

III.1 INTRODUCTION

D'après les travaux de Sciavicco et al. [1], le problème de la commande d'un robot manipulateur peut être formulé comme la détermination de l'évolution des forces généralisées (forces ou couples) que les actionneurs doivent exercer pour garantir l'exécution de la tâche tout en satisfaisant certains critères de performance.

Différentes techniques sont utilisées pour la commande des bras manipulateurs. La conception mécanique du bras manipulateur a une influence sur le choix du schéma de commande. Un robot manipulateur est une structure mécanique complexe dont les inerties par rapport aux axes des articulations varient non seulement en fonction de la charge mais aussi en fonction de la configuration, des vitesses et des accélérations.

La plupart des robots utilisent des servomoteurs électriques comme actionneurs. Dans le cas de servomoteurs ayant de faibles rapports de réduction, ce sont les servomoteurs qui doivent compenser les effets des variations des forces d'inertie et de gravité. Dans le cas de servomoteurs avec de forts rapports de réduction, l'inertie vue par les moteurs varie beaucoup moins et il est alors possible de modéliser le robot par un système linéaire qui permet de découpler les articulations.

Dans le contexte de ce document nous considérons uniquement l'utilisation de servomoteurs avec de forts rapports de réduction comme actionneurs, ce qui produit des robots à articulations rigides.

Deux types de mouvements apparaissent quand on parle de commande du bras manipulateur :

- ✓ Un premier type considère que les mouvements nécessaires pour la réalisation de la tâche sont exécutés dans l'espace libre.
- ✓ Le deuxième type considère des mouvements spécifiques avec des forces de contact pour l'organe terminal qui se déplace dans un espace contraint.

Le projet qui nous avons présentés dans cette mémoire par la réalisation d'un bras manipulateur à base de l'Arduino.

III.2- GENERALITE SUR LA MAIN MECANIQUE

Les mains mécaniques qui ont repoussé encore un peu plus loin les limites du domaine, en développant chacune un aspect propre à la main humaine.

La première est la *Gifu Hand III* [2], qui si elle reprend une implantation classique de 16DDL contrôlés pour 20 au total n'en constitue pas moins une révolution au niveau de l'intégration. Equipée de pas moins de 859 capteurs tactiles répartis sur la main (313 sur la paume, 126 sur le pouce, 105 sur chaque doigt long) ainsi que de capteurs de position sur les moteurs et d'autres de force en bout de doigts (6 axes), elle est à ce jour la main artificielle la plus proche de l'humain du point de vue de l'instrumentation.



Figure III.1 : La Gifu Hand III nue (modèle de gauche) et couverte par son réseau de capteurs de contact (modèle de droite) [2].

Concernant la polyvalence et la robustesse nous trouvons la main *Robonaut II* [2], développée par la NASA pour être embarquée sur le robot éponyme et qui a finalement été envoyé sur la Station Spatiale Internationale en 2011. Cette main représente un excellent équilibre entre une structure cinématique correcte (19 DDL dont 13 contrôlés, avec une emphase sur le trio pouce-index majeur qui comprennent chacun respectivement quatre ,trois et trois DDL contrôlables individuellement ; les autres doigts servant de verrous pour les prises en force), une instrumentation étendue (capteurs de position sur les articulations et les moteurs, jauges de contraintes sur les tendons de transmission, capteurs de contact intégrés sur toute la surface du gant de protection, capteur de force 6 axes pour l'extrémité des doigts) et une excellente conception (intégration, robustesse, espace de travail très proche de celui de l'homme).

