Ministère de l'**E**nseignement **S**upérieur et de la **R**echerche **S**cientifique

جامعة الدكتور يحيى فارس بالمدية

كلية العلوم و التكنولوجيا

Université **D**r. **Y**ahia **F**arès de **M**édéa

Faculté des Sciences et de la Technologie

Département de Génie Mécanique

Mémoire de projet de fin d'étude

Pour l'obtention du diplôme Master option Mécatronique

THEME

Etude et réalisation d'un banc de traitement de surface à un chariot

Dirigé par :

Mr : Tchoketch Kebir Sofian

Mr : Maazouz Mohamed

Présenté par :

Zouaouine Mahfoud

Oudjouadj Karima

2015 - 201 6

ملخص

الهدف من هذا العمل هو دراسة وانجاز مكانيكة لغرض معالجة السطح العناصر انطلاق ا من تحقيق الهيكل الميكانيكي اولا مراعين في ذلك القياسات اللازمة للهيكل ، ثم ننتقل إلى مرحلة البرمجة ذلك باستعمال لوحة الكترونية جد مستعملة في مجال الالكتروني ات الدقيقة والتي تعرف بلوحة **آردوي نو** ، تعتبر المكانيكة في حد ذاتها نظام اوتوماتيكي ذو نمطين ، نمط آلي و نمط يدوي ، كما ننهي عملنا هذا بتحليل التجارب المتحصل عليها بعرض جلّ النتائج

الكلمات المفتاحية : معالجة السطح، البرمجة **آردوي نو**

Résumé

L'objectif de ce travail est l'étude et la réalisation d'un banc de traitement de surface, nous commençons par la réalisation de la structure mécanique du banc tout en respectant les dimensions linéaires appropriées. Nous passons ensuite à la phase de programmation qui est basée sur une carte de développement très utilisée dans le domaine électronique, il s'agit de la carte Arduino_Uno. Notons bien que le banc représente en lui-même un système automatique chose qui va nous permettre d'élaborer deux modes de fonctionnement, le mode automatique et le mode manuel. En fin nous terminons notre travail par les tests nécessaires au bon fonctionnement du dispositif de traitement de surface ainsi que les constats à aboutir.

Mots clés : Traitement de surface, programmation, **Arduin_Uno**,

Abstract

The objective of this work is the study and implementation of a surface treatment bench; we start with the realization of the mechanical structure of the bench while respecting relevant linear dimensions. Then we move to the programming phase which is based on a widely used of the development board in the electronics field, it is the Arduino_Uno Board. Note that the bench itself is an automatic system that will allow us to develop two modes of operations, auto mode and manual mode. At the end we finish our work by a necessary test for the proper surface treatment device to reach the final acceptable results.

Keywords: Surface treatment, Programming, Arduin_Uno

Remerciements

Avant tout nous remercions ALLAH de nous avoir accordé des connaissances de la science et de nous avoir aidés à réaliser ce travail.

Nous tenons à remercier chaleureusement et respectueusement nos promoteurs ***Mr : Tchoketch Kebir Sfian*** *et* ***Maazouz mohamed*** *pour leur aides et leur soutient tout au long de notre projet ainsi que ceux qui ont contribué de prés ou de loin à la réalisation de ce mémoire de fin d'étude, sans oublier d'adresser nos remerciements à tous les enseignants du département Génie mécanique notamment la spécialité* ***Mécatronique.***

Nos vifs remerciements aux membres de jury pour nous avoir fait l'honneur et le plaisir d'accepter de juger notre travail.

Dédicaces

Je remercie Allah le tout puissant de m'avoir donné la force et le courage pour accomplir ce modeste travail.

A mes parents qui m'ont soutenue et encouragé après tant d'années et à qui je leur souhaite une longue et heureuse vie, que dieu les protège et me donne la force pour que je puisse leur rendre un petit peu de leur bienfait et leur bonté.

A ma ***femme*** *et mes enfants* ***zine-eddine*** *et* ***salah-eddine*** *que je dois tout le respect et le bonheur, à tous mes frères et mes sœurs que je leur souhaite tout le bonheur.*

Dédicaces

Aucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours eu pour vous.

Papa chéri, ton premier souci à été d'offrir à tes enfants la formation qu'ils ont désirée et de leur inculquer le sens de la vertu.

A toi maman que j'aime tant par ton aide constante, tes sacrifices quotidiens et tes précieux conseils tu a fait de moi ce que je suis aujourd'hui

A mes frères, sœurs, neveux et nièces chacun par son nom ; je vous souhaite un avenir plein de joie de bonheur, de réussite et de sérénité.

A toi mon âme sœur, tes sacrifices, ton soutien moral, ta gentillesse, ton profond attachement m'ont permis de réussir mes études.

Sans ton aide, tes conseils et tes encouragements ce travail n'aurait vu les jours.

Que dieu réunisse nos chemins pour un long commun serein.

A mes deux anges gardiens mes fils ZINEDDINE et SALAHEDDINE qui sont ma raison de vie

A mes deux étudiants préférés MAHMOUD et AHMED

A tous mes amis (ies)

Je dédier ce modeste travail.

Résumé
SOMMAIRE
liste Figure
Liste Tableaux

Introduction Générale

Chapitre I Généralités

I-1 Introduction
I-2 Traitement de surface
I-3 La carte arduino_Uno
I-3-1 Présentation
I-3-2 principales caractéristiques
I-3-3 les différents éléments de la carte Arduino_Uno
I-3-4 Programmation de la carte Arduino _Uno
I-4 Moteurs pas à pas
I-4-1 Introduction
I-4-2 Moteur à aimant permanent
I-4-3 Moteurs à réluctance variable
I-4-3 Moteurs hybrides
I-4-4 Comparaison des trois types
I-4-5 Fonctionnement du moteur pas à pas à aimant permanent
I-5 Le capteur optique CNY70

Chapitre II Construction mécanique sous SolidWorks

II-1 Introduction
II-2 Le logiciel SolidWorks
II-3 poste de traitement (Pièce 1)
II-4 Le socle et le chariot (Pièce 2 et Pièce 3)
II-5 traverse du coté droit et du coté gauche (Pièce 4 et Pièce 5)
Conclusion

Chapitre III Etude technique

III_1 Introduction
III_2 Etude électrique
III_2_1 Schéma synoptique
III_2_2 Schéma électrique
III-2-3 Etude électronique des cartes
III-3 Etude Automatique
III-3-1 Le projet en tant que système automatisé
III-3-2 Modes de fonctionnement de la maquette
III-3-3 Schéma fonctionnel de la maquette
III-4 Programmation de la maquette
III-4-1 Organigramme du mode automatique
III-4-2 Organigramme du mode manuel
III-5 Conclusion

Chapitre IV Etude pratique des cartes électriques et constats obtenus

IV_1 Introduction
IV_2 Fabrication des circuits imprimés
IV_3 Assemblage des cartes et connexion avec **Arduino**
IV_4 Insertion des deux moteurs dans le chariot
IV_5 Insertion de la carte capteurs au dessus du chariot
IV_6 Commutateur de modes
IV_7 pupitre de commande
IV_8 Constats obtenus

IV-8-1 Constats obtenus de la carte Driver et la carte capteurs
IV_8_2 constats obtenus lors de fonctionnement automatique
Conclusion
Conclusion générale
Bibliographies
Annexe

SOMMAIRE

LISTE DES FIGURES

Figure I-1 : Différents types de traitement de surface
Figure I-2 : carte Arduino_Uno
Figure I-3 : les compositions de la carte Arduino_Uno
Figure I-4 : Interface de programmation
Figure I-5 : la barre des boutons
Figure I-6 : Structure d'un programme
Figure I-7 Moteur à réluctance variable (MRV)
Figure I-8 Moteur à réluctance variable (MRV)
Figure I-9 moteurs pas à pas hybrides
Figure I-10 structure d'un moteur pas à pas bipolaire
Figure I-11 Illustration du capteur CNY70
Figure I-12 Brochage interne du capteur
Figure I-13 Démonstration de la réflexion
Figure II-1 Fichiers assurés par SolidWorks
Figure II-2 Interface du fichier pièce
Figure II-3 Les cotes de la maquette
Figure II-4 Image 3D du poste de traitement sous SolidWorks
Figure II-5 Vue de face du poste de traitement sous SolidWorks
Figure II-6 Vue de dessus du poste de traitement sous SolidWorks
Figure II-7 Cotation du poste de traitement
Figure II-10 cotation du socle en perspective sous SolidWorks
Figure II-11 Vue de dessus du socle
Figure II-12 Cotation du socle
Figure II-13 Hauteur entre la pièce 1 et les rails
Figure II-14 vue en image ombrée
Figure II-15 Cotation des cotés droite et gauche
Figure III-1 : Schéma synoptique
Figure III-2 : Schéma électrique global
Figure III-3 :schéma électrique du bloc de commande manuelle
Figure III-4 :schéma électrique du bloc capteurs
Figure III-5 Polarisation de la LED émettrice du CNY70
Figure III-6 :polarisation du recepteur du CNY70
Figure III-7 :schéma électrique du circuit de commande du moteur pas à pas
Figure III- 8: Schéma fonctionnel de notre système automatique
Figure III- 9 : Photo réelle du poste de traitement
Figure III-10 Organigramme principal
Figure III-11 Organigramme de déplacement à droite du chariot
Figure III-12 Organigramme de déplacement à gauche du chariot
Figure III-13 Organigramme de la montée de la pincette
Figure III-14 Organigramme de la descente de la pincette
Figure IV-1: typon du Driver à double faces
Figure IV-2: circuit imprimé du Driver
Figure IV-3: typon face cuivrée
Figure IV-4: face composants
Figure IV-5: carte Driver finalisée
Figure IV-6: Typon de la carte capteurs

Figure IV-7: la carte capteurs finale fixée au dessus du chariot
Figure IV-8: Le typon de la carte de commande manuelle
Figure IV-9: Schéma d'implantation
Figure IV-10: Typon du circuit de connexion avec Arduino_Uno
Figure IV-11: Schéma d'implantation correspondant
Figure IV-12: La carte de connexion dans son état final
Figure IV-13 Assemblage des deux Drivers
Figure IV-14 connexion des boutons de commande
Figure IV-15 Assemblage de toutes les cartes
Figure IV-16 insertion des moteurs dans le chariot
Figure IV- 17 vue de dessous du moteur chariot entrainé par la courroie
Figure IV- 18 Chariot dans son état final
Figure IV-19 Commutateur Auto/Man/Stop
Figure IV20Commutateur Auto/Man/Stop
Figure IV-21 Pupitre de commande avec la carte de connexion
Figure IV-22 connexion pupitre_Arduino Uno
Figure IV-23 Image complète de la maquette
Figure-IV-24 chronogrammes obtenus de la carte driver sous ISIS avec une fréquence de 100hz
Figure-IV -25 chronogrammes obtenus de la carte driver sous ISIS avec une fréquence de 500hz
FigureIV-26 schéma de simulation de la carte capteurs en absence de chariot
FigureIV- 27 Schéma de simulation de la carte capteurs en présence de chariot
Figure IV-28 positions réelles (8 ; 7 et 4) occupées par le chariot
Figure IV-29 Indications en temps réel via l'interface série du logiciel
Figure IV-30 positions réelles (3 ; 1 et 0) occupées par le chariot
Figure IV-31 Indications en temps réel via l'interface série du logiciel
Figure IV-32 image de l' interface série du logiciel Arduino

Liste Tableaux

Tabeau I-1 les caractéristiques de la carte Arduino_Uno
Tableau I-2 Fonctionnement des E/S numériques
Tabeau I-3 Mode pas entier
Tabeau I-4 Mode fort couple
Tabeau I-5 Combinaison des deux modes
Tabeau III 1 boutons à l'arduino uno
Tableau IV1 connexion CN2 et CN3 avec Arduino_Uno

Les abréviations

TTL : Transistor transistor logic

PWM : pulse wave modulation, modulation de la largeur d'impulsion

SDA : Signal data

SCL : Signal clock

I2C : Inter-Integrated Circuit

USB : universal serial bus

UART : Universal Asynchronous Receiver Transmitter

RxD: received data

TxD*:* transmitted data

MOSI : Master Output, Slave Input

MISO : Master Input, Slave Output

IDE : Integrated Development Environment

MRV : Moteur à réluctance variable

API : Automate programmable industriel

SS: Slave Select

SCK: synchronous clock

SPI : Serial Peripheral Interface

Introduction générale

La mécatronique est une technique industrielle consistant à utiliser simultanément et en symbiose, la mécanique, l'électronique, l'automatisme et l'informatique pour la conception et la fabrication de produits nouveaux en vue d'augmenter ou d'optimiser leurs fonctionnalités. L'ingénieur mécatronicien intervient tout au long du cycle de vie de ces produits industriels à travers différentes phases tel que:

Recherche - avant projet - développement - industrialisation - exploitation.

Pour l'ensemble de ces activités, l'ingénieur possède une solide formation en mécanique et en génie électrique.

Aujourd'hui, la mécatronique englobe de nombreuses applications dans les domaines aussi diverses et variées dans l'industrie notamment les systèmes automatisés afin d'améliorer les conditions de travail et la qualité du produit.

Comme projet de fin d'étude on a opté pour l'étude et la réalisation d'un banc de traitement de surface qui simule l'un des systèmes automatisés industriels. Le traitement de surface est une opération mécanique, physico-chimique qui a pour conséquence de modifier l'aspect ou la fonction de la surface des matériaux afin de l'adapter à des conditions d'utilisation données. Les conditions de travail dans ce domaine sont généralement insupportables et qui présentent de graves dangers pour la santé des travailleurs alors il fallait automatiser ce processus pour mieux améliorer. C'est dans ce contexte que nous allons réaliser un prototype d'un système de traitement de surface à un chariot qui va être réalisé dans le but de répondre à un ensemble de besoins qui spécifient Précisément les services demandés et attendus par l'utilisateur.

La réalisation de notre projet est constituée essentiellement d'une partie construction mécanique du prototype en utilisant le logiciel **SolidWorks** et d'une autre partie dont on fera l'étude électronique théoriquement et pratiquement.

En outre, dans l'objectif d'automatiser le banc une étude automatique est prévue dans notre réalisation, en l'occurrence il peut servir comme maquette didactique dans l'automatique.

Le document est organisé de la manière suivante:

Dans le **chapitre I**, on va aborder notre étude par des généralités sur le traitement de surface ainsi les différents composants que nous voyons essentiels dans notre réalisation.

Le chapitre II est consacré à la construction mécanique de notre prototype de traitement de surface tout en utilisant le logiciel **SolidWorks**.

Pour le **troisième chapitre** nous allons étudier théoriquement la partie électronique des différents circuits de commande ; de traitement de données et d'acquisition des données.

En fin, dans le **dernier chapitre** l'aspect pratique doit être abordé avec ses différents constats que nous allons obtenir.

Comme tout rapport de projet de fin d'étude, on doit clôturer notre étude par une **conclusion générale**.

Chapitre I

Généralités

I-1 Introduction

Dans ce chapitre nous allons présenter tous les éléments nécessaires à l'élaboration de notre projet notamment un bref résumé sur le traitement de surface ; en plus nous étudierons avec éventuelle détailles concernant la carte de développement **Arduino-uno** qui est le cœur du projet. Nous terminerons ensuite ce chapitre avec le moteur pas à pas et le capteur optique **CNY70**.

I-2 Traitement de surface

Le traitement de surface a pour effet de modifier l'apparence d'une surface (décoration, amélioration de l'aspect), ses propriétés physiques ou/et chimique, afin d'apporter une tenue à la corrosion, à l'usure, une amélioration du frottement, une résistance à l'oxydation à chaud, favoriser la soudabilité, la conductibilité électrique, modifier les propriétés optiques, assurer un effet de barrière thermique ou d'isolation aux rayonnements ou encore de préparer et favoriser l'application d'un autre traitement de surface.

Il représente un apport d'épaisseur relativement faible dépassant rarement le mm, et pouvant être de quelques nanomètres.

Dans le contexte industriel, les opérations de traitements de surfaces sont effectuées de nos jours par une multitude de petites entreprises sous-traitantes pour le compte de grands groupes industriels dans les domaines de l'automobile, de l'aérospatiale etc.

Ce contexte induit des conditions de travail généralement pénibles et l'exposition aux vapeurs de différents acides utilisés (acides sulfurique, chlorhydrique, cyanhydrique, fluorhydrique…) présente de graves dangers pour la santé des travailleurs d'où la nécessitée à l'automatisation de ce procédé.

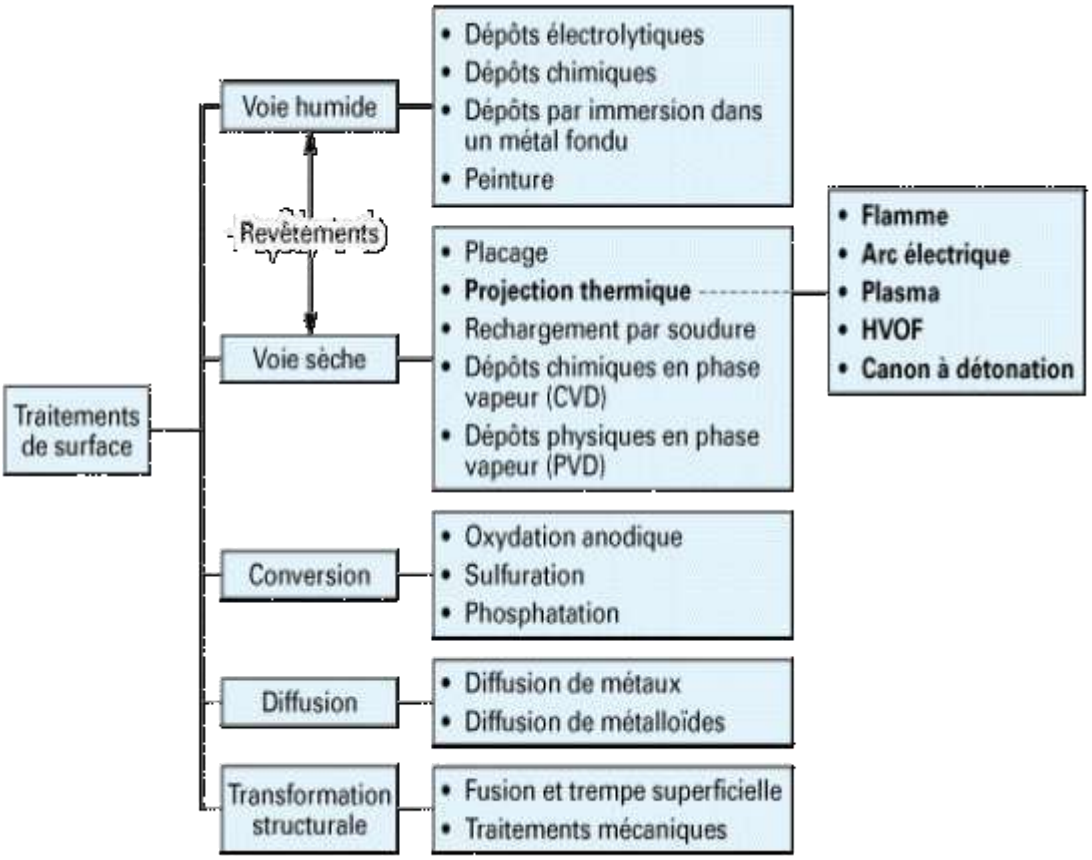


Figure I-1 : Différents types de traitement de surface

I-3 La carte arduino_Uno

I-3-1 Présentation

L'Arduino est une plateforme d'électronique programmable qui est basée sur une carte à microcontrôleur (**ATmega328**) et un logiciel open source. Il peut être utilisé pour développer des objets interactifs munis de certains composants électroniques comme par exemple, interrupteurs et capteurs et peut contrôler une grande variété d'actionneurs.

Arduino propose une gamme assez large de carte de différentes tailles avec plus ou moins d'entrées et de sorties. De plus, il est possible de rajouter de nombreux modules sur la carte (**shields**) afin de rajouter des fonctionnalités comme le Bluetooth, une connexion internet, commande des servomoteurs ou encore le wifi etc.

La **figure-I-2** montre une image représentative de ce modèle.



Figure I-2 : carte Arduino_Uno

I-3-2 principales caractéristiques

Un des modèles les plus répandu de carte Arduino est l'**Arduino Uno** . C'est la première version stable de carte **Arduino**. Elle possède toutes les fonctionnalités d'un microcontrôleur classique en plus de sa simplicité d'utilisation.

Les caractéristiques techniques sont données sur le tableau suivant :

Microcontrôleur	Atmega328
Tension de fonctionnement	5V
Tension d'alimentation (recommandée)	7-12V
Tension d'alimentation (limites)	6-20V
Broches E/S numériques	14 (dont 6 disposent d'une sortie PWM)
Broches Entrées analogiques	6 (utilisables en broches E/S numériques)
Intensité max disponible par broche E/S (5 V)	40 mA (200mA cumulé pour l'ensemble des broches)
Mémoire programme Flash	32 Ko dont 0,5 Ko sont utilisés par le bootloader
Mémoire SRAM (mémoire volatile)	2 Ko
Mémoire EEPROM (mémoire non volatile)	2 Ko
Vitesse d'horloge	20MHz

Tabeau I-1 : caractéristiques de la carte **Arduino_Uno**

I-3-3 les différents éléments de la carte Arduino_Uno

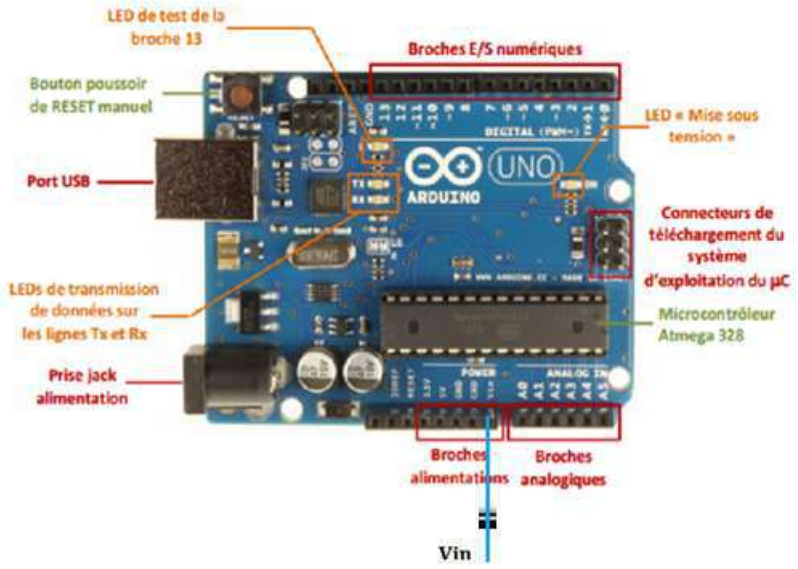


Figure I-3 : les compositions de la carte **Arduino_Uno**

1) L'alimentation

La carte **Arduino_Uno** peut être alimentée par le câble USB, par un bloc secteur externe connecté grâce à une prise jack ou bien par un bloc de piles dont le raccordement est réalisé par l'intermédiaire de **GND** et **Vin**.

L'alimentation extérieure doit présenter une tension comprise entre **7** à **12** V. En revanche la carte génère, par l'intermédiaire des régulateurs intégrés, deux tensions stabilisées **5** v et **3,3** v. Ces deux tensions permettent l'alimentation des composants électroniques de la carte Arduino, elles sont également disponibles sur les connecteurs sécables placés sur le pourtour de la carte, elles permettent également l'alimentation des modules **shields** qui sont conçus spécialement pour Arduino.

NB : Un shield est une carte que l'on connecte directement sur la carte Arduino qui a pour but d'ajouter des composants sur la carte. Ces **shield** viennent généralement avec une librairie permettant de les contrôler.

Le principal avantage de ces **shields** est leurs simplicités d'utilisation. Il suffit des les emboiter sur la carte **Arduino** pour les connecter, les circuits électroniques et les logiciels sont déjà faits et on peut en empiler plusieurs. C'est un atout majeur pour ces cartes pour pouvoir tester facilement de nouvelles fonctionnalités

2) les entrées /sorties

La carte **Arduino_Uno** dispose de **14 E/S** numériques dont **6** peuvent être configurées comme sorties de signaux **PWM** et de **6** entrées analogiques.

❖ **Entrées/sorties numériques**

Chacune des **14** broches numériques (repérées **0** à **13**) peut être utilisée en entrée (input) ou en sortie (output) sous le contrôle du programme. Le sens de fonctionnement pouvant même changer de manière dynamique pendant son exécution.

Elles fonctionnent en logique TTL (**0**V-**5**V) dont chacune pouvant fournir ou recevoir un courant maximal de **40** mA et dispose si besoin d'une résistance interne de pull-up.

N° E/S	N° ligne de port	Fonction
0	PD0	**Rx** : Entrée liaison série asynchrone
1	PD1	**Tx** : Sortie liaison série asynchrone
2	PD2	**INT0** : Entrée interruption externe
3	PD3	**INT1** : Entrée interruption externe
		OC2B : PWM modulation à largeur d'impulsion
4	PD4	**T0** : Entrée Timer/compteur 0
		XCK : Entrée horloge
5	PD5	**T1** : Entrée Timer/compteur1
		OC0B : Sortie module PWM modulation à largeur d'impulsion
6	PD6	**OC0A** : Sortie module PWM modulation à largeur d'impulsion
		AIN0 : Entrée comparateur analogique
7	PD7	**AIN1** : Entrée comparateur analogique
8	PB0	**CLKO** : Sortie de l'horloge de fonctionnement
		ICP1 : Entrée de capture Timer/compteur 1
9	PB1	**OC1A** : Sortie module PWM modulation à largeur d'impulsion
10	PB2	**SS** : Sélect Slave liaison SPI
		OC1B : Sortie module PWM modulation à largeur d'impulsion
11	PB3	**MOSI** : Sortie liaison SPI
		OC2A : Sortie module PWM modulation à largeur d'impulsion
12	PB4	**MISO** : Entrée liaison SPI
13	PB5	**SCK** : Horloge liaison SPI

Tableau I-2: Fonctionnement des E/S numériques

❖ **Entréesanalogiques**

Les six entrées analogiques, repérées **A0** à **A5**, peuvent admettre des tensions analogiques comprise entre **0** et **5** V. Ces entrées analogiques sont gérées par un convertisseur analogique/numérique de **10** bits, dont la sortie peut varier de **0** à **1023**.

Les entrées A4 et **A5** peuvent également être utilisées respectivement comme la ligne de donnée **SDA** et la ligne d'horloge **SCL** de l'interface série **I2C**. Les entrées analogiques peuvent être utilisées comme entrées numériques

3) Communications

La carte **Arduino_Uno** a de nombreuses possibilités de communications avec l'extérieur. L'Atmega328 possède une interface de communication série **UART** accessible grâce aux broches numériques **0** et **1** qui représentent respectivement **Rx** et **Tx**. D'autre part elle supporte le bus **I2C** accessible grâce aux broches analogiques **A4** et **A5** représentant respectivement **SDA** et **SCL** et la liaison série synchrone **SPI** grâce aux broches numériques **10**, **11** ; **12** et **13** qui signifiant dans l'ordre **SS ; MOSI ; MISO** et **SCK.**

4) Reset

A la mise sous tension un reset automatique permet au programme contenu en mémoire du microcontrôleur de démarrer automatiquement dès que la carte **Arduino** est alimentée.

La carte **Arduino_Uno** est également équipée d'un bouton poussoir de reset manuel. Un appui sur celui-ci permet de relancer l'exécution d'un programme si nécessaire, soit parce qu'il s'est planté soit tout simplement parce que l'on souhaite le faire repartir de son début.

5) Protections de sur intensité sur le port USB

Par mesure de sécurité pour l'ordinateur auquel sera relié l'Arduino, un fusible de protection de type polyfus est présent sur la connexion d'alimentation **5** V de la prise **USB**. Toute consommation supérieure à **500** mA provoque le déclenchement de ce fusible, protégeant ainsi le port **USB** de l'ordinateur auquel la carte est reliée. Le fusible étant de type ré-armable, il retrouvera son état normal quelques secondes après que la consommation excessive aura cessée.

I-3-4 Programmation de la carte Arduino _Uno

1) Interface de programmation

L'interface de l'**IDE** Arduino (**Figure I-4**) est plutôt simple à exploiter, elle offre une interface minimale et épurée (finie) pour développer un programme sur les cartes **Arduino**. elle est doté d'un éditeur de code **[4]** avec coloration syntaxique et d'une barre d'outils rapide. Ce sont les deux éléments les plus importants de l'interface, c'est ceux que l'on utilise le plus souvent.

On retrouve aussi une barre de menus **[1]** plus classique qui est utilisée pour accéder aux fonctions avancées de l'IDE ; une barre des Boutons **[2]** ; un onglet de fichiers ouverts **[3]** et enfin, une console **[6]** affichant les résultats de la compilation du code source.



Figure I-4 : Interface de programmation

La **Barre de Boutons** donne un accès direct aux fonctions essentielles du logiciel.

