République Algérienne Démocratique et Populaire
Ministère de L'Enseignement Supérieur et de la A Recherche Scientifique

UNIVERSITE MOULOUD MAMMERI DE TIZI-OUZOU

FACULTE DE GENIE ELECTRIQUE ET D' INFORMATIQUE
DEPARTEMENT D'ELECTRONIQUE

Mémoire de Fin d'Etude de MASTER PROFESSIONNEL

Spécialité : **Electronique Industrielle**
Filière : **Génie électrique**

Présenté par
Djebbari Aissa
Amir Abdelhak

Thème

Conception et réalisation d'un tapis roulant commandé par un système de détection à base d'une carte Arduino

Mémoire soutenu publiquement le 21 septembre 2017 devant le jury composé de :

Mr ATTAF
Grade, Lieu d'exercice, Président

Mr M. LAZRI
Grade, UMMTO, Encadreur

Mr OUALLOUCHE
Grade, Lieu d'exercice, Examinateur

Mémoire réalisé à :FGEI , UMMTO Tizi Ouzou

Remercîments

Nous tenons à remercier notre Dieu, le tout puissant, de nous avoir donné la santé et la volonté pour compléter ce modeste travail.

Nous tenons à remercier notre promoteur, Monsieur LAZRI, pour sa disponibilité, son orientation et son soutien moral.

Nos profondes gratitudes aux membres de jury qui ont l'honneur d'évaluer ce travail de fin d'études.

Tous nos infinis remerciements vont à tous les enseignants qui ont contribué à notre formation durant notre cursus universitaire.

Nos remerciements les plus chaleureux vont à nos chers parents pour leurs encouragements, leur patience et leur grand soutien durant toutes ces années d'études.

Enfin, nous remerciements pour tous ceux qui ont participé de prés ou de loin à la réalisation de ce mémoire.

Dédicaces

Je dédie ce modeste travail :

A Mes très chers parant pour leur bienveillance et leur soutient tout au long de Mon parcours scolaire et universitaire.

A mes amis.

AMIR ABDELHAK

Dédicaces

Je dédie ce modeste travail

A mes chers parents pour leur bienveillance et leurs soutiens tout au long de mon parcours scolaire et universitaire.

A mes très chers frères et sœurs.

A ma grand-mère « yema ouardia »

A tous mes amis et proches.

A mon ami avec qui j'ai partagé ce modeste travail.

A samir Rabahallah pour son soutien.

Aissa

Table des matières

Liste des figures

Liste des figures

INTRODUCTION GÉNÉRALE

Introduction générale :

Le monde a connu au cours du XVIIIe siècle une renaissance scientifique globale, En conséquence la diversification des recherches et d'expériences, à inclure diverses branches de la science a débouchée sur des inventions et des découvertes importantes, qui est la cause directe de la révolution industrielle au XIXe siècle, une révolution qui a eu un impact sur la vie économique, sociale et politique des nations. [1]

A partir de la révolution industrielle et le progrès technologique, l'homme a pu arriver à l'industrialisation et le développement des moyens de transport et d'en faciliter l'usage.

Les tapis roulants (convoyeurs) ont été utilisés pendant des décennies dans le transfert de la majeure partie des marchandises, et ont fait leurs preuves partout parce que les tapis roulant à bandes peuvent être adaptés à presque toutes les conditions locales. La demande d'utilisation de la technologie de convoyeur à bande a augmenté plus que jamais, ce qui a conduit à l'accélération dans son développement pour mettre en œuvre les nouvelles réglementations en particulier en ce qui concerne leur impact sur le transfert des produits. [1]

Les tapis roulant jouent généralement un rôle clé et efficace pour surmonter les difficultés de transfert de divers matières premières et de produits finis, en plus, tous les différents problèmes de transfert par convoyeur doivent être planifiées et organisées par des concepteurs spécialisés afin d'atteindre un transfert économique optimal de façon que toutes les conditions soient remplies. [1]

Notre travail a pour but la conception et la réalisation d'un tapis roulant qui peut aussi appeler convoyeur avec la possibilité de le contrôlé par un système de détection. Le tapis roulant (convoyeur) sera basé sur une carte Arduino Uno comme composant principal ainsi que différents organes de perception (capteurs), de locomotion (moteur), ainsi qu'un moyen de faire les tris (servomoteur).

Notre travail se présente comme suit :

Dans le premier chapitre nous allons présenter des généralités sur la carte Arduino ainsi le model de la carte que on à utilisés (carte Arduino UNO).

Dans le second chapitre nous allons voir les capteurs et les déférents composants nécessaires pour concevoir notre tapis roulant.

En dernier chapitre on va démontrer la réalisation et la conception d'un tapis roulant ainsi leur étape de programmation. Et nous terminons notre travail par une conclusion générale.

CHAPITRE I :

GÉNÉRALITÉS SUR LA CARTE ARDUINO

I.1 Introduction

Le modèle UNO de la société ARDUINO est une carte électronique dont le cœur est un Microcontrôleur ATMEL de référence ATMega328. Le microcontrôleur ATMega328 est un Microcontrôleur 8bits de (la famille AVR) dont la programmation peut être réalisée en langage C.
L'intérêt principal des cartes ARDUINO (d'autres modèles existent) est leur facilité de mise en Œuvre. ARDUINO fournit un environnement de développement s'appuyant sur des outils open source. Le chargement du programme dans la mémoire du microcontrôleur se fait de façon très simple par port USB.

En outre, des bibliothèques de fonctions "clé en main" sont également fournies pour l'exploitation d'entrées-sorties courantes : gestion des E/S TOR, gestion des convertisseurs ADC, génération de signaux PWM, exploitation de servomoteurs ...
L'objectif de ce document est de mettre en évidence certaines informations techniques concernant l'exploitation des périphériques intégrés, en particulier lorsqu'on n'utilise pas les fonctions "clé en main" d'ARDUINO, dans l'objectif de comprendre comment ça marche.

I.2 Historique

Arduino est un projet issu d'une équipe de développeurs composée d'enseignants qui s'appelle Banzi et d'étudiants de l'école de Desing à Ivrea en Italie en hiver 2005.
Les étudiants qui se plaignent envers leur enseignant appelé Banzi, de ne pas avoir accès à des solutions bas prix pour accomplir leurs projets de robotique. Par la suite, l'enseignant Banzi a contacté David Cuartielles, qui est un ingénieur Espagnole spécialisé dans les microcontrôleurs pour créer leur propre carte en embarquant dans leur histoire un des étudiants de Banni, David Mellis qui sera chargé de créer le langage de programmation allant avec la carte. En deux jours David écrira le code ! Trois jours de plus la carte était créée. Ils décidèrent de l'appeler Arduino.

I.3 Définition du module Arduino

Le module Arduino est un circuit imprimé en matériel libre (plateforme de contrôle) dont les plans de la carte elle-même sont publiés en licence libre dont certains composants de la carte : comme le microcontrôleur et les composants complémentaires qui ne sont pas en licence

libre. Un microcontrôleur programmé peut analyser et produire des signaux électriques de manière à effectuer des tâches très diverses. Arduino est utilisé dans beaucoup d'applications comme l'électrotechnique industrielle et embarquée ; le modélisme, la domotique mais aussi dans des domaines différents comme l'art contemporain et le pilotage d'un robot, commande des moteurs et faire des jeux de lumières, communiquer avec l'ordinateur, commander des appareils mobiles (modélisme). Chaque module d'Arduino possède un régulateur de tension +5 V et un oscillateur à quartez 16 MHz (ou un résonateur céramique dans certains modèles). Pour programmer cette carte, on utilise l'logiciel IDE Arduino. [2],[3].

I.4 Caractéristique de la carte Arduino

Parmi les caractéristiques les plus importantes de la carte Arduino, nous avons retenu les suivantes :

- Pas cher : les cartes Arduino sont relativement peu couteuses comparativement aux autres plateformes.
- Multiplateforme : le logiciel Arduino, écrit en java, tourne sous les systèmes d'exploitation Windows, Macintosh et Linux. La plupart des systèmes à microcontrôleurs sont limités à Windows.
- Un environnement de programmation claire et simple : l'environnement de programmation Arduino est facile a utilisé pour les débutants, tous en étant assez flexible pour que les utilisateurs avancés puissent en tirer profit également.
- Logiciel open source est extensible : le logiciel Arduino et le langage Arduino sont publiés sous License open source, qui est disponible et peut être complétée par des programmateurs expérimentés.
- Matériel open source est extensible : les cartes Arduino sont basées sur les microcontrôleurs Atmel ATMEGA2560, ATMEGA 168, ATMEGA 328, etc. les schémas des modules sont publiés sous une licence créative Commons, et le concepteur de circuits expérimentés peuvent réaliser leurs propres versions des cartes Arduino, en les complétant et en les améliorant.
- La carte Arduino est capable de stocker un programme et de le faire exécuter.
- La carte reçoit des informations analogiques ou numériques sur ces entrées et génère des informations analogiques ou numériques.

- Le microcontrôleur traitera ces informations et les transmettra vers les sorties numériques.
- Récupère les données des capteurs en vue de les transmettre à l'interface de commande (pc) et traduire les instructions pour faire fonctionner les actionneurs.

I.5 Architecture de la carte

La figure suivante illustre le schéma synoptique de la carte Arduino.

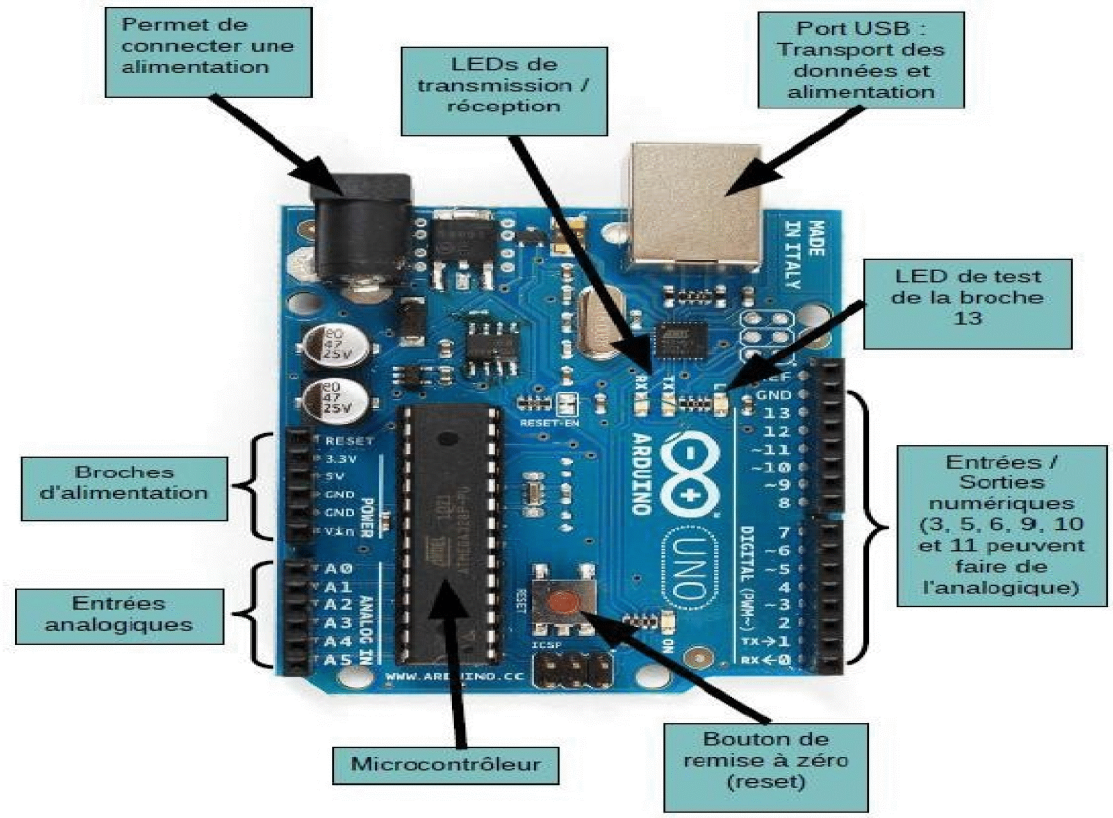


Figure I.1 : Architecture de la carte Arduino UNO

I.6 Présentation matériel de la plateforme Arduino

L'Arduino est composée de deux parties indissociables : La carte qui est la partie hardware avec laquelle on travaille en construisant chaque projet et la plateforme IDE ARDUINO qui est la partie logicielle fonctionnant sur le PC celle-ci permet de mettre au point et de transférer le programme qui sera par la suite exécutée par la carte ARUINO.

I.6.1 Le Microcontrôleur

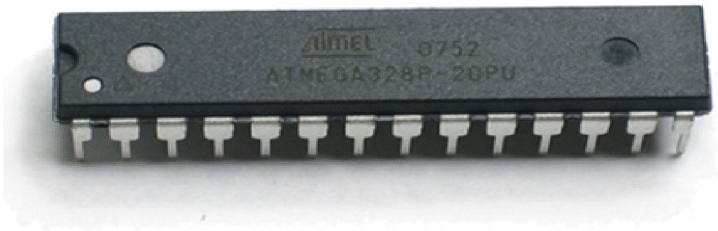


Figure I.2 : microcontrôleur ATMEGA328

I.6.2 Caractéristiques du microcontrôleur ATMEGA 328

- . C'est un microcontrôleur ATMEL de la famille AVR 8bits.
- . FLASH = mémoire programme de 32Ko.
- . SRAM = données (volatiles) 2Ko.
- . EEPROM = données (non volatiles) 1Ko.
- . Digital I/O (entrées-sorties Tout Ou Rien) = 3 ports Port, PortC, PortD (soit 23 broches en tout I/O).
- . Timers/Counters : Timer0 et Timer2 (comptage 8 bits), Timer1 (comptage 16bits) Chaque timer peut être utilisé pour générer deux signaux PWM. (6 broches OCxA/OCxB).
- . Plusieurs broches multifonctions : certaines broches peuvent avoir plusieurs fonctions différentes choisies par programmation.

La figure ci-dessous montre la structure interne du microntroleur ATMEGA328

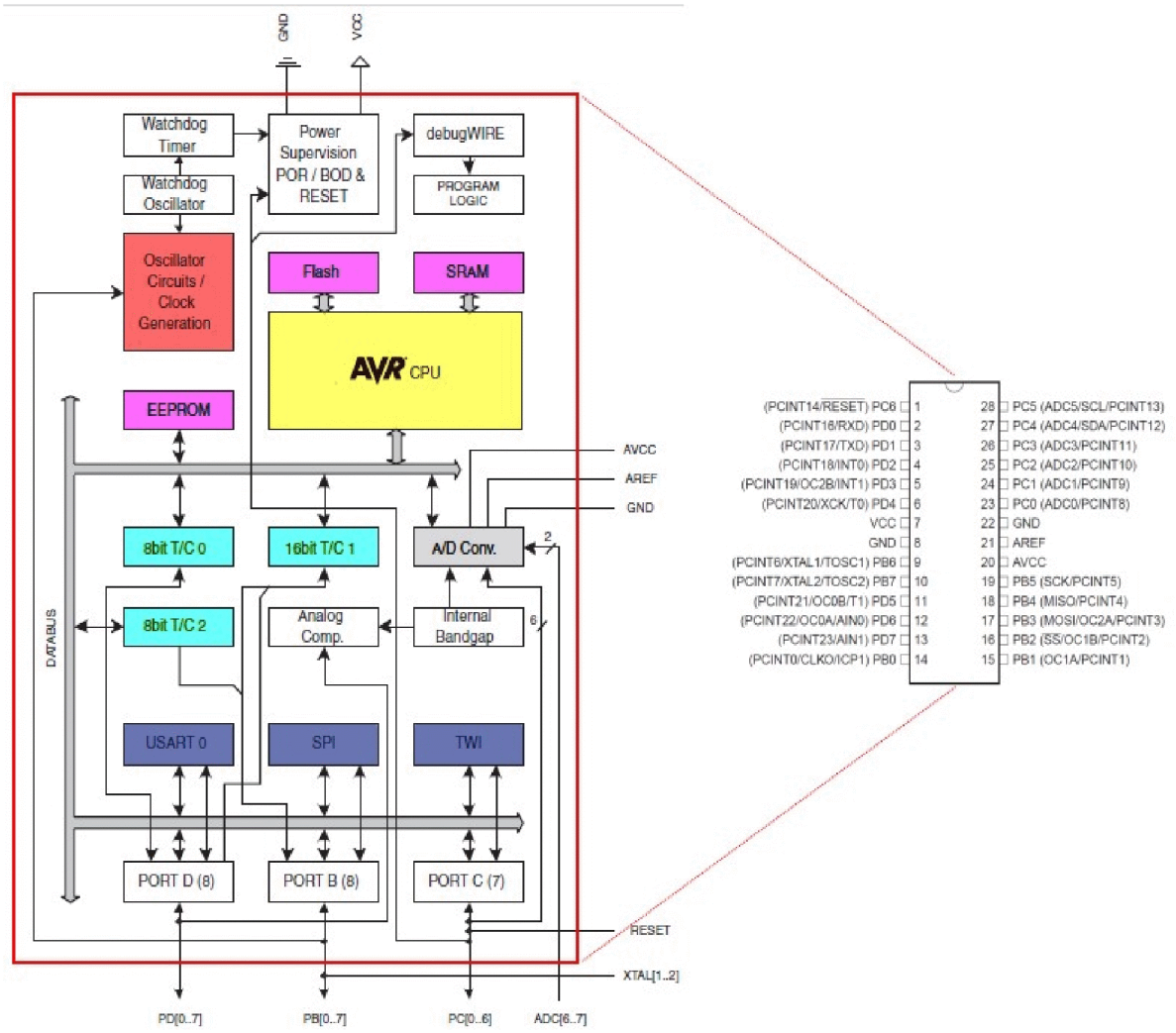


Figure I.3 : Architecture du microcontrôleur ATMEGA328

I.6.2 Les entrées-sorties de la carte

La carte Arduino possède 14 entrées-sorties numériques (pattes 0-13), et 6 entrées analogiques (A0-A5).

