الجمهـــوريـــــة الجزائـــريـــــة الديمقـــراطيـــة الشـعبيـــة
République Algérienne Démocratique et Populaire
وزارة التـــعلـــيـــــم العـــــالـــــي والبـحـــــث العـلمـــي
Ministère de l'Enseignement Supérieure et de la Recherche Scientifique

جامعة وهران 2 محمد بن أ حمد
Universite d'Oran 2 Mohamed Ben Ahmed

معهد الصيانة و الأمن الصناعي

Département de Maintenance en Instrumentation

MÉMOIRE

Pour l'obtention du diplôme de Master

Filière : Génie Industriel
Spécialité : Génie Industriel

Thème

Réalisation d'un Automate Programmable à Base d' ARDUINO

Présenté et soutenu publiquement par :

SAHRAOUI SIDALI

HAOUACINE HAMZA

Devant le jury composé de :

Mr. HACHEMI Khalid	Prof	Univ. Oran-2, IMSI	**Président**
Mr. ADDA NEGGEZ Samir	MAA	Univ. Oran-2, IMSI	**Encadreur**
Mr. NEKROUF Djilali	MAA	Univ. Oran-2, IMSI	**Examinateur**

Année 2021/2022

Remerciement

Avant tout, je remercie Allah le tout puissant de m’avoir donné le courage, la volonté et la patience durant toutes les années d'études et que grâceà lui ce travail a pu être réalisé.

Mes vifs remerciements vont en premier lieu, à mes chers parents, familles etcamarades de m’avoir aidé, encouragées et soutenus tout au long de ces annéeset qui continuent de croire en moi en dépit de tout.

Comme je tiens à exprimer ma profonde gratitude à mon encadreur Mr. AddaNeggaz qui a cru en moi, ainsi que me soutenir et m’orienter tout au long de mon travail.

Qu'il trouve ici l'expression de ma reconnaissance.

Je remercie aussi les membres de jury qui mon fait l’honneur de juger ceProjet, j’espère être à la hauteur de leurs attentes.

Je suis très reconnaissant à tous les enseignants qui mon veillé au bon déroulement de mes formation tout au long de mes cursus. Qu'ils trouvent icil'expression de mon respect et remerciements les plus sincère.

Un grand merci aussi à toute personne qui de près ou de loin a contribué à ceque ce modeste travail.

DÉDICACE

Je dédie ce travail d'abord à ma mère et mon père qui m'ont encouragé toute ma vie et m'ont toujours soutenu

De même, mes frères Yazid et Hicham, qui se sont sacrifiés pour moi à tous égards, notamment

matériels et moraux, sans oublier leurs petites familles, précisément Abd El Basset et Hadil.

Je dédie également mon travail à mes deux chères sœurs et leurs petite familles, précisément Ahmed et Maram

Dédicace

A tout les Sahraoui

A tous mes collègues monsieur 2

A mon très cher binôme .

À toute personne que j'ai connue lors de mon passage à l'université.
A vous tous, je dédie ce modeste travail...

À tous nos professeurs qui nous ont enseignés car si nous sommes là aujourd'hui, c'est bien grâce à vous.

Donc, un grand merci pour vous

SIDALI

DÉDICACE

A Mon père, au meilleur ami de la vie ; Mr Abdel- Madjid, à

La plus belle créature que Dieu a créé sur terre.

À cette source de tendresse, de patience et de générosité.

À ma mère

À tous mes frères et sœurs

À toute ma famille, Les Haouacine

À tous mes amis et collègues

A mon très cher binôme .

À tous les étudiants de la promotion 2021/2022

Option : Génie industriel.

A tous ceux qui, par leurs mots, m'ont donné la force de continuer…

HAMZA

LISTE DES FIGURES

LISTE DES TABLEU

Sommaire

Introduction générale

Le monde de l'industrie a connu ces dernières années des bouleversements importants, les machines automatiques ont été développées pour libérer l'homme de ses tâches quotidiennes, et aussi pour le remplacer dans l'exécution des travaux nombreux.

Les progrès de l'électronique et de l'informatique, ont donné naissance aux automates programmables industriels (API) et d'autres microcontrôleurs qui peuvent s'adapter et s'intégrer dans les processus industriels. Ils peuvent accomplir des tâches plus complexes, non seulement de contrôle, mais aussi de traitement de données, de circulation d'informations et de simulation. Et parmi ces nombreux exemplaires des automates programmables,

Les tapis roulent ont toujours été essentiels dans les industries et important importants dans le cas des grandes usine pour la rapidité et la qualité bine sur et pour ce la nous avons réalisés avec mon binôme ce travaille, Dans ce projet on cherche à résoudre cette problématique en proposant une maquette. Notre programme est basé sur une carte Arduino Méga

Organisation de mémoire :

Dans le chapitre 1 nous avons parlé sur l'automate programmable en générale et le système automatisé et donne une explication de L'API globalement

Et dans le deuxième chapitre, nous avons discutée sur les logiciels que nous avons utilisés dans notre travail Grafcet et Ladder .et IDE

Et en suite dans le troisième chapitre, nous avons parlé sur les carte Arduino et les caractéristiques de chaque carte que ce soit la taille et la capacité, et bien sur parlé de la carte méga 2560 car c'est avec cela que nous avons travaillé.

Et dans le dernier chapitre nous avons réalisé un tapis roulent, avec la simulation.

Chapitre I
Automate Programmable

I.1. Introduction :

L'objectif du développement de la recherche scientifique dans tous les domaines est d'aboutir à un progrès technologique voué principalement au bien être de l'humanité. Par sa nature de chercher à avoir la quantité, la qualité, le confort avec le minimum d'effort et du coût, l'homme a commencé à réfléchir, à concevoir et à réaliser des systèmes autonomes qui peuvent lui assurer tout dont il a besoin. Par conséquent, les outils et les appareils à énergies musculaires actionnés par des opérateurs sont remplacés par leurs équivalents à énergies électriques, mécaniques ou hydrauliques. La substitution d'opérateurs par ces systèmes définit l'automatisation.

La figure ci-dessous montre un schéma synoptique simplifié d'un système automatisé.

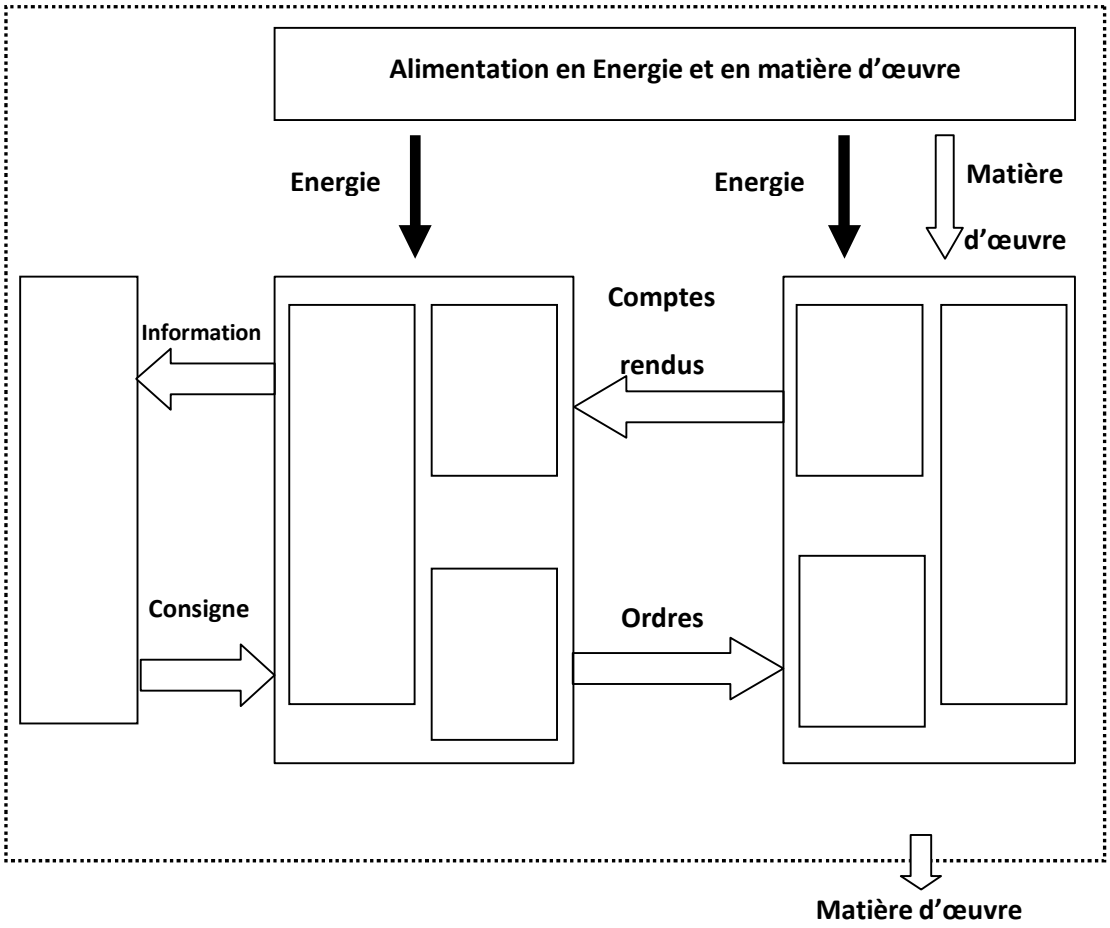


Figure I.1 : Schéma de principe d'un système automatisé

I.2. Historique de l'automate :

On sait maintenant que la deuxième partie XXème siècle a passé à l'histoire comme étant l'ère de l'automatique.

Dans le sillon de l'automatique apparurent plusieurs autre « tiques » et entre autre, l'informatique et la robotique et c'est à travers l'automatique, d'abord en 1968-69 aux Etats-Unis, que les premiers automates industriels ou « contrôleurs programmables » firent leur apparition.

Leurs premières applications furent d'abord le remplacement des horloges de contrôle du temps des employées par la suite, leurs multiples utilisations industrielles, en particulier sur leslignes de production des usines, deviennent indispensables non seulement au point de vue contrôle, mais aussi du côté économique pour l'espace et l'entretien, c'est alors que de nombreux systèmes a relais durent céder leur place.

Les premiers automates programmables n'effectuaient que la commutation ON/OFF (et vice-versa) avec la possibilité de temporisation, comme les relais, leurs applications, étaient limités seulement aux procédés répétitifs ainsi qu'à certaines machines. Par contre, leurs avantages consistaient dans une installation plus facile, la visualisation des étapes, ils possèdent des indicateurs diagnostiques permettant la localisation des pannes. C'est déjà mieux que les relais, en plus de pouvoir être reprogrammé advenant un changement de fonction ou de procédé.

De 1970 à 1974, la technologie des microprocesseurs (du moins les premiers) ajoutèrent une plus grande flexibilité et une « intelligence » à l'automate programmable. Les capacités d'interface avec l'utilisateur s'améliorent l'automate programmable peut maintenant exécuter des taches complexes.

Les opérations arithmétiques en plus des opérations logiques, il manipule les données et les adresses, il effectue la communication avec d'autres automates ou ordinateurs, donnant ainsiune nouvelle dimension aux applications de l'automate programmable. [1]

I.3. Définition de l'automatisme

L'automatisation consiste à « rendre automatique » les opérations qui exigeaient auparavant l'intervention humaine. Elle est considérée comme l'étape d'un progrès technique ou apparaissent des dispositifs techniques susceptibles de seconder l'homme, non seulement dans ses efforts musculaires, également dans ce travail intellectuel de surveillance et de contrôle

Un automatisme est un sous-ensemble d'une machine, destiné à remplacer l'action de l'être humain dans des tâches en général simples et répétitives, réclamant précision et rigueur. On est passé d'un système dit manuel, a un système automatisé.

Dans l'industrie, les automatismes sont devenus indispensables : ils permettent d'effectuer quotidiennement les tâches les plus ingrates, répétitives et dangereuses. Parfois, ces automatismes sont d'une telle rapidité et d'une telle précision, qu'ils réalisent des actions impossibles pour un être humain. L'automate est donc synonyme de productivité et de sécurité.Le savoir –faire de l'opérateur est transposé dans le système automatisé. Il devient le processus. Un processus peut être considéré comme un système organisé d'activités qui utilise des ressources (personnel, équipement, matériels et machine, matière première et information) pourtransformer des éléments entrants en élément de sortie dont le résultat final attendu est un produit.
Un système automatisé est composé de deux parties, la partie commande et la partie opérative. [2]

I.4. La partie opérative « O.P » :

Appelées parfois partie puissance, la partie opérative d'un automatisme assure la transformation de la matière d'œuvre .Les actionneurs : convertissent l'énergie d'entrée disponible sous une certaine forme (électrique, pneumatique, hydraulique) en une énergie utilisable sous une autre forme, par exemple :

- *Energie thermique* destinée a chauffé un four (l'actionneur étant alors une résistance électrique).
- *Energie mécanique* destinée a provoqué une translation de chariot (l'actionneur pouvant être un vérin hydraulique ou pneumatique) Energie mécanique destinée à provoquer une rotation de broche (l'actionneur pouvant être un moteur électrique). Les prés actionneurs : reçoivent les signaux de commande et réalisent la commutation de puissance avec les actionneurs. Les prés actionneurs des moteurs électriques sont appelés contacteurs. Les prés actionneurs des vérins et des moteurs hydraulique et pneumatiques sont appelés distributeurs (à commande électrique ou pneumatique).

Les capteurs : qui communiquent à la partie commande des informations sur la positiond'un mobile, une vitesse, la présence d'une pièce, une pression…..

- *Les capteurs T.O.R* (tout ou rien), qui délivrent un signal de sortie logique, c'est-à-dire0 ou 1. Exemple « détecteur de fin de course ».
- *Les capteurs numériques*, ou « incrémentaux », qui associés à un compteur délivrent des signaux de sortie numérique. Exemple « capteur ou codeur incrémental utilisé pour la mesure des déplacements des chariots de machine à commande numérique ».
- *Les capteurs analogiques* : ou en compte la valeur réelle d'une grandeur physique. Exemple « sonde de température ».

