



Livrable 2 - Robot Kinocito

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	13
6 Plan de tests	18
7 Matrice de vérification des exigences	22
8 Plan d'intégration	25
9 Description des propriétés fonctionnelles	26
10 Avancements pratiques	30
10.1 Alimentation des périphériques 5V	30
10.2 Alimentation du Mac mini	30
10.3 Asservissement des moteurs	32
10.3.1 Modélisation	32
10.3.2 Implantation pratique	32
10.3.3 Implantation électronique	33
10.3.3.1 Les prises de mesures	33
10.3.3.2 PID	35
10.3.3.3 Ajustement de la vitesse selon la positon	35
10.4 Servomoteurs de la Webcam	35
10.5 Extraction des sudocubes	35
10.5.1 Composantes du système	35

TABLE DES MATIÈRES

ii

10.5.2 Les solutions retenus/considérés	37
10.5.3 Les moyens utilisé pour configurer	37
10.6 Solveur de Sudocubes	37
10.7 Recherche de chemin	38
10.8 Communication microcontrôleur - Mac mini	38
10.9 Décodage du signal Manchester - partie logicielle	38
10.10 Orientation du robot	39
10.11 Vision par Kinect	39
10.11.1 Transformation des distances	39
10.11.2 Détection des obstacles	40
10.11.3 Détection du robot	41
10.12 Communication entre le Mac mini et la station de base	41
10.12.1 Composantes du système	41
10.12.2 Les solutions retenues et considérées	41
A Annexes	44
A.1 Asservissement des moteurs	44
A.1.1 Fonction agissant comme PID	44

Table des figures

1.1	Figure présentant le diagramme des fonctionnalités	1
2.1	Diagramme physique de l'implémentation du robot Kinocto	3
4.1	Diagramme des séquences présentant les liens entre l'utilisateur et la kinocto	8
4.2	Diagramme des séquences présentant les liens entre la kinocto et son environnement afin de déterminer sa position	9
4.3	Diagramme des séquences présentant les liens entre la kinocto et la station de base pour la transmission et l'affichage de la trajectoire optimale	10
4.4	Diagramme des séquences présentant les liens entre la kinocto et la table de jeu dans l'optique du déplacement de celle-ci à travers les obstacles vers la zone de lecture ainsi que les liens avec la résolution du sudocube	11
4.5	Diagramme des séquences présentant les liens entre la kinocto et ses différents périphériques lors de la production du dessin et de la confirmation de la résolution de la tâche	12
10.1	Figure présentant les plans de l'alimentation 5V pour les périphériques . . .	30
10.2	Figure présentant une photo de l'alimentation 5V pour les périphériques . .	31
10.3	Figure présentant une photo de l'alimentation 24V pour les périphériques . .	31
10.4	Figure présentant la réponse à un échelon de consigne de $6400[\text{tours}^{-1}s^{-1}]$ du système régulé au moyen du régulateur optimisé dans simulink	32
10.5	Figure présentant la réponse à un échelon de consigne de $6400[\text{tours}^{-1}s^{-1}]$ du système régulé au moyen du régulateur réel et de la réponse en position associée	33
10.6	Figure présentant la réponse à un échelon de consigne de $6400[\text{tours}^{-1}s^{-1}]$ du système régulé au moyen du régulateur réel et de la réponse en position associée	34
10.7	Machine à états servant d'interface d'encodeur en quadrature (Cytron. Quadrature Encoder. http://tutorial.cytron.com.my/2012/01/17/quadrature-encoder/ , consulté le 3 mars.)	34
10.8	Figure présentant un sudocube traité afin d'extraire les informations des cases	36
10.9	Figure présentant un exemple d'images extraites des cases du sudocube et normalisées avant d'être passées au lecteur de chiffres	36
10.10	Schéma représentant la table et les différentes mesures obtenues avec la Kinect pour un point quelconque	40

TABLE DES FIGURES

iv

10.11Diagramme représentant la communication entre les nodes et les différents appareils	42
--	----

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	14
5.2	Registre de risques partie 2	15
5.3	Registre de risques partie 3	16
5.4	Registre de risques partie 4	17
6.1	Plan de tests côté matériel (partie 1)	19
6.2	Plan de tests côté matériel (partie 2)	20
6.3	Plan de tests côté logiciel	21
7.1	Matrice de vérification des exigences (partie 1)	23
7.2	Matrice de vérification des exigences (partie 2)	24
9.1	Description des propriétés fonctionnelles : section "Vision et Traitement Numérique"	27
9.2	Description des propriétés fonctionnelles : section "Communication et Déplacement"	28
9.3	Description des propriétés fonctionnelles : section "Alimentation et affichage"	29

Chapitre 1

Diagramme des fonctionnalités

Ce chapitre présente le diagramme des fonctionnalités.

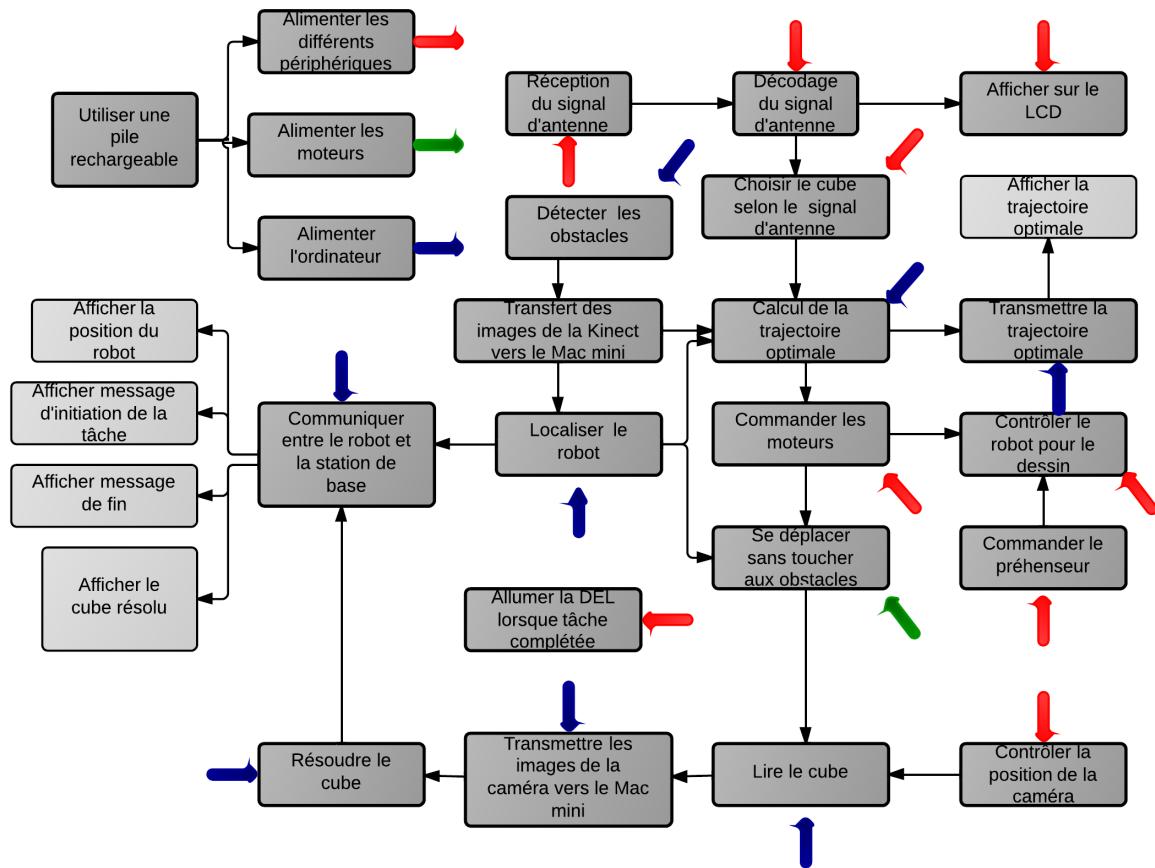


Figure 1.1 – Figure présentant le diagramme des fonctionnalités

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 correspondent 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