I.6.2.1 Les 14 entrées sorties numériques (pattes 0_13)

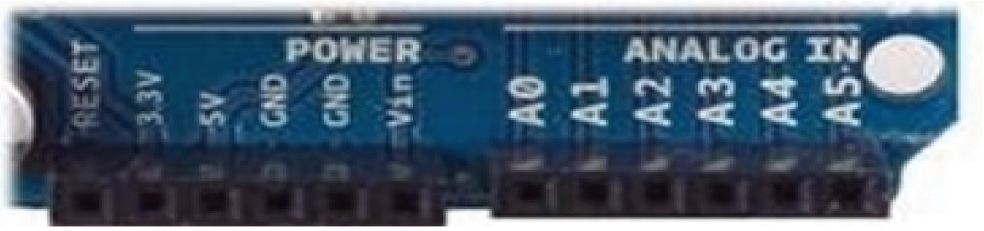
Elles peuvent être configurées en entrée ou en sortie en les indiquant dans le programme.

Côtés entrés, des capteurs qui collectent des informations sur l'environnement comme la variation de température via une sonde thermique, le contact via un bouton-poussoir.

Côtés sortis, des actionneurs qui agissent sur le monde physique telle une petite lampe qui produit de la lumière, un moteur qui actionne un bras articulé…etc.

Chaque broche peut fournir ou recevoir un maximum de 40mA d'intensité. Certaines broches ont des fonctions spéciales.

I.6.2.2 Les entrées analogiques (pattes A0_A5)



Ces pattes sont réservées à la mesure de signaux analogiques et permettent de convertir une tension analogique Ve de 0 à 5 V en une valeur numérique.

Pour pouvoir être traitées par le microcontrôleur, ces entrées analogiques sont prises en charge par un CAN (Convertisseur Analogique Numérique ou ADC pour Analog Digital Converter) dont le rôle est de convertir l'échantillon de tension VE en une grandeur numérique binaire sur n bits.

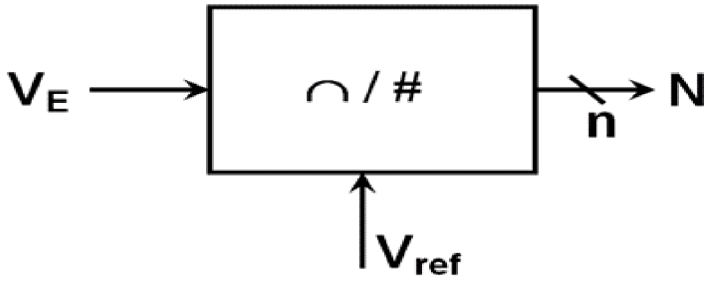


Figure I.4 : Convertisseur analogique numérique

Le principe de la conversion Analogique-Numérique est représenté ci-dessous (avec n=3 bits et la tension de référence Vref=5 Volts) :

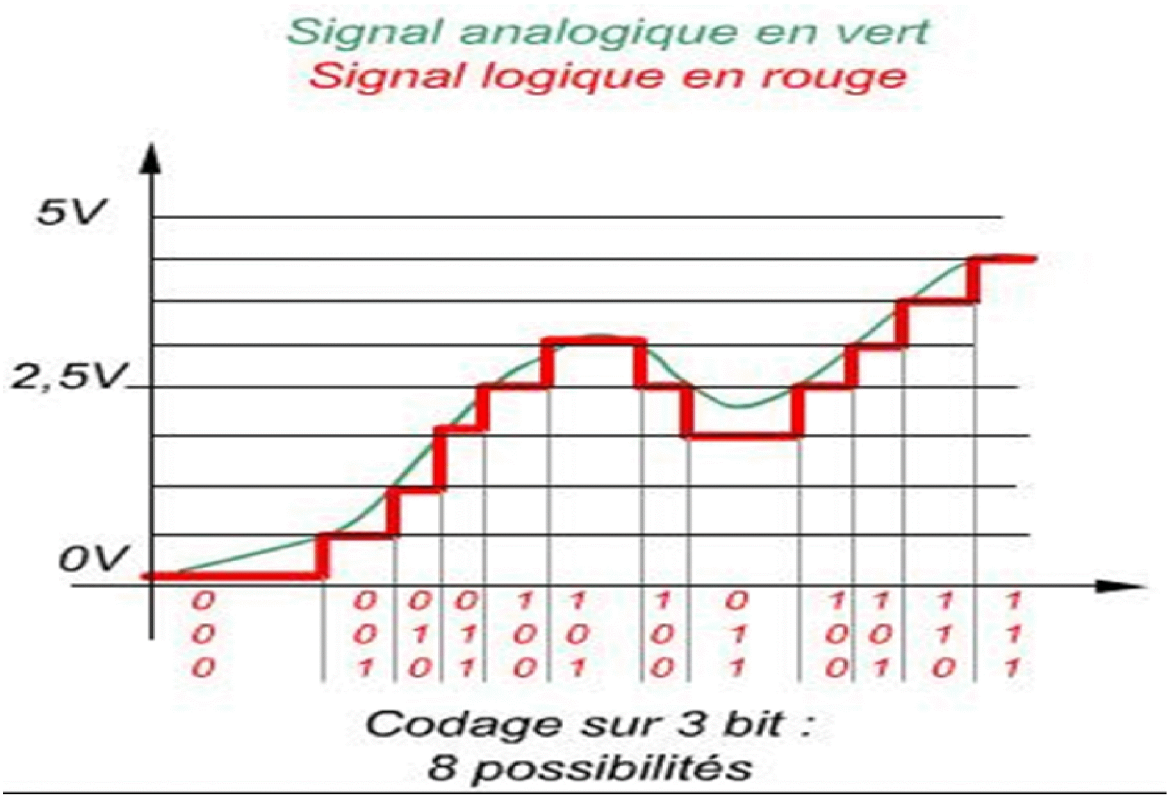


Figure I.5 : Graphe de conversion ADL

I.6.3 Alimentation de la carte

La carte Arduino peut être alimentée via un port qui l'alimente avec +5V. La norme USB limite à 500mA maximum l'alimentation de la carte. Cela suffit pour des leds mais est généralement insuffisante pour des moteurs ou des servomoteurs (avec cette alimentation seule, lors de la mise en route d'un moteur, cela peut entrainer une chute de tension).

La carte peut aussi fonctionner avec une alimentation externe de 6 à 20 volts. Cependant, si la carte est alimentée avec moins de 7V la broche de 5V pourrait fournir moins de 5V et la carte risque d'être instable. Si on utilise plus de 12V, le régulateur de tension de la carte pourrait surchauffer et endommager la carte. Aussi, la plage idéale recommandée pour alimenter la carte Uno est entre 7V et 12V.

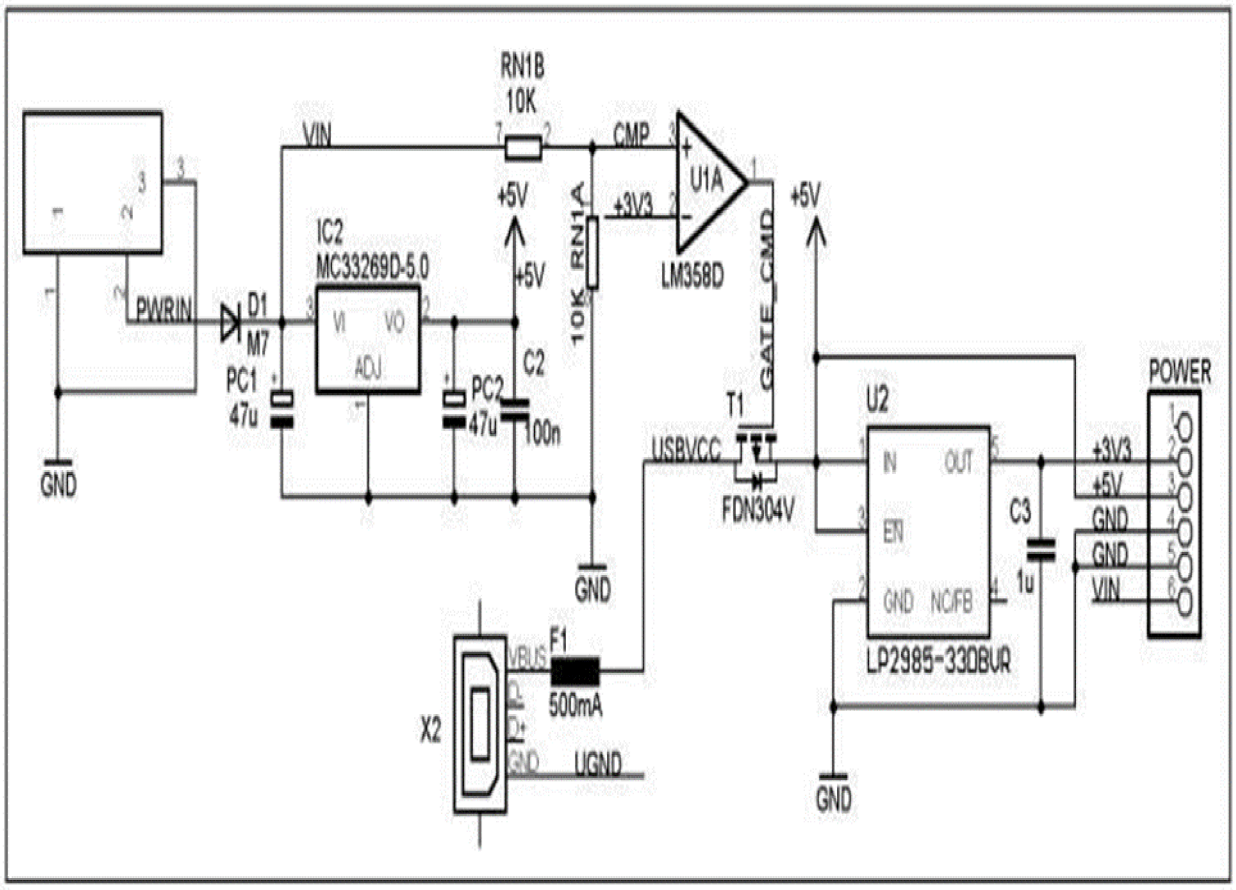


Figure I.6: schéma électrique de la carte

La carte peut être alimentée aussi par les broches V_{in} (+) GND (-) qui peuvent aussi supporter une tension comprise entre 7et 12volt (tension convertie aussi en 5v par l'Arduino).

I.6.4 Visualisation

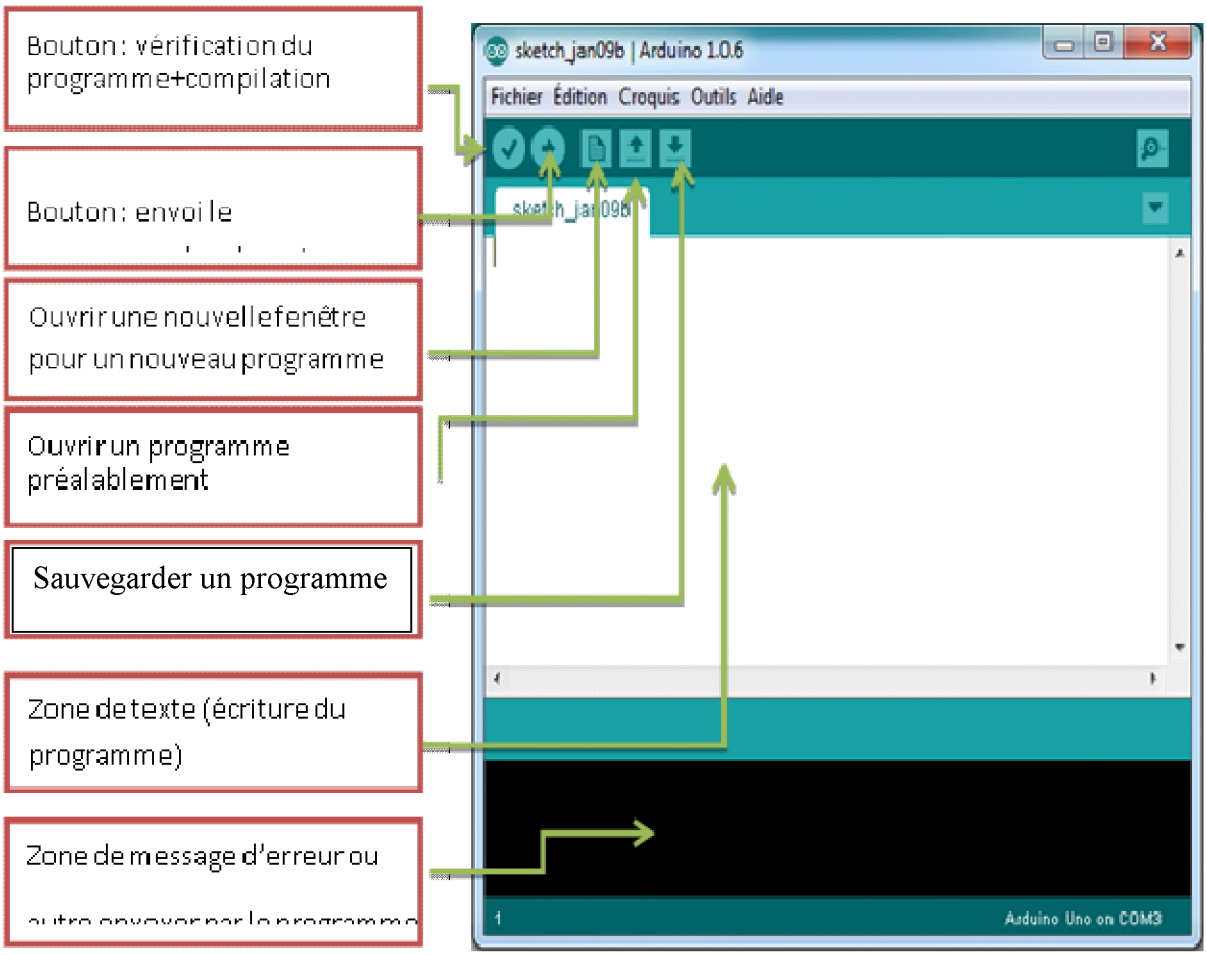
LED verte : alimentation

LED jaune ou verte : active lors de téléchargement du programme (LED de Transmission réception)

LED jaune : de teste

Reset : pour la réinitialisation du microcontrôleur.

I.7 Architecture SOFTWARE de la carte



- **Choisir la carte que l'on va programmer**

Le nom de notre carte est indiqué sur elle. Pour nous, il s'agit de la carte "Mega328". Allez dans le menu, « Tools » (outils) puis dans « Board » (carte). Vérifiez que c'est bien le nom « Arduino Mega328 » qui est coché.

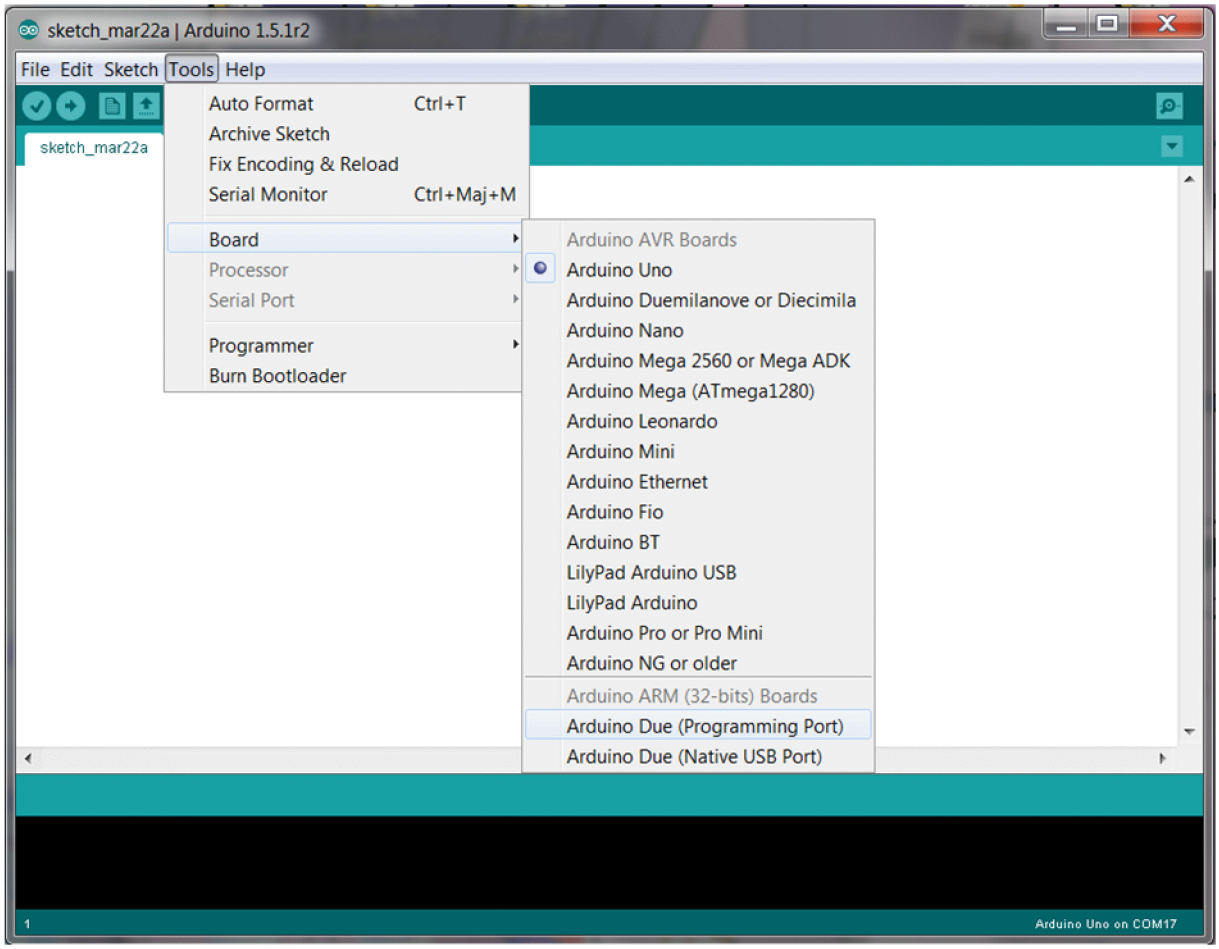


Figure I.7 : choix de la carte Arduino

- **Choisir le port de connexion de la carte**

Nous allons dans le menu Tools, puis Serial port. La, nous choisissons le port COMX, X étant Le numéro de port qui est affiché. Ne choisissons pas COM1 car il n'est quasiment jamais connecté à la carte. Dans notre cas il s'agit de COM18.

