

**Parcours des Ecoles d’Ingénieurs**

**Projet Arduino- PeiP 2**

***Année scolaire 2018-2019***

**“Odin : Robot télécommandé scanneur d’obstacle”**

*Etudiants :* Sylvain Marsili - Aymeric Rebuffel

*Encadrants :* Mr. Masson - Mr. Abderrahmane

**Remerciements**

Nous voulons tout d’abord remercier Mr. Masson de nous avoir permis de réaliser ce projet, ainsi que pour son aide précieuse qui a permis son aboutissement.

Nous remercions aussi Mr. Forner pour son aide lors de nos passages au Fablab.

**Sommaire :**

Introduction ………………………………………………... p.5

I Présentation : p.6-8

I.1. Cahier des charges/Fonctionnalités finales ... p.6

I.2. Vision globale …………………………………… p.7

I.3. Planning …………………………………………. P.8

II Développement : p.9-12

II.1. Evolution du projet …………………………… p.9-10

II.1.1. Le scanner ……………………………... p.9

II.1.2. Les moteurs ……………………………. p.10

II.1.3. La communication ……………………. p.11

II.2. Les algorithmes ……………………………….. p11-12

II.2.1. Le robot ………………………………… p11

II.2.2. La télécommande …………………….. p12

III Problèmes : p13-15

III.1. Problèmes scanner/affichage ……………… p13

III.1.1. Les servomoteurs …………………….....

III.1.2. Le matériel ………………………………

III.1.3. Les branchements ………………………

III.1.4 L’affichage ………………………………

III.2. Problèmes moteurs …………………………. p14

III.3. Problèmes Bluetooth ……………………….. p15

Conclusion ………………………………………………... p16

Bibliographie ……………………………………………... p17

**Introduction :**

Lors de la phase d’élaboration du projet, la question principale était : que devrait représenter ce projet ?

Ce projet devait représenter quelque chose d’intéressant à travailler, quelque chose qui mêle cours et découverte, mais aussi quelque chose d’utile, d’un peu nouveau. En regardant les projets des années passées, un robot sortait du lot : il avait pour but de cartographier une pièce en se servant d’une tourelle munie d’un scanner.

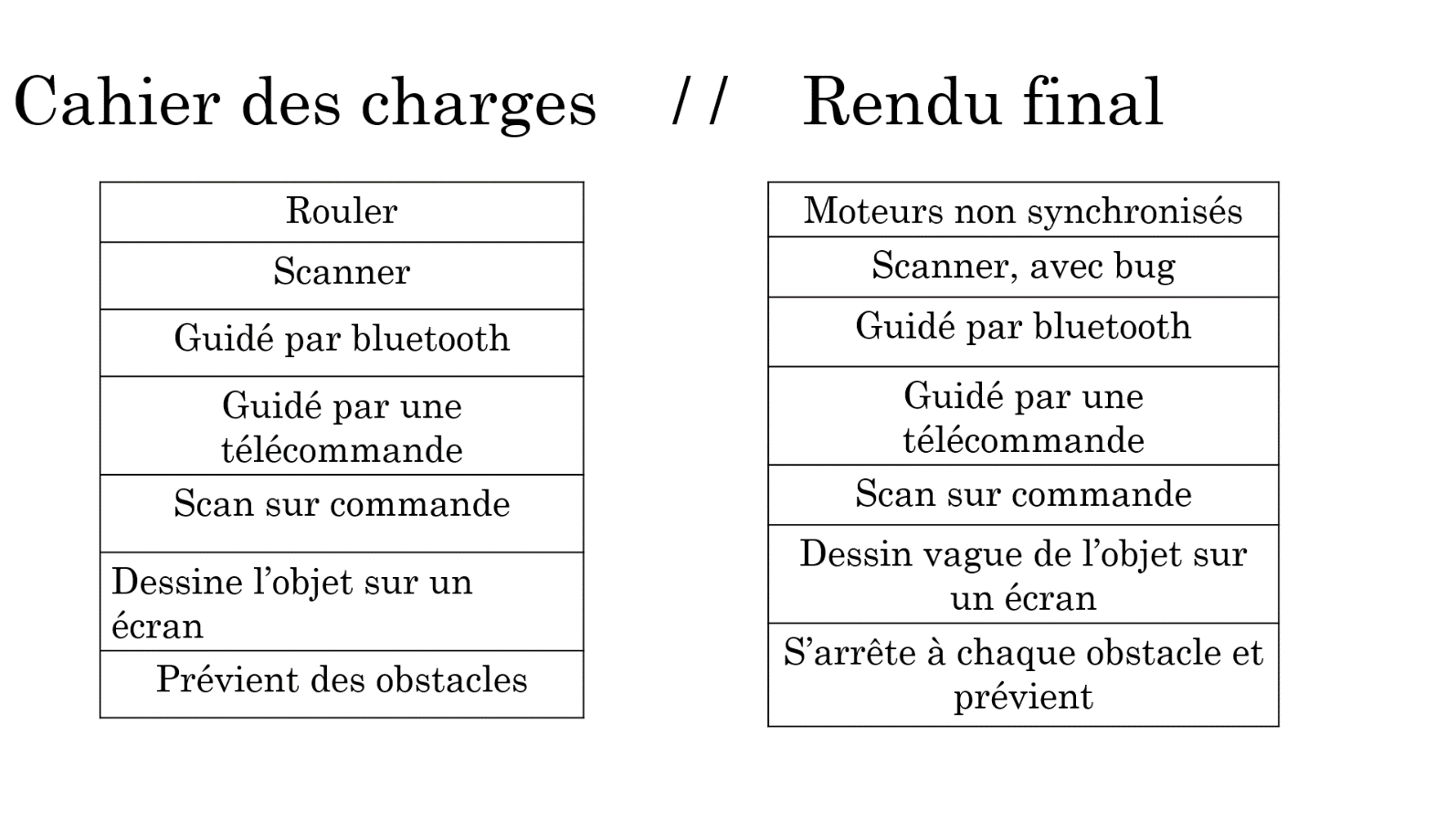
Nous avons voulu reprendre ce projet mais à notre manière. Cette fois ci le robot ne serait pas capable de cartographier une pièce mais il montrerait directement les obstacles sur son chemin. Afficher une image claire d’un objet est compliqué à faire avec du matériel peut couteux, c’est la qu’est venu l’idée de la représentation de l’objet en couleur. La couleur varie en fonction de la distance du point scanné donnant ainsi une impression de profondeur sur l’écran (à la manière des cartes topographiques).

