

ARAGOG®

Rapport final

[ARDUINO]

Remerciements

Nous remercions Monsieur Pascal Masson pour toute l'aide qu'il nous a fourni, concernant le module SSC.

Nous remercions Antoine Blaud et João Brillhante pour la partie code.

INTRODUCTION

1. FORMATION DU PROJET
2. CAHIER DES CHARGES
3. PLANNING

DÉROULEMENT DU PROJET

1. RECUPERATION D'UN ANCIEN PROJET (REMISE EN ETAT)
2. ETUDE DU MOUVEMENT HEXAPODE
3. RECONSTRUCTION (CALIBRER LES MOTEURS EN FONCTION DE NOTRE ETUDE)
4. CONNEXION ENTRE L'ARDUINO ET LE MODULE (`SOFTWARESERIAL SSC(RX,TX);`)
5. LE CODE/ LE BLUETOOTH
6. COMPORTEMENT AUTONOME
7. LE NON RESPECT DU PLANNING

CONCLUSION

1. CE QU'ON AURAIT PU FAIRE AVEC 9 SEANCES SUPPLEMENTAIRES
2. RESSENTIS PERSONNELS SUR LE PROJET
3. COMPARATIF ATTENTES / RÉALITÉ

RESSOURCES

BIBLIOGRAPHIE

INTRODUCTION:

1. FORMATION DU PROJET

La première idée qui nous est venue à l'idée était d'analyser des liquides et donner leur composition sur une application, qui donnerait notamment le taux de sucre, ou les colorants nocifs pour la santé. On aurait pu développer une application à terme qui pourrait garder une trace des boissons analysées et ainsi construire plusieurs plans de suivi des individus (exemple : remise en forme, aide à la perte de poids, maintien de la forme physique).

Cette première idée nous a lancé dans des recherches: on s'est inspirée d'un projet déjà existant sur internet qui recense pas à pas comment construire la machine.

Mais entre sa taille imposante et surtout le prix des composants (il faut utiliser des prismes qui coûtent près de 100\$, et si des particules de poussière tombent dessus, il faut les changer...) on a rapidement changé d'idée.

C'est comme ça qu'on s'est retrouvé à se demander sur quoi on voulait vraiment travailler. Puisqu'on voulait travailler sur de la chimie organique, et rester dans l'idée de l'analyse de substances, on a pensé faire un robot qui puisse analyser les composantes de l'air et diffuser du parfum en fonction des odeurs présentes dans une pièce. Ce robot aurait pris la forme d'un scorpion, son dard servant à la diffusion du parfum ou des huiles essentielles. Mais on aurait toujours eu le problème de l'analyse de la composition d'une atmosphère.

C'est comme ça qu'on a finalement abandonné le dard du scorpion et transformé le scorpion en araignée.

Étant deux fans de la saga Harry Potter, on a eu l'idée de s'inspirer des animaux fantastiques de cet univers.

On s'est donc basée sur l'araignée géante Aragog qui peut servir d'animal de compagnie en métal.

Au départ, l'araignée était censée être totalement autonome. C'est à dire qu'elle pourrait se déplacer dans n'importe quel environnement en détectant et évitant les objets alentours.

C'est ainsi que ce projet est né. Nous avons utilisé différentes sources pour créer un caractère à notre robot telle que l'araignée Lucas the Spider, créée par Josh Slice en animation 3D ou le comportement d'une espèce précise: l'araignée paon.

2. CAHIER DES CHARGES :

Besoin et contrainte liées au projet

Besoins fonctionnels : Pour le réaliser nous avons besoin d'une structure métallique, à savoir, deux plaques pour le corps, des petites plaques prédécoupées qui créeront le haut et le bas des pattes, 18 servo-moteurs pour faire fonctionner notre robot (1 pour chaque patte), et une grande quantité de visse. Pratique, tout ce matériel est déjà en la possession de l'école. Si nous voulons faire à terme un système de sonar pour rendre l'araignée autonome, nous aurons besoin d'un capteur à ultra-son.

Contraintes mécaniques : Nous avons étudié le mouvement de marche d'une araignée à six pattes, et nous tentons de le reproduire le plus fidèlement possible: Les pattes sont coordonnées 2 à 2 selon le modèle d'un pont en H. Les pattes avant doivent tracter le corps, les servos doivent soutenir le poids du châssis métallique, et les pattes ne doivent pas glisser sur le sol (introduction de patins anti-dérapants).











Esthétique: Sur le modèle de l'araignée Lucas, nous voulons allier fonctionnalités technologiques et esthétique: la compagnie du robot doit être agréable et amusante pour l'utilisateur. Nous avons pour ambition de recouvrir le châssis une fois le montage terminé.











Contraintes : Réussir à coder des programmes en C qui permettent d'effectuer les mouvements voulus. Démonter et remonter la structure abîmée, ou qui n'est pas fonctionnelle. Réussir à réaliser toutes les fonctionnalités attendues dans les temps .

Résultats attendus : Nous voulons réaliser toutes les fonctionnalités énumérées en première partie en peu de temps pour pouvoir si possible, au gré de nos envies ajouter des fonctionnalités que nous n'avions pas prévu.

Exigences : Nous espérons réussir à respecter le calendrier dans l'espoir de présenter un projet parfaitement fini aux jurys.