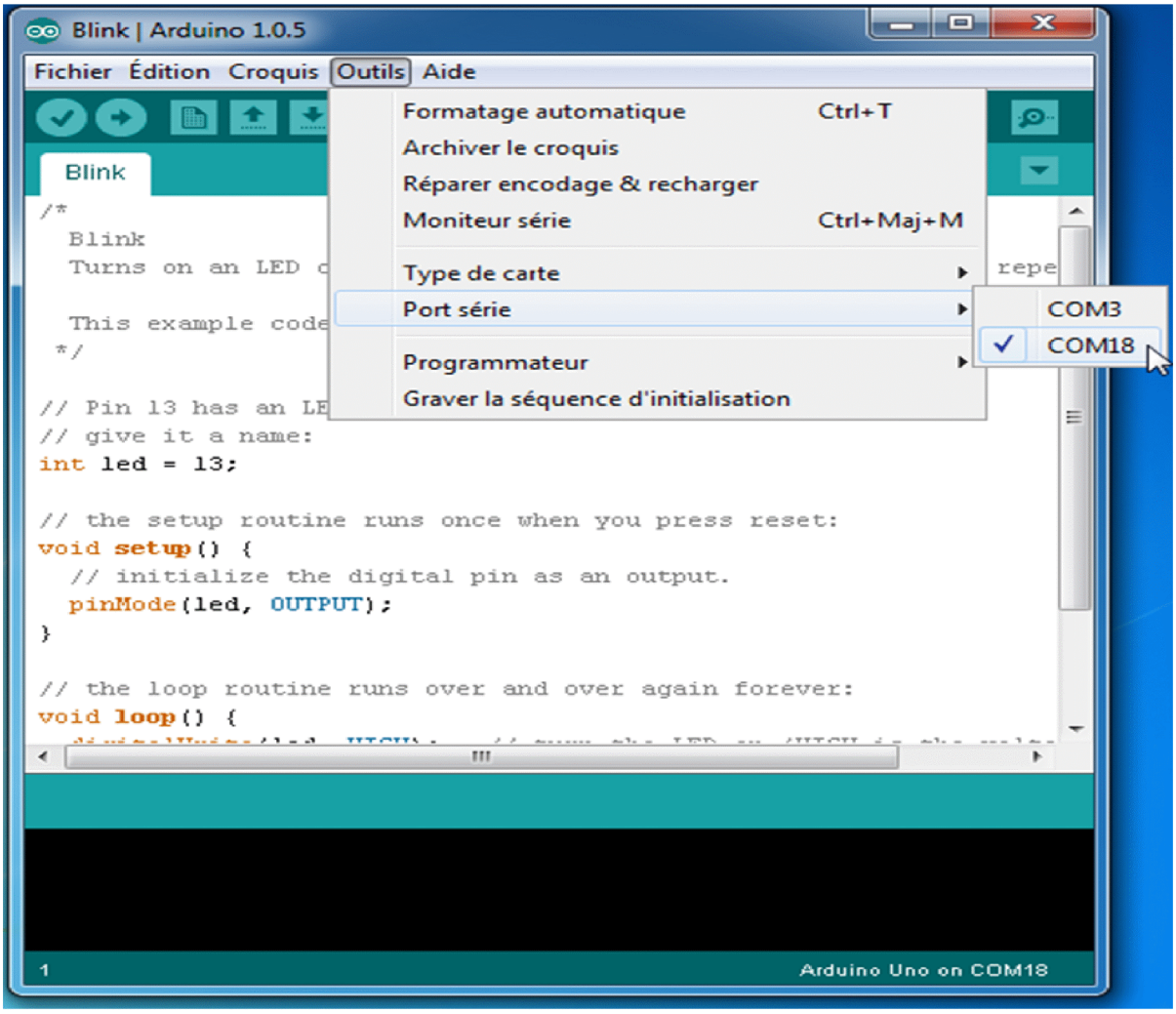


Figure I.8 : choix du port de connexion de la carte

Prochaine étape, il va falloir envoyer le programme dans la carte. Pour ce faire, il suffit de cliquer sur le bouton Upload ou « Télécharger « en français.

Finalement, le processus de rédaction du programme jusqu'à son téléchargement dans la carte peut être résumé grâce au schéma suivant :

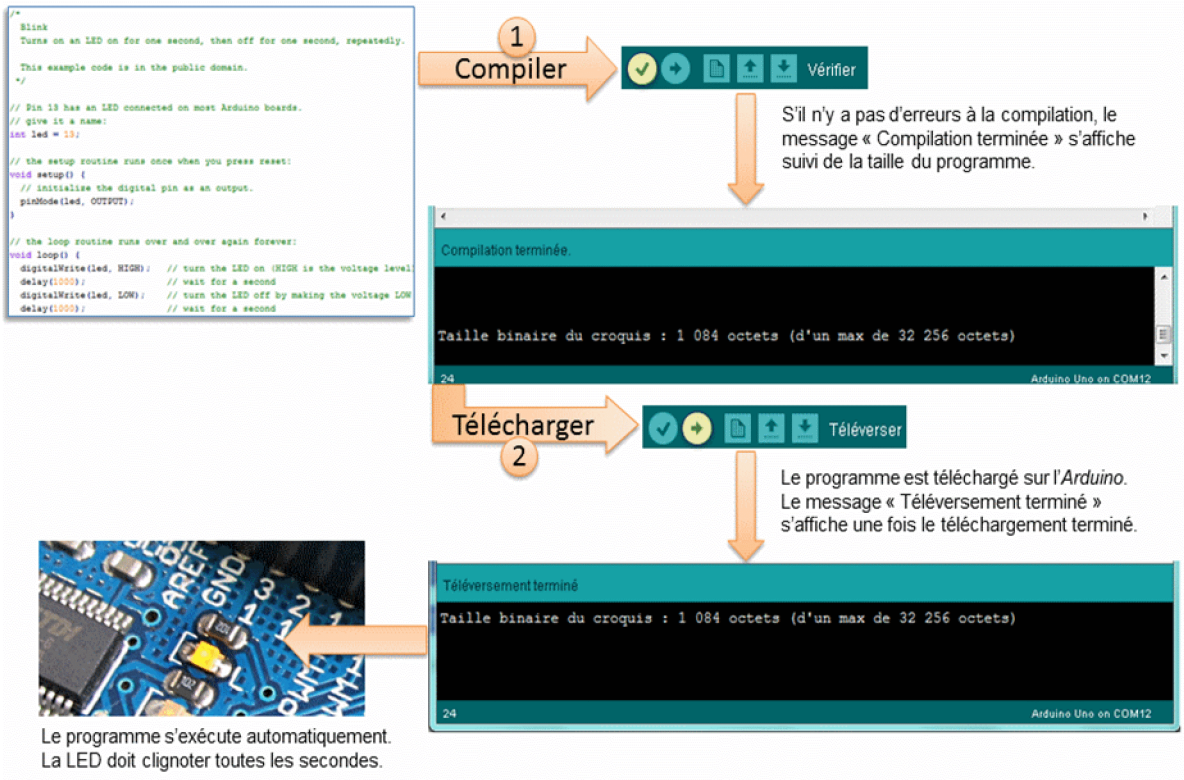


Figure I.9 : résumé du processus de rédaction de programme

I.7.1 Le logiciel

L'environnement de programmation Arduino (IDE en anglais) est une application écrite en Java inspirée du langage Processing.

L'IDE permet d'écrire, de modifier un programme ou de le convertir en une série d'instructions compréhensibles pour la carte.

C'est un logiciel de programmation par code, qui contient une cinquantaine de commandes différentes [4].

I.7.1.1 L'interface logicielle

Sur un ordinateur, le logiciel de programmation de la carte Arduino sert d'éditeur de code (langage proche de C), une fois le programme tapé ou modifier au clavier, il sera transféré et mémorisé dans la carte à travers la liaison USB, le câble USB alimenté à la fois en énergie le carte et transporte aussi l'information.

I.7.1.2 Structure du logiciel

Le logiciel open source (Windows, linux, ou Mac) représenté par la figure ci-dessous fournit avec l'Arduino est un éditeur de texte qui permet :

- De programmer la carte en utilisant un langage simple proche du C.
- De communiquer avec la carte grâce au « terminal série » (faim apparaitre des informations de la carte sur l'écran de l'ordinateur).

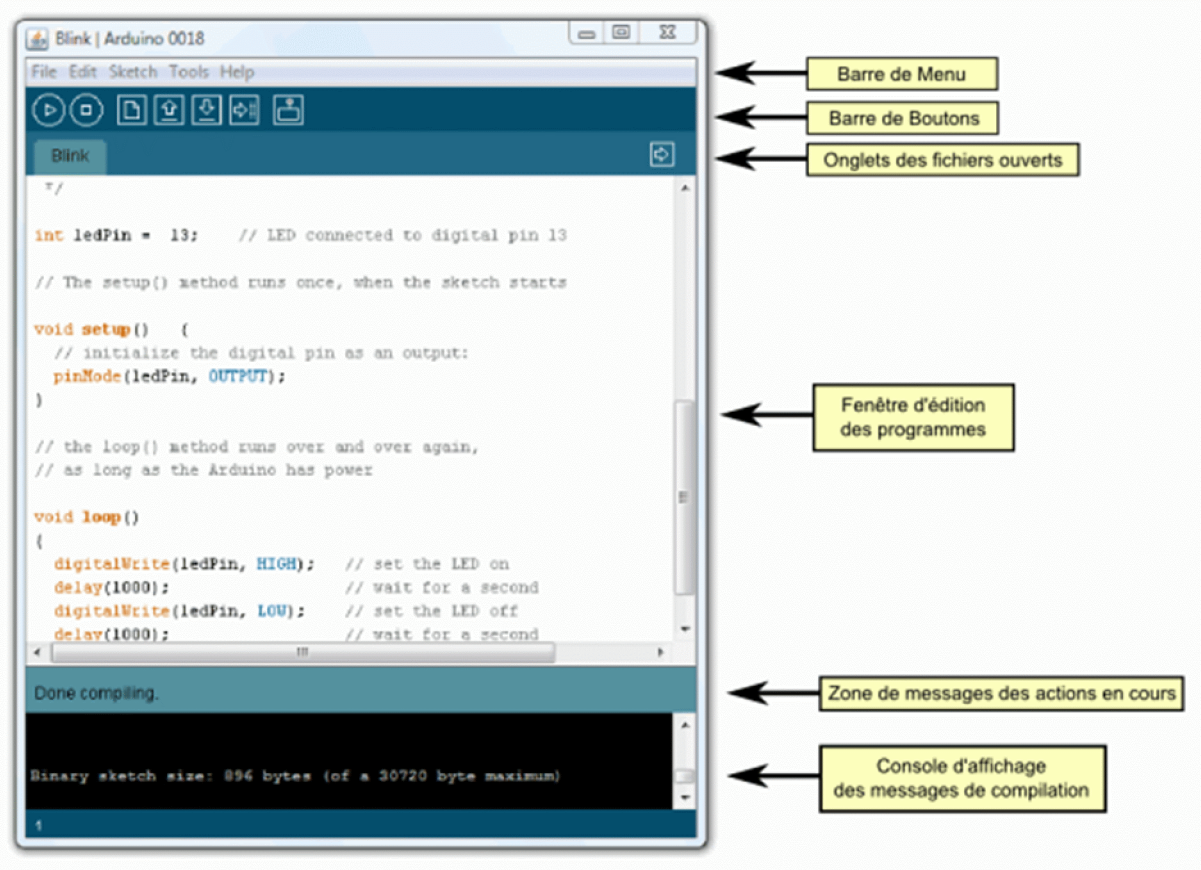
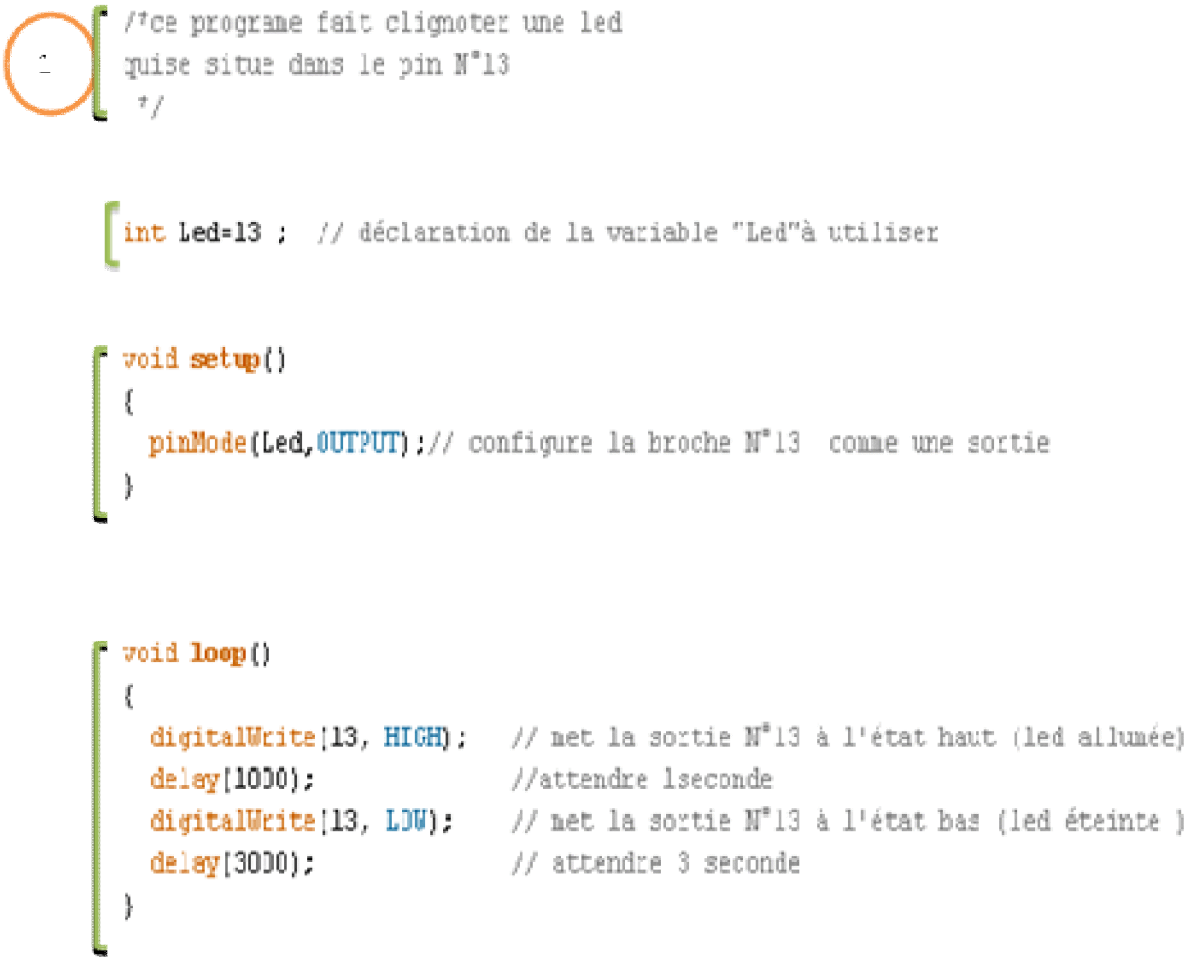


Figure I.10 : structure de logiciel

I.7.1.3 Structure du programme

Un programme utilisateur Arduino est une suite d'instructions élémentaires et séquentielles, dans tous les programmes d'Arduino Il y a trois étapes. Pour mettre en évidence ces phases on va étudier la structure d'un exemple simple de programmation qui consiste à allumer une LED pendant 1 seconde puis l'éteindre durant 3 seconde sur la broche N°13 ainsi de suite jusqu'à introduire un nouveau programme ou bien arrêté l'alimentation.



```
/*ce programe fait clignoter une led
puise situe dans le pin N°13
 */

int Led=13 ;  // déclaration de la variable "Led"à utiliser

void setup()
{
  pinMode(Led,OUTPUT);// configure la broche N°13  comme une sortie
}

void loop()
{
  digitalWrite(13, HIGH);   // met la sortie N°13 à l'état haut (led allumée)
  delay(1000);              //attendre 1seconde
  digitalWrite(13, LOW);    // met la sortie N°13 à l'état bas (led éteinte )
  delay(3000);              // attendre 3 seconde
}
```

- **Commentaire**

Pour écrire des commentaires sur le programme (ça aide à la relecture du programme et sa Compréhension par une personne autre que celle qui l'a fait) on a 2 possibilités : Soit en multi ligne, en les mettant entre les signes /** **/.

Soit sur une ligne de code en les séparant du code avec //.

- **Définition et déclaration de variable**

Pour notre montage, on va utiliser une sortie numérique de la carte qui est par exemple la 13 - ème sortie numérique. Cette variable doit être définie et nommée ici : On lui donne un nom arbitraire Led.

