Robot fonctionnel Câble USB	Vérifier que les informations affichées sont lisibles et qu'elles concordent avec les informations décodées qui se trouvent dans la mémoire du microcontrôleur.
Communication	Commander les moteurs	Avoir une précision de déplacement de 2mm ainsi qu'un temps de réponse (en vitesse) de moins de 1s	Test de manipulation	Robot fonctionnel	Comparer la distance parcourue à la distance commandée au robot. Calculer le temps d'exécution
Communication	Contrôler position de la caméra	Ajuster l'angle et l'orientation de la caméra avec une précision de 5° en moins de 5s	Test de manipulation	Robot fonctionnel	Vérifier la position de la caméra suite à un changement de consigne. Mesurer le temps d'exécution.

Tableau 6.3 – Plan de tests côté logiciel

Fonctionnalité	Sous-fonctionnalité	Exigence	Méthode de vérification	Équipement requis	Méthode d'analyse	
Vision numérique	Déetecter les obstacles	La position des obstacles obtenue à l'aide de la Kinect doit être précise au centimètre près. Le temps de calcul doit être inférieur à 1s	Test de mesure	Station de base Ruban à Mesurer Kinect Algorithme de détection des obstacle Table avec obstacles	Mesurer la distance réelle en X et Y pour différentes position d'obstacles avec et sans obstruction et la comparer avec la position vu par le logiciel. Mesurer le temps d'exécution du logiciel	
	Déetecter le robot	La position du robot obtenue à l'aide de la Kinect doit être précise au centimètre près. Le temps de calcul doit être inférieur à 1s		Station de base Ruban à Mesurer Kinect Algorithme de détection du robot Table avec obstacles Robot	Mesurer la distance réelle en X et Y du robot pour différentes position d'obstacles avec et sans obstruction et la comparer avec la position du robot vu par le logiciel. Mesurer le temps d'exécution du logiciel	
Traitement numérique	Décoder le signal Manchester	Décodage de toutes les combinaisons possibles en tolérant une variation de taux de transmission de ± 50 bits/s	Tests pratiques	Antenne Récepteur Manchester Microcontrôleur	Tester l'algorithme dans tous les cas possibles avec variation de taux de transmission de ± 50 bits/s	
Traitement numérique	Calculer la trajectoire optimale	Calculer la trajectoire optimale en moins de 2s		Tests Pratiques	Ordinateur	Vérifier l'efficacité et le temps de calcul de l'algorithme avec un jeu de positions du robot et des obstacles
Traitement numérique	Résoudre le sudocube	Être capable de résoudre des sudocube de 7 chiffres et plus en moins de 10s		Tests pratiques	Ordinateur	Tester l'algorithme sur un jeu de sudocubes ayant des nombres différents de chiffres
Vision numérique	Lire le sudocube	Le temps de lecture doit être inférieur à 10s	Tests de mesure	Algorithme de lecture Robot fonctionnel Caméra Web Ruban à mesurer Table de jeu	Mesurer la distance entre le sudocube et la caméra après un déplacement vers un sudocube à analyser.	
Orientation	Déterminer l'angle et l'orientation du robot	L'orientation du robot doit être précise au degré près		Algo. d'orientation Caméra embarquée Station de base Table et robot	Trouver l'angle avec l'algorithme et comparer ce résultat avec le résultat calculé manuellement.	

Chapitre 7

Matrice de vérification des exigences

Tableau 7.1 – Matrice de vérification des exigences (partie 1)

Fonctionnalité	Sous-fonctionnalité	Paramètre critique	Méthode de vérification	Commentaires
Vision numérique	Déetecter les obstacles	Temps de calcul (s)	Test	
		Précision en x(cm)	Test	
		Précision en z(cm)	Test	
	Localiser le robot	Temps de calcul (s)	Test	
		Précision en x(cm)	Test	
		Précision en z(cm)	Test	
		Précision angulaire(°)	Test	
Traitement numérique	Lire le cube	Temps de calcul(s)	Test	
	Calculer la trajectoire optimale	Temps de calcul(s)	Test	
	Contrôler le robot pour le dessin	Temps d'exécution(s)	Test	
		Précision (cm)	Test	
	Décoder le signal d'antenne	Taux de transmission (bits/s)	Test	
	Choisir le cube selon le signal d'antenne		Démonstration	Vérification du choix de cube selon un signal d'antenne connu
	Résoudre le sudocube	Temps de calcul (s)	Test	
		Chiffres initiaux minimum	Test	
Communication	Recevoir signal d'antenne		Démonstration	Vérifier que le signal d'antenne reçu est conforme à celui envoyé
	Communiquer entre le robot et la station de base	Vitesse(Mo/s)	Démonstration	Vérifier que les informations transmises du robot et de la station de base sont respectivement reçues
		Latence (ms)	Analyse	
	Communiquer entre le Mac mini et le microcontrôleur	Vitesse(bit/s)	Test	
	Commander les moteurs	Précision (cm)	Test	
		Temps de réponse (ms)	Analyse et test	
	Réception des images de la Kinect	Taux de transfert (Images/s)	Démonstration	Vérifier que les images sont reçues
	Contrôler la position de la caméra	Temps de réponse(s)	Test	
		Précision(°)	Test	
	Commander le préhenseur du crayon		Démonstration	Vérifier que le préhenseur s'active avec une commande

Tableau 7.2 – Matrice de vérification des exigences (partie 2)

Fonctionnalité	Sous-fonctionnalité	Paramètre critique	Méthode de vérification	Commentaires
Déplacement	Se déplacer sans toucher aux obstacles et aux murs	Précision (cm)	Test	
		Vitesse (cm/s)	Test	
Alimentation	Utiliser une pile rechargeable	Énergie (Wh)	Inspection	Lire le manuel d'utilisateur
		Courant maximal (A)	Inspection	Lire le manuel d'utilisateur
		Durée de charge(min)	Test	
	Alimenter les moteurs	Puissance (W)	Test	
		Puissance (W)	Test	
	Alimenter l'ordinateur	Ondulation de tension (mV)	Test	
		Puissance (W)	Test	
		Ondulation de tension (mV)	Test	
Affichage	Afficher message d'initiation de la tâche		Démonstration	Vérifier que le message est bien affiché sur l'écran de base
	Afficher le cube résolu ainsi que la case rouge		Démonstration	Vérifier que le cube résolu et la case rouge sont affichés
	Allumer la DEL lorsque tâche complétée		Test	
	Afficher la trajectoire optimale calculée		Démonstration	Vérifier que la trajectoire optimale calculée est affichée
	Afficher la position et trajectoire réelle	Temps d'actualisation (s)	Démonstration	Vérifier le temps de rafraîchissement de l'affichage de la position en temps réel (inférieur à 15s)
	Afficher message de fin		Démonstration	Vérifier que le message de fin est affiché
	Afficher sur le LCD		Démonstration	Vérifier que le LCD affiche bien les informations