Et c’est ainsi qu’est né Odin, le robot détecteur d’obstacles.

Regardons d’abord l’élaboration de ce projet en partant de nos objectifs de départ pour en arriver au résultat final en passant par le processus d’élaboration ainsi que les problèmes rencontrés. Pour vous faire comprendre Odin nous allons l’étudié partis par partis, au travers de ses différents modules autant qu’au travers de ses algorithmes.

**I Présentation :**

**I.1. Cahier des charges/Fonctionnalités :**



Ci-dessus un tableau comparant le cahier des charges d’Odin au résultat final.

Comme on peut le voir dans ce tableau, les concepts principaux ont été respectés : Odin devait rouler et il roule, Odin devait être guidé via Bluetooth depuis une télécommande et cela est fait, Odin devait pouvoir détecter les obstacles, les scanner et les afficher sur la télécommande si demandé et cela est fait.

Cependant, certain point sont imparfait. Les moteurs ne tournent pas exactement à la même vitesse et le robot est donc difficile à manier. De plus le dessin affiché sur l’écran est censé représenter le relief par différent teinte topographique, ce qu’il fait mais l’image finale est tout de même assez dure à comprendre.

Mais dans l’ensemble, Odin avait été imaginé avec des objectifs précis en tête, objectifs qui sont bien réalisé à terme.

