Livrable 2 - Robot Kinocto

Design III

présenté à

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

matricule	*nom*
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

Chapitre 1

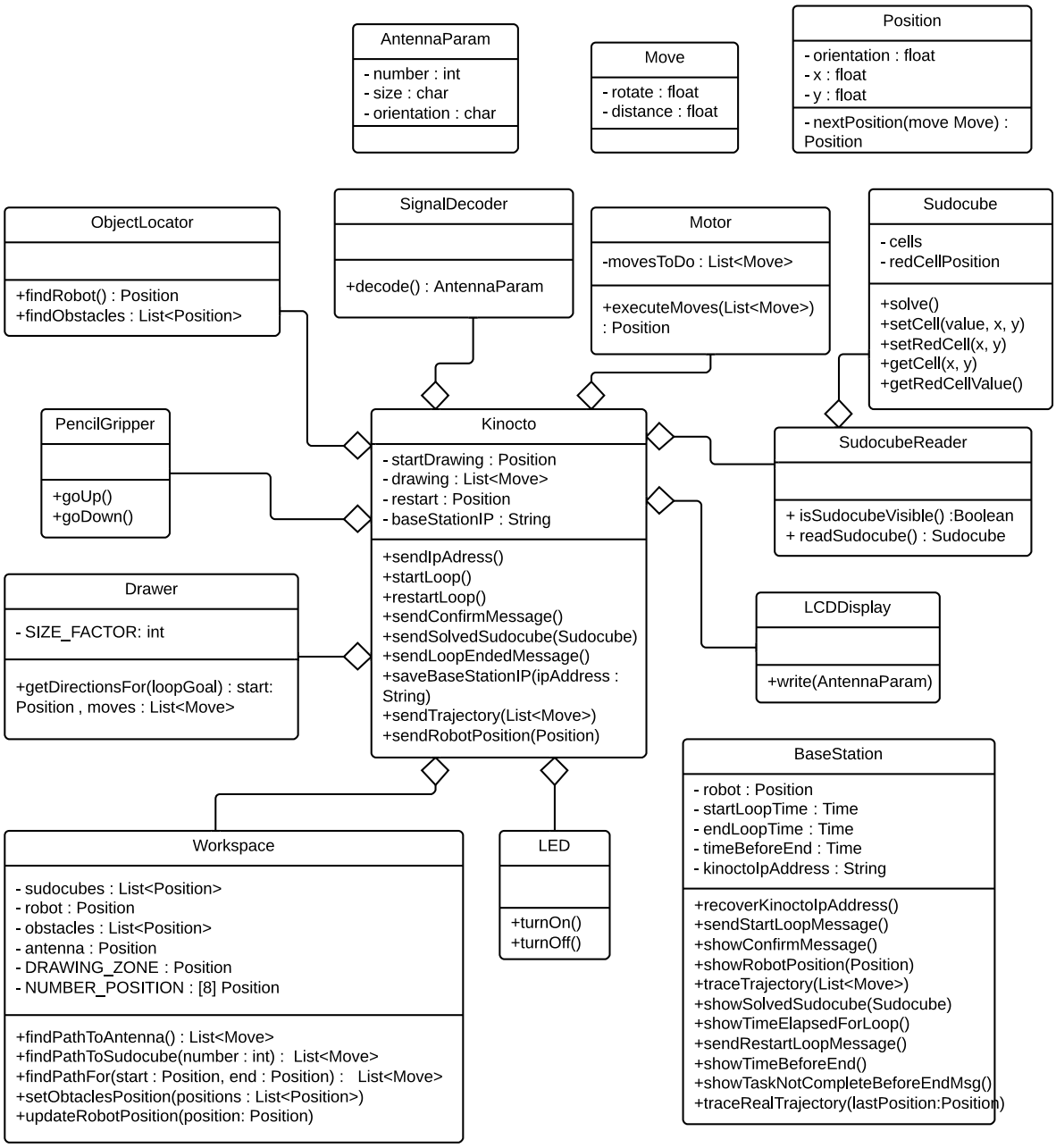
Diagramme des fonctionnalités

Chapitre 2

Diagramme physique

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.

