IMA 4

ROBOT HEXAPODE POUR ESCALIER

INFORMATIQUE, MICROÉLECTRONIQUE ET AUTOMATIQUE

EDUARDO GÓMEZ RIAZA

Encadrants: Alexandre Boé -- Xavier Redon --Thomas Vantrovs

Rapport de projet | Polytech Lille | Université de Lille 1

Remerciements

Tout d'abord, je tiens à remercier les professeurs d'Informatique, Microélectronique et Automatique (IMA) qui m'ont enseigné les savoirs et connaissances utiles et nécessaires pour la réalisation de ce projet.

Ensuite, je remercie également mes tuteurs de projet, Monsieur Alexandre Boé, Monsieur Xavier Redon et Monsieur Thomas Vantroys pour leur aide et conseils.

De plus, je souhaite remercier les responsables du fabricarium pour leur aide et leurs conseils sur la conception mécanique.

Enfin, un merci particulier à Monsieur Thierry Flamen pour le temps et l'aide qu'il m'accordé pendant la réalisation de ce projet.

Sommaire

1. INTRODUCTION

Dans le 4ème année d'Informatique, Microélectronique et Automatique je dois faire une projet pour améliorer mes connaissances et travailler sur la recherche.

Le projet consiste à faire un robot qui ressemble à une araignée, qui est capable de marcher et de monter les escaliers et de le localiser en mesurant la puissance du wifi. Le projet comprend différents domaines: mécanique, informatique et électronique.

La partie mécanique du projet consiste en un mouvement du robot à travers des servomoteurs qui lui permettent de marcher et de monter des escaliers.

La partie informatique du projet consiste à programmer le robot via Arduino pour contrôler les servomoteurs. De plus, on doit configurer le Wifly shield d'Arduino pour se connecter à un réseau Wi-Fi, puis mesurer sa puissance pour localiser le robot.

La partie électronique est basée sur l'alimentation du robot et des servomoteurs pour son bon fonctionnement, ainsi que sur les connexions entre l'Arduino et les servomoteurs.

On peut découper le projet en plusieurs phases : La fabrication d'un robot hexapode avec des servomoteurs où on doit obtenir un design approprié du robot, la programmation du robot pour marcher et monter des escaliers, et finalement mesurer la puissance du Wi-Fi pour localiser le robot.

Dans ce rapport je vais expliquer chaque phase détaillée.

2. CHOIX TECHNIQUES : Matériel et logiciel

On doit d'abord analyser la structure du robot :

Le corps du robot sera créé en utilisant l'imprimante 3D, il sera donc nécessaire d'obtenir les dessins au format STL, qui est le format utilisé par l'imprimante. Après, on va utiliser les imprimantes 3D du Fabricarium pour obtenir les pièces. Pour pouvoir utiliser les imprimantes, il est nécessaire d'effectuer une formation préalable, donnée par l'un des fabricants du Fabricarium.

Pour pouvoir joindre chacune des pièces du robot, on utilisera la colle forte et des vis. En utilisant les vis, on s'assure que les pièces restent ensemble mais qu'elles peuvent tourner entre elles, de manière à ne pas gêner le mouvement.

Le robot ressemble à une araignée, mais il aura six pattes au lieu de huit. Chaque patte aura trois servomoteurs pour obtenir une mobilité complète, donc on aura besoin de 18 servomoteurs en tout pour pouvoir construire le robot.

Les servomoteurs seront contrôlés par un Arduino. Comme ils sont 18 ans et ils utilisent les pins digitales, on aura besoin d'un Arduino Mega pour pouvoir les contrôler tous. En outre, on aura besoin de suffisamment de fils pour connecter chacun des servomoteurs à l'Arduino et à l'alimentation.

Quant à l'alimentation, on peut alimenter les servomoteurs et l'Arduino avec la même batterie 5V, bien qu'il soit probablement nécessaire d'utiliser plus de batteries en parallèle puisque les servomoteurs consomment beaucoup de courant.

On va également ajouter un capteur de distance pour être en mesure de détecter les escaliers. On peut donc ajouter les deux programmes sur l'Arduino (marcher et monter) et exécuter celui qui correspond à chaque instant.

La connexion au Wi-Fi et la mesure de la puissance seront faites en utilisant un Wifly shield, qui est un ajout d'Arduino, et pour cela un Arduino Uno sera utilisé.

On peut voir ici un bref résumé du matériel nécessaire :

Imprimante 3D	Servomoteurs	Capteur distance
Designs STL	Arduino Mega	Wifly shield
Colle forte	Fils	Arduino Uno
Vis	Ecrous	Batteries 5V

2.1 Choix des servomoteurs et des batteries

Il est important de savoir quels servomoteurs utiliser, car il se peut que l'on ne puisse pas supporter le poids du robot. Il est donc nécessaire de faire une étude sur la puissance des servomoteurs pour pouvoir choisir.

Le robot aura 18 servomoteurs, un Arduino Mega, une *protoboard*, 6 pattes, 2 parties centrales et des fils. On doit étudier le poids de chaque élément.

L'Arduino Mega pèse 55 grammes, et la *protoboard* 40 grammes. Avec la simulation du logiciel de l'imprimante 3D on peut voir le poids de chaque pièce qu'on va imprimer, donc le poids de la structure du robot est 240 grammes.