Les appareils de ligne : ceux-ci représentent l'ensemble des composants indispensables à la mise en œuvre et à la bonne marche de l'automatisme.

a) La partie commande :

La Partie Commande est l'ensemble des moyens logiciels et d'informations concernant le pilotage et la conduite du procédé. Elle donne des ordres à la partie opérative à travers les actionneurs et reçoit ses comptes rendus grâce aux capteurs.

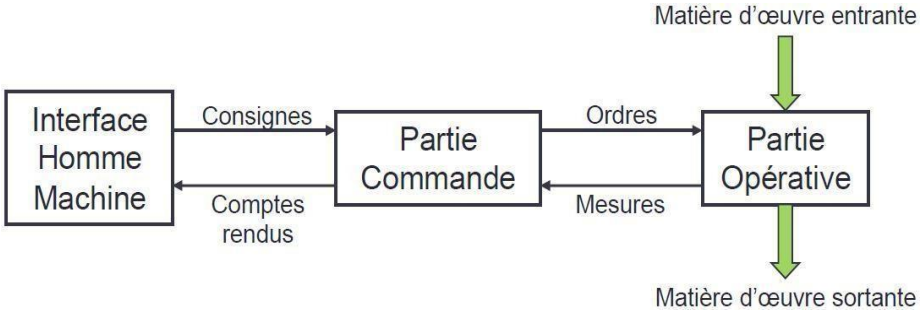


Figure I.2 : Schéma explicatif montrant le rôle de la partie commande et la partie opérative dans unsystème automatisé

I.5. Les systèmes automatisés :

"Depuis toujours l'homme est en quête de bien être". Cette réflexion peut paraître bien Eloignée d'un cours de Sciences Industrielles, pourtant c'est la base de l'évolution des sciences en général, et de l'automatisation en particulier. L'homme a commencé par penser, concevoir et réaliser. Lorsqu'il a fallu multiplier le nombre d'objets fabriqués, produire en plus grand nombre, l'automatisation des tâches est alors apparue : remplacer l'homme dans des actions pénibles, délicates ou répétitives. Citons pour exemple quelques grands hommes, avec les premiers développements de l'ère industrielle au XVIIIème siècle, Watt, avec ses systèmes de régulation à vapeur, Jacquard et ses métiers à tisser automatiques... Une liste exhaustive serait bien difficile à établir !

Enfin, le développement des connaissances, et des outils mathématiques, ont conduit à un formidable essor des systèmes automatisés, et des systèmes asservis, dans la deuxième moitié du 20ème siècle. L'automatisation consiste à « *rendre automatique* » les opérations qui exigeaient auparavant l'intervention humaine, elle est considérée comme l'étape d'un progrès technique ou apparaissent des dispositifs susceptibles de seconder l'homme, non seulement dans ses efforts musculaires, mais également dans son travail intellectuel de surveillance et de contrôle.

Dans l'industrie, les automatismes sont devenus indispensables : ils permettent d'effectuer quotidiennement les taches les plus pénibles, répétitives et dangereuses. Parfois, ces automatismes sont d'une telle rapidité et d'une telle précision, qu'ils réalisent des actions impossibles pour un être humain. L'automatisme est donc synonyme de productivité et de sécurité.

Le savoir-faire de l'opérateur est transposé dans le système automatisé, il devient Le «Processus ». Un processus peut être considéré comme un système organisé qui utilise des ressources (personnel, équipement, matériels et machines, matière première et informations) pour transformer des éléments entrants (les intrants) en éléments de sortie (les extrants) dont lerésultat final attendu est une production.

Figure I.3 : Chaine de production automatisée

I.6. Définition d'automate programmable industriel :

Les Automates Programmables Industriels (API) sont apparus aux Etats-Unis vers 1969 où ils répondaient aux désirs des industries de l'automobile de développer des chaînes de fabrication automatisées qui pourraient suivre l'évolution des techniques et des modèles fabriqués.

Un Automate Programmable Industriel (API) est une machine électronique programmable par un personnel non informaticien et destiné à piloter en ambiance industrielle et en temps réel des procédés industriels. Un automate programmable est adaptable à un maximum d'application, d'un point de vue traitement, composants, langage. C'est pour cela qu'il est de construction modulaire .Il est en général manipulé par un personnel électromécanicien. Le développement de l'industrie à entraîner une augmentation constante des fonctions électroniques présentes dans un automatisme c'est pour ça que l'API s'est substitué aux armoires à relais en raison de sa souplesse dans la mise en œuvre, mais aussi parce que dans les coûts de câblage et de maintenance devenaient trop élevés. [3]

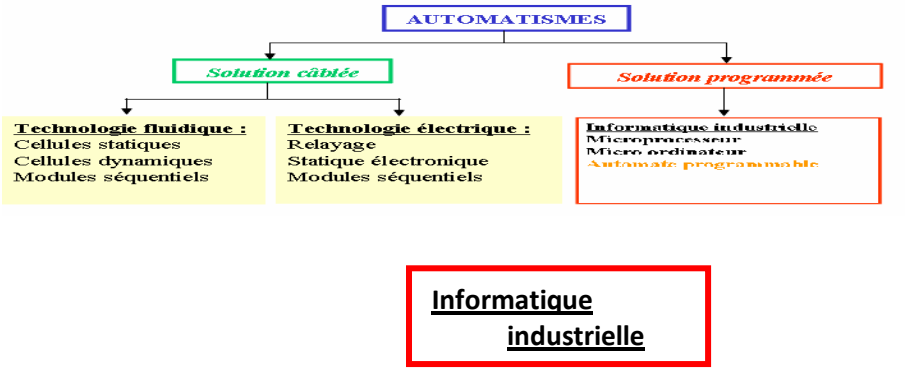


Figure I.4 : Différent technologie automate.

I.7. Les avantages et inconvénient de L'API :

L'A.P.I. est un équipement spécialement conçu pour l'industrie et destiné à piloter des chaînes de montages, productions, manutentions, robots industriels, machines outils …
Au point de vue modélisme ferroviaire c'est un appareil parfait pour qui veut automatiser même partiellement un réseau. Par contre il ne faut pas oublier qu'à l'instar d'un réseau réel, un grand nombre d'entrées/sorties seront nécessaires. La petite démonstration ci-dessus ne demande pas moins de 4 entrées et 5 sorties. [3]

Figure I.5 : Exemple d'API

Les avantages :

- Simplification du câblage.
- Modifications du programme faciles à effectuer par rapport à une logique câblée.
- Enormes possibilités d'exploitation.
- Fiabilité professionnelle.

Les inconvénients :

- En cas de "plantage" (très rare heureusement), c'est une belle pagaille…
- Son prix qui comme nous l'avons vu plus haut ne le met pas à la portée de toutes les bourses. Mais ces équipements évoluant rapidement fait que l'on peut en récupérer quelquefois pour pas trop cher.

I.8. Les caractéristiques de L'API :

- Compact ou modulaire.
- Taille de la mémoire.
- Sauvegarde (EPROM, EEPROM, pile, …).
- Nombre d'entrées/sorties E/S.
- Modules complémentaires (analogique, communication).
- Langage de programmation.
- Tension d'alimentation. [4]

I.9. Structure d'un Automate Programmable :

I.9.1 Structure générale des API :

Un API comprend généralement des modules arrangés l'un à côté de l'autre, tels qu'une alimentation, une unité centrale (CPU) à base de microprocesseur dotée d'une carte de mémoire, des interfaces d'entrées et de sorties, des interfaces de communication, des cartes spéciaux et un dispositif de programmation. On peut effectivement considérer qu'il s'agit d'une unité contenant un grand nombre de relais, compteurs, temporisateurs et unités de stockage de données distincts (généralement EEPROM). [5]

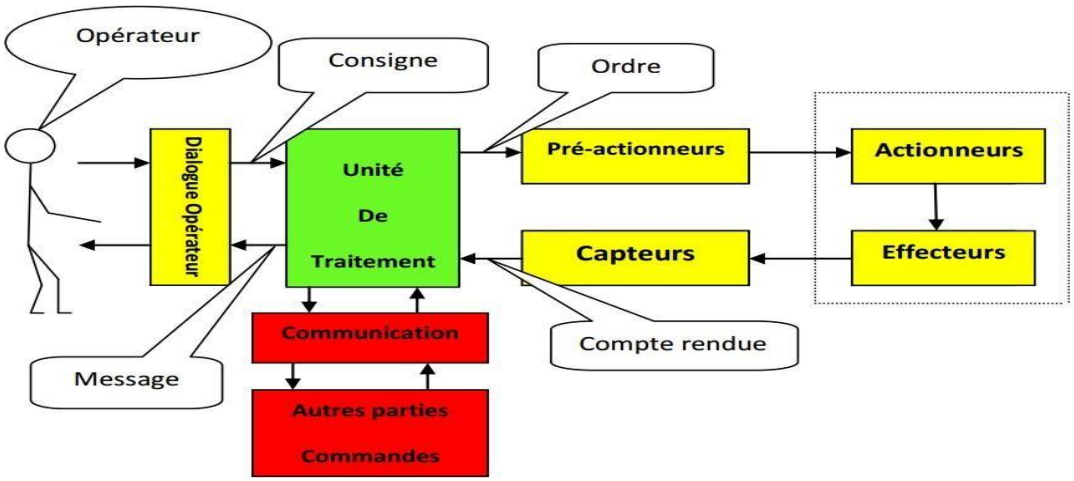


Figure I.6: Structure d'un système automatisé

I.9.2 Structure interne d'un automate programmable industriel (API) :

La structure interne d'un automate programmable industriel (API) est assez voisine de celle d'un système informatique simple, L'unité centrale est le regroupement du processeur et de la mémoire centrale. Elle commande l'interprétation et l'exécution des instructions programme. Les instructions sont effectuées les unes après les autres, séquencées par une horloge.
Les API comportent quatre principales parties :

- Une unité de traitement (un processeur CPU);
- Une mémoire ;
- Des modules d'entrées-sorties et des interfaces d'entrées-sorties ;
- Une alimentation 230 V, 50/60 Hz (AC) - 24 V (DC).

Deux types de mémoire cohabitent :

- La mémoire Programme où est stocké le langage de programmation. Elle est en général figée, c'est à dire en lecture seulement (ROM : mémoire morte).
- La mémoire de données utilisable en lecture-écriture pendant le fonctionnement c'est la RAM (mémoire vive). Elle fait partie du système entrées-sorties. Elle fige les valeurs (0 ou 1) présentes sur les lignes d'entrées, à chaque prise en compte cyclique de celle-ci, elle mémorise les valeurs calculées à placer sur les sorties.

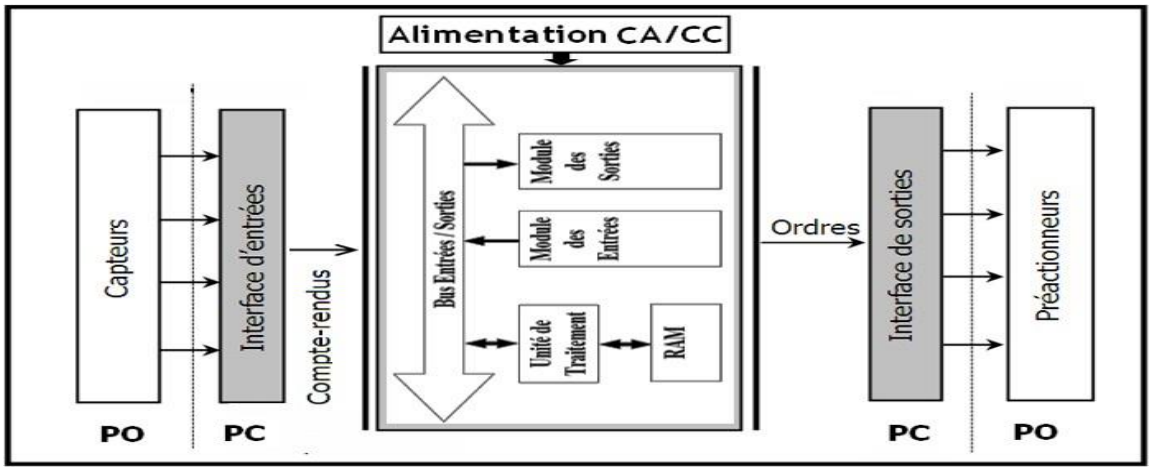


Figure I.7: Structure interne d'un automate programmable industriel API

I.9.3 Fonctionnement :

L'automate programmable reçoit les informations relatives à l'état du système et puis commande les pré-actionneurs suivant le programme inscrit dans sa mémoire.

Généralement les automates programmables industriels ont un fonctionnement cyclique (Figure 4.5). Le microprocesseur réalise toutes les fonctions logiques ET, OU, les fonctions de temporisation, de comptage, de calcul... Il est connecté aux autres éléments (mémoire et interface E/S) par des liaisons parallèles appelées 'BUS' qui véhiculent les informations sous forme binaire. Lorsque le fonctionnement est dit synchrone par rapport aux entrées et aux sorties, le cycle de traitement commence par la prise en compte des entrées qui sont figées en mémoire pour tout le cycle. [5]

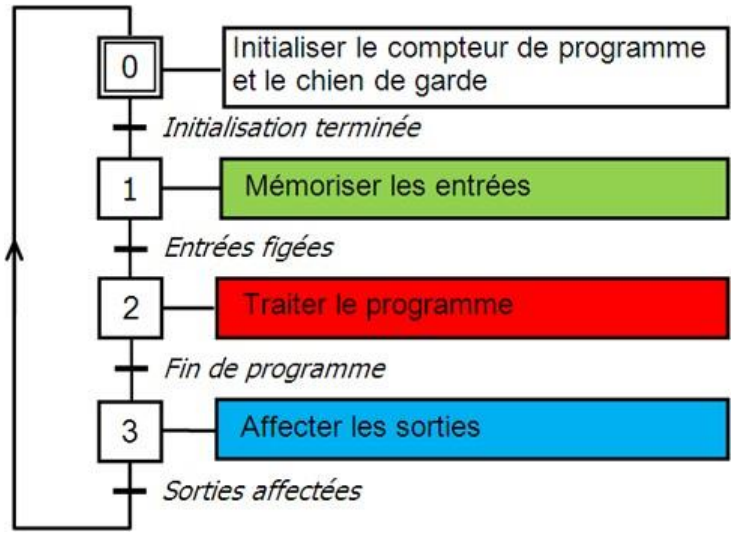


Figure I.8 : Fonctionnement cyclique d'un API

Le processeur exécute alors le programme instruction par instruction en rangeant à chaque fois les résultats en mémoire. En fin de cycle les sorties sont affectées d'un état binaire, par mise en communication avec les mémoires correspondantes. Dans ce cas, le temps de réponse à une variation d'état d'une entrée peut être compris entre un ou deux temps de cycle (durée moyenne d'un temps de cycle est de 5 à 15 ms .