**I.2. Vision globale :**

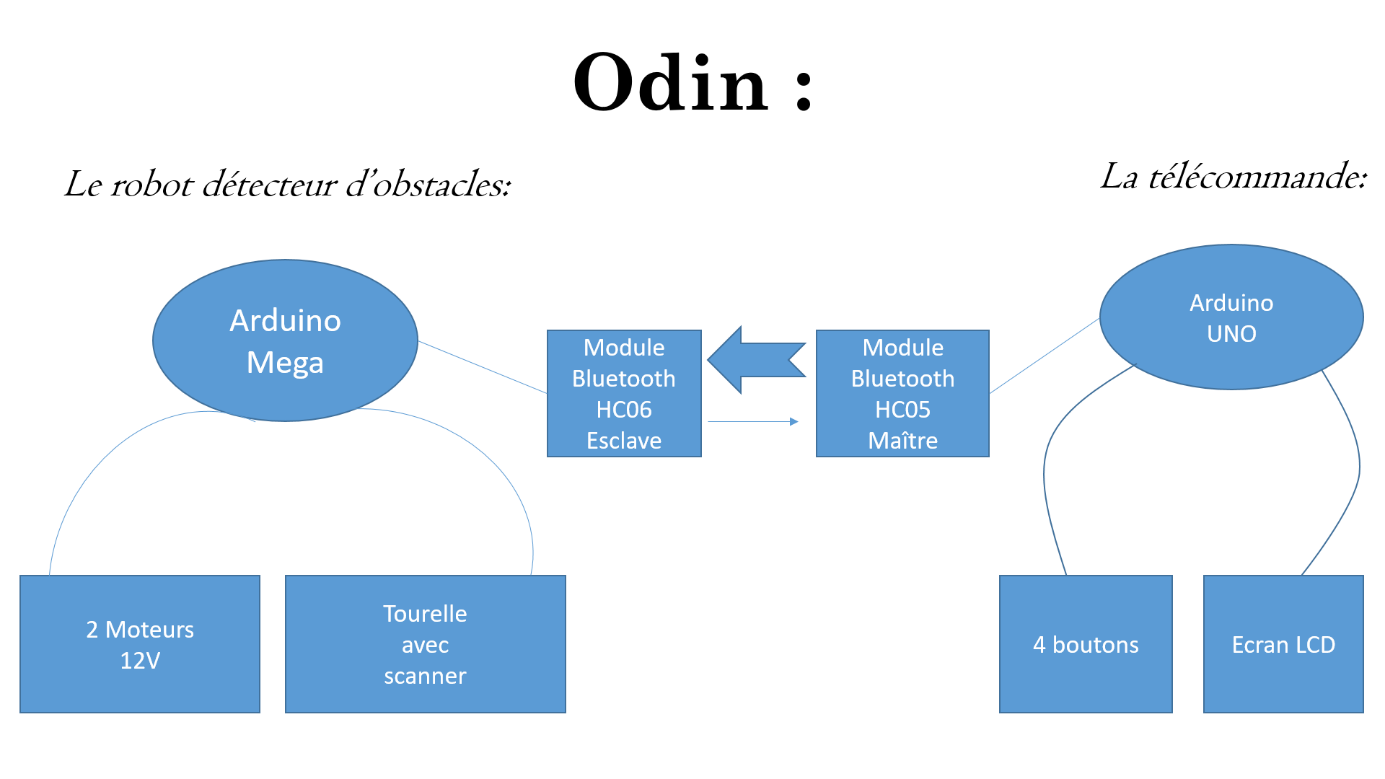
Ci-dessus, un schéma d’Odin et de ses différents modules.

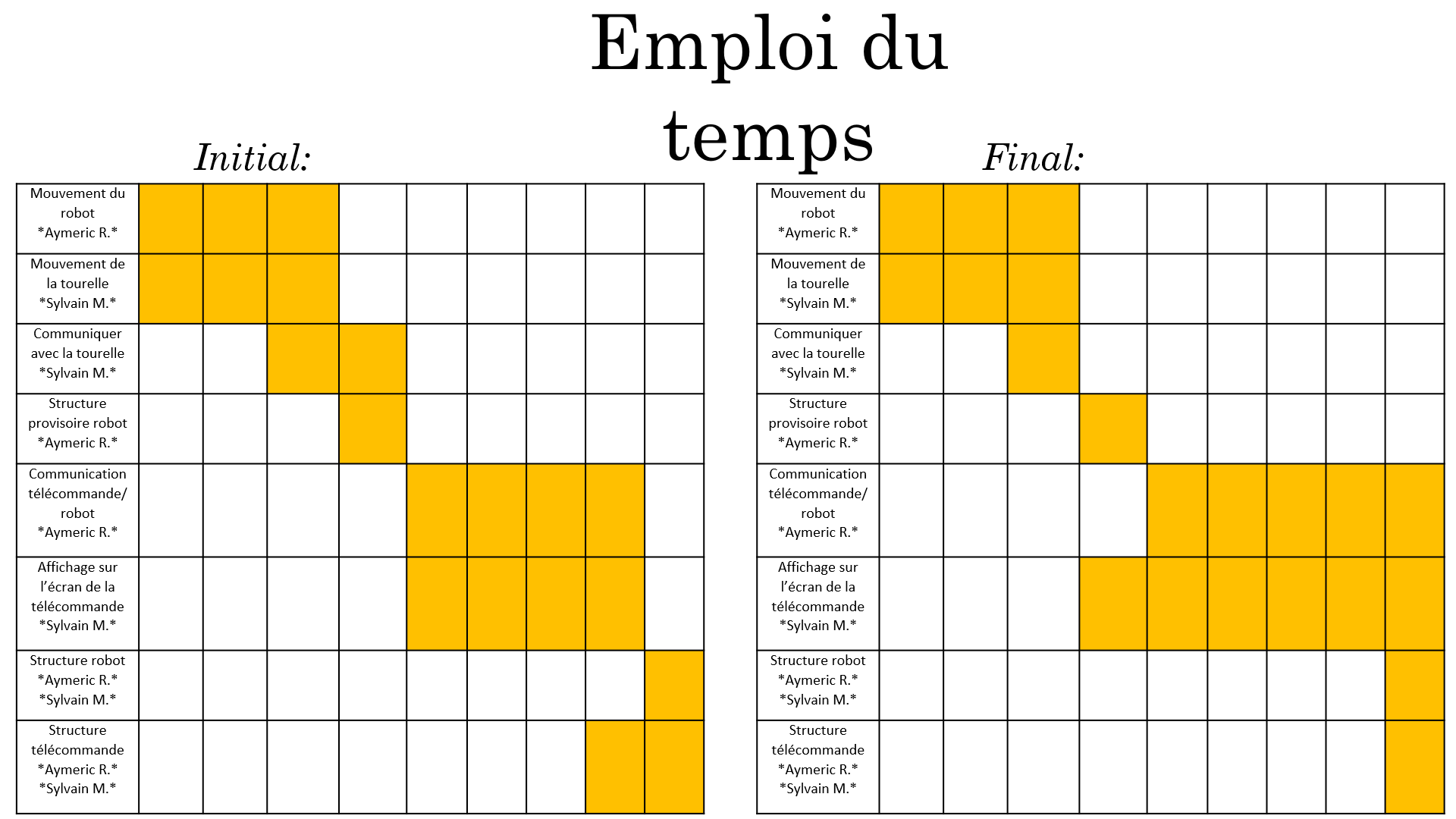
Odin est composé de deux blocs différents : une robot et une télécommande.