Les servomoteurs pèsent plus ou moins 10 grammes chacun, donc le poids total sera 180 grammes.

Si on ajoute tous les poids, on obtient un total de 515 grammes (on peut négliger les fils).

Le paramètre principal des servomoteurs c'est la force. On va utiliser des servomoteurs SG90, qui ont un couple égal à 1.8 kg*cm.

Si on fait la conversion de la force au couple :

$$C = 1.8\, kg \cdot cm \times 9.8\, \frac{N}{kg} \times \frac{1\, m}{100\, cm} = 0.1764\, N \cdot m$$

$$P = C \times w = 0.1764 \times 10.47\, \frac{rad}{s} = 1.847\, W$$

$$I = \frac{P}{V} = 0.369\ A$$

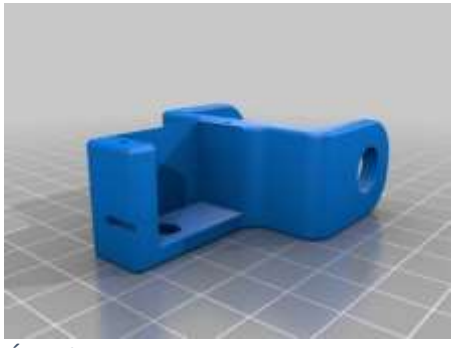
Donc, avec ce servomoteur on aura besoin de 0.369 A au minimum pour avoir la force maximale. En pratique, cette valeur peut être plus grande puisque le rendement d'un servomoteur est relativement faible, en raison du frottement mécanique élevé à l'intérieur.

On a 18 servomoteurs, donc si on veut utiliser toutes les servomoteurs avec sa force maximale on doit alimenter le robot avec 6.65 A au minimum. On utilisera les batteries 5 V en parallèles.

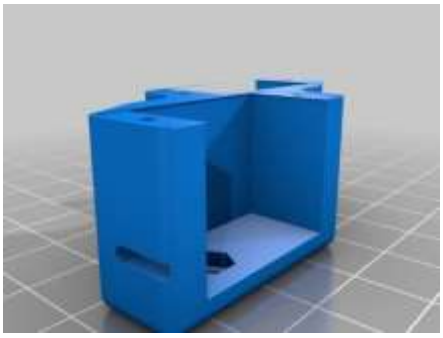
3. CONSTRUCTION DU ROBOT

3.1 Construction d'une patte

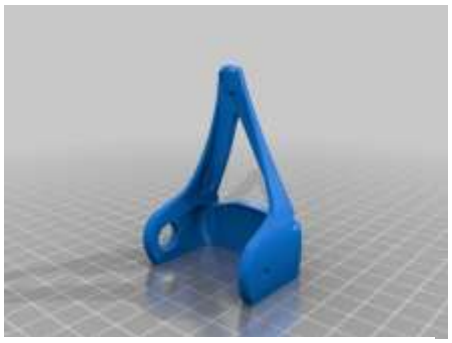
Pour construire une patte on va d'abord regarder sa structure: Une patte est composée de trois parties, qu'on appellera épaule, fémur et tibia. Le design est obtenu de *thingiverse* :



Épaule



Fémur



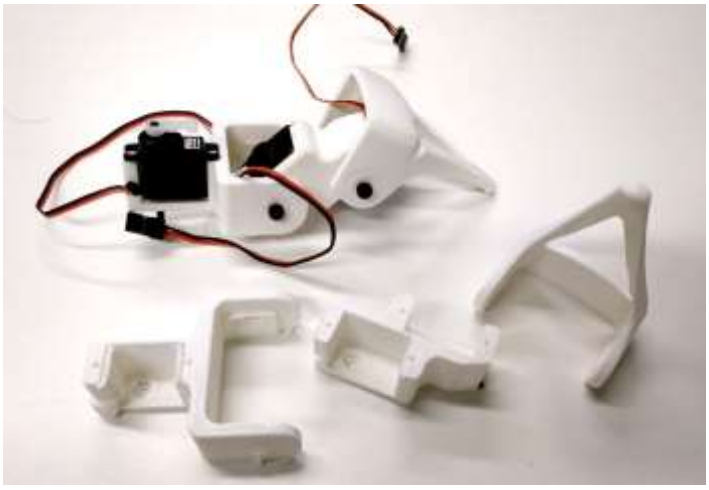
Tibia

On a fait des modifications sur ces designs :

- Le design originale est conçue pour utiliser un roulement d'un côté et pour coller le servomoteur à l'autre. L'utilisation des roulements n'a pas été nécessaire, on les a donc remplacés par des vis. Pour ce faire, il a fallu réduire la taille du trou destiné au roulement à 3 mm de diamètre, parce qu'on utilise des vis M3.

- Le trou préparé pour les fils du servomoteur est très petit et, selon la précision de l'imprimante 3D, il peut ne pas suffire que le câble le traverse. De plus, étant si petit, il est difficile d'enlever l'excès de matériau utilisé par l'imprimante pour remplir les espaces. Par conséquent, on a augmenté les dimensions du trou.