Chapitre 8

Plan d'intégration

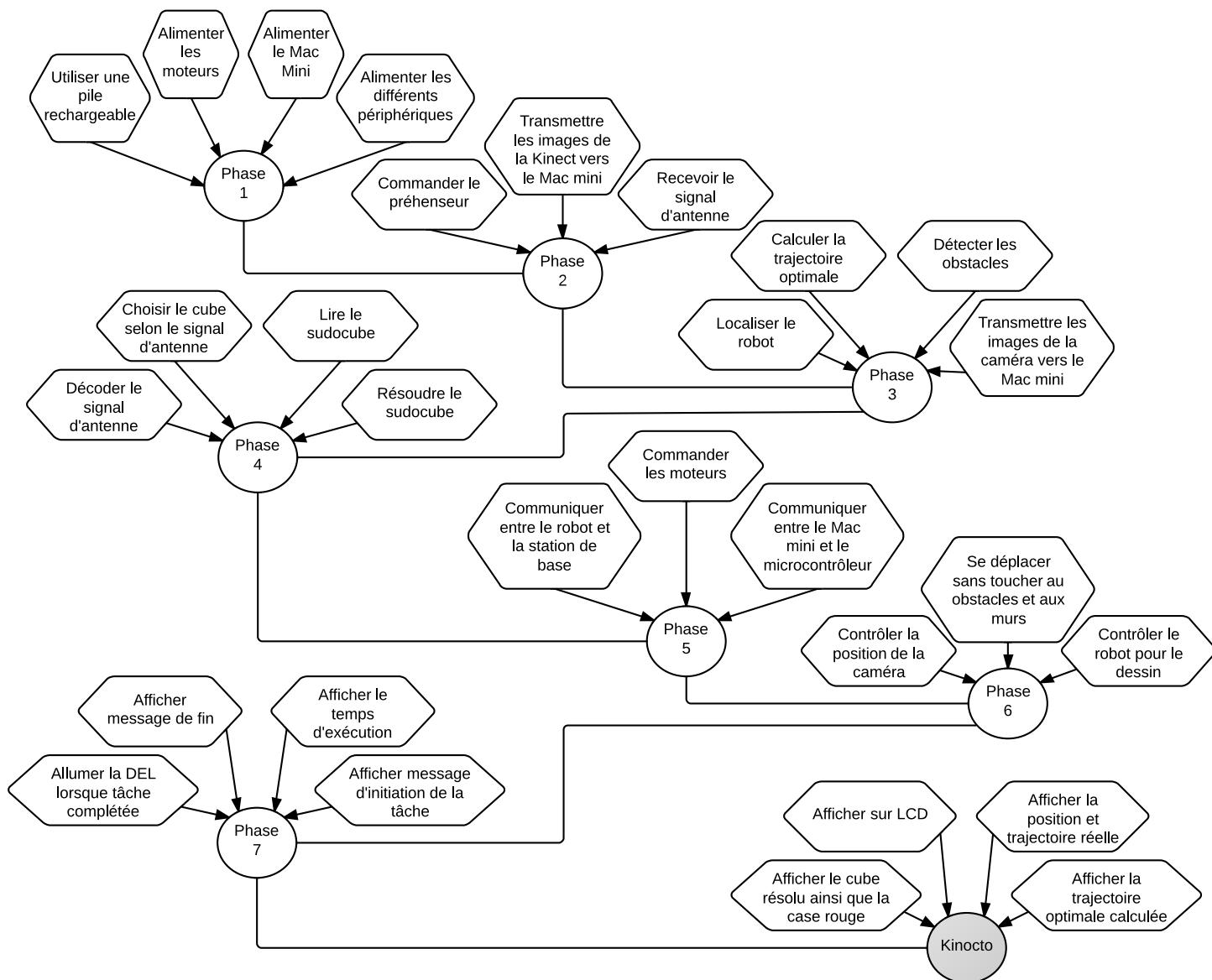


Figure 8.1 – Figure présentant le plan d'intégration

Chapitre 9

Description des propriétés fonctionnelles

Pour simplifier la lecture du tableau de la description des propriétés fonctionnelles, celui-ci a été séparé en trois pages selon trois différentes sections : vision et traitement numérique (présenté dans le tableau 9.1), communication et déplacement (présenté dans le tableau 9.2) ainsi qu'alimentation et affichage (présenté dans le tableau 9.3).

Tableau 9.1 – Description des propriétés fonctionnelles : section "Vision et Traitement Numérique"

Tableau 9.2 – Description des propriétés fonctionnelles : section "Communication et Déplacement"

Tableau 9.3 – Description des propriétés fonctionnelles : section "Alimentation et affichage"

Chapitre 10

Avancements pratiques

Cette section décrit les avancements dans la conception et dans la construction du système Kinocto.

10.1 Alimentation des périphériques 5V

L'alimentation employée pour les périphériques est une alimentation de type buck qui convertit la tension de la batterie (11.1V) vers une tension usuelle de 5V. Cette alimentation utilise un hacheur de tension avec une fréquence autour de 50kHz. Cette fréquence procure une marge de sécurité par rapport à la fréquence d'antenne et évite l'ajout d'un bruit excessif. Cette alimentation a été réalisée avec succès, les plans sont présentés à la figure 10.1. Une photo de ladite alimentation est présentée à la figure 10.2.

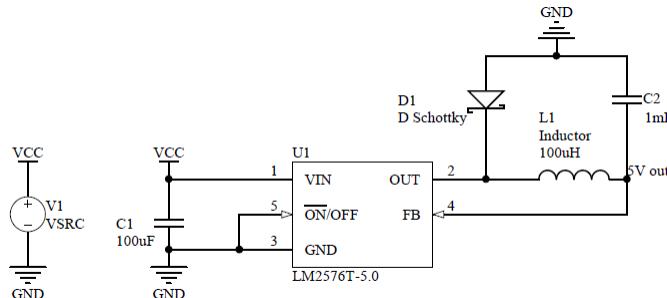


Figure 10.1 – Figure présentant les plans de l'alimentation 5V pour les périphériques

10.2 Alimentation 24V du Mac mini et du préhenseur

Le Mac mini et le préhenseur sont alimentés avec une tension de 24V. Une alimentation correspondante a été achetée déjà préconçue et pré assemblée sur circuit imprimé. Cette solution représente le choix de design le plus économique et le plus fiable selon nos hypothèses. Ce module est un hacheur surveilleur dans lequel la tension de 11.1V de la batterie est amenée à 24V. Le rendement obtenu est d'environ 94% et les ondulations de tensions sont considérées comme négligeables. 10.3.