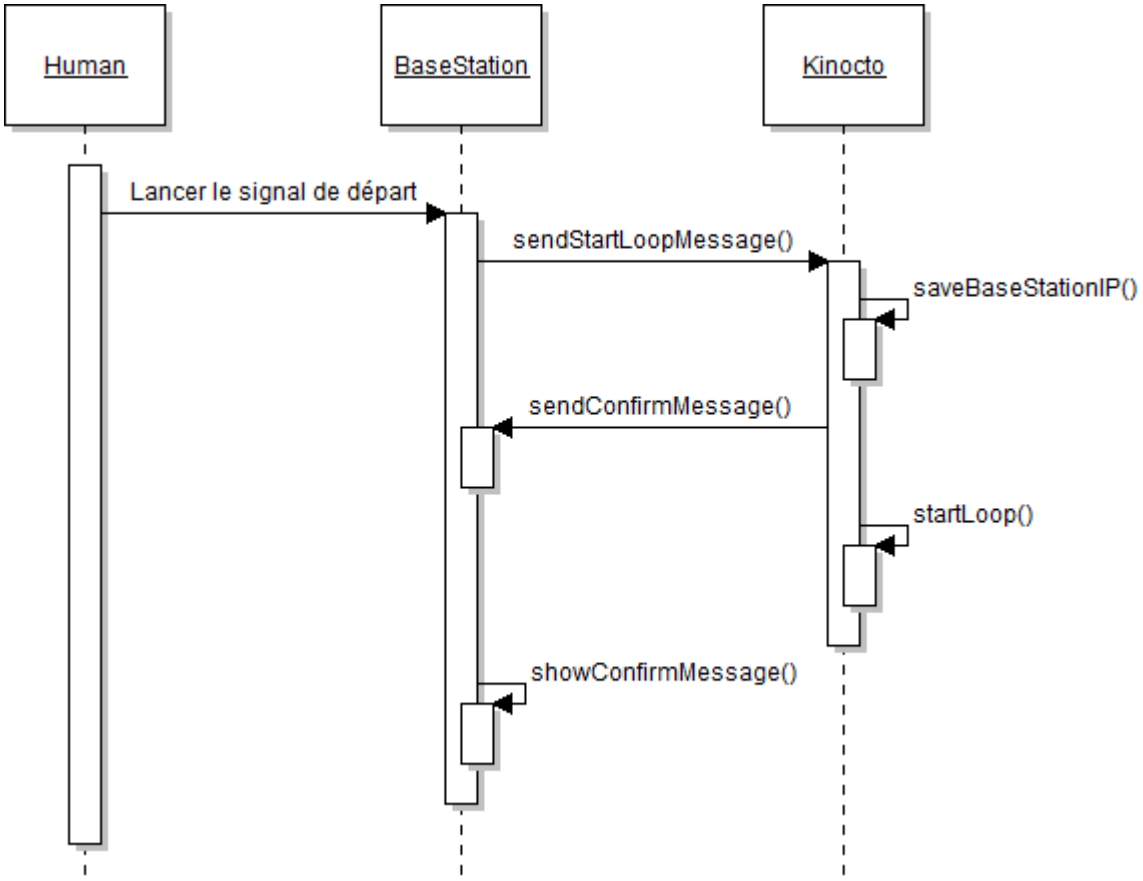


Figure 4.1 – Diagramme des séquences présentant les liens entre l'utilisateur et la kinocto