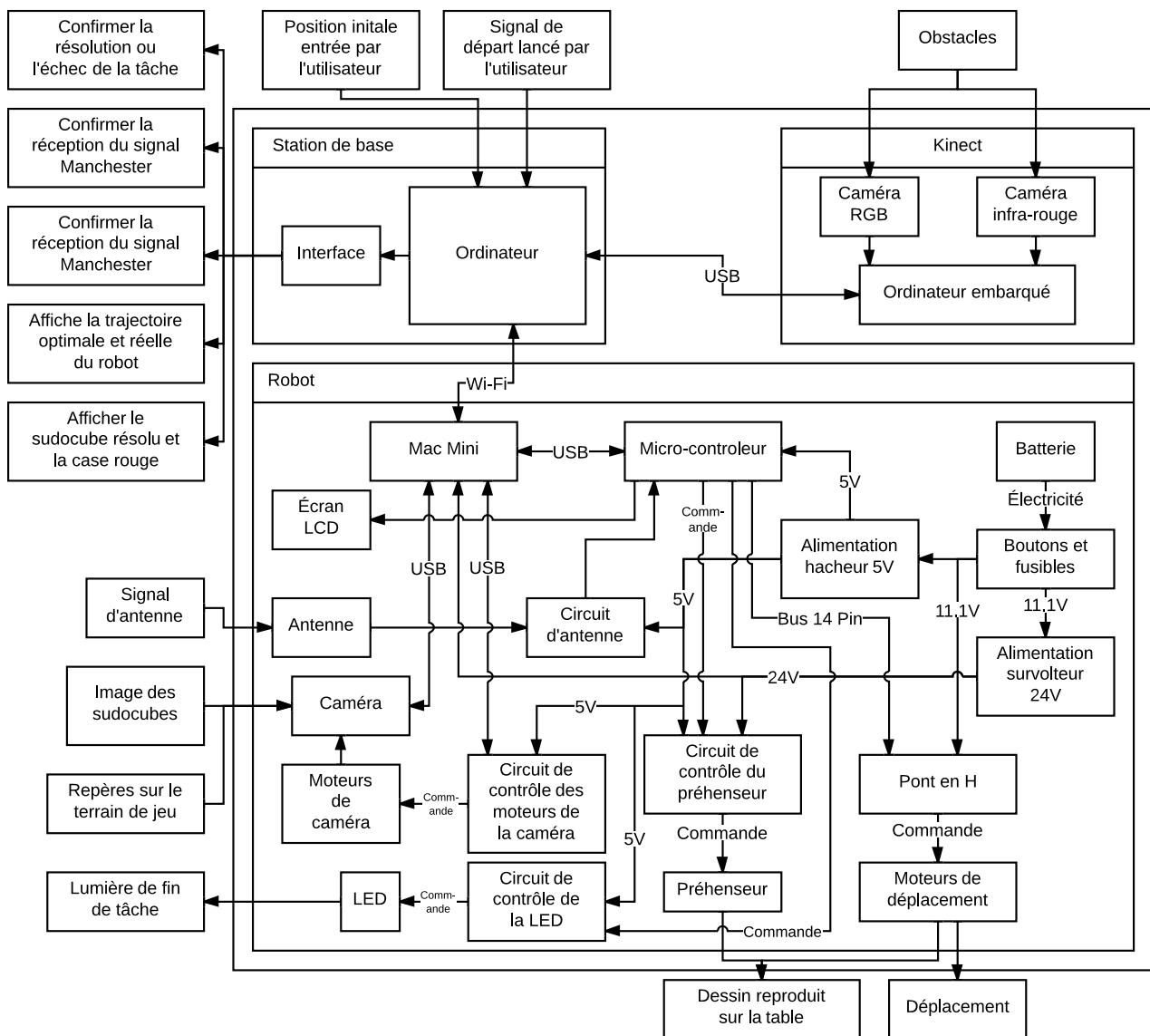


Figure 2.1 – Diagramme physique de l'implémentation du robot Kinocto

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

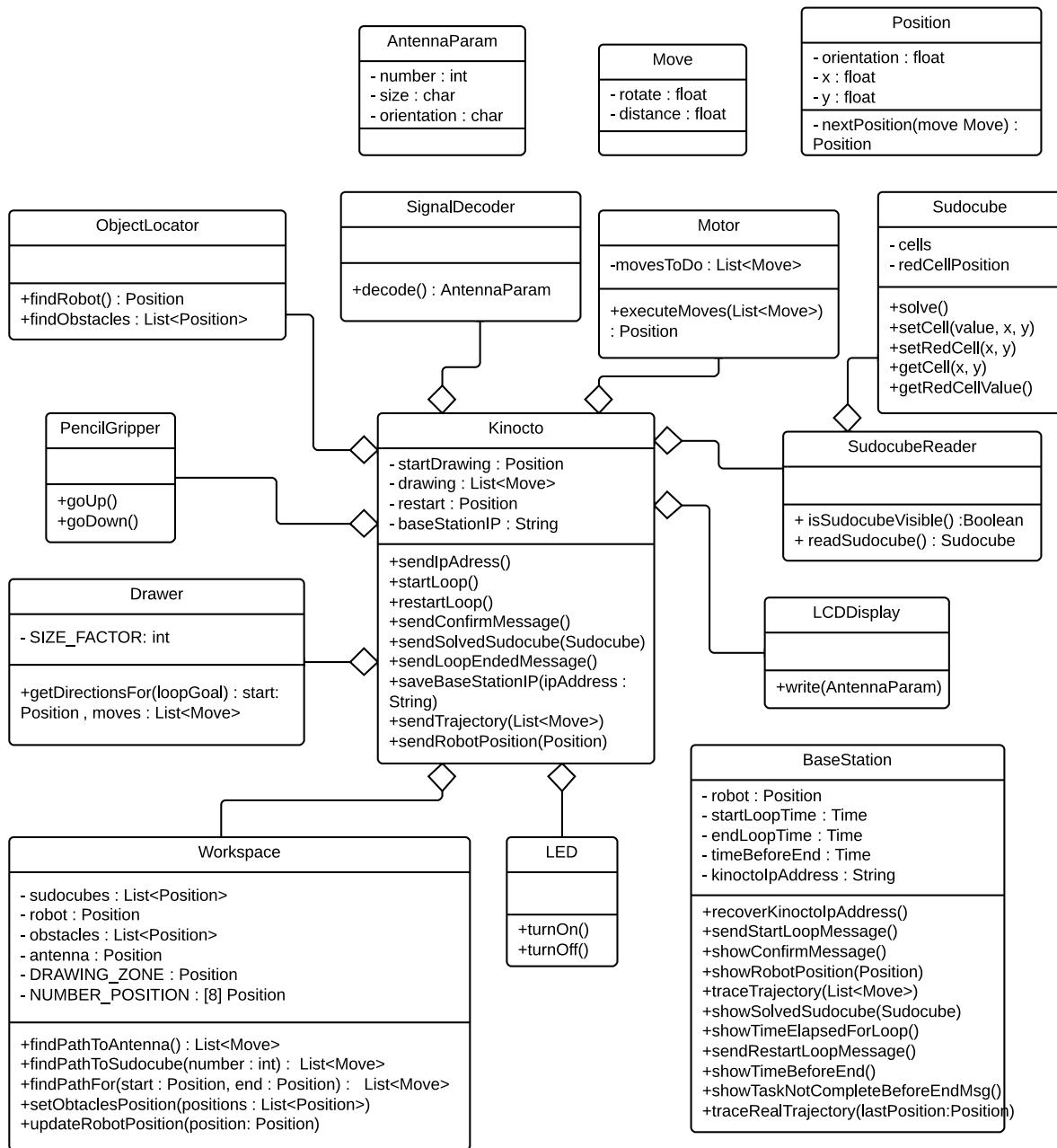
Composantes	Fonctionnalités
Ordinateur	Déetecter les obstacles
	Communiquer entre le robot et la station de base
	Localiser le robot
	Transfert des images de la Kinect vers le Mac mini
Interface	Afficher la trajectoire optimale
	Afficher la position du robot
	Afficher message d'initiation de la tâche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



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. La figure 4.3 présente les liens entre la kinocto et la station de base pour la transmission et l'affichage de la trajectoire optimale. La figure 4.4 présente les liens entre la kinocto et la table de jeu dans l'optique du déplacement de celle-ci à travers les obstacles vers la zone de lecture ainsi que les liens avec la résolution du sudocube. La figure 4.5 présente les liens entre la kinocto et ses différents périphériques lors de la production du dessin et de la confirmation de la résolution de la tâche.

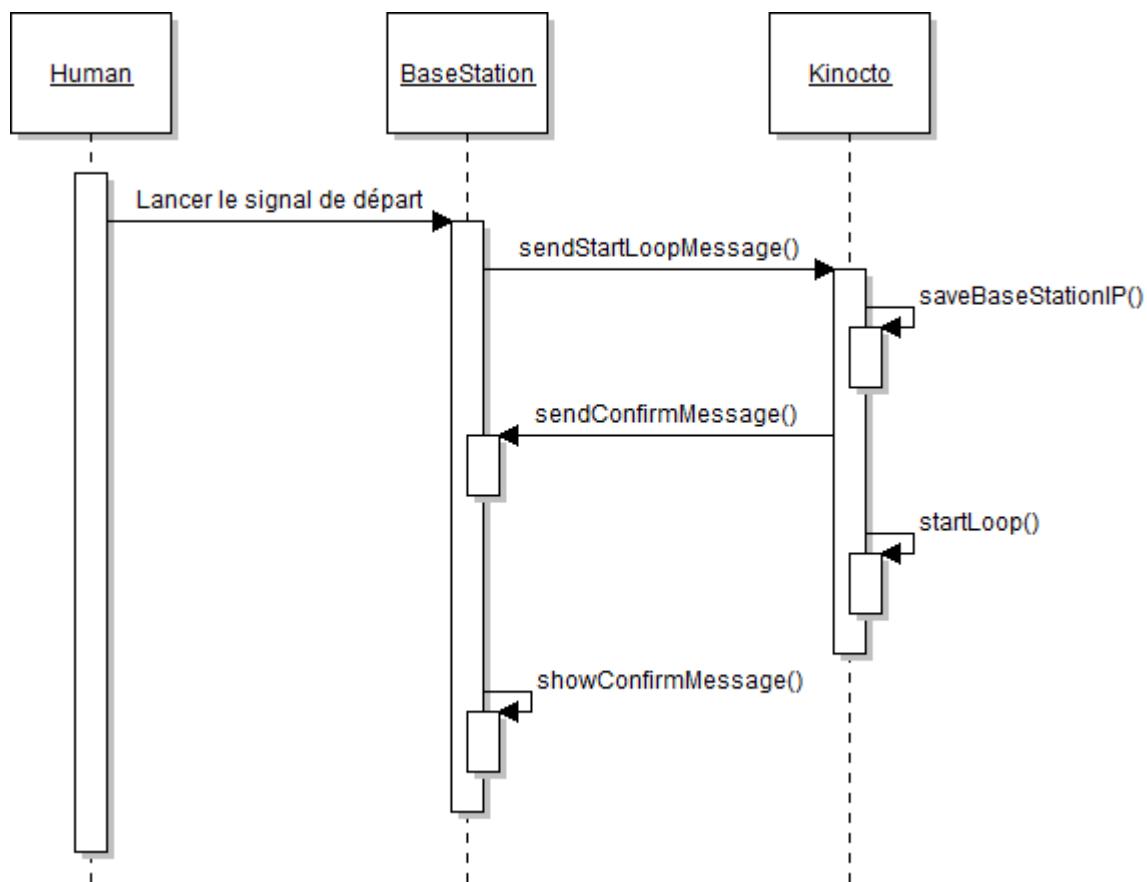


Figure 4.1 – Diagramme des séquences présentant les liens entre l'utilisateur et la kinocto

