II.1- INTRODUCTION

La commande en mouvement des bras manipulateurs est essentiellement réalisée par des circuits numériques programmables tels que les microprocesseurs, les microcontrôleurs et les DSP où les lois de commande sont implémentées sous forme d'algorithmes. A noter que les principaux avantages de la commande numérique sont la rapidité de développement, flexibilité et faible espace d'occupation.

Dans ce chapitre on présente les différents éléments de la carte Arduino, qui on a utilisé dans notre projet. Cette carte de commande est contrôlé par ordinateur qui permettant de contrôlé un bras manipulateur. Pour cela, le dispositif expérimental de la Figure II.1 dans le cadre de ce projet est constitué de deux paries essentielles :

1. Partie logicielle ;
2. Partie réalisation du bras manipulateur.

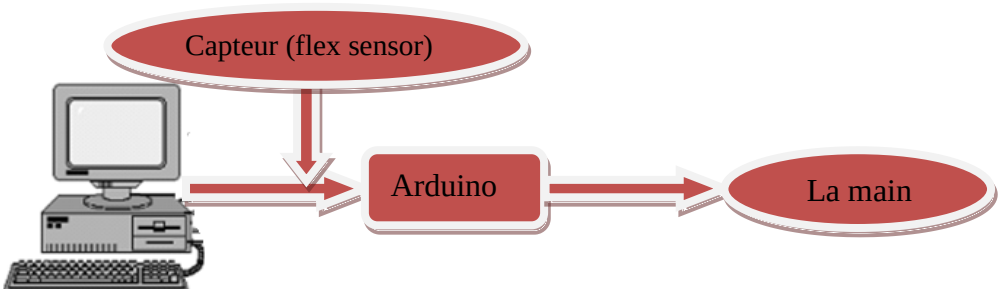

Figure II.1 : Schéma synoptique de commande d'un bras manipulateur.

II.2- DESCRIPTION DES DIFFERENTS ELEMENTS HARDWARE ET DE SOFTWARE DE L'ARDUINO

Arduino est une plate-forme de prototypage d'objets interactifs à usage créatif constituée d'une carte électronique et d'un environnement de programmation. De plus, l'Arduino est un circuit imprimé en matériel libre sur lequel se trouve un microcontrôleur qui peut être programmé pour analyser et produire des signaux électriques, de manière à effectuer des tâches très diverses comme le pilotage d'un robot. C'est une plate-forme basée sur une interface entrée/sortie simple.

Pont tendu entre le monde réel et le monde numérique, Arduino permet d'étendre les capacités de relations humain/machine ou environnement/machine.

II.2.1- Carte électronique de l'Arduino

La carte Arduino repose sur un circuit intégré (un mini ordinateur appelé également microcontrôleur) associée à des entrées et sorties qui permettent à l'utilisateur de brancher différents types d'éléments externes.

Le schéma électrique de notre carte de commande est représenté à la Figure II.2. Cette carte permet d'actionner un bras manipulateur à partir de la sortie USB du PC (voir la Figure II.3). Lorsque la carte arduino est connectée au port USB de l'ordinateur, celui lui fournie l'énergie électrique nécessaire à son fonctionnement. Une fois le programme chargé, on peu débrancher le cordon USB et connecter la carte soit à une pile, soit à un transformateur (voir les Figures II.4 et II.5).