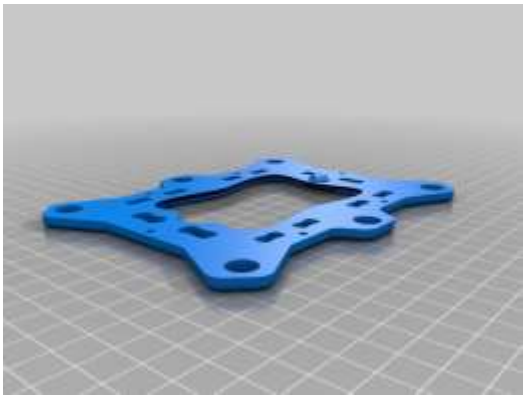
Les servomoteurs sont vissés sur les pièces à l'aide de vis M2 afin qu'elles ne se séparent pas de la pièce et augmenter la précision du mouvement. Dans le design il y a des petits trous pour les vis, donc on n'a pas besoin des écrous ici.



Ci-dessus, on peut voir une patte et les composants séparés.

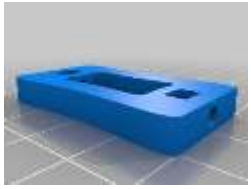
3.2 Assemblage des pièces

Le corps du robot se composera de deux parties grandes et identiques entre lesquelles les pattes se joindront à travers de l'épaule. La pièce utilisée est à droit.



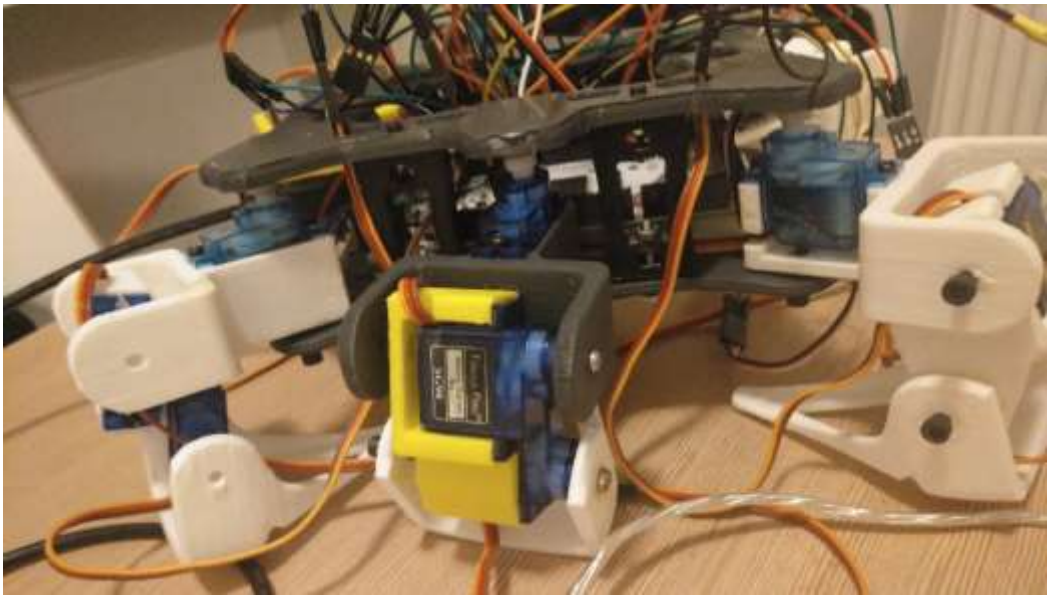
On va faire les mêmes modifications dans les trous que dans les autres pièces pour utiliser des vis et pas des roulements.

Pour assembler les servomoteurs au corps du robot on va utiliser de la colle forte.



On utilisera cette petite pièce pour unir la partie haute et la partie basse du robot et éviter que les servomoteurs se séparent d'une partie á cause du poids.

Une fois l'assemblage des six pattes et l'union des pattes au corps, on obtiendra le robot comme indiqué sur l'image:



4. PROGRAMATION DU ROBOT

4.1. Connection

Avant de commencer à programmer le robot, il est nécessaire de connecter tous les servomoteurs. Pour eux, on les a connectés via des fils à une *protoboard* à laquelle on connecte également la batterie. De cette façon, on va alimenter tous les servomoteurs en parallèle (5 V pour chaque servomoteur).

En utilisant un fil, on connectera également chaque servomoteur à l'une des pins digitals Arduino. Dans le programme, il sera nécessaire de spécifier quel numéro de pin est lié à chaque servomoteur.

L'Arduino est alimenté aussi avec 5 V, donc on peut le connecter à la *protoboard* ou directement à la batterie.