- **Configuration des entrées-sorties Voidsetup ()**

Les broches numériques de l'Arduino peuvent aussi bien être configurées en entrées numériques ou en Sorties numériques. Ici on va configurer Broche LED en sortie PinMode

(nom, état) est une des quatre fonctions relatives aux entrées-sorties numériques.

- **Programmation des interactions voidloop ()**

Dans cette boucle, on définit les opérations à effectuer, dans l'ordre :

DigitalWrite (nom, état) : Une autre des quatre fonctions relatives aux entrées-sorties numériques.

- DeLay (temps en millisecondes) : La commande d'attente entre deux autres instructions
- Chaque ligne d'instruction est terminée par un point-virgule.
- Ne pas oublier les accolades, qui encadrent la boucle.

- **Syntaxe du langage Arduino**

Explications de chaque commande de la syntaxe Arduino dont voici la table des matières. Chaque instruction est suivie de sa traduction entre-parenthèses.

Commandes de structure du programme	
Structure générale	• Voidsetup () (configuration-préparation) • Voidloop () (exécution)
Contrôle et conditions	• if (si...) • If...else (si...alors...) • For (pour...) • Switch case (dans le cas où...)
Opérations de comparaison	• == (équivalent à) • ! = (différent de) • < (inférieur à) • > (supérieur à) • <= (inférieur ou égal à) • >= (supérieur ou égal à)
Operations booléennes	• &&(et) • || (où) • ! (et pas)

Autres commandes	• // (commentaire simple ligne) • /* */ (commentaire multilignes) • #define (donner une valeur à un nom)

➢ **Définition des variables**

Pour composer un programme, il est nécessaire de définir toutes les composantes d'entrée et de sortie qui vont affecter le montage matériel et les calculs à effectuer. Chaque entrée et chaque sortie sera une variable.

Le code ci-dessous déclare la variable capteur1, puis lui affecte (par exemple) le numéro de l'entrée analogique numéro 0. (L'Arduino possède 6 entrées analogiques numérotées de 0 à 5) :

```
Int capteur1 = 0 ; // déclaration de la
variable identifiant le port analogique 0
de la carte
```

La ligne ci-dessous *déclare* la variable LED1, puis lui affecte (par exemple) le numéro de la sortie numérique numéro 13. (L'Arduino possède 13 entrées ou sorties numériques numérotées de 1 à 13) :

```
IntLED1 = 13 ; // déclaration de la variable
identifiant l'autre port numérique 13 de la
carte
```

La ligne suivante *déclare* la variable qui correspond à la valeur de luminosité envoyée par le capteur.

De plus, sa première valeur à l'allumage de la carte Sera 0 :

```
Intlum1 = 0 ; // déclaration de la variable
identifiant la valeur de la luminosité du
capteur 1
```

Nos trois variables sont maintenant déclarées et définies, passons à la configuration des entrées-sorties de la carte.

I.7.1.4 Configuration logicielle du matériel

Pour ce montage, il n'y a qu'une broche à configurer : La broche numérique sur laquelle on va brancher la LED (car elle pourrait être aussi bien configurée en sortie ou en entrée).

Ici, on va configurer cette broche numérique en sortie, car la LED est un actionneur. La broche d'entrée analogique pour le capteur n'est pas à configurer, car la carte Arduino possède 6 entrées analogiques après le voidsetup (), qui précise qu'on est à l'étape de configuration, on définit donc l'état de la broche 13 :

```
Voidsetup ()
{
PinMode (LED1, OUTPUT) ; // configure la broche 13 comme une sortie
}
```

Et on ferme la phase de configuration par une accolade

I.7.1.5 Programmation des interactions

On indique maintenant qu'on crée une boucle avec voidloop () {

Puis on effectue la première opération : Lire la valeur du capteur = lire la variable lum1 identifiant la valeur de luminosité

```
lum1 = analogRead(capteur1) ;
```

On peut maintenant allumer la LED digitalWrite (LED1, HIGH) ;

On patiente un certain temps : en fonction de la valeur de la variable luminosité lum1

```
DeLay(lum1) ;
```

On peut maintenant éteindre la LED digitalWrite (LED1, LOW) ;

On patiente un certain temps : en fonction de la valeur de la variable luminosité lum1

```
DeLay(lum1) ;
```

On peut maintenant boucler avec une accolade, c'est-à-dire faire remonter automatiquement au début de la boucle pour lire la nouvelle valeur du capteur et ainsi de suite... Jusqu'à ce qu'on éteigne l'Arduino.

I.8 Conclusion

Dans ce chapitre, nous avons projeté la lumière sur une carte d'acquisition qui est l'Arduino donnant ainsi les raisons pour lesquelles on l'a choisie, puis nous avons cité des différents types de cette dernière. Ensuite, nous avons expliqué les deux parties essentielles de l'Arduino ; (la partie matérielle et la partie de programmation) plus précisément. Nous avons également expliqué le principe de fonctionnement de la carte Arduino sans oublier ses caractéristiques.

CHAPITRE II :

LES DIFFERENTS CAPTEURS ET MOTEURS

II.1 Introduction

Dans le monde industriel, un capteur est un dispositif qui transforme une grandeur physique observée (température, pression, etc.) en une grandeur utilisable (intensité électrique). Grâce aux avancées technologiques de ces dernières années, principalement dans le domaine de la miniaturisation, les capteurs sont devenus des éléments de très petite taille. Ils sont ainsi dotés de moyens leur permettant: de stocker les résultats de leurs observations, d'effectuer un certain nombre de traitement sur ces résultats et de les transmettre au monde réel via une communication sans fil ou filaire. Toutes ces caractéristiques apportent au monde scientifique l'opportunité de déployer les capteurs à grande échelle dans des milieux, parfois difficiles voire impossible d'y avoir l'accès, pour surveiller, collecter et transmettre les données collectées à un système capable de les analyser et prendre (si possible ou nécessaire) les décisions. Ce déploiement aboutit à un réseau de capteurs.

Dans la suite de ce chapitre, nous verrons les différents capteurs qu'on a utilisés .nous commencerons par donner quelques définitions d'un capteur, et leurs classifications, ainsi que leurs principes physiques…. .

II.2 Les capteurs

Avant de donner la définition d'un capteur, il est nécessaire de connaitre quelques définitions de métrologie.

Le mesurande : c'est l'objet de la mesure ou plus simplement la grandeur à mesurer. Le mesurage : c'est l'ensemble des opérations pour déterminer la valeur du mesurande.la mesure c'est le résultat du mesurage. Autrement dit c'est la valeur du mesurande.

II.2.1 Définition d'un capteur

Un capteur est un dispositif convertissant une grandeur physique analogique (pression, température, déplacement, débit,...) en un signal analogique rendu transmissible et exploitable par un système de conditionnement (courant électrique, radiation lumineuse, radiofréquence). Le capteur est la partie d'une chaine de mesure qui se trouve au contact direct du mesurande.

Dans l'immense majorité des cas, le signal de sortie est électrique en raison de la facilité de transmission de l'information sous cette forme (câblage), même si les signaux optiques transmis par fibre sont de plus en plus fréquents. La tension ou l'intensité de ce signal est alors l'image de mesurande par une loi continue qu'on souhaite idéalement linéaire ou affine,

telle que :

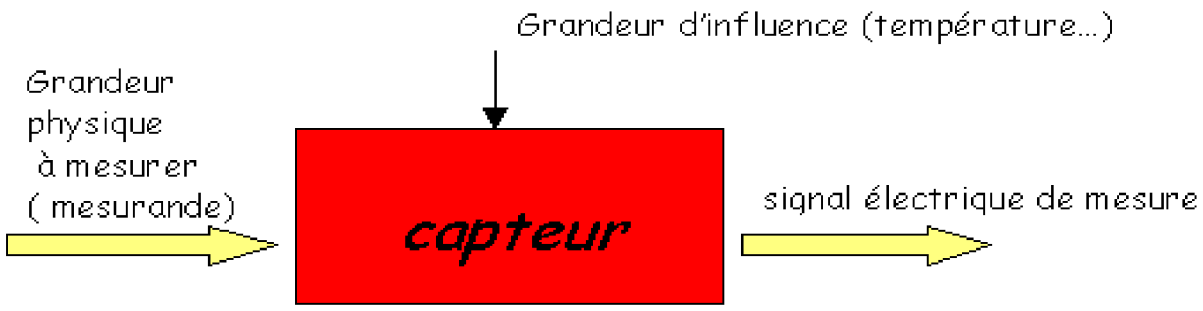


Figure II.1 : Chaine d'action d'un capteur

II.2.2 Chaine de mesure :

Généralement, le signal de sortie de capteur n'est pas directement utilisable. On appelle chaine de mesure l'ensemble des circuits ou appareils qui amplifient, adaptent, convertissent, linéarisent, digitalisent le signal avant sa lecture sur le support de sortie.

Pour obtenir une image d'une grandeur physique, la chaine de mesure peut faite intervenir plusieurs phénomènes différents. Par exemple, la mesure d'un débit peut se faire en plusieurs étapes :

- Transformation du débit en une pression différentielle.
- Transformation de la pression différentielle en la déformation mécanique d'une membrane.
- Transformation de la déformation mécanique en une grandeur électrique (à l'aide d'un piézoélectrique) via un circuit électrique associé.

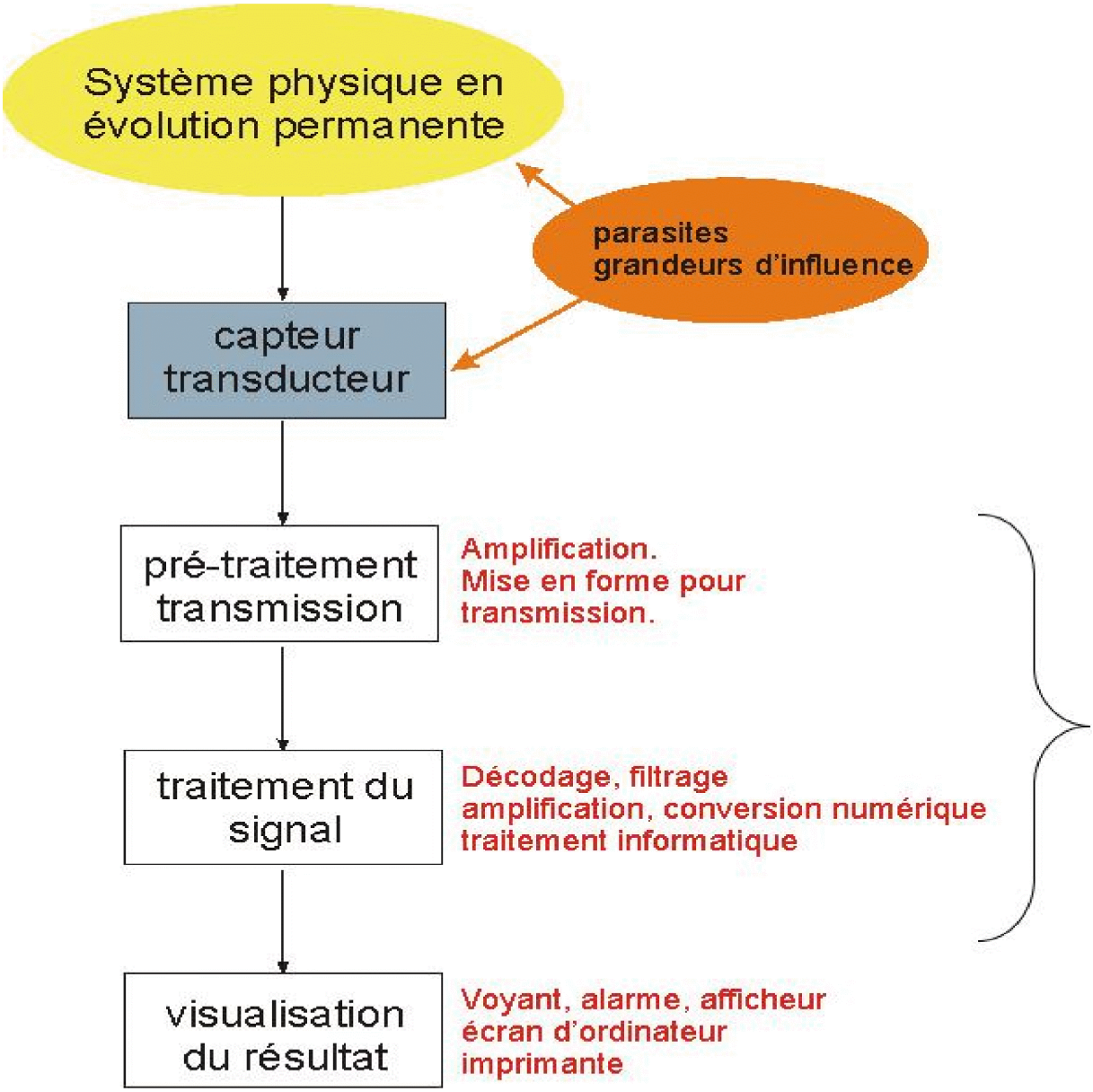


Figure II.2 : Chaine de mesure d'un capteur

II.3 Classification des capteurs

Si l'on s'intéresse aux phénomènes physiques mis en jeux dans les capteurs, on peut classer ces derniers en deux catégories.

- Capteurs actifs
- Capteurs passifs

II.3.1 Capteur actif

Fonctionnant en **générateur**, un capteur actif est généralement fondé dans son principe sur un effet physique qui assure la conversion en énergie électrique de la forme d'énergie propre à la grandeur physique à mesurer (énergie thermique, mécanique ou de rayonnement).

Effet thermoélectrique (ou effet Seebeck) : Un circuit formé de deux conducteurs de nature chimique différente, dont les jonctions sont à des températures T_1 et T_2, est le siège d'une force électromotrice d'origine thermique e (T_1,T_2).

Effet piézo-électrique : L'application d'une contrainte mécanique à certains matériaux dits piézo-électriques (le quartz par exemple) entraîne l'apparition d'une déformation et d'une même charge électrique de signe différent sur les faces opposées.

Effet d'induction électromagnétique : La variation du flux d'induction magnétique dans un circuit électrique induit une tension électrique (détection de passage d'un objet métallique).

Effet photo-électrique : La libération de charges électriques dans la matière sous l'influence d'un rayonnement lumineux ou plus généralement d'une onde électromagnétique.

II.3.2 Capteur passif

Il s'agit en général d'une impédance dont la valeur varie avec la grandeur physique, il faut l'intégrer dans un circuit avec une alimentation.
Exemples : résistance, inductance, thermistance (alerte température), capteur de niveau capacitif, inductance de fin de course.

La variation d'impédance résulte :

• d'une variation de dimension du capteur (capteurs de position, potentiomètre, inductance à noyaux mobile, condensateur à armature mobile).

• d'une déformation résultant d'une force ou d'une grandeur s'y ramenant (pression accélération). Exemples : armature de condensateur soumise à une différence de pression, jauge d'extensomètre liée à une structure déformable.

II.4 Les caractéristiques d'un capteur

Etendue de mesure : Valeurs extrêmes pouvant être mesurée par le capteur.

Résolution : Plus petite variation de grandeur mesurable par le capteur.

Sensibilité : Variation du signal de sortie par rapport à la variation du signal d'entrée.

Précision : Aptitude du capteur à donner une mesure proche de la valeur vraie.

Rapidité : Temps de réaction du capteur. La rapidité est liée à la bande passante.

Linéarité : représente l'écart de sensibilité sur l'étendue de mesure.

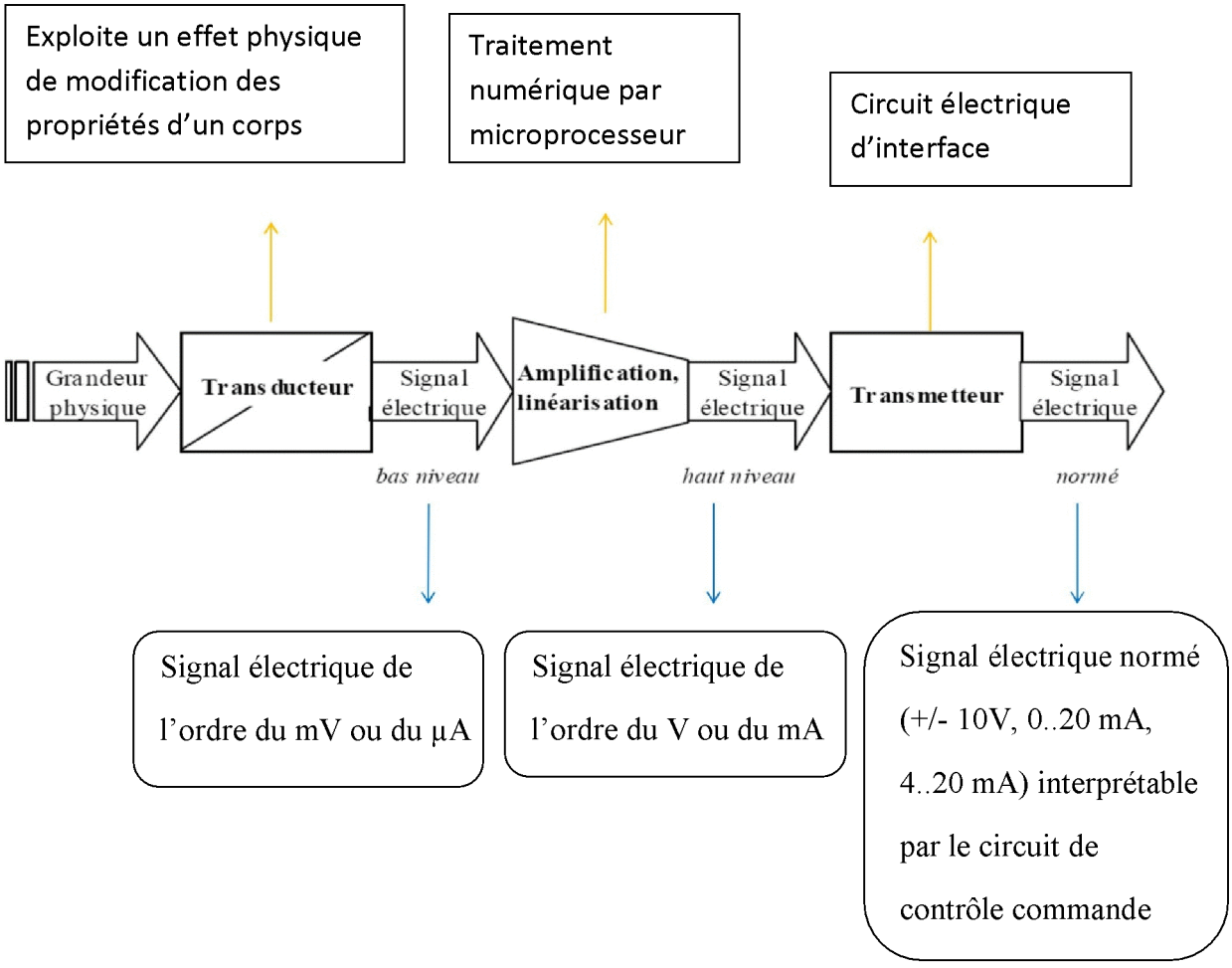


Figure II.3 : Schéma de principe d'un capteur industriel

II.5 Le capteur à ultrason HC-SR04

II.5.1 Définition

Le capteur à ultrason HC-SR04 est un moyen précis pour la mesure de la distance (2 cm à 4m avec une précision de 3 mm).il est très utilisé en robotique et bien d'autre application, pour son efficacité et sa facilité d'interaction avec toute une sorte de microcontrôleurs (Arduino, Microchip PIC, basic stamp, Porpeller chip, etc.)