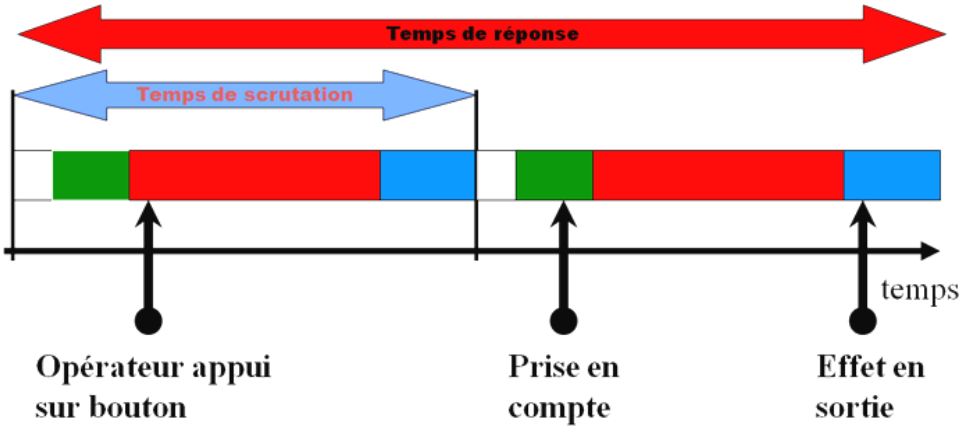


Figure I.9 : Temps de scrutation vs Temps de réponse

Il existe d'autres modes de fonctionnement, moins courants :

- synchrone par rapport aux entrées seulement ;
- asynchrone.

I.10. Description des éléments d'un API :

I.10.1 La mémoire :

Elle est conçue pour recevoir, gérer, stocker des informations issues des différents secteurs du système que sont le terminal de programmation (PC ou console) et le processeur, qui lui gère et exécute le programme. Elle reçoit également des informations en provenance des capteurs.

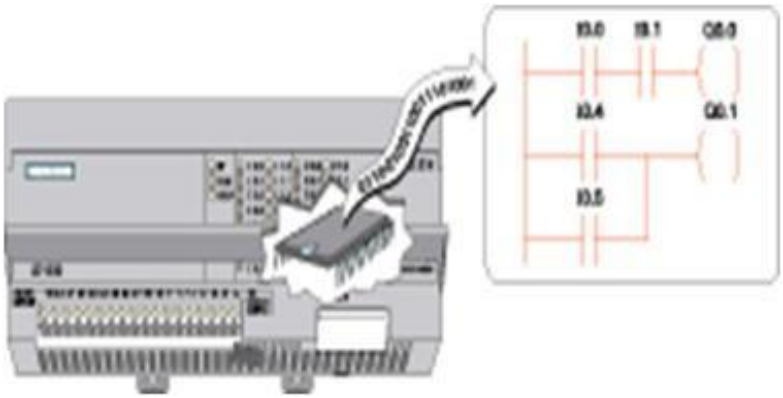


Figure I.10: La mémoire d'un API

Il existe dans les automates deux types de mémoires qui remplissent des fonctions différentes :

- La mémoire Langage où est stocké le langage de programmation. Elle est en général figée, c'est à dire en lecture seulement. (ROM : mémoire morte).

- La mémoire Travail utilisable en lecture-écriture pendant le fonctionnement c'est la RAM (mémoire vive). Elle s'efface automatiquement à l'arrêt de l'automate (nécessite une batterie de sauvegarde).Répartition des zones mémoires :

- Table image des entrées
- Table image des sorties
- Mémoire des bits internes
- Mémoire programme d'application

I.10.2 Le processeur :

Son rôle consiste d'une part à organiser les différentes relations entre la zone mémoire et les interfaces d'entrées et de sorties et d'autre part à exécuter les instructions du programme.

I.10.3 Les interfaces et les cartes d'Entrées / Sorties:

L'interface d'entrée comporte des adresses d'entrée. Chaque capteur est relié à une de ces adresses. L'interface de sortie comporte de la même façon des adresses de sortie. Chaque pré actionneur est relié à une de ces adresses. Le nombre de ces entrées est sorties varie suivant le type d'automate. Les cartes d'E/S ont une modularité de 8, 16 ou 32 voies. Les tensions disponibles sont normalisées (24, 48, 110 ou 230V continu ou alternatif ...). [5]

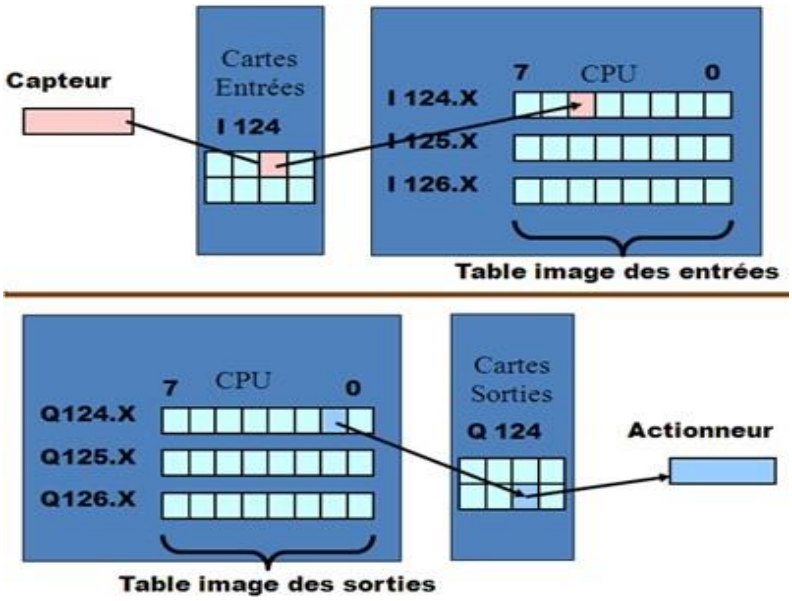


Figure I.11 : Les interfaces d'entrées/sorties

a) Cartes d'entrées :

Elles sont destinées à recevoir l'information en provenance des capteurs et adapter le signal en le mettant en forme, en éliminant les parasites et en isolant électriquement l'unité de commande de la partie opérative.

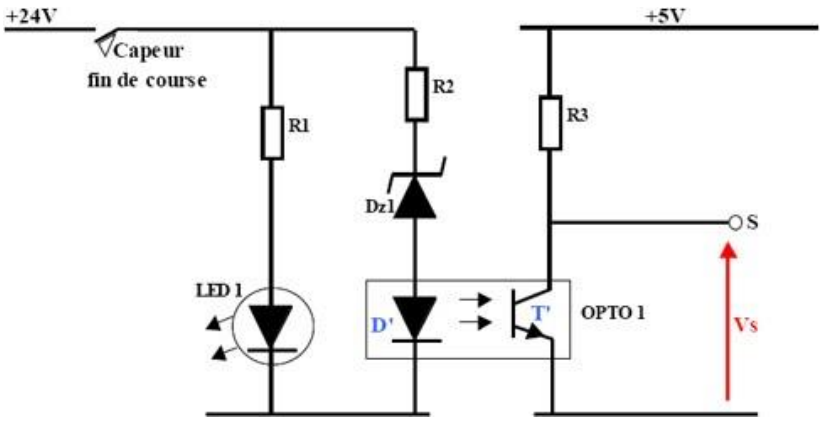


Figure I.12: Exemple d'une carte d'entrées typique d'un API

b) Cartes de sorties:

Elles sont destinées à commander les pré-actionneurs et éléments des signalisations du système et adapter les niveaux de tensions de l'unité de commande à celle de la partie opérative du système en garantissant une isolation galvanique entre ces dernières

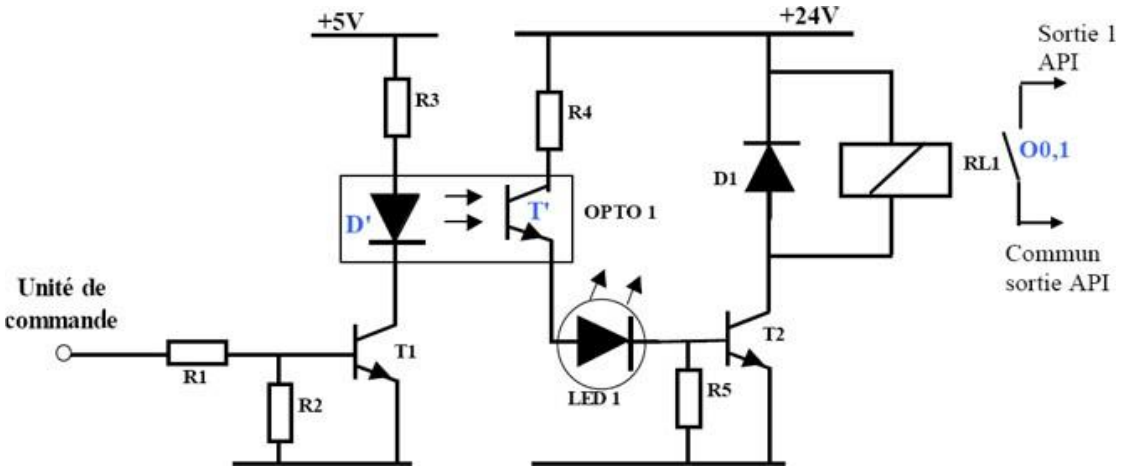


Figure I.13 : Exemple d'une carte de sortie typique d'un API

-Quelque exemple de cartes E/S:

- Cartes de comptage rapide : elles permettent d'acquérir des informations de fréquences élevées incompatibles avec le temps de traitement de l'automate. (signal issu d'un codeur de position)
- Cartes de commande d'axe : Elles permettent d'assurer le positionnement avec précision d'élément mécanique selon un ou plusieurs axes. La carte permet par exemple de piloter un servomoteur et de recevoir les informations de positionnement par un codeur. L'asservissement de position pouvant être réalisé en boucle fermée.
- Cartes d'entrées/sorties analogiques : Elles permettent de réaliser l'acquisition d'un signal analogique et sa conversion numérique (CAN) indispensable pour assurer un traitement par le microprocesseur. La fonction inverse (sortie analogique) est également réalisée. Les grandeurs analogiques sont normalisées : 0-10V ou 4-20mA.
- Cartes de régulation PID.
- Cartes de pesage.
- Cartes de communication (RS485, Ethernet ...).
- Cartes d'entrées / sorties déportées.

I.10.4 L'alimentation électrique :

Tous les automates actuels sont équipés d'une alimentation 240V 50/60Hz, 24V DC. Les entrées sont en 24V DC et une mise à la terre doit également être prévue. [6]

I.11. Jeu d'instructions :

Le processeur peut exécuter un certain nombre d'opérations logiques; l'ensemble des instructions booléennes des instructions complémentaires de gestion de programme (saut, mémorisation, adressage ...) constitue un jeu d'instructions. Chaque automate possède son propre jeu d'instructions. Mais par contre, les constructeurs proposent tous une interface logicielle de programmation répondant à la norme CEI1131-3. Cette norme définit cinq langages de programmation utilisables, qui sont :

- Les langages graphiques :
  - **LD** : **L**adder **D**iagram (Diagrammes échelle)
  - **FBD** : **F**unction **B**lock **D**iagram (Logigrammes)
  - **SFC** : **S**equential **F**unction **C**hart (Grafcet)
- Les langages textuels *:*
  - **IL** : **I**nstruction **L**ist (Liste d'instructions).
  - **ST** : **S**tructured **T**ext (Texte structuré).

Le langage à relais (Ladder Diagram) est basé sur un symbolisme très proche de celui utilisé pour les schémas de câblage classiques. Les symboles les plus utilisés sont donnés au tableau suivant :

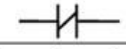
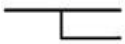
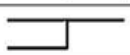
Fonction	Symbole	
	Européen	Américain
Contact ouvert au ropos	---o o---	
Contact fermé au ropos	---o o---	
Début de branchement		
Fin de branchement		
Affectation	---( )---	---( )

Figure I.14 : Symboles usuels en langages LD

I.12. Sécurité :

Les systèmes automatisés sont, par nature, source de nombreux dangers (tensions utilisées, déplacements mécaniques, jets de matière sous pression ...).

Placé au cœur du système automatisé, l'automate se doit d'être un élément fiable car un dysfonctionnement de celui-ci pourrait avoir de graves répercussions sur la sécurité des personnes, de plus les coûts de réparation et un arrêt de la production peuvent avoir de lourdes conséquences sur le plan financier.

Aussi, l'automate fait l'objet de nombreuses dispositions pour assurer la sécurité :

- Contraintes extérieures : l'automate est conçu pour supporter les différentes contraintes du monde industriel et à fait l'objet de nombreux tests normalisés.