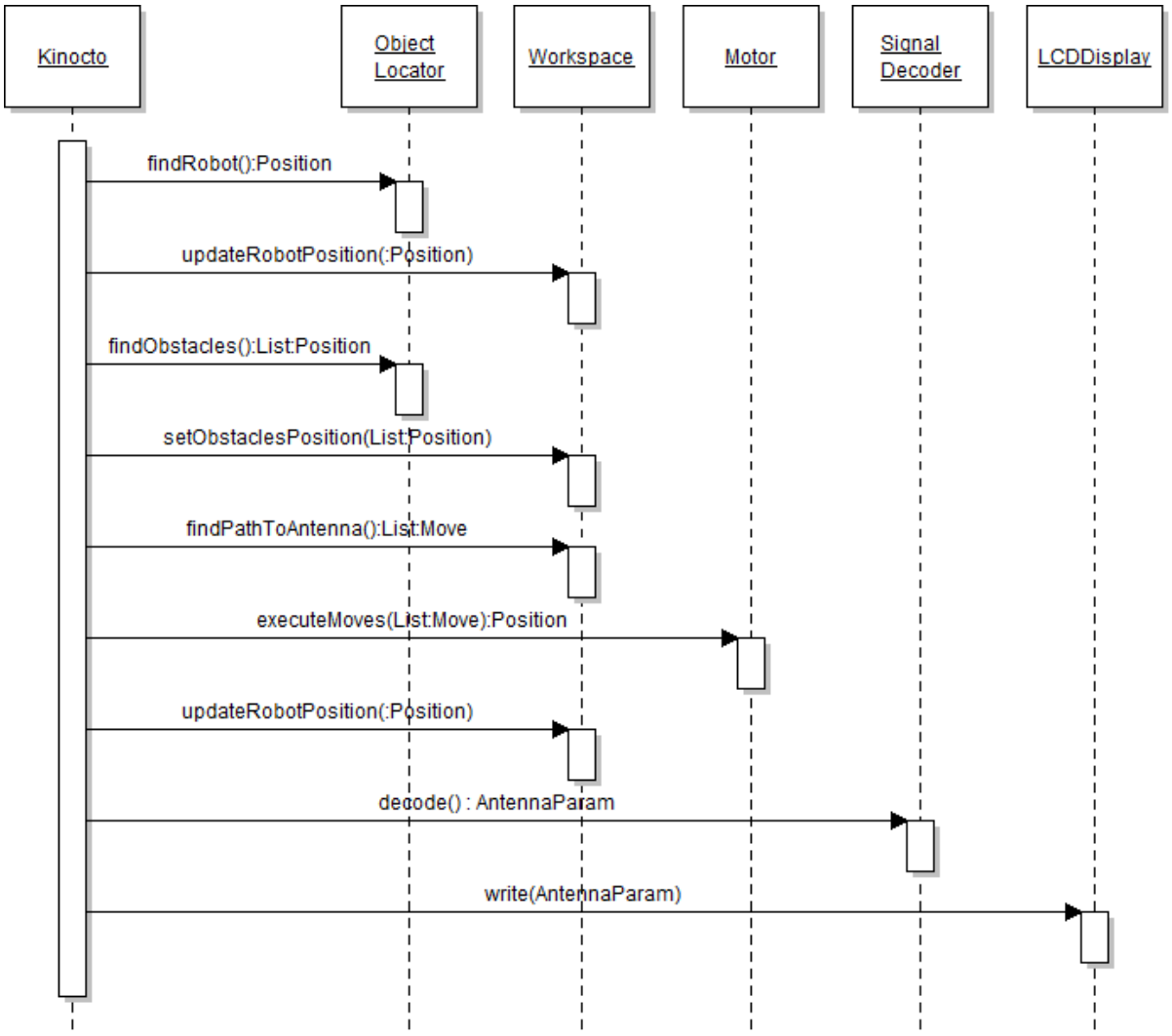


Figure 4.2 – Diagramme des séquences présentant les liens entre la kinocto et son environnement afin de déterminer sa position