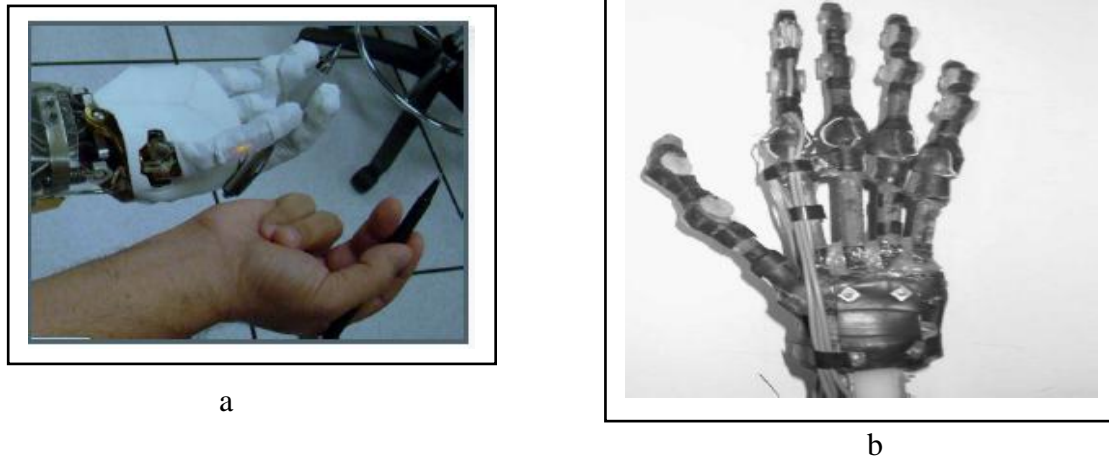


Figure III.2 : Deux mains complètes au plus proche de la réalité biologique : a) la main du Robonaut II par son efficacité, sa versatilité et sa robustesse, b) et Blackfingers Hand par sa conception [2].

Enfin, nous pouvons examiner la *Blackfingers Hand* qui, bien qu'elle soit loin des capacités de la *Robonaut Hand*, ne lui est pas moins supérieure en termes d'anthropomorphisme. Sa structure est en effet très fortement bio-inspirée: les segments qui la composent sont sculptés pour reprendre la forme des os humains ; les articulations sont constituées non pas de liaisons mécaniques classiques mais de surfaces de contact, encapsulées dans des ligaments artificiels ; son actionnement est assuré par des tendons que viennent contraindre des muscles pneumatiques développés par l'équipe (la première version utilisait des muscles hybrides pneumatique/hydraulique). Seule entorse à la réalité biologique : la flexion de la phalange médiane des doigts longs est dépendante de celle de la phalange distale (et non l'inverse, comme nous avons pu le voir à de nombreuses reprises jusqu'à présent). Chaque doigt est ainsi contrôlé par un jeu de six tendons / muscles antagonistes dotés de capteurs d'étirement. C'est à ce jour le modèle le plus proche de la main humaine, même s'il lui manque une réelle instrumentation (contacts et forces en surface) afin de le rendre contrôlable autrement qu'en boucle ouverte, ainsi qu'un revêtement lui permettant d'être tout simplement utilisable.

III.3- PARTIE PRATIQUE

III.3.1-Structure du bras

Les pièces du bras manipulateur ont été faites ainsi car elles permettent d'avoir un système léger et résistant (voir la Figure III.3).

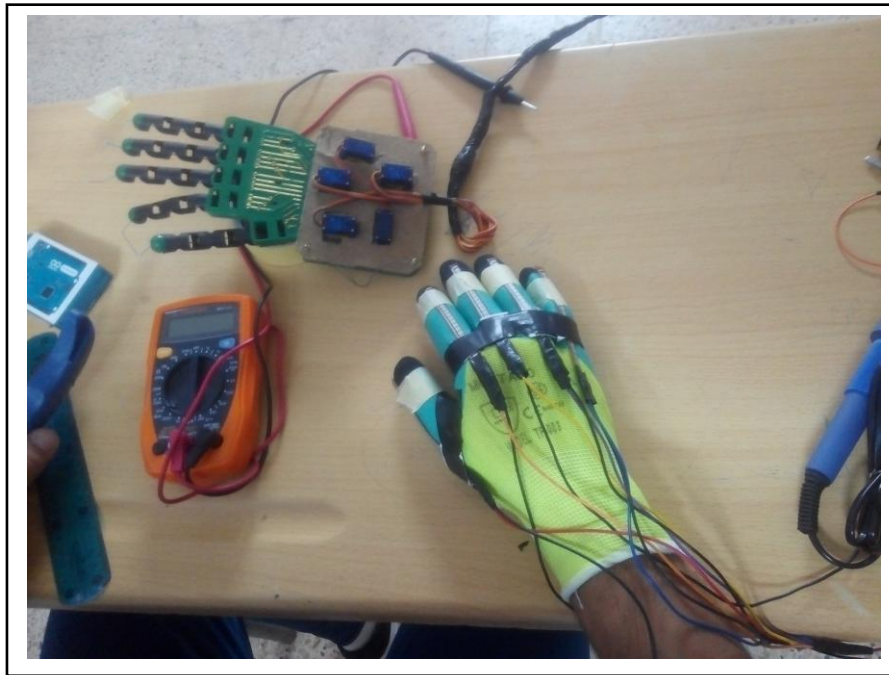


Figure III.3 : Photo de notre bras manipulateur.