Le capteur HC-SR04 utilise les ultrasons pour déterminer la distance d'un objet. Il offre une excellente plage de détection sans contacte, avec des mesures de haute précision et stables. Son fonctionnement n'est pas influencé par la lumière de soleil ou des matériaux sombres, bien que les matériaux comme les vêtements puissent être difficiles à détecter.

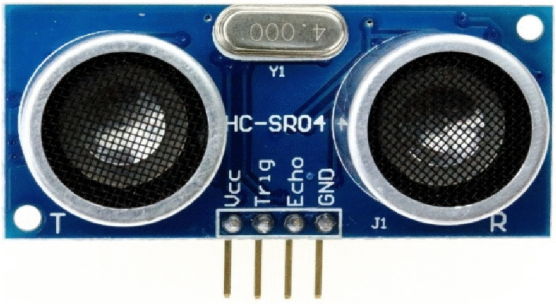


Figure II.4: Capteur à ultrason HC-SR04

II.5.2 Caractéristiques

Le HC-SR04 présente les caractéristiques et spécifications suivantes :

- Dimensions : 45 mm x 20 mm x 15 mm.
- Plage de mesure : 2 cm à 400 cm.
- Résolution de la mesure : 0.3 cm.
- Angle de mesure efficace : 15 °.
- Largeur d'impulsion sur l'entrée de déclenchement : 10 µs.

II.5.3 Broche de connexion

- Vcc = Alimentation +5 V DV

- Trig = Entrée de déclenchement de la mesure (Trigger input)
- Echo = Sortie de mesure donnée en écho (Echo output)
- GND = Masse de l'alimentation

II.5.4 Spécifications et limites

Paramètre	Min	Type	Max	Unité
Tension d'alimentation	4.5	5.0	5.5	V
Courant de repos	1.5	2.0	2.5	mA
Courant de fonctionnement	10	15	20	mA
Fréquence des ultrasons	-	40	-	kHz

Tableau II.5 : spécifications et limites du HC-SR04.

II.5.5 Principe de Fonctionnement

Pour déclencher une mesure, il faut présenter une impulsion « high » (5V) d'au moins 10 µs sur l'entée « Trig ». Le capteur émet alors une série de 8 impulsions ultrasonique à 40 kHz, puis il attend le signal réfléchi.

Lorsque celui-ci est détecté, il envoie un signal « high » sur la sortie « Echo », dont la durée est proportionnelle à la distance mesurée (voire figure II.6) [5] [7].

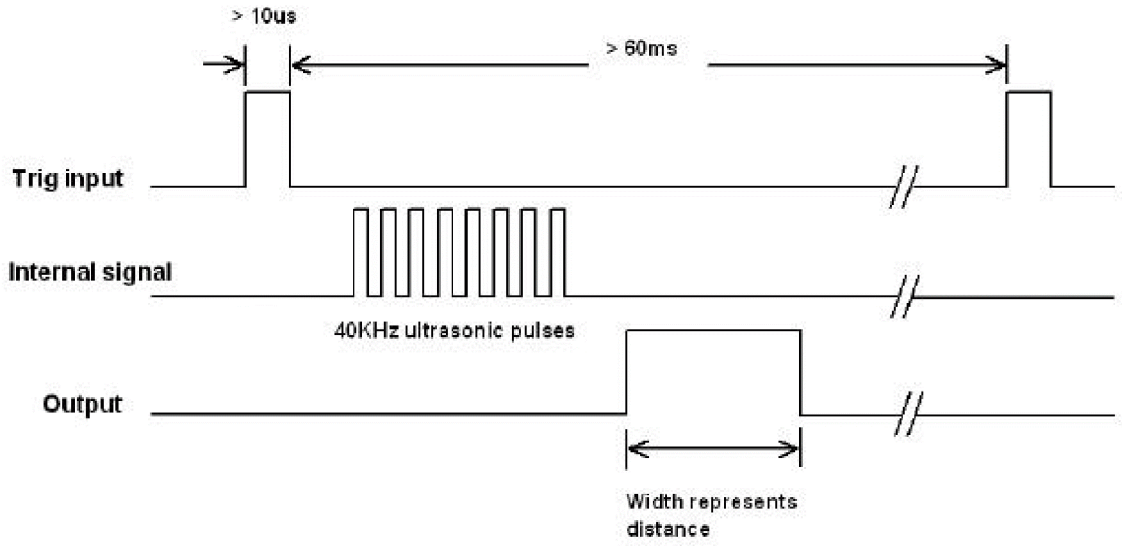


Figure II.6 : Train d'impulsion émissent pas le capteur.

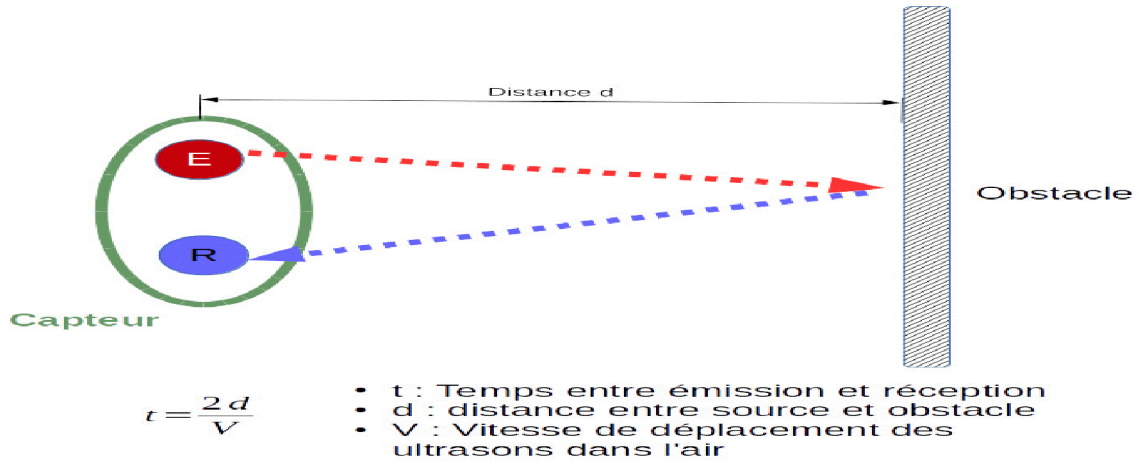


Figure II.7 : Principe de fonctionnement d'un capteur ultrasonique.

II.5.7 Distance de l'objet

La distance parcourue par un son se calcule en multipliant la vitesse du son, environ 340 m/s (ou 34'000 cm/1'000'000 µs) par le temps de propagation, soit :

d = v · t (distance = vitesse · temps)

Le HC-SR04 donne une durée d'impulsion en dizaines de µs. Il faut donc multiplier la valeur obtenue par 10 µs pour obtenir le temps t. On sait aussi que le son fait un aller-retour. La distance vaut donc la moitié [5] [7].

II.6 Les moteurs pas à pas

Tous d'abord, un moteur est un composant de conversion d'énergie électrique en énergie mécanique. Dans notre cas, les moteurs pas à pas transformant l'énergie électrique en énergie mécanique sous forme de rotation.

II.6.1 Définition du moteur pas à pas

Les moteurs pas à pas sont des moteurs spécieux utilisé pour commander aves une grande précision le déplacement et la position d'un objet. Comme leur nom l'indique, ces moteurs tournent par incrément discrets. Chaque incrément de rotation est provoqué par une impulsion de courant fournie à l'un des enroulements du stator.

Selon sa construction, un moteur pas à pas peut avancer de 90° ,45° ,18°, ou d'une fraction de degré seulement par impulsion. En faisant varier la fréquence des impulsions, on peut faire tourner le moteur très longuement, d'un pas a la fois, ou rapidement a des vitesses aussi élevées que 4000 tr/min [8].

Figure II.8 : Moteur pas à pas

II.6.2 Fonctionnement d'un moteur pas à pas

La circulation d'un courant électrique dans un bobinage entraine l'apparition d'un champ magnétique, comme le détaille la (figure ci-dessous) dans le cas du solénoïde, et donc la présence de pôles Nord et Sud (deux pôles de même nature se repoussent, deux pôles Nord et Sud s'attirent) ; c'est sur ce principe de base que repose le fonctionnement de tout moteur électrique, et, de manière plus générale, de bon nombre de dispositifs électro-mécaniques :

relais, compteurs, galvanomètres, certains hautparleurs ou microphones, gâches électriques de porte, etc [6] .

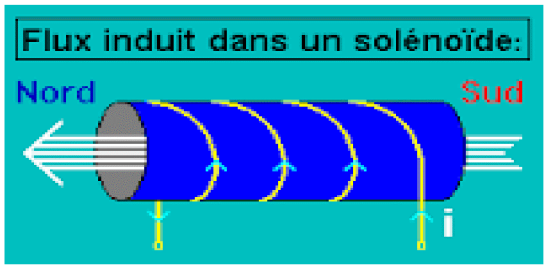


Figure II.9 : flux induit de moteur pas à pas

Le moteur pas à pas, représenté à droite, est constitué d'un rotor aimanté (en gris) avec deux pôles, Nord et Sud, ainsi que d'un double-stator (une partie en bleu, l'autre en vert) : à chacune de ces deux parties, est associé un bobinage avec un point milieu et deux phases ; en alimentant l'une ou l'autre des phases, on peut ainsi inverser l'aimantation au niveau du stator correspondant.

La flèche noire représente l'aiguille d'une boussole qui serait disposée en place et lieu du rotor, elle indique l'orientation du champ magnétique (elle pointe vers le nord, qui attire donc le pôle Sud du rotor) et se décale alors d'un quart de tour à chaque étape.

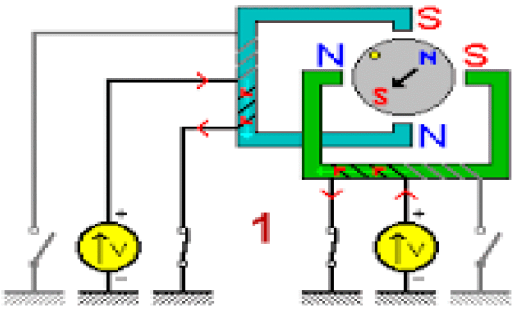


Figure II.10 : Position de moteur pas à pas

II.6.3 Types de moteur pas à pas

On trouve trois types de moteur pas à pas :

- Le moteur à aimants permanents.

- Le moteur hybride.
- Le moteur a réluctance variable.

II.6.3.1 Le moteur à aimants permanents

C'est le modèle dont le fonctionnement est plus simple. Le rotor est constitué d'un aimant permanent, et le stator comporte deux paires de bobines. En agissant sur les bobines alimentées et le sens des courants, on fait varier le champ créé par le stator. A chaque pas, la direction du champ induit par le stator tourne de 90°. L'aimant permanent suit le déplacement du champ magnétique créé par les bobines et s'oriente selon une de ces quatre positions stables. Comme le rotor est aimanté, lorsque le moteur n'est pas alimenté, le flux magnétique du à l'aimant permanent crée un couple résiduel en se plaçant dans l'axe de l'une des bobines.

Pour augmenter le nombre de positions stables et donc de pas du moteur à aiment permanent, on peut alimenter successivement une puis deux paires de bobines : c'est le mode « demi-pas »

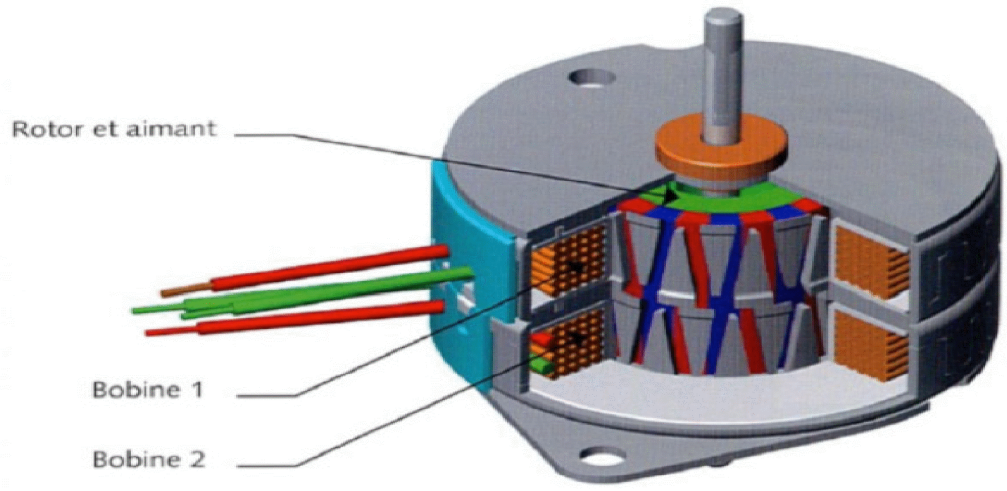


Figure II.11 : Moteur à aimant permanent.

II.6.3.1.1 Avantages du moteur à aimant permanent

- Bon marché
- Dimensions réduites

- Bon rendement
- Bon amortissement des oscillations
- Grand angle de pas (nombre de pas faible : 48)

II.6.3.1.2 Inconvénients du moteur à aiment permanent

- Puissance faible
- Paliers en bronze ou plastique (pas de roulement)
- Couple résiduel sans courant
- Vitesse faible

II.6.3.2 Le moteur à réluctance variable

Ce type de moteur pas à pas est composé d'un barreau de fer doux et d'un certain nombre de bobines. Lorsqu'on alimente une bobine, elle devient un électroaimant et le barreau de fer cherche naturellement à s'orienter suivant le champ magnétique. Dans cet exemple on alimente la phase 1, puis la phase 2, puis la phase 3…Si nous souhaitons changer le sens du moteur, il suffit de changer l'ordre d'alimentation des bobines.

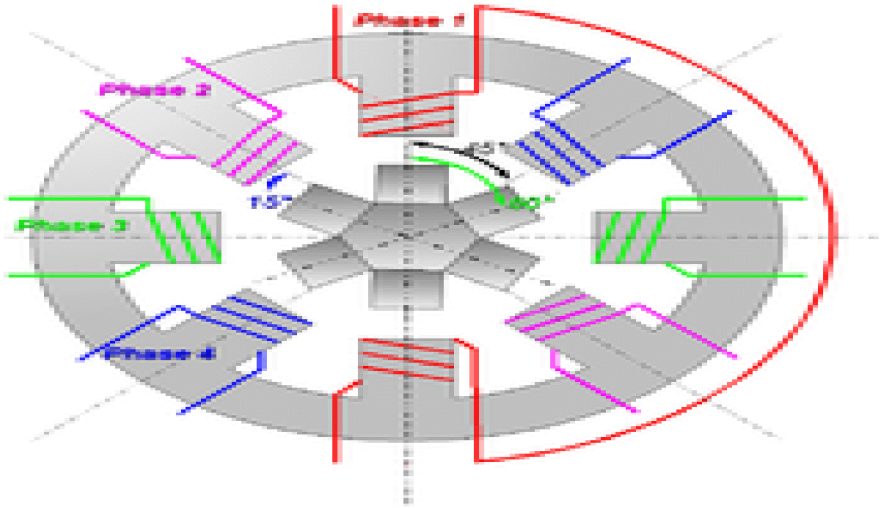


Figure II.12 : Moteur à réluctance variable

II.6.3.3 Le moteur hybride

Le moteur pas à pas « hybride » associe le principe du moteur à réluctance variable à celui du moteur à aimant permanant. Le rotor présente plusieurs dents comme pour un moteur pas à pas à réluctance variable, mais chaque dent est polarisée comme pour un moteur pas à pas à aimants permanant. Physiquement le rotor est composé de deux éléments identiques à un rotor de moteur à réluctance variable, reliés ensemble par un aimant permanent, avec un déphasage d'une demi-dent. De ce fait ces deux éléments ont une polarisation différente (nord et sud) et vont réagir à la polarisation de chacune des dents du stator. C'est cette polarisation qui permet de n'utiliser que deux bobines en même temps [9].