3. Planning :

		Projet ARAGOG							
Maya									
Mathilde									
			13h 30	14h 00	14h 30	15h 00	15h 30	16h 00	
		Séance 1	Introduction aux projets		Réparations	Test Servo			
			Introduction aux projets		Réparations	Démontage de l'araignée			
									
		Séance 2	Montage d'une patte d'Aragog			ramme de test du mouver			
		Vacances							
		Séance 3	Compréhension du mouvement d'1 patte isolée			Etalonnage des servos			
			Appréciation de la marche à 2 pattes			montage 2 dernieres patt			
			Programme de marche a 1 patte		Appréciation de la marche à 2 pattes				
		Séance 4	mise en place du contrôle par bluetoot						

		Terminer la marche à 2 pattes	ation de la coordination de	
	Séance 5	Appréciation de la marche à 4 pattes	ation de la coordination de	
		Module Bluetooth	emriner la marche à 4 patte	
	Séance 6	Coordination de la marche à 4 pattes	Module bluetooth	
		Appréciation de la marche à 6 pattes	Bluetooth	
	Séance 7	Bluetooth	Marche à 6 pattes	
		Marche à 6 pattes	Bluetooth	
	Séance 8	Bluetooth	Marche à 6 pattes	
		Terminer les derniers détails du projet	Derniers tests	
	Séance 9	Terminer les derniers détails du projet	Derniers tests	

DÉROULEMENT DU PROJET:

1.RECUPERATION D'UN ANCIEN PROJET (REMISE EN ETAT) :

Au commencement, nous voulions fabriquer la plus petite araignée possible, et de ce fait construire nous-même le châssis, avec des servomoteurs adaptés à sa petite taille. C'est ainsi que notre enseignant nous a présenté un robot araignée d'une promotion antérieure, afin de commencer à travailler sur des pièces concrètes et apprendre comment fonctionne ce type de robot.

Nous avons manipulé ce qui était déjà monté, en se servant de nos cours sur les servomoteurs, et nous nous sommes rapidement rendues comptes que chaque servomoteur doit être "calibré", c'est-à-dire que l'amplitude total de l'angle du moteur doit correspondre à une certaine amplitude de mouvement et donc dépend de la position qu'il occupe.

En manipulant le projet récupéré tel quel, on s'est vite rendu compte que, pour pouvoir manipuler le robot de la façon la plus optimale, nous devons nous l'approprier, donc comprendre comment ils fonctionnait, et comment il était monté.

C'est pour cela qu'on a commencé par démonter le projet entier :



En revanche, ce robot est très imposant, et nous voulions seulement l'étudier pour se familiariser avec un robot marcheur avant de commencer notre petit robot, créer le châssis et le faire marcher. Cependant, dès les premières séances, nous nous sommes rendues comptes de la masse de travail considérable qu'il fallait fournir pour faire marcher un robot d'une telle ampleur. Nous avons passé énormément de temps à mettre ce projet en place et à le réparer, ce qui nous a conduit à finalement décider de le garder. Plutôt que d'en commencer un quelques semaines plus tard, et prendre beaucoup de retard, donc risquer de ne pas terminer le projet à temps.

C'est comme ça que nous avons passé au moins 4 séances complètes à visser, dévisser, calibrer chaque servomoteur, comprendre comment appréhender ce projet. Avant de construire le code de la marche, l'enjeu principal était mécanique : faire en sorte que le corps, de quelques kilos, puisse se soulever lui-même, sans intervention extérieure.

On a passé du temps sur chaque servomoteur car notre montage n'étant pas précis à la microseconde près, les servos n'ont jamais été monté de manière exactement identique. Cela a résulté à devoir apprendre chaque position de chaque servomoteur, son angle minimum, son angle maximum, afin de pouvoir assurer une bonne remise en état du projet, et espérer le faire totalement marcher sur ses six pattes.

On est arrivé à ceci :



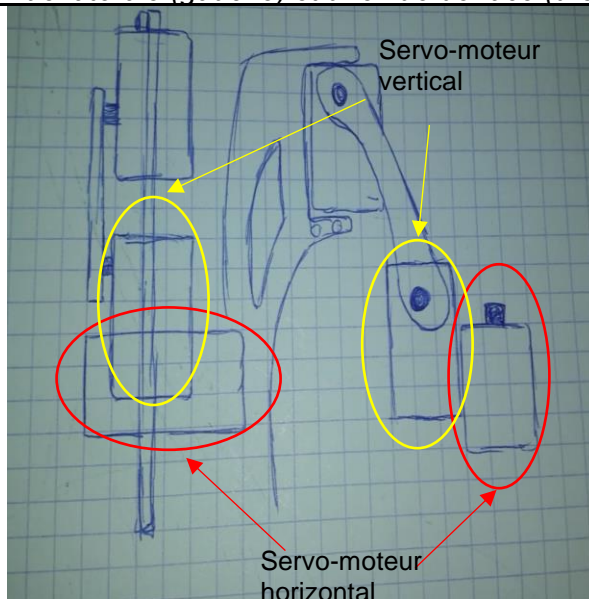
2. ETUDE DU MOUVEMENT HEXAPODE :

L'étude du mouvement d'un hexapode (à 6 pattes, donc tout type d'insecte) a été très fastidieuse puisque nous avons commencé par regarder beaucoup de reportages et de documentaires sur les araignées pour les regarder marcher. Nous voulions créer notre propre mouvement de marche, mais deux problèmes persistaient :

1. Une araignée marche à 8 pattes, alors que nous n'en disposons que de 6.
2. Une vraie araignée peut avancer une ou deux pattes sans avoir à lever les autres, puisqu'elle est articulée, sauf que nous ne pouvons pas faire ça car si une patte avance pour faire se déplacer l'araignée, elle va forcer sur tous les autres moteurs ce qui pourrait les endommager, voire les casser.