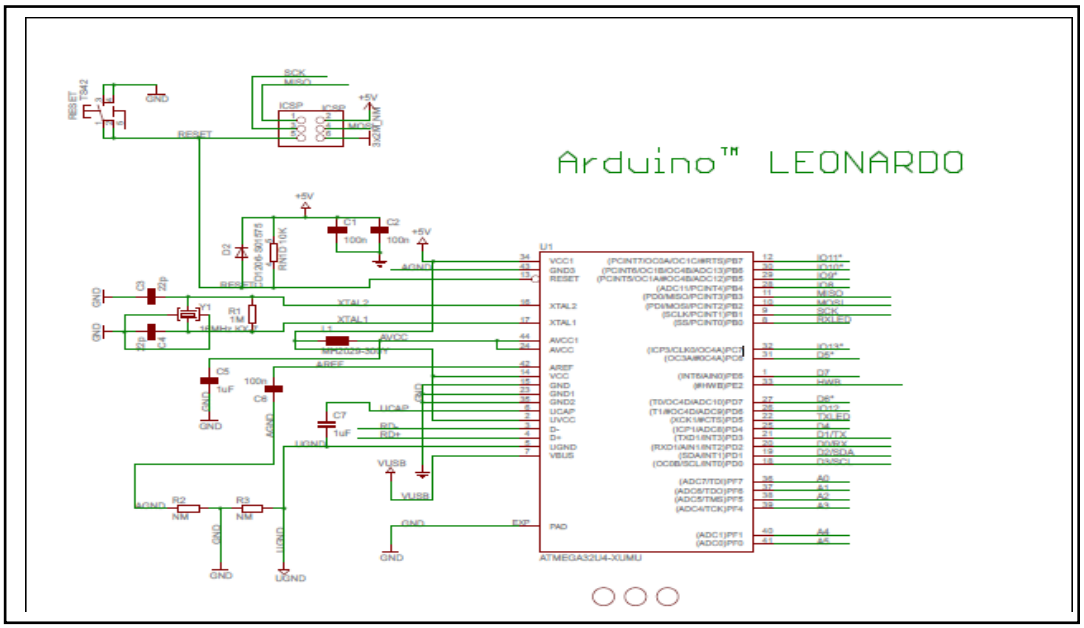

Figure II.2 : Schéma électrique de la carte de commande**.**

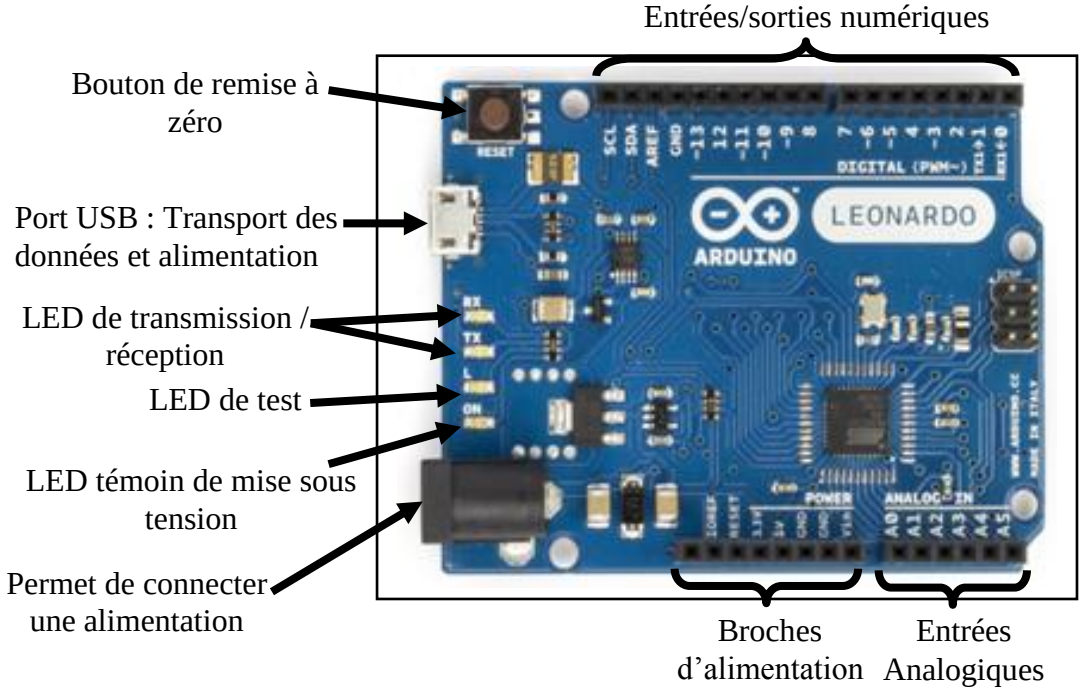

Figure II.3 : Photo de la carte arduino.

Les différents éléments de la carte de commande sont regroupés dans le Tableau II.1.

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

Tableau II.1 : Différents éléments de la carte de commande.

Figure II.4 : Photo d'une carte arduino connectée avec une pile 9 V.

Figure II.5 : Phot d'un transformateur qui convertie la tension du secteur en une tension continue (DC) 9 V.

a- Entrée/Sortie numérique

La carte Arduino possède 14 entrées / sorties numériques (digital en anglais) D0 à D13. Dans « void setup », il faut déclarer une broche comme une entrée ou comme une sortie par une des deux instructions suivantes :

pinMode (nom_de_broche, INPUT) ; // broche en entrée

pinMode (nom_de_broche, OUTPUT) ; // broche en sortie

- En sortie, on envoie soit 5V sur la broche, soit 0V. Cela correspond à un « 1 » ou à un « 0 », à un niveau « haut » ou à un niveau « bas » (voir la figure II.6).

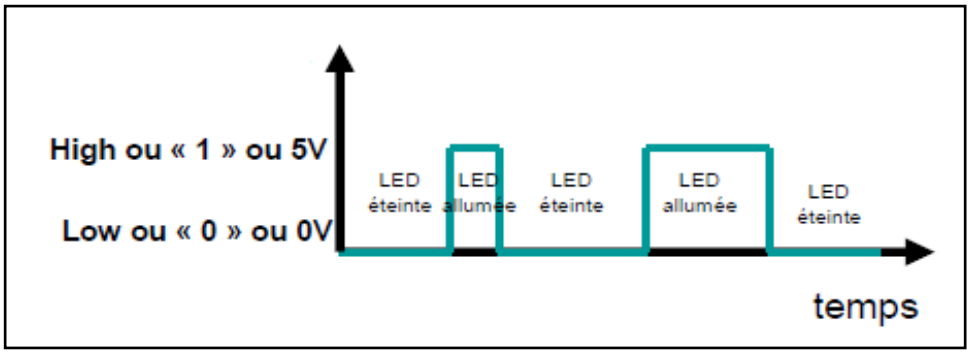

Figure II.6 : Chronogramme de l'entrée/sortie numérique.

Dans le programme cela correspond aux instructions suivantes :

digitalWrite(nom_de_broche, HIGH) ; // envoie 5V sur la broche soit « 1 ».

digitalWrite(nom_de_broche, LOW) ; // envoie 0V sur la broche soit « 0 ».

- En entrée, la carte peut lire soit un niveau haut (« 1 » ou HIGH), soit un niveau bas (« 0 » ou LOW) (voir la figure II.6).

Dans le programme cela correspond aux instructions suivantes :

digitalWrite(nom_de_broche, HIGH) ; // lit 5V sur la broche soit « 1 ».

digitalWrite(nom_de_broche, LOW) ; // lit 0V sur la broche soit « 0 »

b- Les entrées analogiques

Contrairement au signal numérique qui ne peut prendre que deux états différents, un signal analogique peut prendre une infinité de valeurs. Comme une tension que l'on fait varier progressivement de 0V à 5V (voir la figure II.7). Les entrées de A0 à A5 sont dotées de convertisseurs analogique/numérique qui convertit une tension en une suite de « 0 » et de «1 » que la carte fait correspondre à un nombre variant de 0 à 1023.