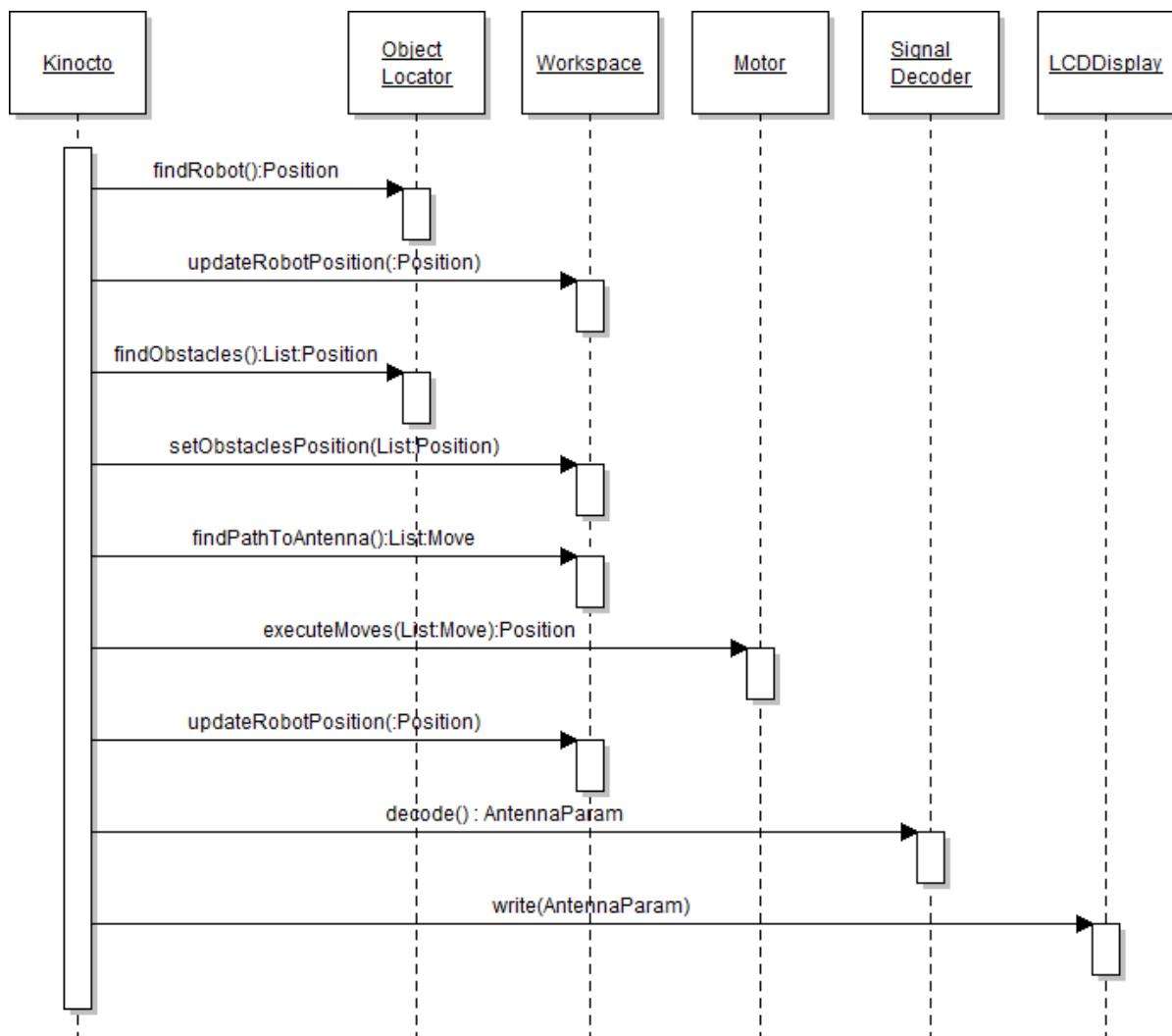


Figure 4.2 – Diagramme des séquences présentant les liens entre la kinecto et son environnement afin de déterminer sa position

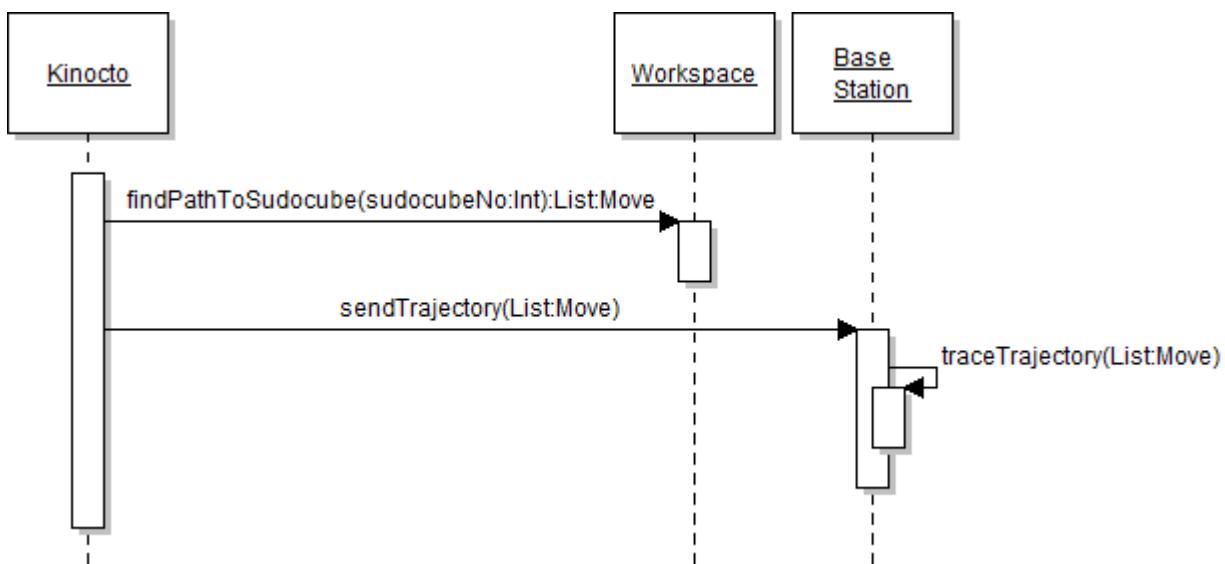


Figure 4.3 – Diagramme des séquences présentant les liens entre la kinocto et la station de base pour la transmission et l'affichage de la trajectoire optimale