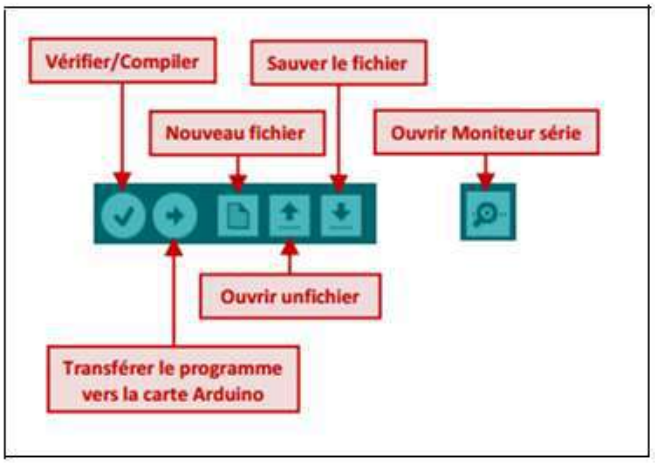


Figure I-5 : la barre des boutons

2) Structure d'un programme sous l'IDE

Il y a de nombreuses similitudes entre le **langage C** et langage **Arduino**. Tous les programmes Arduino doivent contenir au moins deux fonctions. Une première qui s'appelle **setup()** qui permet l'initialisation des entrées et des sorties, exécutée aux démarrage et une seconde **loop()** qui contient les instructions. Donc un programme ou croquis (sketch en anglais) destiné à une carte Arduino peut être représenté de la manière suivante :

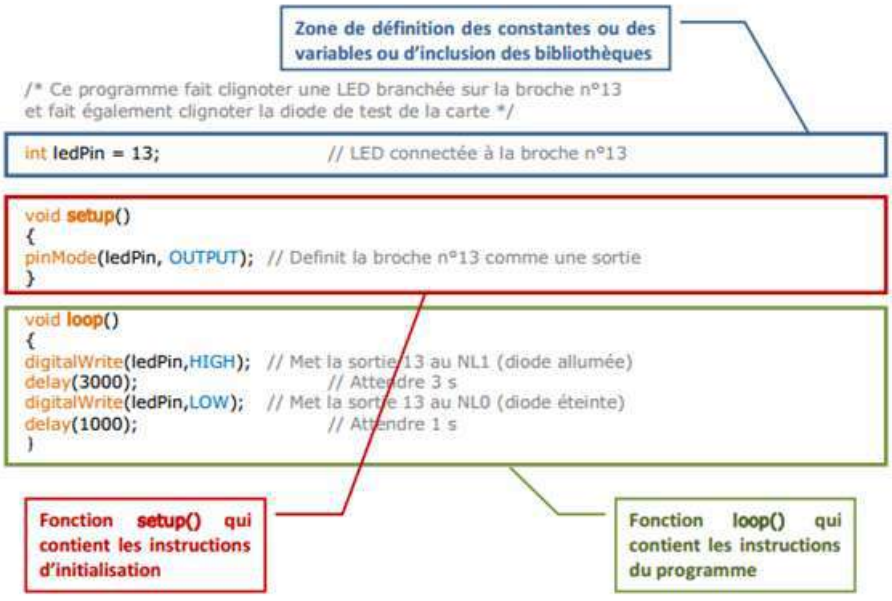


Figure I-6 : Structure d'un programme

Il est clair selon la **Figure. I-6** que l'on peut découper le programme en trois grandes structures *:*

- **Structure de définition**

La première partie du programme définit comment peut on nommer les entrées-sorties ainsi les variables et les constantes suivant les différents types de déclarations, soient en entier « **int** » ,caractère « **char** » , caractère non signé « **unsigned char** » , réel « **float** » etc.

- **Structure du void setup ()**

Elle doit suivre la déclaration des variables au tout début du programme. Il s'agit de la première fonction à exécuter dans le programme. Elle est exécutée une seule fois et sert à configurer les entrées /sorties ; initialiser la communication série, on peut même initialiser les variables et les constantes etc.

Cette structure doit contenir deux accolades entre les quelles un bloc d'instructions est inséré. Sa structure de base est :

```
Void setup ()
        {
          Blocs d'instructions;
}
```

- **Structure du Void Loop ()**

Suit immédiatement void setup et comprend le code à exécuter en continue lisant les capteurs en entrée et déclenchant les actionneurs en sortie ainsi de suite. Cette fonction est le noyau de tout programme **Arduino** et réalise l'essentiel du travail.

```
Void loop ()
        {
  Blocs d'instructions;
}
```

- **Les fonctions**

Des fonctions peuvent être insérées si besoin dans un programme afin d'éviter la répétition et pour rendre le programme plus compréhensif

. A noter que ces fonctions sont des blocs de code possédant un nom et exécutant leurs instructions à chaque appel. Une fonction doit comporter.

- **des arguments (ou variables)**
- **un nom**
- **un type**

Les types de fonctions sont :

- **void** cette fonction n'a pas d'image (elle ne renvoie aucun nombre ou booléen).
- **int** Permet de renvoyer un entier
- **float** Permet de renvoyer un réel
- **char** Permet de renvoyer une liste de caractères
- **boolean** Permet de renvoyer une variable booléenne
- **etc.**

La structure de base de ces fonctions est donnée comme suite :

```
Type nom Fonction (arguments)
        {
            Bloc d'instructions;
                    }
```

- **Les paragraphes de commentaire**

Un commentaire (sur une ligne ou sur plusieurs) est une zone de texte qui est ignoré par le programme lors de la compilation et qui est utilisé afin de décrire le code ou pour le commenter en facilitant sa compréhension. Un paragraphe de commentaire commence par **/*** et fini avec ***/**.

Exemple : /* ceci est un paragraphe de commentaire
Sur plusieurs lignes, on doit pas oublier de le refermer

***/**

- **Les lignes de commentaires :**

Une seule ligne de commentaire commence par // et fini à la fin de la ligne de code. Tout comme le paragraphe de commentaires, la ligne est ignorée lors de la compilation et n'occupe aucun espace mémoire. La ligne de commentaire est souvent utilisée pour donner des informations à propos d'une ligne.

Exemple :

```
// Ceci est une ligne de commentaires
```

I-4 Moteurs pas à pas

I-4-1 Introduction

Les moteurs pas à pas sont des transducteurs électromécaniques qui assurent la conversion des signaux électriques digitaux ou impulsionnels en mouvements de rotation.

Il existe trois types de moteurs pas à pas, les moteurs à aimant permanent, les moteurs à réluctance variable ainsi que les moteurs hybrides. Les moteurs à aimants permanents se subdivisent en deux catégories unipolaire et bipolaire.

Malgré les différences existant entre les moteurs, le résultat recherché est l'avance d'un seul pas, c'est à dire la rotation de leur axe suivant un angle déterminé à chaque impulsion que l'une ou l'autre de leurs bobines recevra. Cet angle qui varie selon la constitution interne du moteur, est en général compris entre **0.9°** et **90°**.

Les moteurs les plus couramment rencontrés présentent des pas de **0.9°** soit **400** pas par tour, **1.8°** soit **200** pas par tour, **3.6°** soit **100** pas par tour, **7.5°** soit **46** pas par tour et **15°** soit **24** pas par tour.

Il est évident que les moteurs pas à pas, de par leurs technologies, présentent une très grande précision grâce au circuit de commande qui comporte une partie logique et une commande de puissance. La partie logique détermine pour chaque pas quelles sont les bobines alimentées et le sens de rotation. La fréquence de l'horloge du circuit logique détermine la vitesse de rotation.

I-4-2 Moteur à aimant permanent

C'est le modèle dont le fonctionnement est le plus simple. Le rotor est constitué d'un aimant permanent, et le stator comporte deux paires de bobines comme le montre la **figure I-7**. En agissant sur les bobines alimentées, et le sens des courants, on fait varier le champ créé par le stator.

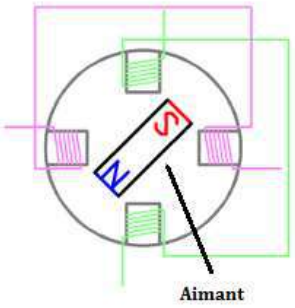


Figure I-7 Moteur à aiment permanent

Le moteur pas à pas à aimant permanent peut fonctionner selon trois types, soit en pas entier ; en fort couple ou en demi-pas.

- **Fonctionnement en pas entier**

Dans ce mode de fonctionnement, les bobines sont alimentées l'une après l'autre dans un sens puis dans l'autre voir la **figure-I-8**. L'aimant permanent suit le déplacement du champ magnétique créé par ces bobines et s'oriente selon une de ses **4** positions stables.

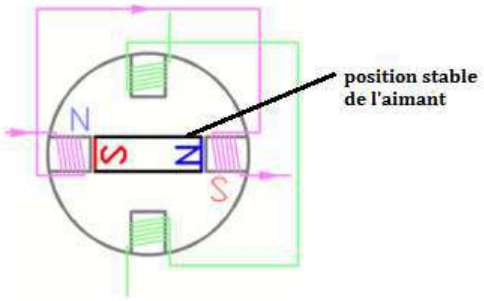


Figure I-8 Fonctionnement en pas entier

Les séquences d'alimentation des bobines sont représentées sur le tableau suivant :

Position	Bobine 1	Bobine 2	Angle
1	+1	0	0°
2	0	-1	90°
3	-1	0	180°
4	0	+1	270°

Tableau I-3 : Mode pas entier

- **Fonctionnement en mode fort couple**

Pour augmenter l'intensité du flux magnétique créé par le stator, et donc le couple moteur, on peut alimenter les deux bobines en même temps, en faisant varier uniquement le sens du courant dans chacune d'entre elles. Le rotor prendra donc également l'une des 4 positions indiquées sur la **figure I-9**, suivant le sens d'alimentation de chacune des bobines.

Les séquences d'alimentation des bobines sont représentées sur le **tableau I-4.**

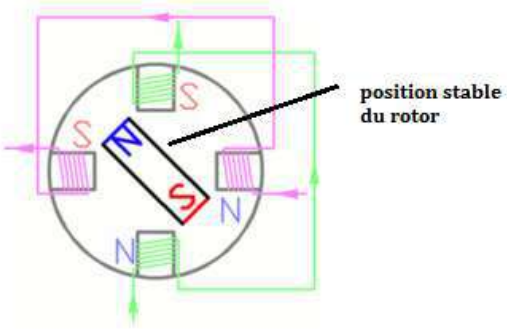


Figure I-9 Fonctionnement en fort couple

Position	Bobine 1	Bobine 2	Angle
1	+1	+1	45°
2	-1	+1	135°
3	-1	-1	225°
4	+1	-1	315°

Tableau I-4 : Mode fort couple

- **Fonctionnement en mode demi-pas**

Pour augmenter le nombre de positions stables et donc de pas du moteur à aimant permanent, on peut combiner les **2** modes précédents dans un mode de commande appelé demi-pas .

Les séquences de commande pour ce mode est données par le **tableau I-5.**

Position	Bobine 1	Bobine 2	Angle
1	+1	0	0°
2	+1	+1	45°
3	0	-1	90°
4	-1	+1	135°
5	-1	0	180°
6	-1	-1	225°
7	0	+1	270°
8	+1	-1	315°

Tableau I-5: Combinaison des deux modes

Les moteurs pas à pas à aimant permanent ont un couple moteur important, mais une résolution (nombre de pas par tour) faible, et une fréquence de rotation faible. La commande de ces moteurs pas à pas nécessite de contrôler le sens du courant dans chaque bobine.

I-4-3 Moteurs à réluctance variable (MRV)

Un moteur pas à pas à réluctance variable (**MRV**) est composé d'un rotor feuilleté en fer doux comportant un certain nombre de dents, et d'un stator également en fer doux feuilleté comportant un certain nombre de bobines. Le nombre de bobines doit être différent du nombre de paires de dents du rotor. L'alimentation de chacune des bobines va permettre la création d'un champ magnétique dans le stator, et le rotor s'oriente suivant les lignes de champ. On peut noter que lorsque le moteur n'est pas alimenté, comme il n'y a pas d'aimant permanent, le rotor peut prendre n'importe quelle position : un moteur pas à pas à réluctance variable n'a pas de couple résiduel ou couple de détente.

1) Moteurs à réluctance variable à plots statoriques non denté

Dans les moteurs pas à pas à reluctance variable à plots statoriques non denté comme le montre la **figure I-10** on a la formule qui calcule le nombre de pas par tour :

$$Npt = \frac{Nps \cdot Ndr}{Nps - Ndr} \quad \text{..... (1)}$$

Avec :

Nps : nombre de plots au stator

Ndr : nombre de dents du rotor

Npt : nombre de pas par tour

Dans ce type de moteur le calcul du nombre de pas par tour est :

Npt= (6x4) / (6 - 4)= 12 pas par tour

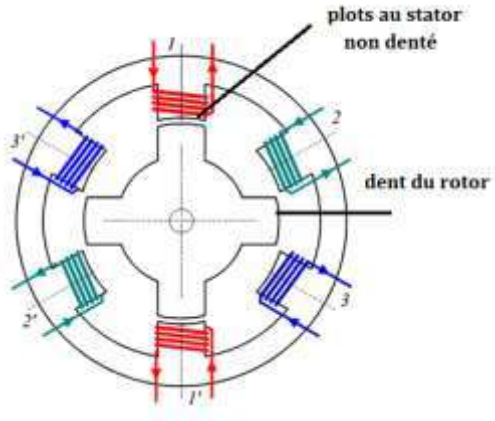


Figure I-10 Moteur à réluctance variable à plots statoriques non denté

2) Moteurs à réluctance variable à plots statoriques dentés

Dans les moteurs pas à pas à réluctance variable à plots statoriques dentés, le rotor est uniformément denté et les plots du stator portent également un certain nombre de dents **Ndps**, dont le pas dentaire est identique à celui du rotor sachant que les plots du stator sont répartis symétriquement voir la **figure I-11**.

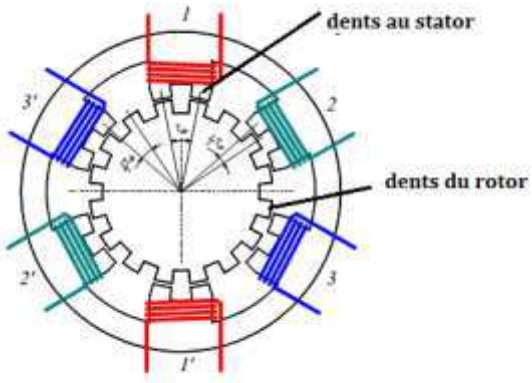


Figure I-11 Moteur à réluctance variable à plots statoriques denté

En vertu de la relation (1) le calcul du nombre de pas par tour est :

Npt= 48

I-4-4 Moteurs hybrides

En superposant le principe du moteur à réluctance variable et à aimant permanent, on obtient un moteur hybride qui combine les avantages des deux moteurs. Le rotor est constitué de deux disques dentés décalés mécaniquement. Entre ces deux disques est inséré un aimant permanent. Le nombre de dents du rotor est différent de celui du stator. Lorsqu'on alimente une bobine, le rotor place les dents Nord et Sud de telle façon à ce que le flux traversant le rotor soit maximal.



Figure I-12 moteurs pas à pas hybrides

I-4-5 Comparaison des trois types

Les valeurs suivantes nous montrent la différence entre les trois types de moteurs vis-à-vis leurs cout ; leur vitesse ; leur rendement et leur précision etc.

	Aimant permanent	**Hybride**	**Reluctance variable**
Coût	Faible	élevé	Moyen
Vitesse	Bas	Très haut	Haut
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	Faible	bonne	Moyenne

Tableau I-6 : Comparaison des trois types

I-5 Le capteur optique CNY70

Le **CNY70** est un capteur optique à réflexion, Il est composé d'une diode infrarouge et d'un phototransistor avec un filtre contre la lumière visible. Il est utilisé dans les applications de détection de contraste à faible distance

Le capteur optique par réflexion **CNY70** se présente sous la forme d'un boîtier noir cubique avec quatre broches d'un coté et deux rond translucide de l'autre.



Figure I-13 Illustration du capteur **CNY70**

Il s'agit d'un émetteur et d'un récepteur infrarouge .L'émetteur et le récepteur sont accordés sur la même longueur d'onde. L'émetteur est une diode infrarouge centrée sur la longueur d'onde **950 nm**. Le récepteur est un phototransistor, disposé de sorte que le faisceau émis par l'émetteur lui soit réfléchi par une surface blanche plane perpendiculaire au capteur à une distance **d** de **1** à **5** mm environ, comme l'indique la **Figure I-15.**

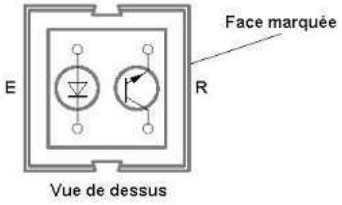


Figure I-14 Brochage interne du capteur

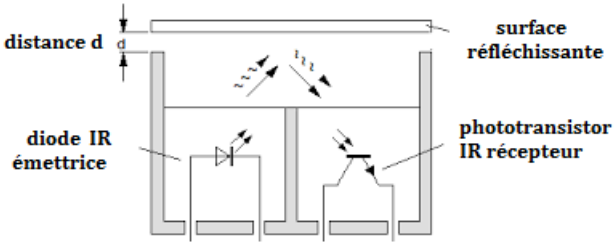


Figure I-15 Démonstration de la réflexion

La figure I-15 montre qu'en présence d'une surface réfléchissante blanche placée à une distance **d** comprise entre **1** et **5mm** (voir caractéristiques en **Annexe**) le phototransistor récepteur va conduire du courant électrique chose qui va permettre de faire sortir de son collecteur ou son émetteur un niveau logique **0** ou **1** selon la circuiterie avec le quel est monté le capteur.

Chapitre II

Construction mécanique sous SolidWorks

II-1 Introduction

L es moyens informatiques modernes et les logiciels dits de conception assistée par ordinateur constituent des moyens puissants de calcul et d'analyse des systèmes.

Ces moyens modernes fascinent par leur facilité, mais ils ne déchargent pas le Concepteur de toute réflexion ; en effet le concepteur ne peut pas modéliser ce qu'il ignore. Parmi ces logiciels on trouve le **SolidWorks.**

II-2 Le logiciel SolidWorks

SolidWorks est un modeleur 3D utilisant la conception paramétrique. Il est souvent utilisé dans le cadre professionnel. Il permet de modéliser des pièces variées, des surfaces, et de les assembler par la suite. Il génère trois types de fichiers relatifs à trois concepts de base :

- la pièce
- l'assemblage
- la mise en plan.

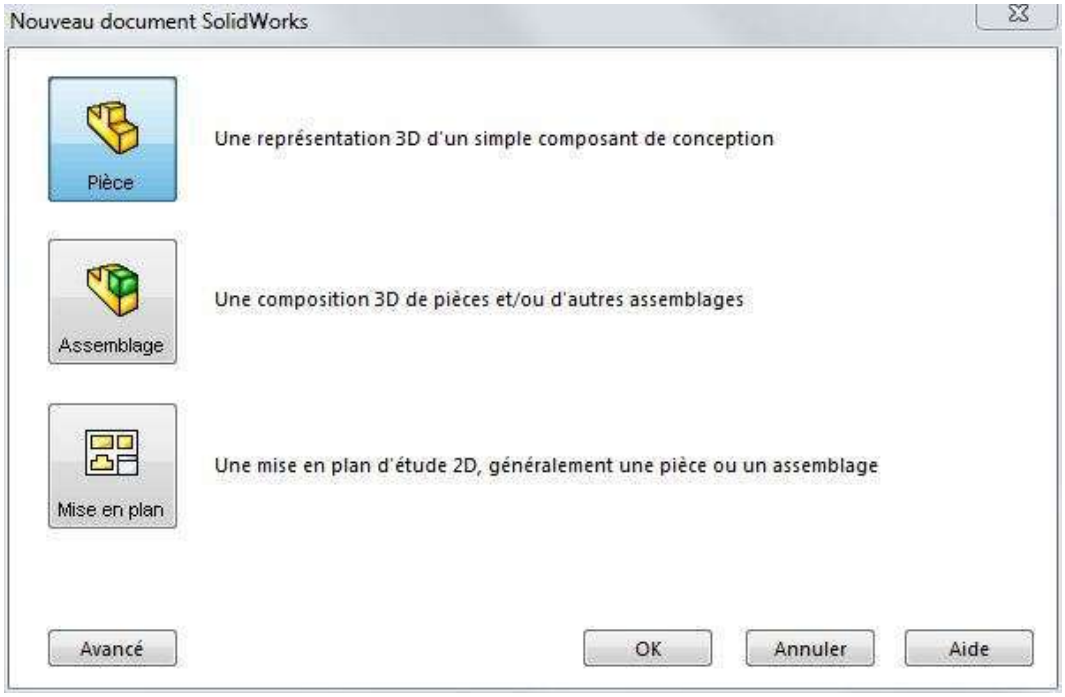


Figure II-1 Fichiers assurés par SolidWorks

Ces fichiers sont en relation. Toute modification à quelque niveau est répercutée vers tous les fichiers concernés. Le fonctionnement et l'exploitation de ces trois fichiers peuvent être explicités comme suit :

❖ Les pièces

La pièce est l'objet 3D monobloc. La modélisation d'une telle entité dépendra de la culture de l'utilisateur. Une pièce est la réunion d'un ensemble de fonctions volumiques avec des relations d'antériorité, des géométriques, des relations booléennes (ajout retrait).

Cette organisation est rappelée sur l'arbre de construction. Chaque ligne est associée à une fonction qu'on peut renommer à sa guise.

❖ Les assemblages

Les assemblages sont obtenus par la juxtaposition de pièces. La mise en position de pièces est définie par un ensemble de contraintes d'assemblage associant, deux entités respectives par une relation géométrique (coïncidence, tangence, coaxialité, parallèle etc.).

Comme pour les pièces, la gestion de l'ensemble est portée par un arbre de création qui donne accès à l'arbre de création de chaque pièce. Il est d'ailleurs possible d'assembler des assemblages, donc de former des sous-groupes de pièces. Cette opération étant préalable ou en cours d'édition. L'intérêt de cet outil, c'est qu'il rend possible la création d'une pièce dans l'assemblage.

❖ Les mises en plan

Les mises en plan concernent à la fois les pièces (dessin de définition) ou les assemblages (dessin d'ensemble. Le logiciel 3D ne fait qu'une projection de l'objet. Les modifications éventuelles sont opérées sur l'objet représenté, et ne concernent pas directement le plan.

Après avoir choisi le fichier Pièce par exemple, on aura l'interface donnée par la **figure II-2** portant les numéros **1** à **6** :

1. Boutons "Nouveau", "Ouvrir", et "Enregistrer".
2. Les Options !
3. Le bandeau principal, que nous utiliserons le plus souvent.
4. Diverses icônes : Vue en coupe, Zoom, Apparence, Vues, ...
5. L'Arbre de conception.
6. Un autre bandeau, que l'on utilisera pour les surfaces.

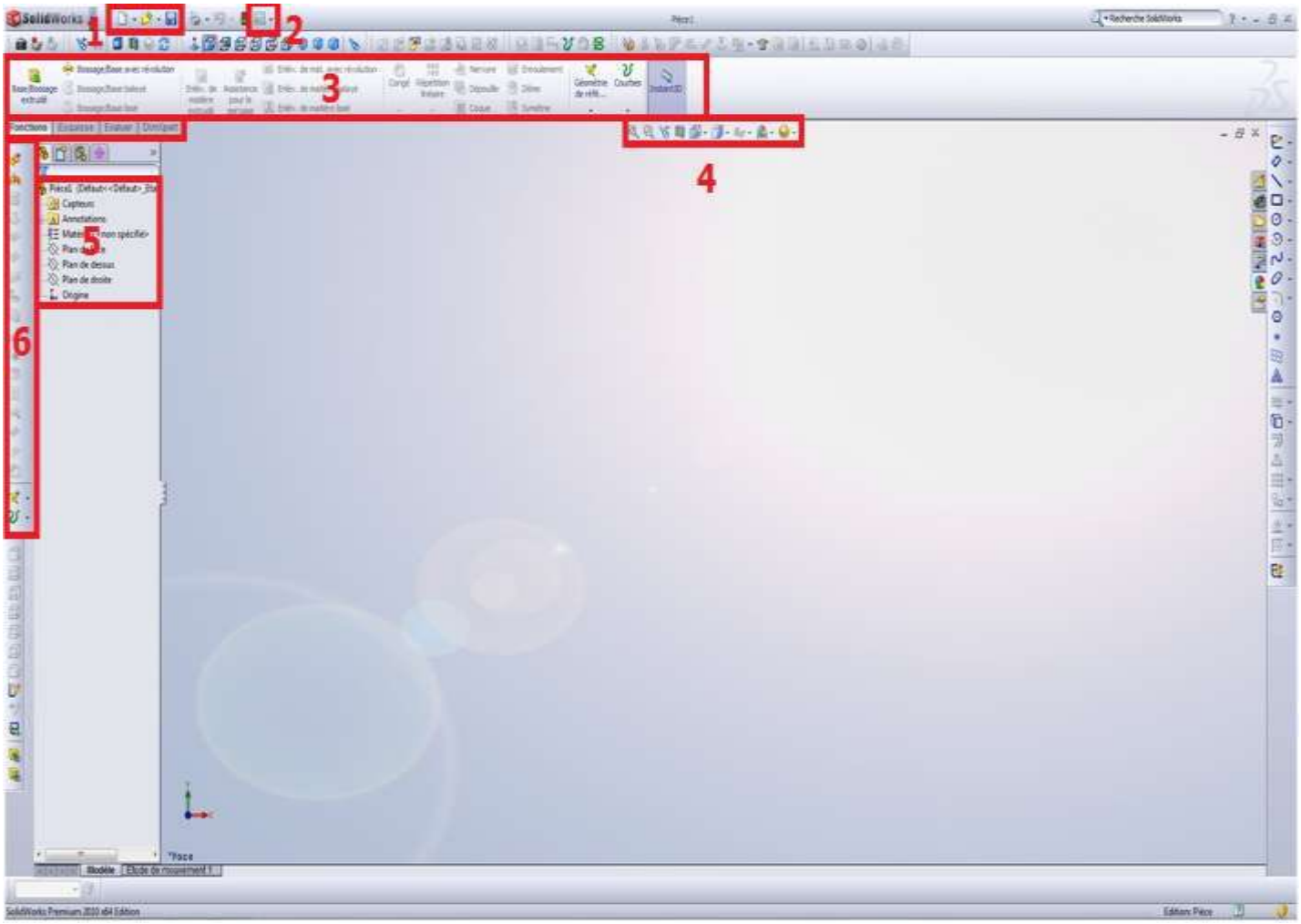


Figure II-2 Interface du fichier pièce

En outre, **solidworks** assure plusieurs **fonctions** principales qui sont :

- **Les esquisses** : Ce sont des petits dessins qui donnent le profil d'une pièce.
- **La modélisation volumique :** Permettent de donner du volume aux esquisses.
- **La modélisation surfacique :** Permet de créer des surfaces définis par des esquisses et possédants des épaisseurs nulles.
- **La tôlerie :** Elle consiste à générer une plaque de tôle, et de la tordre, de la découper etc.
- **L'assemblage :** Explicité précédemment.
- **L'animation :** Permet de réaliser des animations concernant l'objet.

En se basant sur tous ce qui a été évoqué auparavant, la construction mécanique sous **SolidWorks** ainsi que le dimensionnement de notre banc de traitement à réaliser consiste en premier lieu en la justification de cotation dimensionnelle de chacune des pièces constituantes sachant que ces dimensions linéaires nous permettent de définir complètement et sans ambigüité les pièces concernées. La figure suivante réalisée sous SolidWorks montre les différentes cotes du banc.

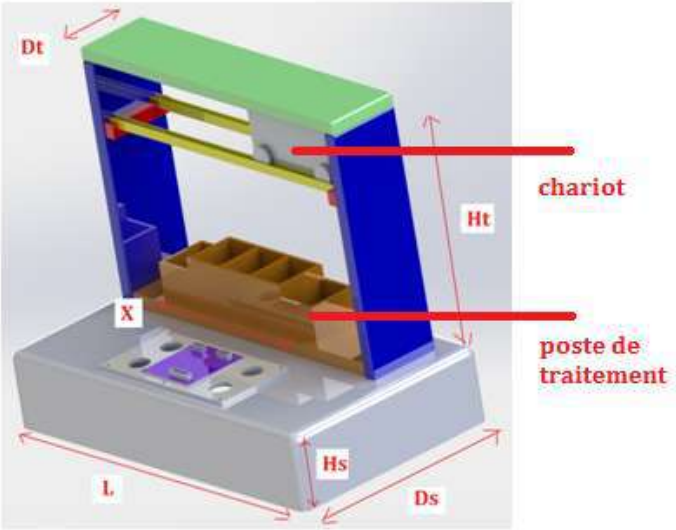


Figure II-3 Les cotes de la maquette

Les cotes mentionnées sur la **figure II-3** X ; L ; HS ; DS ; Ht et Dt sont justifiées dans les sous titres qui s'en suivent:

II-3 poste de traitement (Pièce 1)

Toute la cotation de notre maquette se base essentiellement sur celle du poste de traitement qui est constitué de trois bacs et de deux postes, un pour chargement et l'autre pour déchargement voir la **figure II-4**.

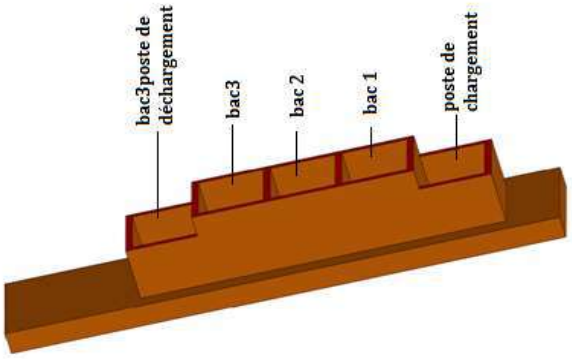


Figure II-4 Image 3D du poste de traitement sous **SolidWorks**

Dans les **figure II-5** et **figure II-6**, la vue de face et de dessus montrent les différentes cotes des trois Bacs ainsi que les deux postes ce qui nous permet d'en conclure la longueur finale du poste de traitement qui est de l'ordre de **273mm** ; sa largeur qui est égale à **70mm** et la hauteur correspondante égale à **60mm.** C'est à partir alors de ces dimensions que notre réalisation est structurée.