On peut ainsi récupérer les informations d'un capteur.

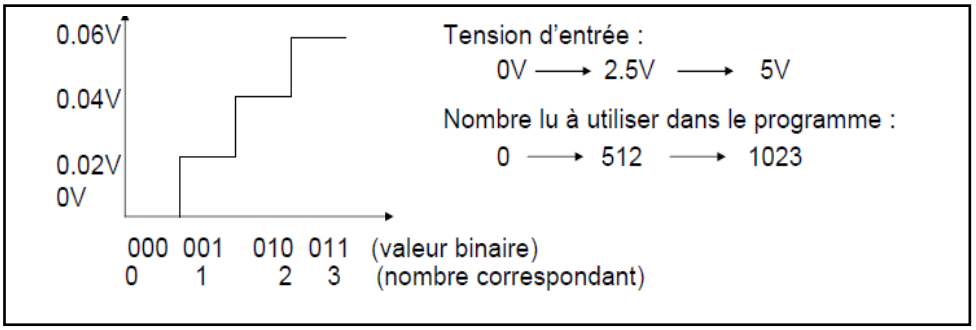

Figure II.7 : Chronogramme des entrées analogiques.

c- Le câble USB

1. Définition du câble USB

Le câblage USB est relativement simple ; il a la même structure quelle que soit la vitesse de transmission. Le câble transporte deux paires de fils : La paire de signal destinée au transfert de données D+ et D- et une seconde paire qui peut être utilisé pour la télé alimentation GND et Vcc. La première paire est non blindée pour les périphériques lents tels que les claviers, souris fonctionnant à 1.5Mbits/s tandis que caméras, micro et autres ont recours à une paire de fils torsadée blindée pour atteindre les 12Mbits/s.

2. Composition du câble USB [1].

Chaque connecteur dispose de deux fils d'alimentation (5V et GND) et deux fils destinés au transfert de données (D+ et D-). Une connexion entre deux PC est aussi possible par l'adjonction d'une interface spéciale qui déjoue la vigilance du PC maître et transforme le second PC en «esclave» En version Low Speed le blindage n'est pas obligatoire (ce qui assure une plus grande souplesse de manipulation en particulier pour une liaison souris).

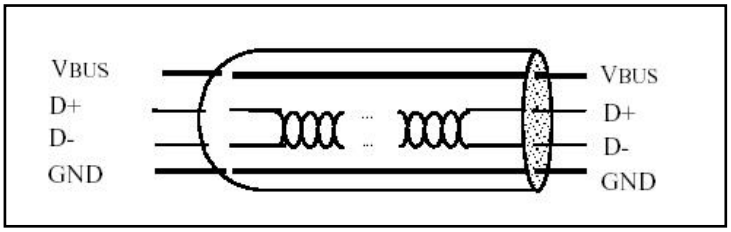

Figure II-8 : Composition d'un câble USB .

La longueur maximale autorisée par la norme est de 3m pour un câble non blindé donc généralement pour un périphérique Low USB (= 1.5Mb/s) et de 5m pour un câble blindé dans le cas d'un périphérique Full USB (=12Mb/s).

Le câble USB est composé de deux fiches bien différentes :

En amont d'une fiche appelé connecteur USB de type A, branché au host (PC).

L'extrémité aval par contre peut se retrouver en deux versions :

Connecteur USB du type B et un mini connecteur type B (appelé souvent optionnel). Ce dernier est réservé aux dispositifs de très faible dimension (ou de grande intégration) tels les appareils photo numériques.

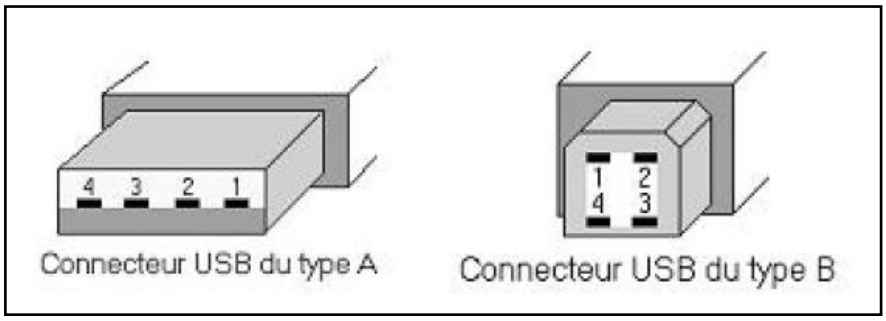

Figure II-9 : Brochage des connecteurs USB de type A et B .

3. Protocole USB

Le protocole USB est, comme tout les autres protocoles un protocole à encapsulation. Mais avant de s'intéresser à la trame proprement dite, voyant ce qu'il en est du protocole.

Le client driver communique les demandes de transfert des applications via des IRP (I/O Packet). Puis, l'USB driver traduit chaque transfert en une suite de transactions. Ensuite l'USB Host Controller driver regroupe les transactions en trames et finalement l'USB Host Controller traduit les transactions en paquets et enchaîne les trames.

Le synoptique ci dessous traduit de façon imagée les liens entre les différents éléments.