Pour expliquer ce deuxième point, voici un dessin qui peut illustrer la composition d'une patte :

Schéma d'une vue latérale (gauche) et une vue de face (droite) d'une patte de l'araignée



Nous avons donc choisi de reproduire un mouvement de robot hexapode après avoir regardé des dizaines de vidéos (de projets arduino ou raspberry). Le mouvement de marche peut être décomposé en 8 temps, comme décrit dans le tableau ci-dessous :

MOTEUR VERTICAL	MOTEUR HORIZONTAL
Down	Back
Down -> Up	Back -> Middle
Up -> Down	Middle -> Front
Down	Front
Down	Front -> Back
Down	Back

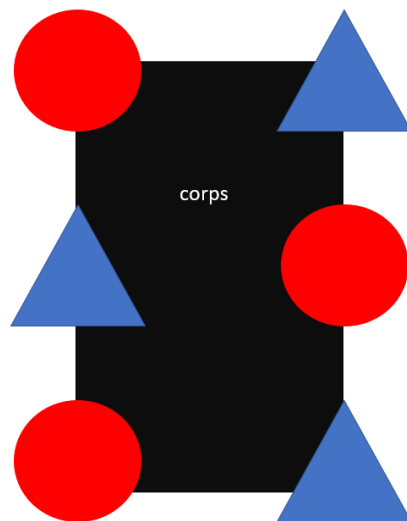
Chaque ligne s'exécute en même temps.

Pour simplifier:

- Avancer:

- Le moteur horizontal tourne (d'une amplitude encore indéterminée à cette étape) de sorte que la patte aille vers l'avant en quatre temps qui comprennent deux temps de pause, à savoir le premier, qui se situe en position arrière et le dernier en position avant (en passant de l'arrière au milieu, à l'avant). Ces temps de pause servent à mieux visualiser le mouvement, en réalité, ils seront retirés plus bas.
- Pendant ce temps-là, le servo-moteur horizontal effectue également un mouvement en 4 temps qui comprend également 2 temps de pause (eux aussi retirés plus bas). Le mouvement consiste à lever la patte, dans le même qu'elle avance. Lorsque le servo-moteur horizontal est arrivé à la moitié de son amplitude, le servo-moteur horizontal redescend pour reposer la patte.
- Reculer:
 - Le moteur horizontal effectue le mouvement inverse en un temps de sorte de revenir à sa position "back" initiale puis reste un temps en arrière
 - Pendant ce temps, le moteur vertical reste en position basse.

Ce mouvement décrit ci-dessus est le mouvement d'une patte. Pour coordonner le tout, on applique ce mouvement aux pattes que l'on met en mouvement par bloc de 3 :



Lorsque le bloc des rouges se pose (arrivé au temps 4), le bloc de bleu se lève pour ne pas gêner le recul du rouge et surtout ne pas endommager ses moteurs horizontaux.

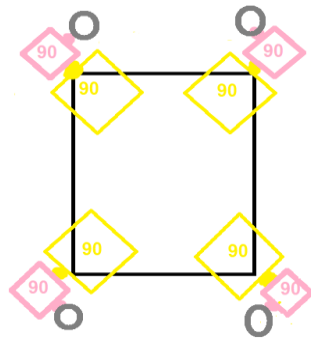
On avait décidé de miser sur l'élégance de la marche, en faisant monter bien haut les pattes lorsqu'elles se déplaçaient. Tous les tests ont été faits sur une potence en bois, mais une fois posée au sol, nous nous sommes rendu compte qu'il fallait donner des angles totalement simplifiés pour que toutes les pattes puissent être alignées, mais ce sera expliqué plus bas.

3 .RECONSTRUCTION (CALIBRER LES MOTEURS EN FONCTION DE NOTRE ETUDE)

Comme dit plus haut, il a fallu manipuler les servomoteurs un à un et à aucun moment nous n'avons pu reproduire les positions d'un servomoteur sur un autre.

Chaque angle étant unique et exprimé en microsecondes, nous avons beaucoup effectué de tests pour déterminer le bon positionnement respectif des servos horizontaux, verticaux et du haut d'une patte.

Nous avons donc positionné les servo horizontaux et verticaux à 90° et nous les avons montés de façon qu'ils s'alignent avec l'orientation du corps, comme montré sur le schéma ci-dessous :

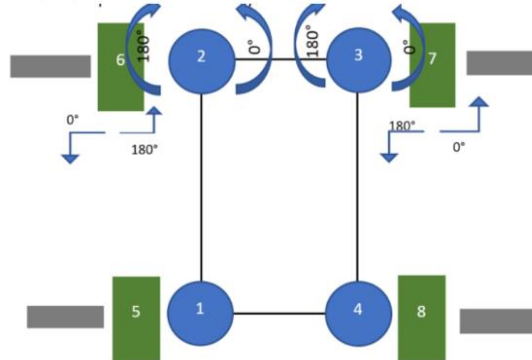


Au repos, on constate que les servos sont montés dans cette direction, avec un angle de 90° .

Les servos jaunes sont les servos horizontaux, les roses sont verticaux et les ronds gris représentent le reste des pattes.

Avec ce montage, on peut ainsi avoir une amplitude allant de 0° à 180° pour chaque servo, afin que leur position de départ soit le milieu de leur amplitude.

On a pu construire le schéma suivant, qui représente donc les différentes amplitudes des servos :



On comprend bien avec ces 2 schémas l'intérêt de positionner les servomoteurs lors de leur montage selon un angle précis, afin de contrôler au mieux leur mouvement par la suite.