4.2 Logiciel IDE Arduino

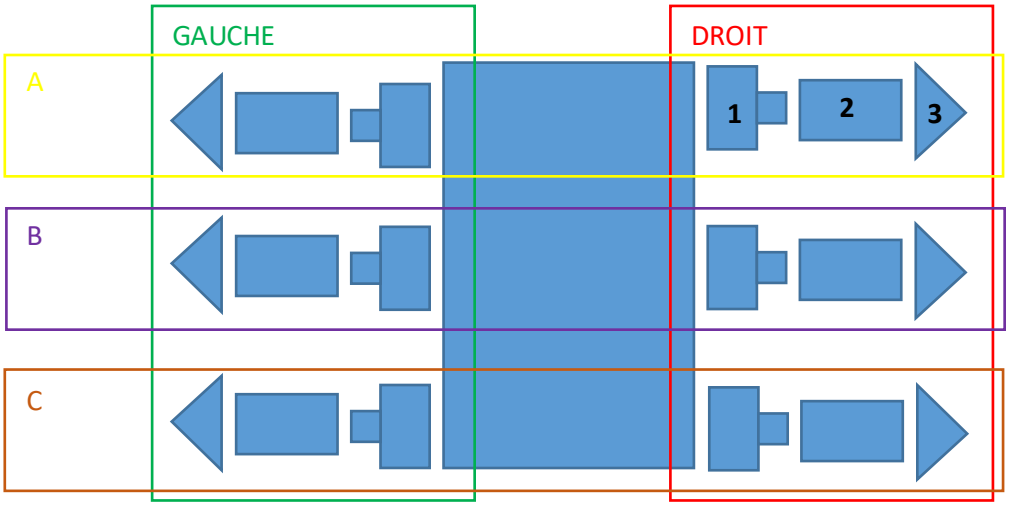
Pour faire le programme du robot on va utiliser le logiciel IDE Arduino. On va travailler avec des servomoteurs, donc on doit télécharger la librairie *Servo.h* pour faciliter le travail. Avec cette librairie on peut contrôler les servomoteurs en précisant seulement la position en grades à laquelle on veut aller. Par exemple :

```
#include <servo.h>

Servo Moteur1 ;

[…]

Moteur1.write(90) ;   //Le servomoteur tourne jusqu'à 90º
```

On a 18 servomoteurs, donc j'ai choisi une méthode pour les nombrer :

Les pattes sont divisés en trois paires (A=avant, B=Au milieu, C=Derrière), elles peuvent être à droit (D) ou à gauche (G) et le servomoteur peut être dans l'épaule (1), le fémur (2) ou la tibia (3).

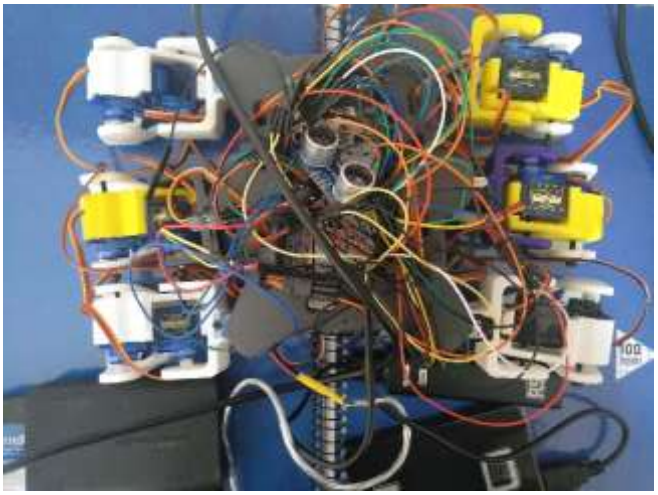
Donc, le nom d'un moteur sera (A, B, C) + (G, D) + (1, 2, 3).



Pour réaliser le programme, la première étape sera d'initialiser les servomoteurs. Pour ce faire, on doit attribuer le pin correspondant à chaque servomoteur. On utilisera la commande *attach* :

```
AD1.attach(2);          // le servomoteur AD1 correspond au pin 2
```

Maintenant, on doit choisir la position initiale des servomoteurs. La position de repos du robot consiste à avoir les pattes perpendiculaires au robot et formant un angle de 90 degrés, de sorte que les servomoteurs puissent mieux supporter le poids :



Pendant la construction, les valeurs initiales peuvent être différentes pour chaque servomoteur. Ça dépend de comment on a collé le servomoteur. Il faut donc tester chaque servomoteur pour voir à quelle valeur il prend la position qu'on veut. Pour les mouvements futurs, on se basera sur cette valeur initiale, il ne sera donc pas nécessaire de re-tester chaque moteur pour chaque position.

On peut voir les positionnes initiales trouvés pour ce robot dans le programme de l'annexe.

Pour marcher, la méthode qu'on va utiliser est la suivante :

1. Avancer les pattes A (d'avant)

2. Avancer le corps en tournant toutes les servomoteurs des épaules au même temps

3. Avancer les pattes B

4. Avancer les pattes C (on revient à la position initiale)

On va ajouter un délai entre chaque mouvement d'un moteur pour laisser le temps à la rotation de se terminer et ne pas déstabiliser le robot. De plus, lorsque plusieurs servomoteurs se déplacent en même temps, il y aura un délai de 10 ms entre eux pour que l'Arduino puisse faire un à un les impulsions.

Pour faire un délai on utilisera la commande *delay(t) ;* où t c'est le temps en ms.

Lorsque le robot monte les escaliers on utilisera le même méthode pour avancer. Quand on monte les escaliers on doit prendre en compte la position des pattes élevées par apport à la position des pattes au sol, parce que le robot peut tomber si l'inclinaison est trop grande. La méthode sera :

1. Monter les pattes A

2. Élever le robot pour maintenir la stabilité.