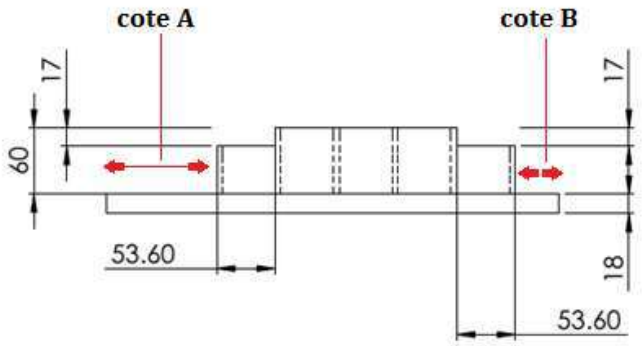


Figure II-5 Vue de face du poste de traitement sous **SolidWorks**

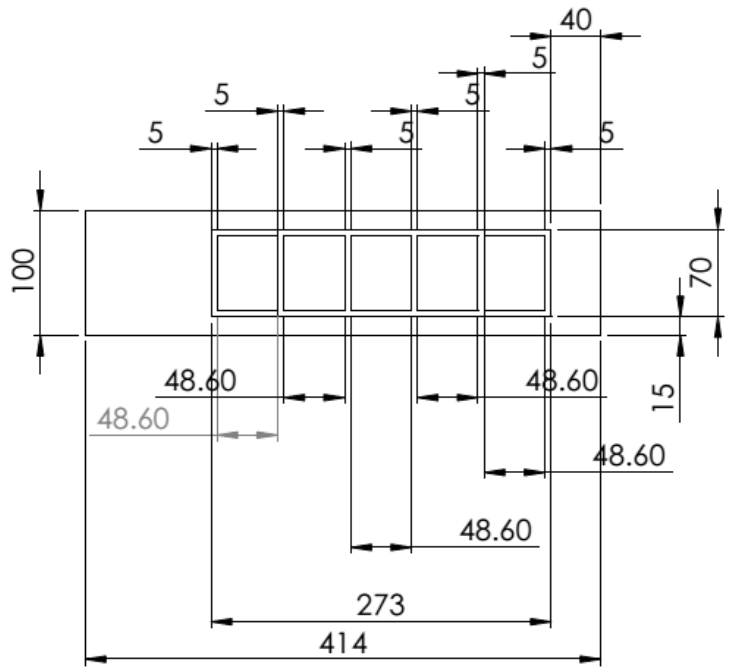


Figure II-6 Vue de dessus du poste de traitement sous **SolidWorks**

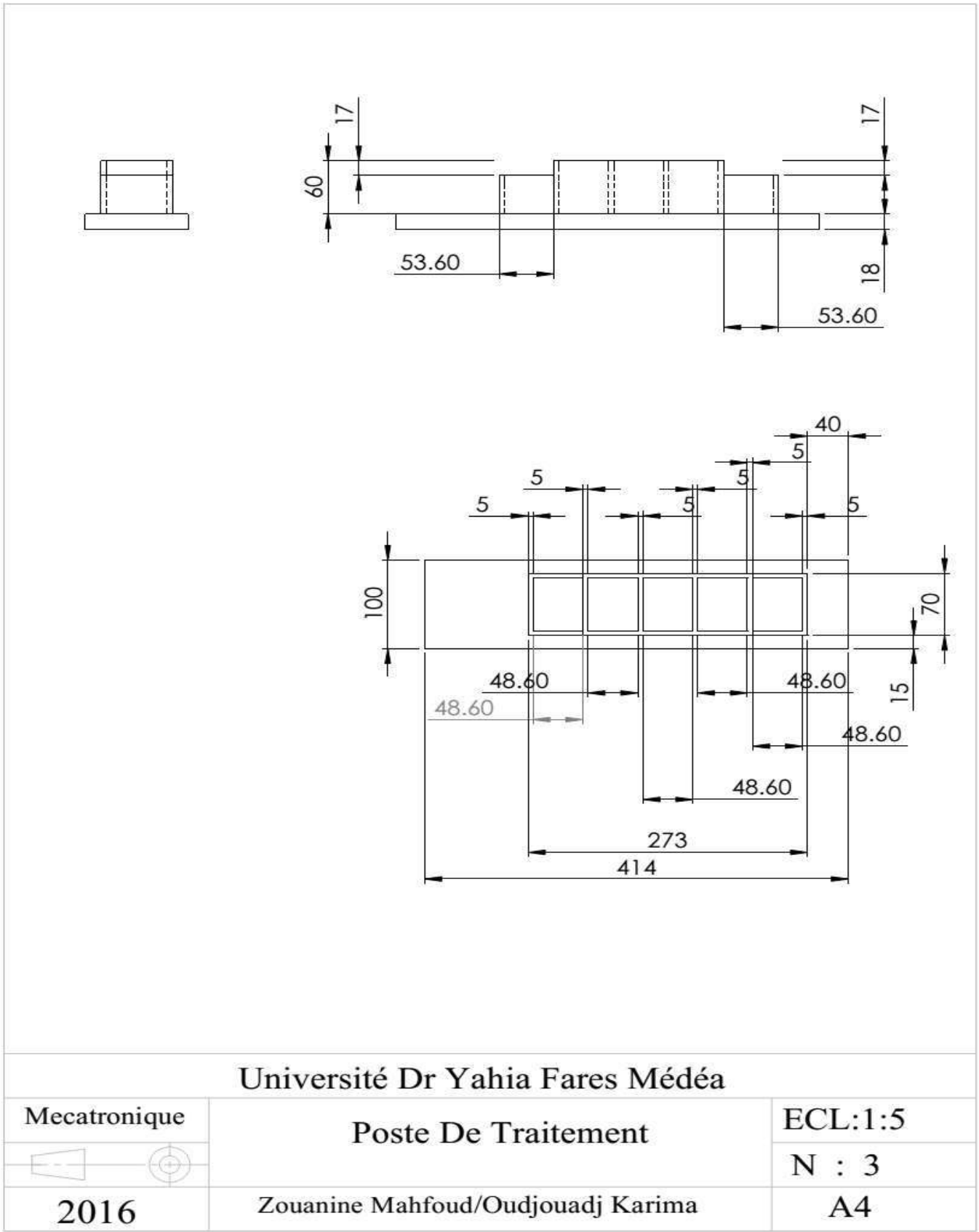


Figure II-7 Cotation du poste de traitement

II-4 Le socle et le chariot (Pièce 2 et Pièce 3)

Les cotes de la base du socle ou on doit poser la **pièce1** peuvent être déterminées Et justifiées comme suit :

Commençons par les cotes **A** et **B** qui se trouvent à droite de poste de chargement et à gauche de poste de déchargement (**voir figure II-5**) qui sont justifiées de sorte à ce que lors de déplacement du chariot la pincette associée pourra atteindre les points extrêmes (point de chargement et point de déchargement) de ces deux postes. Sachant qu'on a prévu deux moteurs à l'intérieure du chariot et selon leur dispositions les dimensions de ces derniers sont choisies comme suit :



Figure II-8 Dimensions des moteurs utilisés

En se basant sur les dimensions des moteurs le chariot pourra avoir la cotation suivante :

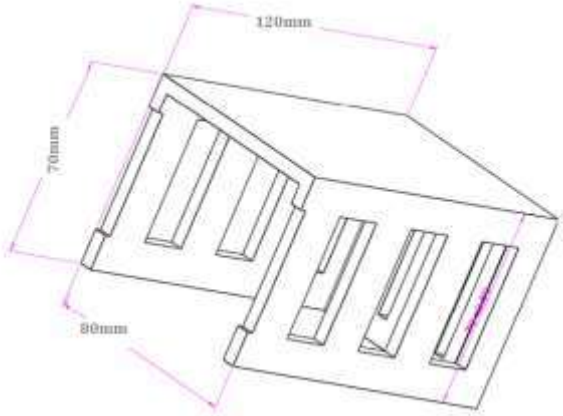


Figure II-9 Dimensions du chariot

En conséquence, la longueur est égale à **120mm** ; la largeur **80mm** et la hauteur **70mm.** La longueur du chariot **120mm** exige à ce que les cote **A** et **B** doivent mesurer au minimum **40 mm** et **101mm** respectivement, en l'occurrence la longueur finale de la **pièce 1** est égale:

Longueur finale=273 + 101 +40 = 414 mm

La **figure II-11** faite sous SolidWorks nous montre en détaille la cotation de la base du socle qui mesure **510mm**x**300mm**.

510mm=414 +18x2 +2X18 +12x2

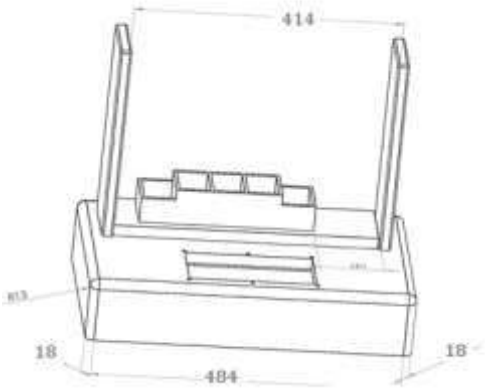


Figure II-10 cotation du socle en perspective sous

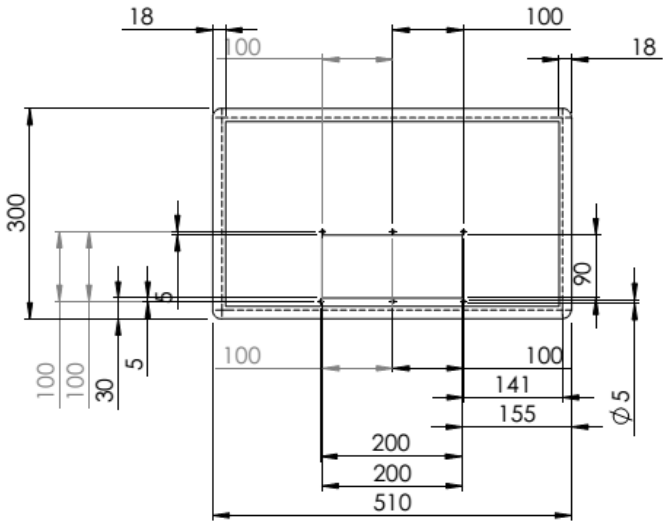
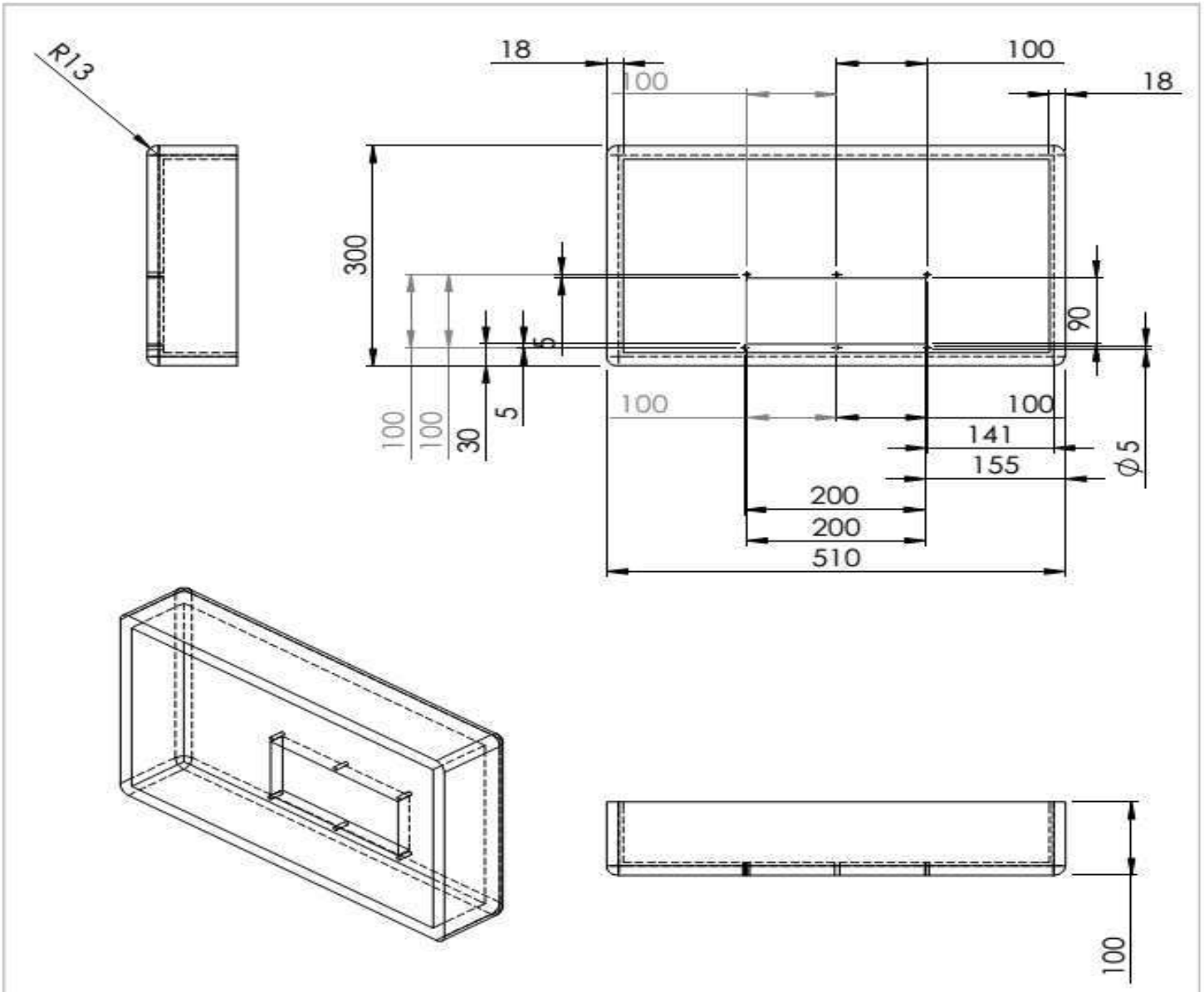


Figure II-11 Vue de dessus du socle

La hauteur est égale **HS= 100mm**, valeur suffisante et permettant d'insérer toutes les cartes électroniques à réaliser à l'intérieur du socle.



Université Dr Yahia Fares Médéa		
Mecatronique	Le Socle	ECL:1:7
		N : 1
2016	**Zouaouine** Mahfoud/Oudjouadj Karima	A4

Figure II-12 Cotation du socle

II-5 traverse du coté droit et du coté gauche (Pièce 4 et Pièce 5)

Nous avons choisi la hauteur **h1=130mm** (voir la **figure II-13**), hauteur qui nous permet de faire monter ou descendre la pincette sans difficulté et sans gène.

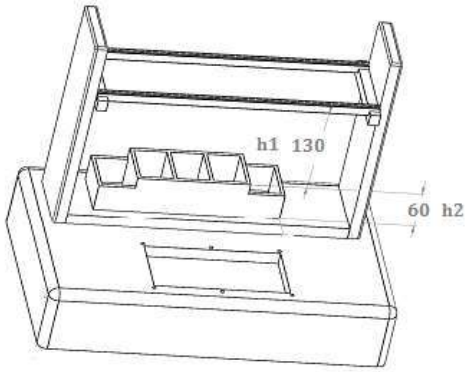


Figure II-13 Hauteur entre la pièce 1 et les rails

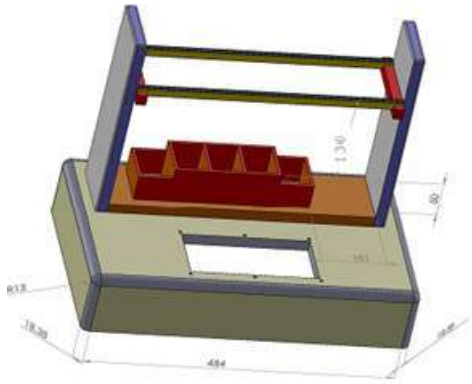


Figure II-14 vue en image ombrée

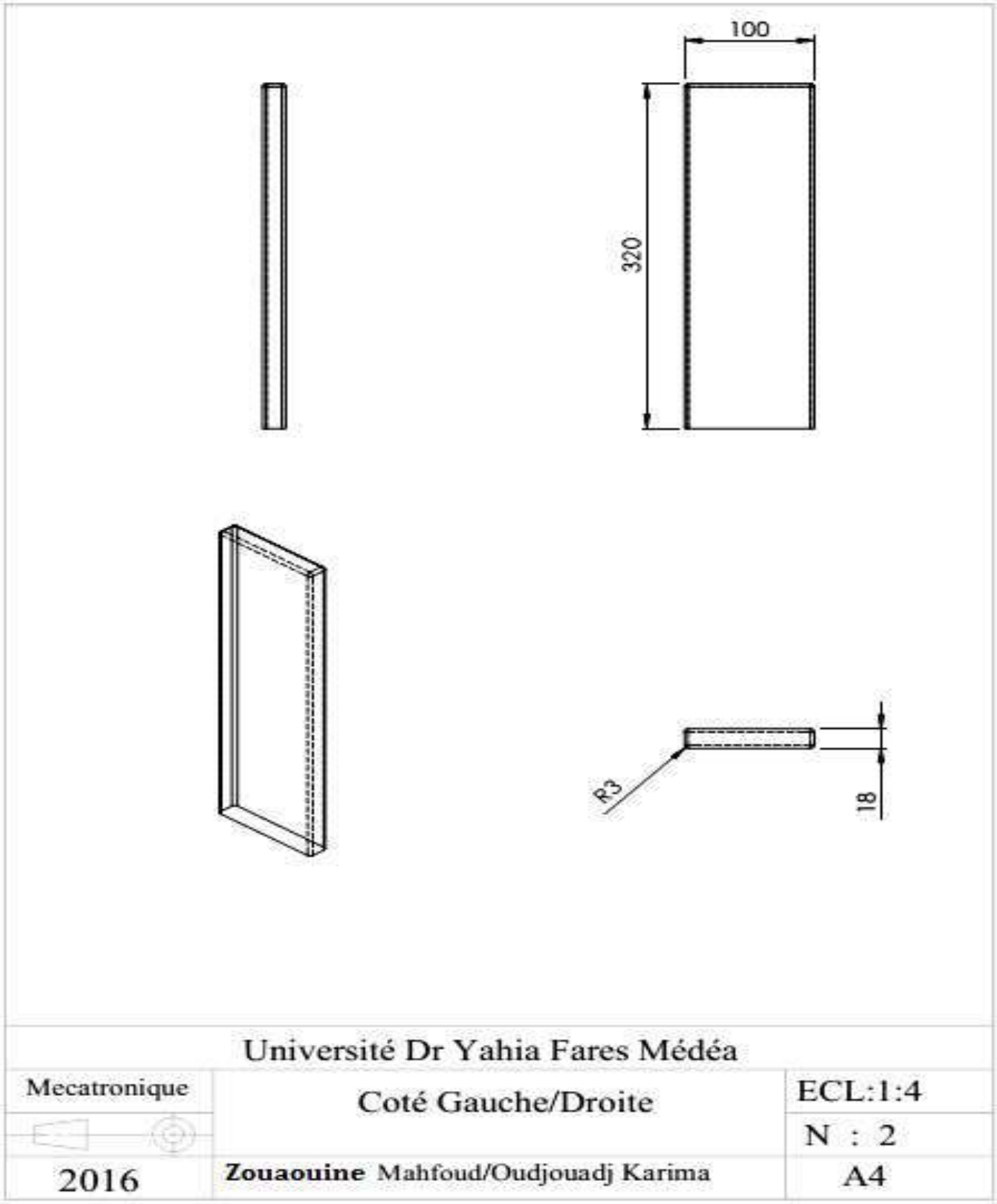


Figure II-15 Cotation des cotés droite et gauche

II-6 Conclusion

Dans ce chapitre de construction mécanique nous avons exploité un logiciel très utilisé notamment dans le domaine mécanique qui est **solidworks**, certes nous n'en avons pas beaucoup détaillé mais on a pu arriver à construire notre banc de traitement de surface utilisant ce logiciel. En l'occurrence nous avons exploité ses deux fichiers (fichier **Pièce** et fichier **Assemblage**) pour la modélisation du banc en **3D,** tel que le **fichier Pièce** nous a permet de construire séparément les différentes pièces alors que le **fichier Assemblage** consiste en l'union des différentes pièces ainsi réalisées.

Chapitre III
Etude Technique

III_1 Introduction

Dans ce chapitre nous allons étudier l'aspect théorique détaillée de la maquette ; en premier lieu nous allons commencer par une étude électrique de toutes les cartes avec un choix judicieux des composants et en deuxième lieu on va passer à la programmation en utilisant deux technologies différentes (Ardions-uno et siemens).

III_2 Etude électrique

III_2_1 Schéma synoptique

Le schéma synoptique donné à la **Figure III-1** montre qu'il s'agit d'une représentation graphique constituée de plusieurs blocs, en effet chacun de ces blocs sera le siège d'une fonction électrique bien explicite.

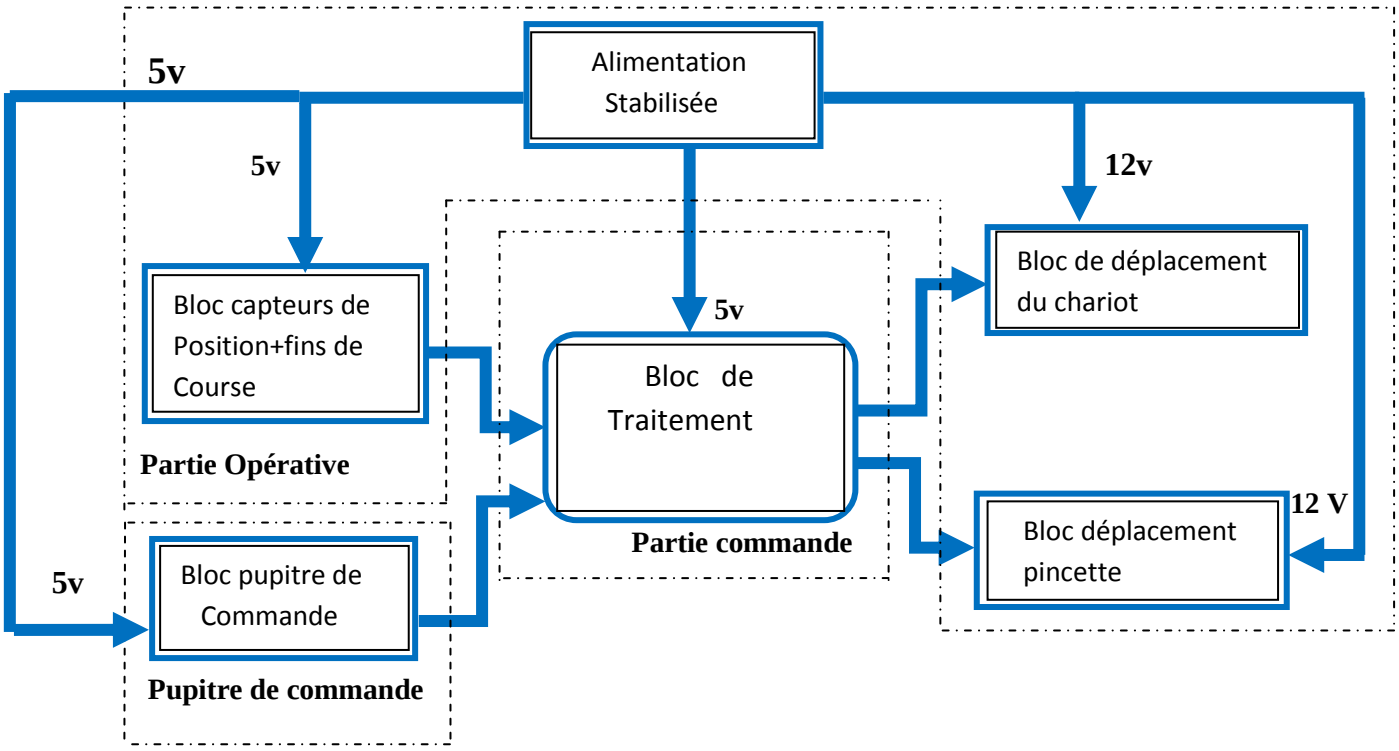


Figure III-1 : Schéma synoptique

Le premier bloc à exposer est l'approvisionnement en énergie électrique de tous les autres blocs, telle que cette fonction d'alimentation fournit deux tensions nécessaires au bon fonctionnement du projet **5** volts et **12** volts.

Le deuxième bloc à aborder ,bloc capteurs, il représente un étage d'acquisition de données nécessaires ;il acquière certaines informations concernant les différentes positions du chariot lors de son déplacement de gauche à droite et vis versa, pour ce faire on a prévu cinq capteurs sensés à détecter cinq positions principales du chariot selon les différents postes de traitement que contient notre banc, à savoir un poste de chargement qui se trouve à droite de la maquette , un poste de déchargement qui se situe à l'extrémité gauche et trois bacs dans les quelles on doit tremper les pièces à traiter, cette dernière fonction s'effectue à l'aide d'une pincette et de deux capteurs fins de course , toutes ces données seront transmises au bloc de traitement .

Le troisième bloc joue un rôle très important, dans la mesure où il traite toutes données provenues du bloc précédent, comme il peut gérer toute les sorties des blocs qui s'en suivent .par conséquent ce bloc traite ces informations selon un programme qu'on va loger dans la mémoire programme du microcontrôleur, en effet nous allons choisir pour cet étage une carte de développement électronique très répandue dans le domaine micro-électronique c'est la carte Arduino_Uno.

Les deux blocs de déplacement sont conçus pour faire déplacer un chariot à droite ou à gauche ainsi pour faire monter ou descendre une pincette que contient le chariot. Il est à motionner que le déplacement du chariot et de la pincette est assuré par des moteurs pas à pas.

En outre, le déplacement du chariot porte avec lui des pièces dont on veut tremper dans les bacs en vue de les traiter, tel que en premier lieu il fait charger de pièces , au poste de chargement ,à l'aide de la pincette puis il se déplace selon un programme préalablement établi pour accomplir un processus de traitement.

Le dernier bloc est le pupitre de commande qui contient des boutons poussoirs pour gérer le mode manuel et pour donner l'impulsion de départ pour le mode automatique donc il est considérer comme entrée pour le bloc de traitement .

III_2_2 Schéma électrique

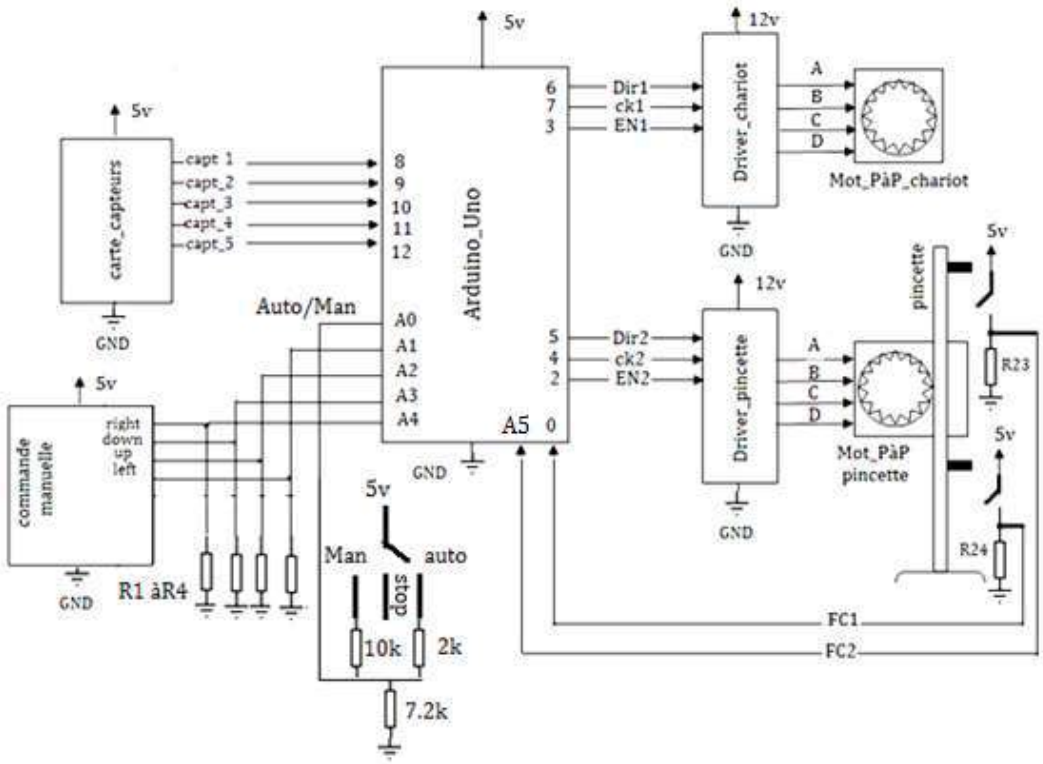


Figure III-2 : Schéma électrique global

Explication du schéma électrique :

Le chariot est déplacé à l'aide du moteur pas à pas « Mot_pàp_chariot » via le driver_chariot, ce dernier nécessite trois lignes de commande **dir1**, **ck1** et **EN1** générées par arduino tels que :

- **Dir1 :** permet de changer le sens de rotation du moteur.
- **Ck1:** signal d'horloge servant comme signal de rotation du moteur et sa fréquence Pour faire varier la vitesse de rotation.
- **EN1:** entrée de validation des sorties (**A, B, C** et **D**) du driver.

La montée et la descente de la pincette, entre les deux fins de courses **FC1** et **FC2**, sont faites grâce au moteur pas à pas « Mot_pàp_pincette » via le driver pincette ; ce dernier nécessite aussi trois lignes de commande **dir2, ck2** et **EN2** générées par Arduino et qu'elles remplissent les mêmes fonctions décrites par **dir1, ck1** et **EN1**.

La carte capteurs qui est composée de cinq capteurs (capt_**1** à capt_**5**) connectés respectivement aux pins **8** à **12** de l'Arduino, sert à détecter la position du chariot lors de son déplacement. Arduino gère les drivers en se référent aux informations transmises par la carte capteurs.

La commande manuelle du chariot et de la pincette est assurée par un pupitre possédant cinq boutons (bouton **right_Dcycl** ; bouton **left** ; bouton **up** ; bouton **down** et bouton **Auto/Man**) reliés respectivement aux pins **A0** à **A4** de l'Arduino. Les résistances **R1** à **R5** servent comme résistances de rappel de manière à ce que les entrées ne soient pas flottantes lorsqu'aucun des boutons n'est sollicité.

III-2-3 Etude électronique des cartes

1- Alimentation

Notre maquette est approvisionnée en énergie électrique par une alimentation stabilisée externe de **12 v,** et pour avoir le 5V on a utilisé le régulateur 7805 tel que les deux drivers sont alimentés de **12 v** alors que les autres modules sont alimentés de **5 v.**

2- Carte de commande manuelle "pupitre de commande"

Le schéma est composé de cinq boutons poussoirs (**Bp1** à **Bp5**) dont chacun est relié d'une part et d'autre à **+5volt** et **GND** via une résistance de rappel (pull up), en l'occurrence un appui sur l'un ramène un niveau **BAS** à l'entrée analogique correspondante de l'arduino.