III.3.2- Les composants électroniques

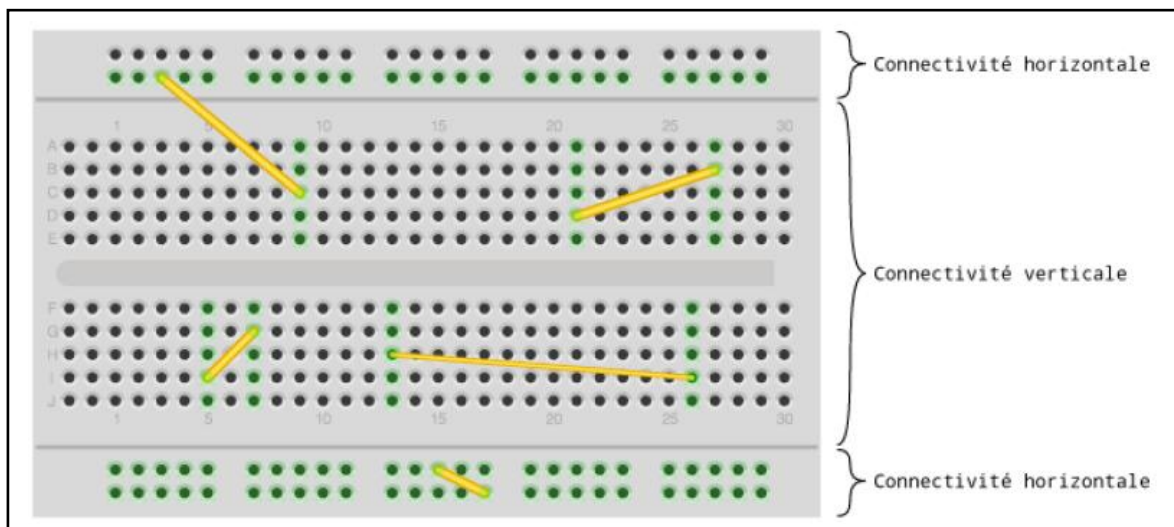


Figure III.4 : Photo d'une plaque d'essai [2].

Quand nous devons réaliser un projet avec Arduino, la première étape consiste à réaliser son montage sur une platine d'essai aussi appelée *breadboard*. Une platine d'essai est un support qui comporte des petits trous dans lesquels vous allez pouvoir positionner vos composants ainsi que des fils qui vous permettront de réaliser votre circuit électrique. À l'intérieur de votre platine, il faut imaginer qu'il y a des chemins déjà existants que nous pouvons voir dans le dessin ci-dessous. À ce stade, les connections doivent être faites avec des fils électriques monobrin dont l'aspect est rigide.

Une attention particulière est à apporter à la connexion de la batterie ou de l'alimentation électrique car un faux contact à ce niveau est particulièrement risqué pour votre montage.

a- Servomoteur

Le moteur électrique est un dispositif électromécanique qui fonctionne grâce à l'électromagnétisme généré par des bobines.

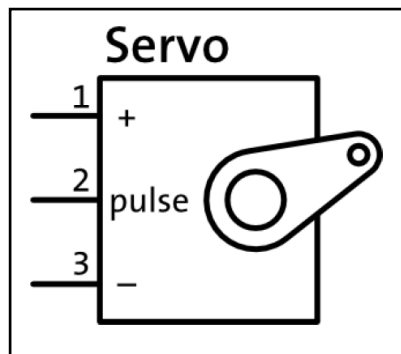


Figure III.5 : Schéma électrique d'un servomoteur [2].

Un servomoteur est constitué d'un moteur, d'un capteur de position et d'un régulateur électronique (voir la Figure III.5). Le fonctionnement du moteur est asservi à la position de l'axe. On le commande en lui indiquant quel angle doit prendre son axe (entre 0 et 180°) le moteur se met alors en marche jusqu'à ce que la position soit atteinte. Il existe de nombreux modèles. La Figure III.6 représente une photo de notre servomoteur utilisé dans notre projet.