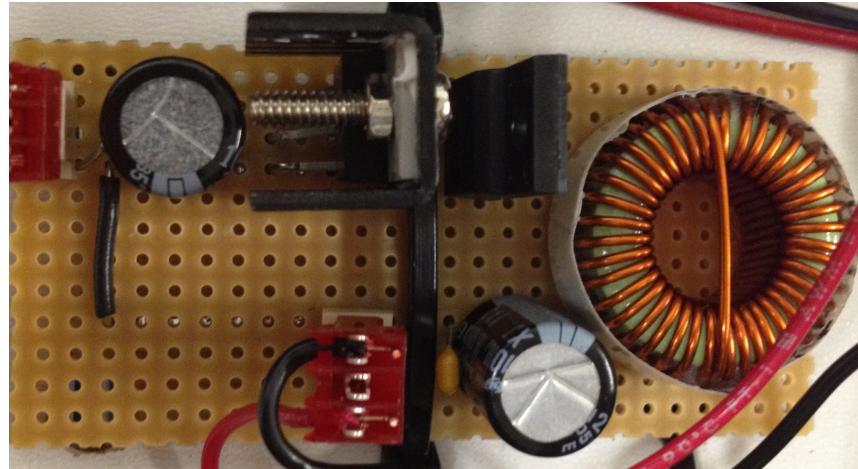


Figure 10.2 – Figure présentant une photo de l'alimentation 5V pour les périphériques

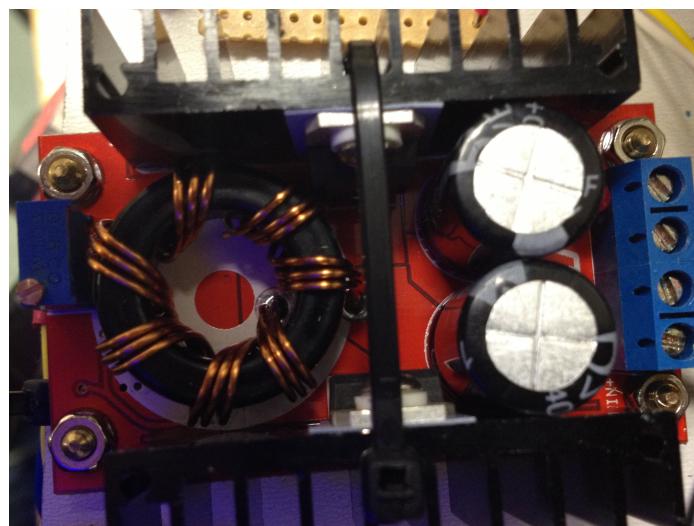


Figure 10.3 – Figure présentant une photo de l'alimentation 24V pour les périphériques

10.3 Réception du signal Manchester

La réception du signal Manchester est réalisée au moyen d'un circuit intégré de décodage de tonalité, le LM567. Ce circuit intégré met la sortie à la masse lorsqu'il détecte la fréquence ciblée sur son entrée. Le signal d'antenne recueilli par le décodeur est capté par un fil bobiné sur plusieurs tours autour du préhenseur (voir la figure 10.6). L'augmentation du nombre de tours de fil permet de maximiser les variations de flux magnétique. Un comparateur à hystérésis est utilisé à la sortie du décodeur afin de contourner les variations de tension causées par l'intensité variable du signal d'entrée. Cependant, comme le signal de sortie du décodeur est inversé, la sortie du comparateur passe dans un inverseur et est envoyée ensuite dans le microcontrôleur qui le décode. Ce récepteur est opérationnel depuis quelques semaines déjà. Le plan de ce récepteur est présenté à la figure 10.4.

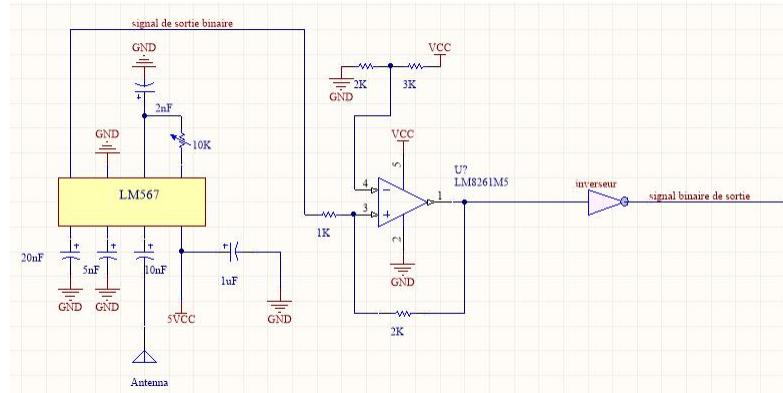


Figure 10.4 – Figure présentant les plans du récepteur Manchester

10.4 Préhenseur

Le préhenseur est réalisé à l'aide d'un actionneur magnétique qui reçoit une commande de 0V ou de 5V qui lui indique s'il doit abaisser ou remonter le crayon. Le circuit magnétique exerce une force vers le bas sur le crayon lorsque la commande est à 5V. Un élastique permet le retour en place du crayon. Pour minimiser les points morts lors des changements de direction du robot, un système de plongeur qui progresse dans un guide à faible tolérance est utilisé. La présence du guide maximise la précision de l'actionneur magnétique (l'angle du crayon reste constant). Le préhenseur est actionné à partir d'une commande analogique 0-5V fournie par le microcontrôleur. Ce signal est injecté sur la gâchette d'un transistor MOSFET qui commute la tension 24V issue de la batterie sur l'entrée du préhenseur. Le préhenseur et son système de commande sont opérationnels depuis un certain temps déjà. On peut observer le schéma du circuit de commande du préhenseur sur la figure 10.5 ainsi que sa réalisation mécanique sur la figure 10.6.

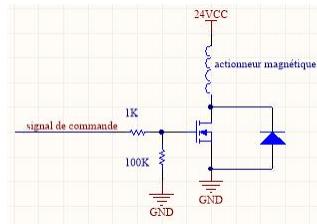


Figure 10.5 – Figure présentant les plans du circuit de commande du préhenseur



Figure 10.6 – Figure présentant la réalisation mécanique du préhenseur

10.5 DEL verte de fin de tâche

Le circuit de la DEL verte qui confirme la fin des tâches du robot utilise un transistor MOSFET de faible puissance. Il reçoit un signal analogique de commande du microcontrôleur sur sa grille et commute une tension de 5V aux bornes de la DEL. Le circuit est présenté à la figure 10.7.

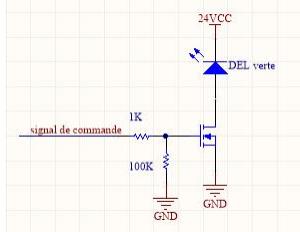


Figure 10.7 – Figure présentant les plans du circuit de commande du pré-henseur

10.6 Circuit de protection

Le circuit de protection utilise trois fusibles en série avec des interrupteurs pour alimenter chacun des circuits d'alimentation 11,1V, 5V et 24V. Chacun des interrupteurs est muni d'un témoin lumineux qui confirme sa mise sous tension et valide que son fusible soit intact.