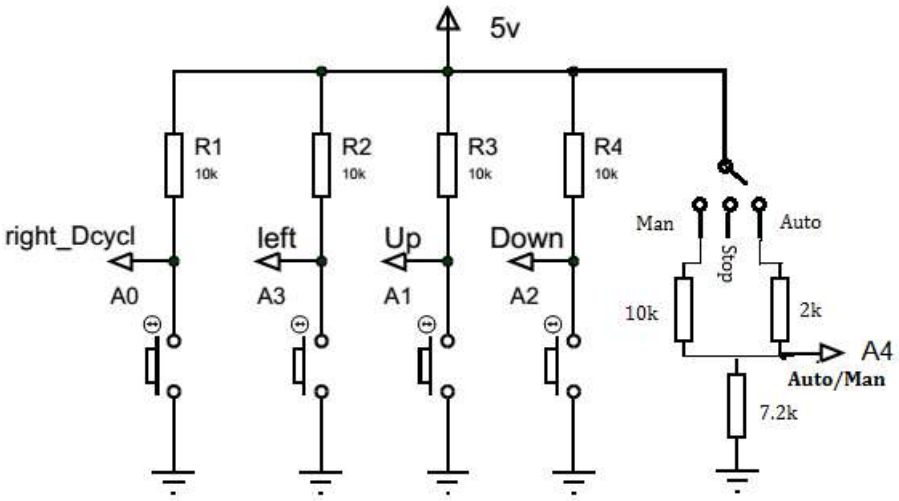


Figure III-3 : Schéma électrique du bloc de commande manuelle

En effet, ce qui est illustré sur la **Figure III-3** l'interconnexion des boutons à l'arduino uno sont Telles que:

Boutons	Fonction	Arduino
Bp1	Right_Dcycl	A0
Bp2	Left	A3
Bp3	Down	A1
Bp4	Up	A2
Bp5	Auto/Man	A4

Tabeau III -1 connexion des boutons à l'arduino uno

Bp1:permet de déplacer le chariot à droite ainsi comme départ cycle pour le mode Automatique et le mode manuel

Bp2:permet de déplacer le chariot à gauche

Bp3:permet la descente de la pincette

Bp4:permet la montée de la pincette

Bp5: Mode de fonctionnement automatique ou manuel

3-Carte des capteurs de position

Le schéma de la **Figure III-4** intègre cinq capteurs de type **CNY70** chacun de ces capteurs est Susceptible de détecter la position du chariot en lui indiquant là où il faut déposer ou de soulever les pièces à traiter.

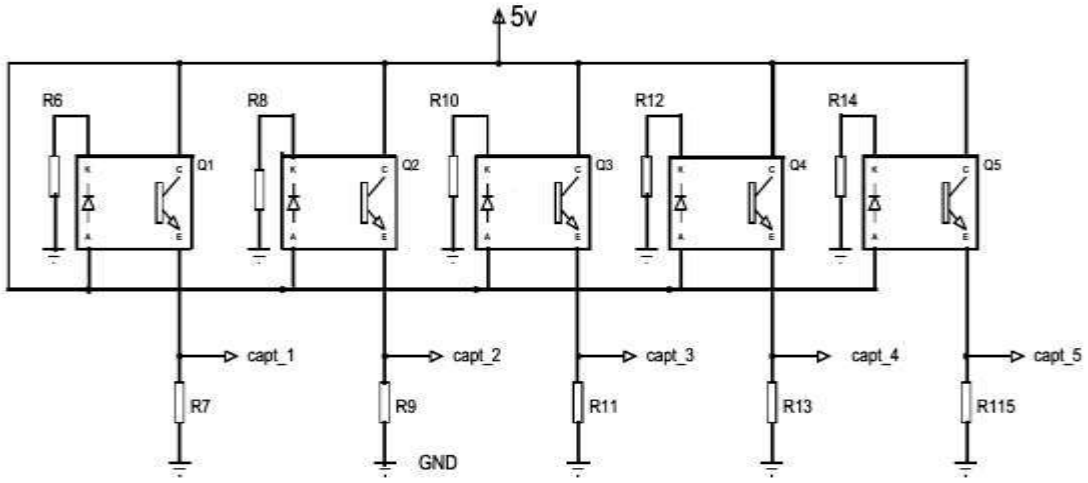


Figure III-4 : Schéma électrique du bloc capteurs

Le principe de fonctionnement de l'opto-coupleur **CNY70** réside dans la réflexion des rayons ultra- violets délivrés par la diode émettrice en présence du chariot ce qui va permettre en conséquence au transistor récepteur de l'opto-coupleur de se saturer en délivrant une tension de **5v** à son émetteur (sortie **capt_1**) pour le premier capteur.

Les résistances de polarisation de l'opto-coupleurs sont calculées de la manière suivante:

Calcul de R6 :

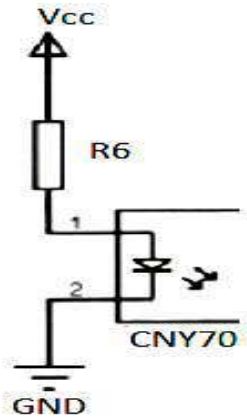


Figure III-5 : Polarisation de la LED émettrice du CNY70

R6 résistance permettant de polariser la diode émettrice de l'opto-coupleur **CNY70,** d'après Les valeurs caractéristiques données par le Datasheet de cet opto-coupleur (voir **Annexe page- XIX-**

-Courant max dans la diode **IDmax**=50mA
-Tension aux bornes de la diode **VD**=1.25v

On prend, par raison de protection de la diode, **ID**=30mA.

R6 = (Vcc-VD) / ID = (5-1.25) / 30 = 125 Ohm

Prenons une valeur normalisée pour la résistance R6.

R6=120 Ohm

Les valeurs des résistances R8, R10, R12 et R14 sont calculées de la même manière.

Calcul de R7

La résistance **R7** polarise l'opto- transistor pour qu'il soit saturé lors de rayonnement ultra_violet ,on écrit alors :

R7 = (Vcc-Vce sat) / ICsa { **Vce sat**=0.3v valeurs caractéristiques ; **ICsat**=1.56mA

On en déduit alors : R7**=(5-0.3)/1.56**=3 kOhm

R7 normalisée :

R7=2.7 KOhm

Les valeurs des autres résistances R9 ; R11 ; R13 et R15 sont calculées de la même façon

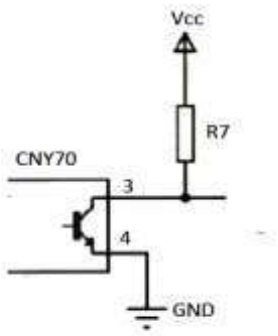


Figure III-6 : polarisation du recepteur du CNY70

4-Carte Driver

Le circuit de commande de moteur pas à pas **L297** est prévu pour être utilisé avec le circuit de puissance **L298** .Il reçoit les signaux de contrôle en provenance de la carte Arduino_Uno et génère tous les signaux nécessaires pour l'étage de puissance (**L298**). Les signaux de contrôle sont :

- Mode direction **dir1/dir2** qui a pour rôle d'inverser le sens de rotation du moteur.
- Mode pas / demi-pas (HALF/ FULL) non utilisé, la broche approprié est mise à **5v**
- Entrée clock **ck1/ck2** dont la fréquence appliquée influe sur la vitesse de rotation du

moteur.

- Entrée Enable **EN1/EN2** qui a pour rôle l'activation du circuit.
- Entrée **CONTROL** du circuit **L297** : quand elle est mise à **0 volt**, elle crée plus d'échauffement Sur le pont **L298**, les résistances shunt et sur les moteurs mais elle permet un fonctionnement rapide des moteurs. Mais quand elle est mise à **5volt**, elle crée moins d'échauffement et en revanche elle ralentie un peu les moteurs.

Ce circuit (**l297**) inclue deux régulateurs à découpage type PWM (modulation de largeur d'impulsion) pour réguler le courant dans le moteur pas à pas. En effet, La broche vref (pin **15**) peut servir comme entrée de réglage d'intensité, Par conséquent les résistances **R20** et **R21** (de L'ordre de **0.5** Ohm) sont des résistances de mesure de courant dans le moteur. Si l'une des tensions à ces bornes **Usens1** ou **Usens2** dépasse **Vref** le circuit **L297** modifie le rapport cyclique du régulateur concerné en un temps limite de façon à faire chuter la tension **Usens 1/Usens2** et donc la limitation du courant dans le moteur pour ne peut pas dépasser la valeur maximale.

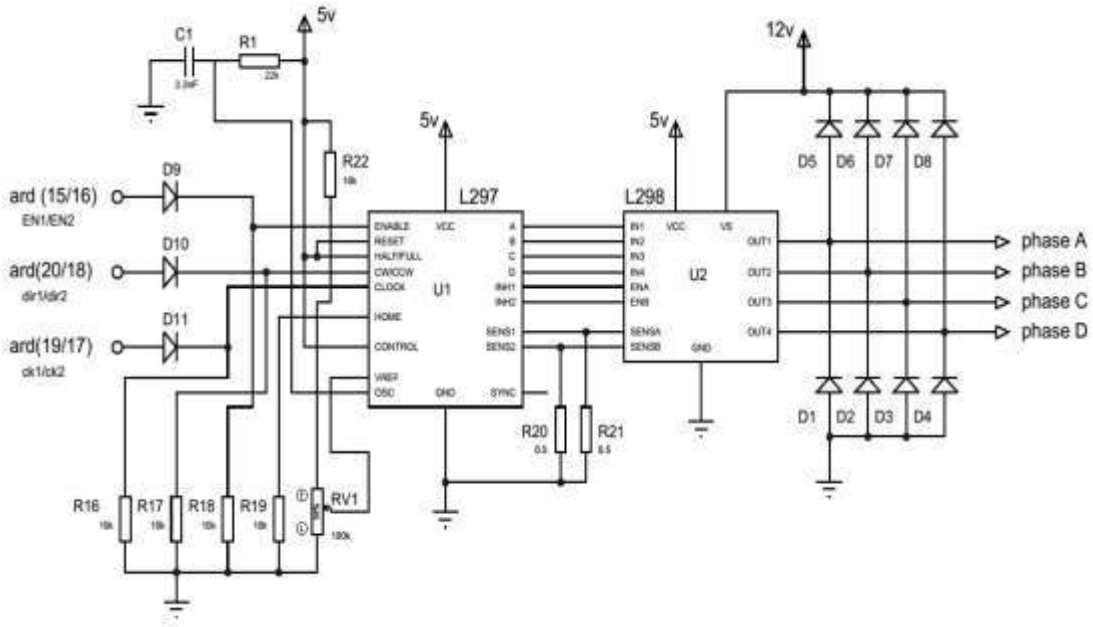


Figure III-7 : schéma électrique du circuit de commande du moteur pas à pas

Les entrées de commande du driver **EN1/EN2** ; **dir1/dir2** et **ck1/ck2** provenues de la carte Arduino_Uno ont pour rôles respectivement l'activation du circuit **L297** ; changement de sens de rotation et en fin la rotation du moteur.

Le moteur pas à pas est connecté dans l'ordre des phases aux sorties de l'étage de puissance représenté par le circuit **L298**.

En l'occurrence , le choix de la résistance **R22** et du potentiomètre **RV1** est justifié comme suit :

R20=**R21**=**0.5** ohm

Courant max **Imax**=**2A**

On aura **Usens1**= **Usens2= R21* Imax =1**v

Par conséquent, on doit ajuster **RV1** de sorte à avoir **Vref**=**1v** d'où le pont diviseur constitué par **R22** et **RV1** permet de fixer par exemple **R22** à **10k** ce qui donne **RV1=100k** ces valeurs donnent une plage de variation de la tension entre **0v** et **4.5 v** ce qui nous permet facilement d'ajuster la valeur de **Vref**.

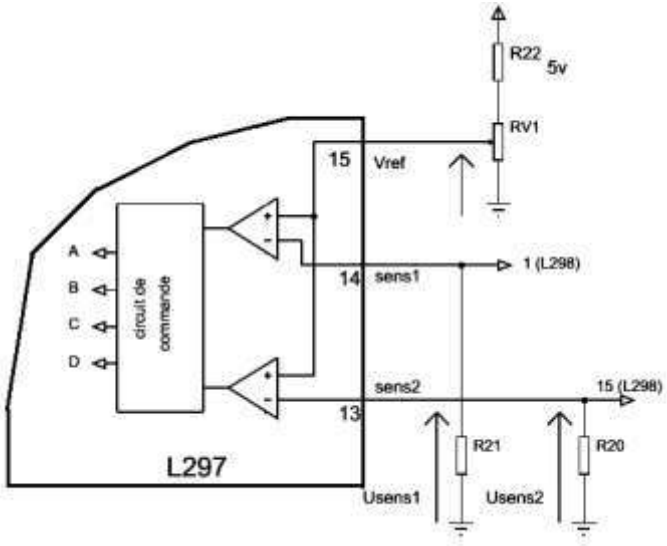


Figure III-8 : Schéma électrique démonstratif pour le calcul des résistances

5-Carte de traitement

Le bloc de traitement est basé sur une carte de développement Arduino_uno avec son microcontrôleur Atmel dont on a logé le programme qui va gérer toutes les cartes précédentes

III-3 Etude Automatique

III-3-1 Le projet en tant que système automatisé

Notre réalisation est destinée à traiter des pièces d'une manière automatisée donc on peut la classer avec ces systèmes.

Comme tout système automatisé notre réalisation est constituée de trois parties :

1 partie opérative :

Elle est composée de cinq capteurs, deux fin de courses et de deux moteurs pas à pas avec leurs driver ainsi d'un poste de chargement/déchargement de pièces, de trois bacs, d'un chariot et d'une pincette.

2 partie commande

C'est le cerveau de notre réalisation qui contient le programme, elle est à base sur la carte Arduino. Cette partie reçoit des comptes rendus venant des capteurs en vue de donner des ordres aux actionneurs (moteurs pas à pas) via les drivers.

Cette partie exécute les deux modes de fonctionnement le mode automatique et le mode manuel.

3 Pupitre de commande

Il contient quatre boutons poussoirs permettant de mouvoir à droite ou à gauche le chariot (right_Dcycl et left) ; de faire monter ou descendre la pincette (up et down) ainsi qu'un commutateur à trois positions auto , stop et manuel.

NB :

Le bouton **right_Dcycl** a double fonction, d'une part comme bouton départ cycle dans les deux modes de fonctionnement et d'autre part comme bouton de déplacement à droite du chariot dans le mode manuel.

III-3-2 Modes de fonctionnement de la maquette

1) Mode automatique

Il faut mettre le commutateur sur la position AUTO, une impulsion sur le bouton poussoir right_Dcyc provoque la mise en route du cycle, à ce moment là le chariot reprend sa position initiale qui est l'extrême droite et la pincette en position haute « **pos_pincette 0** ». Cette position est appelée « **Home_chariot** »

2) Mode manuel

On bascule le commutateur sur le mode MAN et à l'aide du pupitre commande on peut déplacer le chariot à gauche ou à droite et faire monter ou descendre la pincette.

Pour ce mode nous avons inséré quelques conditions de sécurité pour ne pas causer des défaillances à la maquette, ces conditions sont :

- On ne peut pas déplacer le chariot :
  1. A gauche lorsqu'il est sur la position « **pos-chariot 1**» et la pincette sur la position « **pos-pincette 2**» ou le chariot sur « **pos-chariot 10** ».
  2. A droite lorsqu'il est sur la position « **pos-chariot 0** » ou sur la position « **pos-chariot 9** » et la pincette sur « **pos-pincette2** »
- On ne peut pas actionner :
  1. La descente de la pincette à la position minimale « **pos-pincette 2** » lorsque le chariot est sur l'une des positions comprise entre « **pos-chariot 2** » et « **pos-chariot 8** »,voir la **Figure III- 7**
  2. La montée de la pincette lorsqu'elle se trouve sur « **pos-pincette 0**»

3) Mode stop :

Le mode de fonctionnement AUTO ou MAN est arrêté lorsque le commutateur est placé sur la position STOP.

III-3-3 Schéma fonctionnel de la maquette

Le fonctionnement de notre réalisation, selon l'exemple de programmation qui a été établi, pour le mode automatique est illustré par la **figure III- 9.**

En mode automatique, lorsque l'on appui sur le bouton **right/Dcycl** le chariot se déplace vers la position initiale « **home_chariot** » ; puis il soulève la pièce à traiter du poste de chargement pour la tremper dans le **Bac 3** dans lequel elle y restera un temps de 5secondes, par la suite la pièce est ramenée au **Bac 1** pour subir un rinçage, cette l'opération est répétée trois fois. La dernière opération de traitement est de ramener, après rinçage, la pièce au **Bac 2** et en fin cette dernière est transportée au poste de déchargement auquel se terminera le cycle automatique.

.

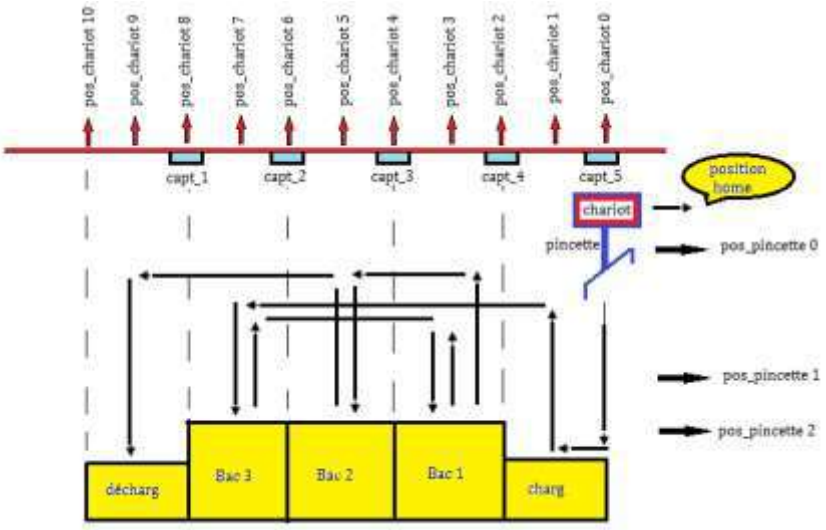


Figure III- 9: Schéma fonctionnel de notre système automatiquese

La **Figure III- 10-** montre une photo réelle du poste de traitement prise du coté arrière de la maquette

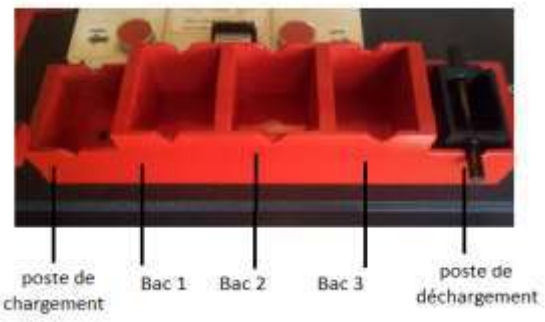


Figure III- 10 : Photo réelle du poste de traitement

III-4 Programmation de la maquette

On a programmé la maquette avec deux manières différentes :

- La carte Arduino-uno et son logiciel open source

-L'automate siemens et son logiciel **step 7**

Mais avant d'entamer la programmation on doit passer par les organigrammes qui vont nous faciliter la compréhension des cycles et en même temps nous guider pour l'élaboration des programmes.

