UNIVERSITÉ LAVAL

Design III

Remise 2

présenté à

Philippe Giguère, Dominique Grenier, Denis Laurendeau

par

Équipe 7 — Zière

matricule	*nom*	*signature*
111 114 478	Garvin, Sébastien	
111 040 128	Kedzierski, Xavier	
111 066 466	Magnan, Charles-Olivier	
111 071 384	Provencher, Jean-Michel	
111 073 630	Bourque, Emile	
111 075 478	Sylvain, Matthieu	
111 074 361	Brown, Jérémy	
907 196 009	Garneau, Laurent	

Université Laval

28 février 2016

Historique des versions		
version	*date*	*description*
1.0	24 janvier 2016	Création du document
2.0	31 janvier 2016	Remise 1
3.0	28 février 2016	Remise 2

Table des matières

1 Diagrammes **1**
- 1.1 Diagramme des fonctionnalités . . . 1
- 1.2 Diagramme physique . . . 2
- 1.3 Diagramme de classes . . . 2
- 1.4 Diagramme de séquences . . . 2

2 Plan de test **3**

3 Plan d'intégration **4**

4 Registre de risques **5**

5 Avancement de la construction du système **6**
- 5.1 Structure mécanique . . . 6
- 5.2 Communications sans fil . . . 6
- 5.3 Alimentation . . . 6
- 5.4 Contrôle et asservissement des moteurs . . . 6
- 5.5 Préhenseur et électroaimant . . . 8
- 5.6 Code Manchester . . . 8
- 5.7 Induction pour recharge . . . 10
- 5.8 Interface de la station de base . . . 11
- 5.9 Planification de la trajectoire . . . 11
- 5.10 Contrôle de la caméra . . . 11
- 5.11 Système de vision . . . 12
- 5.12 Localisation des îles, des trésors et du robot . . . 13