- Coupures d'alimentation : l'automate est conçu pour supporter les coupures d'alimentation et permet, par programme, d'assurer un fonctionnement correct lors de la réalimentation (reprises à froid ou à chaud).
- Mode RUN/STOP : Seul un technicien peut mettre en marche ou arrêter un automate et la remise en marche se fait par une procédure d'initialisation (programmée)
- Contrôles cycliques :
    - Procédures d'autocontrôle des mémoires, de l'horloge, de la batterie, de la tension d'alimentation et des entrées / sorties.
    - Vérification du temps de scrutation à chaque cycle appelée *Watchdog (chien de garde)*, et enclenchement d'une procédure d'alarme en cas de dépassement de celui-ci (réglé par l'utilisateur).
    - Visualisation : Les automates offrent un écran de visualisation où l'on peut voir l'évolution des entrées / sorties.

Remarque : Les normes interdisent la gestion des arrêts d'urgence par l'automate ; celle-ci doit être réalisée en technologie câblée. [6]

I.13 Réseaux d'automates

I.13.1- Principe

Avec le développement des systèmes automatisés et de l'électronique, la recherche de la baisse des coûts et la nécessité actuelle de pouvoir gérer au mieux la production et a partir du moment où tous les équipements sont de type informatique, il devient intéressant de les interconnecter à un mini-ordinateur ou à un automate de supervision [7]

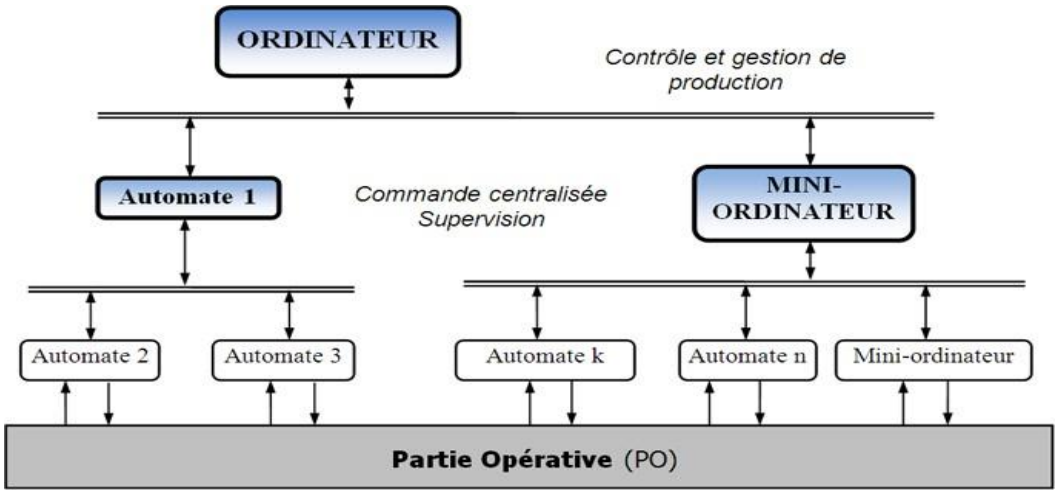


Figure I 15: Exemple d'une structure de contrôle et gestion de production

L'interconnexion entre deux automates peut être réalisée très simplement en reliant une ou plusieurs sorties d'un automate à des entrées de l'autre et vice-versa

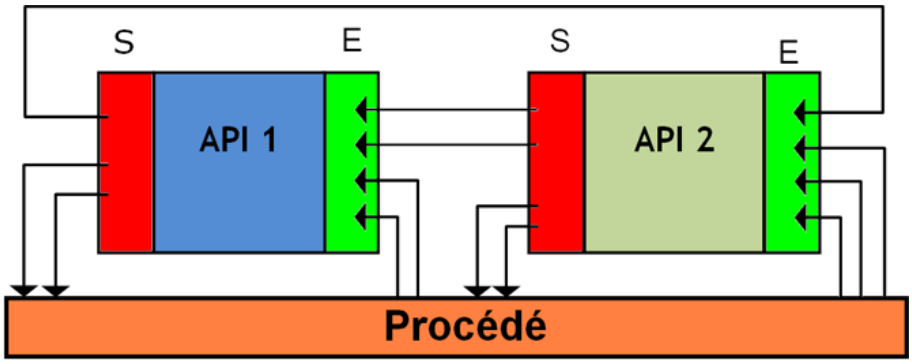


Figure I.16: Interconnexion simple (Entrées/Sorties) entre deux automates (API)

Cette méthode ne permet pas de transférer directement des variables internes d'un automate sur l'autre, de sorte que celles-ci doivent être converties par programme en variables de sortie avant leur transfert. Elle devient coûteuse en nombre d'entrées/sorties mobilisé pour cet usage et lourde du point de vue du câblage, lorsque le nombre de variables qui doivent être échangées devient important. [7]

I.14 Différents types des API

Les automates peuvent être de types compacts ou modulaire.

I.14.1 Type compact

On distinguera les modules de programmation (LOGO de Siemens, ZELIO de Schneider, MILLENIUM de Crouz…) des micros automates.

Il intègre le processeur, l'alimentation, les entrées et les sorties. Selon les modèles et les fabricants, il pourra réaliser certaines fonctions supplémentaires (comptage rapide, ajout d'entrées/sorties analogiques ...) et recevoir des extensions en nombre limité. Ces automates, de fonctionnement simple, sont généralement destinés à la commande de petits automatismes. [8]

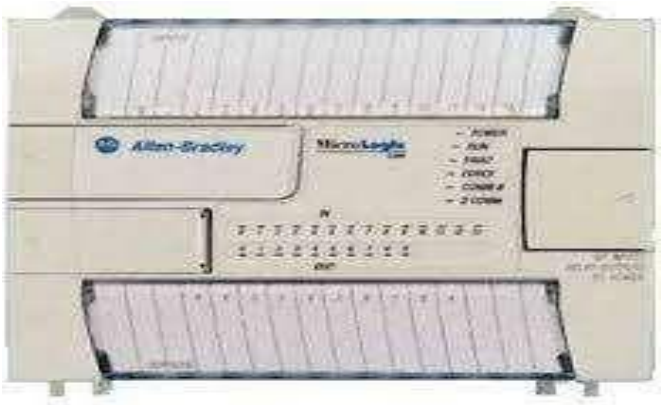


Figure I.17: Automate compact (Allen-Bradley).

I.14.1 Type modulaire

Le processeur, l'alimentation et les interfaces d'entrées/sorties résident dans des unités séparées (modules) et sont fixées sur un ou plusieurs racks contenant le "Fond de panier". Ces automates sont intégrés dans les automatismes complexes où puissants, où la capacité de traitement et flexibilité sont nécessaires. [8]

Figure 1.18 : Automate modulaire (Modicon).

I.14 Les applications de l'automate :

Les automates trouvent leur application en milieu industriel, domestiques. On cite quelques exemples courants :

Exemple n°1: Feux de carrefour

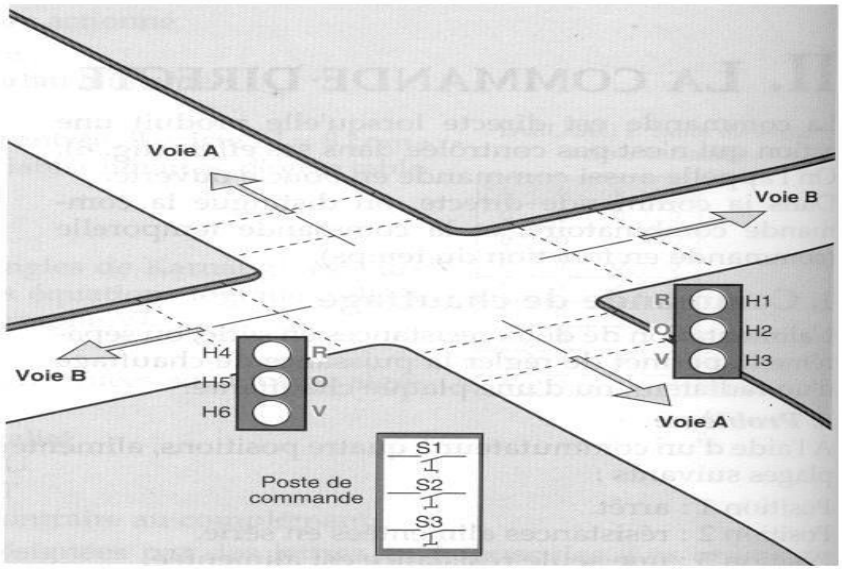


Figure 1.19: Feux de carrefour

Description

On règle la circulation d'un carrefour de deux voies A et B par des feux tricolores (rouges, orange, vert). [9]

I.15 Les consoles

Il existe deux types de consoles : Console d'exploitation : permet le paramétrage et les relevés d'informations (modification des valeurs et visualisation).

Console de programmation, réglage et exploitation. Cette dernière effectue dans laphase de programmation:

- l'écriture.
- la modification.
- l'effacement.
- le transfert d'un programme dans la mémoire de l'automate ou dans une mémoire REPROM

I.16 Domaines d'emploi des automates :

Les API s'adressent à des applications que l'on trouve dans la plupart des secteurs industriels. Ces machines fonctionnent dans les principaux secteurs suivants :

- Métallurgie et sidérurgie.
- Mécanique et automobile.
- Industries chimiques.
- Industries pétrolières.
- Industries agricoles et alimentaires. [10]

Exemple 2 : Portail coulissant.

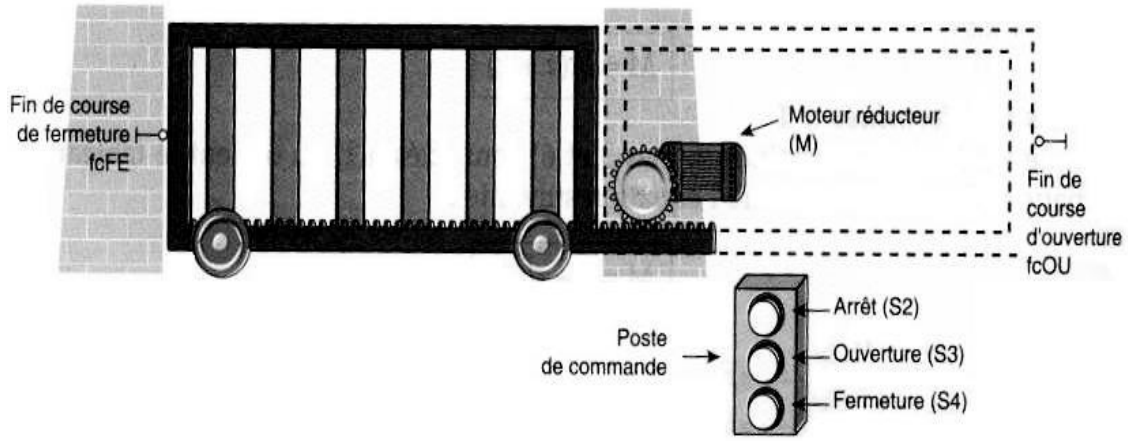


Figure I.20: Portail coulissant.

Soit un portail coulissant à commander :

- Le portail étant fermé, le contact fin de course fcFE est actionné ;
- On appuie sur le bouton-poussoir d'ouverture S3, le moteur actionne le portail et provoqueson ouverture ;

- En fin d'ouverture, le contact fin de course fcOU est actionné, il signale l'ouverture duportail, et il coupe l'alimentation du moteur.
- L'action sur le bouton-poussoir de fermeture provoque l'inversion de sens de marche dumoteur, et la fermeture du portail.
- Le portail libère le contact fcOU, et se déplace jusqu'à actionner le contact fcFE qui provoquel'arrêt du moteur.

I.17 Les étapes à suivre pour raccorder un automate:

Pour raccorder un automate, il est recommandé de suivre :

- Les spécifications du fabricant.
- La technique de raccordement.
- De vérifier si les modules sont dans leurs embases respectives. Vérifier le type, le numéro du modèle et le diagramme de câblage. Vérifier l'emplacement des embases dans le document pour l'assignation des adresses d'E/S.
- De localiser le paquet de fils correspondant à chaque module et le diriger à travers le conduit à l'emplacement du module.

Identifier chacun des fils dans le paquet et s'assurer qu'ils correspondent à cemodule en particulier.

- En commençant avec le premier module, repérer le fil dans le paquet qui se branche à la borne la plus basse. Au point où le fil arrive à la même hauteur que le point de terminaison, plie le fil à angle droit vers la borne.
- De couper le fil pour qu'il dépasse de 6 mm du côté de la vis de la borne. Dégainer l'isolant du fil à approximativement 9 mm.

Insérer le fil sous la plaque de la borne et serrer la vis.

- Si deux modules ou plus utilisent la même source d'alimentation, on peut utiliser du cavalier « jumpers » pour le câblage de la source d'alimentation d'un module à l'autre.
- Si le câble blindé est utilisé, en brancher seulement un bout à la mise à la terre, préférablement au châssis. Ce branchement évitera toutes boucles possibles de retour de masse. L'autre bout doit être coupé et non branché.
- De répéter la procédure de câblage pour chaque fil du paquet jusqu'à ce que le câblage du module soit complété. Après que tous les fils aient été branchés, tirer doucement sur chacun pour s'assurer d'avoir un bon branchement.

De répéter la procédure de câblage jusqu'à ce que tous les modules soient terminés. [11]

Conclusion :

L'automate est un bon produit, facile à programmer, à connecter, adapté aux conditions industrielles. L'expansion considérable de ses possibilités, et celle corrélative de son marché, le prouvent. Il ne faut pas vouloir en faire une solution miracle. Dans tous les cas :

- une bonne analyse du problème à résoudre ;
- le respect des règles d'installation ;
- un léger surdimensionnement pour préserver des marges de modification sont les conditions d'une implantation réussie, dont la durée de vie dépassera largement celles habituelles dans le monde informatique, dont l'API est pour partie issu.

Chapitre II

Les outils utilisés

II.1. Introduction :

La logique Ladder ou langage de programmation Ladder est une méthode pour tracer les schémas en logique électrique. Il s'agit maintenant d'un langage graphique vraiment populaire pour la programmation des automates programmables industriels (API).
Il a été à l'origine inventé pour décrire la logique à relais. Son nom est fondé sur la constatation que les programmes dans cette langue ressemblent à une échelle (Ladder), avec deux "rails" verticaux et, entre eux, une série "d'échelons".
En Allemagne et ailleurs en Europe, le style consiste à placer les rails horizontaux, un en haut de la page et l'autre en bas avec les échelons verticaux dessinés séquentiellement de la gauche vers la droite.