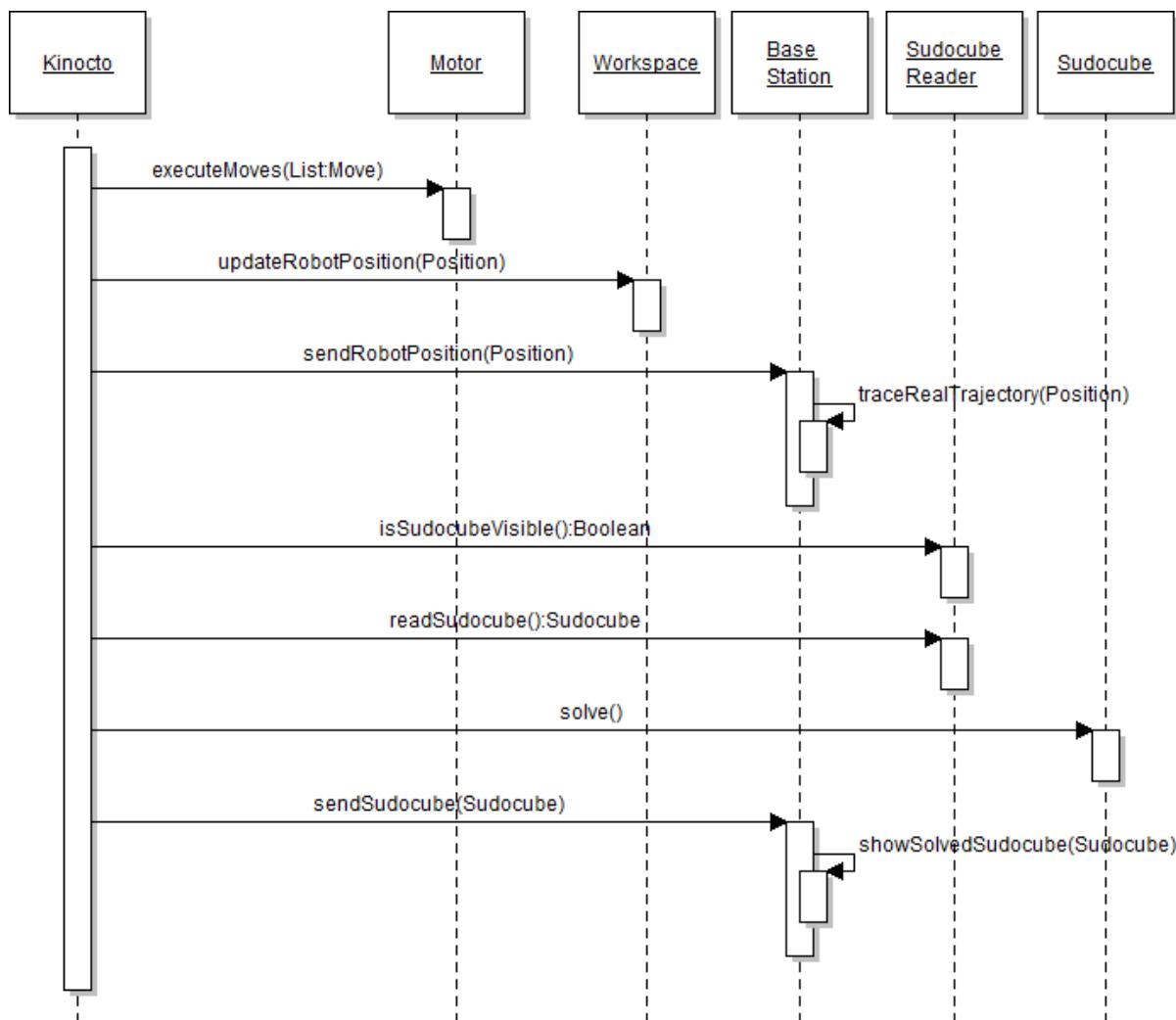


Figure 4.4 – Diagramme des séquences présentant les liens entre la kinocto et la table de jeu dans l'optique du déplacement de celle-ci à travers les obstacles vers la zone de lecture ainsi que les liens avec la résolution du sudocube

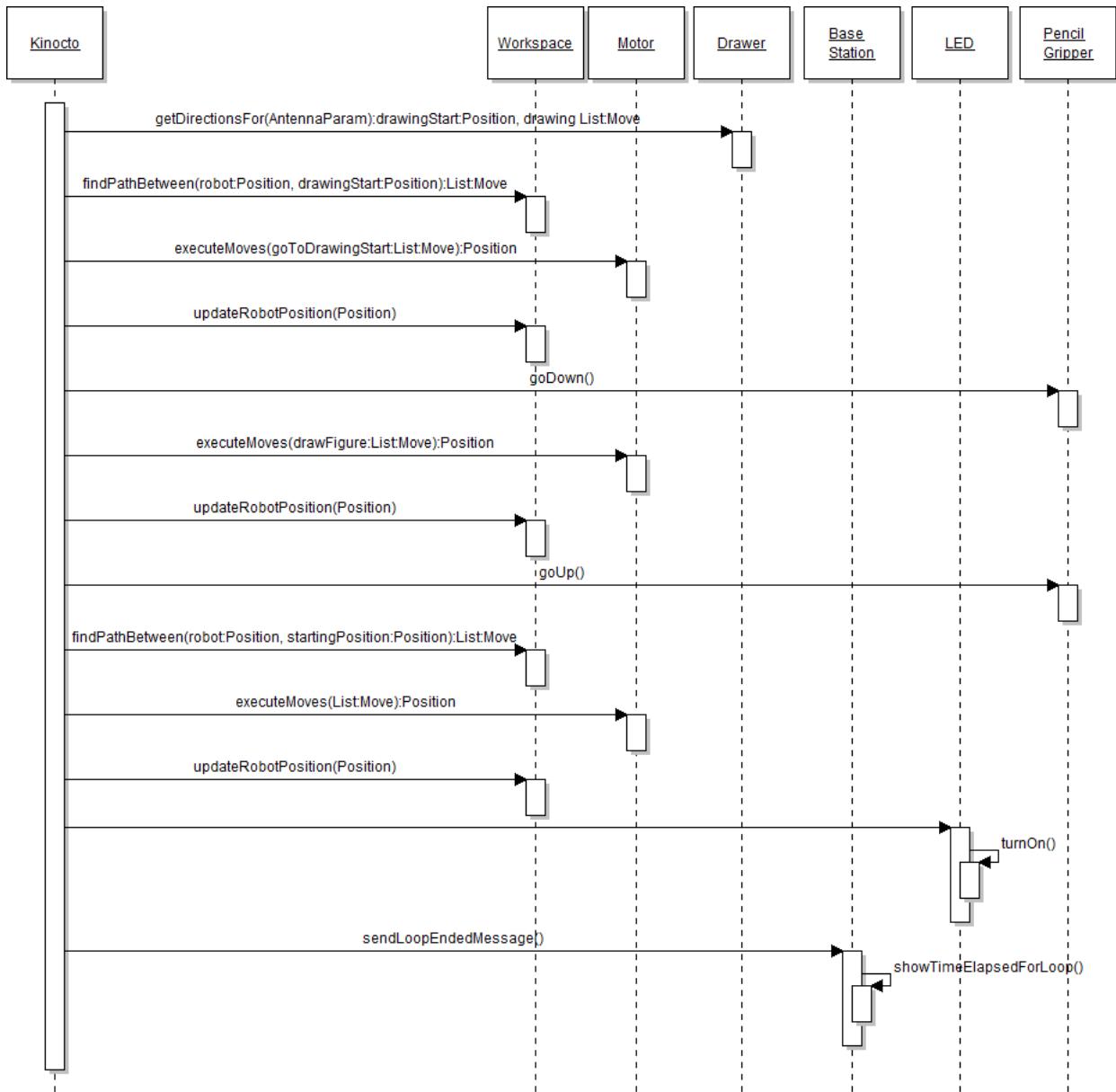


Figure 4.5 – Diagramme des séquences présentant les liens entre la kinocito et ses différents périphériques lors de la production du dessin et de la confirmation de la résolution de la tâche

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	15	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	20	25	Utilisateur de connecteurs protégés (d'un seul sens possible). Dispositifs de protection (fusibles), surdimensionnement des composantes d'alimentation	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 d'alimentation électrique (surtension ou baisse de tension, surchauffe ou ondulation importante)	4	Une surtension importante pourrait être destructrice pour le Mac mini ou le microcontrôleur, une surchauffe causerait un bris des systèmes d'alimentation et une ondulation importante pourrait rendre le système instable	Conséquences des risques 1 et 2 en plus de bris du Mac mini ou du microcontrôleur	10	100	Tests rigoureux et essais multiples pour vérifier la stabilité en température, en tension et en courant lors du fonctionnement	D. Thibodeau

Tableau 5.2 – Registre de risques partie 2

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
6	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.	D.Fournier
7	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
8	Problème de détection des chiffres et de représentation du sudo cube	5	Erreur dans la résolution du sudo cube (mauvais chiffres à la base)	Le chiffre dessiné ne sera pas le bon	5		Tests répétés pour un taux de succès de près de 100% lors de la détection des chiffres	P. Bourdages
9	Problème de résolution du sudo cube	5	Erreur dans la résolution du sudo cube, emballage de l'algorithme et arrêt de la tâche	Le chiffre dessiné ne sera pas le bon, le robot peut arrêter sa séquence d'opération avant le moment prévu	15		Tests répétés pour un taux de succès de près de 100% lors de la résolution du cube	O. Sylvain
10	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 de 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

Tableau 5.3 – Registre de risques partie 3

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
11	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
12	Asservissement des moteurs déficient	3	Un asservissement instable causera des dépassemens ainsi que des oscillations importantes de vitesse dans les roues. Dans le pire des cas, le robot sera incontrôlable.	Robot plus lent ou incapable de se déplacer, hausse du risque de toucher les obstacles, système non fonctionnel	10		Tests rigoureux dans le plus de conditions différentes possibles. Maximiser la robustesse en optant pour un asservissement avec une marge de phase élevé et bande passante assurant la stabilité.	P. Buhler
13	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
14	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
15	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

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'occur-rence (%)	Coût estimé du risque (\$)	Plan de réduction du risque	Responsable du risque
16	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
17	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
18	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
19	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
20	Problème lors du calcul de la trajectoire optimale «pathfinding»	3	La trajectoire trouvée est erronée ou n'est pas optimale	Une trajectoire erronée peut conduire à rencontrer les obstacles ou à être dans l'incapacité de compléter le trajet. Comme il est exigé un calcul de trajectoire optimale, une trajectoire ne l'étant pas est un manque à ce niveau	15		Tests unitaires et vérifications nombreuses, sous plusieurs contraintes.	O. Sylvain