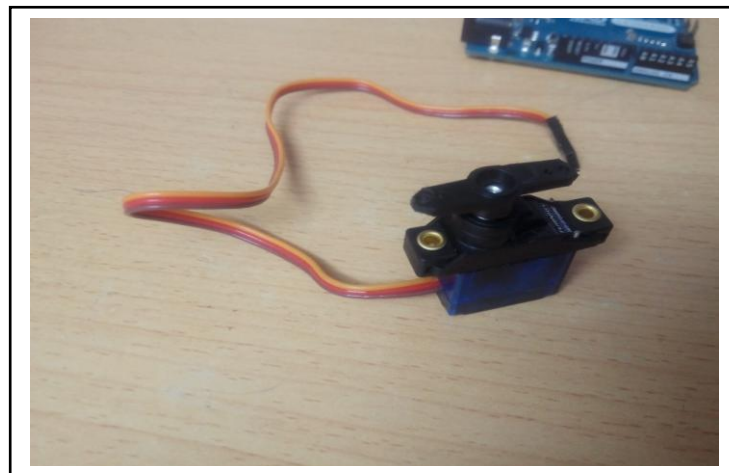


Figure III.6 : Photo d'un servomoteur.

Pour le raccordement des fils du servomoteur, on prend :

- Marron : masse.
- Rouge : +5V.
- Orange : commande, à raccorder à une broche de sortie de l'Arduino.

Dans notre projet, nous connecterons chaque servomoteur à l'Arduino comme le montre la Figure suivante :

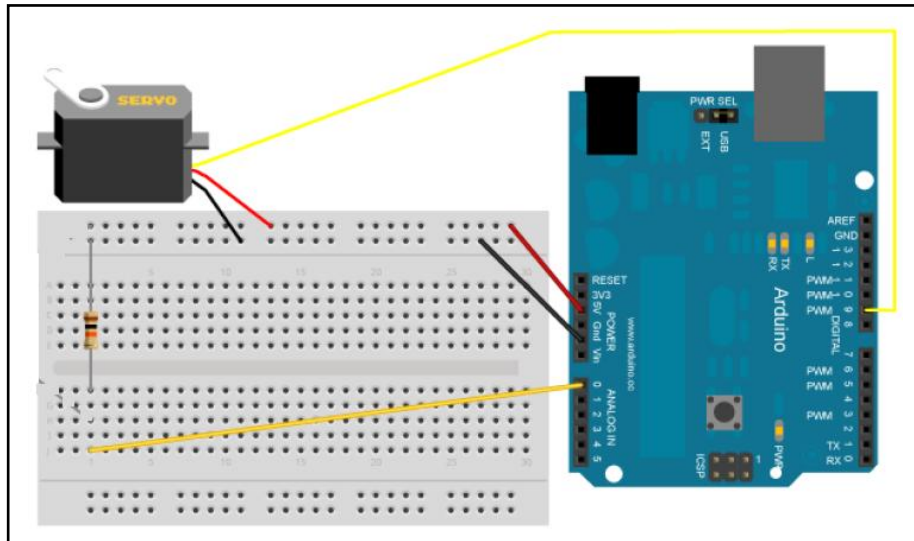


Figure III.7 : Schéma électrique de connexion d'un servomoteur à l'Arduino.

De plus, les servomoteurs sont des moteurs capables de maintenir une position à un effort statique et dont la position est vérifiée en continu 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 convertit un signal numérique en un angle qui sera reproduit sur le palonnier (voir la Figure III.8) grâce au moteur électrique à courant continu présent dans le servomoteur.

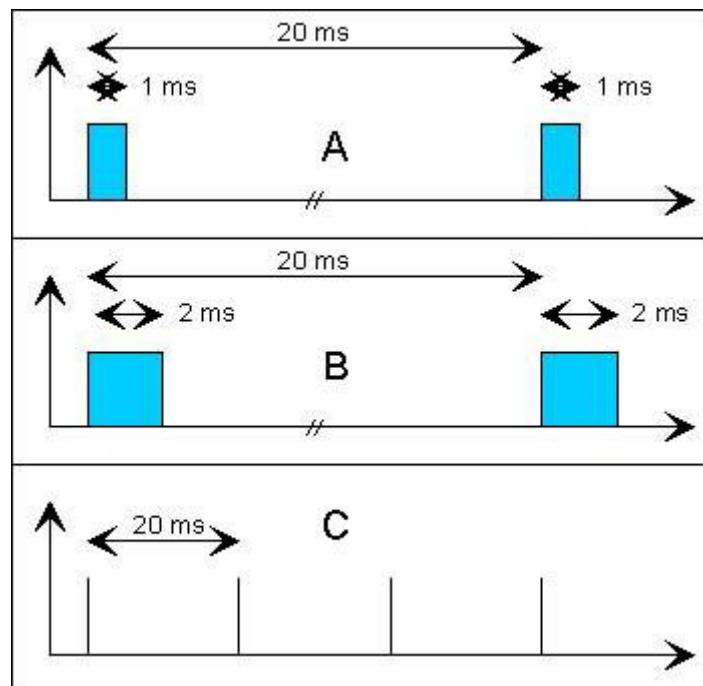


Figure III.8 : Image d'un signal numérique [3].