Le robot est constitué de deux moteurs 12V, deux servomoteurs, un laser TF mini lidar, et d’un module Bluetooth HC06. Les moteurs permettent le déplacement. Les servomoteurs constituent la tourelle du robot qui peut donc tourner verticalement et horizontalement. Le laser permet le scan et fait aussi parti de la tourelle. Le module HC06 enfin et un esclave Bluetooth et va permettre au robot de recevoir les ordres de la télécommandes.

La télécommandes est constituée de quatre boutons, un écran LCD TFT spi 128\*160px, et d’un module Bluetooth HC05. Trois des boutons sont dédiés aux commandes avant, gauche, et droite. Le dernier permet d’envoyer la commande de scan au robot. L’écran LCD va afficher le dessin de l’obstacle scanné. Et le module Bluetooth HC05 est initialisé en maître pour pouvoir envoyer des commandes au module HC06 du robot.

Quant aux parties non-électroniques : le robot est constitué d’une planche de bois sur laquelle les composants sont fixés à l’aide de scotch double face, de deux roues motrices et d’une roue folle ; et la télécommande n’est constitué que de cinq planche de bois vissées ou scotchées entre elles.



 **I.3. Planning :**

**II. Développement :**

**II.1 Le scanner :**

La première étape du développement concerne le scanner. Nous voulions scanner un objet à l’aide des servomoteurs. Pour cela, nous avons créé la fonction scanner(). Cette dernière permet au laser de balayer horizontalement de 40° à 120°, et verticalement de 50° à 110°( à l’aide de boucle for et de la bibliothèque servo).

Par la suite, l’ajout d’un module Bluetooth HC-06 a permis le lancement de la fonction scanner, à tout moment, via le téléphone portable (grâce à la librairie SoftwareSerial). Le choix du lancement de scanner() était alors possible.

Afin de détecter en permanence les obstacles, un balayage horizontale continue était obligatoire. Ce balayage a été programmé pour des angles allant de 10° à 150°. Cet éventail, plus grand que scanner() et associé à un mouvement plus rapide du balayage a permis de détecter tous les objets à proximités, et éviter les collisions avec ceux-ci .

Pour calculer la distance entre notre robot et l’objet, la programmation du laser était nécessaire. Nous avons donc implémenté une fonction fctlaser(), qui nous retourne la distance (grâce à SoftwareSerial) lors de son appel.

La combinaison du Bluetooth, du laser, et du scan a été compilée sur une carte Arduino Mega. Une première tourelle opérationnelle a alors été créée.

Le réglage de l’écran fut l’étape suivante. Après une brève recherche afin de connaitre son branchement, les premiers codes ont été créés (grâce aux librairies SPI et TFT, et aux fonction stroke() pour la couleur et point() pour l’emplacement sur l’écran). Après de nombreux tests et problèmes, une technique d’affichage a été choisie (décrite dans la partie problèmes).

Nous avons ensuite ajouté une valeur limite de distance (60 cm), afin d’informer l’utilisateur d’un obstacle présent devant lui, et de lui demander son scan ou non.

En cas d’approbation, la méthode scanner() est appelée. La formation d’une image commence, via l’envoie par Bluetooth des distances a la télécommande (possédant l’écran).

Une pause de cinq secondes est alors effectué afin de permettre au robot de se déplacer (grâce a la fonction bouton() ) sans détecter le même objet.