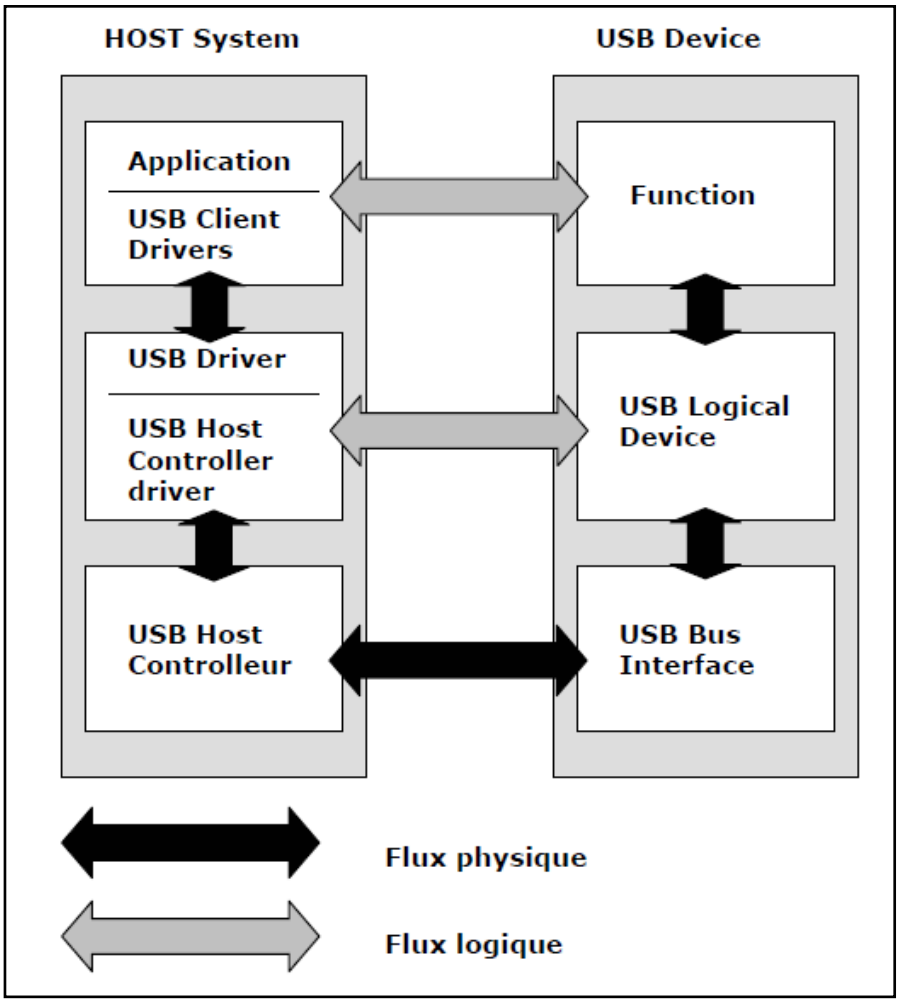

Figure II-10 : Protocole USB .

4. Type de paquet USB

Contrairement à la liaison série RS232 et des interfaces séries similaires où le format des données envoyées n'est pas défini, l'USB lui est composé de plusieurs couches de protocoles bien définis. La plupart des circuits intégrés USB s'occuperont de la couche inférieure, la rendant ainsi presque invisible au regard du concepteur final. Mais il est tout de même intéressant d'en connaître les grandes lignes.

L'USB a quatre types différents de paquet :

- Token (En-tête).
- SOF (Start of frame).
- Data (Optionnel).
- Acknowledge (Handshake).

Les **paquets jetons** indiquent le type de la transaction qui va suivre et a pour but de transporter l'adresse USB et le sens du transfert. Les **paquets de données** contiennent les données utiles. Les **paquets "Handshake"** sont utilisés pour valider les données ou rapporter les erreurs. Et les **paquets début de trame (SOF)** indiquent le commencement d'une nouvelle trame.

L'entité de transfert USB est appelée *transaction*, elle est généralement constituée de paquets juxtaposés, la transition étant un paquet SOF qui indique le début d'une autre transaction. On dit alors qu'un transfert est compose d'une succession de transactions.

Voici ci dessous le détail d'une transaction :

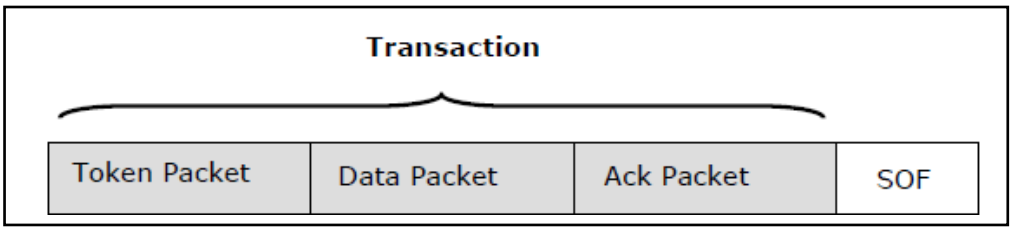

Figure II-11 : Transaction USB .

Le premier bit transmit est le bit LSB.

Les différents paquets ont une structure bien définie que voici:

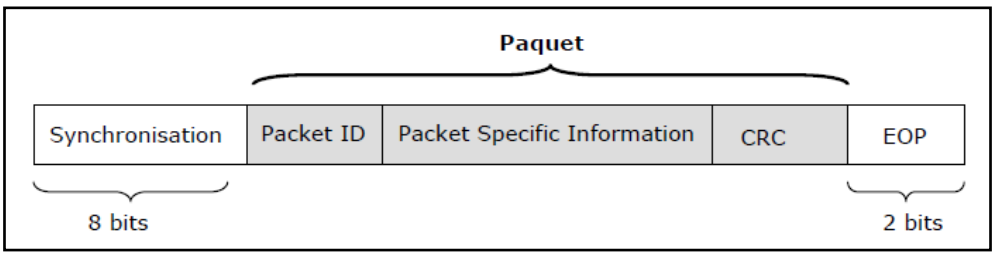

Figure II-12 : Structure des paquets USB [2].