10.7 Source d'énergie

L'alimentation des différents composants du robot est réalisée à l'aide d'une batterie lithium-polymère, connue pour sa grande capacité énergétique par unité de volume. La batterie employée comporte 3 cellules en série et produit une tension totale de 11.1V pour une capacité totale de 5Ah. Une telle capacité permet une autonomie de 30 minutes dans le pire des cas.

10.8 Asservissement des moteurs

10.8.1 Modélisation

L'optimisation des paramètres de réglage a été réalisée grâce à l'utilisation d'un outil de CAO élaboré au moyen de Matlab-Simulink. La fonction de transfert des moteurs a été identifiée au moyen d'une réponse à l'échelon et de l'outil «*ident*» de Matlab. L'ordre choisi de la fonction est élevé et présente une concordance de 94% avec la réponse obtenue expérimentalement. L'outil *pidtool* permet par la suite de configurer adéquatement le PIDF et

d'obtenir la réponse souhaitée et les paramètres associés. Suivant ces paramètres, le PIDF présent dans le microcontrôleur est ajusté.

L'usage de l'outil interactif permet d'optimiser la réponse et d'obtenir des paramètres de prérégagements qui limitent le nombre d'itérations. La réponse obtenue de façon logicielle est présentée à la figure 10.8

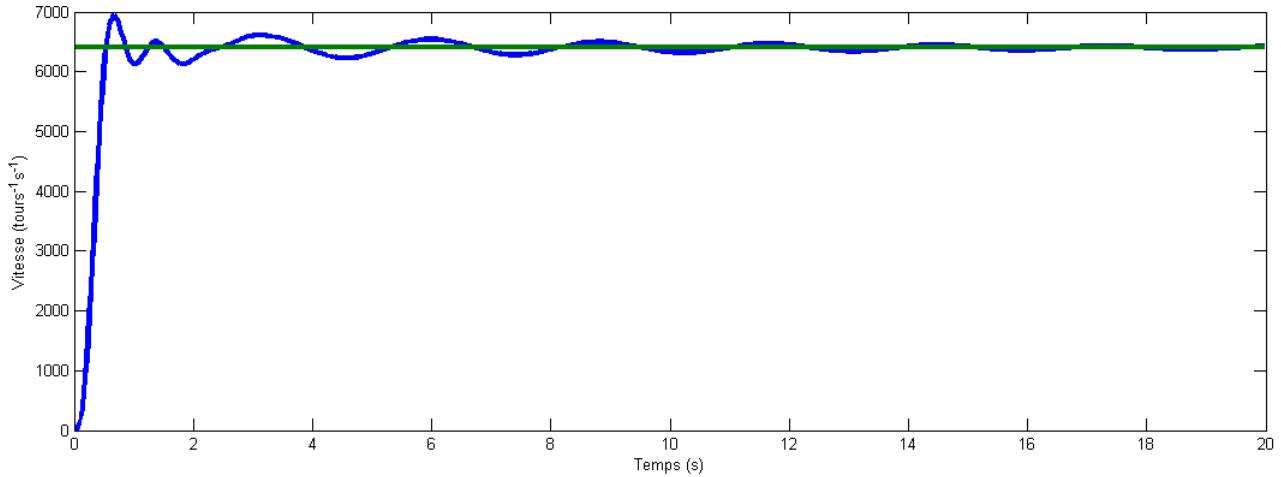


Figure 10.8 – Figure présentant la réponse à un échelon de consigne de $6400[\text{tours}^{-1}\text{s}^{-1}]$ du système régulé au moyen du régulateur optimisé dans Simulink

10.8.2 Implantation pratique

La réponse à un échelon de consigne de $6400 (\text{tours}^{-1}\text{s}^{-1})$ avec les paramètres de réglage réels est présentée à la figure 10.9. À noter que les consignes en $\text{tours}^{-1}\text{s}^{-1}$ indiquent la vitesse en nombre de $1/6400$ de tours de roue. Cette fraction de tour est due au fait que les interfaces d'encodeurs en quadratures captent 6400 transitions par tour complet de roue (voir la sous-section 10.8.3.1 pour plus de détails). Le système a été implanté avec un asservissement en vitesse et sans asservissement de position. Les expériences pratiques réalisées montrent que l'évolution de la position est linéaire, et ce, sans asservissement. La figure 10.10 présente ce phénomène pour une consigne de $6400 (\text{tours}^{-1}\text{s}^{-1})$ qui vise à amener le système à une position de 9100 tours^{-1} . En utilisant les paramètres obtenus dans Matlab comme point de départ, les paramètres de réglages ont été modifiés de manière à obtenir des réponses remplies les critères de stabilité, de vitesse et de dépassement souhaités. Afin d'améliorer la vitesse des déplacements, différents PIDF ont été ajustés selon différentes vitesses d'asservissement. On compte trois de ces PIDF pour les déplacements unilatéraux et un réservé pour les mouvements reliés au dessin. La vitesse de dessin est plus basse que celle du déplacement usuel, en vue de permettre une plus grande précision. Le moteur n'est plus linéaire à de faibles vitesses (roulement et zone morte très limitants) et le système est incapable de démarrer de

lui-même, il est donc impossible, pour de faibles vitesses, de procéder avec une modélisation linéaire du système. Le PIDF a donc été ajusté de manière manuelle par itérations. Les PIDF sont fonctionnels et les systèmes sont stables, cependant la décélération avant l'arrêt et le positionnement critique se font par l'entremise d'un palier de ralentissement à une vitesse intermédiaire avant freinage. Ces courbes de décélérations sont à mettre au point avant de pouvoir entamer le contrôle en dessin. Par ailleurs, les mouvements en diagonale ne sont pas encore parfaitement au point et requièrent quelques heures de travail additionnelles.