III-4-1 Organigramme du mode automatique

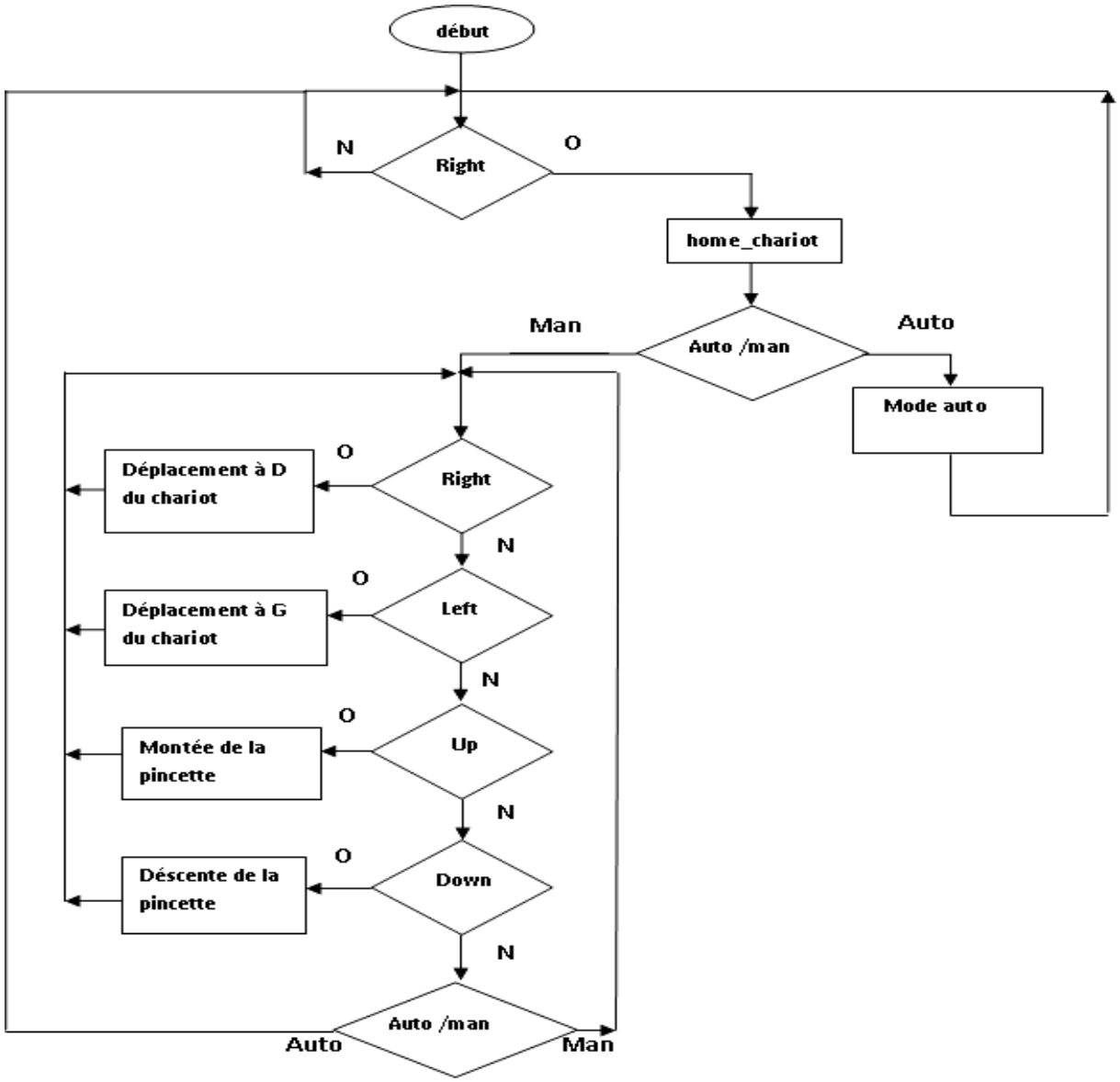


Figure III-11 Organigramme principal

III-4-2 Organigramme du mode manuel

1) Organigramme de déplacement à droite du chariot

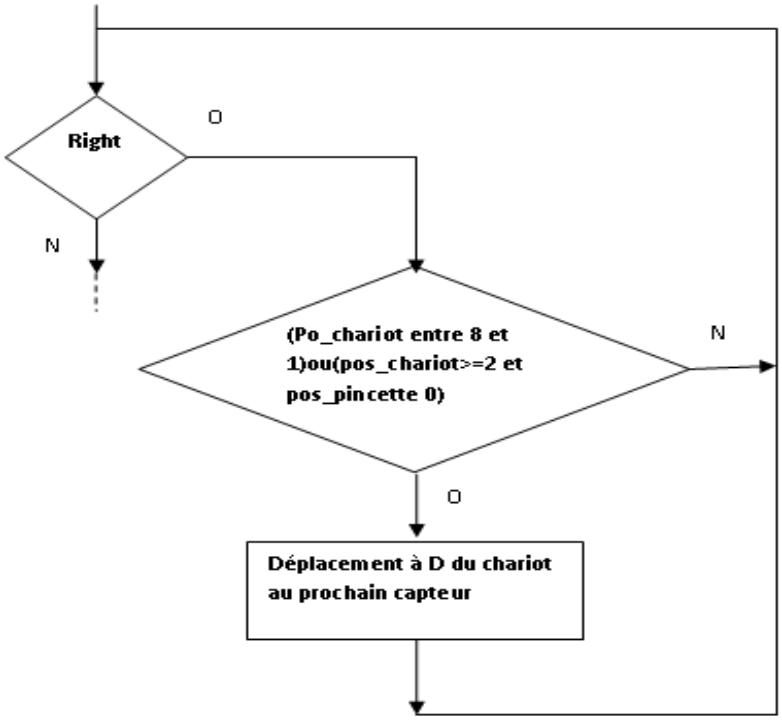


Figure III-12 Organigramme de déplacement à droite du chariot

2) Organigramme de déplacement à gauche du chariot

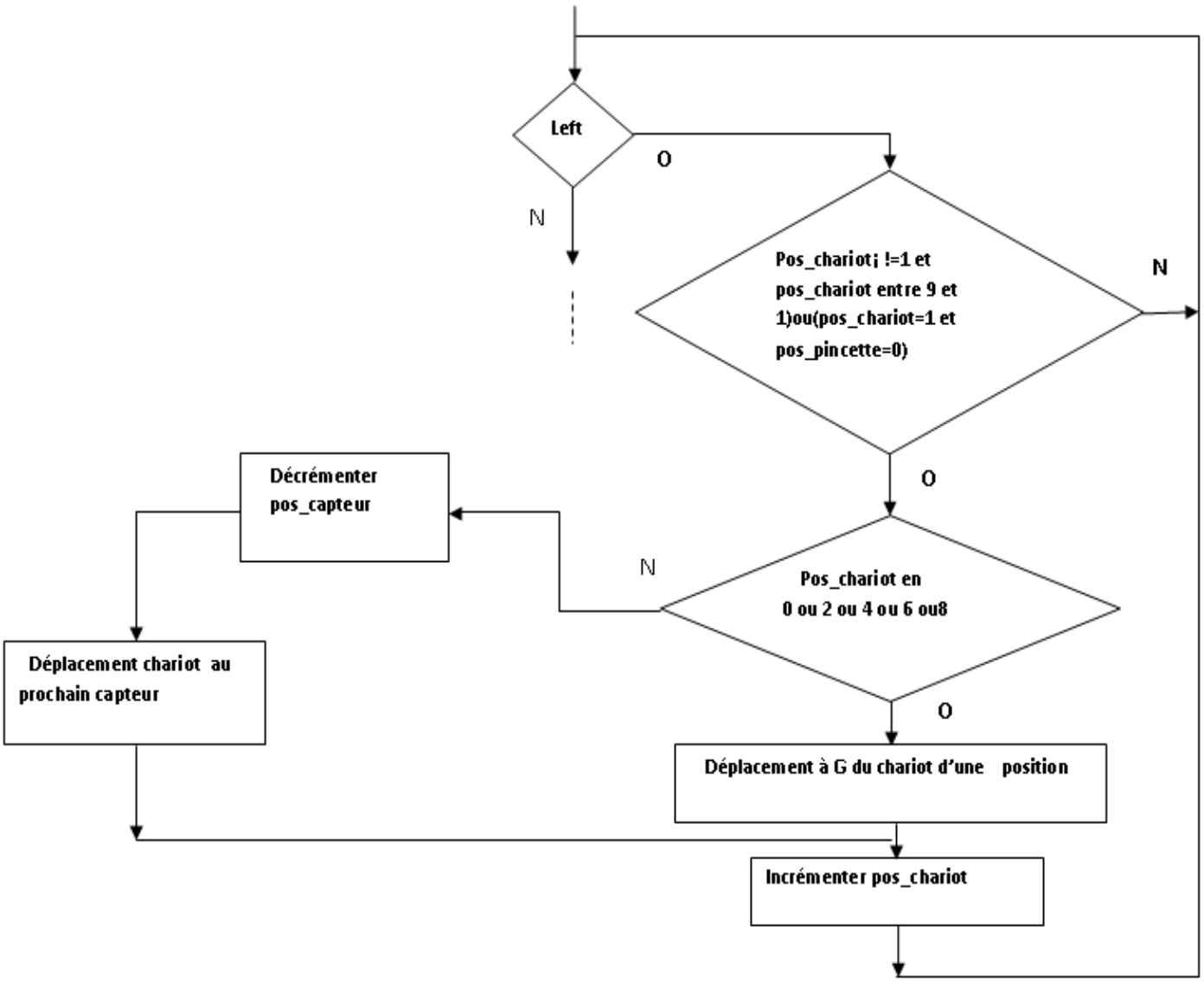


Figure III-13 Organigramme de déplacement à gauche du chariot

3) Organigramme de la montée de la pincette

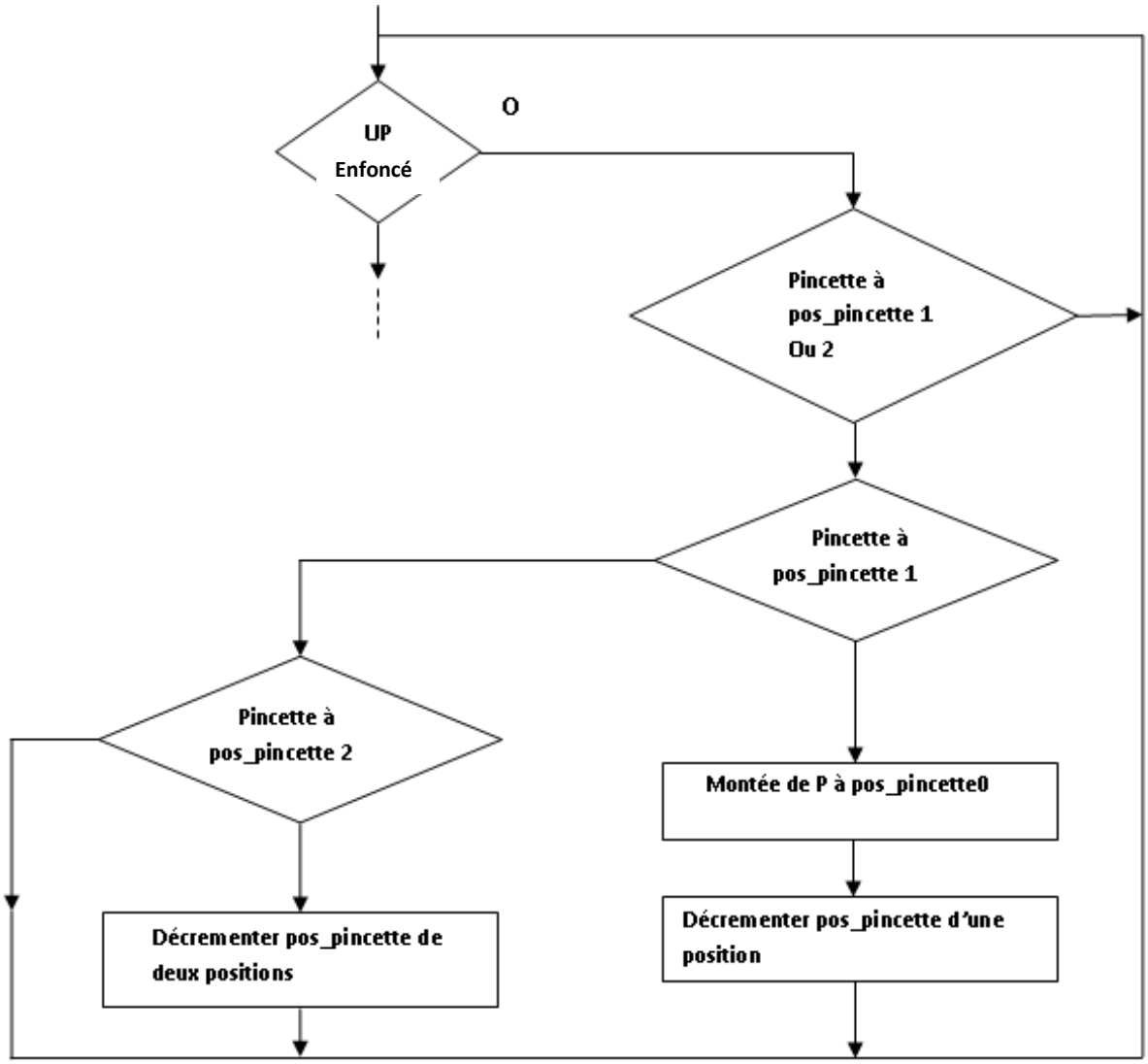


Figure III-14 : Organigramme de la montée de la pincette

4) Organigramme de la descente de la pincette

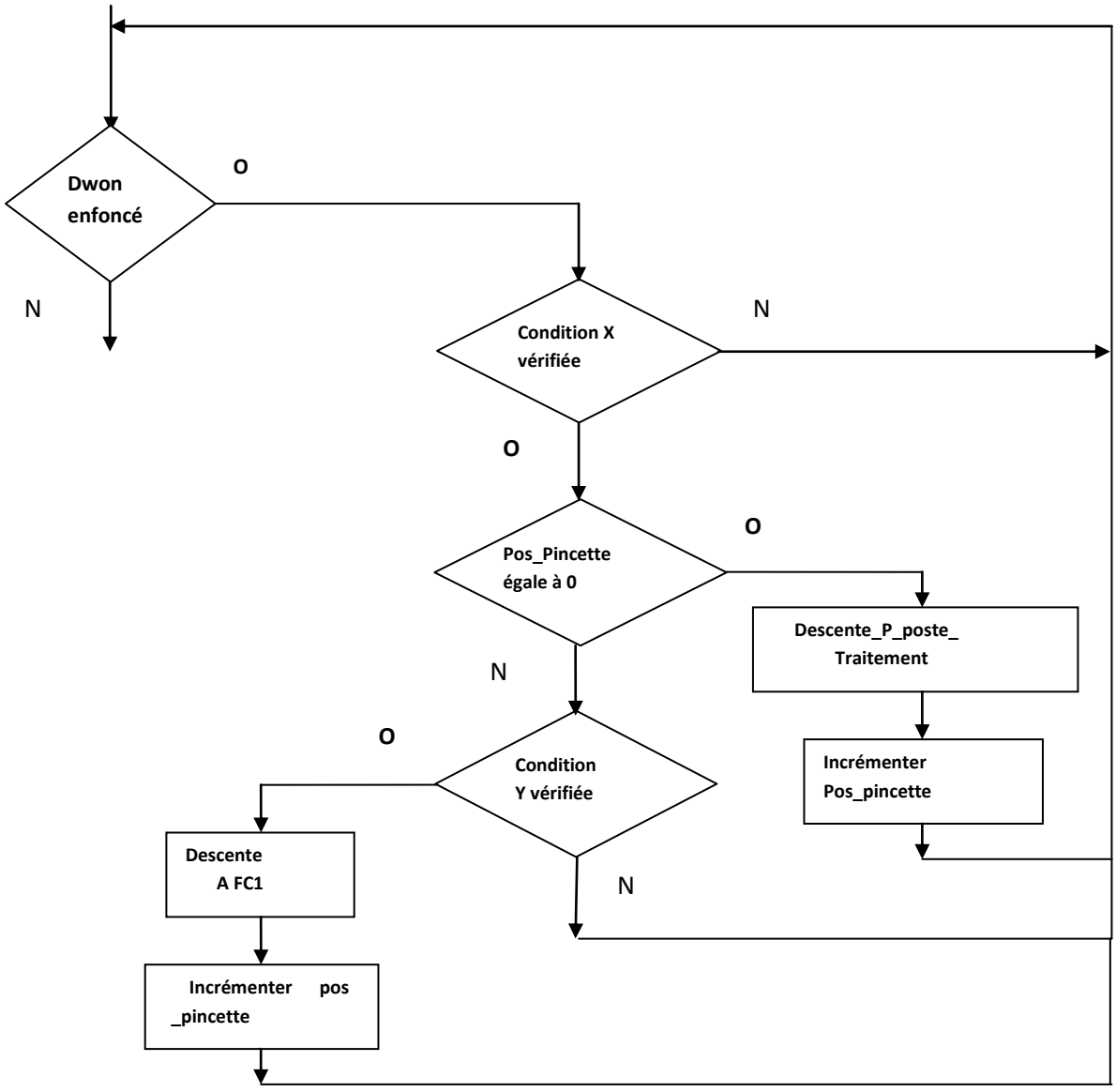
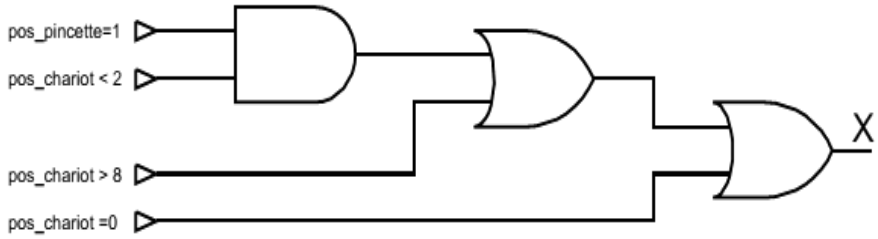
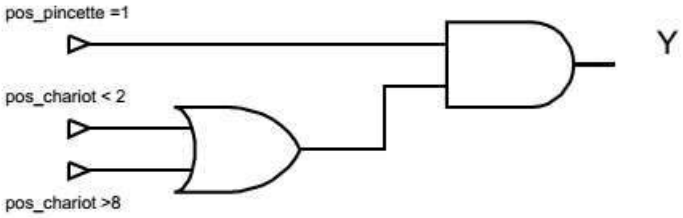


Figure III-5 Organigramme de la descente de la pincette

Condition X



Condition Y



III-4-3 Programmation sous Arduino

La programmation du mode automatique et du mode manuel utilisant le logiciel open source de Arduino est faite de manière à ce qu'on test en premier lieu si le bouton **right_Dcycl** est enfoncé, si c'est le cas on ramène d'abord le chariot à sa position initiale (**home_chariot**) puis on test de nouveau si le commutateur est placé sur **mode-auto**, ou sur **mode-Man**.

Une fois ceci est réalisé, on est sensé alors d'exécuter soit le programme du mode manuel soit celui du mode automatique. Sauf que pour le premier mode des conditions primordiales doivent s'imposées comme nous en avons évoqué dans l'étude automatique au paragraphe **III-3-2.** Voir le programme détaillé en **Annexe** à partir de la **page –I-.**

III-4-4 Programmation avec l'automate siemens

Les automaticiens utilisent les automates programmables industriels(API) afin d'automatiser des processus, parmi ces API on trouve la firme siemens avec le logiciel **step7**. Ce dernier possède quatre types de programmation le plus répandu est le Grafcet.

Le Grafcet des deux modes de notre réalisation est donnés en Annexe à partir de la **page-XII.**

III-5 Conclusion

L'étude technique est une partie primordiale de toute réalisation, elle doit suivre un cheminement bien déterminé à commencer par le schéma synoptique ensuite le schéma électrique. A partir de ces deux schémas nous avons extrait les composants nécessaires avec un choix judicieux des valeurs. Nous avons passé par la suite à l'élaboration des algorithmes correspondants au principe de fonctionnement.

Et finalement on a utilisé ces algorithmes pour programmer la réalisation avec deux manières différentes qui sont : La carte Arduino et l'automate siemens dans sa version **Step7.**

Chapitre IV

Etude pratique des cartes électriques et constats obtenus

IV_1 Introduction

Dans cette partie nous allons montrer toutes les étapes de réalisation de la maquette, en l'occurrence, nous commençons par la réalisation des différents circuits imprimés nécessaires ensuite nous aborderons l'aspect pratique de notre maquette.

IV_2 Fabrication des circuits imprimés

Nous allons commencer notre étude hardware par la fabrication des circuits imprimés des différentes cartes électroniques, à savoir le circuit de commande des moteurs pas à pas(Drivers) ; circuit électrique de la carte de commande manuelle (pupitre de commande) ; circuit électrique des capteurs et la carte de connexion reliant tous les circuits cités à l'Arduino_Uno.

On a utilisé, pour la fabrication des circuits imprimés**,** le logiciel **TCI** qui est très facile à manipuler manuellement. Trois étapes sont essentielles pour fabriquer un circuit imprimé en utilisant un typon (négatif) et une plaquette cuivrée pré-sensibilisée :

a-Insolation

b-Révélation

c-Gravure

*** Circuit imprimé du Driver et sa carte finale :**

La **Figure IV-1** montre qu'il s'agit d'un circuit à double faces dont la couleur rouge représente la face cuivrée et la couleur verte représente la face composants.

C'est ce qui a été illustré respectivement dans les **Figure IV-2** et **Figure IV-3.**

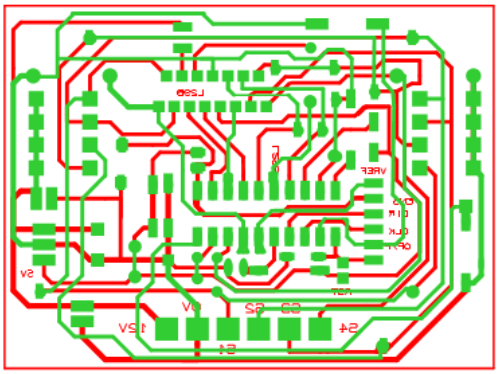


Figure IV-1: typon du Driver à double faces

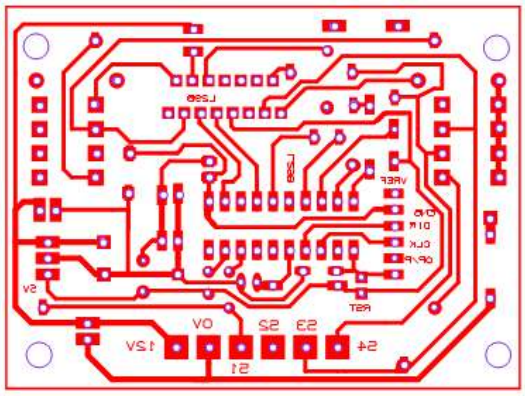


Figure IV-2: typon face cuivrée

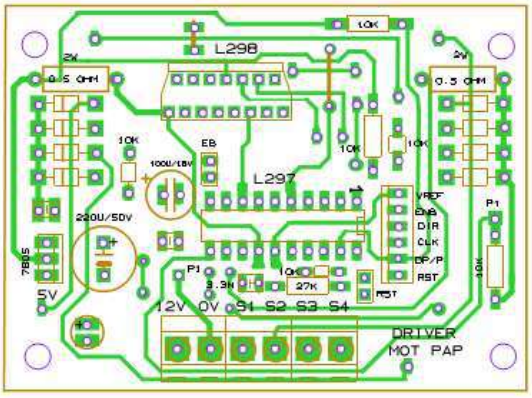
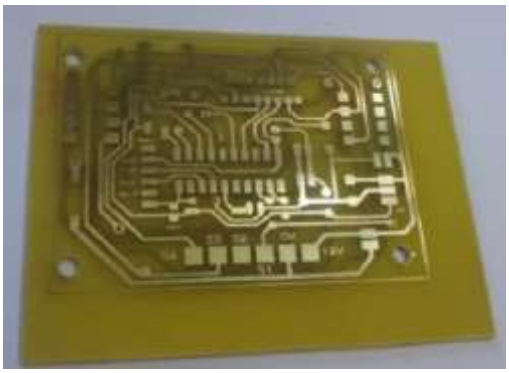
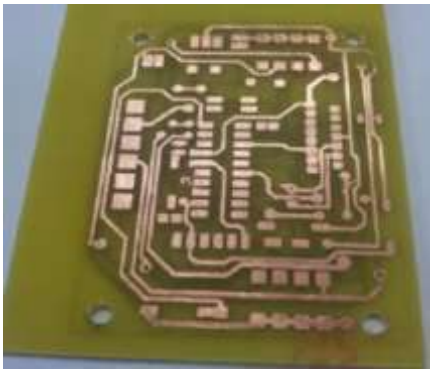


Figure IV-3: face composants

La **Figure IV-4** montre les deux cotés du circuit imprimé du Driver dans son état final mais sans percage



(a) coté cuivrée



(b) coté composants

Figure IV-4: circuit imprimé du Driver

La **Figure IV-5** représente la carte électronique du Driver avec les composants éléctroniques correspondants notament le couple circuits **L297** et **L298**.

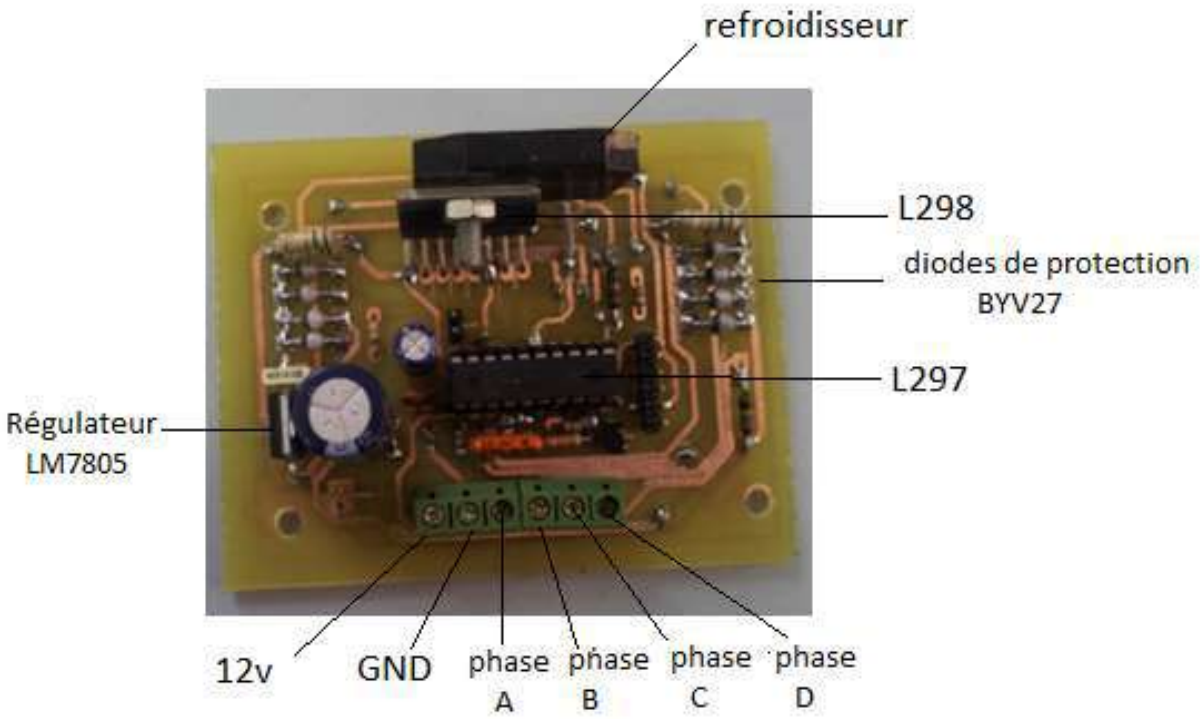


Figure IV-5: carte Driver finalisée

***Circuit imprimé du bloc capteurs et sa carte finale**

La **Figure IV-6** montre que la carte est réalisée sur circuit imprimé à simple face. Nous voyons sur le schéma d'implantation de la **Figure IV-7** l'emplacement des cinq capteurs (**CNY70**) par ordre croissant du gauche à droite.



Figure IV-6: Typon de la carte capteurs

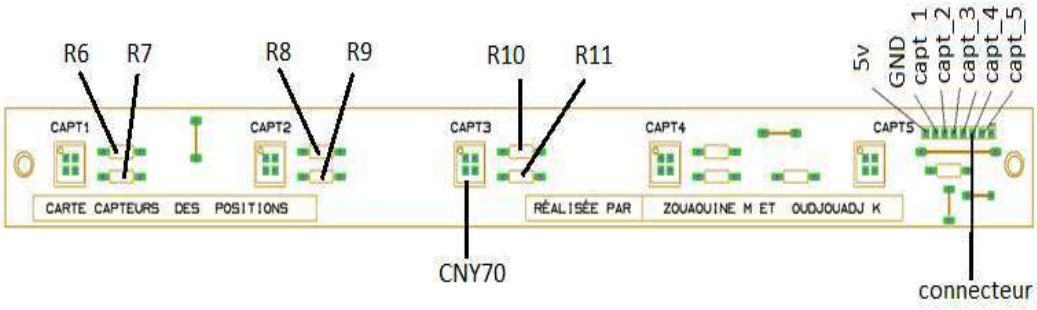


Figure IV-7: Schémad'implantation de la carte capteurs



Figure IV-8: la carte capteurs finale fixée au dessus du chariot

- **Circuit imprimé du pupitre de commande et sa carte finale**

C'est un circuit à simple face ,son typon est donné à la **Figure IV-9** en revanche le schéma d'implantation approprié est sur la **Figure IV-10.**

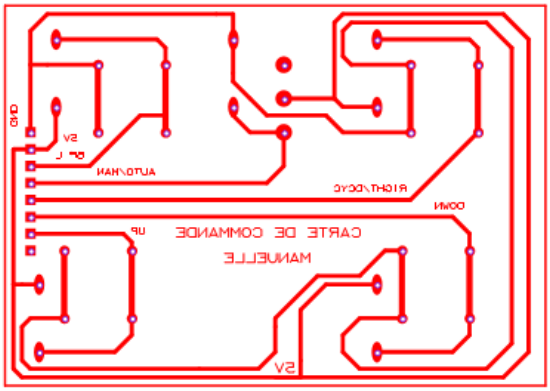


Figure IV-9: Le typon de la carte de commande manuelle

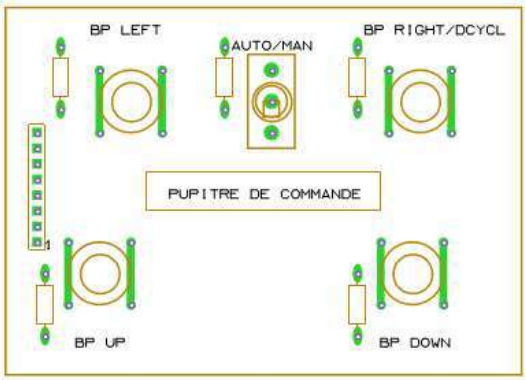


Figure IV-10: Schéma d'implantation

- **Circuit imprimé de la carte de connexion avec Arduino_Uno**

Le typon de cette carte est illustré par la **Figure IV-11** ,le schéma d'implantation est donné à la **Figure IV-12.**



Figure IV-11: Typon du circuit de connexion avec Arduino_Uno

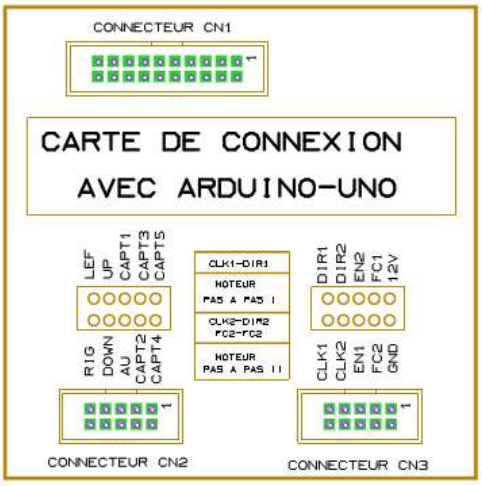


Figure IV-12: Schéma d'implantation correspondant

La **Figure IV-13** représente la carte de connexion finale avec ses trois connecteurs CN1 ; CN2 et CN3 ainsi que le brochage des ces deux derniers avec Arduino_Uno, pour le brochage de CN1et CN2 voir Annexe **page-xx-**.

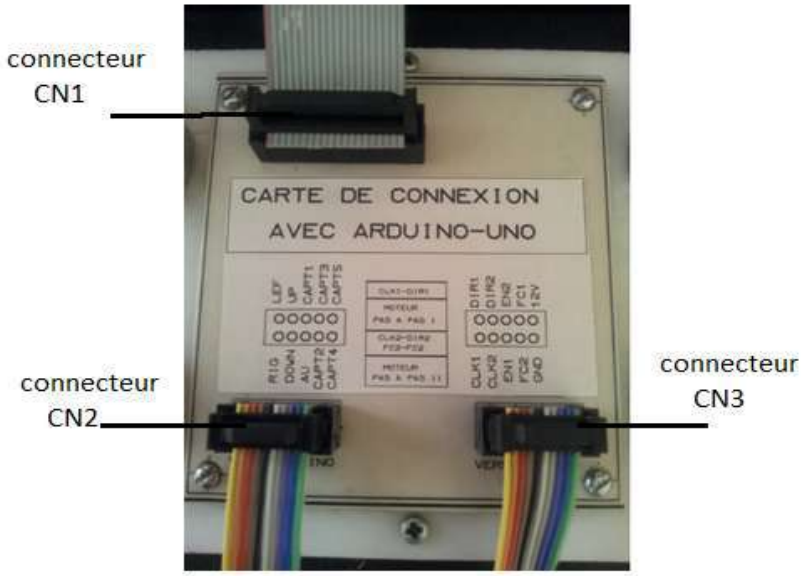


Figure IV-13: La carte de connexion dans son état final

IV_3 Assemblage des cartes et connexion avec Arduino

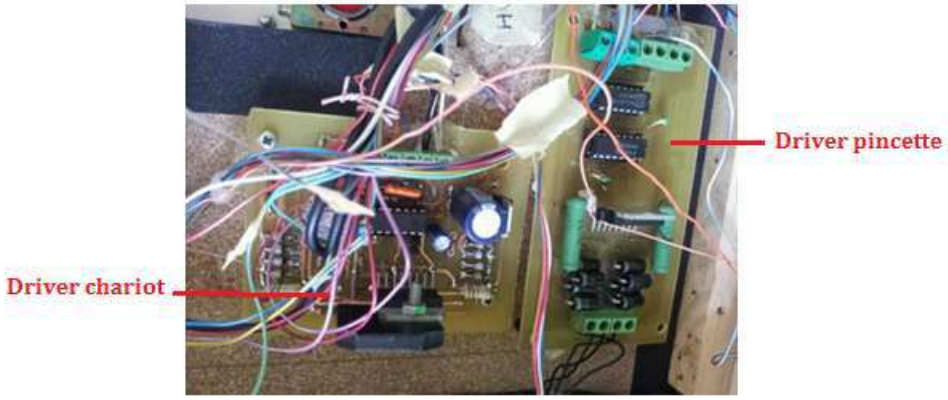


Figure IV-14 Assemblage des deux Drivers

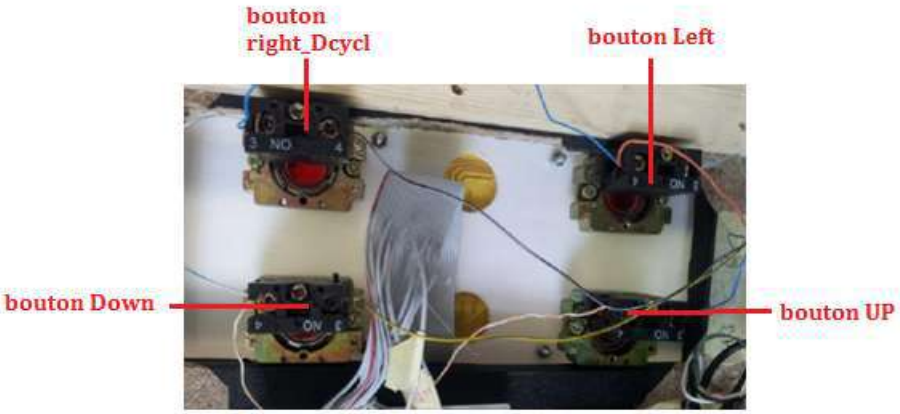


Figure IV-15 connexion des boutons de commande

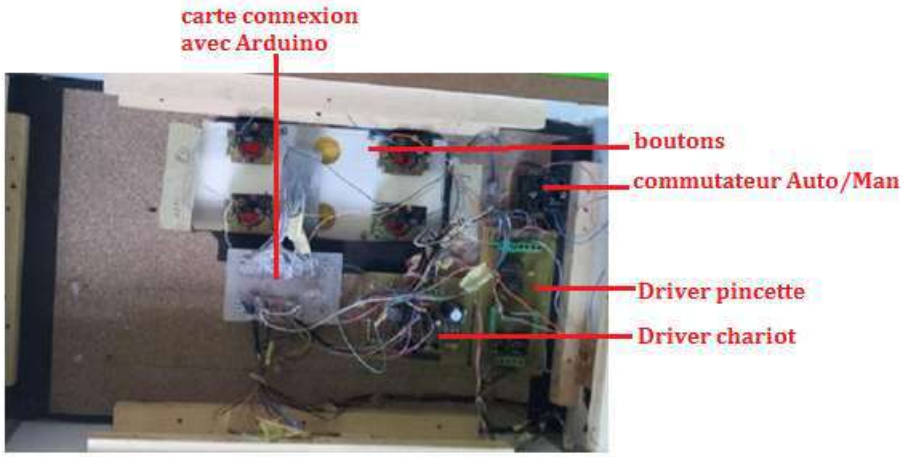


Figure IV-16 Assemblage de toutes les cartes

IV_4 Insertion des deux moteurs dans le chariot

Les deux moteurs pas à pas, qui sont sensés de faire mouvoir le chariot et faire monter ou descendre la pincette, sont insérés à l'intérieure du chariot comme les montrent la **Figure IV-17** et la **Figure IV-18**

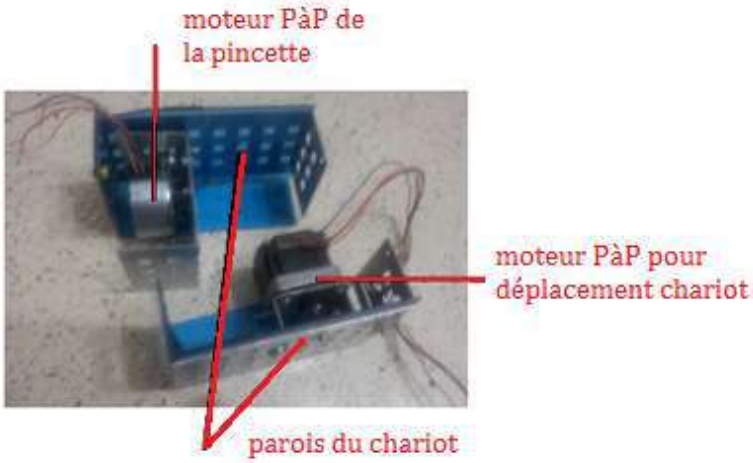


Figure IV-17 insertion des moteurs dans le chariot

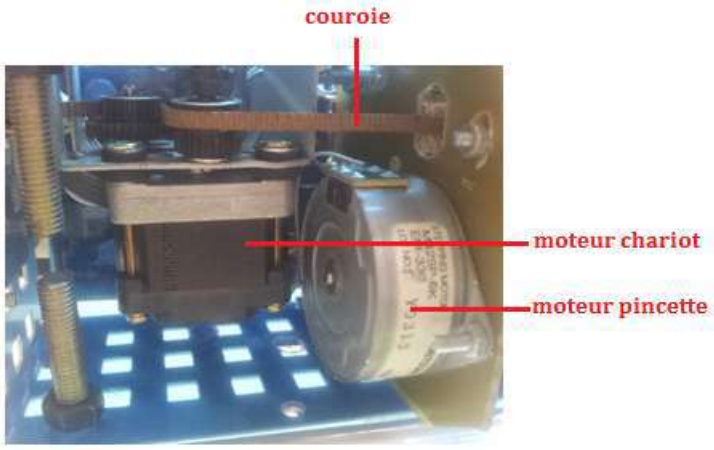


Figure IV-18 vue de dessous du moteur chariot entrainé par la courroie

Figure IV-18 montre l'assemblage du moteur chariot et la courroie, à cet effet lorsqu'il entre en rotation un déplacement linéaire est assuré grâce à la courroie. Le sens de déplacement dépend du sens de rotation.

Figure IV-19 donne une image finale du chariot disposé sur les rails sur les quels il se déplace à droite ou à gauche.