Bibliographie **14**

Chapitre 1

Diagrammes

1.1 Diagramme des fonctionnalités

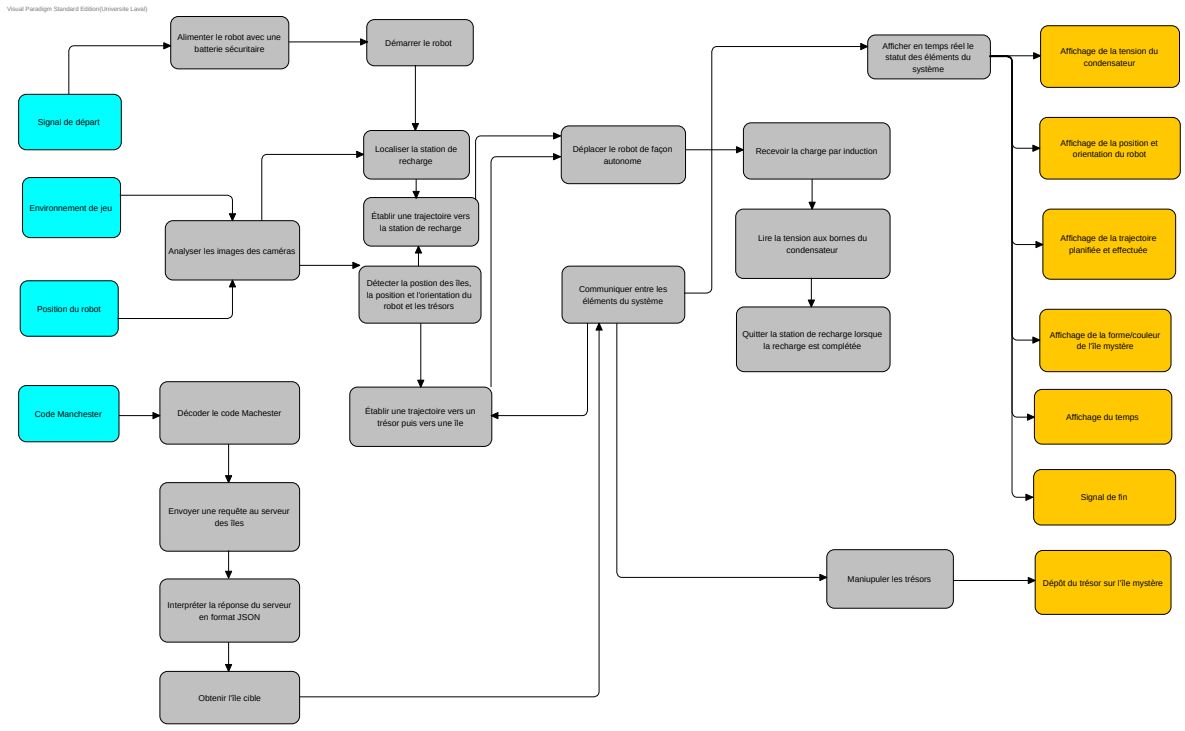

FIGURE 1.1 – Diagramme des fonctionnalités

1.2 Diagramme physique

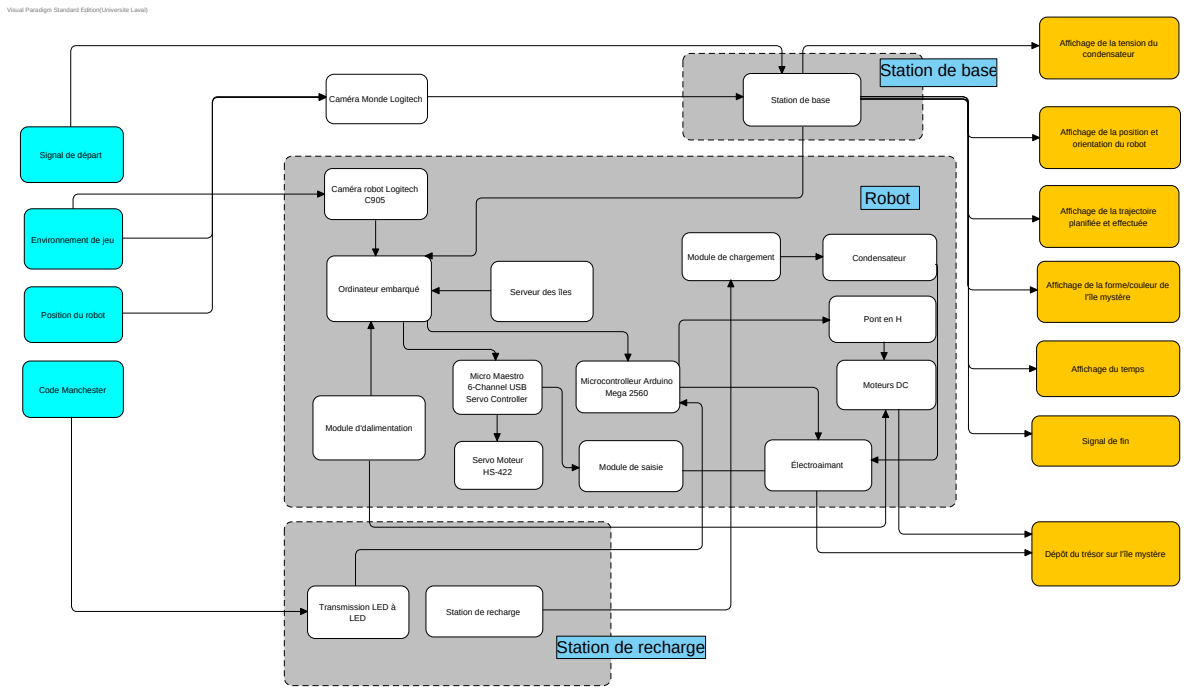

FIGURE 1.2 – Diagramme physique

1.3 Diagramme de classes

1.4 Diagramme de séquences

Chapitre 2

Plan de test

Chapitre 3

Plan d'intégration

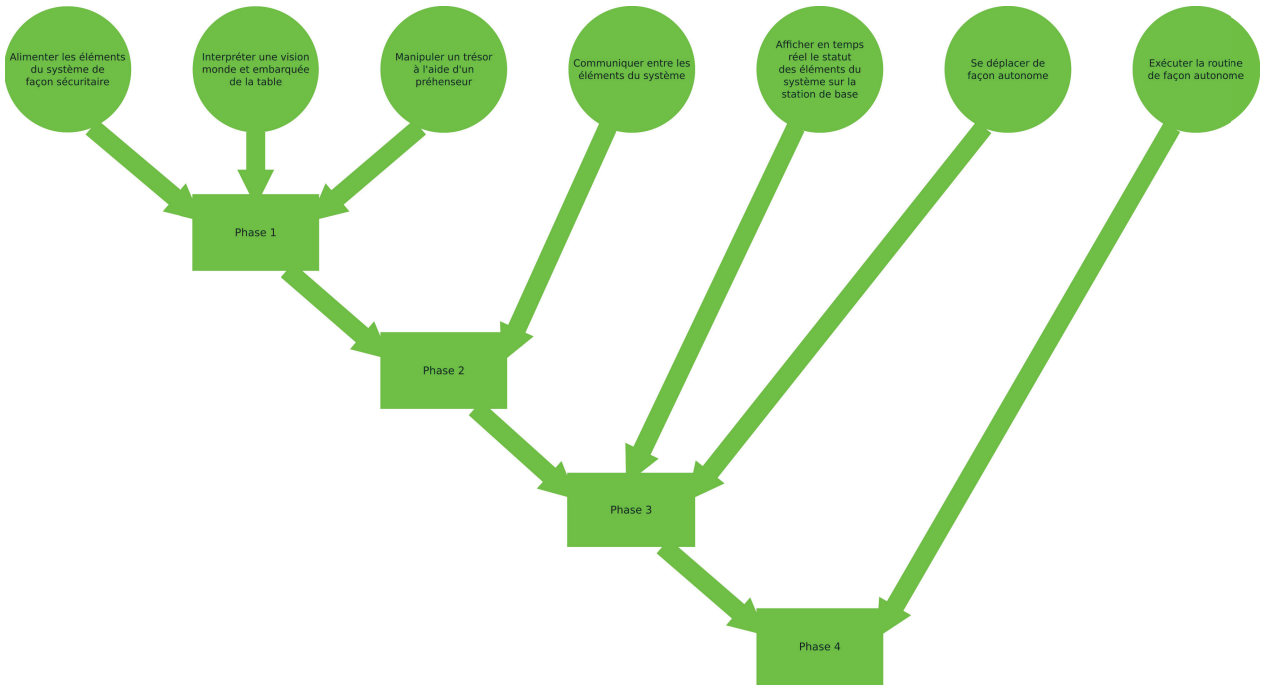

FIGURE 3.1 – Plan d'intégration

Chapitre 4

Registre de risques

Chapitre 5

Avancement de la construction du système

5.1 Structure mécanique

5.2 Communications sans fil

Afin de pouvoir communiquer entre le robot et la station de base, une communication sans-fil est nécessaire. Comme il est important d'établir une connexion avec détection perte de paquet, le protocole TCP a été choisi. Grâce à ce protocole, on s'assure que les paquets perdus seront retransmis afin de ne perdre aucune information dans la communication sans-fil entre le robot et la station de base. Ainsi, en agissant comme serveur, la station de base attendra une connexion client de la part du robot. Une fois cette connexion établie, un échange de fichier JSON contenant des commandes et des paramètres pour le robot s'effectuera. Les avantages d'utiliser un format de fichier JSON pour l'échange est qu'il est simple à implémenter dans plusieurs langages et qu'il est plus facile à parser que le XML.

5.3 Alimentation

5.4 Contrôle et asservissement des moteurs

Afin de contrôller les roues, la première étape était d'alimenter correctement chacun des moteurs des roues. Le pont en H connecté aux moteurs nécessite trois entrées par roue : Deux entrées servant à sélectionner la direction des roues, et une troisième étant un PWM déterminant la vitesse de rotation de la roue.