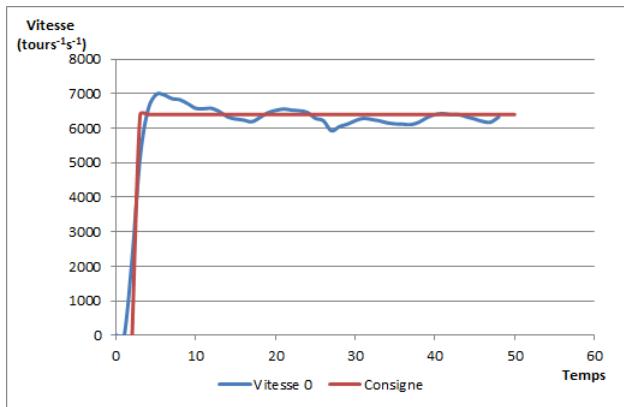


Figure 10.9 – Figure présentant la réponse à un échelon de consigne de $6400[\text{tours}^{-1}\text{s}^{-1}]$ du système régulé au moyen du régulateur réel et de la réponse en position associée

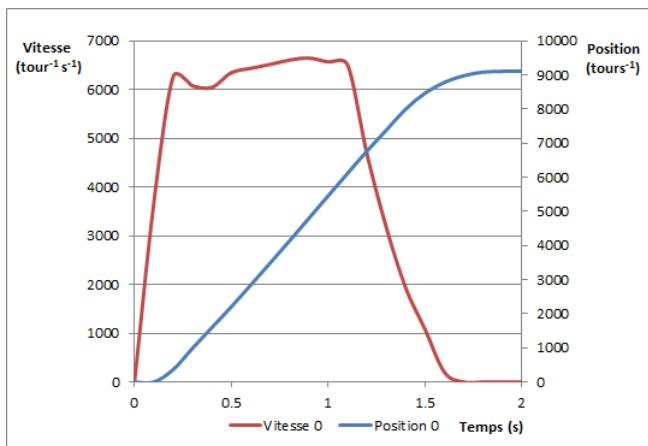


Figure 10.10 – Figure présentant la réponse à un échelon de consigne de $6400[\text{tours}^{-1}\text{s}^{-1}]$ du système régulé au moyen du régulateur réel et de la réponse en position associée

10.8.3 Implantation électronique

10.8.3.1 Les prises de mesures

Deux QEI (Quadrature Encoder Interface) présents sur le microcontrôleur sont utilisés pour prendre des mesures de positions et de vitesses dans l'optique de fermer la boucle d'asservissement. Ces deux interfaces prennent en entrée les sorties des encodeurs situés sur les moteurs et captent les transitions lors de la rotation des moteurs. Puisqu'il y a deux capteurs placés en quadrature sur chaque moteur, on peut détecter la direction de la rotation des moteurs. À prendre note qu'un signal transmis par un encodeur contient 64 transitions par tour complet de moteur et que le ratio de la rotation moteur :roue est 100 :1. Un signal transmis par un encodeur contient alors 6400 transitions pour une rotation complète de roue. Pour les deux autres moteurs, deux interfaces d'encodeurs en quadratures logicielles ont été réalisées à l'aide de deux broches d'entrées/sorties par interface, de courtes interruptions et d'une machine à états. Les interruptions sont lancées lorsqu'une transition est détectée sur l'une des broches. L'état des broches est alors lu et sauvegardé pour être traité par la machine à états. La machine à états utilisée est illustrée à la figure 10.11. Les 0 et 1 indiqués dans les cercles identifient l'état logique des broches. Les -1 ou +1 tracés au-dessus des transitions indiquent s'il faut incrémenter ou décrémenter la position de la roue en rotation.

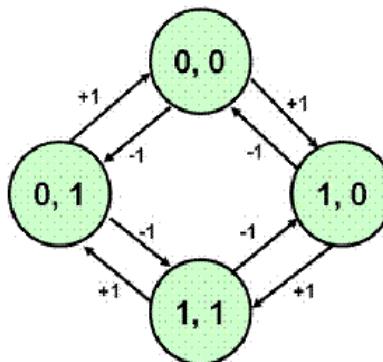


Figure 10.11 – Machine à états qui sert d'interface d'encodeur en quadrature (Cytron. Quadrature Encoder. <http://tutorial.cytron.com.my/2012/01/17/quadrature-encoder/>, consulté le 3 mars.)

10.8.3.2 PIDF

Le PIDF utilisé pour l'asservissement est une courte fonction réalisée dans le microcontrôleur. La fonction théorique implantée est présentée à l'équation 10.1. Cette fonction se retrouve dans l'annexe A.1.1. Prendre note que l'argument "I" est le gain intégral de l'erreur et "dt" est le pas de temps. Cette fonction est exécutée individuellement pour chaque moteur. Des pointeurs transmis en argument indiquent l'emplacement en mémoire des valeurs de chaque moteur utilisées dans le PID. Ensuite, la fréquence d'exécution de la fonction est

gardée constante à l'aide d'un timer dans le microcontrôleur qui déclenche une interruption à chaque période et lance l'exécution de l'asservissement.

$$\text{PIDF}(z) = K_p + K_i \frac{T_s}{z - 1} + K_d \frac{1}{T_f + \frac{T_s}{z - 1}} \quad (10.1)$$

Où

- K_p est le gain proportionnel
- K_i est le gain intégral
- T_s est la durée d'un échantillon
- K_d est le gain différentiel
- T_f définit la coupure du filtre

10.8.3.3 Ajustement de la vitesse selon la position

Comme indiqué précédemment, pour que le robot atteigne la consigne de position, la vitesse de la rotation des roues doit être ajustée selon la distance restante entre le robot et le point de destination. Cet ajustement est implanté par une simple multiplication d'un certain pourcentage avec la consigne de vitesse lorsque la distance restante à parcourir sera de "X" tour^{-1} . Le pourcentage utilisé présentement est hypothétique et sera ajusté lorsque les courbes de décélération discutées dans la section 10.8.2 seront identifiées.

10.9 Servomoteurs de la Webcam

Le contrôle des servomoteurs de la webcam est réalisé avec le contrôleur Micro Maestro de Pololu. L'application "Maestro Control Center" est employée afin de le programmer et de le configurer. Les servomoteurs et le Micro Maestro sont alimentés par l'alimentation 5V. De plus, ils reprennent une position prédéterminée lorsque ceux-ci et le Micro Maestro sont alimentés et restent fixes pour permettre à la Webcam de rester immobile même lorsque le robot est en mouvement. Prochainement, un script sera implanté dans le Micro Maestro à l'aide du "Maestro Control Center" pour transmettre par USB une commande du Mac mini au contrôleur un changement de position des servomoteurs. Cela a pour but de changer l'angle de la caméra web par rapport au sol. Ce changement est nécessaire afin de détecter l'orientation du robot.

10.10 Extraction des sudocubes

10.10.1 Composantes du système

L'algorithme pour extraire les sudocubes est séparé en deux : une classe qui traite une photo afin d'extraire les informations d'un sudocube et un lecteur de chiffres. L'extracteur de sudocubes passe des images au lecteur de chiffres afin d'identifier le chiffre présent.