II.2 Historique de LADDER :

Le LADDER, apparu avec les premiers automates MODICON au milieu des années 70, consiste en un ensemble de réseaux de contacts, semblables à des schémas électriques, ce qui permettait à l'époque aux techniciens électriciens de passer plus facilement du système câblé au système programmé.

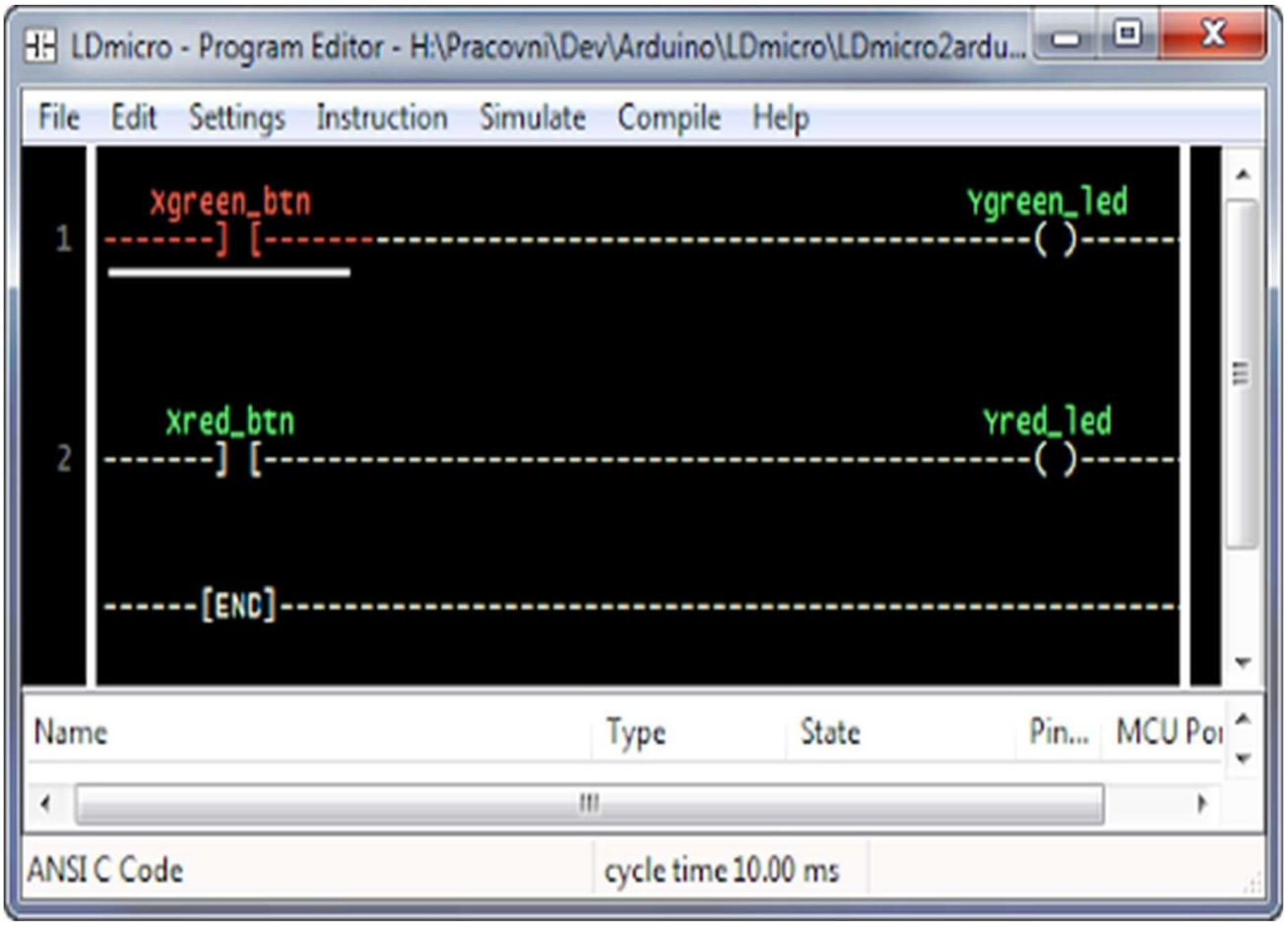


Figure II.1 : Interface Logiciel LDmicro

Ce langage, bien que assez rudimentaire, est aujourd'hui encore très répandu pour la programmation des automates. Il est, en particulier, adapté pour décrire des fonctions combinatoires simples ou pour appeler des blocs fonctions. Il nous servira aussi pour créer des blocs fonctionnels ou pour programmer les réceptivités et actions de nos GRAFCET

(ou SFC).[12]

II.3 Définition de ladder :

Ladder Diagram (LD) ou Langage Ladder ou schéma à contacts est un langage graphique très populaire auprès des automaticiens pour programmer les automates programmables industriels. Il ressemble un peu aux schémas électriques, et est facilement compréhensible Langage Ladder. [13]

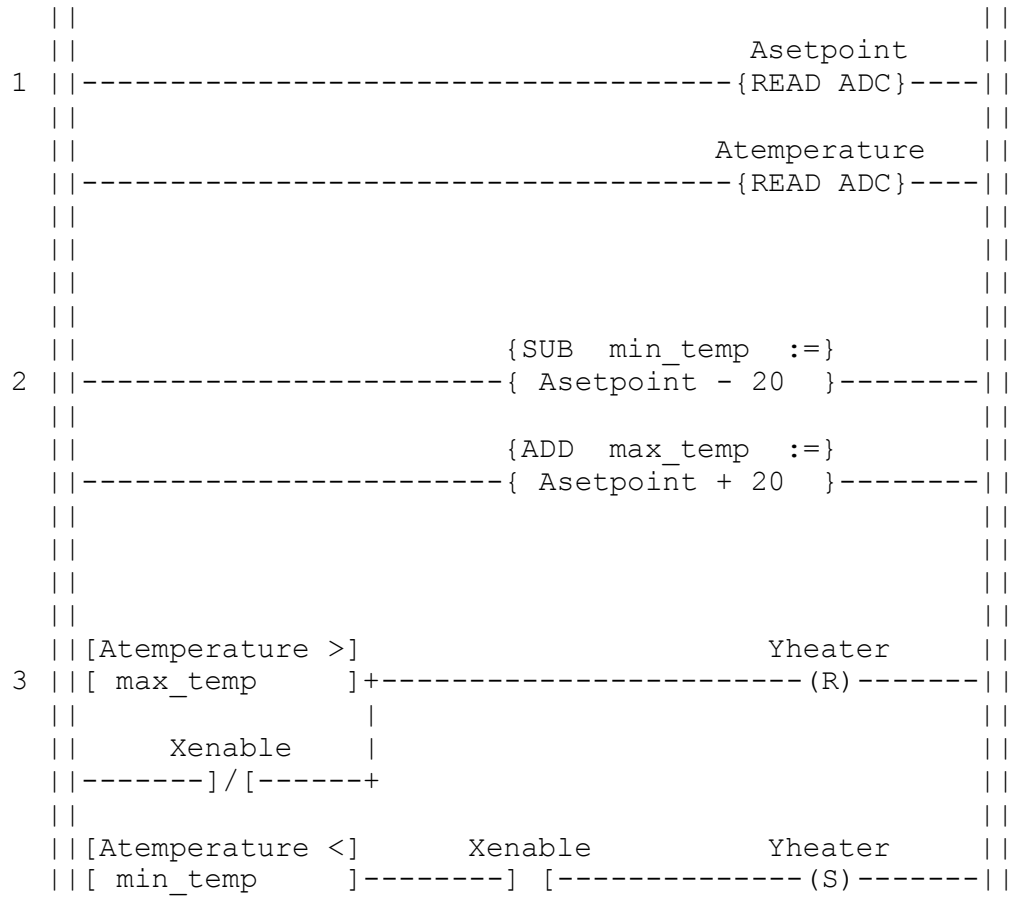


Figure II.2 : schéma de *Ladder*

II.4 LADDER

II.4.1 Origine :

L'idée initiale du Ladder est la représentation de fonction logique sous la forme de schémas électriques. Cette représentation est originellement matérielle : quand l'automate programmable industriel n'existait pas, les fonctions étaient réalisées par des câblages. Par exemple, pour réaliser

un ET logique avec des interrupteurs, il suffit de les mettre en série. Pour réaliser un OU logique, il faut les mettre en parallèle.

Le Ladder a été créé et normalisé dans la norme CEI 61131-3. Il est encore aujourd'hui souvent utilisé dans la programmation des automates programmables industriels, bien qu'ayant tendance à être délaissé en faveur de langages plus évolués, et plus adaptés aux techniques modernes de programmation, tels que le ST par exemple, ou encore le Grafcet, plus adapté à la programmation de séquences [13]

II.4.2 Principe de programmation :

Le LADDER est un langage simple qui reprend les bases du schéma à contact**.** Pour programmer en LADDER, il suffit de transcrire les équations logiques en schéma électrique à l'aide de symboles placés entre deux barres verticales (qui représentent les lignes d'alimentations).

Un programme écrit en *LADDER*, appelé diagramme *LADDER* (en anglais, *Ladder Diagram* = *LD*), se lit dela gauche vers la droite et du haut vers le bas

1. Eléments du langage :

Les contacts

Les contacts(ou interrupteurs) correspondent aux variables d'entrées du système.

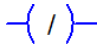
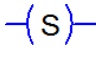
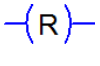
Tableau II.1 : Différent types de contacts

Symbole	Désignation	Fonctionnement
	Contact normalement ouvert (NO)	Le contact est fermé lorsque la variable qui lui est associée est à l'état logique 1. Le contact est ouvert lorsque la variable qui lui est associée est à l'état logique 0.
	Contact normalement fermé (NF)	Le contact est fermé lorsque la variable qui lui est associée est à l'état logique 0. Le contact est ouvert lorsque la variable qui lui est associée est à l'état logique 1.
	Contact à détection de front montant ( )	Le contact est fermé lorsque la variable qui lui est associée passe de 0 à 1 (front montant). Le contact est ouvert lorsque la variable qui lui est associée est à l'état logique 0 ou à l'état logique 1.
	Contact à détection de front descendant ( )	Le contact est fermé lorsque la variable qui lui est associée passe de 1 à 0 (front descendant). Le contact est ouvert lorsque la variable qui lui est associée est à l'état logique 0 ou à l'état logique 1.

b) ***Les bobines :***

Les bobines correspondent aux variables de sortie du système [13]

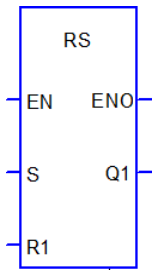
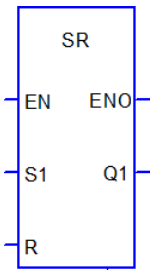
Tableau II.2 : différent type de bobines

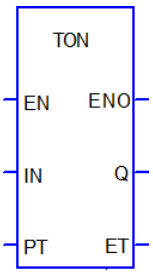
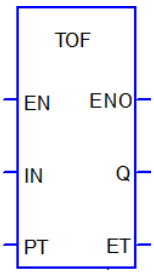
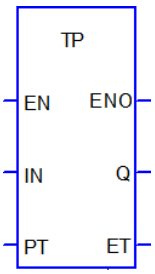
Symbole	Désignation	Fonctionnement
	Bobine	La bobine est activée (excitée) lorsque les contacts auxquels elle est reliée sont fermés. La variable qui lui est associé est alors à l'état logique 1. Dans le cas contraire, la bobine est désactivée et la variable qui lui est associée est à l'état logique 0.
	Bobine « négative »	La bobine est activée (excitée) lorsque les contacts auxquels elle est reliée sont ouverts. La variable qui lui est associé est alors à l'état logique 1. Dans le cas contraire, la bobine est désactivée et la variable qui lui est associée est à l'état logique 0.
	Bobine d'enclenchement (*Set* = mise à 1)	La bobine est activée (excitée) dès que les contacts auxquels elle est reliée sont fermés. La variable qui lui est associé est alors à l'état logique 1. La bobine reste activée même si ensuite les contacts ne sont plus fermés.
	Bobine de déclenchement (*Reset* = mise à 0)	La bobine est désactivée dès que les contacts auxquels elle est reliée sont fermés. La variable qui lui est associé est alors à l'état logique 0. La bobine reste désactivée même si ensuite les contacts ne sont plus fermés.