Le pont en H doit être alimenté avec 12V, ce qui pour les fins de la démonstration est fait avec une source murale. Les roues en tant que telles sont par contre alimentées à une tension de 5V, Selon les spécifications du pont en H, la pin 1 doit reçevoir une tension faible et la pin 2 une tension élevée pour tourner dans le sens des aiguilles d'une montre, le sens

inverse étant obtenu avec une tension élevée sur 1 et faible sur 2. Un des premiers tests est alors effectué, avec le pont en H alimenté et les quatres roues à pleine vitesse. À l'aide de ce test, il fut facile de voir que le branchement des roues de même sens devrait être inversé, car le sens des aiguilles signifie une rotation différente d'un côté à l'autre du robot.

Par la suite, en utilisant des sorties PWM de l'Arduino Mega, des ondes carrées à rapport cyclique de 50% sont envoyées en sortie. Celles-ci, lorsque connectées aux entrées de vitesse correspondant aux quatres moteurs dans le pont en H, permettent de comparer la vitesse des moteurs et de vérifier si celle-ci est consistente. Résultat de ce test : en activant deux roues à la fois, le robot ne va pas en ligne droite. Le moteur 1 fait tourner la roue significativement plus rapidement que le moteur 3 avec la même commande. Un asservissement avec intégrateur est donc absolument nécessaire afin que la vitesse de chaque roue rejoigne la consigne, permettant au robot de rouler en ligne droite.

La librarie PID disponible sur le Playground Arduino permet de rapidement implémenter un asservissement avec une durée d'échantillonage, une commande à rejoindre et des paramètres kp, ki et kd étant défini par l'utilisateur. Pour implémenter la commande à rejoindre, les encodeurs des moteurs sont branché directement au microcontrôlleur Arduino. En effet, ceux-ci génèrent une onde carrée à largeur variable, la longueur de celle-ci pouvant être lue en utilisant la fonction pulseIn(). La durée de lecture maximale (timeout) de la largeur d'impulsion est à la base très lente, environ une seconde par lecture, ce qui rend l'asservissement lent si le robot tourne lentement. Pour remédier à cette situation, un timeout de 3 ms a été choisi.

Avec pulseIn, lorsqu'aucune onde n'est détectée avant la fin du temps maximal, une valeur de 0 est retournée. Hors, l'asservissement fonctionne avec un principe que lorsque la durée retournée par pulseIn est supérieure à la commande, il faut augmenter la sortie vers les moteurs. Afin d'assurer qu'une roue à l'arrêt ne soit pas considérée comme impossiblement rapide, une valeur de 0 est remplacée par la durée maximale de lecture, soit 3 ms.

Les tests initiaux de l'asservissement étaient peu fructueux. En effet, la sortie du PID variait beaucoup trop rapidement entre minimum et maximum, ne permettant que très peu de précision. Ceci était dû à un kp trop élevé, ce qui multipliait l'erreur bien au delà de la valeur maximale du paramètre du PWM, soit 255. En diminuant l'élément proportionnel kp à 0.0005 au lieu de 1, la variation cyclique due à un trop large dépassement est évitée. Le ki est présentement à 0.25 car il permet de rejoindre la consigne suffisemment rapidement sans créer de trop larges dépassements.

Dans l'état actuel de ce sous-système, il y a dépassement en vitesse de plus de 10% pendant environ une seconde. Afin de permettre des déplaements plus précis, du travail sera fait pour diminuer ce dépassement. Pour communiquer à l'Arduino Mega la ccomnsigne à rejoindre, une interruption sur le port série est implémentée en utilisant serialEvent. Lorsque le port série détecte qu'il y a quelque chose à lire sur le port série, le code de lecture de la commande prend alors priorité sur l'exécution du PID.

5.5 Préhenseur et électroaimant

Le préhenseur est constitué d'un électroaimant situé sur un bras mécanique. Cette électroaimant est alimenté par le condensateur de 5F. Lorsqu'un courant circule dans l'électroaimant, il produit une force d'attraction des trésors. Afin de conserver l'énergie dans le condensateur le plus longtemps possible, le courant de décharge dans l'électroaimant doit être limité. En effet, sans cela, le condensateur se déchargera extrêmement rapidement dans l'aimant, car sa résistance interne est de 12 ohms.