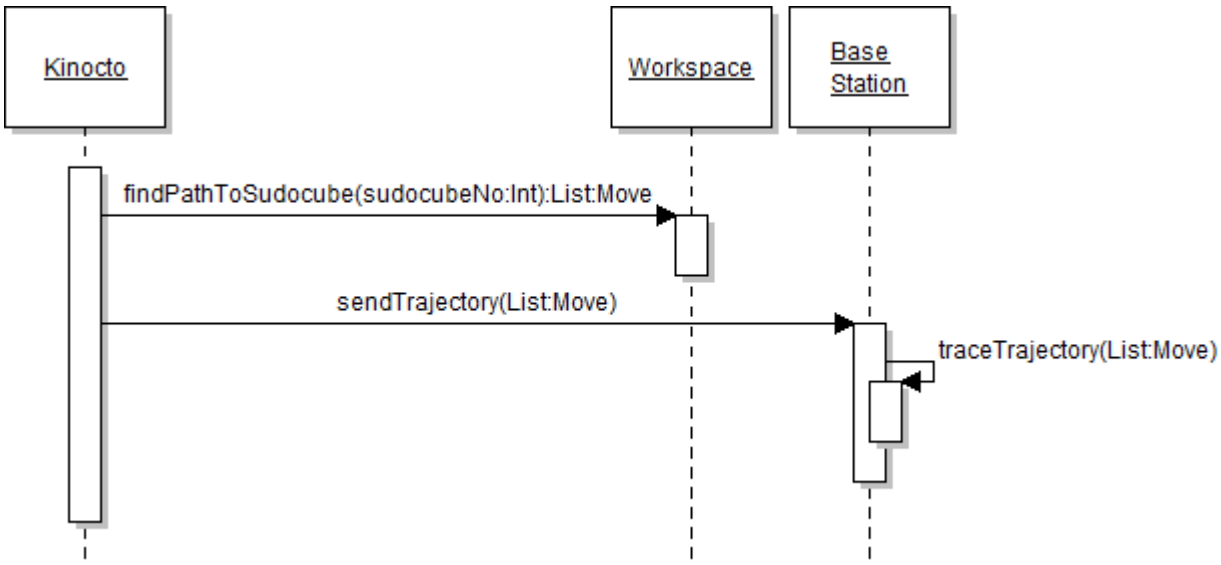


Figure 4.3 – Diagramme des séquences présentant les liens entre la kinocto et la station de base pour la transmission et l'affichage de la trajectoire optimale