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	Tests pratiques	Robot fonctionnel	Effectuer l'ensemble des dessins 3 fois et mesurer l'écart maximal
				Ruban à mesurer	
				Gabarits des dessins	
				Crayon	
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	Vérifier la déviation maximale ainsi que la vitesse moyenne pour une dizaine de trajectoires différentes
				Ruban à mesurer	
				Crayon	
				Rapporteur d'angle	
Alimentation	Alimenter le mac-mini et le préhenseur à une tension de 24 volts	Fournir une tension de 24 volts CC avec des oscillations maximum de 1 volt	Tests de mesure	Mac mini	Vérifier la tension moyenne à la sortie de l'alimentation et l'amplitude des oscillations qui y sont superposées sur l'oscilloscope
				Oscilloscope	
				Sondes de mesure	
				Robot fonctionnel	
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 0.2 volt	Tests de mesure	Oscilloscope	Vérifier la tension moyenne à la sortie de l'alimentation et l'amplitude des oscillations qui y sont superposées sur l'oscilloscope
				Sondes de mesure	
				Robot fonctionnel	
				Oscilloscope	
Récepteur manchester	Recevoir le signal manchester et le rendre utilisable par le micro-contrôleur	Fournir un signal carré 0-5 volts à l'image du signal reçu	Tests de mesure	Antenne fonctionnelle	Vérifier la présence du signal carré en sortie du module de réception lorsque le robot est à proximité de l'antenne
				Sondes de mesure	
				Robot fonctionnel	
				Oscilloscope	
Préhenseur	Monter et descendre le crayon pour passer en mode écriture ou arrêt d'écriture	Descendre le crayon lorsqu'une commande 5V est appliquée et le remonté lorsque la commande 0V est appliquée	Test pratique	Antenne fonctionnelle	Vérifier que le crayon descend lorsqu'on applique la commande de 5V au circuit et qu'il remonte lorsqu'on la relâche
				Sondes de mesure	
				Robot fonctionnel	
				Source 24V CC	
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	Vérifier que la DEL verte allume lorsqu'on applique une commande de 5V à son circuit
				Source 5V CC	
				Robot fonctionnel	
				Robot fonctionnel	
Positionnement de la caméra	Garder la camera en position malgré les perturbations extérieures	Replacer la camera à sa position de départ lorsqu'on allume l'alimentation 5V	Test de manipulation	Robot fonctionnel	Déplacer la camera lorsque l'alimentation est fermé et vérifier qu'elle reprend sa position lorsqu'on actionne l'alimentation 5V

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	Mesurer le temps de charge pour une opération de 10 minutes
				Testeur de batterie	
				Chargeur	
				Chronomètre	
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	Vérifier que les informations affichées sont lisibles et qu'elles concordent avec les informations décodées qui se trouvent dans la mémoire du microcontrôleur.
				Câble USB	

Tableau 6.3 – Plan de tests côté logiciel

Fonctionnalité	Sous-fonctionnalité	Exigence	Méthode de vérification	Équipement requis	Méthode d'analyse
Vision numérique	Détecter les obstacles	La position des obstacles obtenue à l'aide de la Kinect doit être précise au centimètre près.	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 avec le point (0,0) de la table pour différentes position d'obstacles avec et sans obstruction et la comparer avec la position vu par le logiciel.
	Détecter le robot	La position du robot obtenue à l'aide de la Kinect doit être précise au centimètre près		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 avec le point (0,0) de la table pour différentes position d'obstacles avec et sans obstruction et la comparer avec la position du robot vu par le 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	
Traitement numérique	Résoudre le sudocube	Être capable de résoudre des sudocube de 7 chiffres et plus		Ordinateur	
Vision numérique	Lire le sudocube	Le robot doit être situé entre (25.4 et 38.4)cm de distance par rapport au sudocube pour bien lire les informations	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.

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 de calcul(s)	Test	
		Précision (cm)	Test	
	Décoder le signal d'antenne	Nombre de périodes nécessaires	Test	
	Choisir le cube selon le signal d'antenne	Démonstration		
	Résoudre le sudocube	Temps de calcul (s)	Test	
		Chiffres initiaux minimum	Test	
Communication	Recevoir signal d'antenne	Test		
	Communiquer entre le robot et la station de base	Vitesse(Mo/s)	Démonstration	
		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	
	Transmettre les images de la Kinect vers le Mac mini	Taux de transfert (Images/s)	Démonstration	
	Transmettre les images de la caméra vers le Mac mini	Taux de transfert (Images/s)	Démonstration	
	Contrôler la position de la caméra	Temps de réponse(s)	Test	
		Précision(°)	Test	
	Commander le préhenseur du crayon	Temps de réponse(s)	Test	

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 et démonstration	
		Vitesse (cm/s)		
Alimentation	Utiliser une pile rechargeable	Énergie (Wh)	Inspection	
		Courant maximal (A)	Inspection	
		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	
	Alimenter les différents périphériques	Puissance (W)	Test	
		Ondulation de tension (mV)	Test	
Affichage	Afficher message d'initiation de la tâche		Démonstration	
	Afficher le cube résolu ainsi que la case rouge		Démonstration	
	Allumer la DEL lorsque tâche complétée		Test	
	Afficher la trajectoire optimale calculée		Démonstration	
	Afficher la position et trajectoire réelle	Temps d'actualisation (s)	Démonstration	
	Afficher message de fin		Démonstration	
	Afficher sur le LCD		Démonstration	

Chapitre 8

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"

Exigences du client	Fonctionnalités																	
	Vision numérique							Traitement numérique										
	Déetecter obstacles			Localiser le robot				Lire le cube			Calculer la trajectoire optimale		Contrôler le robot pour le dessin		Décoder le signal d'antenne	Choisir le cube selon le signal d'antenne	Résoudre le sudocube	
Temps de calcul (s)	Précision en X (cm)	Précision en Z (cm)	Temps de calcul (s)	Précision en X (cm)	Précision en Z (cm)	Précision angulaire (°)	Temps de calcul (s)	Distance min (m)	Distance max(m)	Temps de calcul (s)	Nombre de déplacement maximum	Temps de calcul (s)	Précision (cm)	Nombre de périodes nécessaires	Temps de calcul (s)	Chiffres initiaux minimum		
Être autonome pendant un minimum de 10 minutes	2	4	4	2	5	5	4	5	3	1	2	2	3	3	2	4	2	1
Se déplacer selon la trajectoire optimale	3	5	5	3	5	5	5				5	5			3	3		
Effectuer une séquence complète en moins de 10 minutes	5	3	3	5	3	3	3	5	3	1	5	3	5		3	3	2	
Alimenter le Mac mini avec une tension de 22V à 30V et ondulation de tension inférieure à 200 mV																		
Résoudre sudocube								5	3	1						5	5	
Dessiner le chiffre selon le signal d'antenne dans une zone prédéfinie (jaune) avec une précision de $\pm 1\text{cm}$													5					
Éviter les obstacles ainsi que les murs de la table	3	5	5	3	5	5	5				3	5						
Décoder le signal d'antenne														5				
Analyser le bon cube selon le signal d'antenne															5	5	1	
Utiliser la communication sans fil																		
Concevoir un système de préhension pour le crayon													3					
Afficher position réelle				3	5	5	5											
Afficher la trajectoire optimale	3	5	5	3	5	5	5				3	5			3	5		
Afficher le cube résolu et le chiffre de la case rouge sur la base								5	2	2						5	5	
Allumer une DEL lorsque tâche terminée																		
Afficher message de fin																		
Afficher message de départ																		
Afficher trajectoire réelle avec un délai maximum de 15s				3	5	5	5											
Afficher les informations sur le robot																		
Le robot doit se retirer de la zone de dessin une fois celui-ci terminée																		
Respecter un budget de 250\$												3	5	3				
Respecter l'échéancier	3	5	5	3	5	5	3	5	1	1	3	3	1	5	3	3	5	3

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

	Fonctionnalités										Déplacement			
	Communication													
	Recevoir le signal d'antenne	Communiquer entre le robot et la station de base	Communiquer entre le Mac mini et le microcontrôleur	Commander les moteurs		Transmettre les images de la Kinect vers le Mac	Transmettre les images de la caméra vers le Mac	Contrôler la position de la caméra	Commander le préhenseur du crayon					
Exigences du client	Vitesse (Mo/s)	Latence (ms)	Vitesse (bits/s)	Précision (cm)	Temps de réponse (s)	Taux de transfert (images/s)	Taux de transfert(images/s)	Temps de réponse (s)	Précision (degré)	Temps de réponse (s)	Résolution (Degrés)	Vitesse (m/s)		
Être autonome pendant un minimum de 10 minutes	4	3	3	5	5	5	5	5	1	1	4	3	1	
Se déplacer selon la trajectoire optimale	3			3	5	5	5	2				5	5	
Effectuer une séquence complète en moins de 10 minutes	5			3	4	3	5					5	5	
Alimenter le Mac mini avec une tension de 22V à 30V et ondulation de tension inférieure à 200 mV														
Résoudre sudocube								3	1	1				
Dessiner le chiffre selon le signal d'antenne dans une zone prédéfinie (jaune) avec une précision de $\pm 1\text{cm}$				5	5	5	3				5	5	5	
Éviter les obstacles ainsi que les murs de la table				5	3	3	5	4				5		
Décoder le signal d'antenne				2										
Analyser le bon cube selon le signal d'antenne	5													
Utiliser la communication sans fil		5	5				5							
Concevoir un système de préhension pour le crayon											5			
Afficher position réelle		5	5	5			5							
Afficher la trajectoire optimale	3	5	2	3			5							
Afficher le cube résolu et le chiffre de la case rouge sur la base		5	2					3						
Allumer une DEL lorsque tâche terminée				5										
Afficher message de fin		5	1											
Afficher message de départ		5	1											
Afficher trajectoire réelle avec un délai maximum de 15s		5	5	5			5							
Afficher les informations sur le robot				5										
Le robot doit se retirer de la zone de dessin une fois celui-ci terminé				5	5	1						5		
Respecter un budget de 250\$	3			1	5	1					2			
Respecter l'échéancier	3	3	2	3	5	2	5	3	1	1	5	3	3	