Afin de limiter ce courant, nous utilisons un MOSFET dans lequel nous envoyons une onde carré de largeur d'impulsion variable dans sa base et permet de contrôler ce courant. Un fort courant doit d'abord être envoyé dans l'aimant afin d'attirer le trésor. Ensuite, le circuit magnétique est fermé et un courant plus faible peut donc être envoyé. Cette façon de procéder est donc utilisée afin de conserver l'énergie du condensateur plus longtemps.

Également, le système comprend un interrupteur électronique afin de permettre de couper la décharge de courant dans l'électroaimant lorsque le déplacement vers l'île cible est terminé. En ouvrant l'interrupteur, il n'y a plus de courant dans l'électroaimant et le force qui permet de tenir le trésor disparait. Le trésor tombe ainsi sur l'île.

5.6 Code Manchester

Le système de transmission retenu par l'équipe pour communiquer le signal Manchester est un système utilisant une LED comme émetteur et une autre LED comme récepteur. Ce système LED à LED est une solution simple et peu coûteuse qui ne présente donc pas un coût monétaire important pour le projet ni beaucoup de temps de travail. Pour décoder le code il est nécessaire de transmettre l'horloge ainsi que le code, le système est donc installé en double à cet effet. Chacun des signaux est transmis par un banque de quatre LED qui seront situées sur la station de recharge. L'utilisation de plusieurs diodes est justifiée pour rendre le système plus efficace en cas de défaillance d'une de celle-ci et donc d'ajouter de la redondance. Les LED en réception sont situées de chaque côté de la face avant du robot de manière à éviter les interférences. Encore une fois, la redondance est utilisée pour la réception et deux LED sont donc placées à cet effet.

L'horloge fournit avec le dispositif du Manchester n'est pas utilisée car elle est jugée trop rapide, une horloge d'environ 1 à 2H est généré par un microcontrôleur PIC. Une fréquence plus élevée n'est pas requise dû au temps passé à recharger le condensateur, de plus avec une basse fréquence le risque d'erreur est jugé moins élevé. Des LED rouges sont utilisées pour la transmission des signaux, après des expériences avec d'autres couleurs il fut remarqué que celles-ci offraient les meilleures performances.

Un circuit d'amplification des signaux situé sur le robot permet d'amplifier ceux-ci après réception et donc de produire des signaux 0 à 5V. Le schéma de ce circuit est présenté à la figure 5.1 et réalisé sur PCB conformément aux exigences. Les signaux amplifiés sont ensuite lu par des entrées numériques de l'Arduino Mega. La signal d'horloge est lu par une entrée

permettant d'utiliser les interruptions tandis que le code Manchester est lu par une entrée ordinaire. Une interruption est lancée sur chaque front montant d'horloge, les deux données sont alors lues et on effectue une opération XOR sur les signaux afin d'obtenir le signal décodé. Ce résultat est ensuite stocké dans un buffer circulaire de 32 éléments, ce qui assure d'avoir au moins une itération de la séquence de 16 bits dans son ensemble. Un algorithme parcours ensuite ce tableau de données pour recherché la séquence de 8 bits à 1 suivit d'un bit à 0 pour finalement prendre les 7 bits d'intérêt composant la lettre ASCII voulue. Cette lettre est finalement transmise à l'ordinateur embarquée pour utilisation future.

