Lycée Jean Perrin

Le Bras Robotisé

Réalisé par : Jules Sery - Jordan Vocisano –
Florian Bekaert – Cyril Gervais – Leo Abraham

PPE 2012

Introduction

De nos jours, l'utilisation des machines dans l'industrie ou la recherche est de plus en plus importante. En effet, l'automatisation des systèmes permet une conception plus rapide et plus sûr que certains ouvrage. Dans la recherche, on remarque de la même manière que l'utilisation de la robotique augmente, car celle-ci permet de manipuler avec beaucoup de précision et donc de manière plus sûr divers produits ou objets, neutres ou dangereux.

Les bras mécaniques peuvent aussi permettre à l'homme de ne pas répéter à longueur de journée un même mouvement. Imaginez-vous en train de visser des bouchons sur des bouteilles à toute une journée. Heureusement que des machines puissent faire se travail à notre place. De plus, les bras robotisés de remplacer l'homme pour exécuter des tâches nuisant à la santé de celui-ci, comme par exemple peindre des voitures car la peinture utilisé contient des produits dangereux pour l'homme surtout lorsqu'on les inhale.
La problématique est donc la suivante : comment réaliser notre bras et comment faire pour que celui-ci soit piloté à distance ?

I-Cahier des charges.

Conception d'un système permettant de piloter à distance une plateforme motorisée.

Analyse de besoin: Bêtes à cornes.

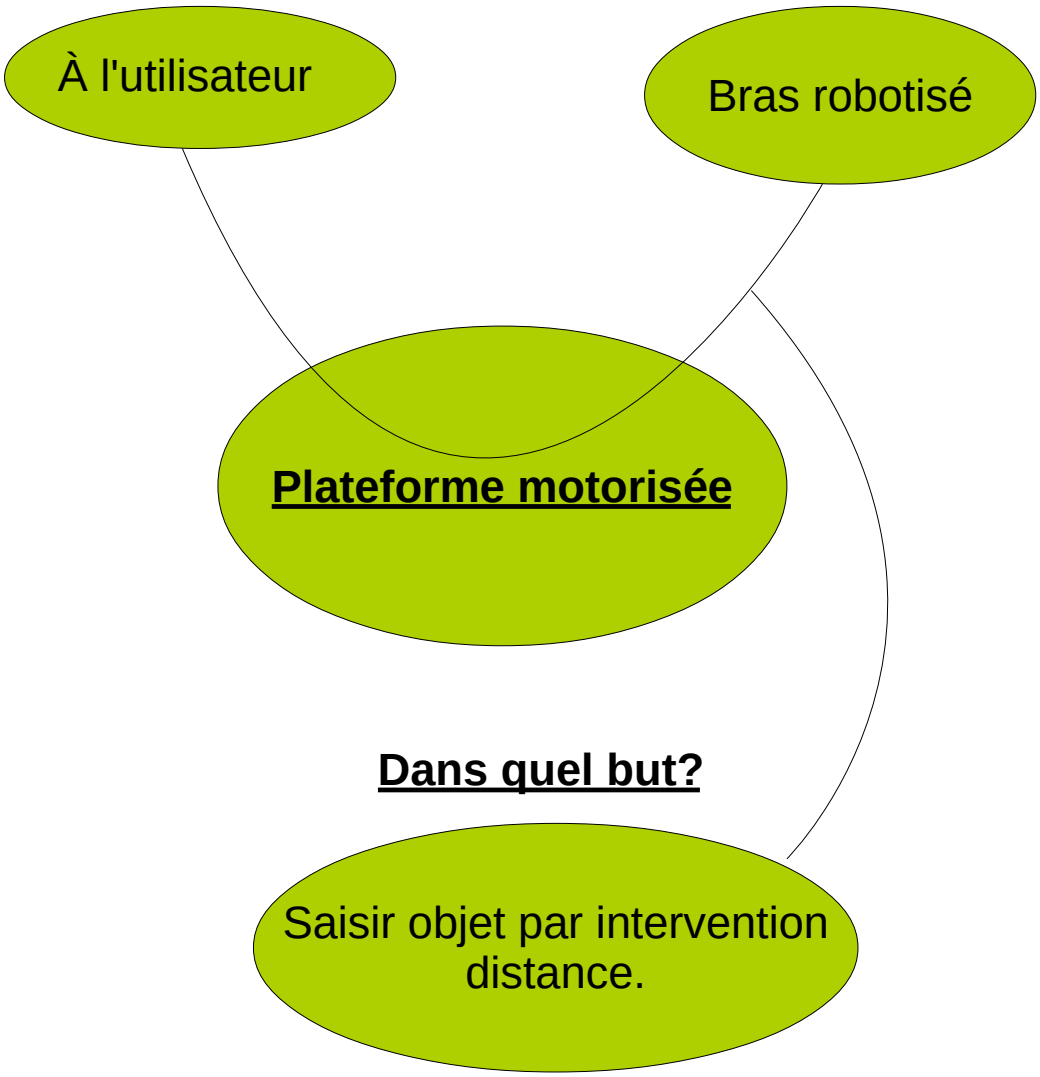
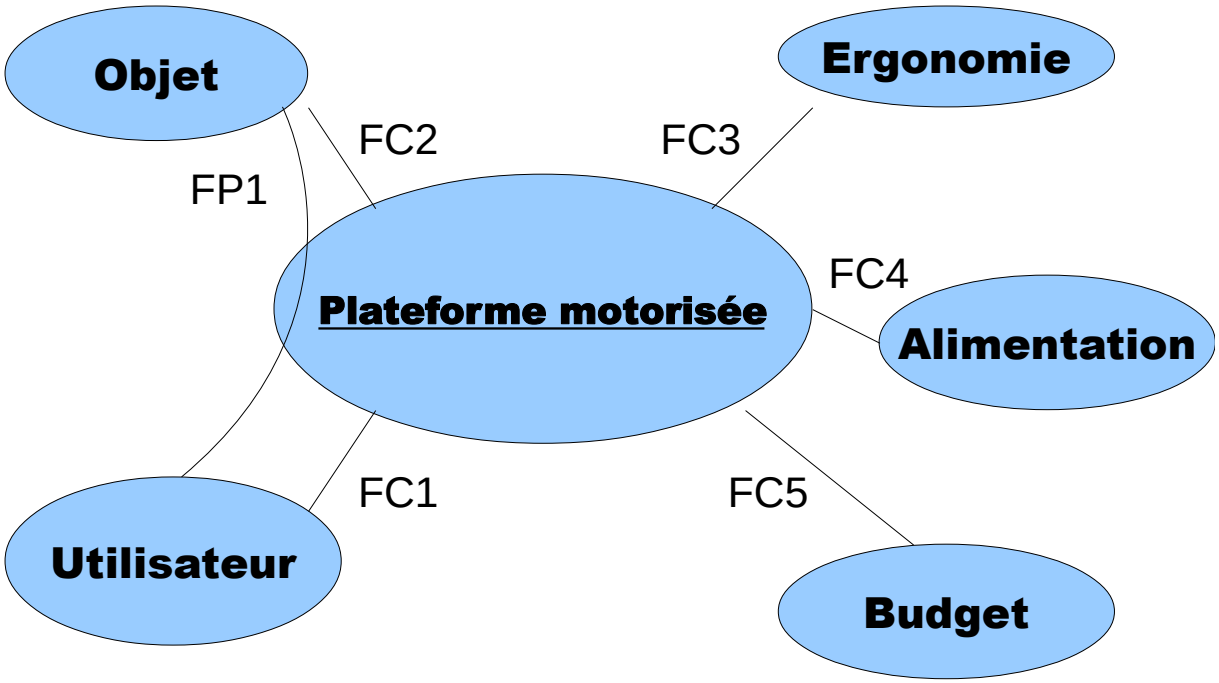


DIAGRAMME PIEUVRE:



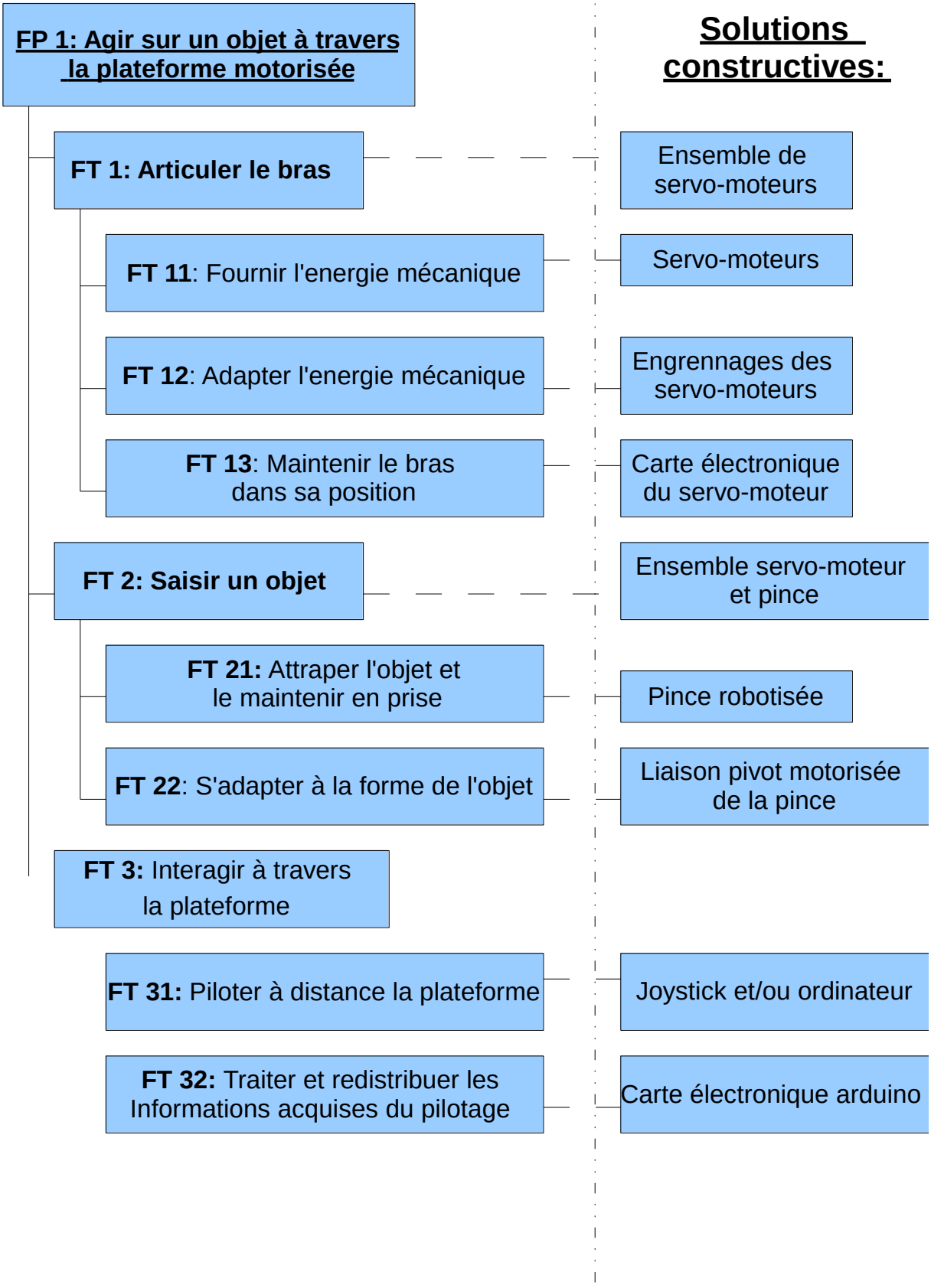
Fonction principale:

FP1: Interagir sur un objet à travers la plateforme motorisée

Fonctions complémentaires:

FC1: Pilotage à distance via ordinateur et/ou joystick.
FC2: Adapter la saisie robotisée en fonction des côtes de l'objet.
FC3: Ergonomie du système: Facilité d'utilisation.
FC4: Autonomie en énergie.
FC5: Le coût fabrication doit être compris entre 100 et 200 euros.

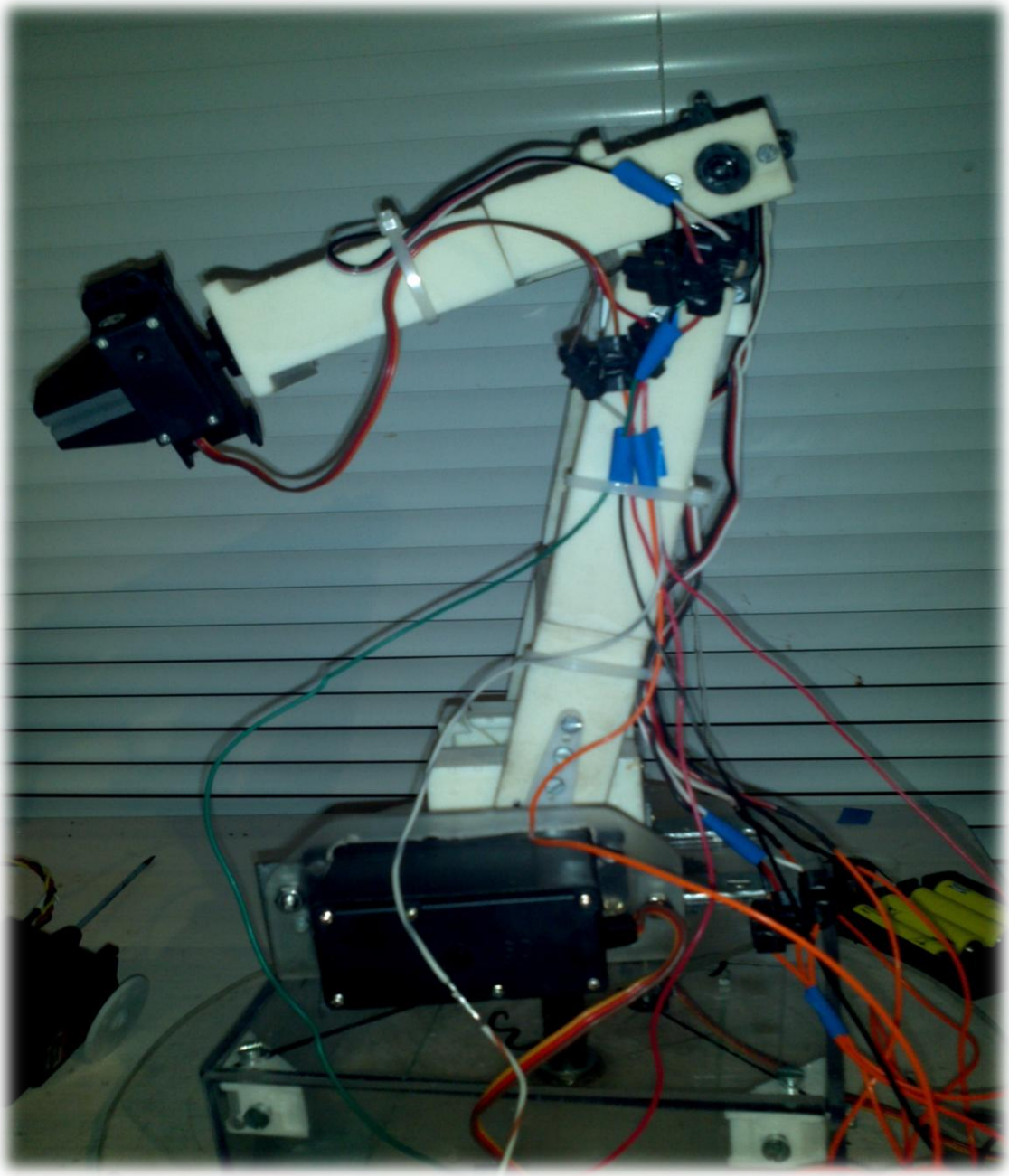
FAST partiel de la fonction principale FP1: « Agir sur un objet à travers la plateforme motorisée ».



II -Partie opérative

1) Structure du bras

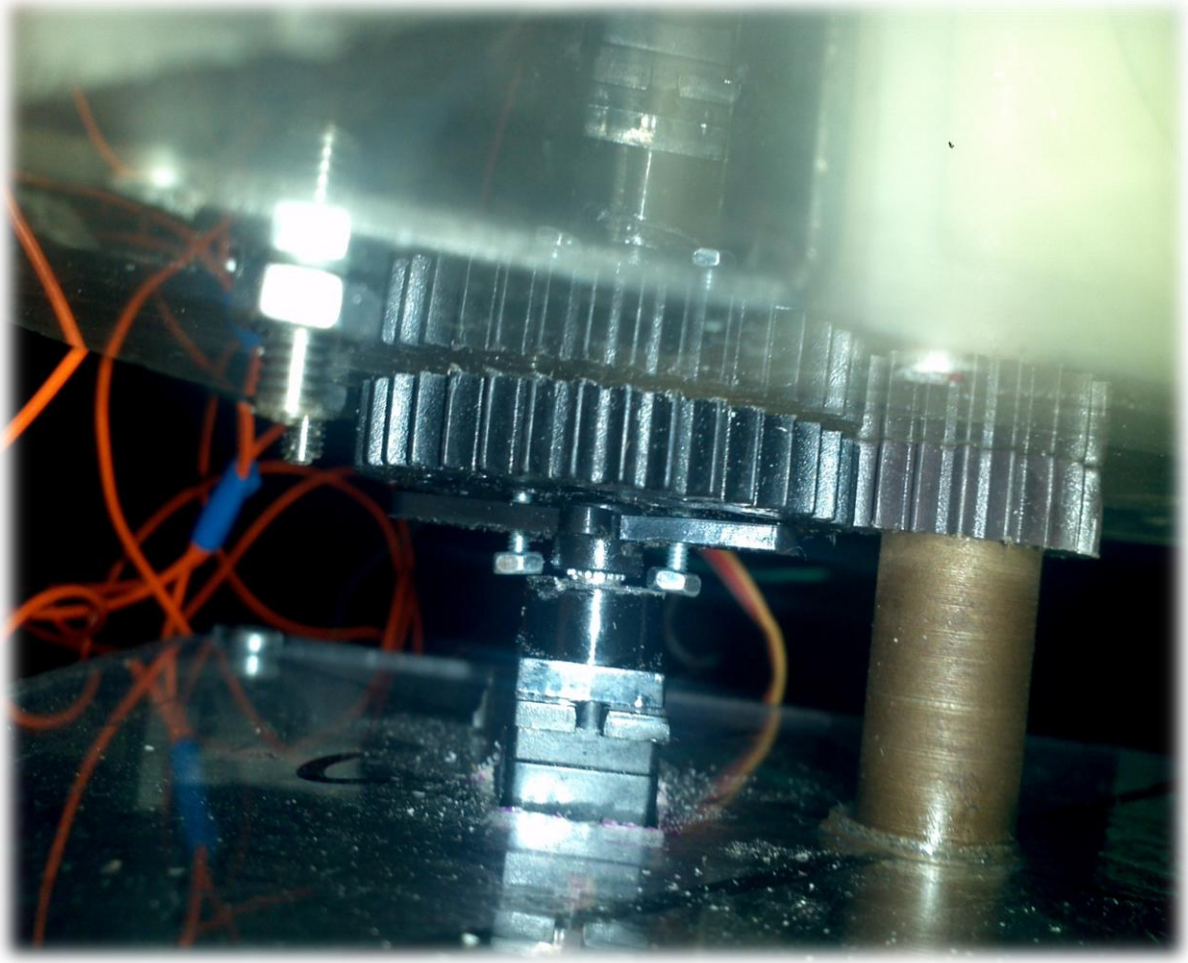
Les pièces du bras ont été faites ainsi car elles permettent de garder un bon équilibre du bras et de plus d'avoir un système léger et résistant. Ces pièces ont d'abord été imaginé sous le logiciel SolidWorks pour nous permettre d'avoir une vue d'ensemble du bras et ainsi pouvoir changer sa structure en fonction des contraintes rencontrées. Ces pièces ont ensuite été modélisées grâce à une imprimante 3D car les dimensions des pièces devaient être précises