3. Avancer le robot (avec les pattes A élevées)

4. Monter les pattes B

5. Incliner le robot vers l'avant en étirant les pattes C pour maintenir la stabilité quand on essaie de monter les pattes C

6. Avancer (avec les pattes A et B élevées)

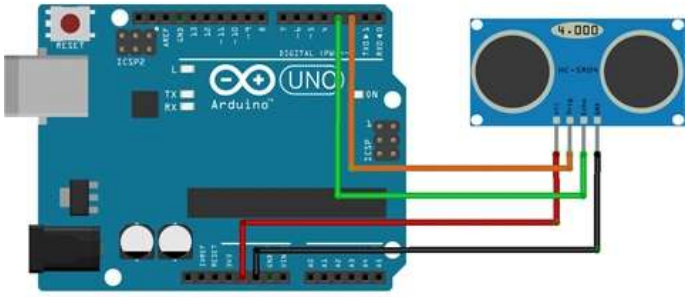
7. Monter les pattes C

8. Marcher (comme dans le cas précèdent)

Il est important que le tibia soit perpendiculaire au sol, sinon le servomoteur pourrait ne pas supporter la totalité du poids (car si le tibia est incliné, la force nécessaire pour supporter le même poids est plus grande).

4.3 Capteur de distance

On va également ajouter un capteur de distance pour détecter les escaliers. Le capteur de distance le plus approprié si on utilise Arduino sera le HC-SR04, car il utilise la même tension de 5 V comme source d'alimentation.



Le capteur a quatre I/O qui doivent être connectés comme indiqué sur l'image : Deux I/O pour l'alimentation et deux I/O pour la commande. Les I/O de la commande seront le Trigger et Echo, et on doit l'indiquer sur le programme comme sortie ou entrée. On peut voir ici un exemple de l'initialisation du capteur dans le programme :

```
const int Trigger = 46;            //pin choisi pour Trigger
const int Echo = 48;               // pin choisi pour Echo


void setup() {
 Serial.begin(9600);               //on initialise la communication
 pinMode(Trigger, OUTPUT); //pin de sortie
 pinMode(Echo, INPUT);             //pin d'entrée
 digitalWrite(Trigger, LOW);   //initialisation à 0
}
```

Pour obtenir la valeur de la distance on doit envoyer un pulse et après écouter avec le capteur. On obtiendra la valeur du temps entre les deux. Comme on sait la vitesse de propagation d'une onde, on peut donc changer cette valeur à cm :

$v = \frac{d}{t} \rightarrow d = v * t \rightarrow d = \frac{t}{59}$ (pour avoir la distance en cm)

Ici, on peut voir un exemple du programme pour mesurer la distance :

```
digitalWrite(Trigger, HIGH);
delayMicroseconds(10);          //On envoie une pulse de 10 us
digitalWrite(Trigger, LOW);


t = pulseIn(Echo, HIGH);        //On écoute le retour
d = t/59;                       //On calcule la distance à partir du temps
```

Par conséquent, pour utiliser le capteur, on enverra une impulsion à chaque instant et, lorsque la distance est inférieure à une valeur prédéterminée (5 cm par exemple), on arrêtera le programme.

5. MESURE DE LA PUISSANCE DU WI-FI ET LOCALISATION

L'objectif c'est de obtenir la puissance Wi-Fi, donc si on fait une première cartographie avec le robot en mesurant la puissance des différentes bornes wifi présentes puis quand on veut se localiser, on remesure la puissance sur le Wi-Fi et on fait peut savoir où on est en comparant avec les mesures enregistrées.

Pour cela, on utilisera un Wifly shield qui nous permettra de nous connecter au réseau Wi-Fi. L'appareil qui nous est fourni est le Wifly RN-131G. La connexion avec l'Arduino UNO c'est la suivante :

La première place, on doit configurer le wifi pour se connecter à un réseau Wi-Fi. Ceci peut être fait en le configurant depuis un ordinateur et le terminal Arduino, ou en utilisant le Wifly comme un réseau Ad-hoc.

On va réaliser la configuration avec le terminal Arduino. On va télécharger une librairie qui nous permettra d'accéder à Wifly depuis le terminal :

https://github.com/sparkfun/SparkFun_WiFly_Shield_Arduino_Library

On doit ouvrir le SpiUart Terminal, qui est un exemple de cette librairie pour modifier les paramètres du Wifly. Après avoir téléchargé l'exemple à l'Arduino, on peut ouvrir le moniteur série.

Le shield a deux modes: data mode et command mode. Pour changer à command mode on doit écrire $$$, donc on aura la réponse CMD. Maintenant on peut configurer le Wifly. Les commandes qu'on doit écrire sont:

- Set wlan join 1 --> se connecter au réseau avec le ssid déterminé
- set wlan chan 0 --> il cherche toutes les canaux
- set wlan ssid AquarisV --> se connecter au Wifi de mon portable
- set wlan pass _____ --> mot de passe
- save
- reboot

Ici on a configuré le Wifly pour se connecter au réseau Wi-Fi de mon portable. Quand l'Arduino se connecte au Wi-Fi, il donne l'IP que lui donne le portable, dans notre cas c'est 192.168.43.103.