**II.2. Les moteurs :**

Les moteurs furent l’une des premières choses à devoir faire fonctionner dans le projet : cela aurait dut être aussi simple que nécessaire au projet. Les brancher ne pris que peu de temps car nous l’avions étudiés en cours, mais les synchroniser fut si dur que cela couta plusieurs séances pour s’avérer finalement infructueux. Les problèmes liés aux moteurs sont détaillés dans la partie *III.2 Problèmes moteurs*.

Une fois les moteurs relativement opérationnelles, il a fallu les connecter en Bluetooth à la télécommande. Ceci ne fut pas très compliqué, voir plus de détail dans la partie *II.3 La communication.*

Enfin, à la fin du projet, lors de la mise en commun des programmes du binôme, les moteurs se sont retrouvés liés à la tourelle ainsi qu’au Bluetooth pour permettre le bon fonctionnement de tout le robot.

**II.3 La communication :**

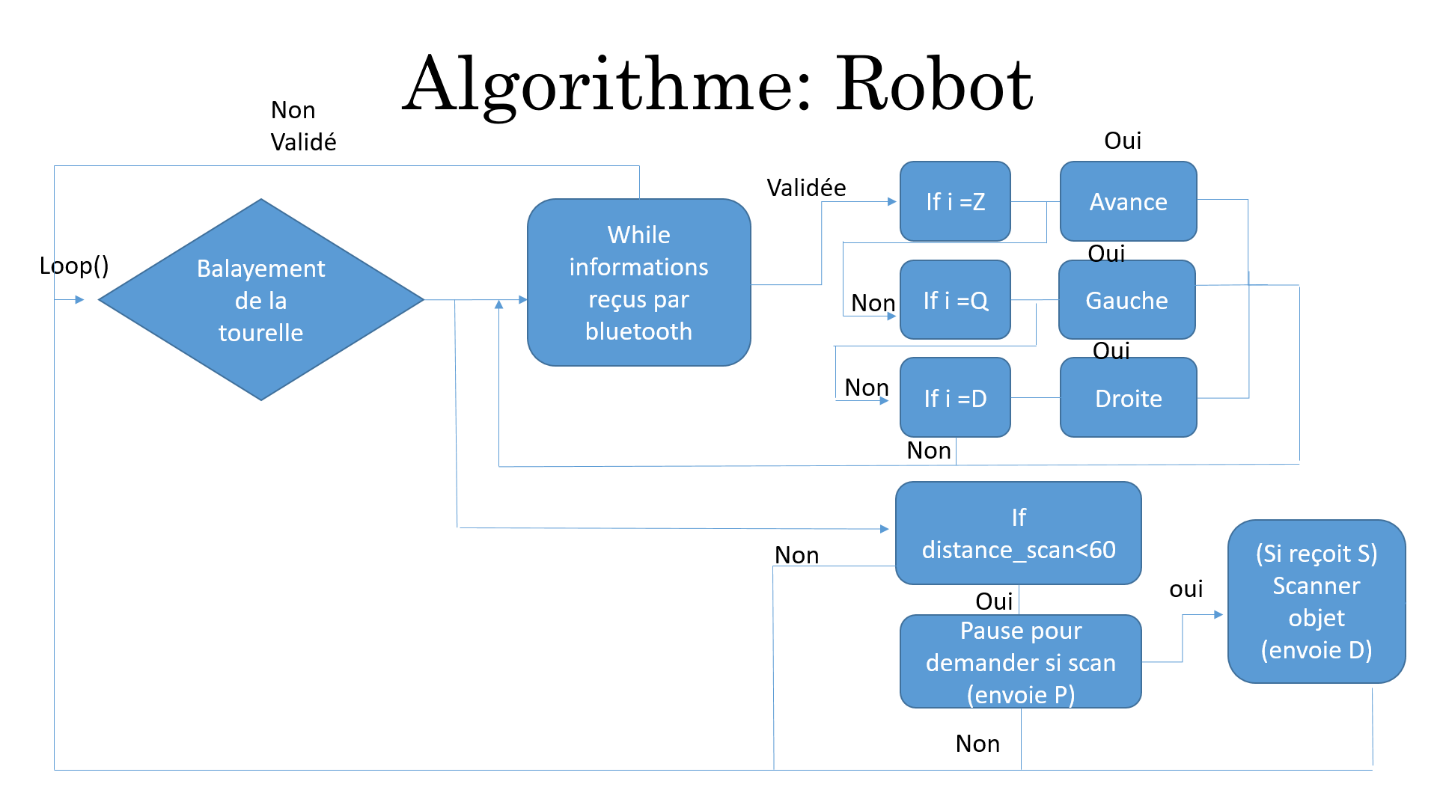
La communication n’a été entamée qu’à la toute fin du premier semestre de cours, avant le rendu de mi-projet. L’objectif était alors de simplement déplacer le robot avec le téléphone, ainsi que déclencher le scan par téléphone pour pouvoir présenter ce qui avait déjà été fait.