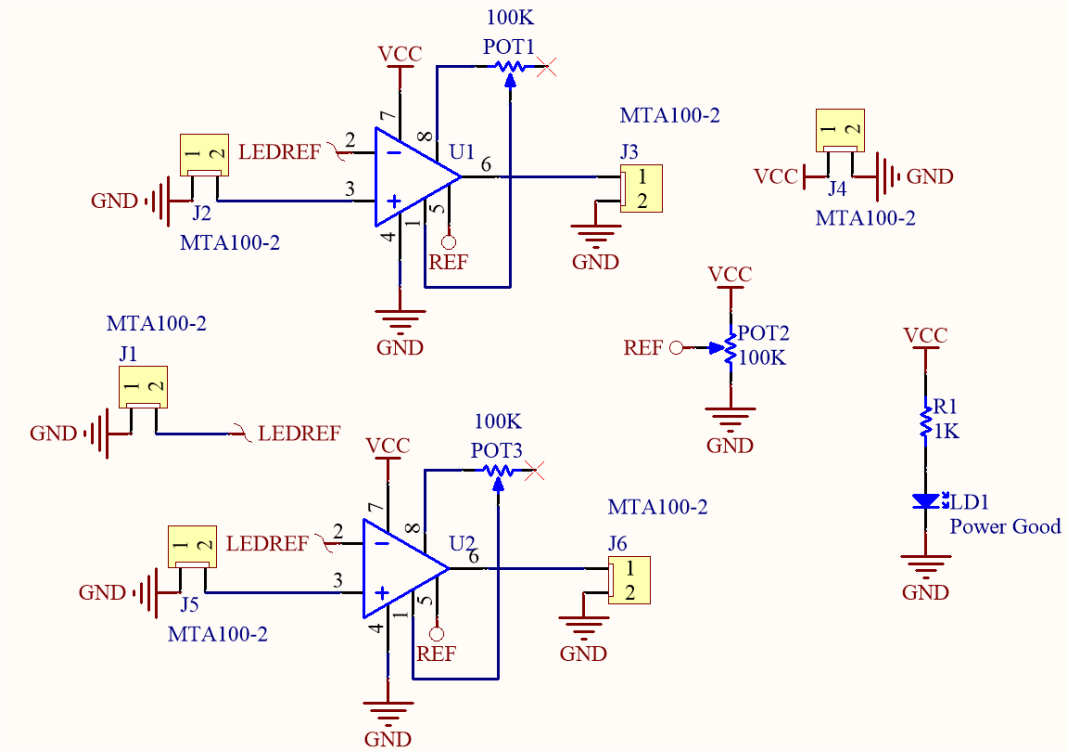

FIGURE 5.1 – Circuit d'amplification des signaux du code Manchester

5.7 Induction pour recharge

L'électroaimant est alimenté par un condensateur. Ce dernier est rechargé par un transformateur. Le primaire du transformateur est situé sur la station de recharge sur le bord de la table. Il est alimenté par la tension DC de 5V/3A qui provient de la carte Manchester. Le transformateur ne fonctionnant qu'en AC, il est nécessaire de hacher cette tension DC.

Plusieurs méthodes existent pour hacher une tension DC. Nous avons choisi d'utiliser un simple MOSFET de puissance dans lequel nous commandons la base avec un horloge. Ce choix a été fait pour sa simplicité. Il est en effet beaucoup plus simple à implémenter qu'un pont en H. Également, avec un seul MOSFET, nous limitions les pertes par dissipation dans les MOSFETs. L'horloge provient d'un microprocesseur situé également sur la station de recharge. Différentes fréquences ont été testées pour ce hacheur. Les fréquences inaudibles (plus de 20kHz) sont plus intéressantes, car notre oreille ne peut pas entendre ces fréquences et le transformateur n'est donc pas bruyant. Toutefois, à ces fréquences, l'effet inductif des bobines du transformateur devient très grand et réduit le courant qui traverse le transformateur.

Comme un plus grand courant dans le transformateur équivaut à une recharge plus rapide du condensateur, une fréquence de 4kHz a été choisie. C'est à cette fréquence que l'effet inductif des bobines du transformateur devient assez faible pour laisser passer un courant dans le transformateur assez grand pour avoir une recharge rapide du condensateur. De plus, si on descend à une fréquence plus basse que 4kHz, l'effet inductif du transformateur devient très faible ; un courant trop élevée y circule et les deux bobines s'attirent fortement par aimantation. C'est pourquoi la fréquence de 4kHz est choisie.

Toutefois, un condensateur ne peut être rechargé par une tension alternative. C'est pourquoi au secondaire se trouve un redresseur. Des tests ont été faits afin de choisir entre un redresseur simple alternance ou double alternance. Le redresseur double alternance a été choisi, parce qu'il est plus efficace.