La structure des paquets étant identique il faut préciser que le format est différent selon la nature du paquet. Voici ci dessous les différents formats de paquet. Les trames ont une durée de 1ms, elles sont marqués par le Token SOF (Start of Frame). Avec le système d'encapsulation, il faut bien noter que les transactions sont regroupées à l'intérieur des trames sans qu'il y ait de chevauchement. Notons également qu'une trame peut contenir une ou plusieurs transactions destinées à un même périphérique.

II.2.2- Software de l'Arduino

Le logiciel de programmation des modules Arduino est une application Java multi-plateforme (IDE en anglais) servant d'éditeur de code et de compilateur. Celui-ci permet de transférer le programme au travers de la liaison série.

Figure II-13 : Logo de l'Arduino.

Le langage de programmation utilisé est le C++, lié à la bibliothèque de développement Arduino, permettant l'utilisation de la carte et de ses entrées/sorties, d'une part.

D'autre part, l'IDE permet d'écrire, de modifier un programme et de le convertir en une série d'instructions compréhensibles pour la carte.

a- Présentation

C'est la première page qui apparaît lors d'exécution de programme (voir la figure II.8). Un programme utilisateur Arduino est une suite d'instructions élémentaires sous forme textuelle, ligne par ligne. La carte lit puis effectue les instructions les unes après les autres, dans l'ordre défini par les lignes de code.

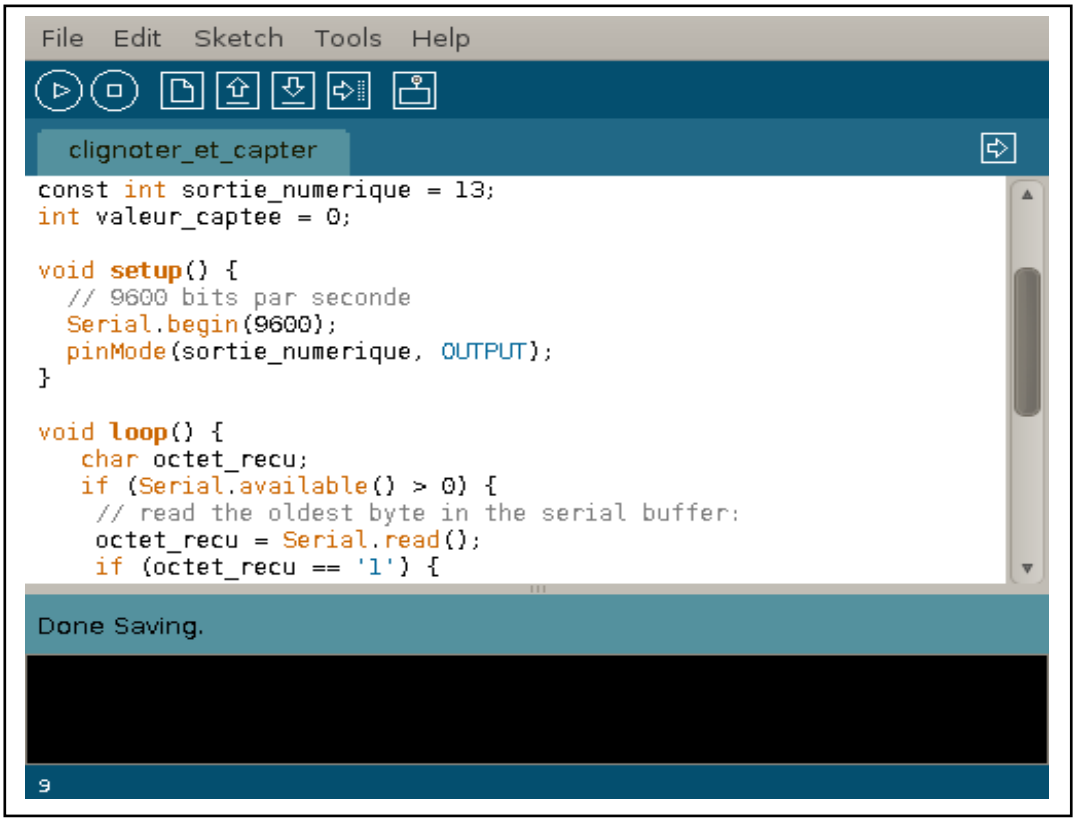

Figure II.14 : La forme de présentation [3].

D'après la Figure II-14, on trouve :

New : pour créer un nouveau programme (sketch).

Open : ouvrir un programme existant. Le menu n'est pas déroulant à cause d'un bug...pour obtenir un menu déroulant passer par file/open.

Save : sauvegarde le programme, si vous voulez le sauvegarder sous un autre nom, passer par file/save as.

Serial Monitor : pour ouvrir la fenêtre qui permet de visualiser les données transmise par le port série de l'Arduino (très pratique pour le débogage...)

La syntaxe des commandes Arduino est donnée par la Figure II-15.