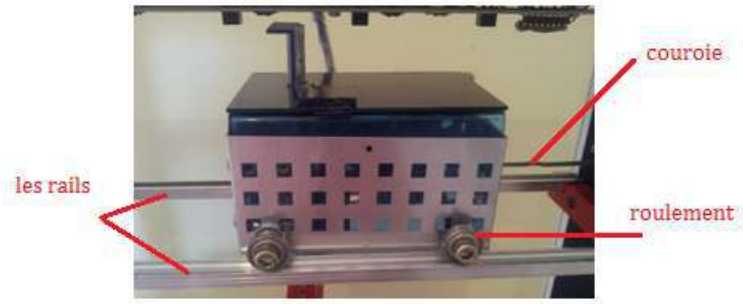


Figure IV- 19 Chariot dans son état final

Figure IV 20 montre l'emboitement des peignant de la pincette et du moteur sans oublier les fins de course FC1 et FC2 néanmoins que le second n'apparait pas sur la photo**.**

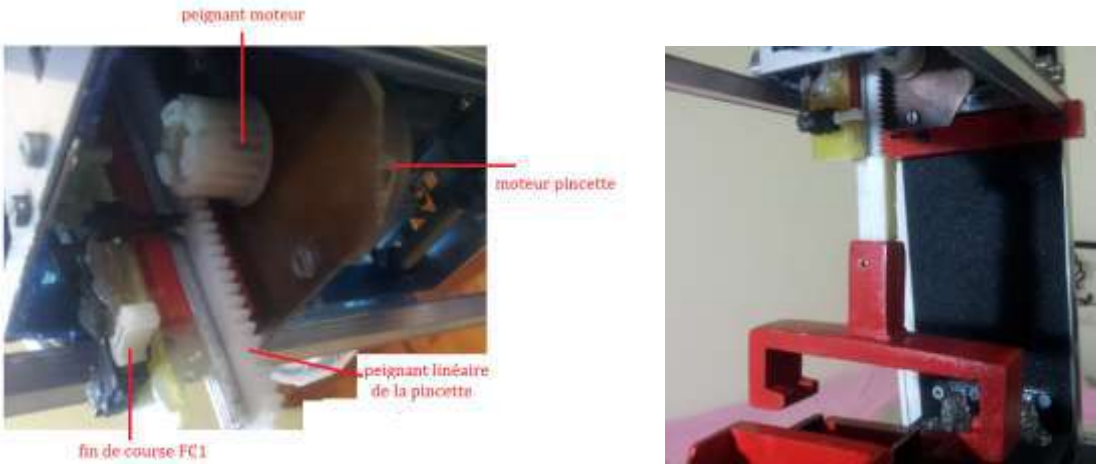


Figure IV-20 emplacement de la pincette

IV_5 Insertion de la carte capteurs au dessus du chariot

La **Figure IV-21** montre l'emplacement de la carte qui doit se trouver au dessus du chariot pour qu'elle puisse repérer les **11** positions du chariot utilisées lors de la programmation de notre réalisation.

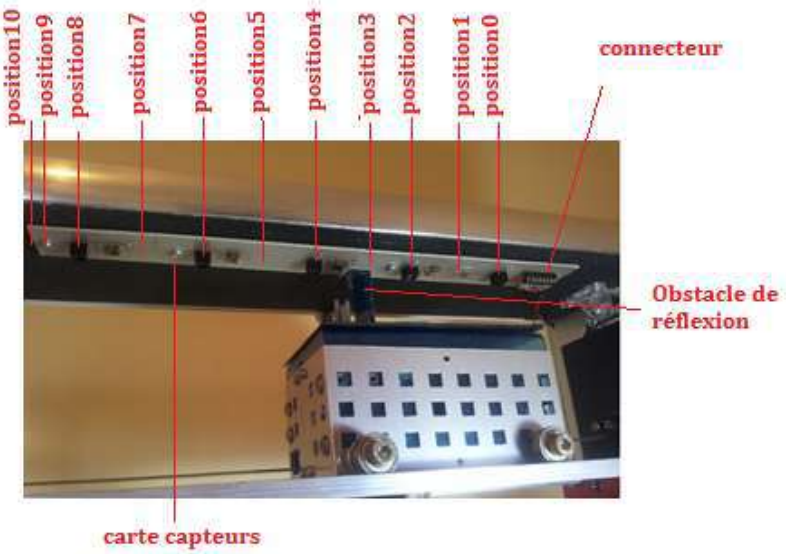


Figure IV-21 Carte capteurs et les 11 positions du chariot

IV_6 Commutateur de modes

Le commutateur de mode Auto/Man/Stop et connecté à la carte Arduino_Uno par la pin analogique A4, par conséquent cette dernière est configurée pour fonctionner comme entrée analogique du faite qu'elle peut recevoir trois commande (Auto ; Man ou Stop). Pour cela nous avons octroyé les valeurs analogique 4V au mode Auto ;2V au mode Man et 0v au mode Stop.

Figure IV-22 représente la position du commutateur sur la maquette ainsi les trois positions concernées.



Figure IV-22 Commutateur Auto/Man/Stop

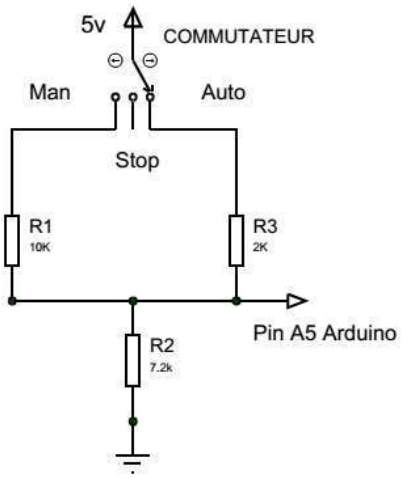


Figure IV- Schema électrique du commutateur Auto/Man/Stop

IV_7 pupitre de commande

Dans la **Figure IV 23** sont illustrés tous les boutons de commande du chariot et de la pincette. A droite se trouve le bouton **right_Dcycl** qui est utilisé pour déplacer le chariot ainsi comme bouton départ cycle

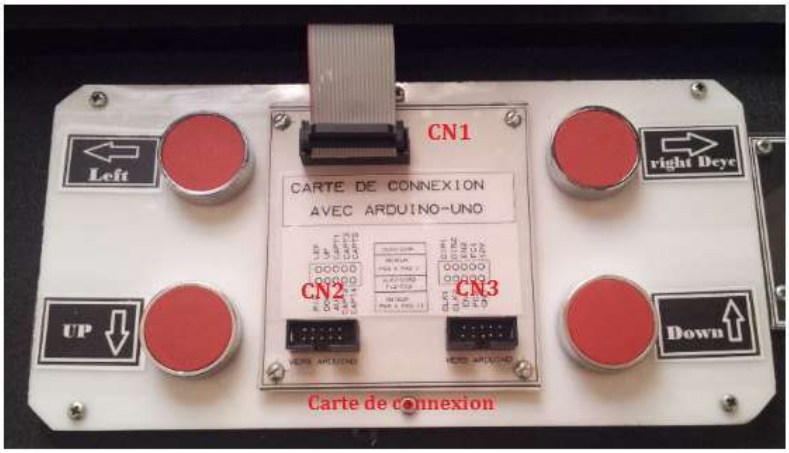


Figure IV-23 Pupitre de commande avec la carte de connexion

Le brochage des connecteurs CN2 et CN3 avec l'Arduino_Uno est donné par le tableau suivant.

Connecteur **CN2**	Pin1 **right**	Pin2 **left**	Pin3 **down**	Pin4 **up**	Pin5 **mode**	Pin6 **Capt1**	Pin7 **Capt2**	Pin8 **Capt3**	Pin9 **Capt4**	Pin10 **Capt5**
Arduino	**A0**	**A1**	**A2**	**A3**	**A4**	**8**	**9**	**10**	**11**	**12**

Connecteur **CN3**	Pin1 **Ck1**	Pin2 **Dir1**	Pin3 **Ck2**	Pin4 **Dir2**	Pin5 **EN1**	Pin6 **EN2**	Pin7 **FC2**	Pin8 **FC1**	Pin9 **GND**	Pin10 **12V**
Arduino	**7**	**6**	**5**	**4**	**3**	**2**	**A5**	**0**	**GND**	**Non connectée**

Tableau IV-1 connexion CN2 et CN3 avec Arduino_Uno



Figure IV-24 connexion pupitre_Arduino Uno

Le connecteur **CN1** est conçu pour faire l'interconnexion entre la carte de connexion avec Arduino_Uno et les différentes électroniques qui sont à l'intérieure du socle à savoir les deux Drivers des moteurs ; connecteur de la carte capteurs et connecteurs de la carte commande manuelle.

La figure suivante montre une photo globale de notre réalisation dans son état final et sur la quelle touts éléments sont apparait.



Figure IV-25 Image complète de la maquette

IV_8 Constats obtenus

IV-8-1 Constats obtenus de la carte Driver et la carte capteurs

Les résultats obtenus lors de la mise en marche de notre maquette vis-à-vis les signaux délivrés que se soit par la carte Driver des moteurs soit par la carte capteurs, sont les mêmes en utilisant la simulation sous le logiciel proteus dans son éditeur ISIS. En effet, les chronogrammes obtenus lors de la simulation sont donnés par ce qui s'en suit :

1) Carte Driver

Lors de la simulation sous ISIS du circuit de commande du moteurs pas à pas qui est cadencée par un signal d'horloge de fréquence **f** ; la commande d'activation du circuit **L297** (**Enable** à 5v) et un choix au préalable du sens de rotation du moteur (**cw/ccw** à 5v).

La **Figure-IV26 -** montre les séquences de pilotage du moteur avec un signal d'horloge de fréquence **f**=100Hz. En revanche, la **Figure-IV-27** montre que l'on peut commande le moteur pas à pas en vitesse tout en variant la fréquence du signal d'horloge du circuit **L297**.

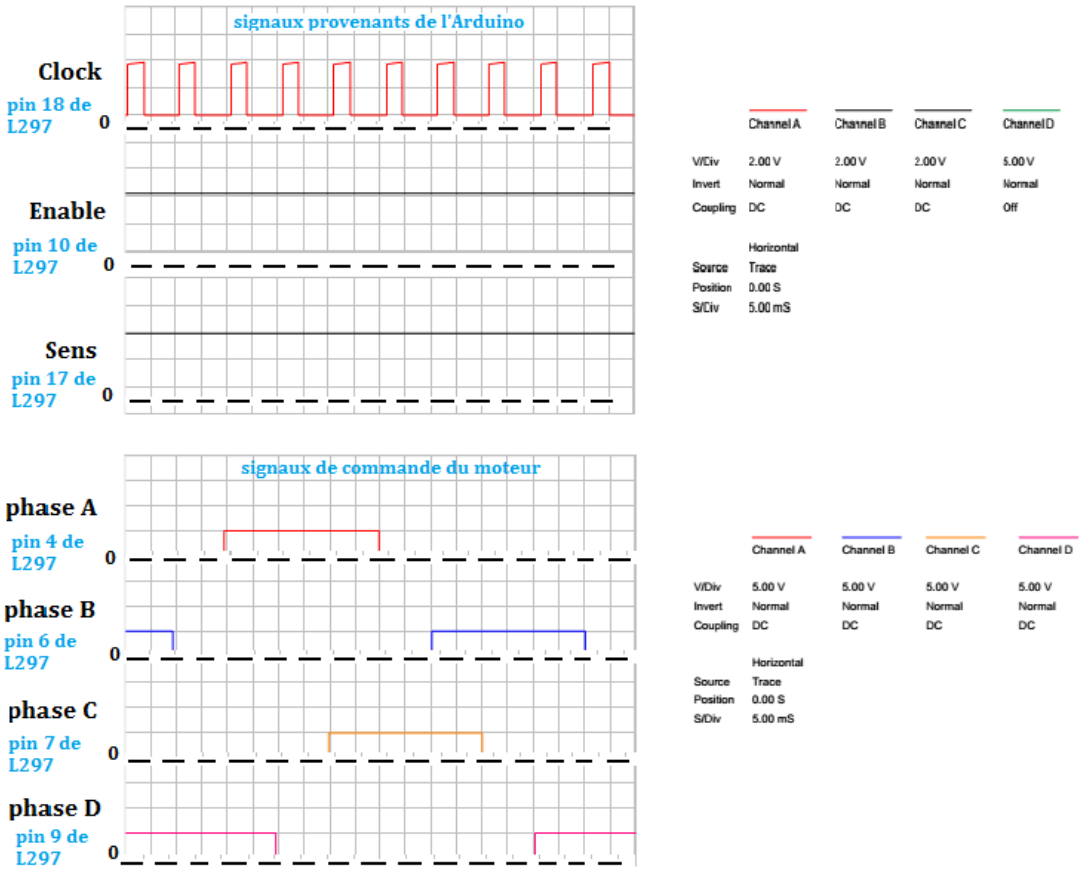


Figure-IV-26 chronogrammes obtenus de la carte driver sous ISIS avec une fréquence de 100hz

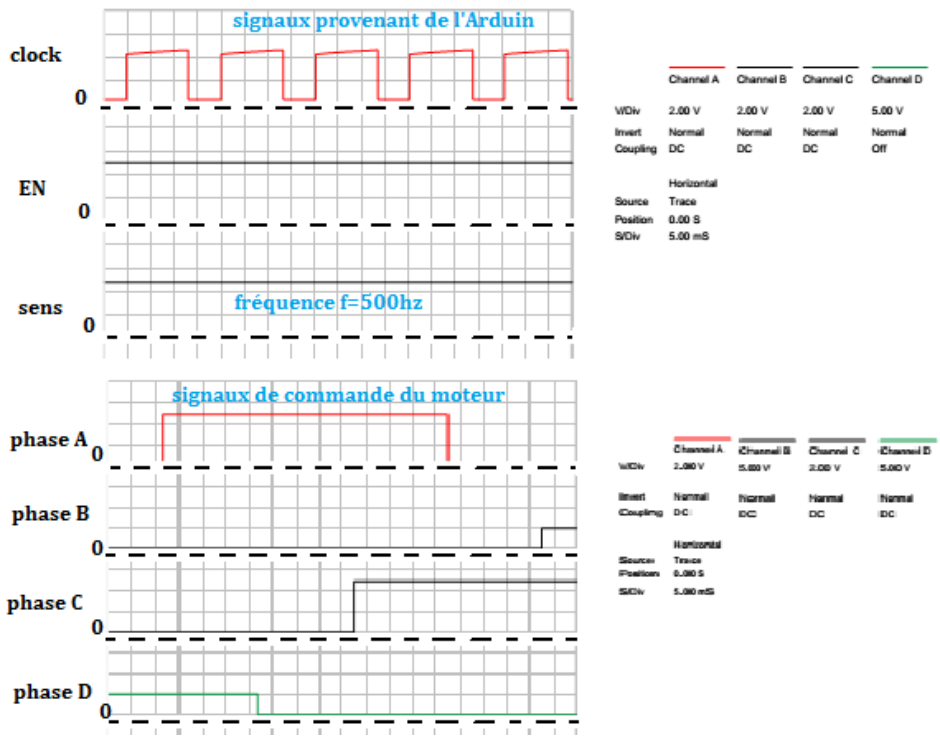
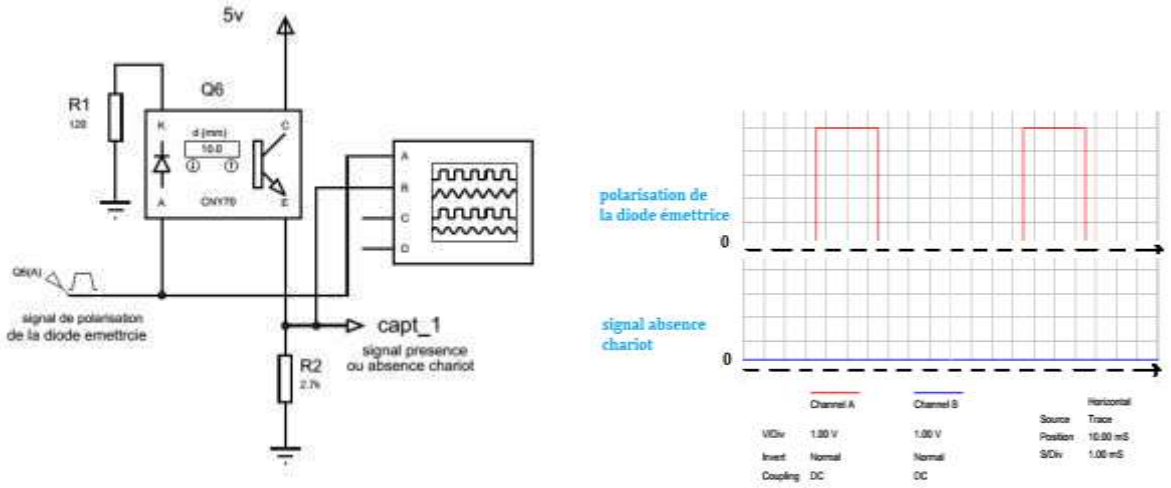


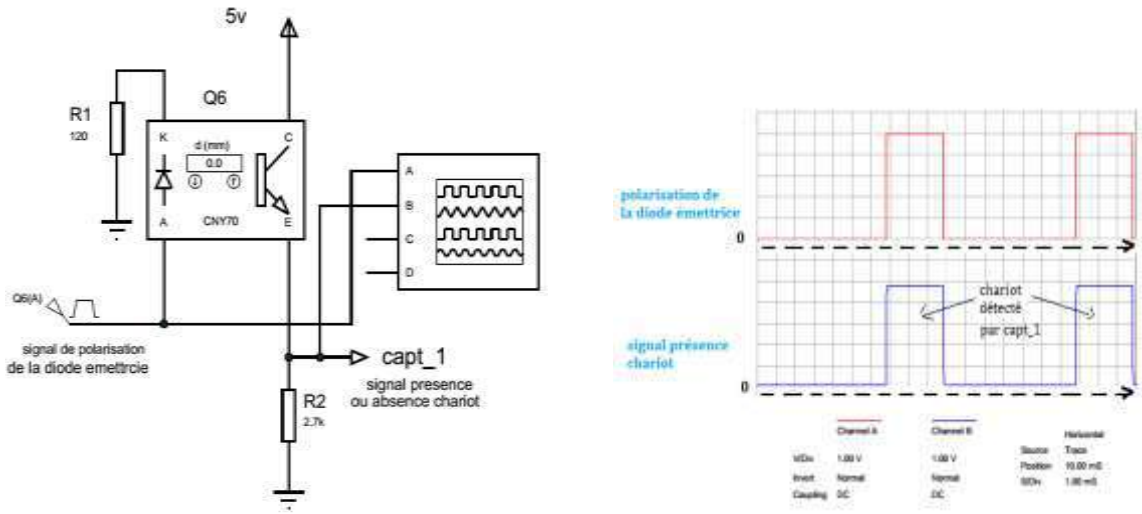
Figure-IV -27 chronogrammes obtenus de la carte driver sous ISIS avec une fréquence de 500hz

2) Carte capteurs

Les **Figure-IV28** et **Figure- IV 29** explicitent un seul étage de la carte capteurs, c'est le **capt_1** qui a pour rôle de détecter la position 8 « **pos_chariot 8 »** du chariot. Les chronogrammes de la première figure correspondent à l'absence du chariot sous le capteur **capt_1** par contre dans la deuxième figure la présence de ce dernier est illustrée par le signal donné par les chronogrammes correspondants.



FigureIV-28 schéma de simulation de la carte capteurs en absence de chariot



FigureIV- 29 Schéma de simulation de la carte capteurs en présence de chariot

IV_8_2 constats obtenus lors de fonctionnement automatique

1.Positions occupées par le chariot lors de fonctionnement



Figure IV-30 positions réelles (8 ; 7 et 4) occupées par le chariot

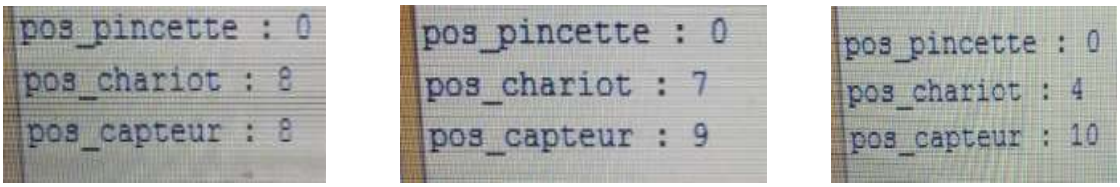


Figure IV-31 Indications en temps réel via l'interface série du logiciel

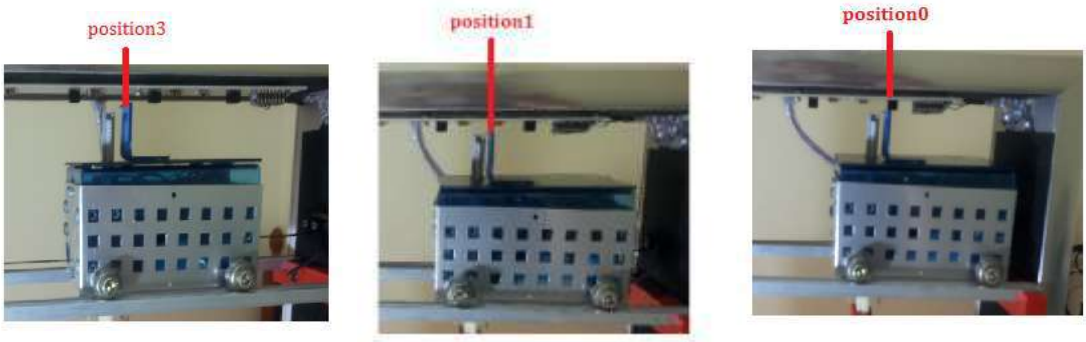


Figure IV-32 positions réelles (3 ; 1 et 0) occupées par le chariot

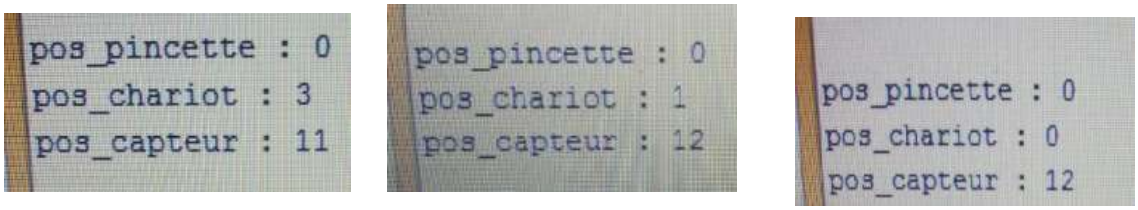


Figure IV-33 Indications en temps réel via l'interface série du logiciel

Les photos précédentes sont prises lors du fonctionnement de la maquette et les indications mentionnées aux dessous de chacune des photos sont photographiées depuis l'interface série du logiciel Arduino dont on donne une image complète à la figure suivante.

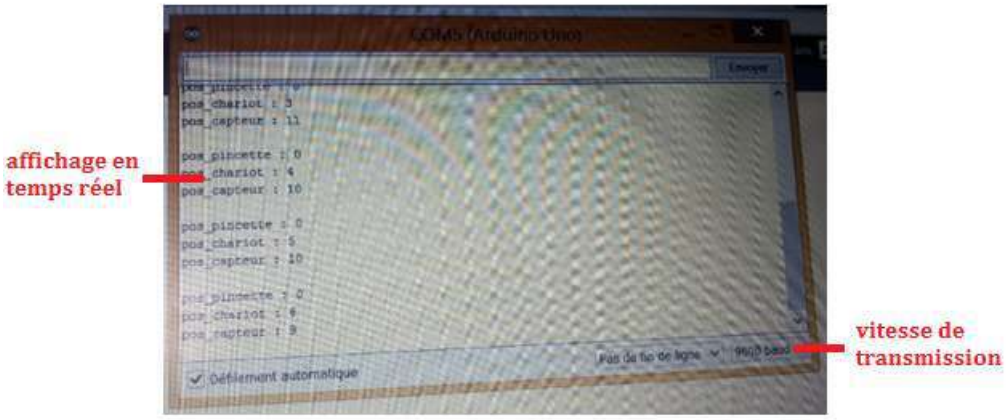


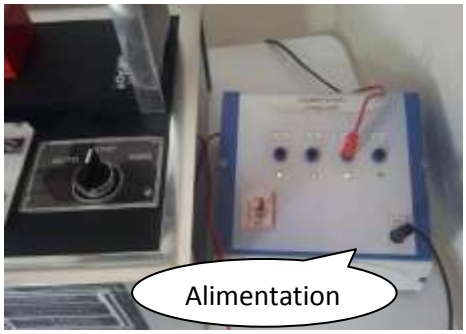
Figure IV-34 image de l' interface série du logiciel Arduino

2.Quelques photos lors de fonctionnement automatique

Les photos suivantes nous montrent toutes les séquences de traitement d'une pièce lors de fonctionnement de notre banc en mode automatique, chaque photo est commentée.



Photo globale de la réalisation.



Alimentation de notre maquette



Position home chariot



Position home chariot d'une autre vue



Chargement de la pièce par la pincette



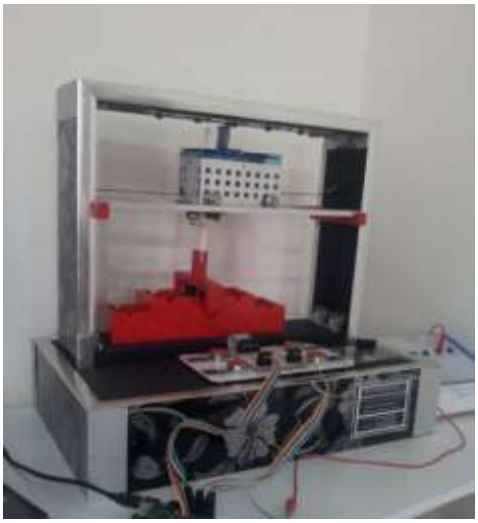
Pièce chargée en position 1 du chariot



Pièce chargée est ramenée au bac3



Pièce traitée au bac3 est ramenée au bac1



La pièce est ramenée ensuite au bac 2



L'étape finale est de ramener la pièce au Poste de déchargement



L a pièce est en position de déchargement



Pièce déchargée et chariot en position Finale, la on doit appuyer sur BP right_Dcycl Pour un nouveau cycle

IV-9 Conclusion

Dans ce chapitre pratique nous avons présenté toutes les parties nécessaires dont dépend notre réalisation en passant par la structure mécanique, chose qui nous permet d'apprendre davantage concernant le logiciel **solidworks**, sauf que nous aurons aimé de mieux le métriser mais

La consistance du projet nous a empêchée. Nous avons entamé par la suite la partie la plus consistante, c'est la partie électronique dans la quelle toutes les cartes que se soient de commande ou d'acquisition de données sont détaillées théoriquement et pratiquement, sachant que les photos que nous avons inclus dans ce chapitre illustrent presque toutes les étapes qu'on a entrepris au cours de la réalisation, il en est de même pour la phase de fonctionnement de notre maquette.

Ce chapitre, nous a permet en vertu de notre expérience dans le domaine électronique de mieux approfondir et bien exploiter nos connaissances acquises lors de notre étude faite à l'**Université de Médéa,** On note également que les constats mentionnés et représentés par les chronogrammes quant aux circuits Driver et capteurs sont faites et simulés par le logiciel **Proteus** dans son éditeur **ISIS** pourront êtres établis directement des cartes ainsi réalisés en utilisant l'oscilloscope.

Conclusion Générale

Au cours de ce travail nous avons pu exploiter nos connaissances théoriques et pratiques que nous avons appris le long de nos études au sein de l'université de **M**édéa **Y**ahia **F**arès et plus précisément dans le département génie mécanique en spécialité Mécatronique.

Notons bien qu'une vision globale de la maquette a été envisagée avant de passer à la réalisation pratique en l'exploitation d'un logiciel très puissant notamment dans le domaine mécanique, c'est le logiciel **SolidWorks.** Il nous a beaucoup aidé à la modélisation et à la conception mécanique de notre prototype.

Par ailleurs, concernant le bon fonctionnement et la programmation optimale du dispositif le passage par le logiciel **Proteus** dans son éditeur **ISIS** est obligatoire notamment quant-il s'agit d'une réalisation ou d'une conception électronique. Il nous a permet même de simuler les résultats envisagés et rectifier certaines taches de programmation dans la mesure où la carte Arduino_Uno est déjà intégrée dans sa bibliothèque,

Nous espérons que ce mémoire apporte néanmoins une contribution sur des points particuliers et ouvre d'autres perspectives ; et qui peut servira comme aide pédagogique dans des travaux pratiques spécialement dans le domaine automatique.

La phase de programmation sous Aruidno a été bien établie et bien détaillée, en effet nous avons beaucoup appris à programmer la carte Arduino_Uno.

En outre, beaucoup de perspectif que se soit **software** ou **hardware** peuvent être entreprises par des future étudiants dont nous citons à titre d'exemple :

Software :

- Réalisation d'une interface graphique sous Matlab qui pourra gérer les deux modes de fonctionnement automatique ou manuel.
- Etablissement des supports pédagogiques pouvant servir comme brochure pour des TPs d'automatique
- Programmation du banc par L'API siemens dans sa version SIMATIC STEP 7 en exploitant les différents langages de programmation à savoir le langage Grafcet et le langage Ladder.

Hardware :

- On pourra également ajouter de modules comme Bluetooth et émetteur récepteur à infra-rouge en vue de faire fonctionner le banc à distance.

- On pourra faire fonctionner le dispositif à l'aide de téléphone portable Smartphone tout en exploitant de modules appropriés tel que **Shield1Sheeld**
- Des composants d'affichage peuvent être ajoutés comme par exemple afficheur LCD ; affichage à LEDs pour indiquer les positions du chariot ainsi l'ajout d'un bouton d'arrêt d'urgence est primordial.
- En effet, On pourra même ajouter un panneau d'affichage dynamique représentant l'évolution de fonctionnement du banc.