Les « blocs fonctions » :

Le langage *LADDER* permet l'utilisation de nombreuses fonctions logiques, telles que les bascules *RS*, les opérateurs à retard, l'opérateur monostable, le compteur et le décompteur, les comparateurs, etc.Ces fonctions logiques complexes sont appelées « blocs fonctions ». [13]

Tableau II.3 : Différent types de blocs

Bascules *RS*	
Bascule *RS* à priorité à l'effacement (Bascule *RS*)	Bascule *RS* à priorité à l'inscription (Bascule *SR*)
	

Opérateurs à retard et opérateur monostable		
Retard à l'enclenchement	Retard au déclenchement	Monostable
		
Q = PT / IN	*Q = IN / PT*	*Q = IN / PT*

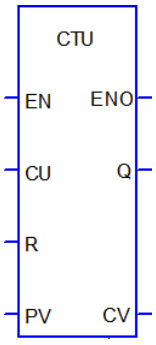
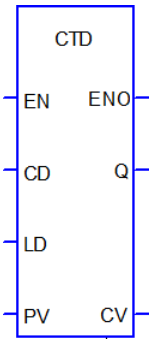
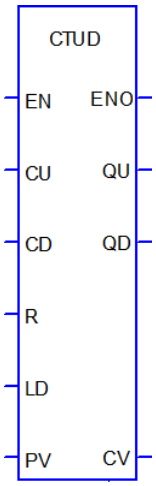
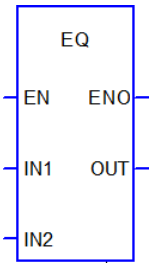
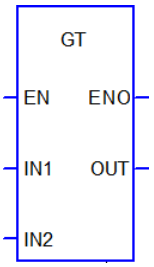
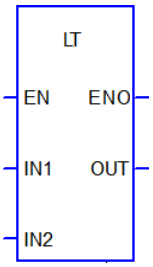
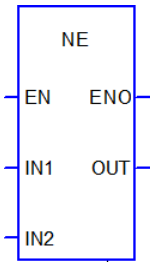
Compteur et décompteur		
Compteur (*CounT Up*)	Décompteur (*CounT Down*)	Compteur/décompteur (*CounT Up Down*)
		

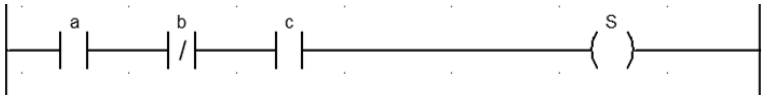
Tableau II 4 : Différent types de blocs (suite)

Compateurs			
Egalité (*EQual to*)	Supériorité (*Greater Than*)	Infériorité (*Less Than*)	Différence (*Not Equal to*)
			
Si *IN1* = *IN2* alors *OUT* = 1	Si *IN1* > *IN2* alors *OUT* = 1	Si *IN1* < *IN2* alors *OUT* = 1	Si *IN1* ¹ *IN2* alors *OUT* = 1

II.4.3 Associations de contacts et de bobines

Contacts en série :

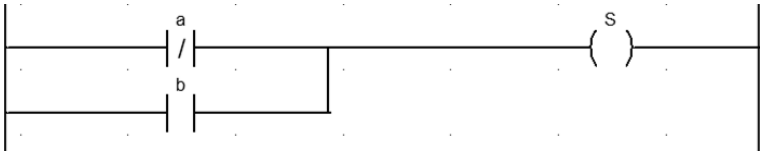
L'association de contacts en série permet de réaliser la fonction logique ET.

Exemple : 

Equation logique : $S = a.\bar{b}.c$

Contacts en parallèle (ou en dérivation) :

L'association de contacts en parallèle permet de réaliser la fonction logique OU.

Exemple : 

Equation logique : $S = \bar{a} + b$ [13]

II.5 Grafcet

II.5.1 Introduction

Un GRAFCET (GRAphe Fonctionnel de Commande Etape-Transition) est un mode de représentation et d'analyse d'un automatisme. C'est un **o**util graphique de description du comportement de la *partie commande*. Il décrit les interactions informationnelles à travers la frontière d'isolement : partie de commande, partie opérative d'un système isolé.

Ce mode de représentation est indépendant de la technologie utilisée dans l'automatisme, et traduit d'une façon cohérente le cahier de charge de l'automatisme. [14]

II.5.2. Historique du grafcet

En 1975, un groupe d'universitaires et industriels de la section "Systèmes Logiques" de l'AFCET (Association Française de Cybernétique Economique et Technique) se sont fixés l'objectif de définir un formalisme adapté à la représentation des évolutions séquentielles d'un système.

Au début, le travail consista à dresser un état de l'art des différentes approches de modélisation du comportement de tels automatismes. L'analyse des avantages et inconvénients de ces outils a mené en 1977 à la définition du GRAFCET, ainsi nommé pour, à la fois marquer l'origine de ce nouvel outil de modélisation « AFCET » et son identité Graphe Fonctionnel de Commande Etapes–Transitions.

Les résultats de ces travaux furent l'objet d'une publication officielle dans la revue ”Automatique et Informatique Industrielle” en décembre 1977, date que la communauté considère aujourd'hui comme correspondant à la date de naissance effective du GRAFCET.

II.5.3. Définition :

Le GRAFCET (Graphe Fonctionnel de Commande Étapes et Transitions) est un diagramme fonctionnel. Est aussi un outil graphique de définition pour l'automatisme séquentiel, en tout ou rien. Mais il est également utilisé dans beaucoup de cas combinatoires, dans le cas où il y a une séquence à respecter mais où l'état des capteurs suffirait pour résoudre le problème en combinatoire.

Le GRAFCET est une représentation graphique alternée des étapes et des transitions. Une seule transition doit séparer deux étapes. Le diagramme ci- dessous représente un GRAFCET. [14]

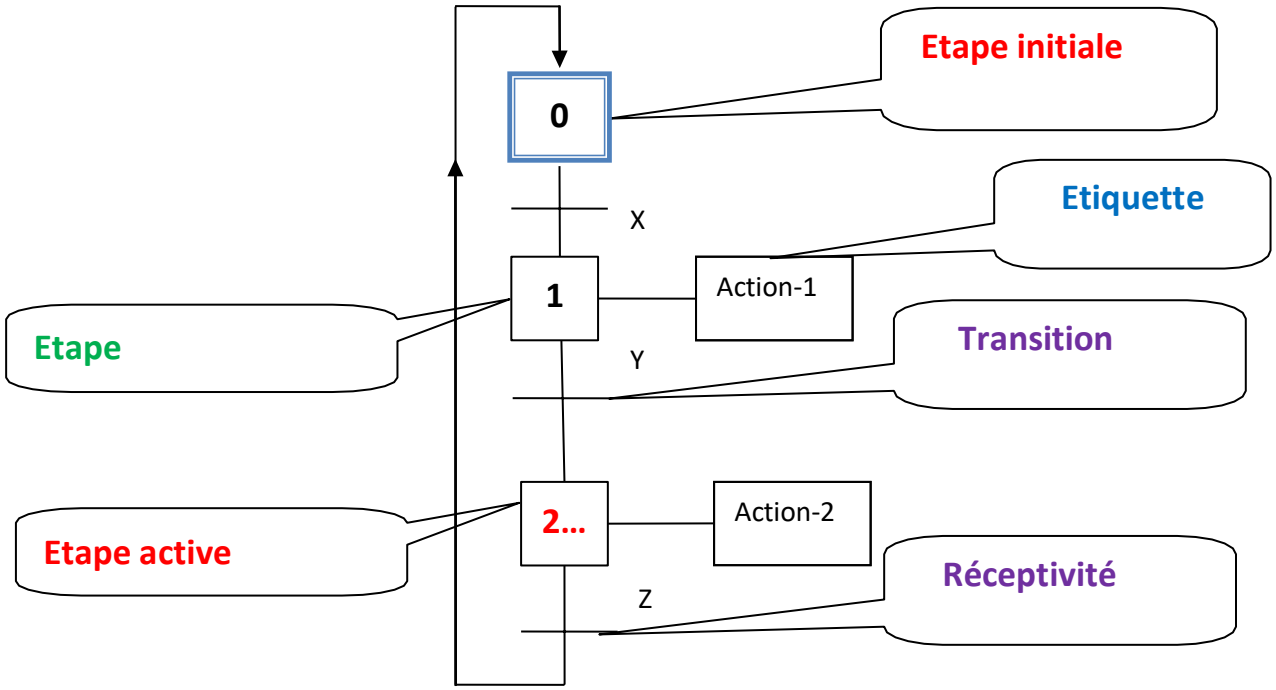


Figure II . 3 **:** Représentation du GRAFCET séquence uniqu

II.5.4 Les caractéristiques du grafcet :

- Simple à utiliser;
- Accepté par tous comme standard à utiliser comme le ladder;
- Intelligible à la fois par les concepteurs et les exploitants;
- Fournissant potentiellement des facilités de passage au logicielle de l'automatisme ainsi spécifié.

II.5.5 Description du GRAFCET :

La description du comportement attendu d'un automatisme peut se représenter par un GRAFCET d'un certain « niveau ». La caractérisation du «niveau» du GRAFCET nécessite de prendre en compte trois dimensions :

a) ***Le point de vue ,*** *caractérisant le point de vue selon lequel un observateur s'implique dans le fonctionnement du système pour en donner une description. On distingue trois points de vue :*
   a) Un point de vue système,
   b) Un point de vue Partie Opérative,
   c) Un point de vue Partie Commande.

b) **La spécifications**, caractérisant la nature des spécifications techniques auxquelles doit satisfaire la Partie Commande. On distingue trois groupes de spécifications :
   a) Spécifications fonctionnelles,
   b) Spécifications technologiques,
   c) Spécifications opérationnelles.

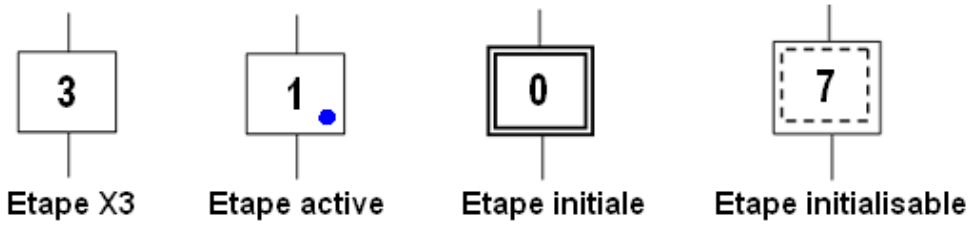
c) **La finesse**, caractérisant le niveau de détail dans la description du fonctionnement, d'un niveau global (ou macro-représentation) jusqu'au niveau de détail complet où toutes les actions et informations élémentaires sont prises en compte. [15]

II.5.6 Les concepts de base du GRAFCET

II.5.6.1 Etape :

Une étape symbolise un état ou une partie de l'état du système automatisé. L'étape possède deux états possibles : active représentée par un jeton dans l'étape ou inactive. L'étape i, représentée par un carré repéré numériquement, possède ainsi une variable d'état, appelée variable d'étape Xi. Cette variable est une variable booléenne valant 1 si l'étape est active, 0 sinon.

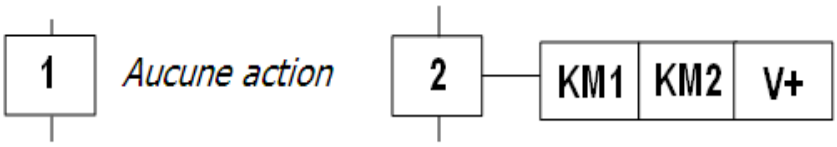
La situation initiale d'un système automatisé est indiquée par une étape dite étape initiale et représentée par un carré double.



Remarque : Dans un Grafcet il doit y avoir au moins une étape initiale.

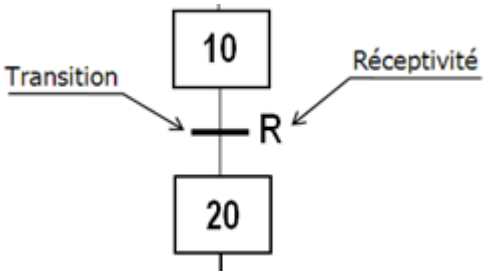
II.5.6.2 Actions associées aux étapes

A chaque étape est associée une **action** ou plusieurs, c'est à dire un ordre vers la partie opérative ou vers d'autres Grafcet. Mais on peut rencontrer aussi une même action associée à plusieurs étapes ou une étape **vide** (*sans action*).



II.5.6.3 Transition

Une transition indique la possibilité d'évolution qui existe entre deux étapes et donc la succession de deux activités dans la partie opérative. Lors de son franchissement, elle va permettre l'évolution du système. A chaque transition est associée une condition logique appelée réceptivité qui exprime la condition nécessaire pour passer d'une étape à une autre.



La réceptivité qui est une information d'entrée qui est fournie par :

1. l'opérateur : pupitre de commande,
2. la partie opérative : états des capteurs,
3. du temps, d'un comptage ou toute opération logique, arithmétique...
4. du Grafcets : d'autres Grafcets pour la liaison entre Grafcets ou de l'état courant des étapes du Grafcet (les Xi),
5. d'autres systèmes : dialoguent entre systèmes,

Remarque: Si la réceptivité n'est pas précisée, alors cela signifie qu'elle est toujours vraie. (=1)

II.5.6.4 Liaisons orientées

Elles sont de simples traits verticaux qui relient les étapes aux transitions et les transitions aux étapes Elles sont normalement orientées de haut vers le bas. Une flèche est nécessaire dans le cas contraire. [15]

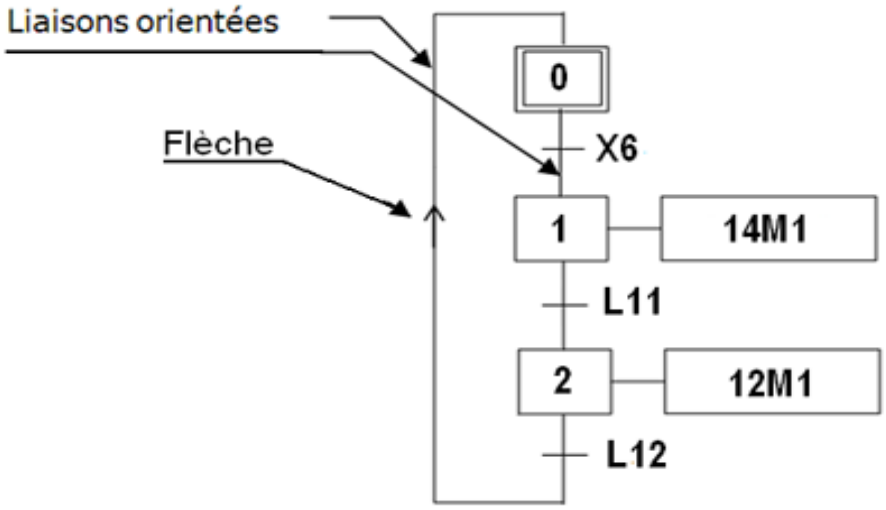


Figure II.4 : les liaisons orientées

II.5.7 Classification des actions associées aux étapes

L'action associée à l'étape peut être de 3 types : continue, conditionnelle ou mémorisée. Les actions peuvent être classées en fonction de leur durée par rapport à celle de l'étape.