Ensuite, les six derniers servomoteurs ont été montés soit avec un angle de 180° soit avec un angle de 0° selon le côté où ils se trouvaient. En effet, à l'inverse des 12 autres, que nous avons monté de sorte qu'ils puissent balayer leur amplitude de 90° vers l'avant et de 90° vers l'arrière, nous avons plutôt décidé de mettre une position maximale (respectivement minimale) sur les derniers.

Ainsi, on les a montés dans le prolongement de la patte (du châssis) et selon qu'ils sont à 180° ou 0° , ils parcourent l'intégralité de leur amplitude pour descendre vers le sol. Une fois l'intervalle entièrement balayés, ils doivent toucher le sol.

Cette dernière manipulation était plus simple à réaliser que si nous avions monté ce dernier avec une amplitude préétablie de 90° , car il aurait fallu trouver l'angle droit correspondant sur la patte (en le montant avec exactitude sur le châssis). Comme nous avons fait tous les montages à la main, sans instruments, ils sont donc très peu précis, et de fait, avoir juste à trouver le prolongement du châssis pour monter la patte s'est révélé plus facile pour nous.

De plus, le montage était compliqué car les moteurs horizontaux sont encastrés dans le châssis, et il vaut mieux les insérer après que le châssis soit vissé au corps, donc il faut un peu forcer pour le faire rentrer, mais s'assurer de ne pas le casser, ou l'abîmer.

Cette partie s'est terminée avec de longues recherches, beaucoup de tests, et de la dextérité lors du montage.

De plus, nous avons rencontré un problème lorsque nous avons posé l'araignée sur le sol ; les pattes glissent et de ce fait dévient la trajectoire.

Nous avons donc opté pour des petits patins en latex ; nous avons fixé des morceaux sur le bout des pattes, mais ceux-ci s'usent beaucoup trop vite et n'adhèrent pas aussi bien au sol que ce que nous espérions.

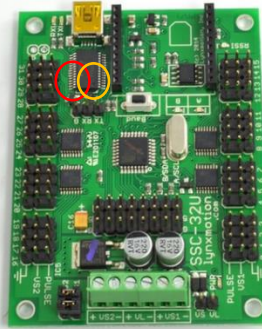
Nous avons donc utilisé un tapis de yoga pour faire marcher l'araignée sur une surface la moins lisse possible, mais le tapis était étroit et pas très long, nous avons dû la repositionner à de très nombreuses reprises, et il n'était pas très pratique de la faire évoluer dans ces conditions.

Cependant, malgré ce petit problème, elle peut quand même se déplacer, même si les moteurs forcent énormément et s'abîment relativement vite (et les boulons se desserrent ou tombent).

Nous avons juste scotché des morceaux de plastique sous tous les modules que nous utilisons car ils ont des composants métalliques, or l'araignée est en métal, ce qui pourrait causer des courts-circuits.

4-PRESENTATION ET CONNEXION DU MODULE SSC-32U

Avec l'araignée était fournie un module de type SSC-32U qui peut contrôler jusqu'à 32 servo-moteurs simultanément comme ci-dessous :



Ce module est conçu pour qu'on lui envoie directement des informations par un câble également fourni grâce à des lignes de commande du type :

```
Serial.println("#0 P750 #4 P750 #11 P750 T500");
```

Or nous faisons un projet Arduino, et avec ce module il n'y avait pas besoin de carte arduino. On a donc utilisé une communication RX TX, (comme pour communiquer avec un module bluetooth) en déclarant les ports 4 et 5, respectivement en RX et TX dans un objet de type "softwareSerial" que nous avons appelé SSC (comme le module) comme il suit :

```
SoftwareSerial SSC(RX, TX);
```

Nous ne nous servons que du port 5 (le port TX) puisque nous envoyons simplement des informations depuis l'arduino, le module ne nous communique rien.

Pour ce qui est des branchements, on a branché le G (entouré en rouge) au GND de l'arduino, puis seulement le RX (entouré en Orange) au port analogique 5 de l'arduino, il n'était pas nécessaire de brancher le TX du module puisqu'il ne nous communique aucune information. Puis pour l'alimentation, nous avons une batterie à brancher sur secteur de type "S-200-5" délivrant 5V et 40A, très peu pratique étant très lourde (gênant le mouvement) et très puissante, ce qui nous a contraint à mettre des fils de cuivre de plus gros diamètre que pour le reste, donc beaucoup plus rigide (nous avons fini par trouver un fil de cuivre beaucoup plus souple et beaucoup plus long qui a libéré le mouvement). De cette alimentation, nous avons mis 2 fils aux bornes + et 2 aux bornes -, que nous avons vissé au module respectivement au VS1+, VS2+ et VS1-, VS2- pour pouvoir fournir assez de courant et de tension à tous les moteurs (en plus de l'arduino).

Parlons maintenant du "baud rate", dans les forums visités qui montraient comment se servir des modules SSC-32U, ce baud rate était à 115200 (rappelons qu'il se déclare dans les setups comme ceci : `Serial.begin(115200)` ;). Or dans les exemples regardés, on téléversait directement le code sur le module SSC-32U, qui devenait donc le Serial.

Jusqu'ici nous avons toujours mis le Serial à un baud de 9600, ne sachant donc pas comment s'y prendre, nous avons effectué plusieurs tests, en mettant le Serial à 9600 et le module à 115200, et inversement, puis les 2 à 115200, toujours rien ne se passait quand on envoyait des informations au module depuis la carte arduino. Puis en mettant les 2 (le Serial et le SSC) à 9600 nous nous sommes rendu compte que la carte arduino communiquait bien avec le module en envoyant des informations depuis le serial en mettant un "retour chariot". Mais nous ne pouvons pas baser tout notre mouvement en envoyant des commandes depuis le Serial, il restait donc à trouver comment envoyer des commandes depuis le code entré dans l'arduino.