2) Rotation verticale du bras

La première rotation, celle qui effectue le pivotement du bras pour saisir un objet n'importe tout autour de lui, est fait par un axe vertical. La rotation est engendrée par un cerveau moteur, à lui tout seul il ne peut que l'entrainer en rotation sur 180 degrés. Nous avons donc disposé une roue dentée sur le cerveau moteur et un pignon sur l'axe pour augmenter le rapport de réduction et avoir ainsi une rotation plus grande que 180 degré. Le rapport de réduction est de 2, ce qui donne une rotation du bras de 360 degré. (R = (Ze / Zs) = (40/20) = 2)

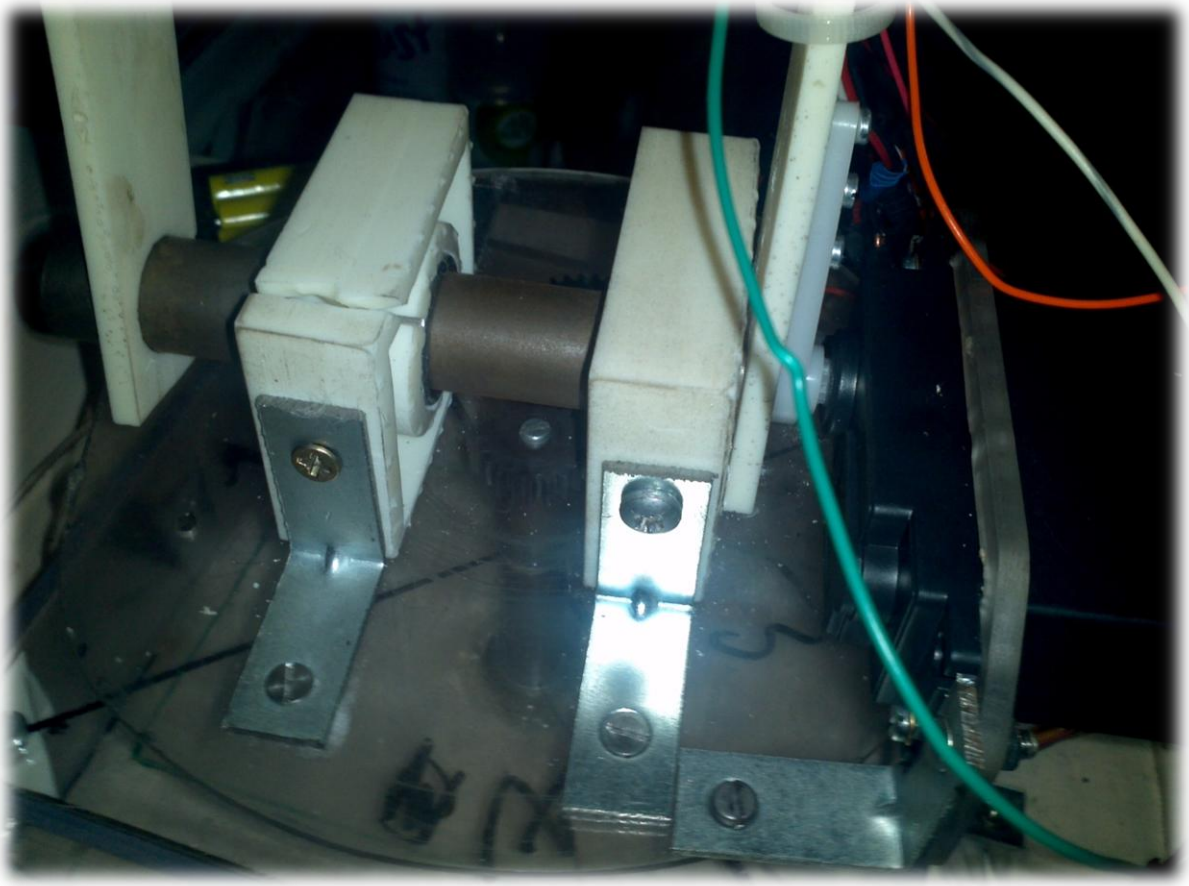
Le roulement à bille qui permet la rotation de l'axe supporte bien les effort s axiaux. Un deuxième roulement à bille aurait dû être mis en place pour une meilleur stabilité (les roulements à billes sur un axe sont toujours par deux), cependant pour cela nous aurions dû usiner l'axe, mais nous manquions de temps.



3) 1ère rotation horizontale

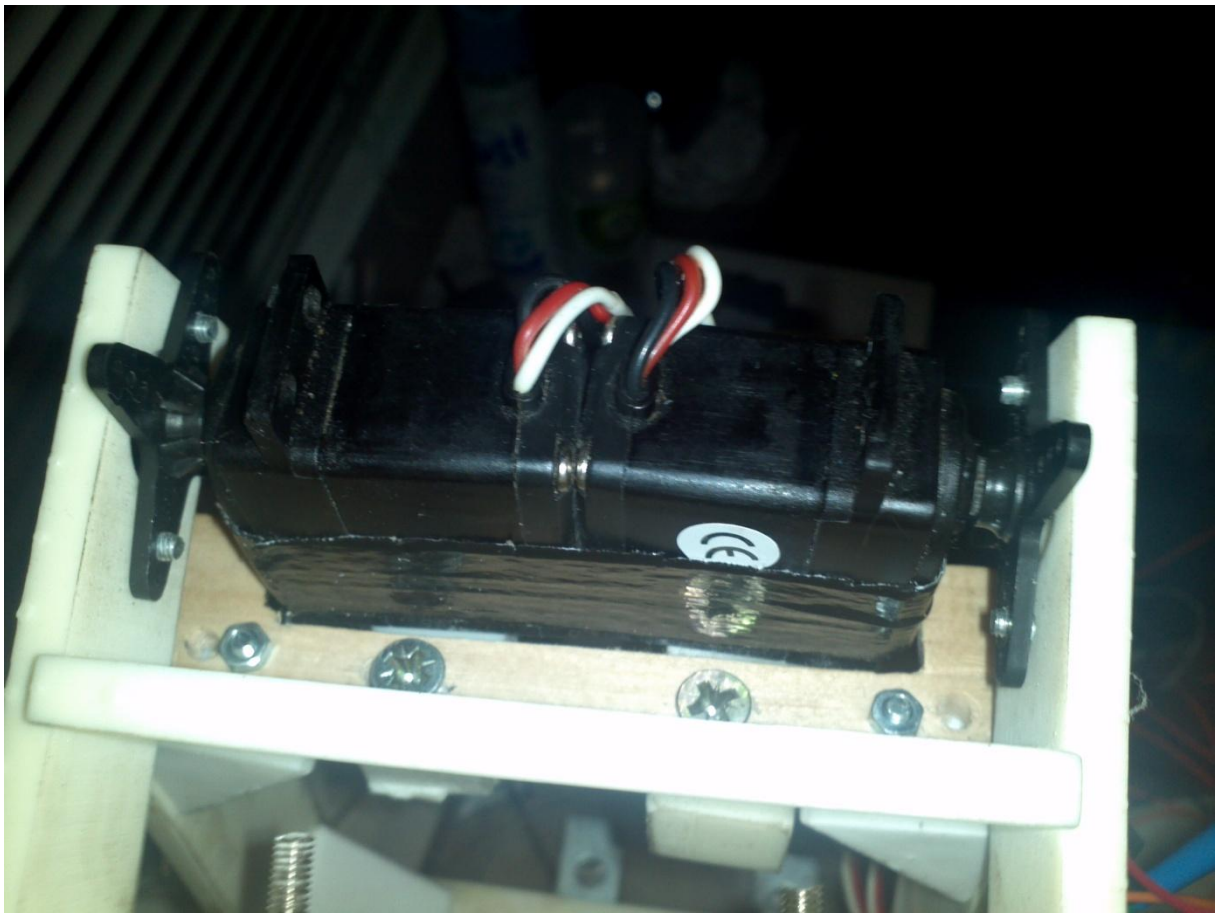
Pour la 2ème rotation de notre bras, nous avons rencontré une contrainte plutôt inattendue. Lors des tests des servos moteurs sur le bras, nous nous sommes aperçu que le servo moteur utilisé ne pouvait pas soulever le bras. Nous avons donc cherché le problème qui a été résolu assez rapidement. Nous avons d'abord pensé que le servo moteur manquait de puissance, ce n'était pas le cas, la cause du problème était au niveau du pignon du servo moteur. En effet, les dents qui sont situés sur le pignon et qui permettent la mise en rotation du palonnier (et donc du bras) étaient détériorés. Par conséquent, le servo tournait correctement mais n'entrainait pas le palonnier donc le bras dans sa rotation.

Le deuxième axe entraine en rotation le bras, l'axe est entraîné en rotation par un cerveau moteur et les 2 roulements à aiguilles permettent la rotation de l'axe. Nous avons choisis ces roulements car ils sont peu couteux, permettent une rotation sans frottement et surtout ne prennent que très peu de place et peuvent subir des efforts radiaux important. Etant donné que notre bras ne fait subir que des efforts radiaux nous avons opté pour ceux-ci plutôt que des roulements à bille « classiques ». L'axe donne aussi un très bon équilibre au bras. Le cerveau moteur a un couple de 25kg/cm. C'est-à-dire que le cerveau moteur est capable de soulever un objet de 25kg à un centimètre. En effectuant la conversion nous en déduisons que le servo moteur peut soulever un poids équivalent à 250 à 1mètre. Or, notre bras ne mesure que 42cm, nous effectuons donc le calcul suivant pour trouver quel est le poids maximum que notre bras pourra soulever => 0.25/0.42 = 0.595kg. Le bras peut donc soulever 595g sans compter le poids du bras.