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

Exigences du client	Fonctionnalités												
	Alimentation				Affichage								
	Utiliser une pile rechargeable		Alimenter les moteurs	Alimenter le Mac		Alimenter les différents périphériques		Afficher message d'initiation de la tâche	Afficher le cube résolu ainsi que la case rouse	Allumer la DEL lorsque tâche complétée	Afficher la trajectoire optimale	Afficher la position et trajectoire réelle	Afficher message de fin
Exigences du client	Énergie (Wh)	Courant maximal (A)	Durée d'une charge (min)	Puissance (W)	Puissance (W)	Ondulation de tension (mV)	Puissance (W)	Ondulation de tension (mV)	Puissance (W)			Temps d'actualisation (s)	
Être autonome pendant un minimum de 10 minutes	5	3	5	3	3	3	3	5					
Se déplacer selon la trajectoire optimale	5	3	1	5	5	5	3	3					
Effectuer une séquence complète en moins de 10 minutes	5	3	2	5	5	5	5	5					
Alimenter le Mac mini avec une tension de 22V à 30V et ondulation de tension inférieure à 200 mV	5	5	3		5	5	3	3					
Résoudre sudo-cube	5	5	2		5	5	5	5					
Dessiner le chiffre selon le signal d'antenne dans une zone prédéfinie (jaune) avec une précision de $\pm 1\text{cm}$	3	3	1	3	3	3	3	3					
Éviter les obstacles ainsi que les murs de la table	3	3	1	3	3	3	3	3					
Décoder le signal d'antenne	5	3	1		3	3	5	5					3
Analyser le bon cube selon le signal d'antenne	5	2	1		1	2	3	3	3	3	3		
Utiliser la communication sans fil													
Concevoir un système de préhension pour le crayon													
Afficher position réelle					3	2	3	3				5	5
Afficher la trajectoire optimale	3	1	1		1	1	3	3			5		
Afficher le cube résolu et le chiffre de la case rouge sur la base	1	1	1		1	1				3			
Allumer une DEL lorsque tâche terminée	3	1	1		3	2	5	5		5			
Afficher message de fin	3	1	1		3	2							5
Afficher message de départ	3	1	1		3	2			5				
Afficher trajectoire réelle avec un délai maximum de 15s					3	2	3	3				5	
Afficher les informations sur le robot	3	1	1			2	3	3					5
Le robot doit se retirer de la zone de dessin une fois celui-ci terminé	3	1	1	5	5	5	5	5					
Respecter un budget de 250\$	5	5	2	2	3	3	5	5					
Respecter l'échéancier	3	3	3	3	3	3	3	3	1	1	1	1	1

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.

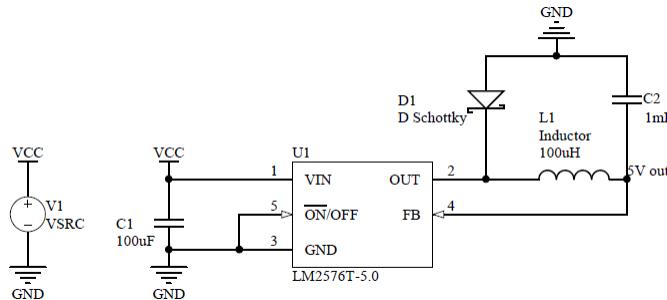


Figure 10.1 – Figure présentant les plans de l'alimentation 5V pour les périphériques

10.2 Alimentation du Mac mini

L'alimentation du Mac mini a été réalisé au moyen d'un circuit de type Boost réglable, acheté déjà monté. Les plans ne sont pas disponibles, mais une photo du dispositif l'est à la figure 10.3.

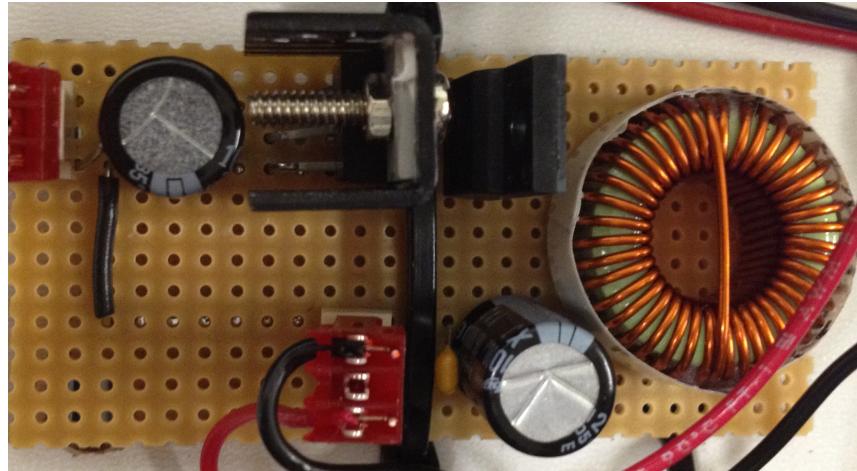


Figure 10.2 – Figure présentant une photo de l'alimentation 5V pour les périphériques

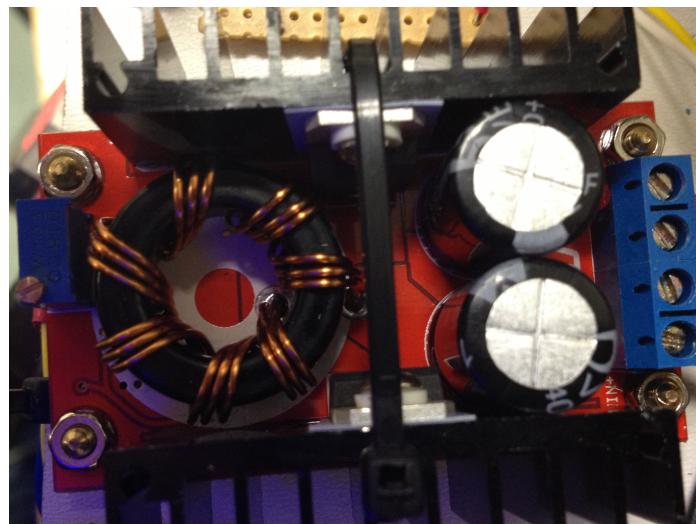


Figure 10.3 – Figure présentant une photo de l'alimentation 24V pour les périphériques

10.3 Asservissement des moteurs

10.3.1 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 logiciellement est présentée à la figure 10.4

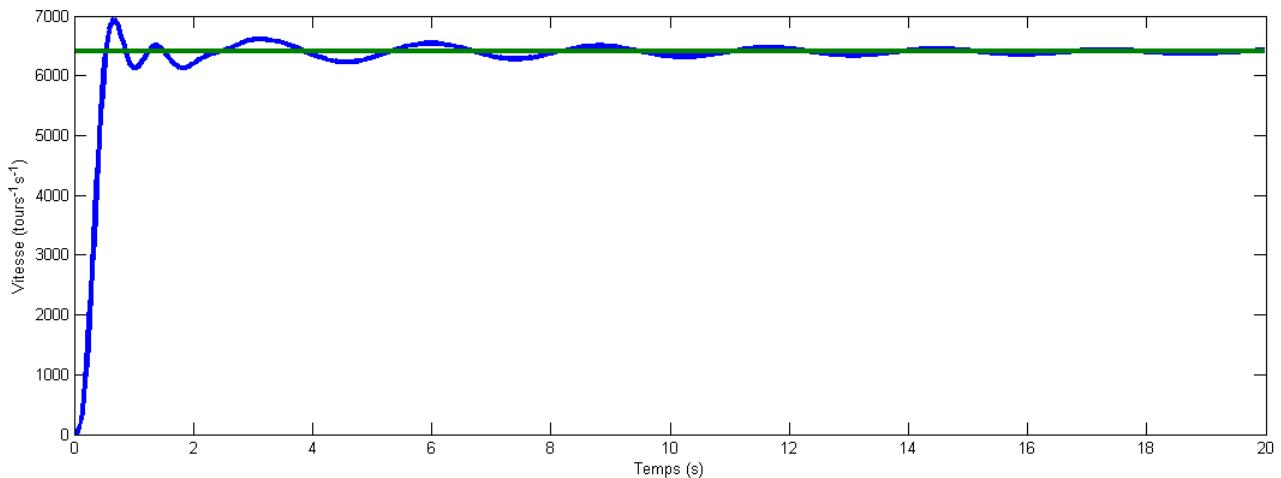


Figure 10.4 – Figure présentant la réponse à un échelon de consigne de $6400[\text{tours}^{-1}\text{s}^{-1}]$ du système régulé au moyen du régulateur optimisé dans simulink

10.3.2 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 10.5. À noter que les consignes en $\text{tours}^{-1}\text{s}^{-1}$ indique 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 10.3.3.1 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 10.6 présente

ce phénomène pour une consigne de $6400 \text{ (tours}^{-1}\text{s}^{-1}\text{)}$ 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 remplissant les critères de stabilité, de vitesse et de dépassement souhaités. Afin d'améliorer la vitesse des déplacements, différents PID ont été ajustés selon différentes vitesses d'asservissement. On compte 3 de ces PID pour les déplacements unilatéraux et 1 déplacement réservé pour les mouvements reliés au dessin. La différence est que la vitesse de dessin est beaucoup plus basse que celle du déplacement usuel en vue de permettre une plus grande précision. Le moteur n'étant plus très linéaire à ces faibles vitesses (roulement et zone morte très limitants) et le système étant incapable de démarrer de lui même pour procéder à l'identification, le PID a donc été ajusté de manière manuelle, par itérations. Les PID sont fonctionnels et les systèmes 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.