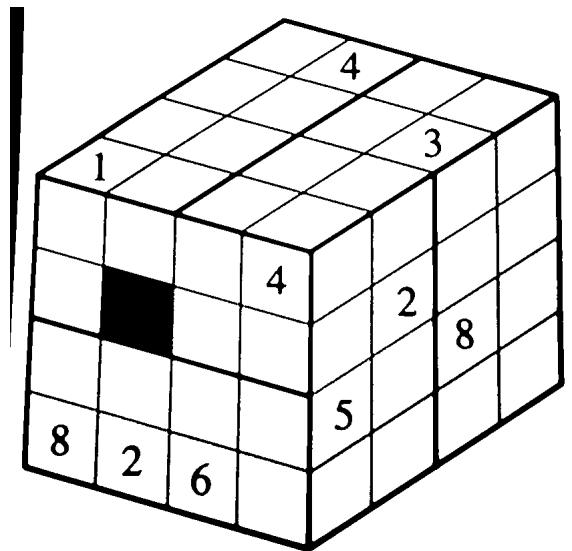


Figure 10.12 – Figure présentant un sudocube traité afin d'extraire les informations des cases



Figure 10.13 – Figure présentant un exemple d'images extraites des cases du sudocube et normalisées avant d'être passées au lecteur de chiffres

10.10.2 Les solutions retenues/considérées

Deux solutions furent considérées pour le lecteur de chiffres : la librairie Tesseract et l'algorithme d'intelligence artificielle KNearest implanté dans la librairie OpenCV. Tesseract est une librairie qui permet de faire la lecture de caractères de plusieurs langues. Cette solution offre des fonctionnalités non nécessaires en plus d'imposer une dépendance supplémentaire au projet. Puisqu'OpenCV est une dépendance obligatoire et que la méthode KNearest permet d'obtenir le même résultat qu'avec Tesseract, le choix s'est arrêté sur cette première solution.

10.10.3 Les moyens utilisés pour configurer

L'extracteur de sudocube fut configuré (seuils, coefficients de dilatation et d'érosion) à partir d'un lot de tests de 42 images (angles, distances sudocube/robot et luminosités variées). La configuration fut réalisée par essais/erreur pour trouver les paramètres optimaux pour toutes les images.

Le lecteur de chiffres est entraîné avec 40 échantillons par chiffres. Puis, il est testé avec un lot de 60 nouvelles images par chiffres afin de vérifier sa robustesse.

10.11 Solveur de Sudocubes

10.11.1 Composantes du solveur

L'algorithme de résolution des sudocubes a été conçu en C++. L'emploi de la programmation par contraintes (CSP) aurait été plus simple à implanter, mais cette solution n'a pas été envisagée au départ. Un prototype conçu en C++ était déjà réalisé au moment de la prise de conscience de l'existence de ce type de programmation. Le développement du solveur est actuellement terminé et ce dernier est fonctionnel.

L'algorithme utilise une structure de données afin de garder en mémoire le sudocube qui est représenté par un tableau à trois dimensions. La structure devient donc une interface pour la vision et permet de stocker les données obtenues lors de l'extraction du sudocube. Les stratégies utilisées pour résoudre le sudocube sont les mêmes que celles utilisées dans le monde des sudokus conventionnels, mais adaptés aux sudocubes. Elles sont présentées dans la liste suivante (ici citées en anglais pour les retrouver plus facilement sur Internet) :

- Naked pairs
- Hidden pairs/triples
- Pointing pairs/triples
- Box and line reduction

Si ces stratégies ne suffisent pas à résoudre le sudocube, la technique de force brute est employée. Il s'agit d'une solution acceptable puisque la probabilité d'occurrence de la situation est faible.

L'algorithme peut trouver une solution pour des sudocubes ayant uniquement 4 chiffres au départ. Il satisfait donc les exigences du client étant donné que les sudocubes présentés au laboratoire ont entre 9 et 14 chiffres. Le temps d'exécution varie de quelques millisecondes à quelques dizaines de millisecondes pour les sudocubes testés.

10.11.2 Stratégie de test

Pour tester entièrement notre classe solveur, il faudrait tester chacune des stratégies utilisées unitairement. Toutefois, tester chacune des stratégies requiert une architecture différente, puisque ces stratégies sont des méthodes privées de la classe solveur. Trois solutions sont envisageables :

Premièrement, il est possible de briser l'encapsulation de la classe solveur, ce qui n'est généralement pas une bonne pratique.

Sinon, un nouvel objet duquel hériterait chacune des stratégies peut être créé. De cette façon, les stratégies deviendraient des objets et seraient injectées dans la classe solveur à sa construction, préservant l'encapsulation. On aurait donc l'avantage de pouvoir tester chacune des stratégies unitairement, en plus de pouvoir isoler la classe solveur pour la tester (en utilisant des mocks). Cette façon de procéder correspond en fait au patron de conception "strategy". Cette façon de tester est la meilleure des trois côté qualité de code.

Finalement, l'encapsulation peut être préservée en ne testant tout simplement pas chacune des stratégies. Nous testerions uniquement la méthode "solve" en passant au solveur des sudocubes et en observant le résultat, ce qui ressemble plus à un test d'intégration.

Dans le cas de ce projet, la troisième option est employée puisqu'elle est plus simple et couvre suffisamment les différents cas du solveur. Ce test est un bon compromis entre le respect du bris d'encapsulation et le fait de tester unitairement chacune des stratégies. La deuxième stratégie serait plus complète, mais requiert trop de temps à implanter.

10.12 Recherche de chemin

La fonctionnalité de recherche de chemin n'en est encore qu'au stade du prototype. La solution développée dans le prototype est viable, mais il reste du travail à faire pour qu'elle soit fonctionnelle. Il faudra entre autres ajouter une couche logicielle afin de réduire le nombre de points dans le chemin trouvé et ainsi réduire le nombre de rotations du robot.

10.13 Communication microcontrôleur - Mac mini

La communication entre le Mac mini a été réalisée avec un port série UART avec le protocole RS-232 à travers le module ICDI du microcontrôleur LM3S9B92. Ce protocole a été choisi en raison de sa simplicité et de la facilité du déverminage par rapport au protocole USB plus complexe. Une interface de communication qui passe des commandes sous forme de caractères a été développée du côté microcontrôleur. Un terminal série écrit en Python

et utilisant la librairie Pyserial a été développé du côté ordinateur. Ce terminal pour usage diagnostique est compatible avec Linux et Windows et permet de récolter des données utiles sous forme de fichiers CSV pour le développement et les tests d'asservissement et de décodage d'antenne. Le terminal final aura une forme légèrement différente afin de s'interfacer avec ROS.

10.14 Décodage du signal Manchester - partie logicielle

Le décodage logiciel du signal de l'antenne doit transformer les bits en encodage Manchester en bits traditionnels. La méthode choisie doit être robuste aux changements de taux de transmission de bits, car il est spécifié dans la description du projet que ce taux peut varier de plus ou moins 50 bits / seconde. Parmi les algorithmes permettant de détecter des signaux de taux de transmission inconnu, il en existe deux types principaux :

1. Les algorithmes basés sur l'échantillonnage

On effectue une mesure du signal à un intervalle prédéfini plus petit que la période du signal mesuré. C'est le décompte des bits consécutifs à zéro et à un qui permet de décoder le signal.