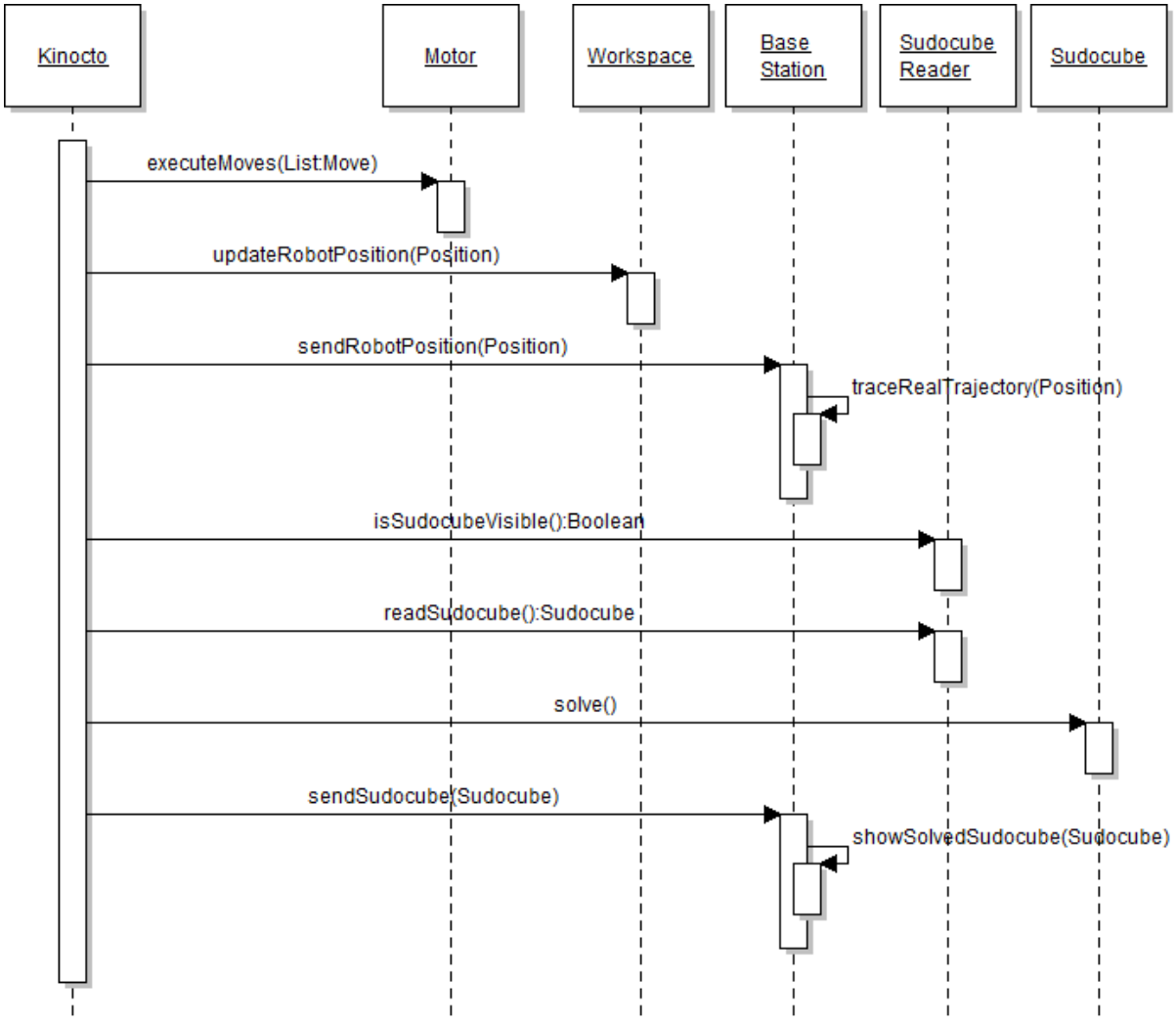


Figure 4.4 – Diagramme des séquences présentant les liens entre la kinocto et la table de jeu dans l'optique du déplacement de celle-ci à travers les obstacles vers la zone de lecture ainsi que les liens avec la résolution du sudocube