Le vrai travail sur la communication a été entamé immédiatement après et la première mesure a été d’initialiser le module HC05 en maître et de connecter le module HC05 de la télécommande avec le module esclave HC06 du robot.

Ensuite, des premières données test sont échangées pour vérifier le bon fonctionnement et la bonne communication des modules. Puis, le module esclave est lié au moteur pour que l’on puisse les commander depuis la télécommande.

Enfin, le module esclave et les moteurs ont été mis en relation avec la tourelle pour que la télécommande puisse contrôler tout le robot.

**II.2. Les algorithmes :**

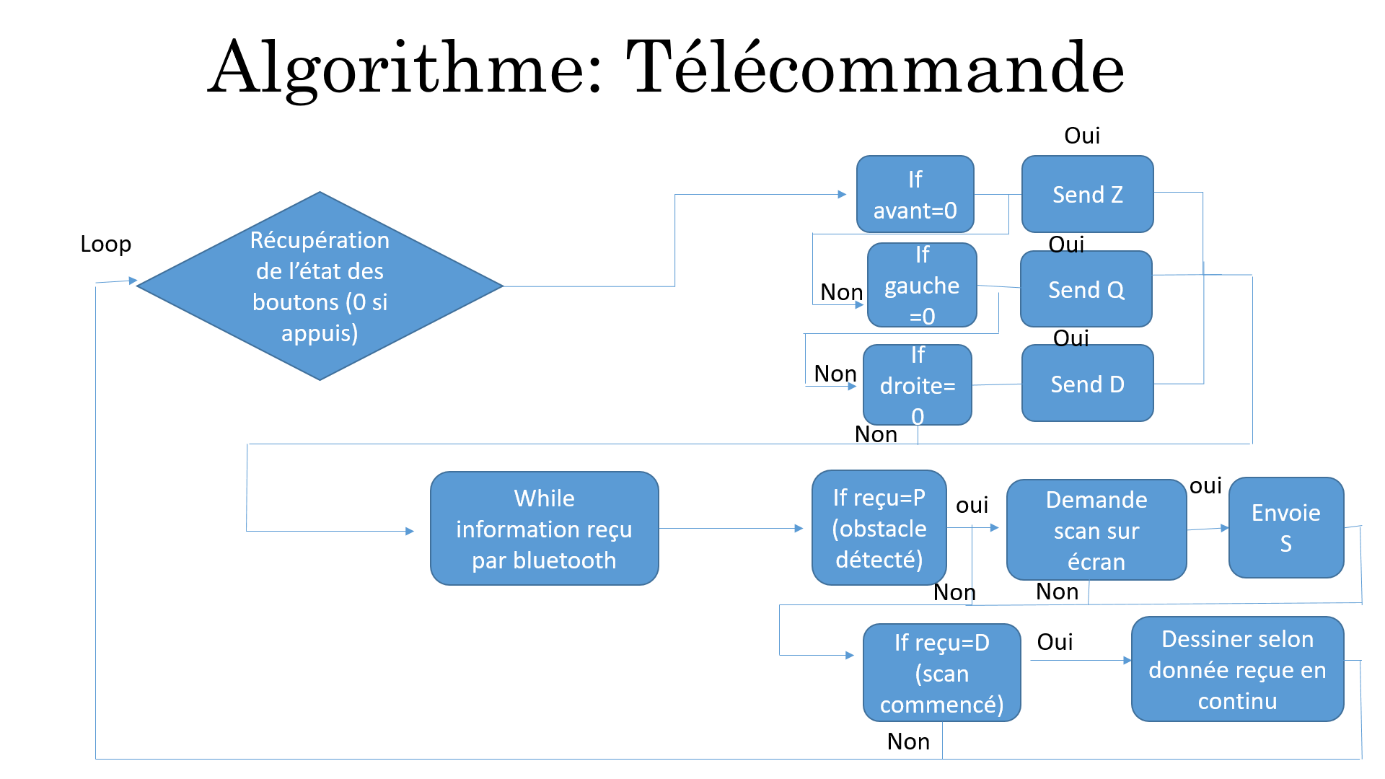
**II.2.1. Le robot :**

Ci-dessus l’algorithme du programme téléversé dans la carte Arduino mega sur le robot. L’algorithme se déroule de 2 manières différentes en parallèle.