Réussir à communiquer les informations au module était une autre affaire. En se servant de la doc du module, nous avons pu comprendre comment former une ligne de commande qui seront traduites par le module, avec quelques informations basiques :

`"#0 P1600 T1000"`

Par exemple permet d'envoyer du courant pendant 1600 μ s, ce qui pour nous équivaut à un angle de 90°, pendant un temps T de 1000 μ s. Dans tous les forums consultés, il n'y avait que des codes permettant de communiquer directement avec le module, et non depuis l'arduino, ce qui fait que les lignes de commande étaient envoyées au Serial comme ceci :

`Serial.println("#0 P1600 T1000");`

Or nous avons défini un objet de type `SoftwareSerial` désignant notre module, et jusqu'à présent, pour communiquer avec le module souhaité en RX TX, nous avons toujours utilisé la fonction objet `.write()`. Mais rien ne se passait lors du téléversement, en effet, le problème est le suivant : le module reçoit des lignes de commande encodées en ASCII 13, et qui de plus, ont un retour à la ligne à la fin. Nous avons donc changé pour un simple `"println()"` avec le module en objet courant, qui remplit ces deux conditions de la manière suivante:

`SSC.println("#0 P1600 T1000");`

Et c'est ainsi que nous avons pu commencer à coder le mouvement, comme décrit dans la partie 2.

5. LE COMPORTEMENT AUTONOME :

Pendant les dernières séances, alors que le robot marchait selon les programmes donnés, il n'était pas conscient de son environnement.

Puisqu'à la base de notre projet, nous voulions un petit robot de compagnie, nous voulions lui offrir un caractère propre, ainsi qu'un apprentissage de son environnement au fil de son utilisation.

On peut s'inspirer d'un robot préprogrammé qui est très souvent utilisé dans l'approche d'Arduino: le robot Thymio.

Si on analyse ses modes préprogrammés, on remarque que tous ou presque sont construits autour de l'équipement de capteurs infrarouges : afin d'analyser des obstacles par ses propres "yeux", le robot doit être capable de modéliser son environnement, et répondre en fonction de ce qu'il voit.

Il existe de nombreux comportements que peut adopter le robot face à un obstacle : le suivre si celui-ci est en mouvement, ou à l'inverse s'en éloigner, le contourner, etc...

Pour programmer Thymio, il existe les programmes Aseba ou Thymio VPL, et bien d'autres mais ceux-ci sont les principaux. Ils sont malheureusement incompatibles avec d'autres robots car les ports de l'Arduino sont assez instables notamment avec Aseba, comme le démontre un post trouvé sur le forum www.thymio.org :

"Aseba can be ported on any microcontroller supporting 16-bit integers and having enough (about >1 KB) memory. However, there is no stable port on Arduino yet, but some people are working on one: <https://github.com/aseba-community/aseba-targets-arduino>"

- Stephane Magnenat ; mars 2016

Ce qui s'est imposé à nous pendant la réalisation du projet, a été de savoir comment le robot se comporterait. Nous voulions un comportement très polyvalent, qui permette au robot de s'adapter à tout type d'environnement, et surtout se familiariser avec tous les objets que l'on peut retrouver dans un salon, ou une chambre par exemple.

Seulement, il nous fallait beaucoup de temps pour la rendre autonome, et si nous voulions aller au bout de nos idées, il aurait fallu programmer une petite intelligence artificielle, à termes.

Or le temps, c'est ce que nous manquait, nous arrivions à la fin du temps imparti.

Nous avons donc une autre alternative: la télécommande Bluetooth, comme vue en cours.

Selon nos choix, le matériel est différent. En effet, si l'on décide de contrôler le robot avec une télécommande, alors il nous faudra un module Bluetooth et coder les panels nécessaires au fonctionnement du robot (flèches directionnelles, slider pour contrôler la vitesse, et boutons pour actionner les différentes danses).

En revanche, si l'on décide de lui donner un comportement autonome, c'est-à-dire que l'humain n'aura pas un contrôle direct sur son comportement, alors il faudra un capteur infrarouge de distance. Cela implique que le robot, dès lors qu'il est alimenté, avance et se dirige dans son environnement sans ordre direct. D'après notre idée de départ, on voulait en faire une araignée de compagnie, qui saurait se repérer dans son environnement : elle avance jusqu'à croiser un obstacle, le contourne ou recule pour changer de direction.

Nous avons finalement opté pour la télécommande, très facile à gérer, puisque nous avons déjà eu l'opportunité de travailler avec ces modules.

Nous l'avons donc programmé, comme expliqué dans la partie ci-dessous :

6- LE CODE/ LE BLUETOOTH

Le module étant configuré, et l'étude de la marche terminée, nous nous sommes attelées à la partie code de notre projet. Tout a commencé à l'élaboration de la marche avant, qui a pris très longtemps. N'ayant pas fait un montage parfait, tous les servo-moteurs n'ont donc pas la même valeur d'angle pour une même position, de plus, les pattes avant pourront aller plus loin devant que les pattes arrière, puisqu'elles risqueraient de heurter le structure et de forcer en poussant dessus, ce qui pourrait fortement les endommager, voire les casser. On a donc testé tous les servo-moteurs horizontaux un à un, pour leur déterminer une plage d'amplitude qui leur est propre, que nous pourrions exploiter par la suite pour la marche (pour les autres, un tel calibrage n'est pas utile). Sachant que pour tous les servo-moteurs on a la même amplitude de départ (possible de l'atteindre à vide) de 0°-180°, qui pour nous correspond à des angles en micro-secondes de 850 μ s à 2700 μ s.