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'Analogique 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).

Pour réaliser ce montage, il vous faut connecter le servomoteur à la carte Arduino. Celui-ci comporte trois fils de couleurs habituellement rouge, noire et jaune (parfois blanc). Le fil noir doit être connecté à la *pin* GND, le fil rouge dans la *pin* 5 V (section power) de la carte et le fil jaune ou blanc doit être connecté à la *pin* 9 de la carte. Les fils rouge et noir servent à fournir l'alimentation au servomoteur. Le fil jaune sert à recevoir un signal PWM qui sera envoyé par l'Arduino afin de positionner le servomoteur.

Il est important de noter que pour ce projet, il n'y a cinq moteurs branché à la carte et utilisant la puissance 5 V qui provient du régulateur de la carte Arduino (qui l'obtient par le port USB), comme le montre la Figure suivante[14].

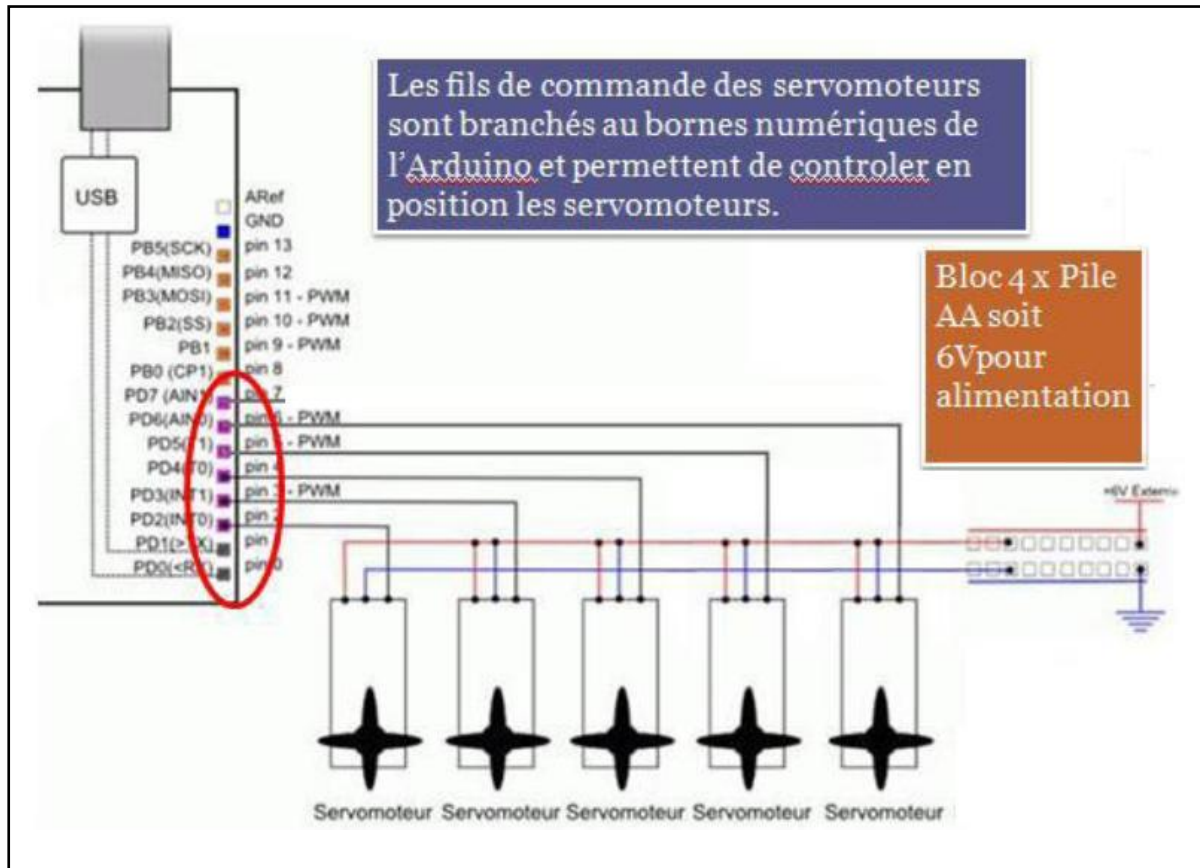


Figure III.9 : Connexion de cinq servomoteurs à l'Arduino.