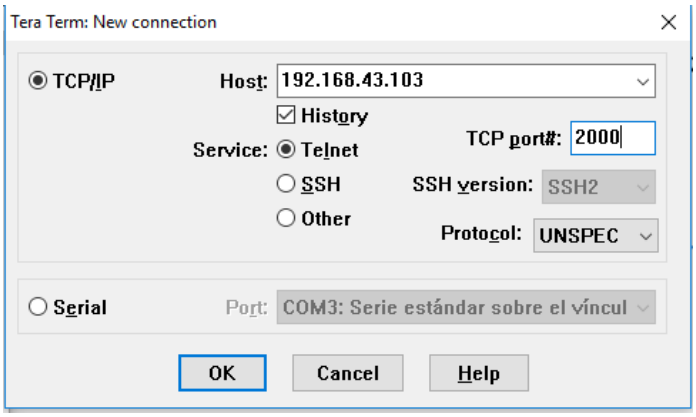
Pour obtenir la puissance du Wi-Fi on doit utiliser la commande *show rssi*. La réponse sera une valeur en dB. On peut vérifier que si on s'éloigne du portable, qui est l'origine du réseau dans ce cas, la valeur de la puissance sera plus faible. Donc si on fait une première cartographie de la puissance avec le robot, quand on remesure la puissance on sait la position du robot.

Pour pouvoir se connecter par TCP avec l'Arduino, on doit ouvrir a *socket*. Il permet d'envoyer et recevoir des donnés par TCP. Pour le configurer on doit écrire :

- set ip proto 0x2 --> pour utiliser TCP
- set ip host 192.168.43.103 --> On ajoute l'IP donné à l'Arduino
- set ip remote 2000 --> on precise le port
- open 192.168.43.103 2000 --> ouvrir la connexion

Le logiciel répondra *OPEN* si la connexion est bien fait.

Pour se communiquer avec l'Arduino via telnet on va utiliser le logiciel Tera Term. On doit écrire l'adresse IP de l'Arduino et le port (2000 dans notre cas).



Le logiciel se connectera avec l'Arduino et on pourra écrire les commandes :

- $$$ --> mode commande
- Show rssi --> Donner la valeur de la puissance en dB

Remarque : L'ordinateur doit être connecté au même réseau que l'Arduino pour se communiquer avec cette IP.

Si l'Arduino n'est pas connecté à l'ordinateur, on peut savoir son état avec les couleurs des leds :

Condition	PIO6=Red LED	PIO5=Yellow LED	PIO4=Green LED
ON solid			Connected over TCP
Fast blink	Not Associated	Rx/Tx data transfer	No IP address
Slow blink			IP address OK
OFF	Associated		

6. TESTS ET PROBLÈMES RECONTRÉS

Lors de la construction du robot, le principal problème rencontré était que les servomoteurs utilisés étaient incomplets. Beaucoup des servomoteurs n'avaient pas la surface supérieure à laquelle la pièce doit être collée.

Avec les servomoteurs des pattes cela a pu être résolu au moyen d'une vis, puisque les articulations des pattes sont mieux saisies et ne supportent pas tant d'effort de flexion, mais avec les servomoteurs du corps a été un vrai problème.

J'ai essayé de les coller avec différents types de colle, mais beaucoup d'entre eux n'étaient pas assez solides. Enfin, la colle forte est celle qui a le mieux fonctionné.

Même ainsi, les surfaces de collage n'étant pas très grandes, donc les servomoteurs doivent supporter une forte tension dans une zone très concentrée, et certains ont fini par se décomposer par le poids.

En ajoutant la partie inférieure du corps, les efforts de flexion ont été réduits et la situation a été améliorée.

L'idéal serait d'utiliser des servomoteurs qui ont leurs propres surfaces de collage, car ils seront mieux préparés pour supporter les efforts.

En effectuant les tests du robot, on a pu vérifier que si les servomoteurs sont alimentés avec une tension inférieure à 5 V, ils perdront de la force dans le mouvement. Cela peut poser un problème car si la batterie n'est pas complètement chargée, elle peut donner moins de tension et le robot peut ne pas être en mesure de se tenir debout.

En outre, si les servomoteurs ne reçoivent pas assez de courant, ils peuvent commencer à vibrer et échouer sans suivre le programme. Au début, cela ressemblait à un problème de l'Arduino, mais finalement, il pourrait être réparé en utilisant plus de batteries en parallèle, car les servomoteurs consomment beaucoup de courant.

7. AMÉLIORATIONS POSSIBLES

- Dans le projet de mécanique de ce même cours, nous avons pu travailler sur le logiciel Onshape qui permet de modifier ou de créer des dessins au format STL. Dans ce projet j'ai fait une petite amélioration dans la conception des pattes du robot pour diriger les fils des servomoteurs, de sorte qu'ils ne soient pas exposés et ne puissent pas s'emmêler avec le robot ou d'autres fils

On peut trouver les détails de ce projet sur la page Fabricarium ou sur le lien suivant: http://fabricarium.polytech-lille.fr/#!/projects/patte-d-un-robot-araignee.

- On peut essayer aussi d'ajouter le Wifly shield avec l'Arduino Mega qu'on utilise pour contrôler les servomoteurs et les capteurs. Pour faire cette connexion, il est nécessaire de changer certains pins qui ne peuvent pas être utilisées avec le Wifly :

D10	---------->	53
D11	---------->	51
D12	---------->	50
D13 ----------> 52		

- Si on utilise des servomoteurs suffisamment puissants, on peut essayer d'inclure les batteries à l'intérieur du robot. Pour cela, il faut utiliser une batterie moins lourde, par exemple une batterie LIPO, capable de fournir des valeurs de courant très élevées mais sans être trop grande pour le robot. Il faut beaucoup de courant pour les servomoteurs, il est donc difficile de trouver des batteries capables de donner autant d'e courant et elles sont donc plus chères.