Le premier chemin, correspondant au déplacement du robot, est une boucle while qui se lance lorsque le module HC06 reçoit des informations par bluetooth. Il va regarder, dans l’ordre, si la commande est d’avancer, de tourner à gauche ou de tourner à droite. S’il rentre dans l’une des boucles if correspondant à ces commandes, il effectue la commande puis retourne au début de la boucle while. S’il ne rentre dans aucune boucle, il retourne directement au début de la boucle while. Il en sort dès qu’il arrête de recevoir des informations par le module HC06.

Le deuxième chemin correspond à la détection d’obstacle. Il est emprunté si le robot détecte un obstacle à moins de 60cm de lui. Si cela arrive, le robot s’arrête et envoie un message à la télécommande. S’il reçoit la bonne réponse de la part de la télécommande, il scan l’objet et renvoie les données à la télécommande. S’il n’y a pas d’obstacle à moins de 60cm, ou si la télécommande n’a pas envoyé de demande de scan au bout de 5secondes après la pause, le programme revient au début du loop.

Les deux chemins sont empruntés simultanément de manière à ce que même lorsque le robot reçoit des informations de déplacement, il puisse s’arrêter si un objet arrive trop près.

**II.2.2. Le télécommande :**

Ci-dessus, l’algorithme téléversé dans la carte Arduino UNO de la télécommande.

La boucle commence par la récupération de l’état des boutons (si on a appuyé dessus ou non). On vérifie dans l’ordre si on demande à aller vers l’avant, la gauche, ou la droite. Si on rentre dans l’une des boucles, on envoie au robot la direction dans laquelle on veut aller puis on passe à la prochaine boucle while. Si on ne rentre dans aucune des boucles if (l’utilisateur n’a pas appuyé sur un bouton), on va également vers la prochaine boucle while.

Cette boucle permet de regarder si le robot a envoyé des informations à la télécommande. Si oui, et, plus précisément, si le robot dis qu’il a détecté un obstacle, on affiche un message sur la télécommande pour demander à l’utilisateur s’il veut scanner l’obstacle. Si oui (l’utilisateur appuie sur le bouton de scan), on envoie un message au robot pour qu’il commence à scanner. Lorsque celui-ci nous indique qu’il a bel et bien commencé à scanner, la télécommande récupère les données qu’il envoie et dessine progressivement une image du robot sur l’écran de la télécommande.

**III: Les problèmes**

**III.1. Le scanner et l’affichage :**

**III.1.1. Les servomoteurs**

L’installation et la programmation des servomoteurs, pour scanner, ont posés quelques problèmes techniques. Au niveau de leurs mouvements, peu fluide et trop saccadés, et ne parcourant pas tous les angles prévus. La gestion des pauses entre chaque nouvelle position et le manque de puissance fournit par la carte Arduino en était la cause principale. L’ajout de fonctions delay() au programme et de capacités pour fluidifier le courant ont résolu ce premier problème. Malgré cela persistent encore quelques perturbations, mais qui n’altèrent pas le bon fonctionnement du robot.

**III.1.2. Le matériel**

L’ajout d’une fonction Bluetooth afin de lancer le scanner n’était pas concluant aux premiers essais. Cela était dû à un disfonctionnement du module Bluetooth HC-06 fourni.

De nombreux fils cassés, difficiles à détecter, ont provoqués un disfonctionnement fréquent du scanner. Leur remplacement étant impossible, de nombreuses soudures et du renforcement avec du scotch ont été obligatoires pour garder le scanner fonctionnel.

Le laser et le Bluetooth ne peuvent fonctionner en même temps sur une carte Arduino Nano, tous deux utilisant chacun une communication. La carte Arduino Nano ne peut gérer qu’une communication à la fois. Le remplacement avec une carte Arduino Mega 2560 fut nécessaire, cette dernière pouvant gérer jusqu’à trois communications à la fois.

**III.1.3. Les branchements**

Le changement de carte a amené un nouveau branchement. Or la carte Arduino Nano et Arduino Mega ont des emplacements de ports différents et cela a posé un problème pour le branchement de l’écran LCD. En effet deux ports n’étaient pas situés aux mêmes endroits. Une recherche sur internet a permis de trouver leur équivalent sur la carte Mega (port 52 et 51 au lieu de 11 et 12).

**III.1.4 L’affichage**

L’affichage sur l’écran LCD couleur a été un des plus gros problèmes à surmonter. L’utilisation de la fonction point(x,y) de la librairie TFT d’Arduino permet de dessiner un point aux coordonnées x et y de l’écran. Or, gérer ces coordonnées, la distance et l’affichage sur l’écran s’est avéré compliqué et a entraîné plusieurs échecs.

Ayant tout d’abord opté pour un affichage utilisant les coordonnées de l’objet par rapport à notre laser (coordonnées sphériques) le résultat était très approximatif. Une simulation d’affichage sur Excel avec cette technique, fut concluante. Par déduction, le problème venait bien de l’écran qui gère mal la valeur des coordonnées envoyées.