Le programme de servomoteur utiliser dans la fenêtre de programmation Arduino est le suivant :

```
#include <Servo.h> // inclure la librairie pour l'utilisation des servomoteurs
Servo servo1; // créer l'objet "servo1" pour référer à notre servomoteur
// *****

// définition des variables
int pinCapteur = 0; // variable pour désigner quelle pin de l'Arduino est utilisée pour le capteur
int valeurCapteur = 0; // variable pour contenir la valeur du capteur.
int servoPos = 0; // variable pour contenir et spécifier la position en degrés (0-180)
void setup()
{
  Serial.begin(9600); // établir la connexion série à 9600 baud
  pinMode(pinCapteur, INPUT); // définir la pin 0 comme une entrée (pinCapteur)
  servo1.attach(9); // associer la PIN 9 au servomoteur
}
void loop()
{
```



```

// montrerValeurCapteur();
// delay(20); // laisser un court délai pour éviter le trop-plein d'informations
testerServomoteur();
}
void testerServomoteur()
{
while (1==1)
{
servoPos = random(181); // attribuer une valeur aléatoire comprise entre 0 et 180 à
servoPos
servo1.write(servoPos); // spécifier la position au servomoteur
delay(2000); // attendre 2 secondes
}
}
void montrerValeurCapteur()
{
valeurCapteur = analogRead(pinCapteur); // lire la pin analogique et mettre la valeur dans
valeurCapteur
Serial.println(valeurCapteur); // communiquer au moniteur sériel la valeur du capteur.
}

```

b- Résistance

La résistance s'oppose au passage du courant, proportionnellement à sa "résistance" exprimée en Ohm (voir la Figure III.10). Un code de couleurs, ci dessous permet de reconnaître cette valeur (voir la Figure III.11).



Figure III.10 : Symbole européen d'une résistance.

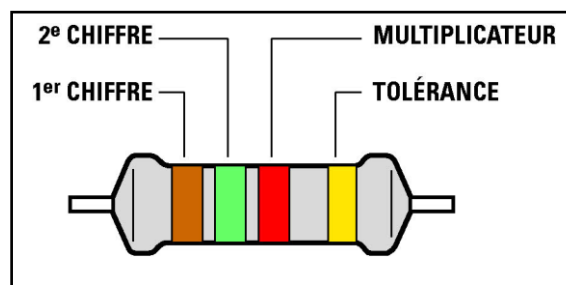


Figure III.11 : Résistance.

La résistance possède une suite d'anneaux de couleurs différentes sur son boîtier. Ces couleurs servent à expliciter la valeur de la résistance sans avoir besoin d'écrire en chiffre dessus. Le premier anneau représente le chiffre des centaines, le second celui des dizaines et le troisième celui des unités. Enfin, après un petit espace vient celui du coefficient multiplicateur.

Avec ses quatre anneaux et un peu d'entraînement vous pouvez alors deviner la valeur de la résistance en un clin d'œil. Le tableau III.1 nous permettra de lire ce code qui correspond à la valeur de la résistance :

| <u>Couleurs</u> | <u>1^{er} chiffre</u> | <u>2^{ème} chiffre</u> | <u>Multiplicateur</u> | <u>Tolérance</u> |
|-----------------|-------------------------------|--------------------------------|-----------------------|------------------|
| Noir | = | 0 | 10^0 | = |
| Marron | 1 | 1 | 10^1 | 1% |
| Rouge | 2 | 2 | 10^2 | 2% |
| Orange | 3 | 3 | 10^3 | = |
| Jaune | 4 | 4 | 10^4 | = |
| Vert | 5 | 5 | 10^5 | = |
| Bleu | 6 | 6 | 10^6 | = |
| Violet | 7 | 7 | 10^7 | = |
| Gris | 8 | 8 | 10^8 | = |
| Blanc | 9 | 9 | 10^9 | = |
| Or | = | = | 10^{-1} | 5% |
| argent | = | = | 10^{-2} | 10% |
| Sans couleur | = | = | = | 20% |

Tableau III.1 : Code couleur d'une résistance.

Dans notre nous somme utilisé cinq résistances de valeur égale à 22 K Ω . La Figure III.12 représente une résistance connectée à l'Arduino.