Puis nous avons déterminé que tous les moteurs horizontaux nécessitent d'avoir une même amplitude de mouvement, car sinon, la marche ne serait pas droite et aurait tendance à dériver du côté où les amplitudes de mouvement sont plus grandes. Par la suite, nous avons codé le mouvement de tous les 18 servo-moteurs présents selon le modèle de marche, et en prenant en compte le fait que les moteurs marchent en symétrie selon l'axe du corps. En effet, un angle qui correspondrait à la position avant serait 180° (~2700 μ s) du côté droit par exemple, et pour la même position du côté gauche serait 0° (850 μ s). Une fois la marche terminée, on l'a placée dans une fonction appelée '`forward()`'; nous avons repris exactement le même modèle pour coder les mouvement pour tourner à droite et à gauche, en augmentant respectivement les amplitudes des servo-moteurs horizontaux de gauche et de droite, puis en diminuant les amplitudes des servo-moteurs opposés. La fonction de marche arrière est faite exactement sur le même modèle que la marche avant, mais simplement inversé. On a également écrit un code de marche sur place, car on ne pouvait pas simplement stabiliser les moteurs en les amenant tous à 90° d'un coup, puisque cela aurait eu pour effet de forcer sur ceux qui se trouvaient loin puisque la patte ne se lève pas pour permettre d'accéder à cet angle. Puis on a ajouté un module bluetooth de type HC-06, que nous avons déclaré comme un objet de type '`softwareSerial`', comme le module SSC-32U, mais cette fois-ci, nous l'avons appelé Oups (c'est un petit rituel). Le HC-06 est un module de communication bluetooth de type esclave. Le bluetooth nous a permis de communiquer avec notre téléphone, sur lequel nous avons créé une télécommande sur l'application "`Bluetooth electronics`". Sur cette télécommande, il y a quatre boutons:

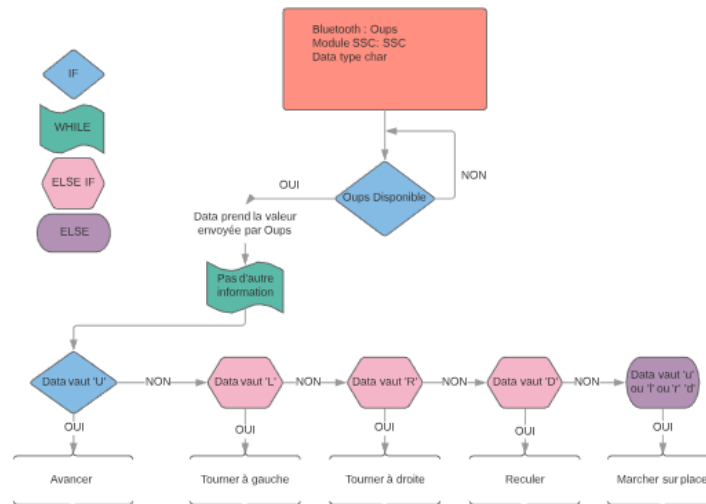


1. Celui avec la flèche vers le haut envoie un 'U' lorsqu'on appuie dessus, un 'u' lorsqu'on relâche
2. Celui avec la flèche vers la droite envoie un 'R' quand on appuie dessus, un 'r' quand on le relâche
3. Celui avec la flèche vers le bas envoie un 'D' quand on appuie dessus, un 'd' quand on le relâche
4. Celui avec la flèche vers la gauche envoie un 'L' quand on appuie, un 'l' quand on relâche

Voici l'algorithme assez basique de notre code :

ALGORITHME

Mathilde Croizat | March 12, 2019



Les fonctions de marches s'appellent :

- Pour avancer: `Forward()`
- Pour reculer: `Backward()`
- Pour tourner à droite: `Right()`
- Pour tourner à gauche: `Left()`
- Pour rester sur place: `stay()`

Au début, on utilisait des lignes de commandes écrites en type `String`, manuellement, du type :

```
SSC.println("#0 P2000 T1000 #1 P1600 T2000");
```

Mais avec toutes les fonctions de mouvement, la mémoire de l'arduino était presque remplie (à 95%). Il fallait donc optimiser le code, car il manquait de place et que nous souhaitions ajouter d'autres mouvements, mais également car lorsque la mémoire est saturée, les fonctions qu'on transmet à l'arduino risquent de ne pas être correctement exécutées. Dans notre code, le temps d'exécution demandé est toujours le même, à savoir `"T1000"` nous avons donc affecté cette valeur à une variable de type `Char`, qui est une chaîne de caractères beaucoup moins conséquente en taille que la même variable en type `String`. Nous avons également fait une fonction qui permet d'envoyer une commande au module en prenant en argument le moteur voulu avec l'angle souhaité, le temps (si nous voulions le changer par la suite) et un délai, comme ceci:

```
void moveMotor(int motor, int angle, int duration, long delai) {
    SSC.print("#");
    SSC.print(motor);
    SSC.print(" P");
    SSC.print(angle);
    SSC.print(" T");
    SSC.println(duration);
    delay(delai);
}
```

Puis après avoir retranscrit toutes les commandes de toutes les fonctions de mouvement avec cette fonction, nous avons pu compiler. Cette simple fonction nous a fait gagner une place inimaginable, puisque la mémoire est passée à 32%.