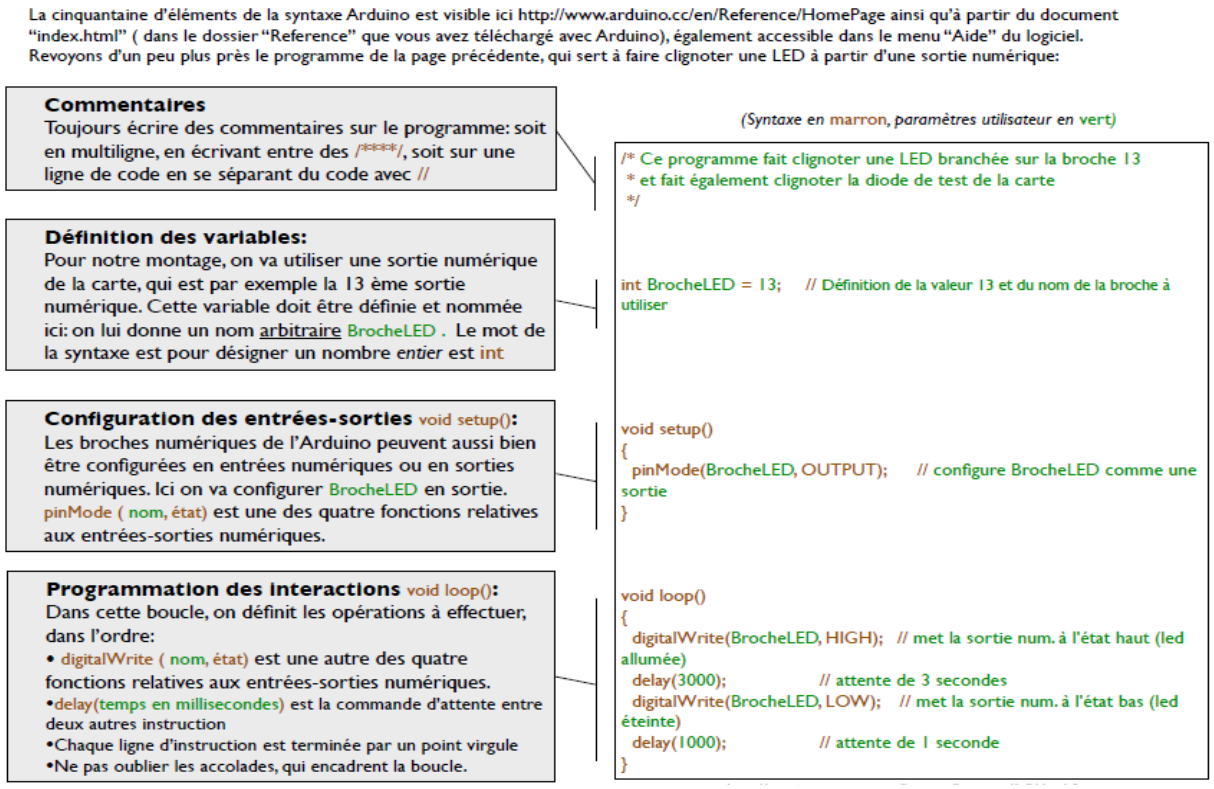

Figure II-15 : Syntaxe des commandes Arduino.

Le programme est lu par le microcontrôleur de haut vers le bas. Une variable doit être déclarée avant d'être utilisée par une fonction.

La structure minimale est constituée :

- en tête : déclaration des variables, des constantes, indication de l'utilisation de bibliothèques etc...

- ❖ un setup (= initialisation) cette partie n'est lue qu'une seule fois, elle comprend les fonctions devant être réalisées au démarrage (utilisation des broches en entrées ou en sortie, mise en marche du midi, du port série de l' I2C etc.....).
- ❖ une loop (boucle) : cette partie est lue en boucle ! C'est ici que les fonctions sont réalisées.

En plus de cette structure minimale, on peut ajouter :

- ❖ des « sous-programmes » ou « routines » qui peuvent être appelées à tous moments dans la boucle, très pratiquer pour réaliser des morceaux de codes répétitifs.
- ❖ Des « callbacks », ce sont des fonctions qui sont rappelées automatiquement depuis une bibliothèque.

Par exemple pour désigner le bon port Série (USB-Série), on suive les étapes comme le montre la figure II-16.

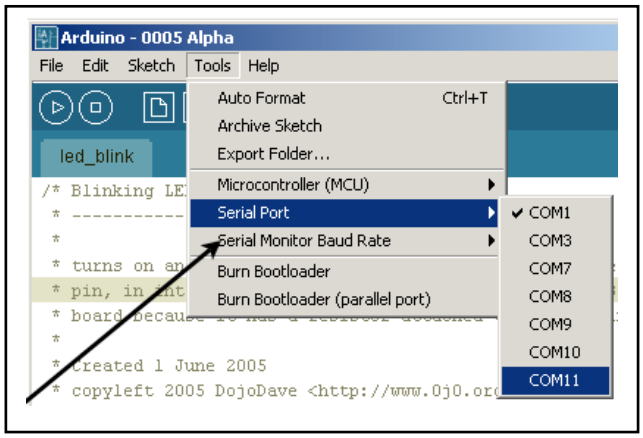

Figure II.16 : Comment désigner un bon port série.

b- Le programme

****

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 // 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 utilisées pour la conversion analogique-numérique

Pour obtenir un bon fonctionnement entre la carte Arduino et notre bras manipulateur il faut suivit les étapes suivantes :