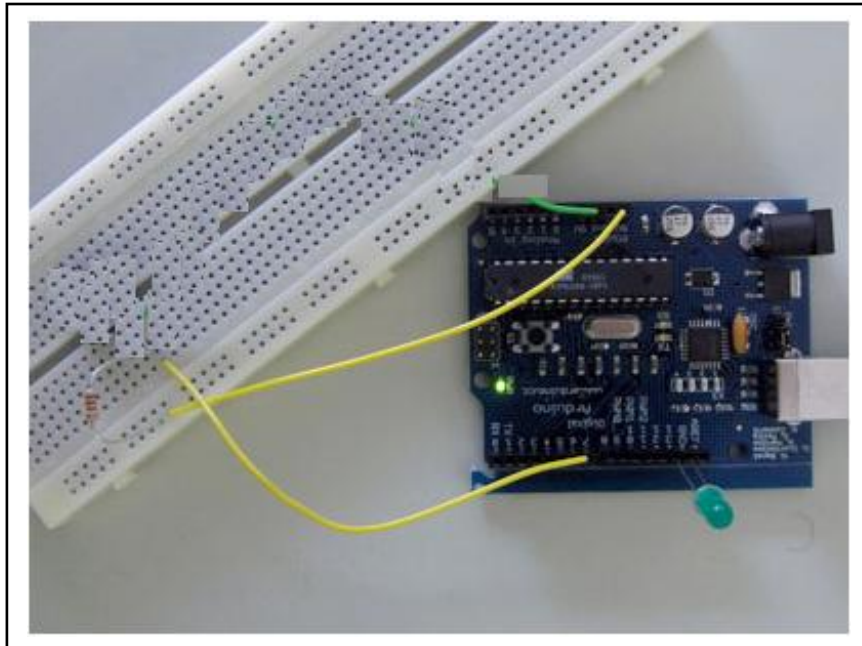


Figure III.12 : Schéma électrique de connexion d'une résistance à l'Arduino.

Le programme utilisé pour les résistances à l'Arduino est le suivant :

```
/*  
Adafruit Arduino - Resistance  
*/  
int redPin = 11;  
int greenPin = 10;  
int bluePin = 9;  
void setup()  
{  
  pinMode(redPin, OUTPUT);  
  pinMode(greenPin, OUTPUT);  
  pinMode(bluePin, OUTPUT);  
}  
void loop()  
{  
  setColor(255, 0, 0); // red  
  delay(1000);  
  setColor(0, 255, 0); // green  
  delay(1000);  
  setColor(0, 0, 255); // blue  
  delay(1000);  
  setColor(255, 255, 0); // yellow  
  delay(1000);  
  setColor(80, 0, 80); // purple  
  delay(1000);  
  setColor(0, 255, 255); // aqua  
  delay(1000);  
}
```

```

int redPin = 11;
int greenPin = 10;
int bluePin = 9;
void setup()
void setColor(int red, int green, int blue)
{
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
}

```

c- Capteur flexible

Le principe de ce capteur de flexion est ne peut plus enfantin : il se présente en effet sous nla forme d'une fine barette de 7.3 cm de long sur 6.3 mm de large (voir la Figure III.13). Au repos, la résistance appliquée est 25 kOhm. Plus vous pilez le capteur et plus la résistance augmente avec un niveau maximal de 125 kOhm.



Figure III.13 : Photo d'un capteur flexible.

Le capteur présente à son extrémité une limande souple dotée de deux connecteurs mâles à souder. A noter : prenez garde de bien piler le capteur de flexion uniquement sur sa zone active. Le piler trop souvent au niveau de sa base pourrait endommager le dispositif.

Le schéma électrique du capteur flexible est donnée par la Figure III.14.

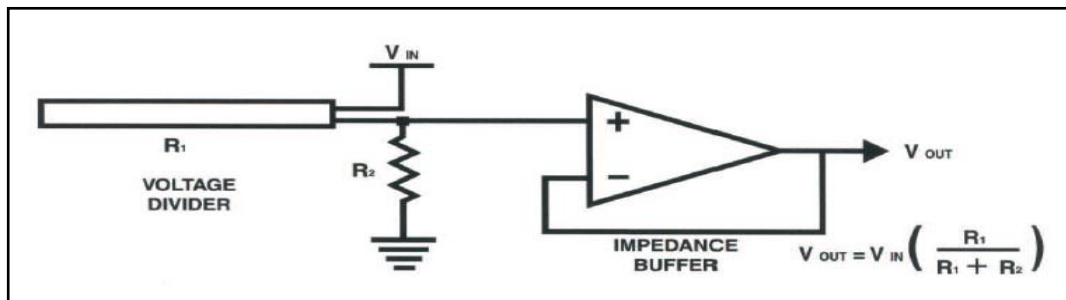


Figure III.14 : Schéma électrique équivalent d'un capteur flexible.