2. Les algorithmes basés sur les timings

On effectue une mesure du temps entre les transitions du signal de zéro à un et d'un à zéro. C'est le temps entre les montées et descentes qui permet de décoder le signal.

Puisque le LM3S9B92 possède des compteurs d'usage général qui implantent une fonction de capture d'événements, la méthode par timings a été employée. Ces compteurs peuvent être utilisés avec un canal du DMA du microcontrôleur. Les interruptions générées par les transitions déclenchent des transferts DMA vers un tampon de données qui contient les valeurs du compteur lors des interruptions. Les timings sont obtenus en soustrayant la valeur courante de la valeur précédente dans le tampon (le compteur est décrémental).

La capture d'événements a d'abord été testée avec un signal PWM interne au microcontrôleur. Par la suite, des échantillons ont été pris avec le signal du récepteur d'antenne pour évaluer la variation du signal et pour tester l'algorithme de décodage qui comporte trois étapes principales.

Premièrement, le plus petit timing qui n'est pas un artéfact est recherché. Étant donné la façon dont le circuit du récepteur est conçu, ce timing correspond à un zéro simple, puisque les montées sont plus rapides que les descentes. Ce timing permet d'avoir une référence pour les valeurs à zéro et permet d'attribuer une valeur de 0 ou 1 à chaque timing, sachant qu'elles alternent. En parcourant les valeurs à un, il est possible d'obtenir une référence similaire pour les valeurs à 1.

Ensuite, les données brutes sont traitées pour enlever les artéfacts et assigner à chaque timing une valeur de '0' simple, '1' simple, deux '0' consécutifs ou deux '1' consécutifs. Ceci est possible grâce aux références pour les 0 et les 1 qui permettent de déterminer un seuil pour assigner une valeur double plutôt qu'une valeur simple.

Finalement, un algorithme de recherche de séquences tel que l'algorithme de Knuth-Morris-Pratt est utilisé pour trouver la séquence des bits d'arrêt et de départ afin de trouver le point de départ et décoder le signal utile.

10.15 Orientation du robot

Pour la détermination de l'angle d'orientation du robot, le choix de la caméra embarquée offre une meilleure précision que celui de la Kinect.

Dans un premier temps, la caméra doit être calibrée. Pour ce faire, deux méthodes sont envisageables : soit l'application de l'algorithme de Zhang ou la calibration avec OpenCV. La méthode retenue est celle employant l'algorithme de Zhang, car elle est plus efficace.

Ensuite, la détermination de l'angle du robot par rapport à la ligne rouge présente sur la table est réalisée avec un traitement d'image supportant la variation de luminosité, d'ombres, ainsi qu'au moyen de règles de trigonométrie.

Enfin, la détermination de l'orientation du robot (N, S, E, O) est effectuée en détectant les coins de la table. L'algorithme de cette opération est assez simple. Si la caméra détecte le coin orange avant le bleu, le robot sera orienté vers le Nord. Si elle détecte le coin bleu avant le coin orange, le robot sera orienté au Sud. La détection des deux coins bleus montrera une orientation vers l'Ouest et enfin la détection des deux coins orange montrera une orientation vers l'Est.

Les deux dernières parties sont commencées, mais ne sont pas encore complétées à ce jour.

10.16 Vision par Kinect

Pour aider au déplacement du robot, la Kinect a été utilisée afin de détecter la position des obstacles, la position du robot lui-même ainsi que sa position angulaire par rapport au point (0,0) de notre représentation cartésienne de la table. Toutefois, comme la Kinect est située à l'extérieur de la table de jeu, il faut une translation et une rotation des données obtenues afin de positionner les objets avec exactitude. Les 3 sections qui suivent expliquent en détail les différentes parties de l'algorithme de vision de la Kinect.

10.16.1 Transformation des distances

En premier lieu, il est important de clarifier quelles distances il est possible d'obtenir à l'aide de la Kinect. Le «Framework» OpenNI est capable de retourner 2 types de distances pour chacun des points dans l'image infrarouge obtenue de la Kinect. Le premier type de distances retourné est la longueur, en mètres, entre l'objectif et un point quelconque sur

l'image. Le second type est dérivé du premier et correspond aux composantes X, Y et Z du vecteur de longueur entre l'objectif et la cible. Pour aider à la compréhension, les deux types de distances peuvent être représentés par les lignes vertes sur l'image 10.14. Toutefois, comme l'origine de notre représentation cartésienne de la table est située sur le coin inférieur droit de la table et que la Kinect n'est pas située à cet endroit, il est nécessaire d'obtenir la distance X et Y (lignes bleues sur l'image 10.14) entre l'origine et le point ciblé sur l'image obtenue de la Kinect. Pour y arriver, il suffit d'obtenir la position X et Y (lignes rouges sur l'image 10.14) de la lentille infrarouge de la Kinect ainsi que l'angle de la Kinect avec l'axe X de la table. Avec ces valeurs, dans notre cas 14cm, -56cm et 23.5° , un logiciel a été mis en oeuvre pour permettre la transformation des distances de la Kinect vers les composantes X et Y recherchées à l'aide de règles de trigonométrie simples. Cet algorithme possède une précision de plus ou moins 1cm pour n'importe quel point situé sur l'image obtenue avec la Kinect.

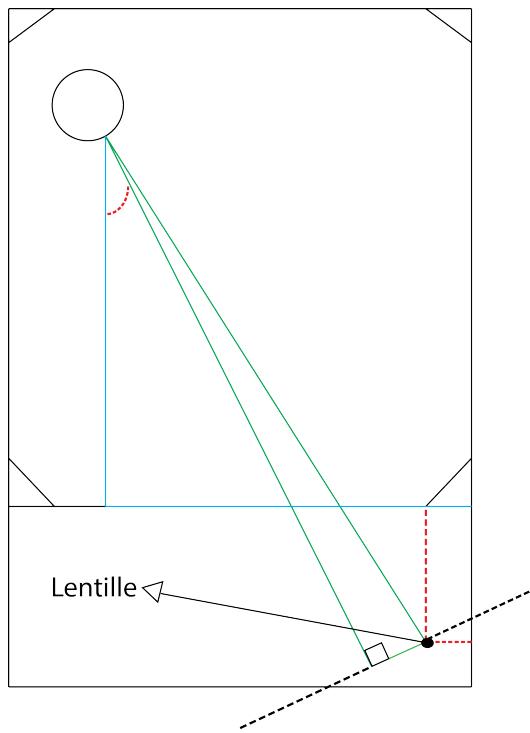


Figure 10.14 – Schéma représentant la table et les différentes mesures obtenues avec la Kinect pour un point quelconque