7.LE NON-RESPECT DU PLANNING

Nous avons prévu d'étudier une marche à 2 pattes. En effet, c'était pour comprendre l'équilibre d'un point de vue mécanique : comment répartir le poids d'un objet sur 2 pattes afin qu'il se déplace sans tomber. Mais puisque nous avons très rapidement remonté le robot avec 4 pattes, cette étape n'a pas eu lieu d'être, et nous n'y avons donc passé aucune séance dessus.

Nous avons donc directement appréhendé un mouvement qui nous a pris plus de temps à étudier que ce à quoi nous nous attendions.

En effet, nous pensions avant de commencer le projet, qu'avec toutes les études de mouvement à 2 et 4 pattes, la marche à 6 pattes coulerait sous le sens, et de ce fait serait rapide à programmer et mettre en place. Nous avons prévu de la terminer en une séance. Or, comme expliqué plusieurs fois, chaque servomoteur est unique. De ce fait, programmer 1 patte en elle-même est long et requiert de très nombreux tests. Il a donc fallu s'y atteler beaucoup plus longtemps qu'une seule séance.

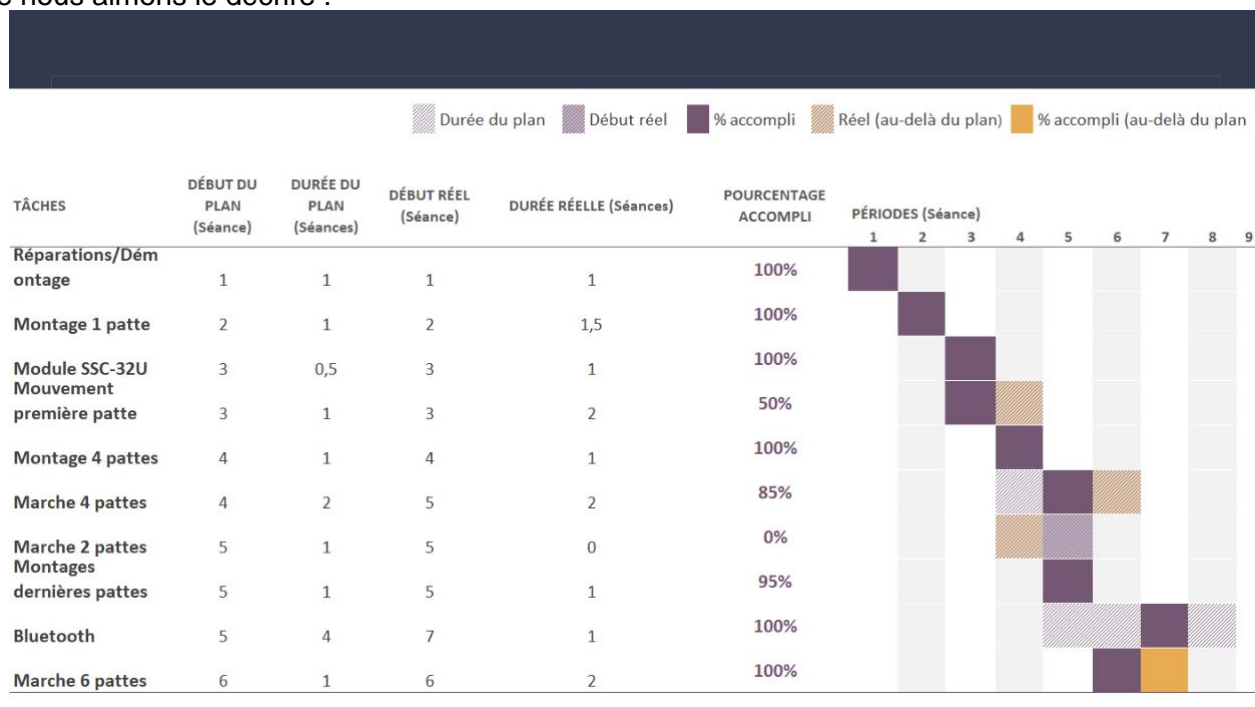
Ensuite, comme nous n'avons posé l'araignée sur le sol que lorsque nous avons abordé la marche à 6 pattes, nous nous sommes rendues comptes que le calibrage avec le niveau du sol n'était pas du tout bon, donc il a fallu démonter certains servos, et les recalibrer, ce qui nous a également pris beaucoup de temps.

C'est pour cela que la marche à 6 pattes fut beaucoup plus longue que prévue.

Puis il y a eu la situation inverse : nous avions prévu 5 séances pour ce qu'on a appelé le "Bluetooth", mais qui désignait plus la façon dont l'araignée devait se comporter. Comme nous n'avons pris la décision d'utiliser une télécommande que très tard, nous avons quand même prévu de quoi créer un comportement autonome, et programmer l'intelligence artificielle. Or, en ayant pris du retard sur le planning, nous avons manqué de temps, et nous n'avons utilisé que moins d'une séance pour mettre en place la commande sans fil.

Ces exemples montrent donc à quel point envisager un planning au tout début d'un projet, sans en connaître ses difficultés, ses spécificités et sans pouvoir anticiper les problèmes qui se posent lors de sa réalisation est compliqué et nous impose de planifier un enchaînement de tâche selon une durée erronée.

Voici en résumé un planning de Gantt, qui nous permet de visualiser facilement ce non-respect de planning, comme nous aimons le décrire :



CONCLUSION

1. Ce qu'on aurait pu faire avec 9 séances supplémentaires
2. Ressentis personnels sur le projet
3. Comparatif Attentes / Réalité

Ce qu'on aurait pu faire avec 9 séances supplémentaires :

Puisque le robot fonctionne, et est opérationnel, nous pouvons seulement améliorer la marche, donc retravailler sur les calibrages des servomoteurs, pour avoir un alignement qui tende à être parfait, et ne pas avoir de problème de déviation lors des déplacements de l'araignée.