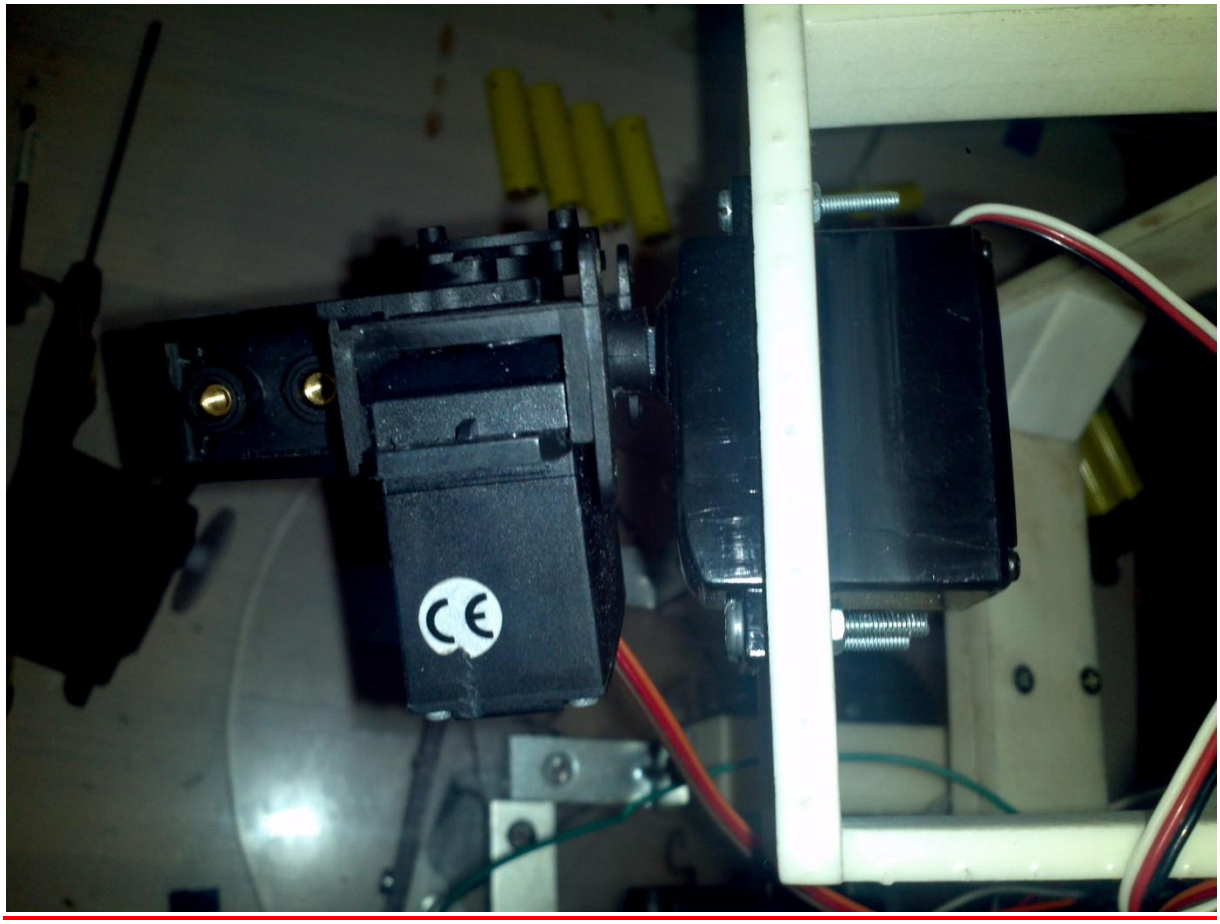
4) 2ème rotation horizontale

Pour la troisième rotation il n'y aucun axe, car mettre un axe nous posait un problème. Il aurait fallu prévoir un support pour le servo moteur ce qui aurait causé un problème pour l'équilibre de notre bras. De plus ajouter un axe aurait ajouté du poids à notre système, par conséquent nous aurions donc pu soulever un poids plus léger. A ceci s'ajoute un problème de budget. Pour résoudre ce problème, nous avons donc créé une pièce qui nous permet de positionner un servo moteur qui est directement vissé à la seconde partie de notre bras, celui-ci permet donc le mouvement de rotation qui se fait selon l'axe du pignon du servo moteur. Un servo moteur avec un couple de 3kg/cm et une longueur de bras de 0.21m permet donc de soulever d'après le calcul ci-contre : 0.03/0.21 = 0.143kg soit 143g. Or avec un servo moteur nous avons encore rencontré un problème d'équilibre, nous avons donc placé 2 servo moteurs de chaque côté du bras pour créer une rotation avec un bon équilibre. Etant donné que les servo moteurs sont identiques, la synchronisation est donc facile, le couple est alors doublé et donc le poids maximal possible à soulever se voit lui aussi doublé. Nous pouvons donc soulever un poids maximal de 2 x 0.143 = 0.286kg



5) Ensemble servo-moteur/pince

Un dernier problème a été rencontré en fin de projet, comment faire pour positionner notre pince sans que celle-ci provoque le déséquilibre de notre pince ? Nous pensions réaliser une pièce pour répondre à cette contrainte, mais la complexité de la solution nous a rebuté. On a alors prit plusieurs dizaines de minutes pour trouver une autre solution Nous nous sommes rendu compte que nous cherchions à faire trop compliqué et que la meilleure solution était d'encastré notre système servo-moteur/pince sur une plaque, celui-ci est alors soutenu par des vis et des écrous.



III-Partie commande

Carte Arduino-Servos-Joysticks

1- Le rôle des joysticks

- Les joysticks sont composés de deux résistances variables linéaires insérées sur deux axes différents qui vont faire varier la tension de 5V délivrée en sortie par la carte (voir image: sortie 5V). Cette tension va alors varier entre 0 et 5V, et cette variation sera la première donnée analogique nécessaire pour positionner le servomoteur à l'angle voulu. La résistance variable est composée de 3 broches (voir image) : une entrée pour le courant, une masse (elle n'est pas polarisée) et une sortie pour la tension modifiée appelée « analog ».
- En effet cette tension (située entre 0 et 5 V) va être appliquée sur une entrée analogique de la carte dite « ANALOG IN » (ex: A0, A1, A2, …) et sera pris en charge comme variable analogique par le microcontrôleur.
- Un seul joystick peut alors émettre deux variables analogiques qui à leurs tours peuvent contrôler deux servos en position.
- Ces résistances variables doivent impérativement être linéaires pour faciliter le contrôle des servos par l'opérateur. On aura alors besoin de deux joysticks et d'une résistance variable indépendante en raison de l'utilisation de 6 servomoteurs (Une seule résistance contrôle deux servomoteurs

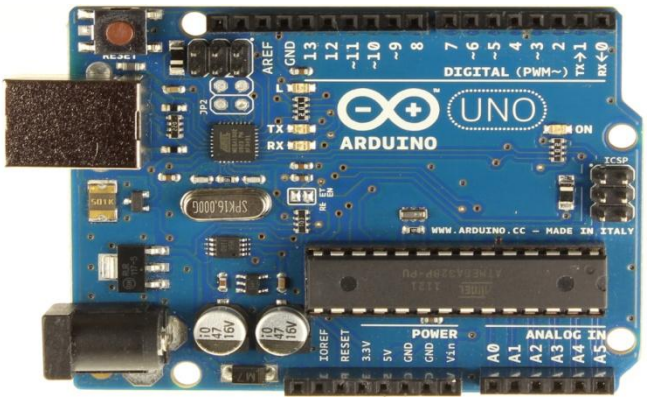
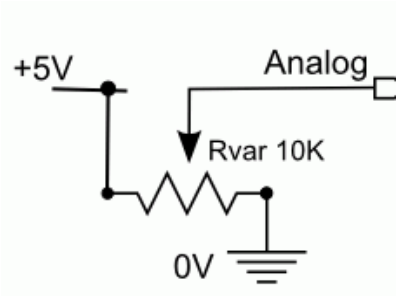


Schéma caracréristique d'une résistance variable :



2-Les servos-moteurs

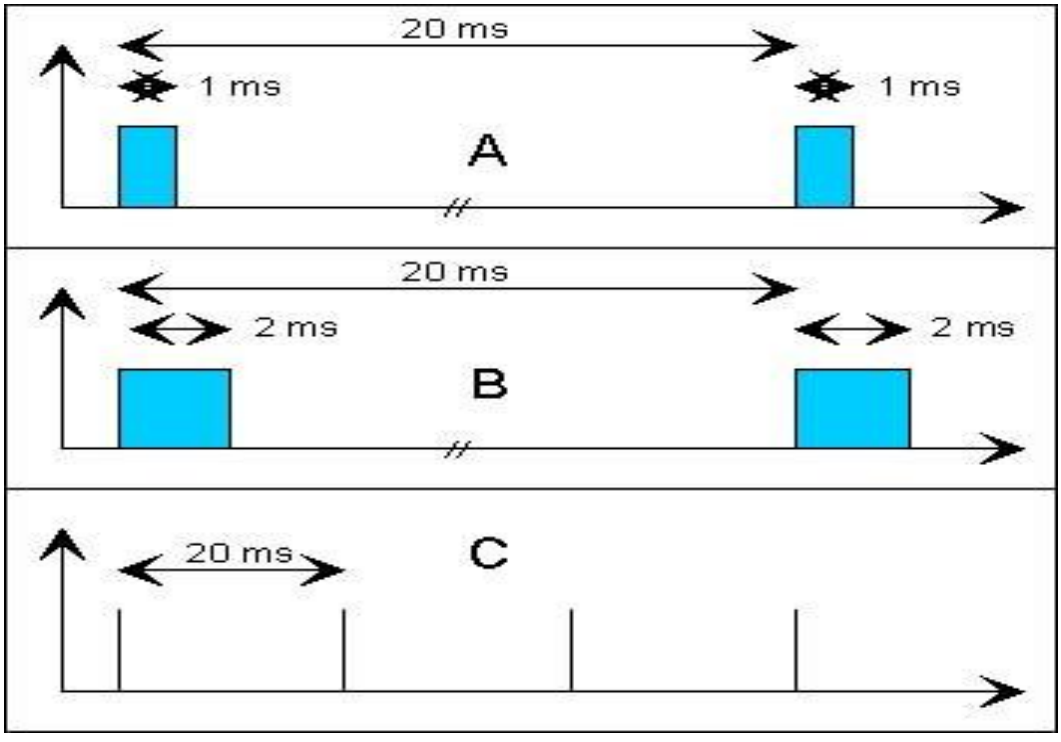
- Les servomoteurs sont des moteurs capables de maintenir une position à un effort statique et dont la position est vérifié en continue (grâce au potentiomètre) et corrigé en fonction de la mesure (ici la mesure fait référence au signal entrant dans le servo qui dépend de la tension en sortie des joysticks). C'est un système motorisé capable d'atteindre des positions prédéterminées puis de les maintenir. Le servomoteur intègre un système électronique qui convertie un signal numérique en un angle qui sera reproduit sur le palonnier (voir l'image) grâce au moteur électrique à courant continu présent dans le servomoteur.