La Figure III.15 représente la connexion du capteur flexible avec l'Arduino.

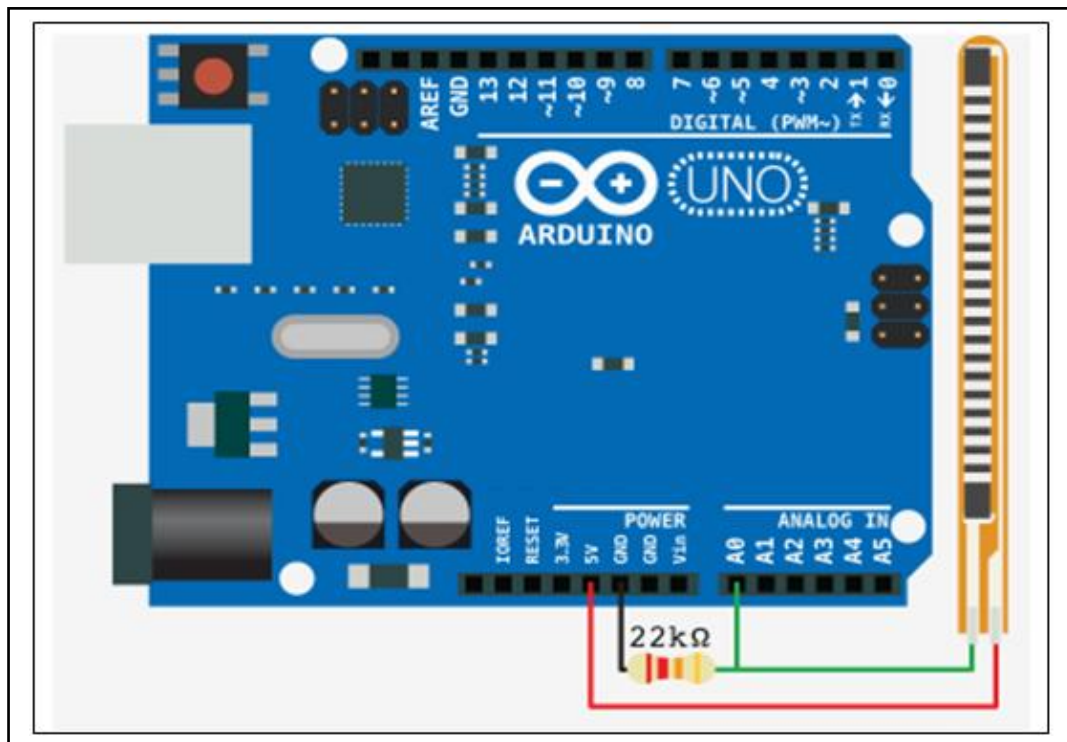


Figure III.15 : Schéma électrique de connexion d'un capteur flexible à l'Arduino.

Le programme utilisé pour le capteur flexible par l'Arduino est le suivant :

```
int flexSensorPin = A0; //analog pin 0

void setup(){
  Serial.begin(9600);
}

void loop(){
  int flexSensorReading = analogRead(flexSensorPin);
  Serial.println(flexSensorReading);
  //In my tests I was getting a reading on the arduino between 512, and 614.
  //Using map(), you can convert that to a larger range like 0-100.
  int flex0to100 = map(flexSensorReading, 512, 614, 0, 100);
  Serial.println(flex0to100);
  delay(250); //just here to slow down the output for easier reading
}
```

III.4- CONCLUSION

Le présent chapitre constitue le cœur de notre projet. En effet, nous avons réalisé un bras manipulateur à base de l'Arduino. Elle consiste à gérer les mouvements de notre bras en appliquant la commande qui utilise le port série USB.

Dans le contexte de ce document nous considérons uniquement l'utilisation de servomoteurs avec de forts rapports de réduction comme actionneurs, ce qui produit des robots à articulations rigides.

Deux types de mouvements apparaissent quand on parle de commande du bras manipulateur :

- ✓ Un premier type considère que les mouvements nécessaires pour la réalisation de la tâche sont exécutés dans l'espace libre.
- ✓ Le deuxième type considère des mouvements spécifiques avec des forces de contact pour l'organe terminal qui se déplace dans un espace contraint.

Le projet qui nous avons présentés dans cette mémoire par la réalisation d'un bras manipulateur à base de l'Arduino