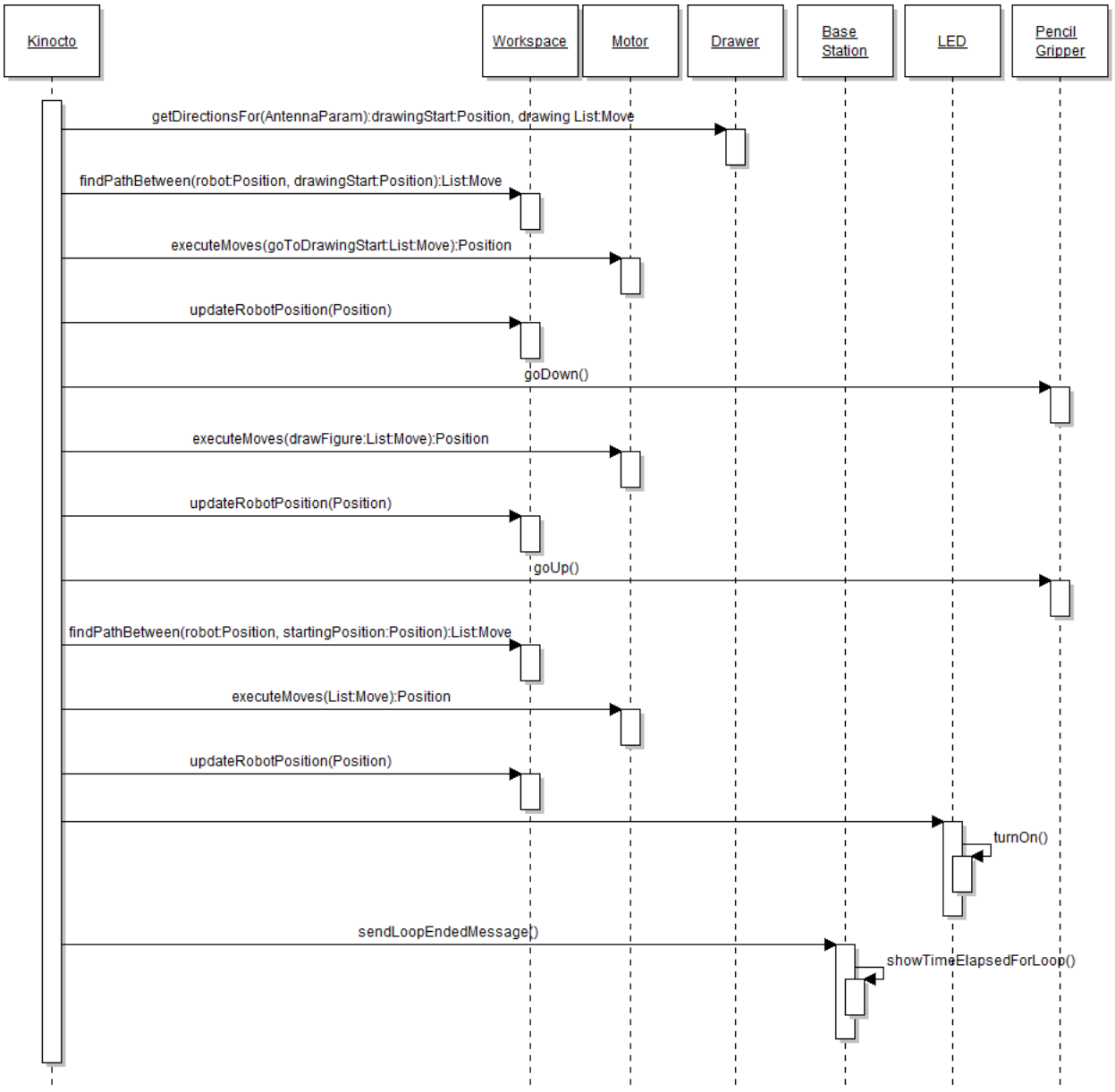


Figure 4.5 – Diagramme des séquences présentant les liens entre la kinocto et ses différents périphériques lors de la production du dessin et de la confirmation de la résolution de la tâche

Chapitre 5

Registre des risques

Chapitre 6

Plan de tests avec matrice de vérification des exigences intégrées

Chapitre 7

Plan d'intégration

Chapitre 8

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 8.1), communication et déplacement (présenté dans le tableau 8.2) ainsi qu'alimentation et affichage (présenté dans le tableau 8.3).

Tableau 8.1 – Description des propriétés fonctionnelles : section "Vision et Traitement Numérique"

	Fonctionnalités								
	Vision numérique			Traitement numérique					
	Détecter 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
Exigences du client	Temps de calcul (s)	Taux d'erreur (%)	Temps de calcul (s)	Temps de calcul (s)	Temps de calcul (s)	Taux d'erreur (%)			Temps de calcul (s)
Être autonome pendant un minimum de 10 minutes	3	2	5	2	3	3	2	4	2
Se déplacer selon la trajectoire optimale	5	5		5			3	3	
Effectuer une séquence complète en moins de 10 minutes	5	5	5	5	5			3	3
Alimenter le Mac mini avec une tension de 22V à 30V et ondulation de tension inférieure à 200 mV									
Résoudre sudocube			5						5
Dessiner le chiffre selon le signal d'antenne dans une zone prédéfinie (jaune) avec une précision de ± 1cm						5			
Éviter les obstacles	5	5							5
Analyser le bon cube selon le signal d'antenne									
Utiliser la communication sans fil									
Concevoir un système de préhension pour le crayon									
Afficher position réelle	5					5			3
Afficher la trajectoire optimale			5						
Afficher le cube résolu sur la base									
Allumer une DEL lorsque tâche terminée									
Afficher message de fin									
Afficher message de départ									
Afficher trajectoire réelle avec un délai maximum de 15s									
Afficher les informations sur le robot									
Respecter un budget de 250$							3		

Tableau 8.2 – Description des propriétés fonctionnelles : section "Communication et Déplacement"

	Fonctionnalités								
	Communication							Déplacement	
	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 caméra vers le Mac	Contrôler la position de la caméra	Commander le préhenseur du crayon	Se déplacer sans toucher aux obstacles	
Exigences du client		Vitesse (Mo/s)	Vitesse (bits/s)					Résolution (Degrés)	Vitesse (m/s)
Être autonome pendant un minimum de 10 minutes	4							3	1
Se déplacer selon la trajectoire optimale	3		3	4	2			5	5
Effectuer une séquence complète en moins de 10 minutes	5		3	4				5	5
Alimenter le Mac mini avec une tension de 22V à 30V et une ondulation de tension inférieure à 200 mV									
Résoudre sudocube					3	1			
Dessiner le chiffre selon le signal d'antenne dans une zone prédéfinie (jaune) avec une précision de ± 1cm			5	3			5	5	5
Éviter les obstacles			5	3	4			5	
Analyser le bon cube selon le signal d'antenne	5								
Utiliser la communication sans fil									
Concevoir un système de préhension pour le crayon							5		
Afficher position réelle				5					
Afficher la trajectoire optimale	3	5							
Afficher le cube résolu la base		5			3				
Allumer une DEL lorsque tâche terminée			5						
Afficher message de fin		5		5					
Afficher message de départ									
Afficher trajectoire réelle avec un délai maximum de 15s									
Afficher les informations sur le robot									
Respecter un budget de 250$	3		1	5			2		