- Ce signal numérique est une dérivée de la technique PWM ou MLI (Modulation en Largeurs d' Impulsions). Le servo est alimenté avec 3 fils: une entrée 5V (ou plus), une masse et une entrée d'impulsion (la commande du servo). C'est dans cette entrée d'impulsion qu'est envoyé le signal numérique modulé en impulsions.

- Ces impulsions sont des créneaux à rapport cyclique variable (5 à 10% pour le bon fonctionnement du servomoteur), et la période de ce créneau est fixée à 20 ms:

    - Ce signal numérique va alors contrôler le servomoteur en position. Et il est le fruit de la conversion du signal analogique en sortie de l'Analog de la résistance variable du joystick. En conclusion c'est le contrôle du servo via le joystick.

    - Le rapport cyclique de 5% donne un état haut de 1ms ce qui correspond à un angle de 0° (A sur le graph), le rapport de 10% donne un état haut de 2ms ce qui correspond à un angle de 180° (B sur le graph). La valeur de l'angle est linéaire (1,5 ms pour un angle de 90°). Les impulsions doivent être constante (C sur le graph) sinon le servomoteur devient instable (période fixe).



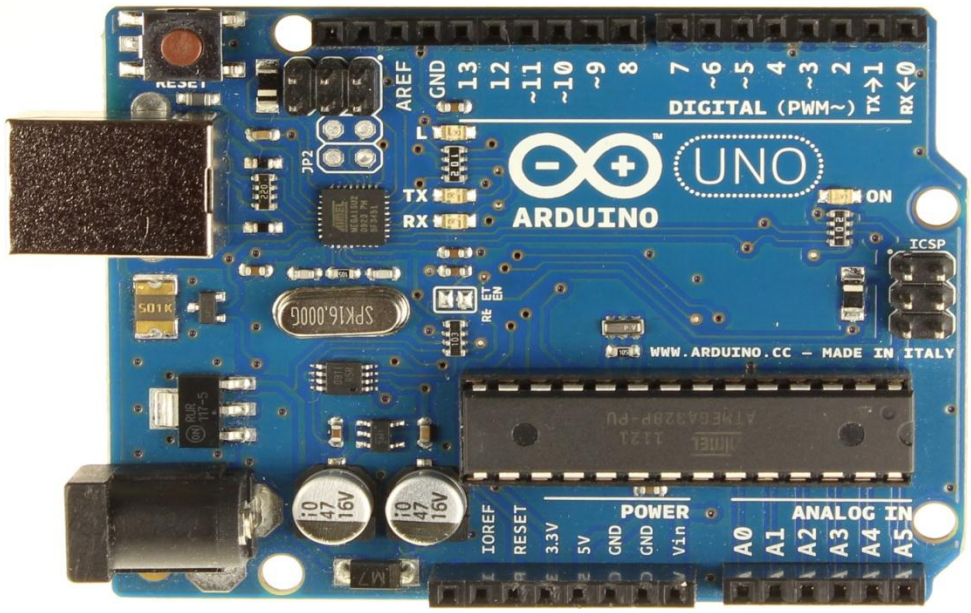
3-La carte Arduino le cœur de la partie commande

- La carte Arduino gère la fonction principale de cette partie commande appelée « CAN » (« Convertisseur Analogique Numérique » sur 10 bits dans ce projet). Cette fonction a pour but de convertir les signaux analogiques en entrée (issus des joysticks) en signaux numériques. C'est cette carte qui permet de faire la liaison entre joysticks et servomoteurs.

- Cette carte a pour cœur un microcontrôleur, ici un Atmega 328, qui lui intègre le programme qui gère la fonction CAN. Mais la fonction CAN n'est pas spécifique au microcontrôleur puisqu' elle est issue de la programmation que nous pouvons téléchargée dans ce dernier. En plus de gérer cette fonction il envoie et reçoit des signaux numériques ou analogiques dans plusieurs entrées et sorties (« ANALOG IN » et « DIGITAL », cf. photo carte). En conclusion cette carte électronique peut gérer beaucoup de fonctions (contrôle de LEDs, Digits, enceintes, écran, moteur, réseaux, …).

- Pour ce projet nous avons créé un programme spécifique que nous avons téléchargé dans le microcontrôleur pour que la carte puisse positionner les servomoteurs à l'aide des joysticks. Ce programme intègre la conversion analogique numérique ainsi que les chemins à suivre entre entrées analogiques (joysticks) et sorties numériques (servos).

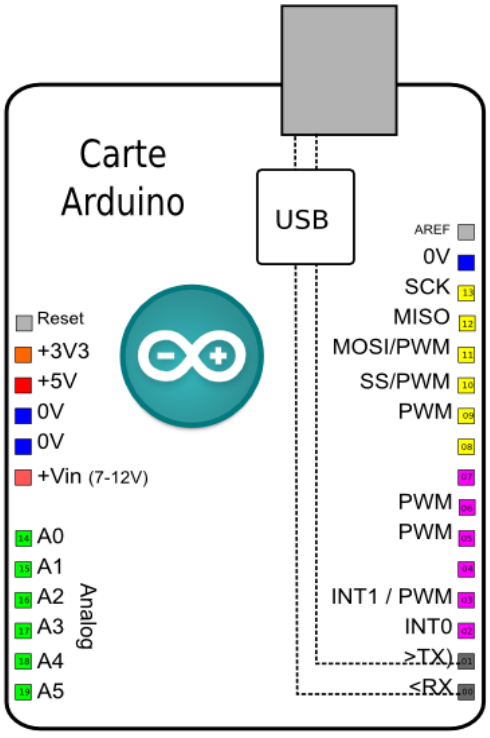
- Cette carte dispose :
    - de 14 broches numériques d'entrées/sorties (dont 6 peuvent être utilisées en sorties PWM (largeur d'impulsion modulée)),
    - de 6 entrées analogiques (qui peuvent également être utilisées en broches entrées/sorties numériques),
    - d'un quartz 16Mhz,
    - d'une connexion USB,
    - d'un connecteur d'alimentation jack,
    - d'un connecteur ICSP (programmation "in-circuit"),
    - et d'un bouton de réinitialisation (reset).

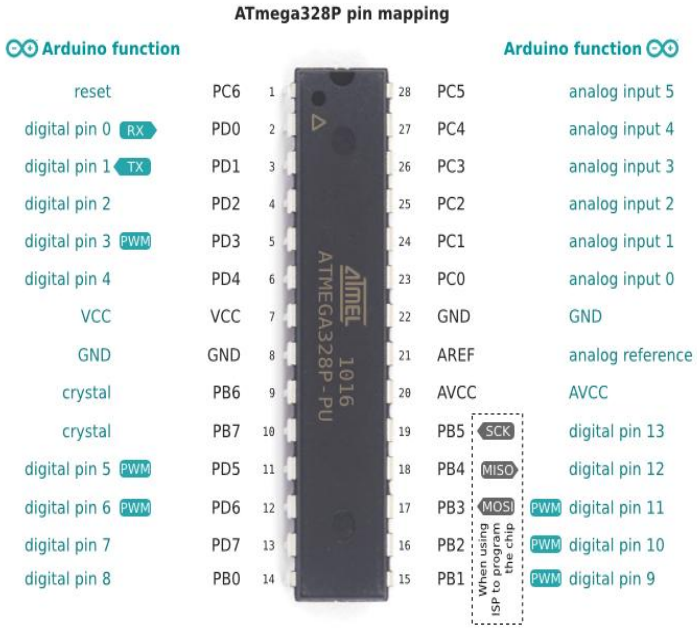


- **Caractéristiques globales :**

Microcontrôleur	ATmega328
Tension de fonctionnement	5V
Tension d'alimentation (recommandée)	7-12V
Tension d'alimentation (limites)	6-20V
Broches E/S numériques	14 (dont 6 disposent d'une sortie PWM)
Broches d'entrées analogiques	6 (utilisables en broches E/S numériques)
Intensité maxi disponible par broche E/S (5V)	40 mA (ATTENTION : 200mA cumulé pour l'ensemble des broches E/S)
Intensité maxi disponible pour la sortie 3.3V	50 mA
Intensité maxi disponible pour la sortie 5V	Fonction de l'alimentation utilisée - 500 mA max si port USB utilisé seul
Mémoire Programme Flash	32 KB (ATmega328) dont 0.5 KB sont utilisés par le bootloader (programme de base préprogrammé conçu pour établir la communication entre l'Atmega et le logiciel Arduino)
Mémoire SRAM (mémoire volatile)	2 KB (ATmega328)
Mémoire EEPROM (mémoire non volatile)	1 KB (ATmega328)
Vitesse d'horloge	16 MHz