II.5.7.1 Actions continues :

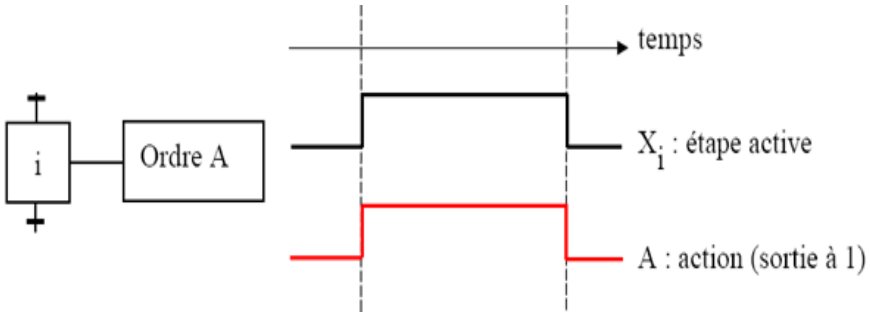


Figure II . 5 : les actions continues

L'ordre est émis, de façon continue, tant que l'étape, à laquelle il est associé, est active.

II.5.7.2 Actions conditionnelles:

Une action conditionnelle n'est exécutée que si l'étape associée est active et si la condition associée est vraie. Elles peuvent être décomposées en 3 cas particuliers:

II.5.7.3 Action conditionnelle simple : Type C

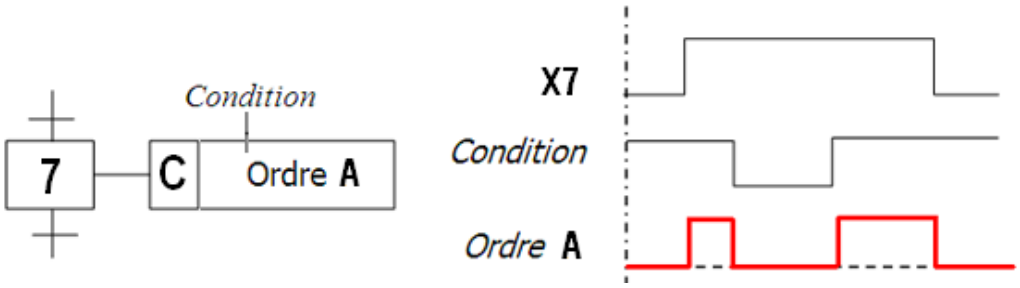


Figure II . 6 : le s action conditionnelle simple type c

II.5.7.4 Action retardée : Type D (delay)

Le temps intervient dans cet ordre conditionnel comme condition logique. L'indication du temps s'effectue par la notation générale " t / xi / q " dans laquelle "xi" indique l'étape prise comme origine du temps et "q" est la durée du retard. [15]

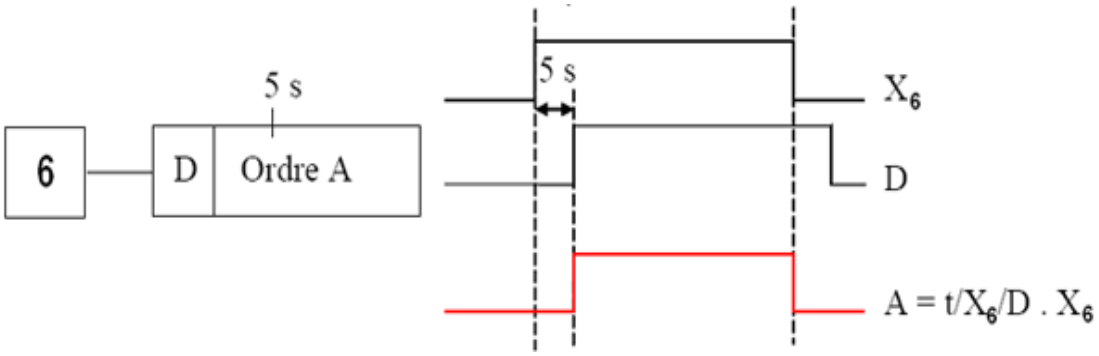


Figure II .7 : Action retardé type D

II.5.7.5. Action maintenue sur plusieurs étapes:

Afin de maintenir la continuité d'une action sur plusieurs étapes, il est possible de répéter l'ordre continu relatif à cette action, dans toutes les étapes concernées ou d'utiliser une description sous forme de séquences simultanées (*Les séquences simultanées seront traitées ultérieurement*).

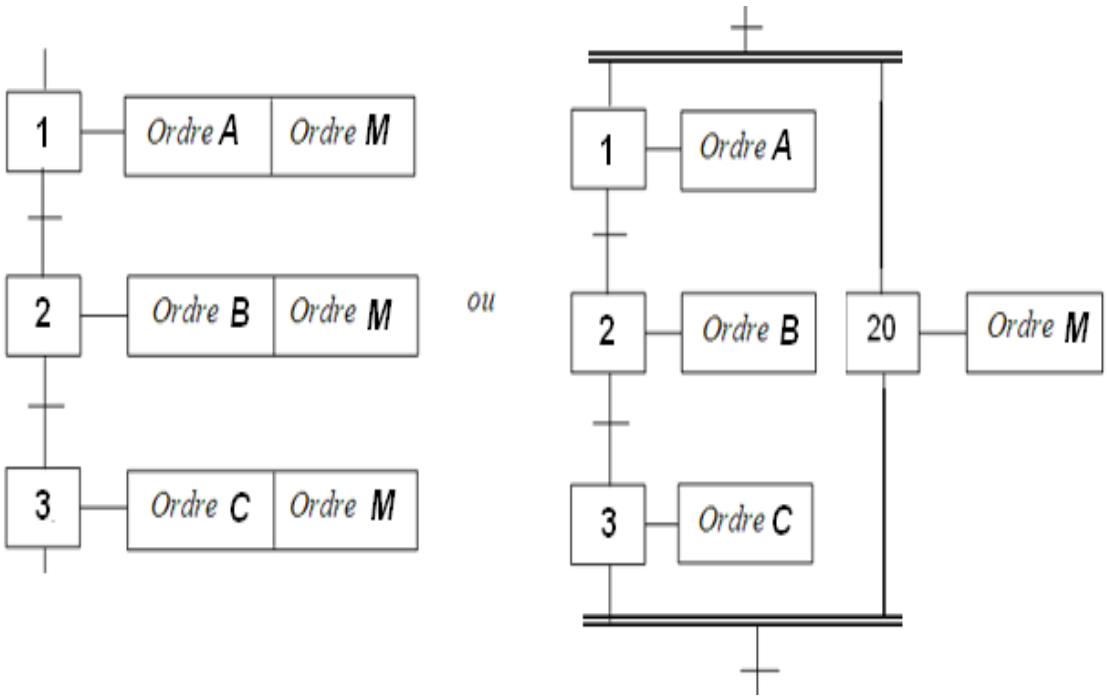


Figure II . 8 : *Action maintenue sur plusieurs étapes*

II.5.7.6 Action mémorisée :

Le maintien d'un ordre, sur la durée d'activation de plusieurs étapes consécutives, peut également être obtenu par la mémorisation de l'action, obtenue par l'utilisation d'une fonction auxiliaire appelée fonction mémoire.

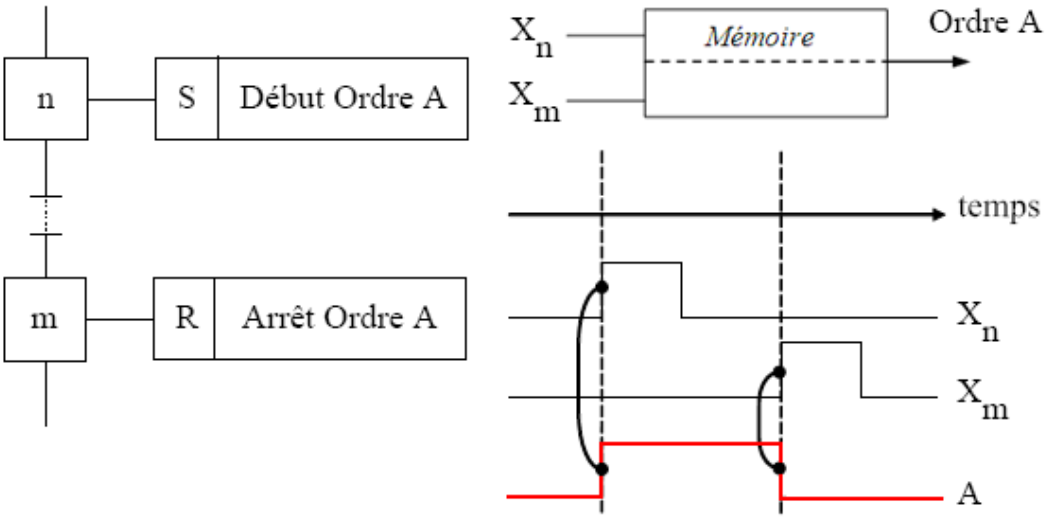


Figure II.9 : Action mémorisée

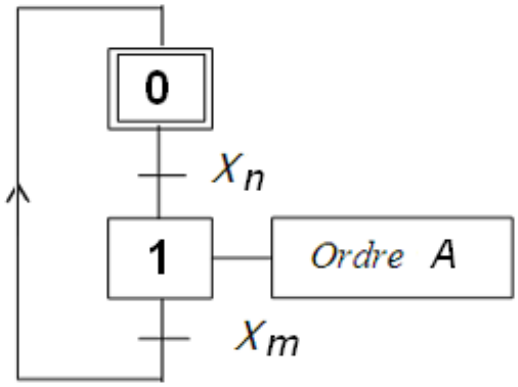


Figure. II 10 : GRAFCET Action mémorisée

II.5.8. Règles d'évolution d'un GRAFCET :

Règle N°1 : Condition initiale

A l'instant initial, seules les étapes initiales sont actives.

Règle N°2 : Franchissement d'une transition.

Pour qu'une transition soit validée, il faut que toutes ses étapes amont (immédiatement précédentes reliées à cette transition) soient actives. Le franchissement d'une transition se produit lorsque la transition est validée, ET seulement si la réceptivité associée est vraie.

Règle N°3 : Evolution des étapes actives

Le franchissement d'une transition entraîne obligatoirement l'activation de toutes les étapes immédiatement suivantes et la désactivation de toutes les étapes immédiatement précédentes. [8]

Règle N°4 : Franchissement simultané

Toutes les transitions simultanément franchissables à un instant donné sont simultanément franchies.

Règle N°5 : Conflit d'activation

Si une étape doit être simultanément désactivée par le franchissement d'une transition aval, et activée par le franchissement d'une transition amont, alors elle reste active. On évite ainsi des commandes transitoires (néfastes à la partie opérative).