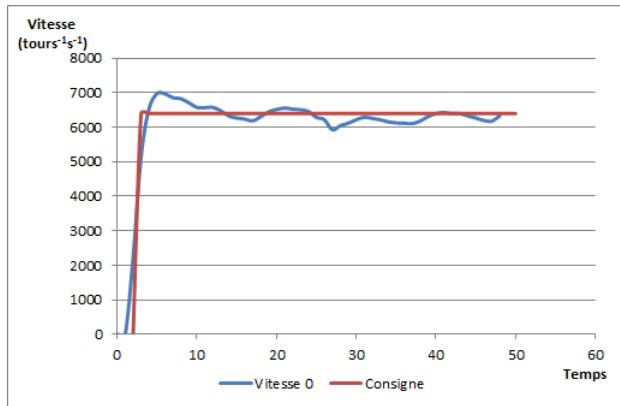


Figure 10.5 – Figure présentant la réponse à un échelon de consigne de $6400[\text{tours}^{-1}\text{s}^{-1}]$ du système régulé au moyen du régulateur réel et de la réponse en position associée

10.3.3 Implantation électronique

10.3.3.1 Les prises de mesures

Pour prendre des mesures de positions et de vitesses pour effectuer l'asservissement, nous utilisons deux QEI (Quadrature Encoder Interface) matériels qui sont sur le microcontrôleur. 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 senseurs 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

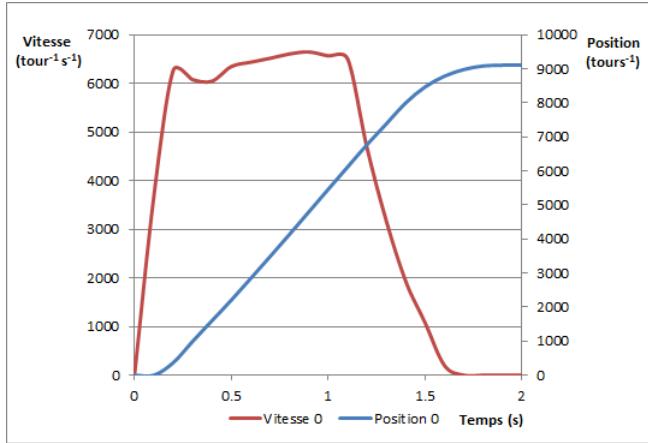


Figure 10.6 – Figure présentant la réponse à un échelon de consigne de $6400[\text{tours}^{-1}\text{s}^{-1}]$ du système régulé au moyen du régulateur réel et de la réponse en position associée

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, nous avons réalisé deux interfaces d'encodeurs en quadratures logicielles 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 10.7. 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.

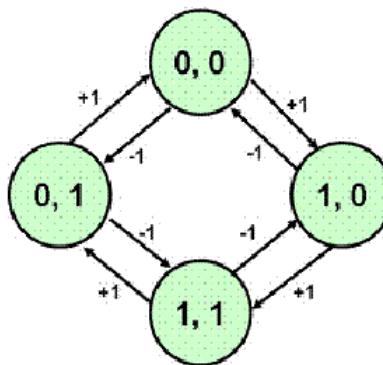


Figure 10.7 – Machine à états servant d'interface d'encodeur en quadrature (Cytron. Quadrature Encoder. <http://tutorial.cytron.com.my/2012/01/17/quadrature-encoder/>, consulté le 3 mars.)

10.3.3.2 PID

Le PID utilisé pour l'asservissement est une courte fonction réalisée dans le microcontrôleur. Cette fonction se retrouve dans l'annexe A.1.1. À prendre note que l'argument "I" est l'intégrale de l'erreur et "dt" est la différence en temps entre chaque exécution de l'asservissement des moteurs. 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.

10.3.3.3 Ajustement de la vitesse selon la positon

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. Cette 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 décélérations discutées dans la section 10.3.2 seront identifiées.

10.4 Servomoteurs de la Webcam

Pour contrôler les servomoteurs de la webcam, nous utilisons le contrôleur Micro Maestro de Pololu. Pour le programmer/configurer nous utilisons l'application "Maestro Control Center". Les servomoteurs et le Micro Maestro sont alimentés par l'alimentation 5V. De plus, les servomoteurs reprennent une position pré-déterminée 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 Webcam par rapport au sol. Ce changement est nécessaire pour détecter l'orientation du robot.

10.5 Extraction des sudocubes

10.5.1 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.

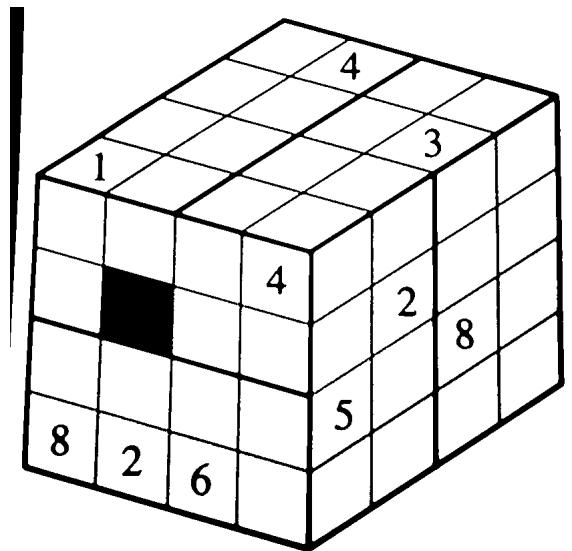


Figure 10.8 – Figure présentant un sudocube traité afin d'extraire les informations des cases



Figure 10.9 – Figure présentant un exemple d'images extraites des cases du sudocube et normalisées avant d'être passées au lecteur de chiffres

10.5.2 Les solutions retenus/considérés

Pour le lecteur de chiffres, deux solutions furent considérées : 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. Nous avons jugé que cette solution offre des fonctionnalités superflues en plus d'imposer une dépendance supplémentaire au projet. Puisqu'OpenCV est une dépendance obligatoire et que la méthode KNearest nous permet d'obtenir le même résultat qu'avec Tesseract nous avons choisi cette solution.

10.5.3 Les moyens utilisé 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, testé avec un lot de 60 nouvelles images par chiffres afin de vérifier sa robustesse.

10.6 Solveur de Sudocubes

« « « < HEAD =====

10.6.1 Composantes du solveur

L'algorithme qui solutionne les sudocubes a été développé en C++. L'utilisation de la programmation par contraintes (CSP) aurait été plus simple à planter et nous aurait fait économiser du temps, mais nous n'avions pas envisagé cette solution au départ. Nous avions déjà un prototype au moment notre prise de conscience de l'existence de ce type de programmation. Le développement du solveur est actuellement terminé et il fonctionne. » » » > d365caa0e2a74da5d8f8e17cc5e66e531cada395

L'algorithme qui résout les sudocubes a été développé en C++. L'utilisation de la programmation par contraintes (CSP) aurait été plus simple à planter et nous aurait fait économiser du temps, mais nous n'avions pas envisagé cette solution au départ. Nous avions déjà un prototype au moment de notre prise de conscience de l'existence de ce type de programmation. Le développement du solveur est actuellement terminé et il fonctionne. Il utilise une structure de données afin de garder en mémoire le sudocube qui n'est en fait qu'un simple tableau à trois dimensions. Elle devient donc en même temps une interface pour la vision afin de stocker ses données obtenues lors de l'extraction du sudocube. Les stratégies utilisées pour résoudre le sudocube sont les mêmes qui sont utilisées dans le monde des sudokus normaux, mais adaptées aux sudocubes. Plus spécifiquement, nous avons utilisé les stratégies (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, nous utilisons la technique de force brute. Nous pensons que c'est une solution acceptable puisque la probabilité d'occurrence de la situation est faible. L'algorithme peut trouver une solution à 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.

«««< HEAD =====

Si ces stratégies ne suffisent pas à solutionner le sudocube, nous utilisons la technique de force brute. Nous pensons que c'est une solution acceptable puisque il sera plutôt rare que cette situation surviendra.

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

10.6.2 Stratégie de test

Pour tester entièrement notre classe solveur, il faudrait tester chacune des stratégies utilisées unitairement. Toutefois, si nous avions voulu tester chacune des stratégies, il aurait fallu adopter une architecture différente, puisque ces stratégies sont des méthodes privées de la classe solveur. Trois choix s'offraient alors à nous.