Bibliographies

Les livres techniques

[1] Les systèmes automatisés (de la connaissance à la conception)

C.BOURBONNE .J.COJEAN _TOME 1 et **TOME 2**
EDITION FOUCHER
128, Rue de Rivoli – PARIS (1er)

[2] Les Automatismes LYCEES TECHNUQUES
FORMATION CONTINUE COLLECTION A. Capliez
Educalivre 32 rue de Grenelle, 75007 Paris

Mémoire de fin d'étude

[1] Etude, conception et réalisation d'un robot parallèle type delta
Proposé par Dr : Bouhhada Mounir
Elaboré par : Euldji Rfik et Selami Yassine

Université Dr Yahia FARES de Médéa Année : 2014-20

[2] Etude, conception et réalisation d'un support d'un panneau solaire commandé par camera et entrainé par moteur pas à pas
Proposé par Dr : Boumahdi Mouloud
Tchoketch kebir Sofian
Elaboré par : Chebouki Abderrahman
Benramdane abdallah haroun
Université Dr Yahia FARES de Médéa Année 2013-2014

Webographie

[1] http://circuitdiagram.net/arduino-uno.html
Date de consultation 02-03-2016

[2] http://perso.iut-nimes.fr/fgiamarchi/wp-content/uploads/2011/02/TP-1-CNY70.pdf
Date de consultation 06-03-2016

[3] http://electronics.stackexchange.com/questions/95404/l297-l298-circuit-not-working
Date de consultation 17-03-2016

[4] http://sdz.tdct.org/sdz/apprenez-a-utiliser-solidworks.html#ApprenezutiliserSolidWorks
Date de consultation 07-04-2016

[5] http://moteurindustrie.com/moteurs-pas-a-pas/technique/50-le-moteur-pas-a-pas-a-aimant-permanent.html
Date de consultation 09-04-2016

[6] http://electronics.stackexchange.com/questions/95404/l297-l298-circuit-not-working
Date de consultation 12-04-2016

[7] http://www.transmission-expert.fr/pdf_fabricant/00005923.pdf
Date de consultation 20-04-2016

[8] http://sdz.tdct.org/sdz/apprenez-a-utiliser-solidworks.html#ApprenezutiliserSolidWorks
Date de consultation 24-04-2016

[9] http://modelleisenbahn.triskell.org/spip.php?article59
Date de consultation 05-05-2016

[10] http://www.locoduino.org/spip.php?article57
Date de consultation 06-05-2016

A-1 Programme de notre maquette

```
// …… partie de déclaration des variables et constantes……..
#define EN1Pin 3
#define step1Pin 7
#define dir1Pin 6
#define EN2Pin 2
#define step2Pin 5
#define dir2Pin 4
#define right_Dcycl A0
#define up A2
#define down A1
#define left A3
#define Auto_man A4
#define capt_1 8
#define capt_2 9
#define capt_3 10
#define capt_4 11
#define capt_5 12
#define FC1_bas 0
#define FC2_haut A5 //1
//…….. déclatation de la constante du mode Stop……...
#define stop_mode (analogRead(Auto_man) < 400)
//……. Déclaration des variables ……..
unsigned char pos_chariot  = 0;
unsigned char pos_pincette = 0;
unsigned char pos_capteur=12;

//…… Partie de configuration des pins et l'interface série ……

void setup() {

   Serial.begin(9600);
   pinMode(step1Pin, OUTPUT);
   pinMode(dir1Pin, OUTPUT);
   pinMode(step2Pin, OUTPUT);
   pinMode(dir2Pin, OUTPUT);
   pinMode(EN1Pin, OUTPUT);
   pinMode(EN2Pin, OUTPUT);
   pinMode(right_Dcycl, INPUT);
   pinMode(left, INPUT);
   pinMode(up, INPUT);
   pinMode(down, INPUT);
   pinMode(13, OUTPUT);
```

```
  pinMode(FC1_bas, INPUT);
  pinMode(FC2_haut, INPUT);
  pinMode(capt_1, INPUT);
  pinMode(capt_2, INPUT);
  pinMode(capt_3, INPUT);
  pinMode(capt_4, INPUT);
  pinMode(capt_5, INPUT);
  digitalWrite(step2Pin, LOW);
 }

//……..Partie du programme principal ……..

void loop() {

 while(digitalRead(right_Dcycl) == HIGH)
 {
 }

  home_chariot();                    //   le chariot doit retourner à sa position home
  Serial.println("Goto HOME:");      //   afficher sur l'interface série goto home
  affichage_des_positions();         //   afficher les positions actuelles chariot,pincette et
  int  val_auto_man =analogRead(Auto_man);                    //      posiotion capteur
  delay(100);
  Serial.print("la valeur val_auto_man : ");
  Serial.println(val_auto_man);
  delay(600);

 //……. On teste si le commutateur auto_man est sur la position Auto ou Man……..

 if (val_auto_man > 500 && val_auto_man < 600 )       // commutateur en position Man
val_auto_man < 16 )                                   // fonctionnement en mode manual
 {

  Serial.println("\n\rStarting MAN mode:\n\r");
  while ( analogRead(Auto_man) > 500 && analogRead(Auto_man) < 600)    // mode MAN
  {
   if ( digitalRead(right_Dcycl) ==LOW )
   {
    if (( (  pos_chariot < 9 ) && ( pos_chariot > 0 ) ) || (pos_chariot >= 9) && ( pos_pincette ==0 ) )
    {
     Serial.println("\nRight:");
     if
((pos_chariot==2)||(pos_chariot==4)||(pos_chariot==6)||(pos_chariot==8)||(pos_chariot==10))
        {
         if (pos_chariot!=10) pos_capteur ++;
```

```
            deplacement_chariot_D( pos_capteur );
            if(pos_capteur > 12){pos_capteur=12;}

            if(pos_chariot==0){pos_capteur=12;}
            pos_chariot = pos_chariot - 2;
          }
        else if
((pos_chariot==1)||(pos_chariot==3)||(pos_chariot==5)||(pos_chariot==7)||(pos_chariot==9))
            {
              deplacement_chariot_D( pos_capteur );
              if(pos_capteur > 12){pos_capteur=12;}
              pos_chariot = pos_chariot - 1;
              if(pos_chariot==255){pos_chariot=0;}
            }
        affichage_des_positions();
        delay(100);
     }
    }
    else if ( digitalRead(up) ==LOW )
    {
      if (( pos_pincette ==1) || ( pos_pincette ==2 ))
        {
        // function up
          if (( pos_pincette ==1))
            {
              Serial.println("\nUp:");
              monte_p_FC2();
              pos_pincette --;
              affichage_des_positions();
              delay(100);
          }
          if (( pos_pincette ==2))
            {
              Serial.println("\nUp:");
              monte_p_FC2();
              pos_pincette -=2;
              affichage_des_positions();
              delay(100);
          }

        }
    }
    else if ( digitalRead(down) == LOW )
    {
```

```
      if  (( pos_pincette == 0 ) || (( pos_pincette == 1 ) && ( ( pos_chariot < 2 ) || ( pos_chariot > 8 ) ) )
)
      {
        // function down
       if (pos_pincette == 0 )
         {
          Serial.println("\nDown:");
          descente_p_poste_traitement();
          pos_pincette ++;
          affichage_des_positions();
          delay(100);
         }
       else if ( ( pos_pincette == 1 ) && ( ( pos_chariot < 2 ) || ( pos_chariot > 8 ) ) )
         {
          Serial.println("\nDown:");
          descente_p_FC1();
          pos_pincette ++;
          affichage_des_positions();
          delay(100);

         }
       }
    }
    else if ( digitalRead(left) ==LOW )     //left
    {
      if ( ( ( pos_chariot != 1 ) && ( pos_chariot < 10 ) ) || ( ( pos_chariot == 1 ) && ( pos_pincette ==0 )
) )
      {
        // function left
       if ( ( pos_chariot == 0 ) || ( pos_chariot == 2 ) || ( pos_chariot == 4 ) || ( pos_chariot == 6 ) || (
pos_chariot == 8 ) || ( pos_chariot == 9 ) )
        {
         Serial.println("\nLeft:");
         deplacement_chariot_G_2cm();
         pos_chariot ++;
         affichage_des_positions();
         delay(100);
        }
        else
        {
         Serial.println("\nLeft:");
         pos_capteur --;
         deplacement_chariot_G( pos_capteur);
         pos_chariot ++;
         affichage_des_positions();
```

```
        delay(100);
       }
      }
    }
   }
  }
else if(  val_auto_man > 750)                          // fonctionnement en mode Auto
{
   Serial.println("\r\nStarting mode AUTO\r\n");

   if (!stop_mode){                  //  si le mode stop n'est pas sollicité on exécute la suite
      Serial.println("\r\nPos-Pincette 2");          // si non sort de la boucle, affiche pos_pincette puis
      descente_p_FC1();                              // on fait déscendre la P à la position 2
      delay(500);
   }


   if (!stop_mode){
    Serial.println("\r\nPos-chario 1");
    deplacement_chariot_G_2cm();
    delay(1000);
   }

   if (!stop_mode){
    Serial.println("\r\nMontee de la piece");
    monte_p_FC2();
    delay(1000);
   }

   if (!stop_mode){
    Serial.println("\r\nDeplacement de la piece vers le Bac 3");
    deplacement_chariot_G(capt_2);
    delay(1000);
   }

   if (!stop_mode){
    Serial.println("");
    deplacement_chariot_G_2cm();
    delay(1000);
   }

   if (!stop_mode){
    Serial.println("\r\nTremper la piece dans le bac 3");
    descente_p_poste_traitement();
    Serial.println("\r\nTemporisation de 5 sec");
```

```
  delay(5000);
}

if (!stop_mode){
  Serial.println("\r\nMontee de la piece");
  monte_p_FC2();
  delay(1000);
}

if (!stop_mode){
  Serial.println("\r\nDepplacement de la piece vers le bac 1");
  deplacement_chariot_D(capt_4);
  delay(1000);
}

if (!stop_mode){
  deplacement_charot_G_2cm();
  delay(1000);
}

if (!stop_mode){
  Serial.println("\r\nRincage de la piece dans le bac 1");
  for(int i=0;i<3;i++){
     Serial.print("\r ");
     Serial.print( i+1);
     descente_p_poste_traitement();
     delay(3000);
     monte_p_FC2();
     delay(1000);
     if (stop_mode)break;
  }
}

if (!stop_mode){
  Serial.println("\r\n\nDepplacement de la piece vers le bac 2");
  deplacement_chariot_G(capt_3);
  delay(1000);
  deplacement_chariot_G_2cm();
  delay(1000);
}

if (!stop_mode){
  Serial.println("\r\nTremper la piece dans le bac 2");
  descente_p_poste_traitement();
  Serial.println("\r\nTemporisation 5 sec");
```

```
    delay(5000);
   }

   if (!stop_mode){
    monte_p_FC2();
    delay(1000);
   }

   if (!stop_mode){
    Serial.println("\r\nDechargement de la piece");
    deplacement_chariot_G(capt_1);
    delay(1000);
   }

   if (!stop_mode){
    deplacement_chariot_G_2cm();
    delay(1000);
   }

   if (!stop_mode){
    descente_p_FC1();
    delay(1000);
   }

   if (!stop_mode){
    deplacement_chariot_G_4cm();
    delay(500);
   }

   Serial.println("\r\nFIN de cycle");

 }

}


void home_chariot(){
//..................................... Fonction home du chariot .....................
      if (digitalRead(FC2_haut)==LOW){
    //*************************** montée de la pincette
*******************************
      digitalWrite(EN2Pin, HIGH);          //  activer le driver
      digitalWrite(dir2Pin,HIGH);
      for (int j=0; j<100; j++) {
            digitalWrite(step2Pin, HIGH);
```

```
                delay(30);
                digitalWrite(step2Pin, LOW);
                delay(30);
                if (digitalRead(FC2_haut)==HIGH){goto chariot;}
            }
        }
chariot:
        digitalWrite(EN2Pin, LOW);
        if (digitalRead(capt_5)==LOW){
                digitalWrite(EN1Pin,HIGH);
                digitalWrite(dir1Pin, LOW);
                for (int j=0; j<2700; j++) {
                        digitalWrite(step1Pin, HIGH);
                        delayMicroseconds(1000);
                        digitalWrite(step1Pin, LOW);
                        delayMicroseconds(1000);
                        if (digitalRead(capt_5)==HIGH) { goto arret;}
                    }
arret:
        digitalWrite(EN1Pin, LOW);

}
pos_capteur  = 12;
pos_chariot  = 0;
pos_pincette = 0;
}
//..... Fonction descente de la pincette à la position du capteur finde course FC1 .....

void descente_p_FC1() {

        digitalWrite(dir2Pin, LOW);
        digitalWrite(EN2Pin, HIGH);
        if(digitalRead(FC1_bas)==HIGH){goto arret1;}
           for (int j=0; j<60; j++) {
                digitalWrite(step2Pin, HIGH);
                delay(30);
                digitalWrite(step2Pin, LOW);
                delay(30);
                if(digitalRead(FC1_bas)==HIGH){goto arret1;}
          }

arret1:
      digitalWrite(EN2Pin, LOW);
      delay(100);
```

```
}

//..... Fonction déplacement chariot à gauche d'une distance de 2cm .....

void deplacement_chariot_G_2cm(){
        digitalWrite(dir1Pin, HIGH);
        digitalWrite(EN1Pin, HIGH);
        for (int j=0; j<197; j++) {
                digitalWrite(step1Pin, HIGH);
                delayMicroseconds(1000);
                digitalWrite(step1Pin, LOW);
                delayMicroseconds(1000);
     }

      digitalWrite(EN1Pin, LOW);
      delay(10);
}

//..... Fonction déplacement chariot à gauche d'une distance de 4cm .....

void deplacement_chariot_G_4cm(){
        digitalWrite(dir1Pin, HIGH);
        digitalWrite(EN1Pin, HIGH);
        for (int j=0; j<295; j++) {
                digitalWrite(step1Pin, HIGH);
                delayMicroseconds(1000);
                digitalWrite(step1Pin, LOW);
                delayMicroseconds(1000);
     }

      digitalWrite(EN1Pin, LOW);
      delay(10);
}

//..... Fonction montée de la pincette à la position du capteur fin de course FC2 .....

void monte_p_FC2() {

        digitalWrite(dir2Pin,HIGH);
        digitalWrite(EN2Pin, HIGH);

        if(digitalRead(FC2_haut)==HIGH){goto arret2;}
           for (int j=0; j<60; j++) {
                digitalWrite(step2Pin, HIGH);
                delay(30);
```

```
            digitalWrite(step2Pin, LOW);
            delay(30);
            if(digitalRead(FC2_haut)==HIGH){goto arret2;}
        }

arret2:
    digitalWrite(EN2Pin, LOW);
    delay(100);

}

//..... Fonction déplacement chariot à gauche à la position définie par le parametre pos_capteur
.....

void deplacement_chariot_G(int pos_capteur){
        int i;
        i=1000;
        digitalWrite(dir1Pin, HIGH);       // direction gauche du chariot
        digitalWrite(EN1Pin, HIGH);
        if(digitalRead(pos_capteur)==HIGH){goto arret3;}
        for (int j=0; j<2400; j++) {
                digitalWrite(step1Pin, HIGH);
                 delayMicroseconds(i);
                digitalWrite(step1Pin, LOW);
                 delayMicroseconds(i);
                if(digitalRead(pos_capteur)==HIGH){goto arret3;}
            }

arret3:
        digitalWrite(EN1Pin, LOW);
        delay(10);
}

//.............. Fonction descente de la pincette à la position pos_pincette 1 .................

void descente_p_poste_traitement() {
  digitalWrite(dir2Pin, LOW);
  digitalWrite(EN2Pin, HIGH);
  for (int j=0; j<33; j++){
    digitalWrite(step2Pin, HIGH);
    delay(30);
    digitalWrite(step2Pin, LOW);
    delay(30);
  }
```

```
  digitalWrite(EN2Pin, LOW);
  delay(10);
}

//..... Fonction déplacement chariot à droite à la position définie par le parametre pos_capteur .....

void deplacement_chariot_D(int pos_capteur){
        digitalWrite(dir1Pin, LOW);    //direction droite du chariot
        digitalWrite(EN1Pin, HIGH);
        if(digitalRead(pos_capteur)==HIGH){goto arret3;}
        for (int j=0; j<2400; j++) {
                digitalWrite(step1Pin, HIGH);
                delayMicroseconds(1000);
                digitalWrite(step1Pin, LOW);
                delayMicroseconds(1000);
                if(digitalRead(pos_capteur)==HIGH){goto arret3;}
              }

arret3:
        digitalWrite(EN1Pin, LOW);
        delay(10);
}

//..... Fonction d'affichage des différentes positions du chariot;pincette et position capteurs .....

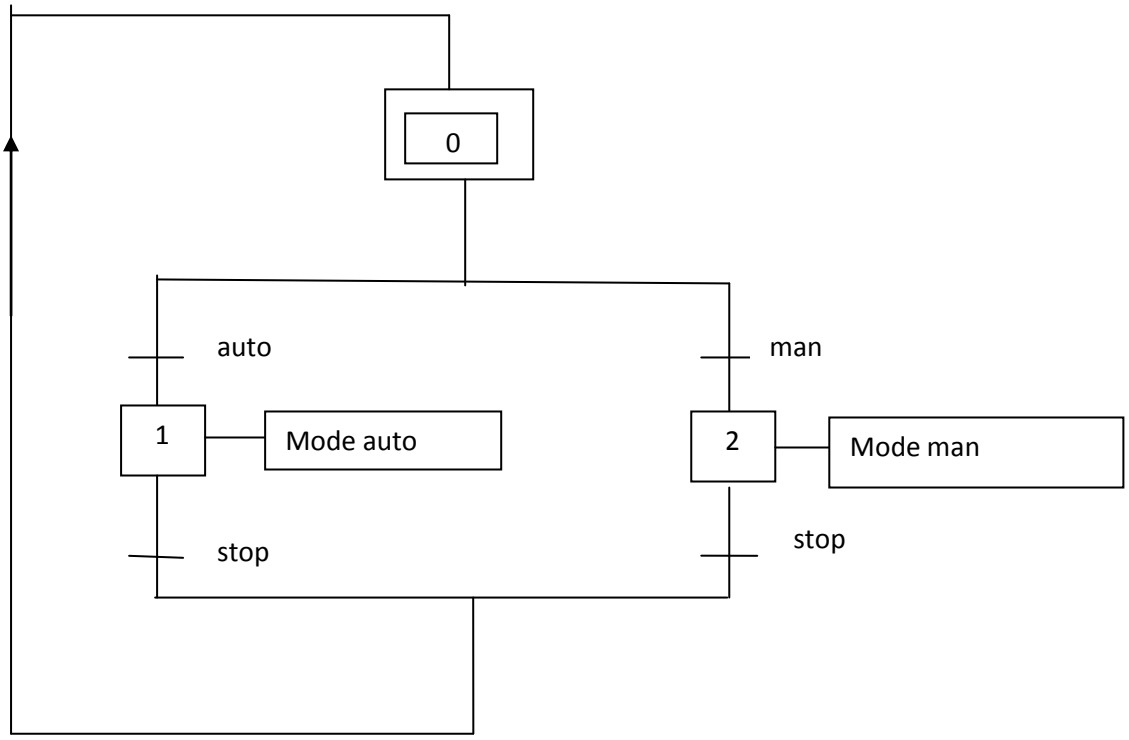
void affichage_des_positions() {

       Serial.print("pos_pincette : ");
       Serial.println(pos_pincette);
       Serial.print("pos_chariot  : ");
       Serial.println(pos_chariot);
       Serial.print("pos_capteur  : ");
       Serial.println(pos_capteur);
       delay(20);

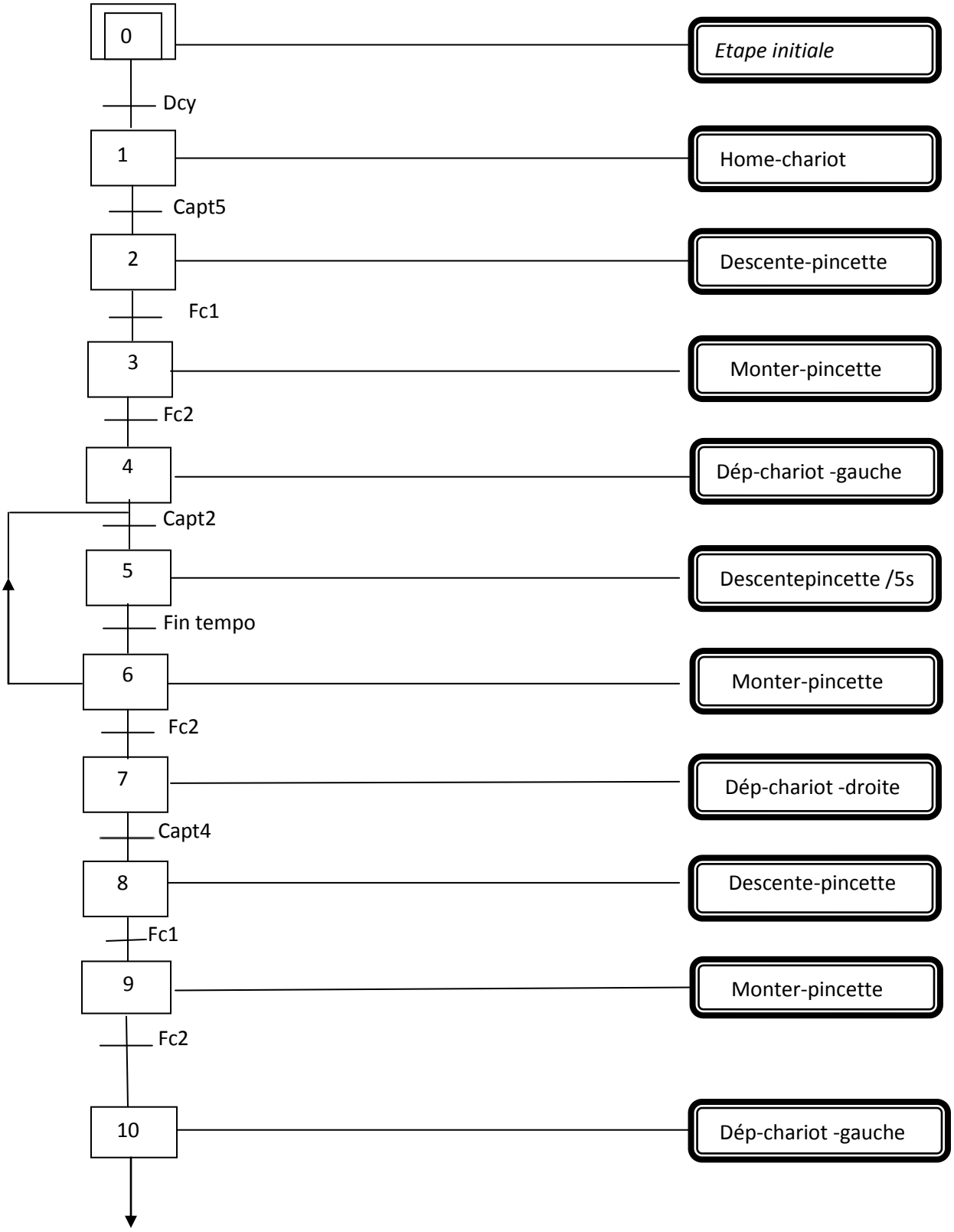
}
```

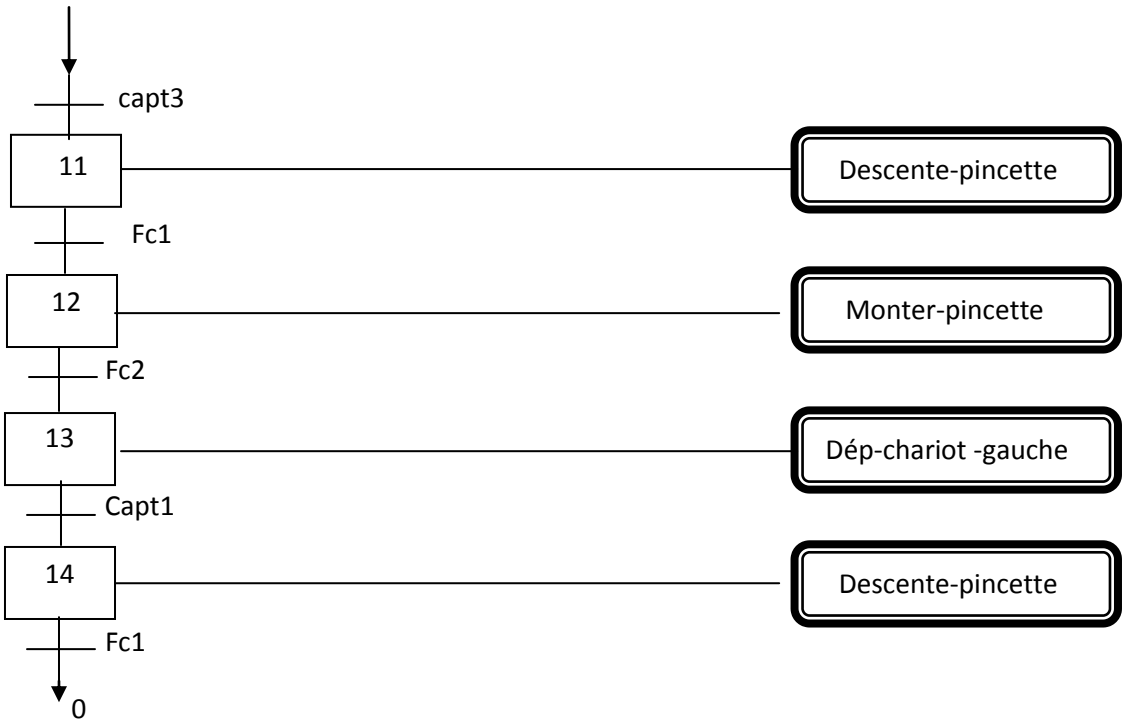
A-2 GRAFCET :

- **les deux modes** :

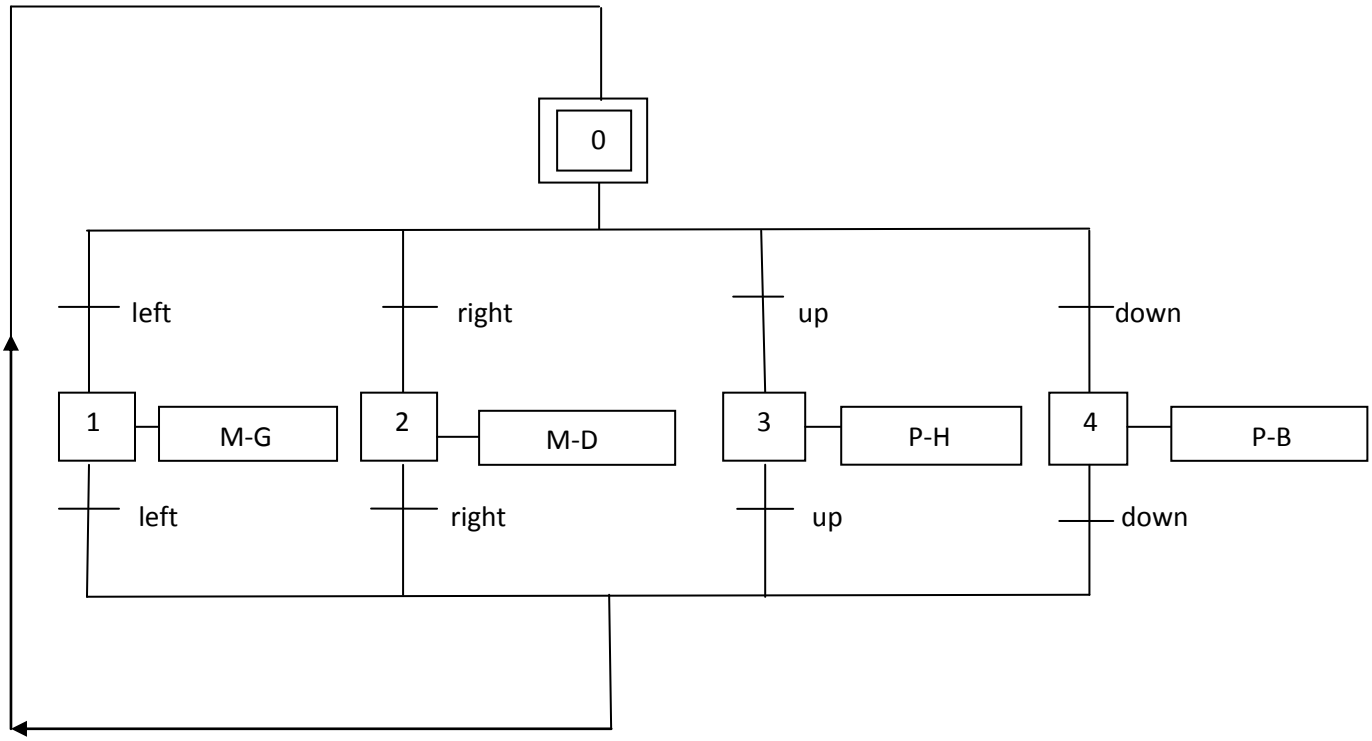


- **Mode Automatique** :



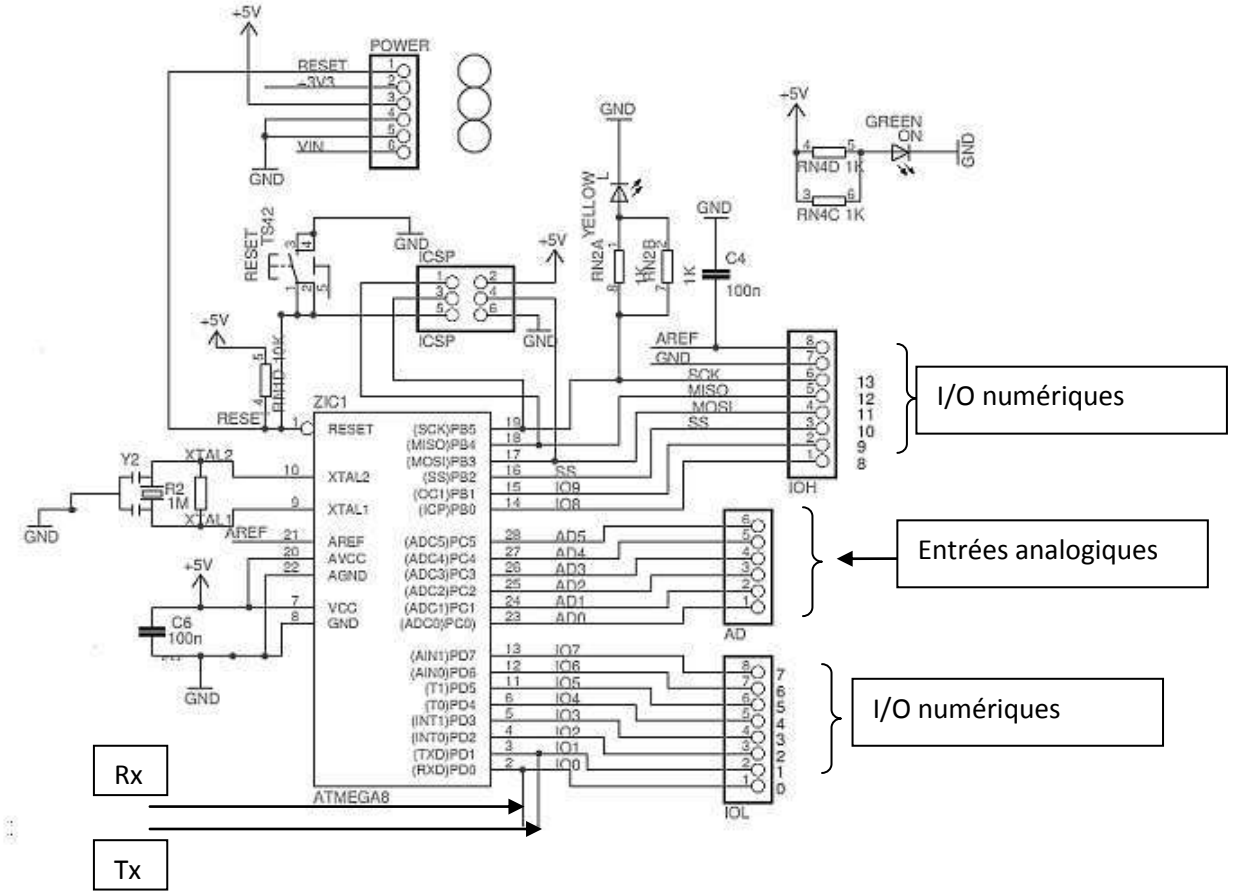

capt3
11
Descente-pincette
Fc1
12
Monter-pincette
Fc2
13
Dép-chariot -gauche
Capt1
14
Descente-pincette
Fc1
0