On pourrait aussi s'occuper des patins que nous avons fixé au bout des pattes, qui malheureusement continuent de glisser sur le sol, et empêchent les pattes de se stabiliser, donc forcent sur les moteurs, et dévient la trajectoire, en plus de les abîmer par la même occasion. Lors de la journée portes ouvertes, Monsieur Masson nous a suggéré d'utiliser des morceaux de pneus, qu'on aurait pu utiliser pour obtenir une meilleure adhérence aux pattes.

Lors de cette journée porte ouverte, nous avons beaucoup mis notre robot à contribution, en effet nous l'avons beaucoup fait marcher ce qui a eu pour effet de desserrer beaucoup de boulons, certains même sont tombés, on aura donc dû les resserrer.

Enfin, on pourrait évidemment programmer notre intelligence artificielle pour faire d'Aragog un robot totalement autonome, qui puisse évoluer dans tout type d'environnement sans aucune distinction, et qui soit « à l'aise » avec le contact humain.

Ressentis personnels sur le projet

Nous exprimons une énorme fierté face à ce projet.

Nous avons été énormément soutenues et épaulées par notre enseignant, et les diverses personnes qui ont rencontré Aragog. Nous avons passé beaucoup de temps à faire aboutir notre projet le mieux possible, et nous sommes tristes de devoir lui dire adieux.

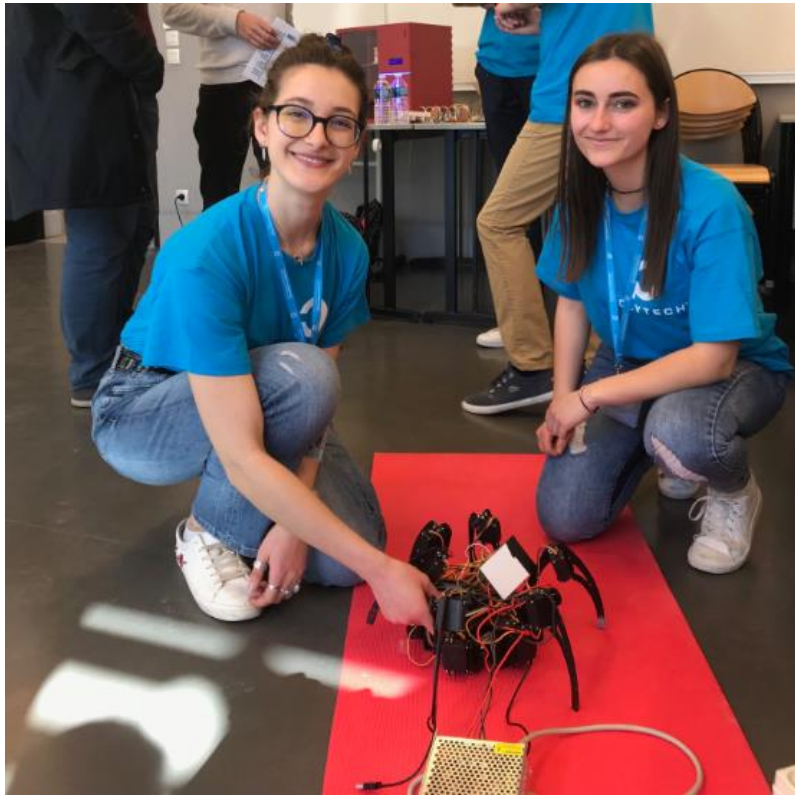
Nous bénéficions d'une grande expérience en Arduino grâce à nos nombreuses recherches, nos tests, et nos cartes brûlées.

C'est une réalisation riche d'enseignements sur bien des domaines ; en électronique évidemment, mais également sur notre capacité à gérer un projet en groupe, et apprendre à être patient, et autonome : gérer nos projets seuls est sans doute le meilleur enrichissement dont nous avons pu bénéficier.

Comparatif Attentes / Réalité

Nous sommes très satisfaites de notre réalisation.

Bien que l'on ait dû changer de route plusieurs fois au cours du projet, nous sommes satisfaites du rendu final, et bien que le robot soit encore largement améliorable, nous avons, selon nous, atteint nos espérances en réussissant à faire marcher l'araignée.



BIBLIOGRAPHIE :

Nous avons évidemment utilisé les cours de notre enseignant M. Masson, ainsi que d'autres ressources en ligne:

Première prise en main du module SSC-32U :

<http://marc-tetrapod.blogspot.com/2012/10/arduino-ssc-32-servo.html>

Manuel d'utilisation du module SSC-32U

https://www.robotshop.com/media/files/pdf2/lynxmotion_ssc-32u_usb_user_guide.pdf

Premières observations du processus de marche d'un hexapode :

<https://www.youtube.com/watch?v=jkbwfoFylc4&t=137s>

Araignée paon :

<https://www.youtube.com/watch?v=IBkXcVVnS84>

Site d'achat de l'araignée (pour photos) :

<http://www.lynxmotion.com/c-117-phoenix.aspx>

Communication :

<http://users.polytech.unice.fr/~pmasson/Enseignement/Elements%20de%20robotique%20avec%20arduino%20-%20Communications%20RF%20-%20Projection%20-%20MASSON.pdf>

Comportement autonome:

<http://exercices.openclassrooms.com/assessment/604?id=4076871&slug=sinitier-a-la-robotique&login=8192423&tk=c97dbd8813e822e407e4457dbe611096&sbd=2016-02-01&sbdtk=fa78d6dd3126b956265a25af9b322d55>