Une autre technique a été choisie. Le mouvement du laser définit les coordonnées au fur et à mesure, et prend seulement en compte la distance par rapport au sol (parallèle à ce dernier), qui définit alors les couleurs de l’affichage sur l’écran.

**III.2.Problèmes moteurs :**

Le but était ici d’avoir un robot qui pouvait se déplacer dans chaque direction, et qu’il soit facilement maniable. Ainsi il fallait non seulement monter et brancher correctement les moteurs, il fallait aussi réussir à les synchroniser. En effet, tous les moteurs, bien que du même type, ne tournent pas à exactement la même vitesse et, même, ne tournent pas exactement à la vitesse demandé.

Pour pallier à ce problème, le but était originellement d’utiliser des encodeurs qui nous auraient permis d’asservir les moteurs et de leur imposer la vitesse désirée. Cependant, les dis-encodeurs se sont révélés extrêmement durs à maîtriser. Tout d’abord il a fallu comprendre le principe de fonctionnement des encodeurs et comment s’en servir pour asservir les moteurs. Là sont apparus les premiers problèmes ; en cherchant des aides où indications sur internet, nous avons trouvé divers site proposant divers solutions. Cela s’avéra être un problème puisque, ne sachant pas comment les encodeurs fonctionnaient, il était dur de savoir quel site proposais la meilleur aide. Beaucoup de temps à était perdu à essayer de réparer des programmes défectueux, à ne serait-ce qu’en comprendre d’autre avant de se rendre compte qu’ils ne seraient d’aucune aide, ou encore simplement à essayer de créer notre propre programme à partir de la compréhension limité des encodeurs.

Nous n’avons au final, après plus d’une dizaine d’heure de recherche, pas réussi à trouver une solution viable à ce problème d’encodeur et nous avons donc abandonné cette piste.

Nous avons donc été contraint d’utilisé une méthode plus simple mais moins fiable pour faire tourner les deux moteurs à la même vitesse : en essayant plusieurs fois le robot, nous avons finis par trouver quel moteur tournait le plus vite et à quel point il tournait plus vite, puis nous avons progressivement abaissé sa vitesse afin qu’elle corresponde à celle du deuxième moteur.

**III.3.Problèmes Bluetooth :**

La communication bluetooth entre le module HC05 (en maître) et le module HC06 (en esclave) a pu être mis en place sans rencontrer de problème majeur. Les principales pertes de temps sur cette partie étaient liées à des fils défectueux ou encore au fait que nous ne pouvions tous simplement pas communiquer avec notre premier module HC06, nous supposons que sa vitesse de communication avait était altérée.

Il fut tout de même fastidieux de trouver les quelques lignes de code spécifiques permettant l’envoie des donnés d’un module à l’autre, mais quelque recherche sur internet et l’aide de notre professeur eurent raison de ce problème.

**Conclusion :**

Odin est donc l’aboutissement de quatre mois de projet, de recherches, et d’efforts. Comme prévu Odin est capable de se déplacer et d’être dirigé par une télécommande (qui n’est pas le téléphone). S’il rencontre un obstacle, il l’indique sur la télécommande et si l’usager décide de scanner l’objet, Odin le scan et en renvoie une image sur l’écran. Cependant l’image n’est pas forcément facile à comprendre et la durée de scan est longue. De plus les moteurs ne sont pas parfaitement accordés et le robot dévie un peu lorsqu’il essaie de se déplacer tout droit.

Avec neuf séances de plus, nous réglerions ces problèmes de maniabilité et d’affichage. De plus, nous rajouterions un mode automatique au robot. Il pourrait ainsi se déplacer tout seul (à la demande de l’utilisateur) en se contentant d’éviter les obstacles. A ça s’ajouterais une fonctionnalité de cartographie qi permettrais au robot de retenir où se trouve les obstacles qu’il a déjà rencontré et d’en afficher la position, par rapport à lui-même, sur la télécommande.

**Bibliographie :**

* <https://randomnerdtutorials.com/guide-to-1-8-tft-display-with-arduino/>
* <https://www.arduino.cc/en/Guide/TFT>
* <https://www.arduino.cc/en/reference/SPI>
* <https://www.gotronic.fr/pj2-35741-arduino-1710.pdf>
* <https://www.pobot.org/Asservissement-d-un-moteur-a.html>
* <http://www.ferdinandpiette.com/blog/2012/04/asservissement-en-vitesse-dun-moteur-avec-arduino/>
* <https://knowledge.parcours-performance.com/arduino-bluetooth-hc-05-hc-06/>
* http://www.matechnologie.com/spip.php?article416