10.16.2 Détection des obstacles

Comme il est possible d'obtenir n'importe quelle distance entre l'origine et un point quelconque sur l'image, un algorithme de recherche d'obstacles a été créé. Sachant que les obstacles sont situés dans une zone précise sur la table et que ceux-ci font 40cm de hauteur, il suffit de rechercher tout objet de cette hauteur situé dans la zone prédéfinie. La Kinect

retourne la hauteur de chacun des points sur l'image infrarouge et permet de localiser les obstacles. De plus, à l'aide de calculs statistiques, l'algorithme est en mesure de trouver les obstacles lorsqu'ils sont presque alignés ou obstrués par un autre objet comme le robot. Comme cet algorithme dépend entièrement de l'algorithme précédent, les positions obtenues pour chacun des obstacles possèdent la même incertitude de 1 cm sur chaque mesure de distance effectuée. En ce qui concerne l'efficacité de l'algorithme, différents tests montrent un temps de calcul d'environ 30 ms sur un Core 2 Duo 2.4 GHz pour obtenir la position du centre des deux obstacles.

10.16.3 Détection du robot

En ce qui concerne la détection du robot, un algorithme semblable à la détection des obstacles a été utilisé. Comme le robot possède une hauteur d'environ 20cm et une profondeur semblable, il est possible d'isoler, dans la matrice de distance, un carré qui correspond aux dimensions du robot. De plus, la précision de la détection du robot peut être améliorée par la mise en place d'une recherche de symboles de couleur spécifique sur le robot à l'aide de la caméra RGB de la Kinect. Avec cette méthode qui n'est pas encore implémentée, il sera possible d'obtenir la position réelle du robot, peu importe sa position sur la table, et ce avec une vitesse qui se rapproche de l'algorithme de détection des obstacles. Pour le moment, en se servant seulement de la détection par contrainte de distance, il est possible de détecter le robot avec une incertitude de 5 cm en environ 30 ms si le robot n'est pas situé derrière un obstacle.

10.17 Communication entre le Mac mini et la station de base

10.17.1 Composantes du système

Avant de pouvoir communiquer avec le robot, l'adresse IP du robot est récupérée grâce à un script bash exécuté lors du démarrage de celui-ci. Le script récupère l'adresse IP et l'envoie par courriel à un compte GMail.

Ensuite, ROS est employé pour gérer la communication entre les différents nodes ROS. Un node est une application qui utilise l'API de ROS. Actuellement, l'envoi de messages au Mac mini pour démarrer la séquence du robot a été testé.

Le diagramme 10.15 présente la répartition des nodes sur les différents appareils ainsi que les appareils avec lesquels chaque node travaille. Le node du Mac mini est séparé en deux en deux afin de simplifier les tests pour les composantes qui communiquent avec le microcontrôleur. Grâce au système de communication de ROS, il est possible d'envoyer, par ligne de commande, des messages afin de déplacer le robot ou encore d'afficher un message sur l'écran LCD, et ce, sans passer par l'écriture d'une application de tests.

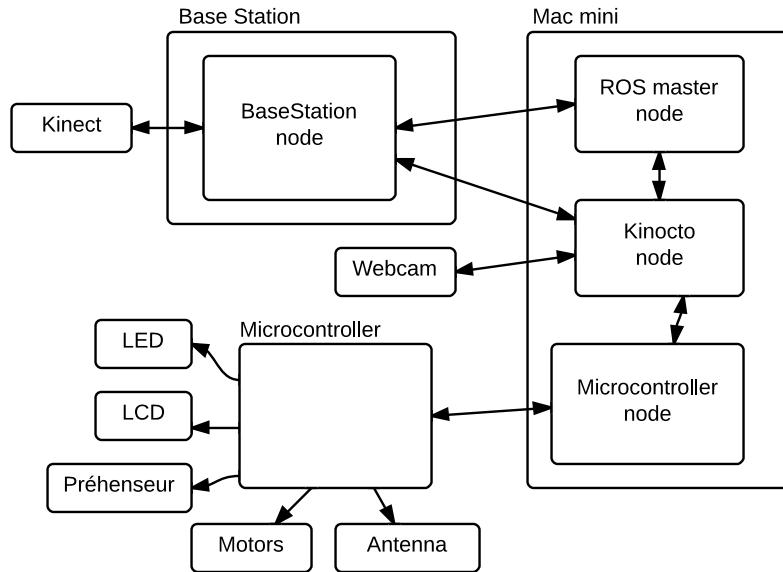


Figure 10.15 – Diagramme représentant la communication entre les nodes et les différents appareils

10.17.2 Les solutions retenues et considérées

Deux solutions ont été considérées : POCO, une librairie C++, et ROS, un environnement de développement pour robot.

POCO permet d'envoyer des chaînes de caractères par TCP/IP. L'envoi de commandes avec POCO consiste en ces étapes : concaténer le nom d'une commande (choisi au préalable) et ses arguments dans une chaîne de caractères. Puis, envoyer, à l'aide de la librairie, la chaîne de caractères. Il faut extraire la commande et ses paramètres lors de la réception d'un message.

ROS fait exactement la même chose que POCO en arrière-plan, mais il y a une couche logicielle qui donne des avantages supplémentaires. Pour envoyer des messages entre deux applications ROS, il suffit de déclarer, pour chaque type de message, des variables et leur type dans un fichier texte. Puis, lors de la compilation, le projet ROS génère des struct en C pour chaque message. Ensuite, l'api de ROS est employé pour définir le contenu des messages et envoyer les messages.

La courbe d'apprentissage est plus prononcée avec ROS. Cependant, avec celui-ci les messages sont hachés avec une somme MD5 et lorsqu'un message est reçu par une node (une application ROS) le contenu est vérifié avec la somme MD5, nous garantissant ainsi que le message s'est transmis.

De plus, il est possible d'enregistrer tous les messages qui sont envoyés par ROS depuis le démarrage de la séquence, de les consulter séquentiellement dans le temps grâce à une interface graphique et de les exécuter à nouveau afin de reproduire la même séquence de message. C'est un outil de déverminage qui sera utile lors du développement de l'agent intelligent et que l'on ne possède pas avec POCO.

ROS a été retenu pour ses nombreux avantages.

Annexe A

Annexes

A.1 Asservissement des moteurs

A.1.1 Fonction agissant comme PID

```
1 long PIDHandler( volatile long *consigne, volatile long *measured_value, volatile float *I,←
      volatile long *previous_error, float dt)
2 {
3     long error = *consigne - *measured_value;
4     *I = *I + error*dt;
5     float D = (error - *previous_error)/dt;
6     long output = Kp*error + Ki>(*I) + Kd*D;
7     *previous_error = error;
8     return output;
9 }
```