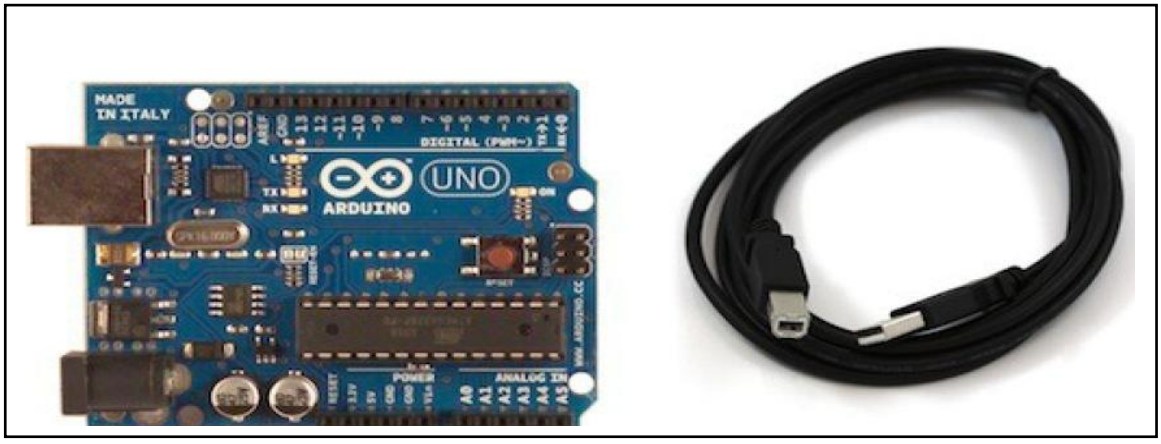

Figure II-17 : Photo de la carte Arduino et la câble USB.

1. Raccorder la carte à l'ordinateur : La diode verte doit s'allumer.

2. Installation des pilotes du périphérique Série-USB.
   - lors d'un premier raccordement d'une carte UNO à l'ordinateur, ce dernier recherche
   - automatiquement un pilote. Après quelques instants, l'ordinateur va indiquer qu'il n'a pas
   - trouvé de pilote...
   - ouvrir « exécuter » en passant par le menu démarrer
   - taper devmgmt.msc et taper sur la touche « entrée »
   - Chercher et déployer la rubrique « ports (COM et LPT).
   - Il doit apparaître une ligne «Arduino UNO (COMxx) »
   - faire un click droit sur «Arduino UNO (COMxx) » et choisir « mettre à jour le pilote... ».
   - Choisir « rechercher un pilote sur mon ordinateur »
   - sélectionner le fichier "ArduinoUNO.inf", qui se trouve dans le repertoire « drivers » de
   - l'installation de l'Arduino...
   - Windows va terminer l'installation des pilotes.
3. Lancer l'application Arduino : Double -cliquer su arduino.exe
4. Ouvrir l'exemple « blink » : Ouvrir le programme (sketch)
   - File > Exemples > 1.Basics > Blink.
5. Choisir la carte UNO

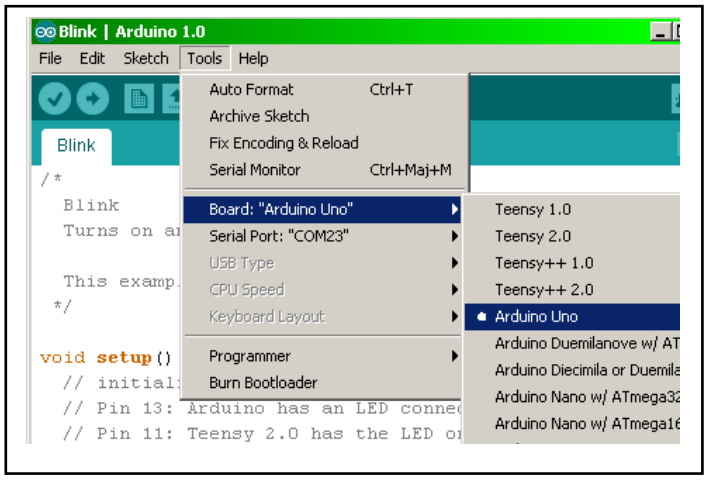

Figure II-18 : Photo comment choisir la carte UNO.

6. Sélectionner le port série : Choisir le port série qui est utilisé par la carte Arduino : Tools > serial port > … pour connaître quel port série est utilisé par la carte Arduino, lancer le gestionnaire de périphérique

7. Charger le programme dans la carte Arduino :Cliquer sur la touche « upload »

II.3- DOMAINE D'UTILISATIO DE LA CART ARDUINO

Parmi les plusieurs applications de la carte de l'Arduino, on peut citer :

Figure II-19 : Photo du Frida V [4].

Fixé sur un vélo, Frida V prend la forme d'un boîtier contenant un ensemble de capteurs qui analysent et compilent les informations récoltées durant le parcours du cycliste : pollution atmosphérique (capteurs de gaz), wifi ouvert dans la ville (routeur wifi), position géographique (GPS), documentation audiovisuelle des parcours (caméra et microphone). L'ensemble est assemblé dans une coque résistante et portable adaptable sur tout type de vélos.

Les données recueillies sont archivées sur un serveur et représentées graphiquement sur une carte, la *MeTaMap*. Ces informations peuvent être annotées par les participants en direct grâce aux contrôleurs de la Frida. La Frida est constituée notamment d'un routeur Linux embarqué et d'un système de micro-Arduino. Les logiciels sont développés en libre et adaptables sur de nombreuses autres plates-formes matérielles de type ordi phones.