8. CONCLUSION

Ce projet s'inscrit dans le cadre de mon quatrième année d'IMA (Informatique, Microélectronique et Automatique) à Polytech Lille.

L'objectif était de créer un robot hexapode avec des servomoteurs, donc j'ai fait aussi un travail de recherche afin de connaître le fonctionnement des logiciels, de l'Arduino, les capteurs et du Wifly.

Ce projet m'a permis la mise en pratique de mes connaissances et l'approfondissement de ces dernières (Arduino, programmation, mécanique et électronique). De plus, j'ai travaillé la plupart du temps en autonomie, ce qui m'a formé.

Ce projet m'a apporté beaucoup car j'ai fait de réels choix technologiques et matériels. De plus, j'ai pu travailler sur toutes les parties du projet : mécanique et logicielle. De plus, j'ai travaillé dans une langue autre que ma langue maternelle, ce qui m'a aidé à améliorer mon niveau dans cette langue.

ANEXE : Programme du robot

```
#include <Servo.h>

Servo AD1; Servo AD2; Servo AD3; Servo AG1; Servo AG2; Servo AG3;

Servo BD1; Servo BD2; Servo BD3; Servo BG1; Servo BG2; Servo BG3;

Servo CD1; Servo CD2; Servo CD3; Servo CG1; Servo CG2; Servo CG3;

int i;

const int Trigger = 46; const int Echo = 48;

void setup() {

Serial.begin(9600);

pinMode(Trigger, OUTPUT);

 pinMode(Echo, INPUT);

digitalWrite(Trigger, LOW);

AD1.attach(2); AD2.attach(3); AD3.attach(4);

AG1.attach(5); AG2.attach(6); AG3.attach(7);

BD1.attach(26); BD2.attach(22); BD3.attach(24);

 BG1.attach(36); BG2.attach(38); BG3.attach(40);

CD1.attach(8); CD2.attach(9); CD3.attach(10);

 CG1.attach(13); CG2.attach(11); CG3.attach(12);

}

void loop() {

long t; long d;

//initialisation servomoteurs

AD1.write(60); delay(10);

AG1.write(150); delay(10);

BD1.write(135); delay(10);

BG1.write(25); delay(10);

CD1.write(60); delay(10);

 CG1.write(80); delay(10);

 AD2.write(85); delay(10);

 AG2.write(90); delay(10);

 AD3.write(90); delay(10);

 AG3.write(90); delay(10);

 BD2.write(120); delay(10);

 BG2.write(115); delay(10);

 BD3.write(110); delay(10);

 BG3.write(100); delay(10);

 CD2.write(70); delay(10);

 CG2.write(55); delay(10);

 CD3.write(120); delay(10);

 CG3.write(90); delay(10);

 //fin initialisation

delay(1000);

 digitalWrite(Trigger, HIGH);

 delayMicroseconds(10);

digitalWrite(Trigger, LOW);

t = pulseIn(Echo, HIGH);

d = t/59;

if (d>5){

//Avance patte A droit

  AD2.write(115); delay(500);

 AD1.write(30); delay(500);

  AD2.write(85); delay(500);

 //Avance patte A gauche

AG2.write(120); delay(500);

 AG1.write(120); delay(500);

  AG2.write(90); delay(1000);

//Avance du corps
```

```
 CD1.write(90); delay(10);
 CG1.write(110); delay(10);
 BD1.write(165); delay(10);
 BG1.write(0); delay(10);
 AD1.write(60); delay(10);
 AG1.write(150); delay(10);
 delay(1000);
//Avance patte B droit
 BD2.write(150);  delay(500);
  BD1.write(135); delay(500);
 BD2.write(120); delay(500);
//Avance patte B gauche
 BG2.write(145); delay(500);
 BG1.write(20); delay(500);
 BG2.write(115); delay(500);
//Avance patte C droit
  CD2.write(100); delay(500);
  CD1.write(60); delay(500);
 CD2.write(70); delay(500);
//Avance patte A droite
 AD2.write(115); delay(500);
  AD1.write(20); delay(500);
 //Avance patte C gauche
  CG2.write(85); delay(500);
  CG1.write(80); delay(500);
 CG2.write(55); delay(500);
  AD2.write(85);
 delay(500);
}
```

```
//Programme pour monter
If (d < 5) {
//A droit monte
AD2.write(135); delay(500);
  AD1.write(30); delay(500);
  AD2.write(105); delay(500);
 //A gauche monte
  AG2.write(140); delay(500);
 AG1.write(120); delay(500);
  AG2.write(110); delay(1000);
//élever robot
 AD2.write(85); delay(10);
 AG2.write(90); delay(10);
 BD2.write(100); delay(10);
 BG2.write(95); delay(10);
 BD3.write(100); delay(10);
 BG3.write(90); delay(10);
 delay(1000);
 //avancer
 CD1.write(90); delay(10);
 CG1.write(110); delay(10);
 BD1.write(165); delay(10);
 BG1.write(0); delay(10);
 AD1.write(60); delay(10);
 AG1.write(150); delay(10);
delay(1000);
 //Avancer patte A droit en haut
AD2.write(105); delay(500);
  AD1.write(30); delay(500);
  AD2.write(85); delay(500);
 //Avancer patte A gauche en haut