- **Mode Manuel**:

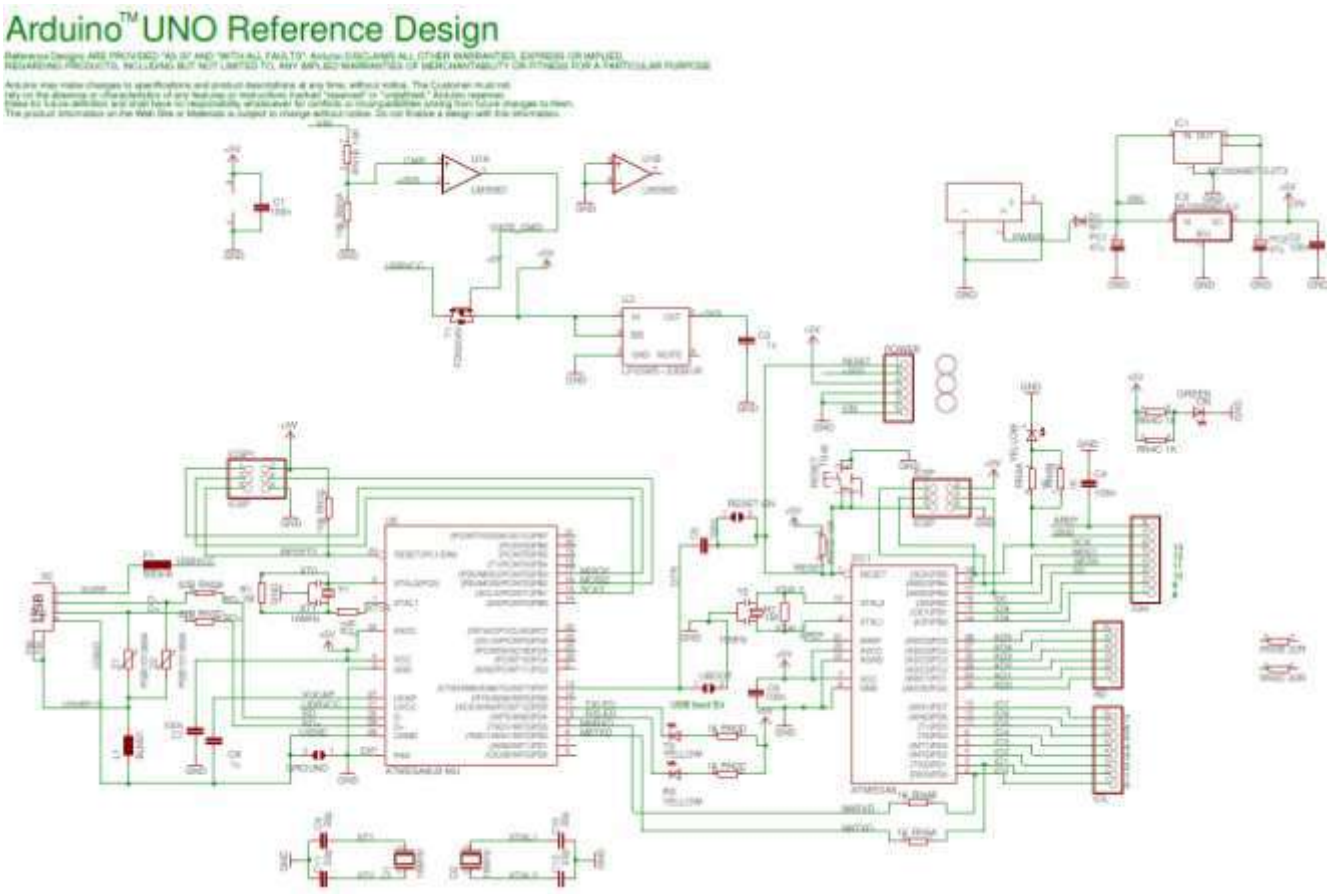


A-3 Arduino_Uno

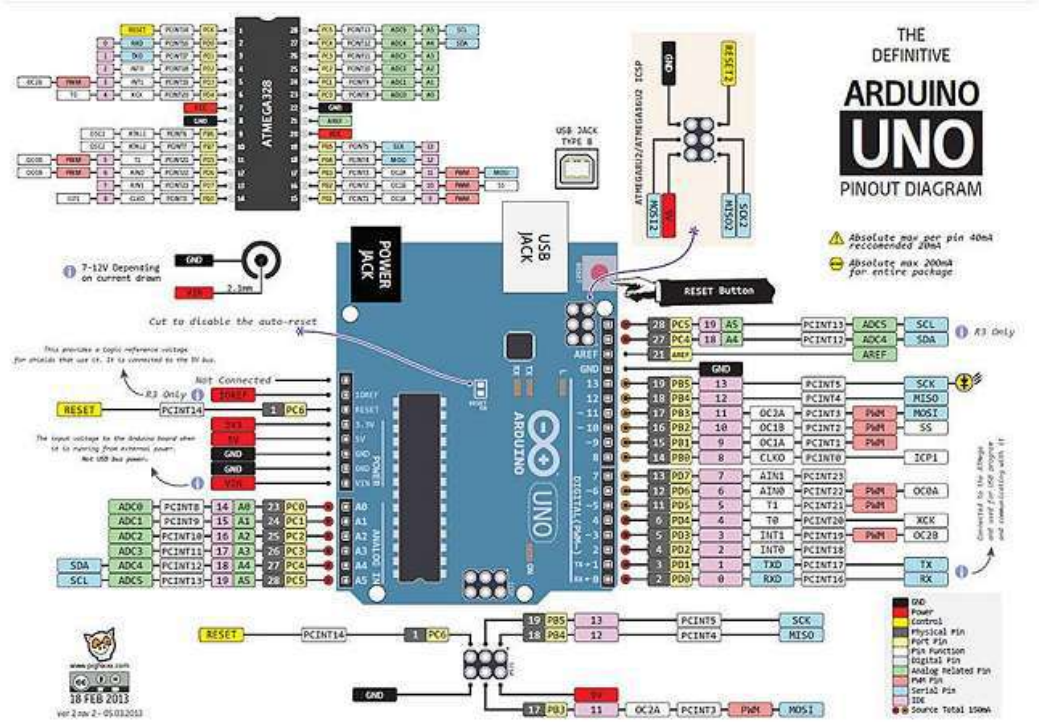
- **Schéma électrique simplifié**



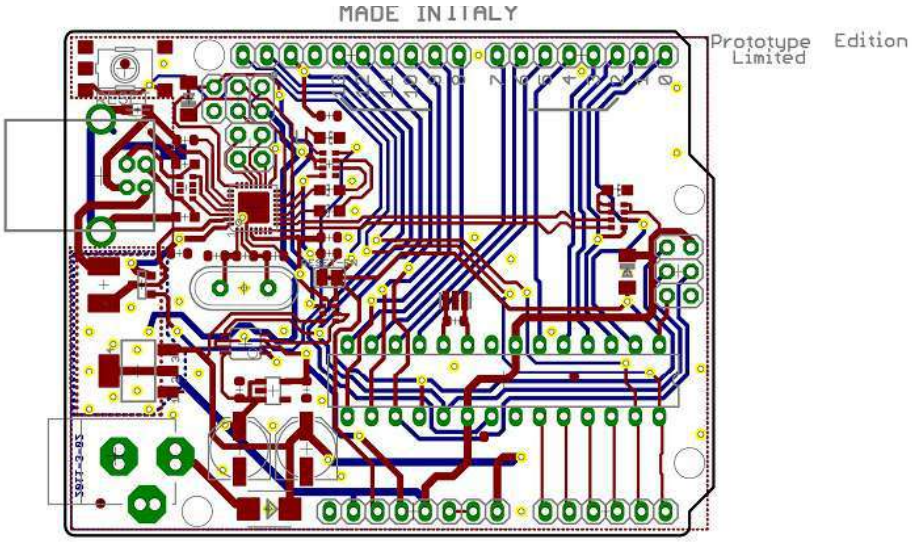
Le schéma montre les connexions du microcontrôleur Atega8 aux entrées/sorties de l'Arduino.



➢ Brochage de l'Arduino_Uno



➢ **Schéma d'implantation de l'Arduino_Uno**

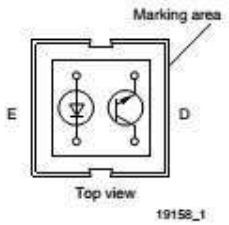


A-4 Capteur optique CNY70

➢ **Quelques caractéristiques**

Reflective Optical Sensor with Transistor Output





FEATURES

- Package type: leaded
- Detector type: phototransistor
- Dimensions (L x W x H in mm): 7 x 7 x 6
- Peak operating distance: < 0.5 mm
- Operating range within > 20 % relative collector current: 0 mm to 5 mm
- Typical output current under test: I_C = 1 mA
- Emitter wavelength: 950 nm
- Daylight blocking filter
- Lead (Pb)-free soldering released
- Material categorization: For definitions of compliance please see www.vishay.com/doc?99912

DESCRIPTION

The CNY70 is a reflective sensor that includes an infrared emitter and phototransistor in a leaded package which blocks visible light.

APPLICATIONS

- Optoelectronic scanning and switching devices i.e., index sensing, coded disk scanning etc. (optoelectronic encoder assemblies).

➢ Caractéristiques distances/courants

PRODUCT SUMMARY				
PART NUMBER	DISTANCE FOR MAXIMUM CTR_{rel} (1) (mm)	DISTANCE RANGE FOR RELATIVE I_{out} > 20 % (mm)	TYPICAL OUTPUT CURRENT UNDER TEST (2) (mA)	DAYLIGHT BLOCKING FILTER INTEGRATED
CNY70	0	0 to 5	1	Yes

(1) CTR: current transfere ratio, I_{out}/I_{in}
(2) Conditions like in table basic charactristics/sensors

➢ Valeurs maximales absolues

ABSOLUTE MAXIMUM RATINGS (T_{amb} = 25 °C, unless otherwise specified)				
PARAMETER	TEST CONDITION	SYMBOL	VALUE	UNIT
COUPLER				
Total power dissipation	$T_{amb} \leq 25$ °C	P_{tot}	200	mW
Ambient temperature range		T_{amb}	- 40 to + 85	°C
Storage temperature range		T_{stg}	- 40 to + 100	°C
Soldering temperature	Distance to case 2 mm, t £ 5 s	T_{sd}	260	°C
INPUT (EMITTER)				
Reverse voltage		V_R	5	V
Forward current		I_F	50	mA
Forward surge current	$t_p \leq 10$ µs	I_{FSM}	3	A
Power dissipation	$T_{amb} \leq 25$ °C	P_V	100	mW
Junction temperature		T_j	100	°C

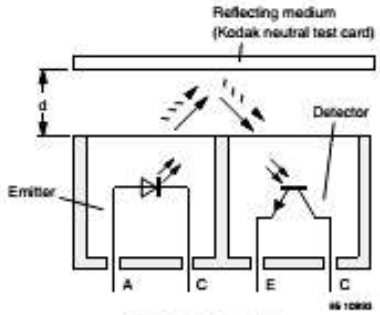


Fig. 2 - Test Condition

BASIC CHARACTERISTICS (T_{amb} = 25 °C, unless otherwise specified)

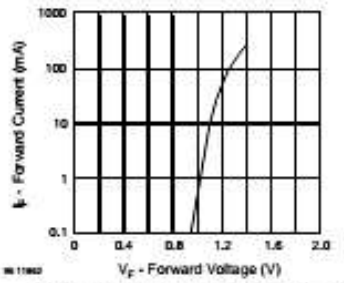


Fig. 3 - Forward Current vs. Forward Voltage

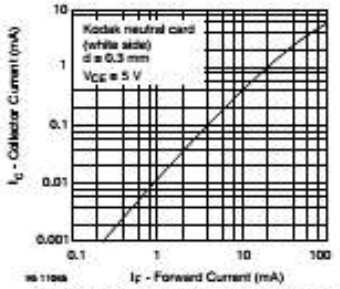


Fig. 5 - Collector Current vs. Forward Current

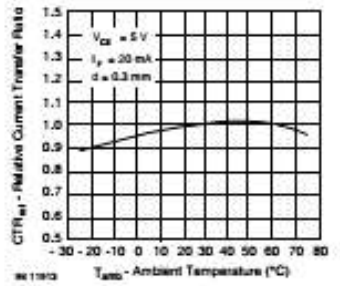


Fig. 4 - Relative Current Transfer Ratio vs. Ambient Temperature

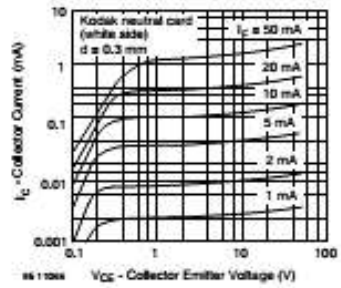
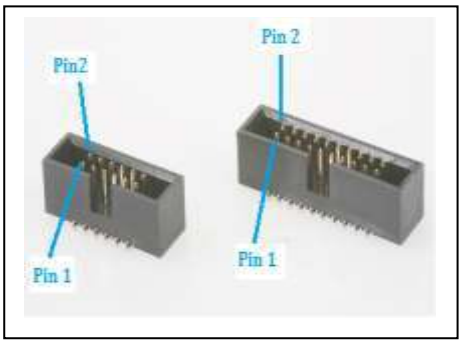


Fig. 6 - Collector Current vs. Collector Emitter Voltage

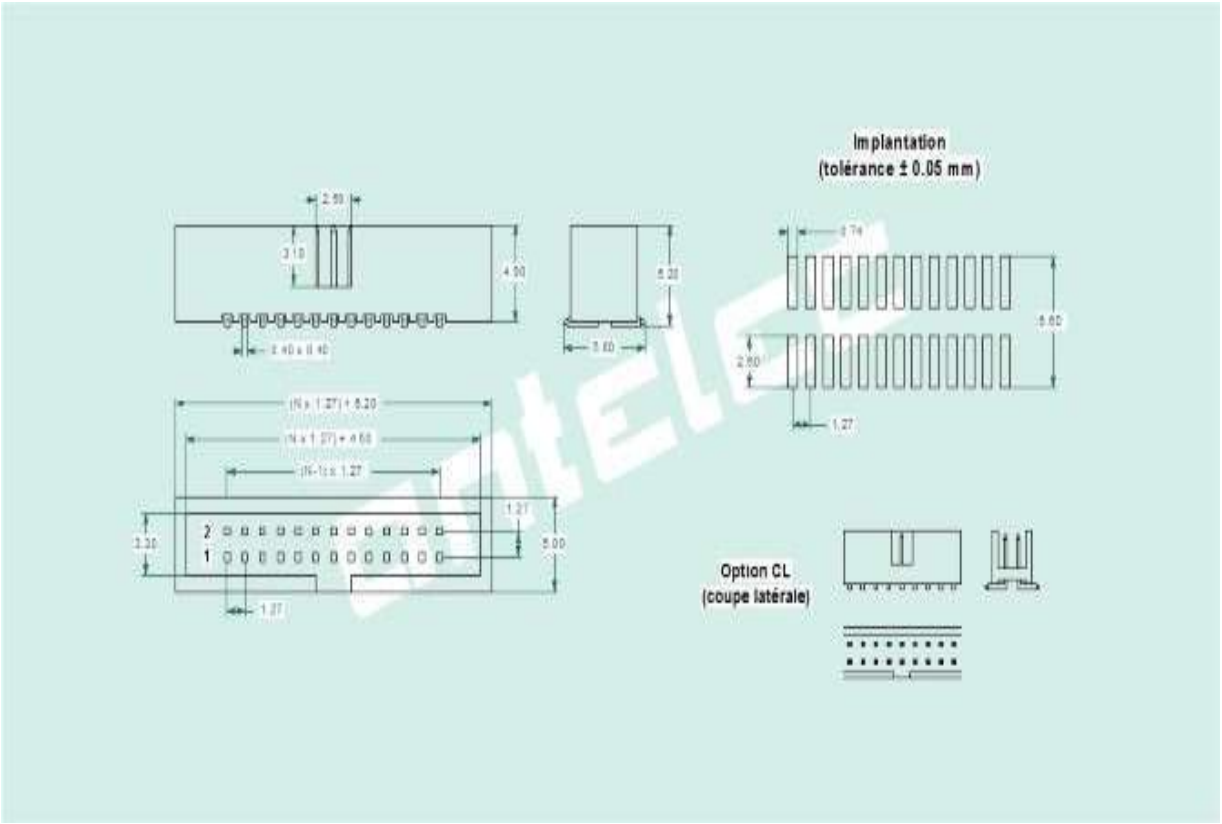
A-5 Connecteur type HE10 mâle droit

➢ Caractéristiques et spécifications

Intensité admissible : 1.0 A
Tenue diélectrique : 500 Volts AC pendant 1 minute
Résistance d'isolement : > 1000 MΩ
Résistance de contact : < 20 mΩ
Gamme de température : -55°C à +125°C
Température de brasure max : 260°C pendant 5 sec
Contact : alliage de cuivre
Finition : flash or (standard)
Isolant : nylon-6T, UL94V-0

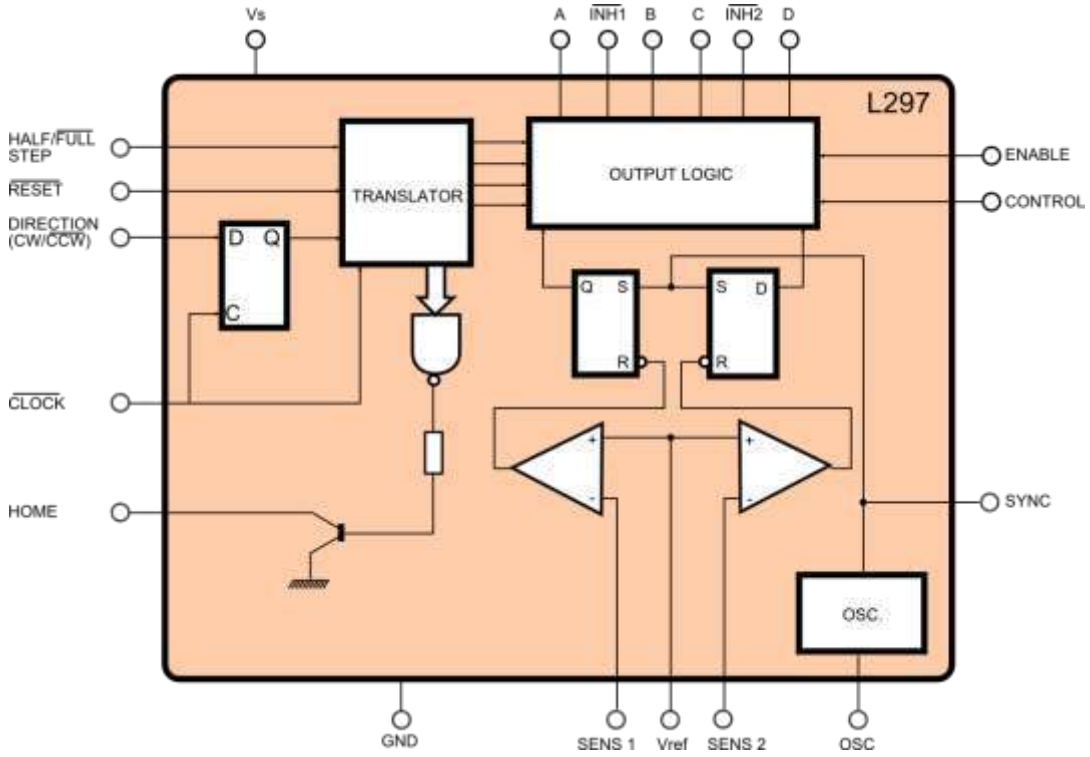


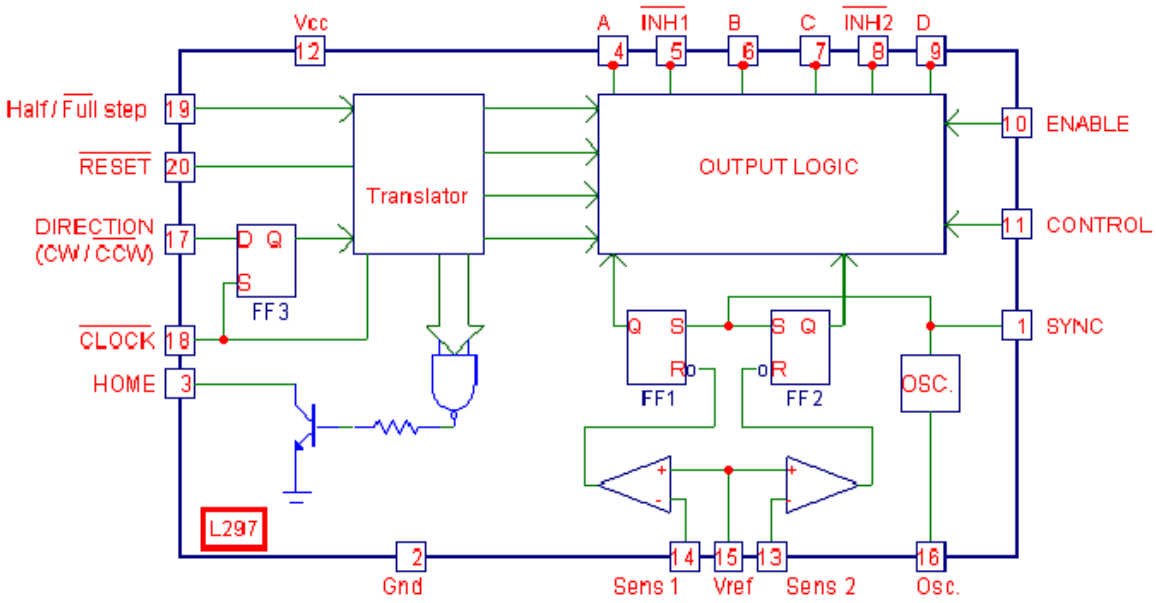
- Brochages correspondants



A-5 Circuits L297 et L298

➢ L e circuit L297





Le circuit **L297** reçoit les signaux de contrôle en provenance d'un système à base de microprocesseur et génère tous les signaux nécessaires pour l'étage de puissance.

Ce circuit inclue deux régulations à découpage type **PWM** (modulation de largeur d'impulsion) pour réguler le courant des enroulements moteur.

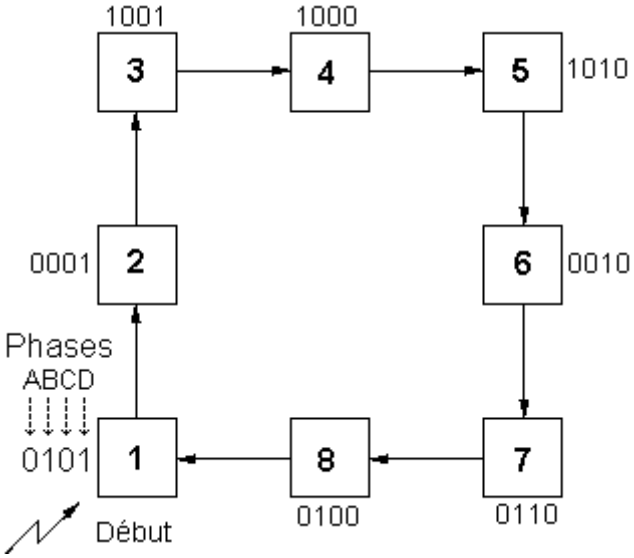
De plus il est capable de gérer le mode demi pas et pas complet. Toutefois, le **L297** est utilisable avec toute sorte d'étage de puissance et peut s'affranchir la commande des composants discrets tel que des transistors de puissance ; il fourni jusqu'à 20 mA sur ces sorties.

Le séquenceur

Le cœur du **L297** est le block appelé "translator" sur le schéma-block. C'est lui qui génère les séquences appropriées pour l'alimentation des phases en mode demi-pas, pas entier une phase / biphase.

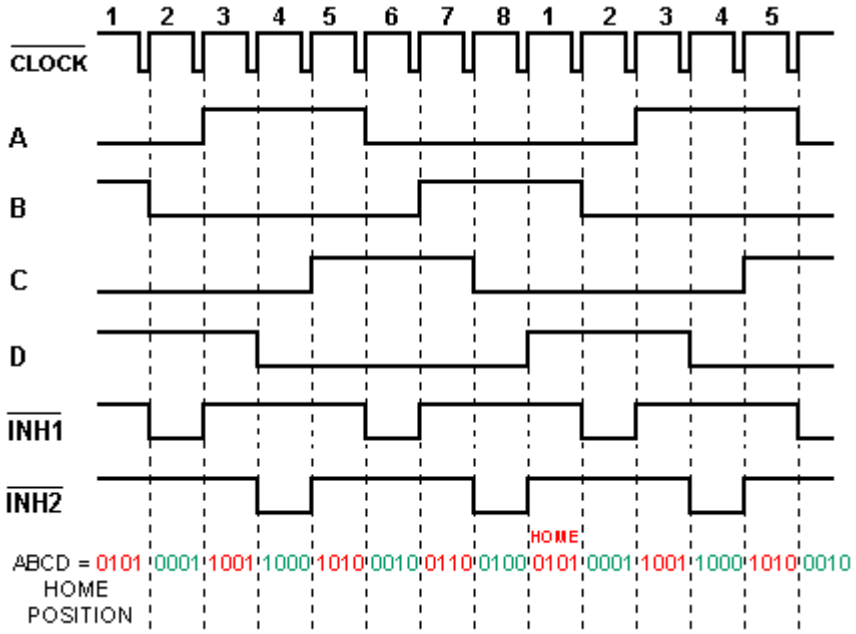
Ce block est contrôlé par deux entrées de mode - le mode direction (**CW / CCW**) et le mode pas / demi-pas (**HALF/ FULL**)- et l'entrée **clock** dont les impulsions d'horloge qu'on lui applique font évoluer le séquenceur vers le prochain pas.

La structure interne du séquenceur consiste en un compteur 3 bits (plus une logique combinatoire) qui génère les huit étape de base (en **code gray**) comme nous le voyons sur la figure suivante.



Sur ce diagramme nous sommes en mode demi-pas (**Half**) et en sens horaire (**CW**). Les trois séquences de commande possibles sont extraites directement de cette séquence principale.

Le chronogramme des sorties en mode demi pas est visible ci-après.


1 2 3 4 5 6 7 8 1 2 3 4 5
CLOCK
A
B
C
D
INH1
INH2
HOME
ABCD = 0101 0001 1001 1000 1010 0010 0110 0100 0101 0001 1001 1000 1010 0010
HOME
POSITION

➢ L e circuit L298

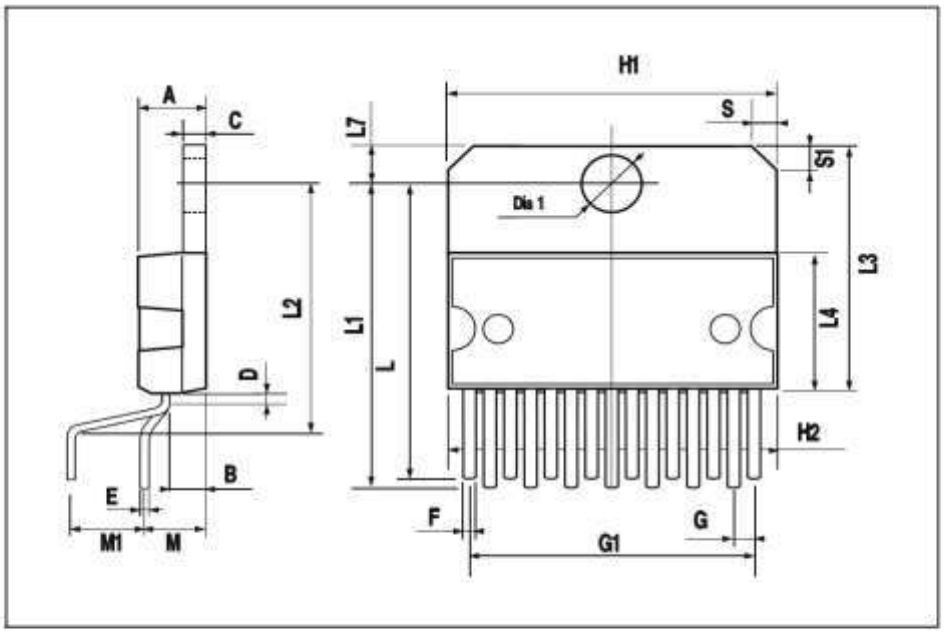
L298

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			5			0.197
B			2.65			0.104
C			1.6			0.063
D		1			0.039	
E	0.49		0.55	0.019		0.022
F	0.66		0.75	0.026		0.030
G	1.02	1.27	1.52	0.040	0.050	0.060
G1	17.53	17.78	18.03	0.690	0.700	0.710
H1	19.6			0.772		
H2			20.2			0.795
L	21.9	22.2	22.5	0.862	0.874	0.886
L1	21.7	22.1	22.5	0.854	0.870	0.886
L2	17.65		18.1	0.695		0.713
L3	17.25	17.5	17.75	0.679	0.689	0.699
L4	10.3	10.7	10.9	0.406	0.421	0.429
L7	2.65		2.9	0.104		0.114
M	4.25	4.55	4.85	0.167	0.179	0.191
M1	4.63	5.08	5.53	0.182	0.200	0.218
S	1.9		2.6	0.075		0.102
S1	1.9		2.6	0.075		0.102
Dia1	3.65		3.85	0.144		0.152

OUTLINE AND MECHANICAL DATA



Multiwatt15 V



Son brochage interne