- **Brochage de la carte Arduino et « mapping » du microcontrôleur :**

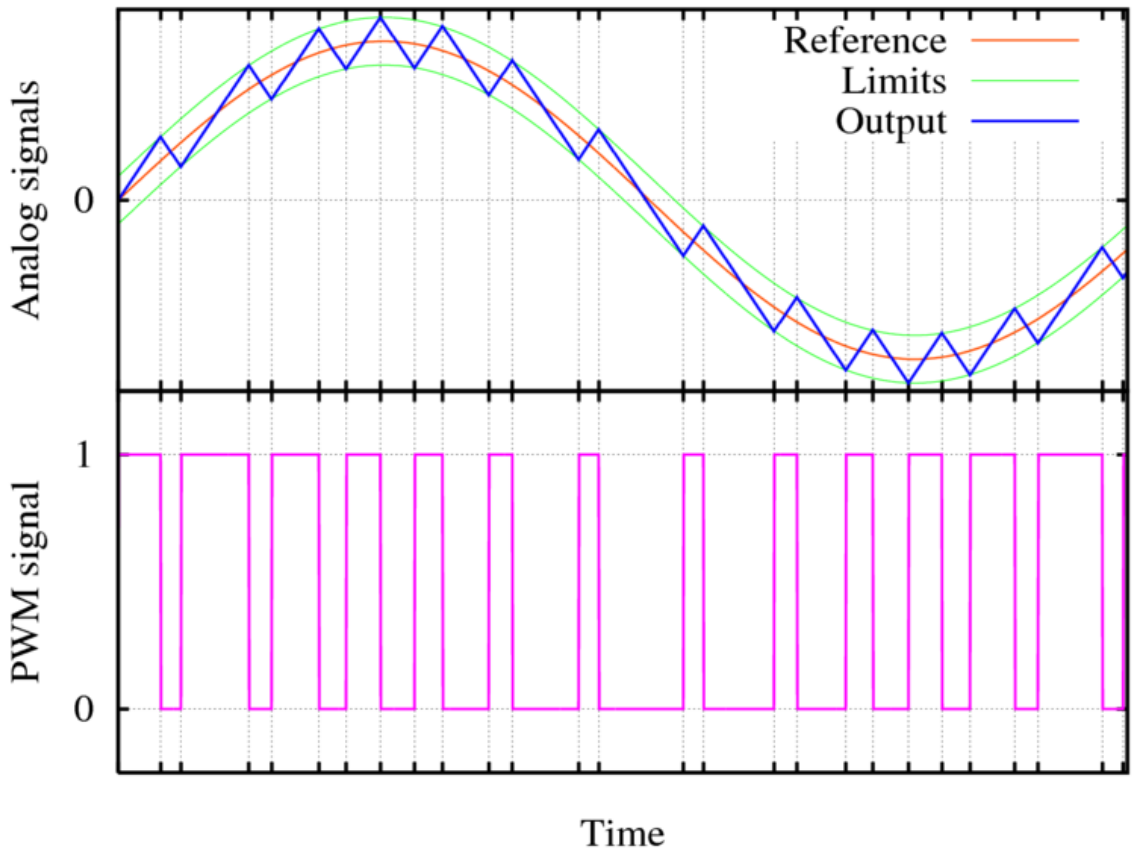




A-La fonction « CAN »

- Le microcontrôleur une fois programmé se sert de la conversion analogique-numérique pour faire marcher le bras. Il créé à partir de la valeur analogique du joystick une valeur numérique codé sur 10 bits (de 0 à 1023) proportionnelle à la valeur analogique d'entrée mais non-exploitable par les servos. C'est une autre fonctionnalité programmée qui va étalonner le résultat de cette conversion sur une autre échelle exprimé en ms(en réalité cette échelle représente la durée de l'état haut du signal). C'est-à-dire que l'échelle 0 à 1023 va être transformée en l'échelle 1ms à 2ms : Le résultat de la conversion va alors être reporté proportionnellement entre 1ms et 2 ms (durée de l'état haut du signal entrant dans la commande su servomoteur).

- Les différents niveaux analogiques sont convertis en une seule valeur de référence (en rouge). Ensuite cette valeur de référence est traduite en valeur numériques exploitables par les servomoteurs. C'est-à-dire que pour 0 V en sortie du joystick on obtient un rapport cyclique de 5% (donc un état logique haut d'une durée de 1 ms>>0°) en sortie numérique de la carte et pour 5V en sortie du joystick on obtient un rapport cyclique de 10% (état logique haut de 2 ms>>180°). De même cette conversion est linéaire donc pour 2,5 V on obtient un rapport cyclique de 7,5%(état logique haut de 1,5ms>>90°).Par conséquent la valeur analogique module le signal numérique en sortie de carte. En conclusion la conversion analogique-numérique est un l'élément fondamental dans le contrôle des servos grâce aux joysticks.

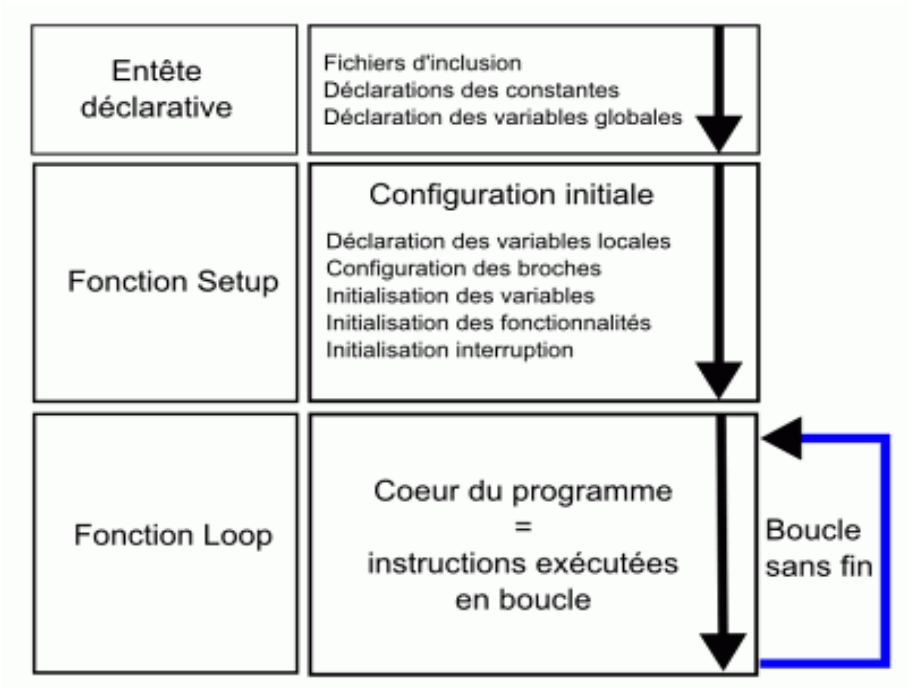


B- La programmation.

- La programmation est un code qui va dicter des ordres au microcontrôleur, on la télécharge dans ce dernier pour qu'il puisse suivre les consignes dictées par le programme. Pour ce faire on utilise un logiciel nommé Arduino qui peut vérifier et compiler plusieurs programmes, l'avantage est de pouvoir vérifier le programme édité avant de le compiler vers le microcontrôleur.

- Ce programme est codé en langage C, c'est un langage de programmation impératif pour la carte. Son avantage est qu'il intègre des fonctions préinstallées dans une seule ligne de code grâce à des librairies. Dans ce programme nous n'utilisons que la librairie SERVO qui regroupe un grand nombre de sous fonctions en une seule ligne de code.

- Dans cette programmation la conversion analogique numérique sur 10 bits est représentée par l'instruction suivante:
    - **Ex:** mesure_brute=analogRead (Voie_2);
    - **Généralement:** value=analogRead(pin)
      Ce code permet de lire la valeur sur l'entrée analogique de la carte et de la convertir en numérique sur 10 bits (de 0 à 1023).
- Après l'on doit convertir le résultat de la conversion compris entre 0 et 1023 en un angle entre 0 et 180° c'est-à-dire créer un signal numérique avec un rapport cyclique compris entre 5% et 10%:
    - **Généralement:** angle=map(value, fromLow, fromHigh, toLow, toHigh)
    - **Ex:** angle_servo_2=map(mesure_brute,0,1023,0,180);
- Après avoir créé le signal il faut dire au microcontrôleur où l'envoyer grâce à l'instruction suivante :
    - **Ex:** bras2.write(angle_servo_2);
    - **Généralement:** servo.write(angle)



C- Structure de la programmation.