Tableau 8.3 – Description des propriétés fonctionnelles : section "Alimentation et affichage"

	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	Allumer la DEL lorsque tâche complétée	Afficher la trajectoire optimale	Afficher la position réelle	Afficher message de fin	Afficher sur le LCD
Exigences du client		Puissance (W)		Ondulation de tension (V)					Temps d'actualisation (s)		
Être autonome pendant un minimum de 10 minutes	5	3	3	3							
Se déplacer selon la trajectoire optimale	5	5	5	3							
Effectuer une séquence complète en moins de 10 minutes											
Alimenter le Mac mini avec une tension de 22V à 30V **et** une ondulation de tension inférieure à 200 mV			5	5							
Résoudre sudocube	5		5	5							
Dessiner le chiffre selon le signal d'antenne dans une zone prédéfinie (jaune) avec une précision de ± 1cm	3	3	3	3							
Éviter les obstacles	3	3	3	3							
Analyser le bon cube selon le signal d'antenne	5		1	3	3	3	3				
Utiliser la communication sans fil											
Concevoir un système de préhension pour le crayon											
Afficher position réelle			1	3			3				5
Afficher la trajectoire optimale	3		1	3	3	3	3			5	
Afficher le cube résolu sur la base		5	1		3	3	3	5			
Allumer une DEL lorsque tâche terminée			1				2		5		
Afficher message de fin											
Afficher message de départ				3	5						
Afficher trajectoire réelle avec un délai maximum de 15s				3					5		
Afficher les informations sur le robot				3							5
Respecter un budget de 250$	5	2	3	5							

Chapitre 9

Développement logiciel

Chapitre 10

Test unitaires

Chapitre 11

Avancement de la conception et de la construction du système

Annexe A

Annexes

A.1 Code test pour affichage sur LCD

A.1.1 ecran.h

```
#ifndef ECRAN_H_
#define ECRAN_H_

#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/debug.h"
#include "driverlib/gpio.h"
#include "driverlib/interrupt.h"
#include "driverlib/rom.h"
#include "driverlib/timer.h"
#include "inc/lm3s9b92.h"

//definition de trucs qu vont etre pratiques

//#define STCTRL (*((volatile unsigned long *)0xE000E010))
//#define STRELOAD (*((volatile unisgned long *)0xE000E014))
//#define attend(t) SysCtlDelay((SysCtlClock()/3)/(1000/t))


#define ECRAN_DATA GPIO_PORTE_DATA_R
#define ECRAN_CTRL GPIO_PORTA_DATA_R
//volatile unsigned long ulLoop;

//bit de data du LCD
#define ECRAN_D0 0x01 //PE0
#define ECRAN_D1 0x02 //PE1
#define ECRAN_D2 0x04 //PE2
#define ECRAN_D3 0x08 //PE3
#define ECRAN_D4 0x10 //PE4
#define ECRAN_D5 0x20 //PE5
#define ECRAN_D6 0x40 //PE6
#define ECRAN_D7 0x80 //PE7

//bit de control du LCD
#define ECRAN_RS 0x01 //PA0  reset
#define ECRAN_RW 0x02 //PA1  read/write
#define ECRAN_EN 0x04 //PA2  enable
#define ECRAN_BF 0x08 //PA3 "busy flag", indique que l'ecran est occupe

//volatile unsigned long ulLoop;
void ecranClear(void);
void ecranInit(void);
void ecranWriteChar(char caractere);
void ecranWriteLine(char * line, unsigned short size);
```

```
46 void ecranSetPosCursor(short pos);
47 void ecranAttend(void);
48 void ecranControl(unsigned long input);
49
50 #endif /*ECRAN_H_*/
```