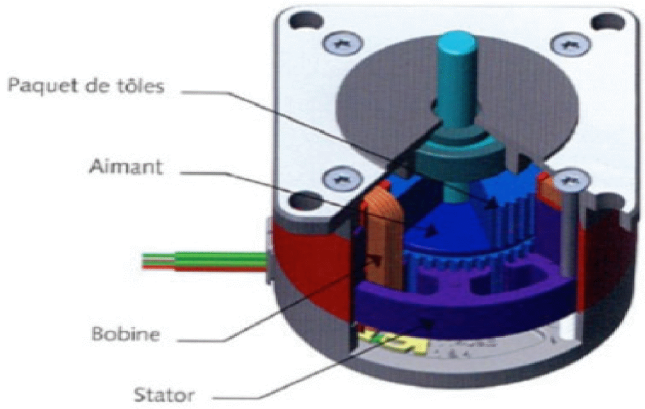


Figure II.13 : Moteur hybride

II.6.3.4 Comparaison des différents types de moteurs pas à pas

	Aimant permanant	Hybride	Reluctance variable
Cout	Bas	Haut	Moyen/haut
Vitesse	Bas	Très haut	Haut
Couple résiduel	Haut	Moyen	Minimum
Amortissement	Bon	Moyen/bon	Mauvais
Inertie rotor	Haut	Bas	Bas

Rendement	Moyen	Très haut	Moyen
Angle de pas	7.5°/15° /18°	0.9°/1.8°	1.8°
Nbrs de pas/tour	48 /24/20	400/200	200
Précision du pas	Bas	Haut	Moyen

Tableau II.14 : comparatifs des moteurs pas à pas

II.6.4 Commande de moteur pas à pas

Pour faire tourner un moteur pas à pas avec l'Arduino Uno :

On ne peut pas brancher directement le moteur à la carte Arduino, il est en effet indispensable d'utilisé un circuit intégré le (**L293D**), qui est un pont de puissance composé de plusieurs transistors et relais qui permet d'activer la rotation d'un moteur.

Le L293D est un double pont-h, ce qui signifie qu'il est possible de l'utiliser pour commander quatre moteurs distincts (dans un seul sens). (Voir figure II.15)

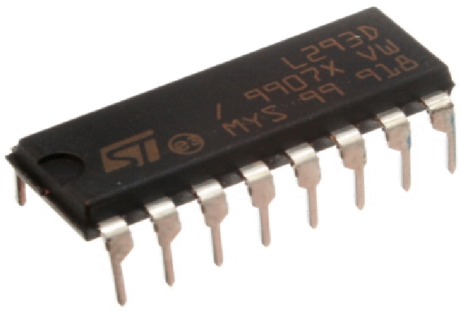


Figure II.15 : circuit intégré L293D

II.7 Les Servomoteurs

Un servomoteur (souvent abrégé en « servo », provenant du latin *servus* qui signifie « esclave »). Un servomoteur est un système qui a pour but de produire un mouvement précis en réponse à une commande externe, C'est un actionneur (système produisant une action). Un servomoteur est capable de maintenir une opposition à un effort statique et dont la position est vérifiée en continu et corrigée en fonction de la mesure, cela grâce à un système de correction qu'il embarque. Un servomoteur est donc un système asservi [6] .

Figure II.16 : Un servomoteur

II.7.1 Principe de fonctionnement

Les servomoteurs sont des moteurs particuliers, car contrairement aux autres types de moteur, leur rôle n'est pas seulement de tourner mais aussi de maintenir une position donnée. Le mouvement de sortie est une rotation qui entraine un changement de position de l'axe du servomoteur. Ils intègrent dans un même boitier un moteur à courant continu jumelé avec un train d'engrenage démultipliant (réducteur), qui entraine à son tour un axe avec une grande force de torsion mais avec une vitesse réduite. Ils embarquent aussi un capteur de retour de position (généralement un potentiomètre), ainsi qu'un dispositif électronique de commande qui permet la correction en temps réel de la position de l'axe pour qu'elle soit égale à la consigne émise par le microcontrôleur, ainsi qu'un hacheur permettent de réaliser la commande adéquate, les servomoteur sont donc des systèmes asservis (les figure .. et ..)Représentent respectivement le schéma synoptique du fonctionnement de l'asservissement de servomoteur, et une vue intérieur d'un servomoteur.

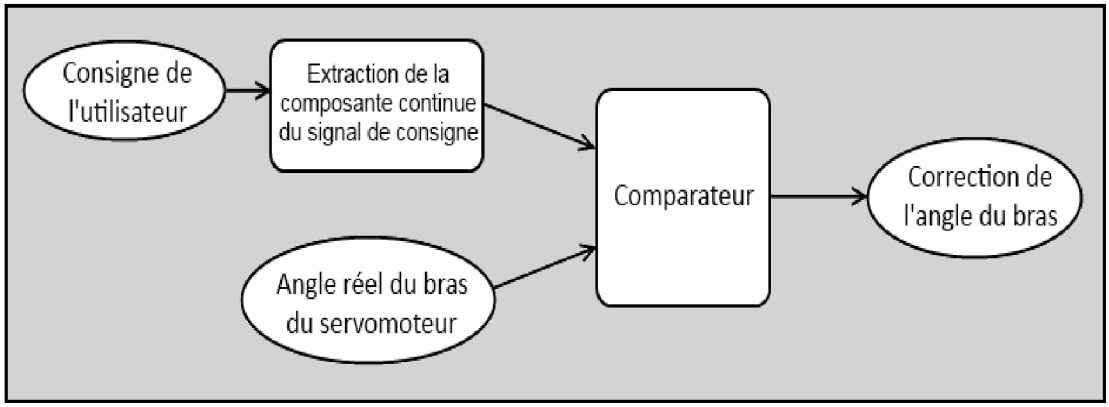


Figure II.17 : Synoptique de fonctionnement de l'asservissement du servomoteur

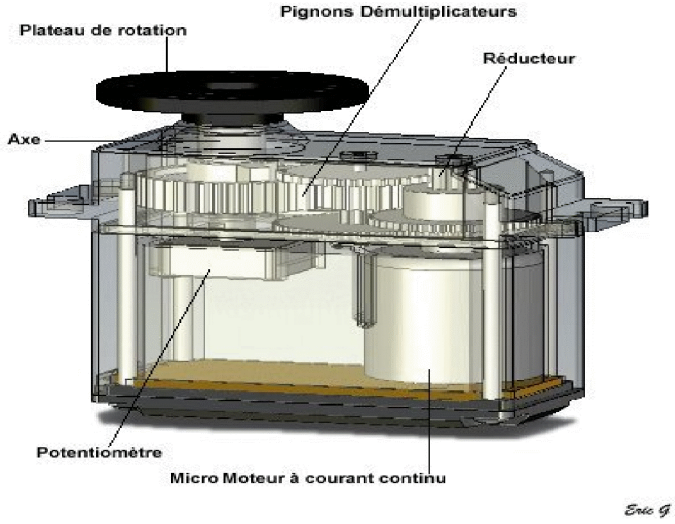


Figure II.18 : Vue intérieur d'un servomoteur

II.7.2 La mécanique

Comme on le voie sur la figure précédente, le servomoteur possède plusieurs pignons (engrenages) en sortie du petit moteur CC, cet ensemble est ce qui constitue le réducteur, ce réducteur fait deux choses : d'une part il réduit la vitesse de rotation en sortie de l'axe du servomoteur (et non du moteur CC), d'autre part il permet d'augmenter le couple en sortie du servomoteur (là encore non en sortie du moteur CC). Les moteurs CC se débrouillent très bien pour tourné très vite mais lorsqu'ils font une si petite taille ils sont bien moins bons pour fournir du couple. On va donc utiliser ce réducteur qui va réduire la vitesse, car nous n'avons pas besoin d'avoir une vitesse trop élevée, et augmenter le couple pour ainsi pouvoir déplacer une charge plus lourde.

Conformément à la formule :

$$R = \frac{\omega entrée}{\omega sortie} = \frac{C\,sortie}{C\,entrée}$$

Le rapport de réduction R du réducteur définit le couple et la vitesse de sortie (en sortie du réducteur). Selon la vitesse et le couple d'entrée (en sortie du moteur CC) [6].

Sur le datasheet de chaque servomoteur il est mentionné le couple qu'il délivre, qui est calculé par la formule :

$$C = F * R$$

C : est le couple du servomoteur (Kg.m).

F : est la force exercée sur le bras du servomoteur (kilos).

R : la distance en (m).

Prenons l'exemple d'un servomoteur que nous utiliserons dans notre travail le MG945. Sa documentation nous indique que lorsqu'il est alimenté sous 4.8V il peut fournir un couple (torque en anglais) de 10Kg/cm ; 12Kg/cm (6v). C'est-à-dire, qu'ou bout de sont bras, s'il fait 1 cm, il pourra soulever une charge de 10Kg. Si le bras fait 10 cm, l'on perdra 10 fois la capacité à soulever une masse, on se retrouve alors avec un poids de 1Kg [10]. (Voir figure II.18).

Ainsi donc en peut calculer avec la formule suivante la charge maximale que peut supporter le servomoteur :

$$Pmax = \frac{c}{d}$$

Pmax : c'est le poids maximal de charge en kg.

C : couple du servomoteur, en kg.cm.

d : distance a laquelle le poids est placé en cm.

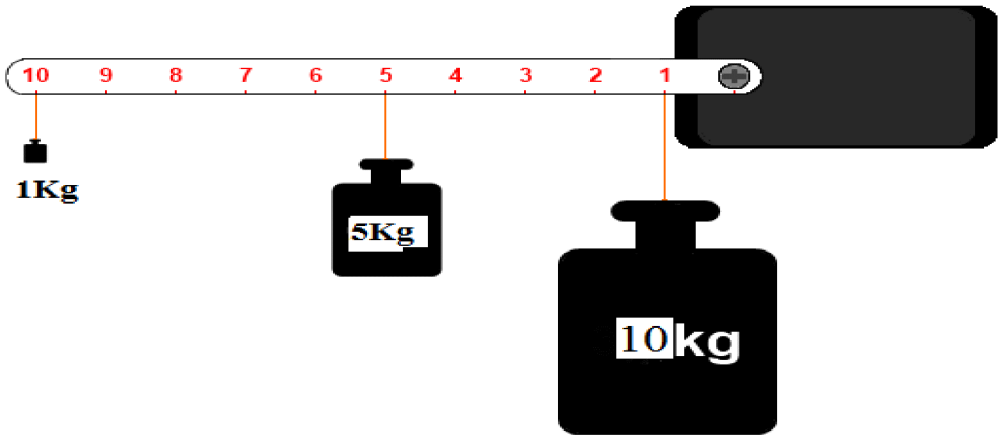


Figure II.19 : La variation du poids que peut supporter un servomoteur MG945.

II.7.3 Le branchement

Le servomoteur a besoin de trois fils de connexion pour fonctionner. Deux fils servent à son alimentation, le dernier étant celui qui reçoit le signal de commande :

Rouge : pour l'alimentation positive (4.5V à 6V en générale).

Noir ou **Marron** : pour la masse (0V).

Orange, **Jaune**, **Blanc**, … : entrée du signal de commande.

II.7.4 Le signal de commande des servomoteurs

La consigne envoyée au servomoteur n'est autre qu'un signal électronique de type PMW (modulation largeur d'impulsion). Il dispose cependant de deux caractéristiques indispensables pour que le servomoteur puisse comprendre ce qu'on lui demande. A savoir : Une fréquence fixe de valeur 50hz et une durée d'état HAUT fixée à une certaine limite comme l'illustre la figure …

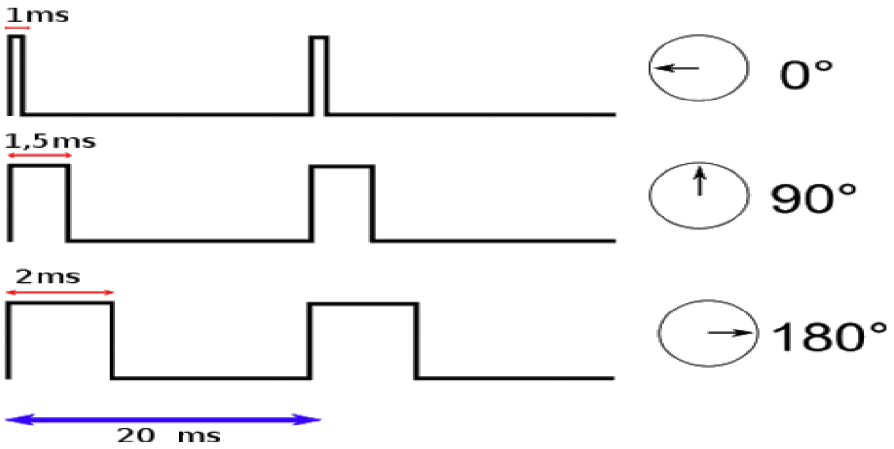


Figure II.20 : Commande de la position du servomoteur

- Une impulsion de 1ms de large place le servomoteur dans sa position de 0°.
- Une impulsion de 1.5ms centre le servomoteur à 90° dite position de repos.
- Une impulsion de 2ms de large permet de faire tourner le servomoteur à 180° qui est sa position maximal.
- Toute autre largeur d'impulsion, comprise entre 1 et 2 ms, permettent d'obtenir les innombrables positions intermédiaires.

II.6.5 Les réducteurs

II.6.5.1 Définitions des réducteurs

Le nom de réducteur est un mécanisme séparé s'intercalant entre un moteur et un récepteur. Lorsque le moteur est fixé sur le carter du réducteur, l'ensemble porte le nom de moto-réducteur.

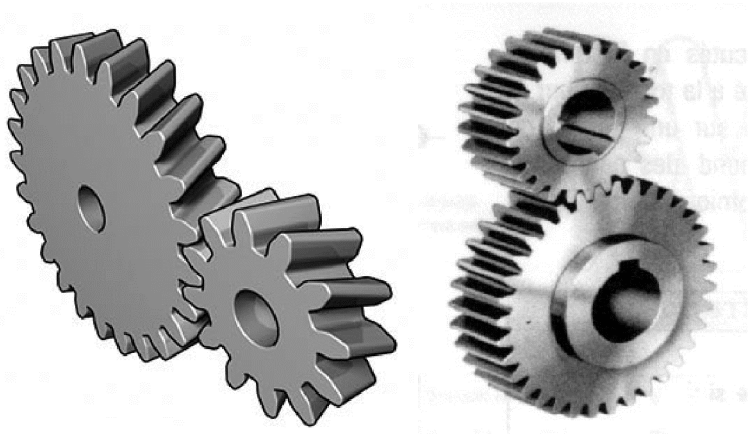


Figure II.21 : Réducteurs à dentures droite et hélicoïdale.

II.6.5.2 Fonctionnement

Un réducteur sert à réduire la vitesse d'un moteur (électrique, hydraulique, pneumatique…) avec transmission de la puissance motrice vers une machine réceptrice en absorbant le moins d'énergie. Il permet d'augmenter le couple moteur afin d'entrainer en relation un organe récepteur sous l'effet d'un nouveau couple. Les réducteurs réversibles peuvent être utilisés comme multiplicateur.

Le but de rajouter un réducteur pour un moteur est donc d'avoir un couple élevé, cette notion est très importante en mécanique car en a besoin d'avoir des moteurs qui peuvent supporter le poids de notre tapis roulant et d'avoir une certaine précision.

Un réducteur donc est un système qui modifie deux grandeurs qui sont liées : le couple est la vitesse. On peut schématiser son fonctionnement de la manière suivante :

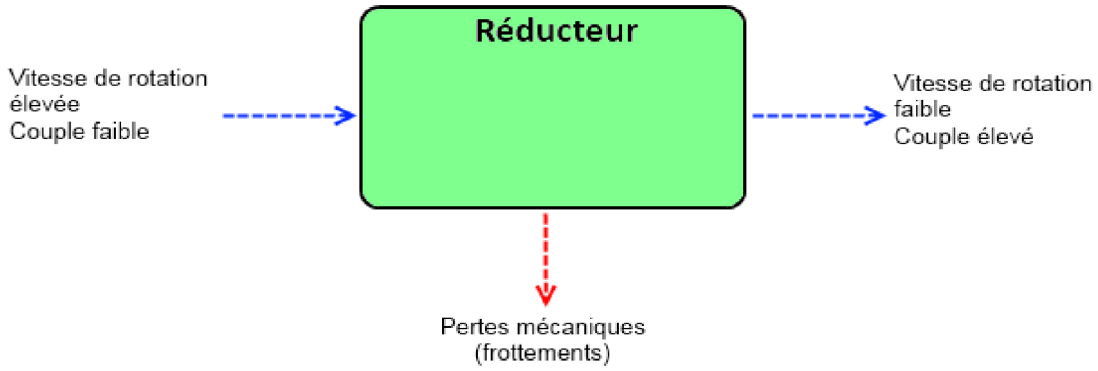


Figure II.22 : Schéma de fonctionnement d'un réducteur.

II.7 Conclusion

A travers ce chapitre, nous avons étudié et mis en évidence, les caractéristiques générales de la configuration matérielle de notre tapis roulant et puis nous nous sommes focalisés d'une manière approfondie sur le principe de fonctionnement de chaque constituant en dressant un tableau détaillé sur leurs spécificités techniques.

Dans le chapitre qui suit nous allons mettre en œuvre toutes les connaissances et compétences acquises durant cette études afin d'élaborer une réalisation pratique.

CHAPITRE III :

LA RÉALISATION PRATIQUE

I. Introduction

Après avoir fait une étude théorique on passe maintenant à la réalisation pratique. En pratique il y a plusieurs défis à relever.

Dans le chapitre précédant on a définis les différents composant qui constitue notre tapis roulant, maintenant il nous reste plus qu'à passer à la mise en ouvre pratique. Pour cela il nous faut tout d'abord établir une méthode à suivre pour la réalisation matérielle et logicielle.