Au niveau de la partie déclarative :

Inclusion des librairies utilisées

- On inclut les librairies des fonctionnalités utilisées :
  - Inclusion de la librairie pour les servomoteurs :

```
#include <Servo.h> // librairie pour servomoteur
```

Déclaration de constantes utiles

- On déclare les constantes utiles dans le programme :
  - Déclaration des constantes utiles pour les servomoteurs :

```
const int POS_MIN=550; // largeur impulsion pour position 0° servomoteur
                       // POS_MIN=550 pour futaba S3003
const int POS_MAX=2330; // largeur impulsion pour position 180° servomoteur
                                                    // POS_MAX=2330 pour futaba s3003
```

Déclaration des constantes de broches

- Déclaration des constantes pour les broches utilisées dans le programme :

```
const int SERVO_0=2; //declaration constante de broche numérique 2
const int SERVO_1=3; //declaration constante de broche "
const int SERVO_2=4; //declaration constante de broche "
const int SERVO_3=5; //        "
const int SERVO_4=6;//         "
const int SERVO_5=7;


const int Voie_0=0; //declaration constante de broche analogique 0
const int Voie_1=1; //declaration constante de broche analogique 1
const int Voie_2=2; //declaration constante de broche analogique 2
const int Voie_3=3; //declaration constante de broche analogique 3
const int Voie_4=4; //
```

Déclaration des variables globales

- Déclaration des variables globales du programme :
  - Déclaration des variables globales utilisées pour la conversion analogique-numérique

```
int mesure_brute=0;// Variable pour acquisition résultat brut de conversion analogique numérique

float mesuref=0.0;// Variable pour calcul résultat décimal de conversion analogique numérique
```

  - Déclaration des variables globales utilisées pour le positionnement des servomoteurs

```
int angle_servo_0=0; // variable pour angle du servomoteur

int angle_servo_1=0; // variable pour angle du servomoteur

int angle_servo_2=0; // variable pour angle du servomoteur

int angle_servo_3=0;

int angle_servo_4=0;
```

Déclarations des objets utiles pour les fonctionnalités utilisées

- Déclaration des objets utiles pour les fonctionnalités utilisées :
  - Déclaration de 3 objets servomoteurs

```
Servo rotation;  // crée un objet servo pour contrôler le servomoteur

Servo bras;  // crée un objet servo pour contrôler le servomoteur

Servo bras2;  // crée un objet servo pour contrôler le servomoteur

Servo bras_2;

Servo rotation_pince;

Servo pince;
```

<u>Au niveau de la fonction d'initialisation setup( ) :</u>

Initialisation des fonctionnalités utilisées :

- On initialise les différentes fonctionnalités utilisées :
  - Initialisation du servomoteur

```
rotation.attach(SERVO_0, POS_MIN, POS_MAX);  // attache l'objet servo à la broche de commande du servomoteur

bras.attach(SERVO_1, POS_MIN, POS_MAX);  // attache l'objet servo à la broche de commande du servomoteur

bras2.attach(SERVO_2,POS_MIN,POS_MAX);

rotation_pince.attach(SERVO_3,POS_MIN,POS_MAX);  // attache l'objet servo à la broche de commande du servomoteur

pince.attach(SERVO_4,POS_MIN,POS_MAX);

bras_2.attach(SERVO_5,POS_MIN,POS_MAX);
```

Configuration des broches utilisées :

- Configuration des broches en sortie :

```
pinMode(SERVO_0, OUTPUT); //met la broche en sortie

pinMode(SERVO_1, OUTPUT); //met la broche en sortie

pinMode(SERVO_2, OUTPUT); //met la broche en sortie
```

<u>Au niveau de la boucle principale, la fonction loop ( ) :</u>

- Pour chaque servomoteur :
  - on commence par lire la valeur de la tension sur la broche analogique de commande à l'aide de l'instruction analogRead()
  - on transpose la valeur en degrés à l'aide de l'instruction map()
  - on positionne le servomoteur à l'aide de l'instruction write()
- Exemple pour 2 servomoteurs :

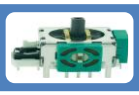
```
void loop(){ // debut de la fonction loop()

// --- ici instructions à exécuter par le programme principal ---

// rotation

mesure_brute=analogRead(Voie_0);// acquisition conversion analogique numérique  sur broche analogique indiquée - résultat 10bits==>(0-1023)

angle_servo_0=map(mesure_brute,0,1023,0,180); // convertit la valeur mesurée comprise entre 0 et 1023 en une impulsion entre 300µs et 2350µs càd entre 0 et 180°

// map(value, fromLow, fromHigh, toLow, toHigh); // permet changement d'échelle simplifié

rotation.write(angle_servo_0); // positionne le servo a l'angle voulu

delay(30);

// bras

mesure_brute=analogRead(Voie_1); // acquisition conversion analogique numérique (100µs env.) sur broche analogique indiquée - résultat 10bits (0-1023)

angle_servo_1=map(mesure_brute,0,1023,0,180); // convertit la valeur mesurée comprise entre 0 et 1023 en un angle entre 0 et 180

// map(value, fromLow, fromHigh, toLow, toHigh); // permet changement d'échelle simplifié

bras.write(angle_servo_1); // positionne le servo à l'angle voulu



delay(30); // impose un delai de 30 ms entre chaque acquisitions
```

3- Les détails du montage : les branchements .

Résumons le matériel nécessaire a la programmation, et a aux controles du bras robotisé :

1x Arduino

6x Servomoteurs



5x Potentiomètres

2x Alimentations

A- Arduino Uno

1 : Gestion des potentiomètres

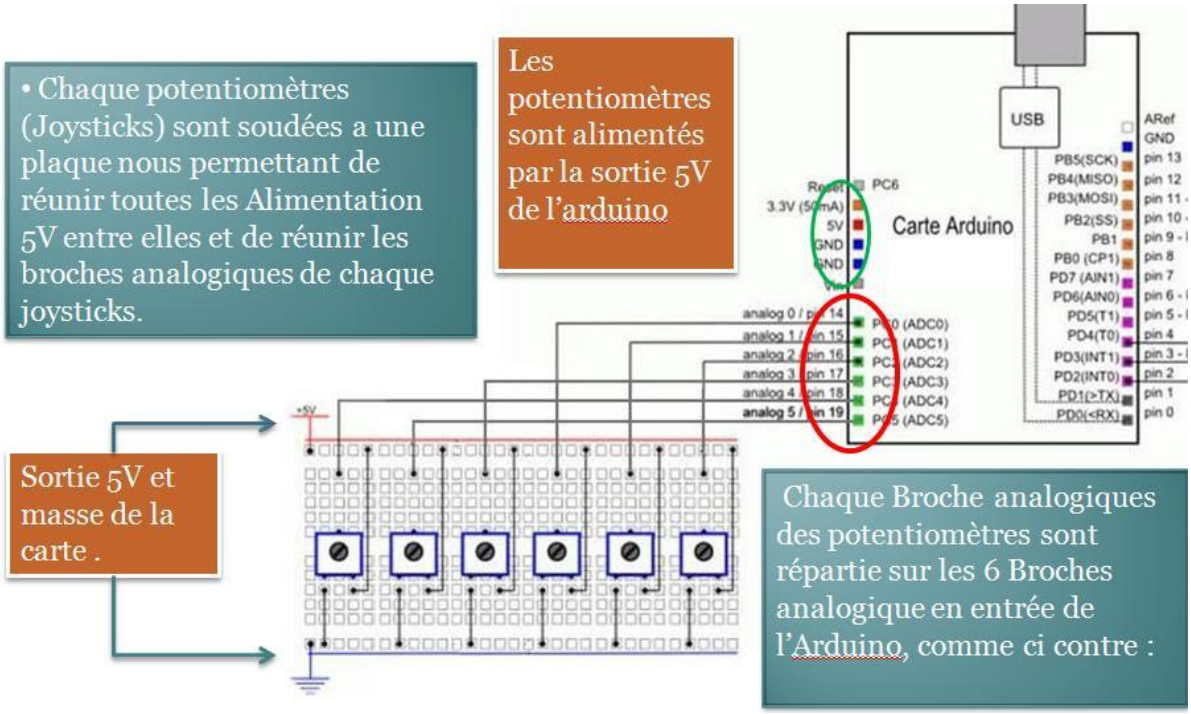
- La carte Uno dispose de 6 entrées analogiques (numérotées de 0 à 5), chacune pouvant fournir une mesure d'une résolution de 10 bits (càd sur 1024 niveaux soit de 0 à 1023) à l'aide de la très utile fonction analogRead() du langage C. Par défaut, ces broches sont traduites en numérique entre 0V (valeur 0) et 5V (valeur 1023) : tension de référence par défaut, mais il est possible de modifier la tension de référence en appliquant une tension compris entre 0 et 5V sur la broche AREF et en utilisant l'instruction analogReference() du langage C.

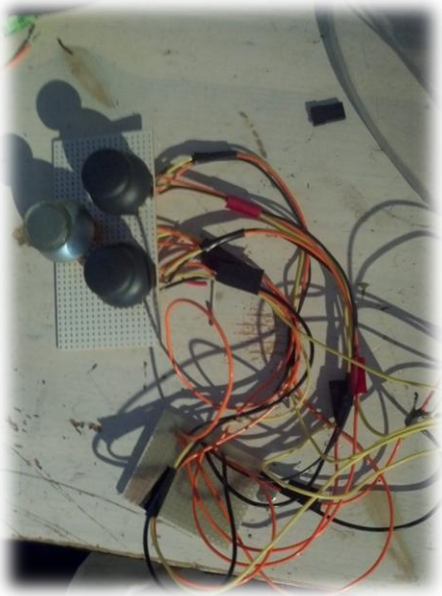
- Note : les broches analogiques peuvent être utilisées en tant que broches numériques : elles sont numérotées en tant que broches numériques de 14 à 19.

Les entrées de A0 à A4 seront utilisées et relier au 5 potentiomètres :

Ainsi, la totalité des signaux des Potentiomètres pourront être gérés avec une unique carte arduino

Afin d'assurer le contrôle des servomoteurs ; nous avons opter pour 5 potentiomètres (plus précisement de 3 joysticks). Ces derniers sont soudées a une plaque sur laquelle est reliée la sortie d'alimentations 5V et la masse.





Chaque fils de sortie analogiques des potentiomètres sont reliés à l'arduino par des broches, nous permettant d'assurer l'absence de faux contacts

2 : Gestion de l’alimentation externe de l’arduino

- La carte Arduino Uno peut-être alimentée soit via la connexion USB (qui fournit 5V jusqu'à 500mA) ou à l'aide d'une alimentation externe. La source d'alimentation est sélectionnée automatiquement par la carte. L'alimentation externe (non-USB) peut être soit un adaptateur secteur (pouvant fournir typiquement de 3V à 12V sous 500mA) ou des piles (ou des accus) La carte peut fonctionner avec une alimentation externe de 6 à 20 volts. Cependant, si la carte est alimentée avec moins de 7V, la broche 5V pourrait fournir moins de 5V et la carte pourrait être instable. Si on utilise plus de 12V, le régulateur de tension de la carte pourrait chauffer et endommager la carte.
  Aussi, la plage idéale recommandée pour alimenter la carte Uno est entre 7V et 12V.