Le condensateur choisi est un condensateur de 5F. Avec notre système d'induction, il est possible de charger ce condensateur jusqu'à 5V. Des tests ont été faits avec ce condensateur et ont permis de déterminer qu'à partir de sa pleine charge, il est capable d'alimenter notre électroaimant avec le courant nécessaire pour tenir un trésor pendant 3 minutes 30 secondes, ce qui est amplement suffisant pour le projet.

Finalement, nous avons un interrupteur électronique pour ouvrir le circuit de recharge lorsque la recharge est complétée ; sans cela, un faible courant de fuite circule dans le pont redresseur et décharge lentement le condensateur.

5.8 Interface de la station de base

5.9 Planification de la trajectoire

La planification de la trajectoire est effectuée avec les positions obtenues de la localisation (voir 5.12). Pour l'instant, elle se déroule comme suit. Premièrement, une grille de cellules est initialisée avec les coordonnées pixels de l'image ou chaque cellules représente une coordonnée (x,y). Lors de cette création, les pixels qui sont situées à un intervale près des centres des îles (en x et en y) sont déclarées comme étant innaccesible. Il y a donc une zone tampon en forme de carré autour de chacune des îles. De plus, la création de la grille est créée avec une incrémentation en x et en y de sorte à conserver seulement les coordonnées pixels situées à plus ou moins un centimètre de distance (sur la table).

La raison d'avoir une zone tampon en forme de carré au lieu de cercle est simplement puisque l'implémentation est grandement simplifiée. Bien que ceci ne soit pas optimal, une trajectoire pourra tout de même être trouvée dû au nombre réduit d'îles sur la table.

Une fois cette grille initialisée, il suffit de fournir une coordonnée de départ et d'arriver à l'algorithme de planification de trajectoire. Celui-ci est une implémentation de l'algorithme A*. La raison pour laquelle cet algorithme fut retenue est principalement pour sa rapidité d'éxécution. En effet, bien qu'il ne choisice pas le trajet le plus court, il le trouve très rapidement. Cela est primordial puisque le trajet sera recalculé de façon fréquente afin de compenser pour les incertitudes et les erreurs d'approximation. L'autre candidat qui pourrait remplacer l'algorithme A* est l'algorithme de Dijkstra. Cela permettrait d'obtenir le trajet le plus court. Par contre, cela dépendra de sa performance. Cette solution sera explorée prochainement.

En ce qui concerne la distance physique de plus ou moins un centimètre des pixels dans la grille de cellules, il est sujet à changement. Présentement, avec environ 3000 cellules dans la grille, l'algorithme A* s'éxécute presque instantanément. Des tests seront effectuées plus en profondeur selon l'algorithme choisi afin d'avoir une distance physique optimale entre les cellules.

5.10 Contrôle de la caméra

Le contrôle de la caméra embarquée s'effectue par des servomoteurs contrôlés par le Micro Maestro 6-Channel USB Servo Controller. Afin de simplifier les commandes nécessaires au contrôle de la position de la caméra, quatre positions prédéterminées ont été programmées : gauche, droite, avant, trésor. Comme ces quatre positions permettent de couvrir entièrement le champ de vision intéressant, il n'était pas nécessaire d'ajouter davantage de possibilités de contrôle de caméra. La position trésor correspond à la position avant dans laquelle la caméra regarde complètement vers le bas. Afin de déterminer ces positions, le logiciel Maestro Control Center a été utilisé pour noter les valeurs nominales de ces positions. Une fois les valeurs de position en main, une librairie Arduino a été développée afin de déplacer la position de la

caméra dans ces différentes valeurs. Pour communiquer entre le Arduino Mega 2560 et le contrôleur de servomoteur.

5.11 Système de vision

Les responsabilités du système de vision sont partagées entre la station de base et le robot. Le système de vision sert à faire une analyse en profondeur de la table de jeu et à prendre des décisions grâce aux informations venant du robot. Le système doit donc être performant et stable afin de ne pas induire le robot en erreur.