A.1.2 ecran.c

```
1  #include "ecran.h"
2
3
4  /*
5  * Cette fonction permet de s'assurer que l'ecran n'est pas occupe avant d'ecrire
6   */
7  void ecranAttend(void)
8  {
9      ECRAN_CTRL &= ~(ECRAN_RS); // RS = 0
10     ECRAN_CTRL |= ECRAN_RW; // RW = 1
11     volatile unsigned long busyflag = ECRAN_CTRL & ECRAN_BF;
12     while (busyflag != 0) //on regarde le busyflag pour s'assurer que l'ecran est pas ↩
           occupe
13     {
14             busyflag = GPIO_PORTD_DATA_R & ECRAN_BF;
15     }
16      return;
17 }
18
19 /*
20  * Cette fonction execute la routine d'initialisation de l'ecran dans le mode qu'on veut
21  */
22 void ecranInit(void)
23 {
24     ecranControl(ECRAN_D5 | ECRAN_D4 | ECRAN_D3 | ECRAN_D2);
25     ecranControl(ECRAN_D3 | ECRAN_D2 | ECRAN_D1);
26 }
27
28 /*
29  * Cette fonction permet de faire des commandes tel que l'initalisation, changement de ↩
       positions et
30  * trucs du genre, tout ce qui est pas l'ecriture de caracteres sur l'ecran
31  */
32 void ecranControl(unsigned long input)
33 {
34 ecranAttend();
35  volatile unsigned long ulLoop;
36 ECRAN_DATA = input; //on met notre entree sur le port de donnees
37 ECRAN_CTRL &=~(ECRAN_RS | ECRAN_RW); //reset et read/write a zero pour pas qu'il re-ecrive↩
       par erreur
38 ECRAN_CTRL ^= ECRAN_EN;
39 ulLoop = SYSCTL_RCGC2_R;
40         for (ulLoop = 0; ulLoop < 200; ulLoop++)
41         {
42              // on met un delai pour que le LCD puisse voir le enable
43         }
44 ECRAN_CTRL ^=ECRAN_EN;
45 ulLoop = SYSCTL_RCGC2_R;
46         for (ulLoop = 0; ulLoop < 200; ulLoop++)
47         {
48         // on attend un peu pour que l'instruction s'execute avant de changer la pin 7
49         }
```

```
50
51 ECRAN_DATA &= ~(ECRAN_D7); //on reset la pin 7 parce elle output aussi le busy flag et ont↩
       pas veux etre pogner dans ecranAttend()
52 }
53
54
55
56 /*
57  * Cette fonction efface le contenu de l'ecran
58  */
59 void ecranClear(void)
60 {
61     ecranControl(ECRAN_D0);
62     volatile unsigned long ulLoop;
63
64     for (ulLoop = 0; ulLoop < 2000; ulLoop++)
65     {
66         //loop pour etre sur que ton est bien clearer
67     }
68 }
69
70 /*
71  * Cette fonction change la position du curseur d'ecriture de l'ecran
72  */
73 void ecranSetPosCursor(short pos)
74 {
75     ecranControl(GPIO_PIN_7 | pos);
76 }
77
78 /*
79  * Cette fonction permet d'ecrire un caractere sur l'ecran
80  */
81 void ecranWriteChar(char caractere)
82 {
83     volatile unsigned long ulLoop;
84     ecranAttend();
85     ECRAN_DATA = caractere;
86     ECRAN_CTRL &= ~(ECRAN_RW); //RW = 0
87     ECRAN_CTRL |=  ECRAN_RS; // rs = 1
88     ECRAN_CTRL ^=  ECRAN_EN; //En = 1
89     ulLoop = SYSCTL_RCGC2_R;
90     for (ulLoop = 0; ulLoop < 200; ulLoop++)
91     {
92         //petit delai, juste pour etre sur
93     }
94     ECRAN_CTRL ^= ECRAN_EN; //En = 0
95 }
96
97
98 /*
99  * Cette fonction permet d'ecrire une chaine de caracteres sur l'ecran
100  */
101 void ecranWriteLine(char * line, unsigned short size) //todo
102 {
103     unsigned short i=0;
104     for(i=0; i <size;i++)
105     {
106          ecranWriteChar(line[i]);
107     }
108 }
```