Nous avons utilisé un bloc de 6 piles AA de 5V afin d’assurer l’autonomie énergétique de l’arduino et de réduire les couts de construction.

En utilisant une alimentation extérieure autre que celle de l’arduino, nous devons impérativement relier la masse de l’arduino, a celle de l’alimentation extérieur.

Pour cela, nous avons récupérer un boitier d’alimentation d’une voiture électrique, puis récupérer une prise 2.1mm positif pour la brancher dans le connecteur jack de la carte. Quelques soudures ont été nécessaires.

3 : Gestion des servomoteurs

- Chacune des 14 broches numériques de la carte UNO (numérotées des 0 à 13) peut être utilisée soit comme une entrée numérique, soit comme une sortie numérique. Ces broches fonctionnent en 5V. Chaque broche peut fournir ou recevoir un maximum de 40mA d'intensité

Nous allons donc utiliser les 6 broches numériques suivantes : Broches 2, 3, 4, 5, 6,

Toujours afin d'assurer le bon maintien des fils dans les pins, nous avons souder les câbles a des broches.

- 4 : Gestion de l'alimentation des servomoteur

Dans le cas d'une carte Arduino :
- l'intensité maximale disponible sur une broche est de 40mA
- l'intensité maximale cumulée pour l'ensemble des broches est 200mA
- l'instensité maximale que peut fournir l'alimentation 5V de la carte est 500mA.

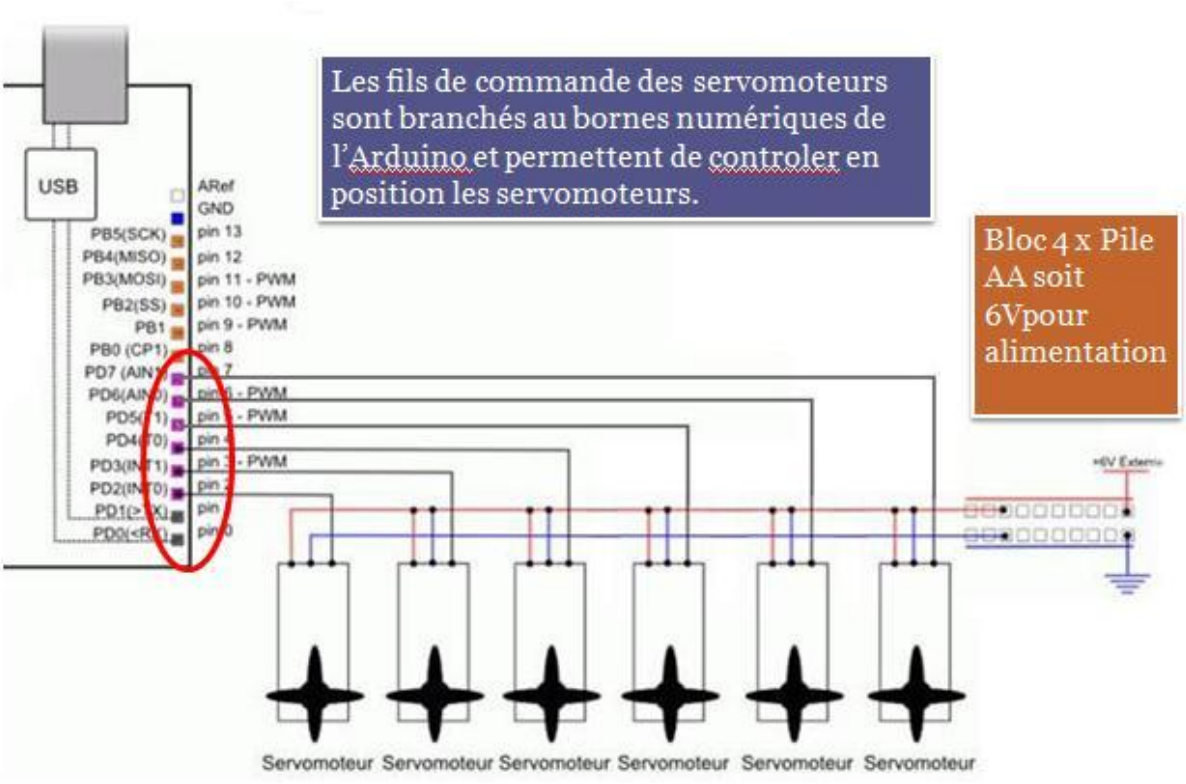
Le point essentiel ici est l'alimentation des servomoteurs qui doit être externe à la carte Arduino

En moyenne un servo nécessite 120 mA en alimentation → 500/120= 4 servos possibles branchés à la carte.

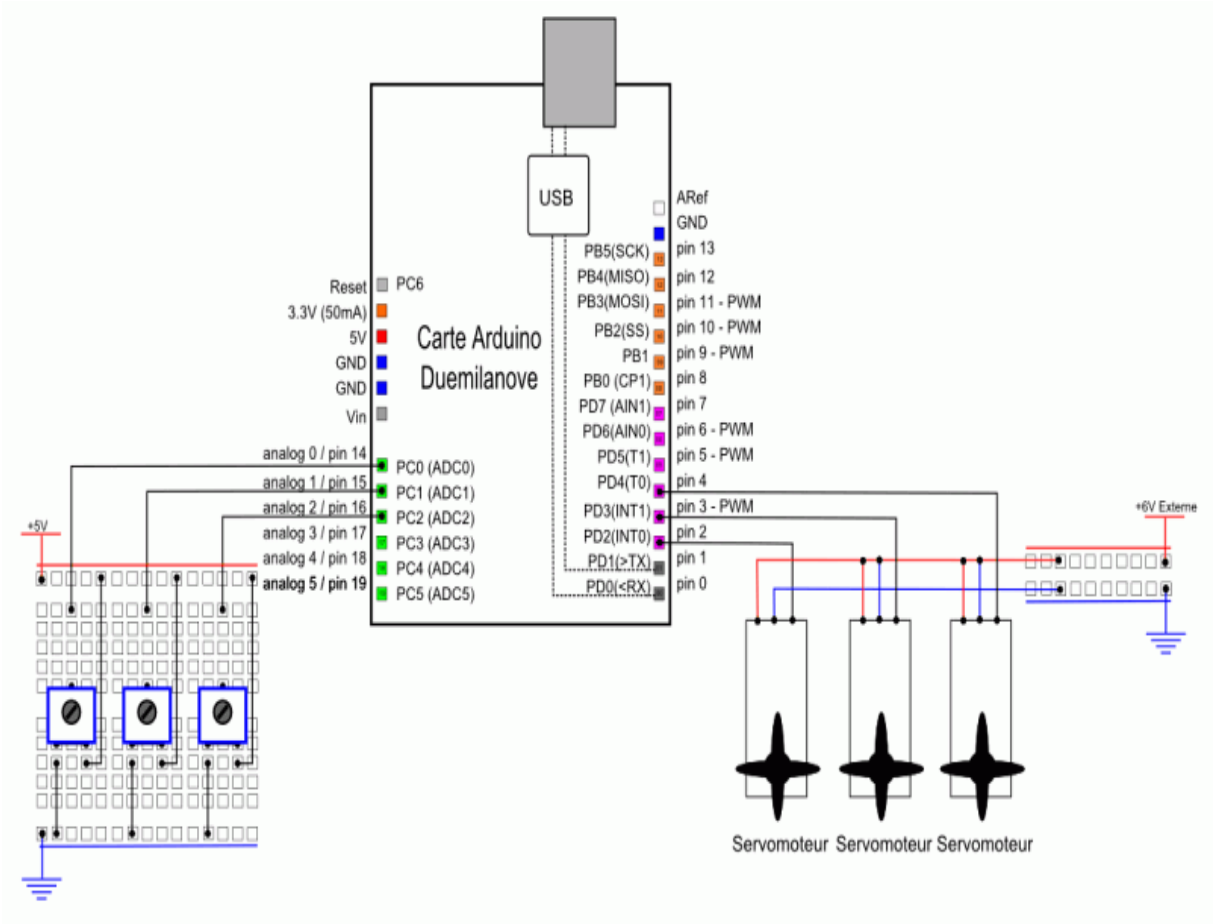
Il est donc nécessaire de mettre une alimentation extérieur pour pouvoir connecter 6 servomoteurs

Nous avons opté pour une alimentation 6V, a pile, idéal pour donner toute la puissance énergétique nécessaire à notre montage à moindre frais.

- 4 : Contrôle des servomoteurs



5 : montage global



Conclusion

Le PPE a permis à l'ensemble du groupe d'améliorer le processus de travail en équope déjà abordé l'an dernier avec les TPE pour Jules et Léo et les différents PPE réaliser l'année dernière par les redoublant. Travailler ensemble a permis d'échanger des idées pour trouver plus facilement les solutions puis de les selectionner. Cela nous a appris à coordonner nos efforts et à partager les connaissances et les savoir-faire de chacun en communiquant. Nous avons dû également apprendre à être plus autonome, faire des choix, reconnaitre nos erreurs, les corriger et gérer correctement le temps. Ces PPE sont en quelque sorte un aperçu de la démarche d'un projet, chose que nous rencontrerons certainement dans nos futures études, dans le métier d'ingénieur et probablement aussi dans la vie professionnel en général.

Pour finir, nous avons vraiment été enthousiasmé par le fait d'avoir à travailler sur un projet que nous avions choisi. La mise en route a été relativement difficile car l'horizon était très flou, mais après s'être concerté et le fait d'avoir défini une ligne de conduite grâce à l'étude de système, nous avons pu nous fixer des objectifs et travailler afin d'y répondre de la meilleur façon possible.