I.1 La réalisation matérielle

I.1.2 Structure de notre tapis roulant

Pour réaliser notre tapis roulant on a opté pour déférents matériaux, on commençant par le chassie, qui est fait avec du bois, et puis en a ajouté a chaque extrémité des rouleaux en pvc avec des axes qui son fixé a leur tours par un jeu de roulement sur le chassie, ainsi assuré une fluidité libre de mouvement d'une partie. Et pour l'autre coté on a fait apparaitre un axe de commande motorisé pour pouvoir rattachés le réducteur qui est raccordé au moteur. Et pour assuré un convoyage des objets on a mis un tapis (bande porteuse) posé sur les roulouls de pvc.

La commande de ce moteur exige l'utilisation de deux capteurs à ultrason HC-SR04, l'un pour détecter la présence de l'objet mis sur le tapis et l'autre pour détecter la taille de cet objet, ensuite faire le tri avec le servomoteur.

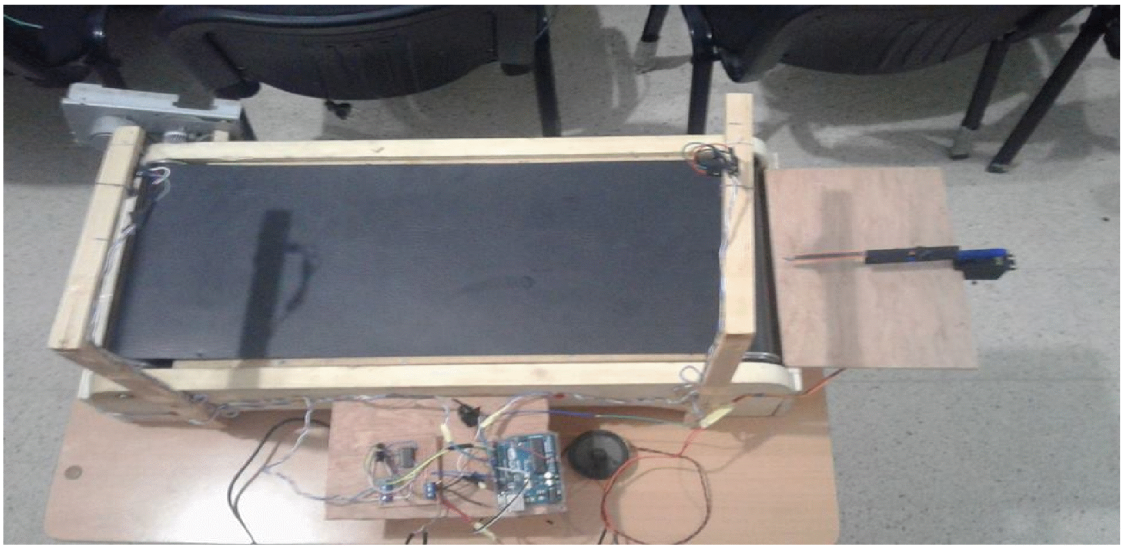


Figure III.1 : Structure du tapis roulant (convoyeur)

III.1.3 Branchement du capteur HC-SR04 à la carte Arduino

Le branchement du capteur HC-SR04 sur une carte Arduino ATMEGA 328 est plus simple. .les deux éléments sont alimentés en 5V (ou 4.5 V), il suffit de relier les bornes d'alimentations +/- entre elles.

.les entés « Trig » et «Echo » du capteur doit êtres reliée au entrées « DIGITAL » de carte Arduino.

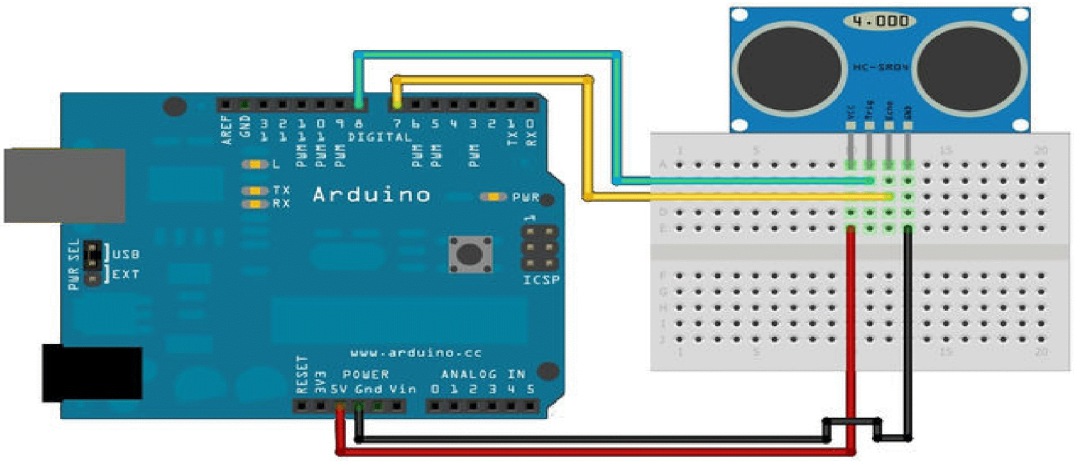


Figure III.2: branchement du capteur à ultrason.

III.1.4 Branchement du servomoteur à la carte Arduino

La connexion du servomoteur à la carte Arduino est très simple comme le montre la figure ci-dessous, tout d'abord la masse de servomoteur doit être commune avec celle de l'Arduino. Par contre, la tension positive ne doit pas être prélevée sur le 5V destiné à l'Arduino car, d'une part la consommation du servomoteur est trop importante. D'autre part chacun des mouvements provoque des appels de courant susceptible de perturber la carte en causant une baisse anormale de sa tension d'alimentation [6].

La figure ci-dessus représente la connexion du servomoteur a l'Arduino :

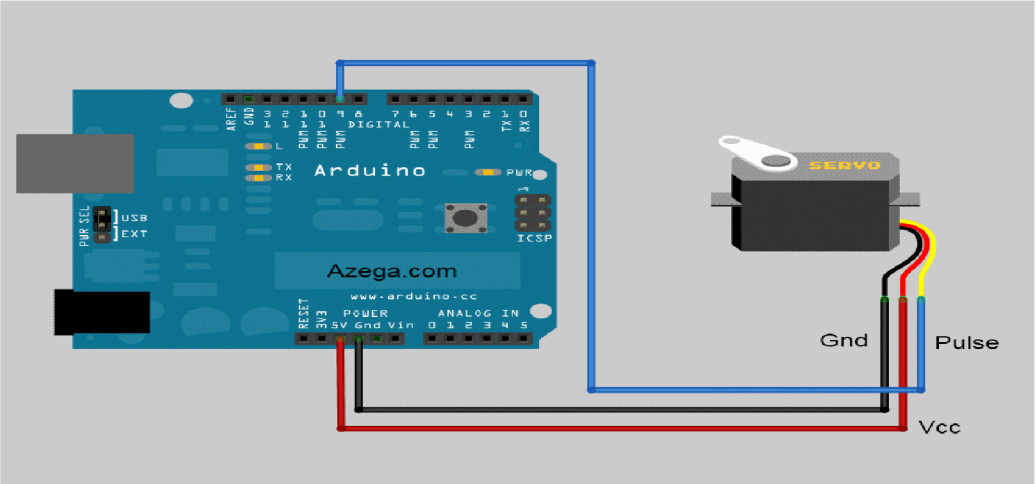


Figure III.3 : Connexion du servomoteur à l'Arduino

III.1.5 Branchement du moteur à la carte Arduino

Comme nous l'avant dit dans le chapitre précédent pour faire piloté notre moteur pas à pas il est indisponible d'utilisé un circuit intégré **(L293D)** .la **(figure III.4)** représente le branchement de moteur pas à pas bipolaire à la carte Arduino via le circuit L293D :

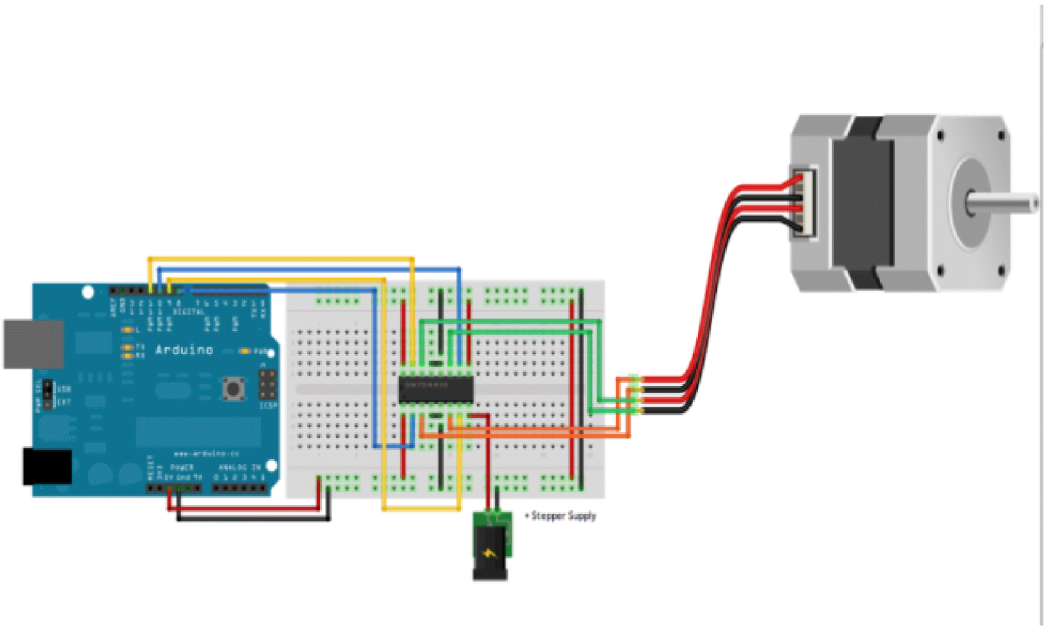


Figure III.4 : Branchement du moteur pas à pas

III.2 La réalisation logiciel

III.2.1 La programmation de l'Arduino

Pour programmer l'Arduino, on doit la brancher a un ordinateur via un câble USB (fiche A vers B) (voir la figure III.5), puis sur l'ordinateur on a besoin d'installer l'environnement de développement Arduino IDE (Integrate Development Environment) [22] qui va nous permettre de programmer la carte Arduino.

Figure III.5 : Cable USB (fiche A vers B)

L'IDE d'Arduino en faite, il s'agit d'un compilateur. Alors qu'est-ce que c'est exactement ?

III.2.2 Un compilateur

Un compilateur est un terme qui désigne un logiciel qui est capable de traduire un langage informatique, ou plutôt un programme utilisant un langage informatique, vers un langage plus approprié afin que la machine qui va le lire puisse le comprendre. Le compilateur va donc traduire les instructions du programme, écries en langage texte, vers un langage dit « machine ». Ce langage utilise uniquement des 0 et des 1. Cela peut se traduire de la façon suivante :

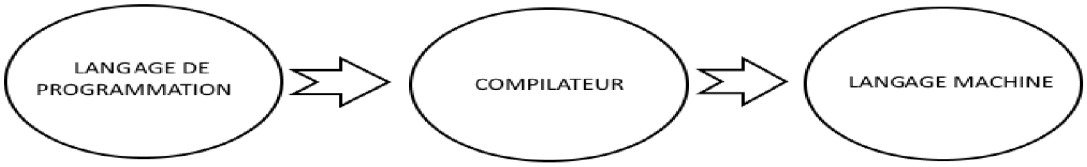


Figure III.6 : Traduction d'un langage de programmation en langage machine

III.2.3 présentation de logiciel :

La figure III.7, représente l'interface de l'IDE d'Arduino.

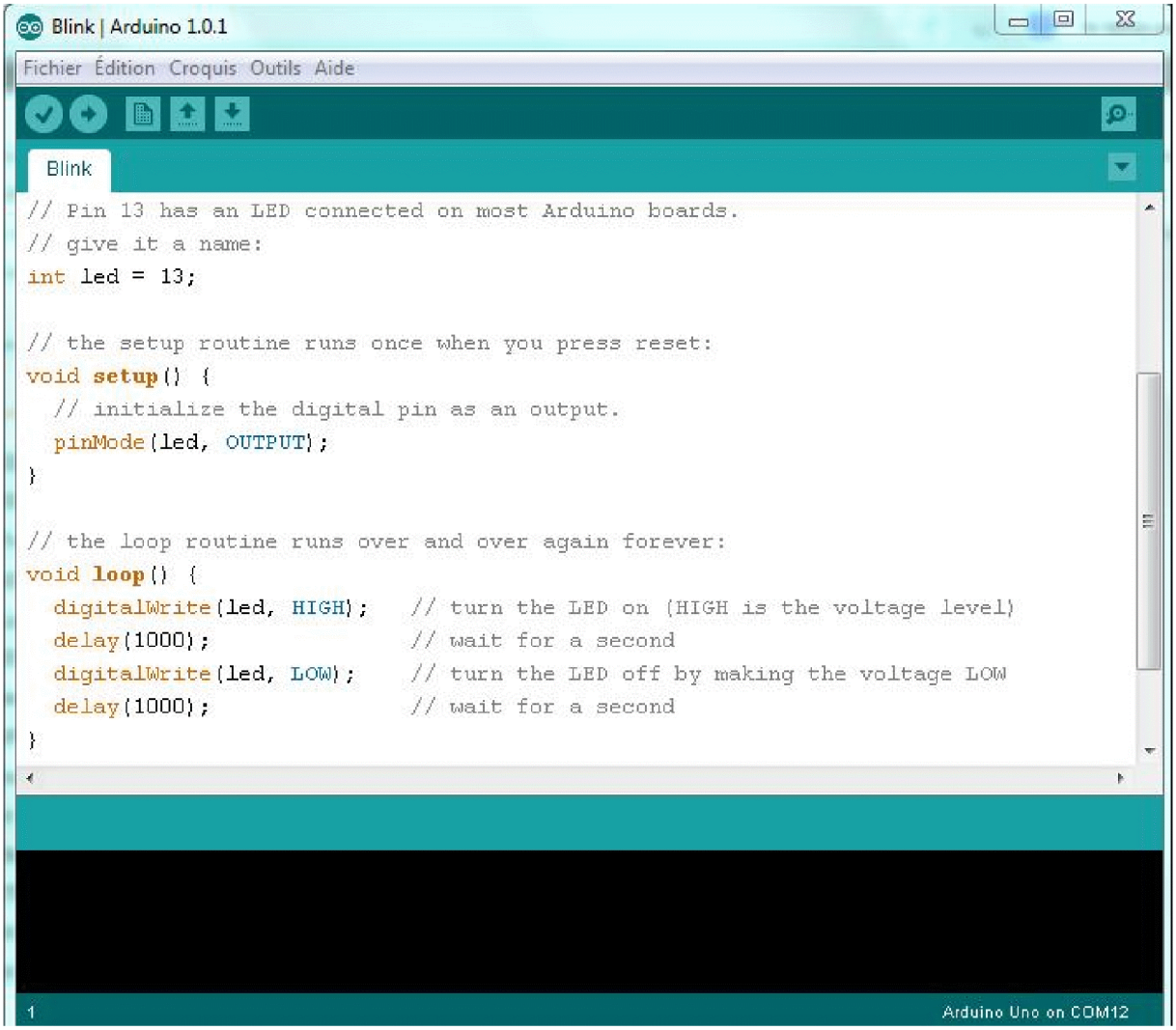


Figure III.7 : interface de l'environnement de développement Arduino IDE.

III.2.3.1 Correspondance

Le cadre numéro 1 : ce sont les options de configuration de logiciel.

Le cadre numéro 2 : il contient les boutons qui vont nous servir lorsque l'on va programmer nos cartes.

Le cadre numéro 3 : ce bloc va contenir le programme que nous allons créer.

Le cadre numéro 4 : celui-ci est important, car il va nous aider à corriger les fautes dans notre programme. C'est le débogueur.

III.2.4 fonction des icones du logiciel IDE

La figure III.7, représente les icones de l'IDE d'Arduino.

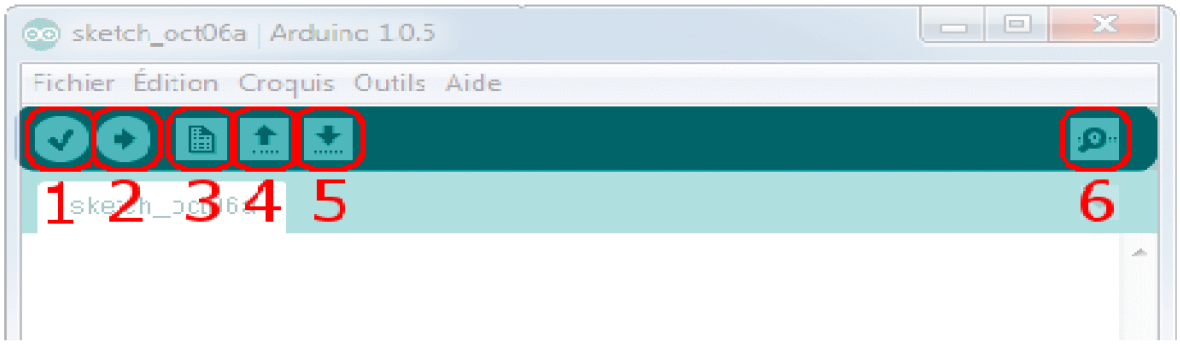


Figure III.8 : Les icones de l'IDE

Bouton 1 : ce bouton permet de vérifier le programme, il actionne un module qui cherche les erreurs de syntaxe dans le programme.

Bouton 2 : charge (Télé verser) le programme dans la carte Arduino.

Bouton 3 : crée un nouveaux croquis.

Bouton 4 : ouvrir un croquis déjà existant.

Bouton 5 : enregistre le fichier courant.

Bouton 6 : ouvre le moniteur série, qui permet le communication entre la carte Arduino et l'ordinateur à lequel on l'a branché.