  AG2.write(110);  delay(500);
```

```
 AG1.write(120); delay(500);
  AG2.write(90); delay(1000);
//Avancer patte B droit
BD2.write(130); delay(500);
  BD1.write(135); delay(500);
 BD2.write(100); delay(500);
//Avancer patte B gauche
 BG2.write(115); delay(500);
 BG1.write(20); delay(500);
 BG2.write(95); delay(500);
delay(2000);
//élever le robot
CD2.write(40); delay(10);
 CG2.write(25); delay(10);
 CD3.write(100); delay(10);
 CG3.write(80); delay(10);
 AD2.write(95); delay(10);
 AG2.write(100); delay(10);
 delay(1000);
//Avancer patte C gauche
  CG2.write(60);  delay(500);
  CG1.write(80); delay(500);
 CG2.write(40); delay(500);
//Avancer patte C droit
 CD2.write(60); delay(500);
  CD1.write(60); delay(500);
 CD2.write(40); delay(500);
 delay(1000);
//Boucle pour s'approcher
for(i=0; i<2; i++){
//Avancer le corps
 CD1.write(90);  delay(10);
 CG1.write(110); delay(10);
 BD1.write(165); delay(10);
 BG1.write(0); delay(10);
 AD1.write(60); delay(10);
 AG1.write(150); delay(10);
 delay(1000);
//Avancer patte A en haut
 AD2.write(105); delay(500);
  AD1.write(30); delay(500);
  AD2.write(85); delay(500);
 //Avancer patte A gauche en haut
  AG2.write(110); delay(500);
 AG1.write(120); delay(500);
  AG2.write(90); delay(1000);
//Avancer les pattes B
 BD2.write(130); delay(500);
  BD1.write(135); delay(500);
 BD2.write(100); delay(500);
 BG2.write(115); delay(500);
 BG1.write(20); delay(500);
 BG2.write(95); delay(500);
delay(2000);
//Avancer les pattes C
 CG2.write(60); delay(500);
  CG1.write(80); delay(500);
 CG2.write(40); delay(500);
 CD2.write(60); delay(500);
  CD1.write(60); delay(500);
 CD2.write(40); delay(500);
 delay(1000);
}
//Avancer le corps
```

```
CD1.write(90);  delay(10);
 CG1.write(110); delay(10);
 BD1.write(165); delay(10);
 BG1.write(0); delay(10);
 AD1.write(60); delay(10);
 AG1.write(150); delay(10);
 delay(1000);
//Avancer les pattes A
 AD2.write(105); delay(500);
  AD1.write(30); delay(500);
  AD2.write(85); delay(500);
 AG2.write(110); delay(500);
 AG1.write(120); delay(500);
  AG2.write(90); delay(1000);
//Monter les pattes B
 BD2.write(150); delay(500);
  BD1.write(115); delay(500);
 BD3.write(110); delay(100);
 BD2.write(130); delay(500);
 BG2.write(140); delay(500);
 BG1.write(50); delay(500);
 BG3.write(100); delay(100);
 BG2.write(115); delay(500);
delay(1000);
//Avancer les pattes C
 CG2.write(60); delay(500);
  CG1.write(80); delay(500);
 CG2.write(40); delay(500);
 CD2.write(60); delay(500);
  CD1.write(60); delay(500);
 CD2.write(40); delay(500);
 delay(1000);
```

```
//Boucle pour s'approcher
for(i=0; i<7; i++){
 //Avancer corps
CD1.write(90);  delay(10);
 CG1.write(110); delay(10);
  BD1.write(135); delay(10);
 BG1.write(25); delay(10);
 AD1.write(60); delay(10);
 AG1.write(150); delay(10);
 delay(1000);
//Avancer pattes A
  AD2.write(105); delay(500);
  AD1.write(30); delay(500);
  AD2.write(85); delay(500);
  AG2.write(110); delay(500);
 AG1.write(120); delay(500);
  AG2.write(90); delay(1000);
//Avancer pattes B
  BD2.write(150); delay(500);
  BD1.write(115); delay(500);
 BD3.write(110); delay(100);
 BD2.write(130); delay(500);
 BG2.write(140); delay(500);
 BG1.write(50); delay(500);
 BG3.write(100); delay(100);
 BG2.write(115); delay(500);
delay(1000);
//Avancer pattes C
CG2.write(60); delay(500);
  CG1.write(80); delay(500);
 CG2.write(40); delay(500);
 CD2.write(60); delay(500);
```

```
 CD1.write(60); delay(500);
 CD2.write(40); delay(500);
 delay(1000);
}
//Avancer corps
CD1.write(90); delay(10);
 CG1.write(110); delay(10);
  BD1.write(135); delay(10);
 BG1.write(25); delay(10);
 AD1.write(60); delay(10);
 AG1.write(150); delay(10);
 delay(1000);
//Avancer pattes A
 AD2.write(105); delay(500);
  AD1.write(30); delay(500);
  AD2.write(85); delay(500);
 AG2.write(110); delay(500);
 AG1.write(120); delay(500);
  AG2.write(90); delay(1000);
delay(1000);
//Monter pattes C
CG2.write(85); delay(500);
  CG1.write(80); delay(500);
 CG3.write(90); delay(10);
 CG2.write(55); delay(500);
 CD2.write(100); delay(500);
  CD1.write(60); delay(500);
 CD3.write(120); delay(10);
 CD2.write(70); delay(500);
 delay(1000);
}
}
```