La station de base, par l'entremise de la caméra monde, gère la phase de déplacement tandis que le robot gère la phase d'alignement à l'aide de sa caméra embarquée. La phase de déplacement comporte la détection des éléments cartographiques ainsi que la planification de la trajectoire alors que la phase d'alignement permet au robot de se positionner de manière optimale afin de faciliter l'exécution de la tâche demandée.

La caméra monde est utilisée pour la détection des éléments cartographiques au début de la routine ainsi que pour suivre le déplacement du trésor. Grâce à ces informations, la station de base peut générer une trajectoire et transmettre des commandes au robot.

La caméra embarquée ne sera utilisée que dans deux situations. Tout d'abord, elle servira lorsqu'aucun trésor n'est détecté par la caméra world, le robot tentera alors d'identifier la position des trésors grâce à son point de vue plus rapproché. Ensuite, la caméra du robot servira lors de la phase d'alignement, afin de se rapprocher sécuritairement d'une cible comme une île, un trésor ou la station de recharge. Donc, une fois la phase de déplacement terminée et le robot rendu devant la cible, le système de vision du robot prendra le contrôle afin d'amorcer la phase d'alignement.

La détection d'éléments est une partie cruciale du système de vision. Le système doit être en mesure de détecter tous les éléments présents sur la table pour que le robot puisse effectuer sa tâche avec succès.

Au début de la routine du robot, des photos sont prises par la caméra monde et sont analysées pour identifier la position, la forme ainsi que la couleur de toutes les îles présentes sur la table de jeu. La caméra monde nous offre une résolution intéressante de 1600x1200 pixels, ce qui nous permet de faire une analyse de qualité. Grâce à cette résolution, le système détecte facilement les trésors placés contre les parois de la table. Toutefois, si aucun trésor n'est détecté, le robot utilisera sa caméra embarquée pour effectuer un balayage des parois afin de les identifier.

Suite à plusieurs tests, nous avons conclu que la précision de notre système de détection était plus précis en appliquant un léger estompement à l'image avant de faire tout traitement. Ceci a pour but d'uniformiser les différentes variations de contraste et de couleurs qui pourraient biaiser nos résultats.

Pour la localisation des îles, chacune des quatre couleurs est filtrée et placée dans un masque afin d'avoir un traitement individuel. Pour la localisation des îles, nous utilisons une méthode de recherche par formes prédéfinies. Le système possède donc en mémoire les contours des formes qui pourraient représenter les îles. Le système est donc en mesure d'iden-

tifier la forme qui possède le plus haut taux de compatibilité avec les formes en mémoire. Cette méthode est beaucoup plus fiable que celle utilisée au départ. Notre méthode initiale était de détecter le nombre de sommets des formes détectées pour ensuite déduire quelle forme il s'agissait. Cette méthode avait une faiblesse particulièrement pour les formes qui se ressemblent, comme le cercle et le pentagone. Cependant, avec notre nouvelle méthode, les taux de compatibilités sont très précis. Par exemple, une île ronde possède un taux de compatibilité dix fois plus élevé à un cercle qu'à un pentagone.

Pour ce qui est du suivi de la position et de l'orientation du robot, un patron unique sera placé au dessus de celui-ci afin d'avoir une référence facilement détectable. Le robot est le seul élément que le système de vision continue de détecter tout au long de la routine puisqu'il est le seul élément mobile sur la carte de jeu.

5.12 Localisation des îles, des trésors et du robot

Bibliographie

[1] ETH Zurich, *An LED-to-LED Visible Light Communication System with Software-Based Synchronization*, [En ligne], http://people.inf.ethz.ch/schmist/papers/OWC12Slides.pdf, Page consultée le 27 janvier 2016

[2] Robotshop, *Arduino Mega 2560 Microcontroller*, [En ligne], http://www.robotshop.com/ca/en/arduino-mega-2560-microcontroller-rev3.html, Page consultÃ©e le 30 janvier 2016

[3] Adafruit, *Adafruit Motor/Stepper/Servo Shield for Arduino v2 Kit - v2.3*, [En ligne], https://www.adafruit.com/products/1438, Page consultÃ©e le 30 janvier 2016