Pour programmer n'importe quelle carte Arduino il faut suivre les étape suivantes :

- Allez dans le menu **outils**, ensuite cliquer sur **type de carte** et puis sélectionner la carte dont on a besoin.
- Ensuite il faut s'assurer qu'on est connecté au bon port série .il nous faut donc aller dans le menu **outils** puis **port** et enfin le port adéquate.
- Par la suite il nous reste plus qu'à programmer l'application que l'on souhaite réaliser. Une fois le programme terminé, on clic sur le bouton **téléviser** qui va se charger de transmettre le code source si celui-ci n'a pas d'erreurs. Voila la carte Arduino a maintenant été programmée.

III.2.5 Programmation sous l'IDE d'Arduino

Comme nous l'avons déjà dit dans le premier chapitre, le langage de programmation de l'Arduino étant dérivé de C et C++, il respecte les règles de syntaxe propres à ces langages qui restent au demeurant, relativement abordables.

Le contenu du programme, donc le programme en lui-même, est ce qui va définir chaque action que va exécuter la carte Arduino.

Au final, tous ce que va faire la carte est inscrit dans le programme. Sans programme, la carte ne sert à rien.

III.6 Conclusion

Dans ce chapitre nous avons procédé à la réalisation pratique d'un tapis roulant autonome commandé par un système de détection.

Cette partie est la mise en œuvre des différents composants garantissant la réalisation de ce tapis roulant automatisé. Ce dernier se base sur une carte Arduino Uno, ainsi qu'un moyen de locomotion consistant un moteur pas à pas commandé par un circuit intégré (driver). La perception quant à elle est assurée par deux capteurs ultrasoniques (HC-SR04) qui permettent de détecter la présence des objets et ainsi mesurer leurs tailles pour ensuite faire le tri à l'aide d'un servomoteur.

Une fois notre tapis roulant réalisé, nous avons procédé à toute une panoplie de tests pour nous assurer de son bon fonctionnement.

CONCLUSION GÉNÉRALE

Conclusion générale

Notre travail traite les difficultés de transfert de divers produit et matières premières et de produits fini dans une chaine de distribution. Ainsi on a pour but la conception et la réalisation d'un tapis roulant totalement automatisé.

En premier lieu, nous avons fait un aperçu générale sur la carte d'acquisition qui est l'Arduino, juste après nous avons fait une études théorique des différents éléments constituant notre tapis roulant, pour ensuite jumeler cette panoplie de composants en une réalisation pratique.

Dans ce mémoire, on a pu réaliser avec succès un tapis roulant automatisé basé sur une carte Arduino Uno, et un circuit intégré (L293D) comme pilote de notre moteur pas à pas.

Notre application est dotée de deux capteurs ultrasoniques, le premier permet de détecter la présence et le dimensionnement d'objet posé sur le tapis et en plus donner ordre au moteur pas à pas de démarrer , et le deuxième fait en sort d'arrêter le moteur pas à pas à l'arriver de l'objet et donne une impulsion au servomoteur de basculer vers la droite ou à gauche et ainsi faire le tri .

Les possibilités d'amélioration du tapis roulant présentées dans ce mémoire sont nombreuses, par exemple,on peut concevoir un bras robot de manière qu'il doit être capable de prendre des objets et les déplacer de la chaine de production vers le point de stockage. Ceci permettra au système de réduire le temps d'a chenûment des produits.

BIBLIOGRAPHIE

Bibliographie

[1]: H. BREIDENBACH, Conveyor Belt Technique Design and Calculation, NETHERLANDS. 04/2017

[2] : Patrick Chantereau et Erik Bartmann, éd. Eyrolles, *Le grand livre d'Arduino*, 2014

[3] : http : //zoetrop.io/teck-blog/esp8266-bootloader-modes-and-gpuio-state-startup [en ligne] ;le /05/2017.

[4] : Arduino.cc :http ://forum.arduino.cc/index.php?topic=86882.0 ;[en ligne].06/2017

[5] : les capteur pour Arduino et Raspberry pi- Tutoriels et projets de KIMMO Karvinen et ville Valtokari pp : 37-39, 61-63.

[6] : Arduino : premier pas en informatique embarquée de Simon Landrault et Hippolyte Weisslinger. 06/2017

[7] :http://geonobotwiki.free.fr/doku.php?id=robotics:electronics:utiliser_un_capteur_ultrason_srf04. 06/2017

www.didel.com/coursera/ArduinoTimer2.pdf.

[8] : « Systèmes électromécaniques» ; Haute Ecole d'ingénierie et de Gestion Du Canton du Vaud, CD/SEM/Cours/Chap07. 07/2017

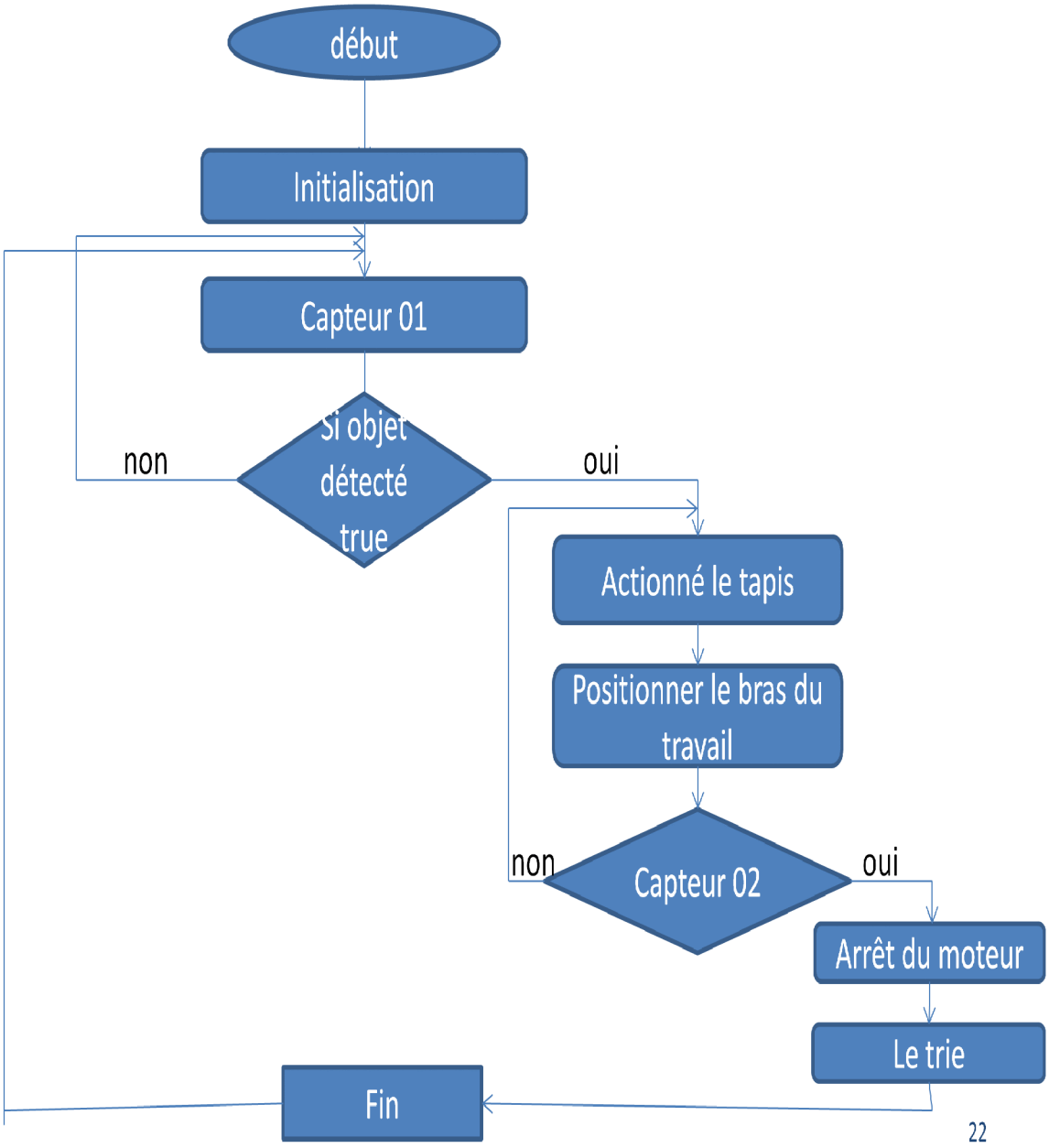
[9] : Copyright © 27/01/2016 Moteur industrie. Tous droits réservés, Moteur industrie 33 rue Paul Gauguin 31100 Toulouse.

[10] : https://www.itead.cc/tower-pro-mg-945-digital-high-speed-servo.html 07/2017

ANNEXES

ANNEXE A

➢Organigramme de fonctionnement du tapis :



Le programme de notre tapis roulant :

```
#include <Servo.h>

Servo AXE;

int TRIGCP1=9,ECHOCP1=8;

int TRIGCP2=12,ECHOCP2=11;


//Min and max pulse

int minPulseRate       = 600;

int maxPulseRate       = 2400;

int MAX_AXE = 180;


int pin1=2,pin2=3,pin3=4,pin4=5,pinEN = 6;

int temp = 8,cmp=0;

int ETAT=0;

long SaveMillis=0;

int CMPTM=0;

int Distance=0,CMPD=0;

int ChoixPD=90,ChoixPF=90;

int NBRTotal=0,NBRPP=0,NBRGP=0;

//#######################################

 void setup(){

   bip(800,300);

   pinMode(10,OUTPUT);

   pinMode(pin1,OUTPUT);

   pinMode(pin2,OUTPUT);

   pinMode(pin3,OUTPUT);

   pinMode(pin4,OUTPUT);
```

```
  pinMode(pinEN,OUTPUT);
  Serial.begin(9600);//moniture série
  initAXE();
  MOTEUR_POSITION(90);
  Serial.println("#### INITIALISATION DU SYSTEME ####...");
  bip(800,300);bip(800,300);
  Serial.println("#### SYSTEME ... OK ####");

  delay(2000);
}
//#########################################################################
void loop(){
 if(ETAT!= 1){   CMPD=0;
          do{
           CMPD++;
           Distance=Capteur01US();
           //Serial.print("Capteur N01=");Serial.print(Distance); Serial.println(" mm");
           delay(200);
           }while((Distance>140)||(CMPD<4));
           NBRTotal++;
           ETAT=1;
           bip(1200,100);bip(1200,100);
           delay(2000);
           Serial.print("####  DISTANCE EST =");Serial.print(Distance); Serial.println(" mm #####");
           if(Distance>85){NBRPP++;ChoixPD=160;ChoixPF=20;}
           else{NBRGP++;ChoixPD=20;ChoixPF=160;}
```

```
            MOTEUR_POSITION(ChoixPD);
}
if(millis()-SaveMillis>2000){
SaveMillis=millis();
Serial.print("Capteur N02=");Serial.print(Capteur02US()); Serial.println(" mm");
 if(Capteur02US()<120){
  ETAT=0;CMPTM=0;
  do{
   CMPTM++;
   moteur();
   delay(1);
   }while(CMPTM<1000);
   delay(1000);
   MOTEUR_POSITION(ChoixPF);
   delay(2000);
   Serial.println("########################################");
   Serial.print("Nombre de Pieces Total =");Serial.println(NBRTotal);
   Serial.print("Nombre de Petite Pieces =");Serial.println(NBRPP);
   Serial.print("Nombre de Grande Pieces =");Serial.println(NBRGP);
   Serial.println("########################################");
   MOTEUR_POSITION(90);
   bip(1200,300);bip(1200,300);
 }
}
if(ETAT!= 0){
moteur();
}
```

```
 }
//#########################################################################
void moteur(){
 cmp++;
 if(cmp>4){cmp=0;}

 digitalWrite(pinEN,LOW);

if(cmp == 1){
  digitalWrite(pin1,HIGH);
  digitalWrite(pin2,LOW);
  digitalWrite(pin3,LOW);
  digitalWrite(pin4,LOW);
        }

if(cmp == 2){
  digitalWrite(pin1,LOW);
  digitalWrite(pin2,HIGH);
  digitalWrite(pin3,LOW);
  digitalWrite(pin4,LOW);
        }

if(cmp == 3){
  digitalWrite(pin1,LOW);
  digitalWrite(pin2,LOW);
  digitalWrite(pin3,HIGH);
```

```
 digitalWrite(pin4,LOW);
        }

if(cmp == 4){
 digitalWrite(pin1,LOW);
 digitalWrite(pin2,LOW);
 digitalWrite(pin3,LOW);
 digitalWrite(pin4,HIGH);
        }
  digitalWrite(pinEN,HIGH);
   delay(temp);
}
 //################################################################
int Capteur01US(){

pinMode(TRIGCP1, OUTPUT);
pinMode(ECHOCP1, INPUT);

digitalWrite(TRIGCP1,LOW);
delayMicroseconds(5);
digitalWrite(TRIGCP1, HIGH);
delayMicroseconds(10);

 long temps = pulseIn(ECHOCP1, HIGH);
 long distance = int(0.17 * temps); // 0,017 pour CM et 0,17 pour MM

 return distance;
```

```
}
//#############################################################
 //#############################################################
int Capteur02US(){


pinMode(TRIGCP2, OUTPUT);
pinMode(ECHOCP2, INPUT);


digitalWrite(TRIGCP2,LOW);
delayMicroseconds(5);
digitalWrite(TRIGCP2, HIGH);
delayMicroseconds(10);


  long temps = pulseIn(ECHOCP2, HIGH);
  long distance = int(0.17 * temps); // 0,017 pour CM et 0,17 pour MM


 return distance;
}
//#############################################################
//################################################
void initAXE() {
  AXE.attach(7);
}
//################################################
//################################################
//Changeement de la valeur de l'AXE
void MOTEUR_POSITION(int POS) {
```

```
  AXE.writeMicroseconds(ValeurCVRT(POS));
}
//################################################
//################################################
            //Fonctions de gestion des interruptions
//------------- fonction calibrage impulsion des ServoMoteurs
int ValeurCVRT(int valeur) {
    int impuls=0;
    impuls=map(valeur,0,MAX_AXE,minPulseRate,maxPulseRate);
    return impuls;
} // fin fonction impulsion
//############################################################
//#################################################################
void bip(int frq,int Tm){
 tone(10,frq);
 delay(Tm);
 noTone(10);
 delay(Tm);
 }
//#################################################################
```

ANNEXE B

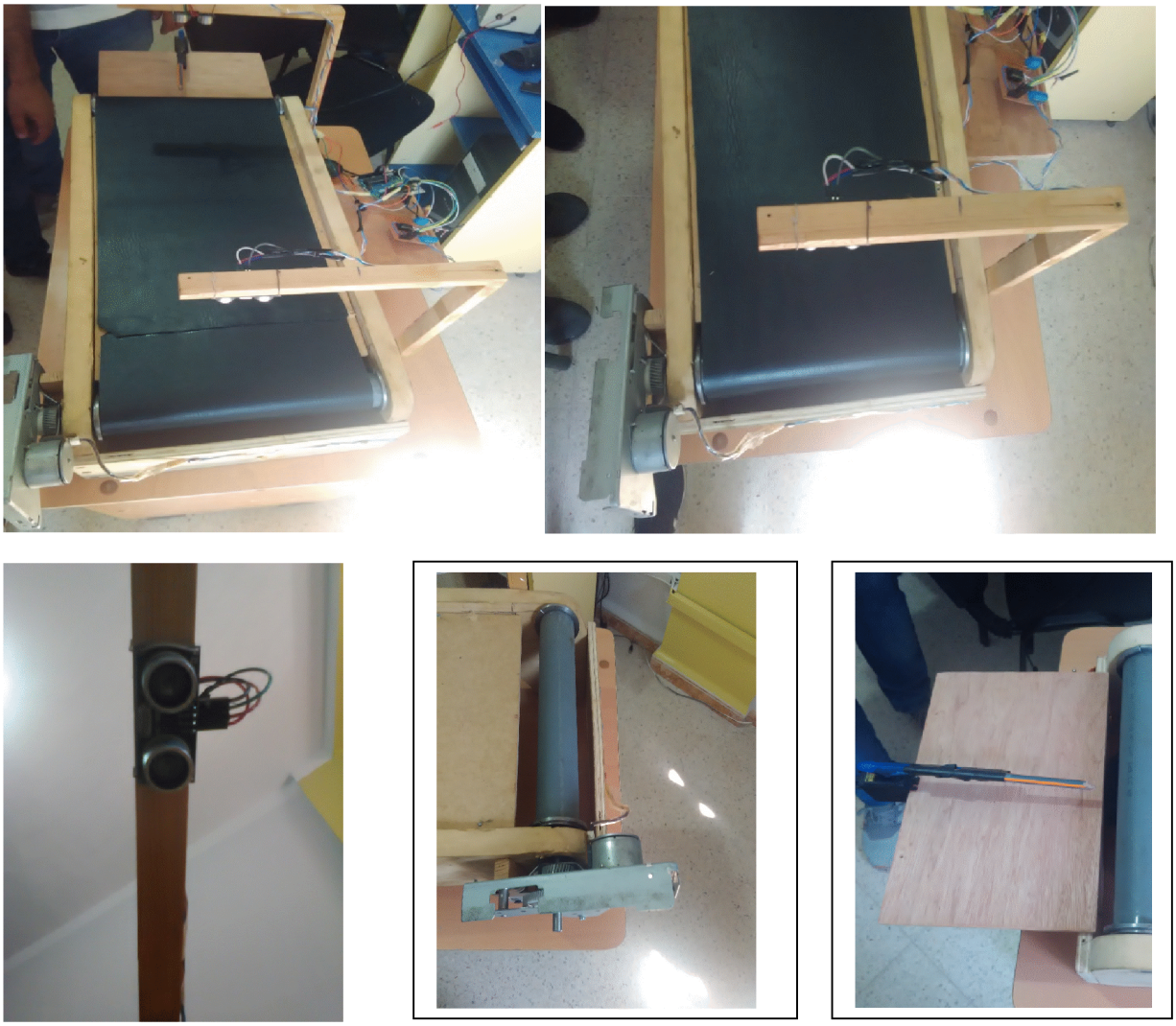


Figure III.9 : photos de notre tapis roulant