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

Sinon, nous aurions pu nous créer un nouvel objet duquel hériterait chacune des stratégies. 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, nous pourrions préserver l'encapsulation en ne testant tout simplement pas chacune des stratégies. Nous testerions uniquement la méthode "solve" en passant au solveur des sudokubes et en observant le résultat, ce qui ressemble plus à un test d'intégration.

Dans notre cas, nous avons préféré utiliser la troisième option. Nous pensons que de tester ainsi couvrira suffisamment notre solveur. Selon nous, ce test est un bon compromis entre le respect du bris d'encapsulation et le fait de tester unitairement chacune des stratégies. Nous savons que la meilleure façon de procéder serait la deuxième. Par contre, dans le cadre du

projet, nous considérons que le temps qui serait investi dans l'écriture de ces tests est trop grand par rapport au gain que l'on ferait en qualité de code. Nous préférons mettre du temps ailleurs et y revenir au besoin.

10.7 Recherche de chemin

La fonctionnalité recherche de chemin n'en est encore qu'au stade prototype. Cette fonctionnalité n'a pas reçu beaucoup d'attention depuis le livrable 1. On pourrait donc se fier aux données énoncées dans ce livrable pour connaître l'état de la recherche de chemin : "L'algorithme A* a été choisi, car il est très rapide et facile à implémenter". Nous savons que la solution développée dans le prototype est viable, mais il reste du travail à faire pour que ce soit fonctionnel. 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.

»»»> d365caa0e2a74da5d8f8e17cc5e66e531cada395

10.8 Communication microcontrôleur - Mac mini

La communication entre le Mac mini a été réalisée avec un port série UART avec protocole RS-232 à travers le module ICDI du microcontrôleur LM3S9B92. Nous avons choisi ce protocole 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.

10.9 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. Parmis 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 de 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 implémentent une fonction de capture d'événement, nous avons choisi d'utiliser la méthode par timings. Pour l'instant, nous pouvons mesurer des temps entre transitions sur un signal interne au microcontrôleur généré par un PWM.

10.10 Orientation du robot

Pour la détermination d'angle et l'orientation du robot, le choix de la caméra embarquée est très recommandé pour sa précision comparé à la kinect.

Dans un premier temps, nous devons calibrer la caméra. Pour ce faire, deux méthodes s'offrent à nous : l'application de l'algorithme de Zhang ou la calibration avec Opencv. Dans ce cas nous retenons la première méthode, car malgré sa complexité, elle est plus efficace.

Ensuite, nous procédons à la détermination d'angle du robot par rapport à la ligne rouge déjà tracée sur la table. Cette partie est réalisée grâce à un traitement d'image supportant la variation de luminosité, d'ombres... ainsi que grâce à des règles de trigonométrie.

Enfin, la détermination de l'orientation du robot (N, S, E, O) est effectuée par la détection des 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.

10.11 Vision par Kinect

Pour aider au déplacement du robot, la Kinect a été utilisée pour 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 a fallu effectuer une translation et une rotation des données obtenues pour positionner avec exactitude les objets. Les 3 sections qui suivent expliquent en détail les différentes parties de l'algorithme de vision de la Kinect.

10.11.1 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 mètre 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 représenté par les ligne verte sur l'image 10.10. Toutefois, comme l'origine de notre représentation cartésienne de la table est situé sur le coin inférieur droit de la table et que la Kinect n'est pas situé à cet endroit, il est nécessaire d'obtenir la distance X et Y (lignes bleues sur l'image 10.10) 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 10.10) 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° , nous avons mis en oeuvre un logiciel qui permet la transformation des distances de la Kinect vers les composantes X et Y recherchée à l'aide de règles de trigonométrie simples. Cet algorithme permet d'obtenir avec une précision de plus ou moins 1cm pour n'importe quel point situé sur l'image obtenue avec la Kinect.

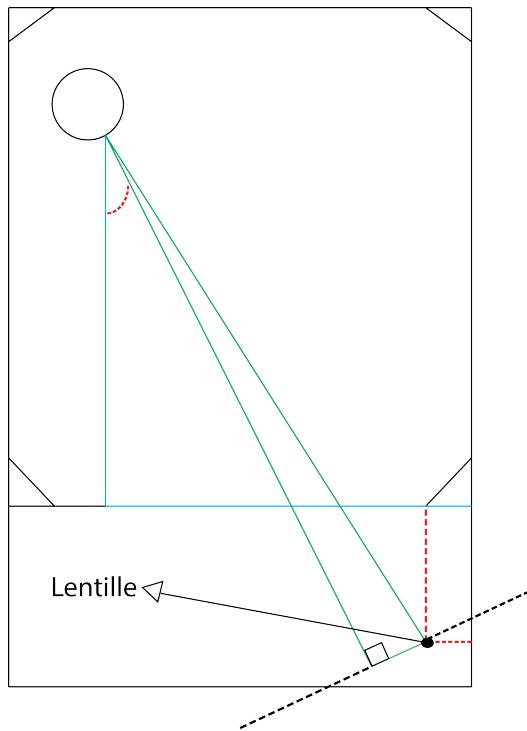


Figure 10.10 – Schéma représentant la table et les différentes mesures obtenues avec la Kinect pour un point quelconque

10.11.2 Détection des obstacles

Maintenant que nous pouvons obtenir n'importe quelle distance entre l'origine et un point quelconque sur l'image, nous avons créé l'algorithme de recherche d'obstacles. 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 et situé dans la zone prédefinie. Comme la Kinect est capable de retourner la hauteur de chacun des points sur l'image infrarouge,

il a été facile de trouver les obstacles de cette manière. De plus, à l'aide de simples calculs de statistiques, notre algorithme est capable de trouver les obstacles lorsqu'ils sont presque enlignés ou bien les obstacles avec une obstruction comme le robot devant eux. 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.

10.11.3 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, pour améliorer la précision de la détection du robot, nous allons mettre en place 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, nous devrions être capable 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, nous sommes capable de détecter le robot avec une incertitude de 5 cm en environ 30 ms si le robot n'est pas situé en arrière d'un obstacle.

10.12 Communication entre le Mac mini et la station de base

10.12.1 Composantes du système

Avant de pouvoir communiquer avec le robot, nous récupérons l'adresse IP du robot grâce à un script bash exécuté lors du démarrage de celui-ci. Le script récupère l'adresse IP et l'envoie sur le site pastebin.com.

Ensuite, nous utilisons ROS pour gérer la communication entre les différents nodes ROS. Une node c'est une application qui utilise l'API de ROS. Actuellement, nous avons testé l'envoi de messages au Mac mini pour démarrer la séquence du robot.

On peut voir sur le diagramme 10.11 la répartition des nodes sur les différents appareils ainsi que les appareils avec lesquels chaque node travaillent. Nous avons choisi de séparer le node du mac mini en deux afin de simplifier les tests pour les composantes qui communiquent avec le microcontrôleur. Grâce au système de communication de ROS on peut envoyer par ligne de commande des messages afin de déplacer le robot ou encore afficher un message sur l'écran LCD, et ce, sans passer par l'écriture d'une application de tests.

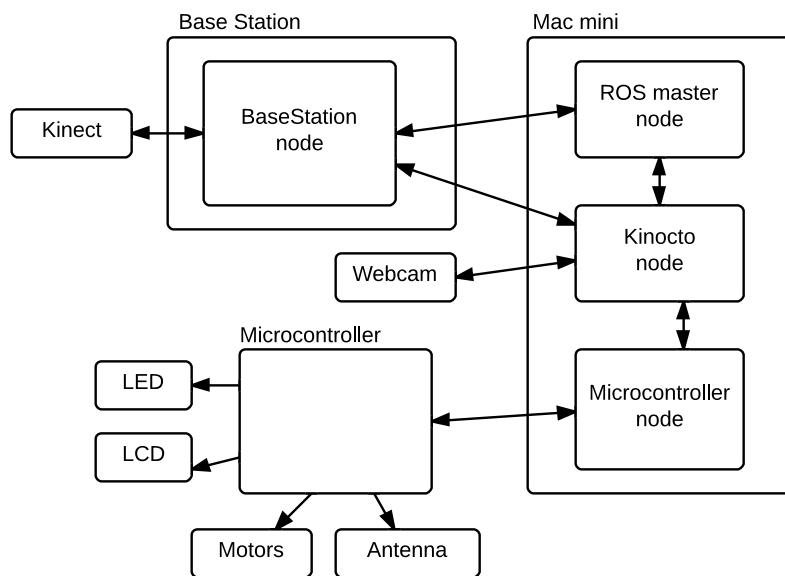


Figure 10.11 – Diagramme représentant la communication entre les nodes et les différents appareils

10.12.2 Les solutions retenues et considérées

Nous avons considéré deux solutions : POCO une librairie C++ et ROS un environnement de développement pour robot.

POCO permet d'envoyer des chaînes de caractères par TCP/IP. L'envoie de commandes avec POCO consiste en ces étapes : concaténer le nom d'une commande (choisit 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 logiciel qui nous 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 du projet ROS génère des struct en C pour chaque message. Ensuite, on utilise l'api de ROS pour définir le contenu des messages et envoyer les messages.

La courbe d'apprentissage est nettement 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 }
```