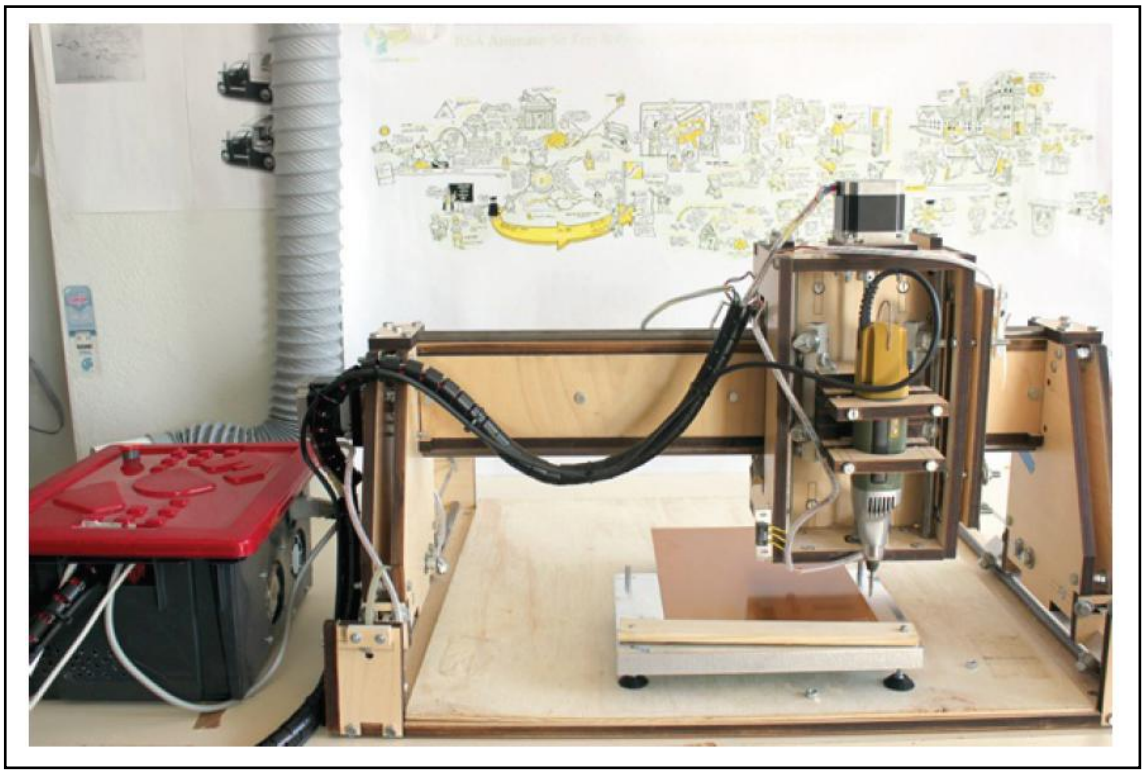

Figure II-20 : Photo d'un Fraiseuse numérique [6].

Arduino peut agir comme un automate de commande pour piloter une machine à commande numérique fabriquée maison, qui coûte bien moins cher qu'un modèle acheté, comme cette fraiseuse, inspirée du modèle canadien « Oomloot » adapté aux mesures métriques et réalisée durant le boot camp « Make ta machine » à Nantes au printemps 2011.

Cette machine peut réaliser des gravures, des découpes et des usinages dans divers matériaux (bois, aluminium, circuits électroniques, etc.).

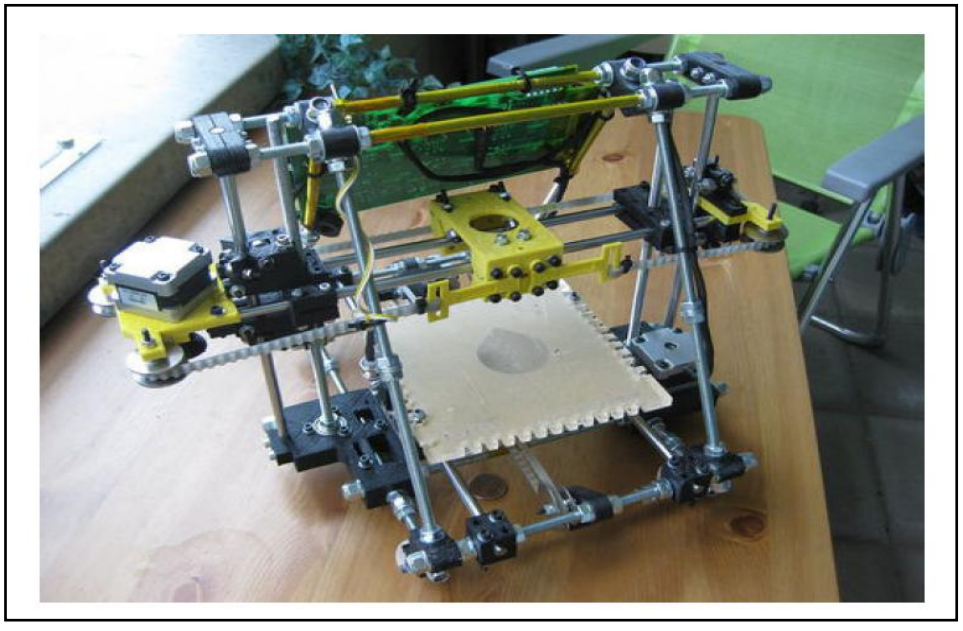

Figure II-21 : Photo d'un imprimante 3D Reprap [8].

RepRap est une imprimante de bureau 3D libre capable de fabriquer des objets en plastique. Comme la plupart des pièces de la Reprap sont faites de cette matière et que la Reprap peut les créer, cette machine est considérée comme « auto-réplicable ».
Reprap s'inscrit dans le mouvement du matériel libre et de la fabrication de machines-outils à faible coût. L'absence de brevet et la mise à disposition gratuite des plans offrent la possibilité à quiconque de la construire avec du temps et un minimum de matériel.

II.4- CONCLUTION

Dans ce chapitre, nous avons présenté les différents éléments de la carte de commande ainsi que le logiciel. Cette interface graphique permettant la communication Homme/Machine. Elle consiste à gérer le positionnement de notre bras manipulateur en utilise le port USB de l'ordinateur.

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.