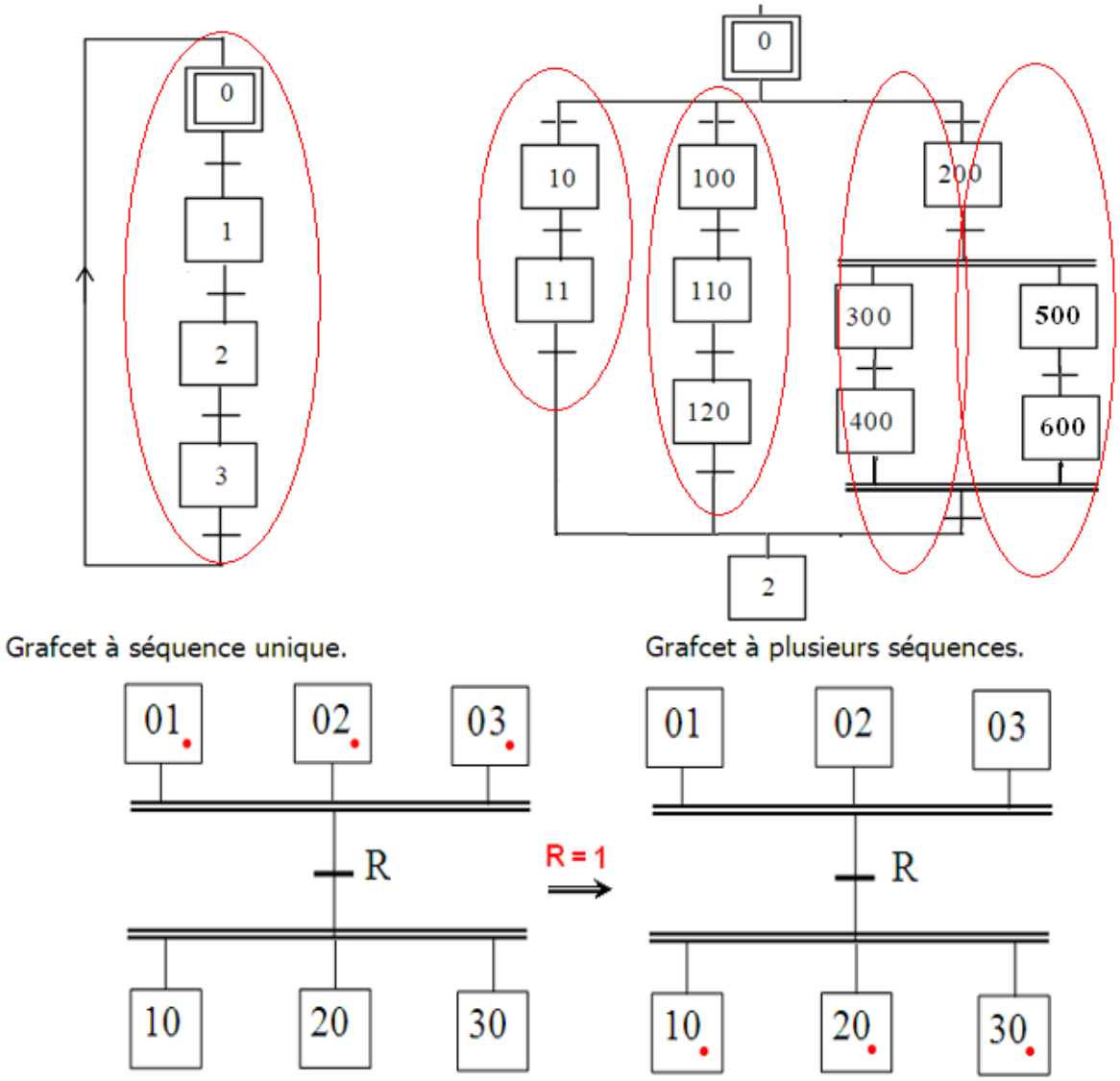


Figure II.11 : Règles d'évolution d'un GRAFCET

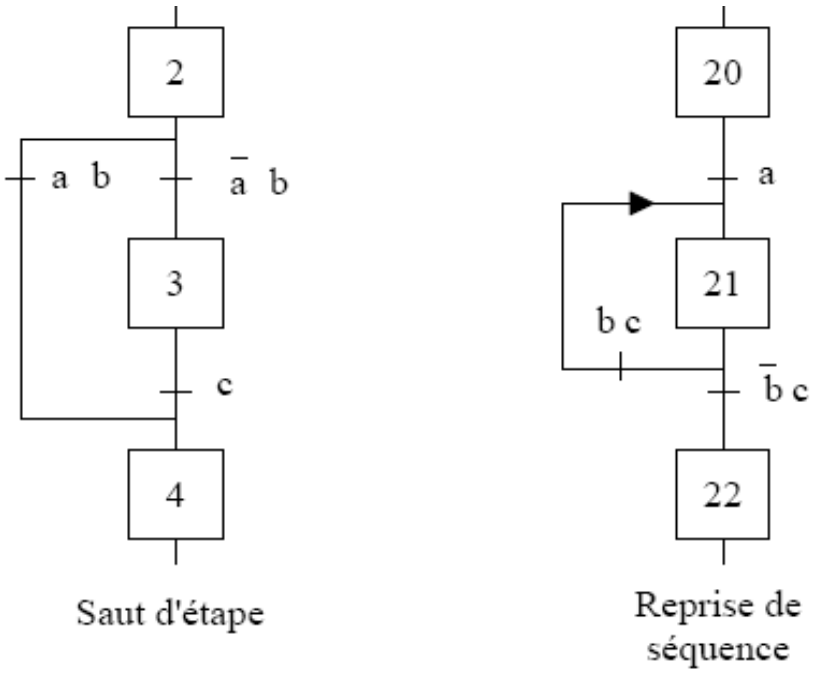
II.5.8.1 Les structures de base

*a) . **Notion de Séquence :***

Une séquence, dans un Grafcet, est une suite d'étapes à exécuter l'une après l'autre. Autrement dit chaque étape ne possède qu'une seule transition AVAL et une seule transition AMONT [8].

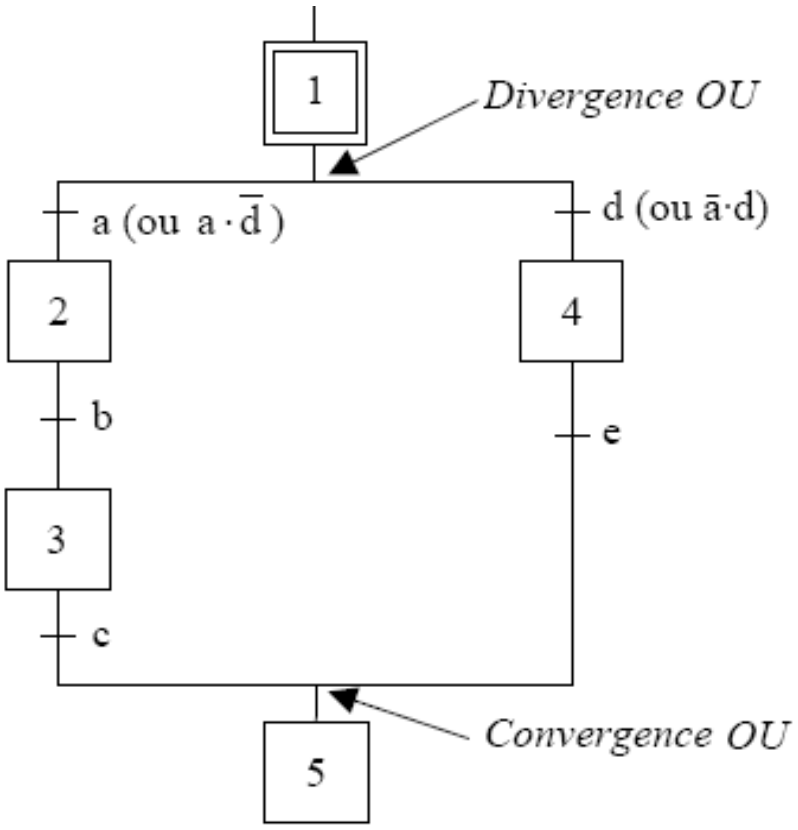
. Saut d'étapes et reprise de séquence

Le saut d'étapes permet de sauter une ou plusieurs étapes lorsque les actions associées sont inutiles à réaliser, La reprise de séquence (ou boucle) permet de reprendre, une ou plusieurs fois, une séquence tant qu'une condition n'est pas obtenue [15].



Aiguillage entre deux ou plusieurs séquences (Divergence en OU)

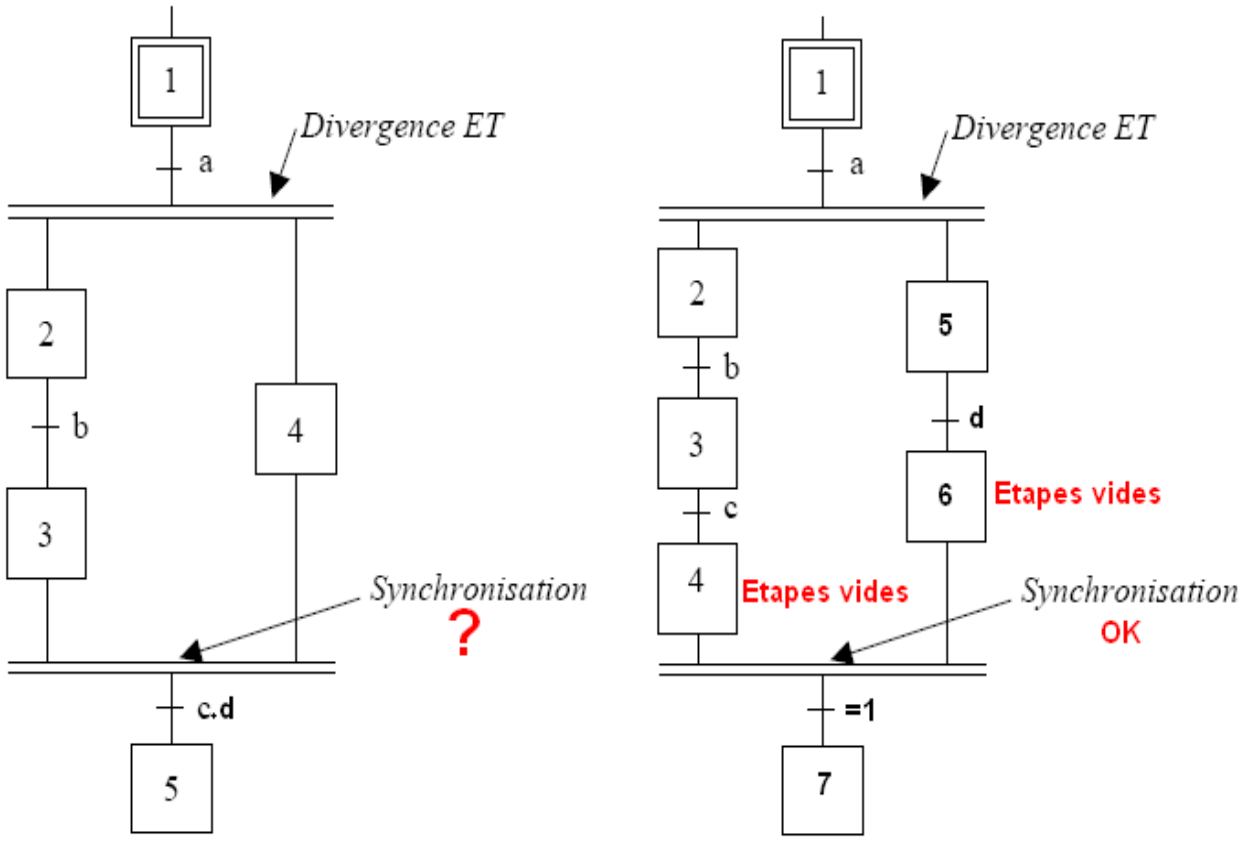
On dit qu'il y a Aiguillage ou divergence en OU lorsque le Grafcet se décompose en deux ou plusieurs séquences selon un choix conditionnel. Comme la divergence en OU on rencontre aussi la convergence en OU. On dit qu'il y a convergence en OU, lorsque deux ou plusieurs séquences du Grafcet converge vers une seule séquence. [15]



Si les deux conditions a et d sont à 1 simultanément, les étapes 2 et 4 vont devenir actives simultanément, situation non voulue par le concepteur. Donc elle doivent être des conditions exclusives

Parallélisme entre deux ou plusieurs séquences (ou séquences simultanées ou divergence-convergence en ET) :

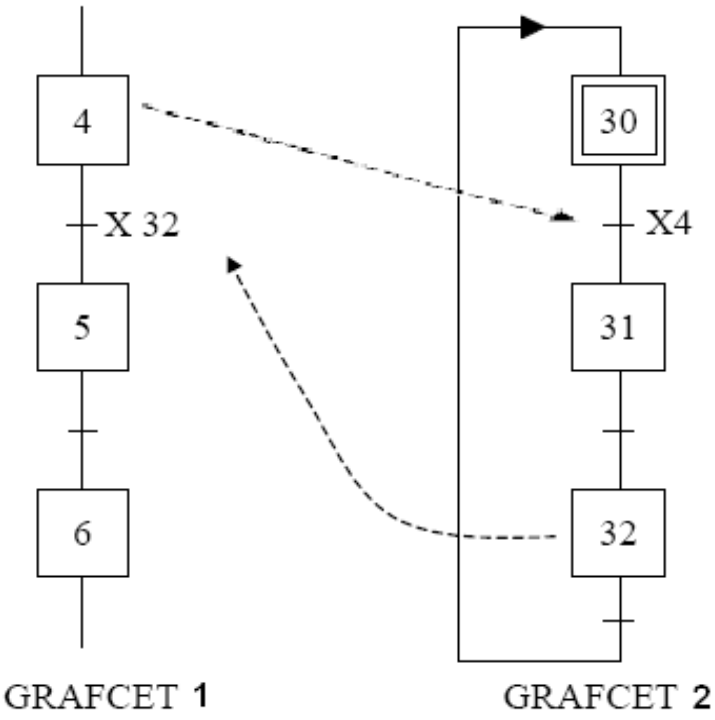
Au contraire de l'aiguillage où ne peut se dérouler qu'une seule activité à la fois, On dit qu'on se trouve en présence d'un parallélisme structurel, si plusieurs activités indépendantes pouvant se dérouler en parallèle. Le début d'une divergence en ET et la fin d'une convergence en ET d'un parallélisme structurel sont représentés par deux traits parallèles. [15]



La synchronisation permet d'attendre la fin de plusieurs activités se déroulant en parallèle, pour continuer par une seule.

. Liaison entre Grafcets :

Une étape dans un Grafcet peut servir comme réceptivité à une autre étape d'un autre grafcet. Cette méthode est utilisée aussi pour synchroniser deux Grafcet c'est à dire rendre l'évolution de l'un dépendent de l'évolution de l'autre. [8]

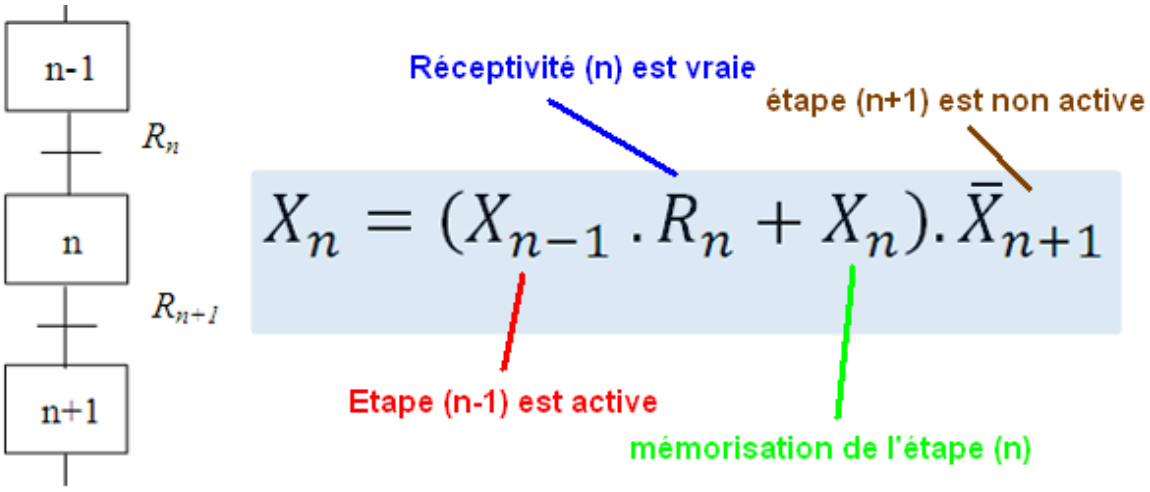


II.5.9. Mise en équation d'un grafcet :

II.5.9.1 Règle générale :

Pour qu'une étape soit activée il faut que :

- L'étape immédiatement précédente soit active ;
- La réceptivité immédiatement précédente soit vraie ;
- L'étape immédiatement suivante soit non active ;
- Après activation l'étape mémorise son état. [15]



Equation d'activation de l'étape de rang n

II.6. Logiciel de programmation l'IDE d'Arduino

Arduino IDE (Integrated Development Environment). Le logiciel est gratuit et open source dont la simplicité d'utilisation est remarquable. Ce logiciel va nous permettre de programmer les implémentions de la commande directement sur la carte, et son interface se compose d'un ensemble d'outils développés qui rend les cartes Arduino facilement programmables. [16]

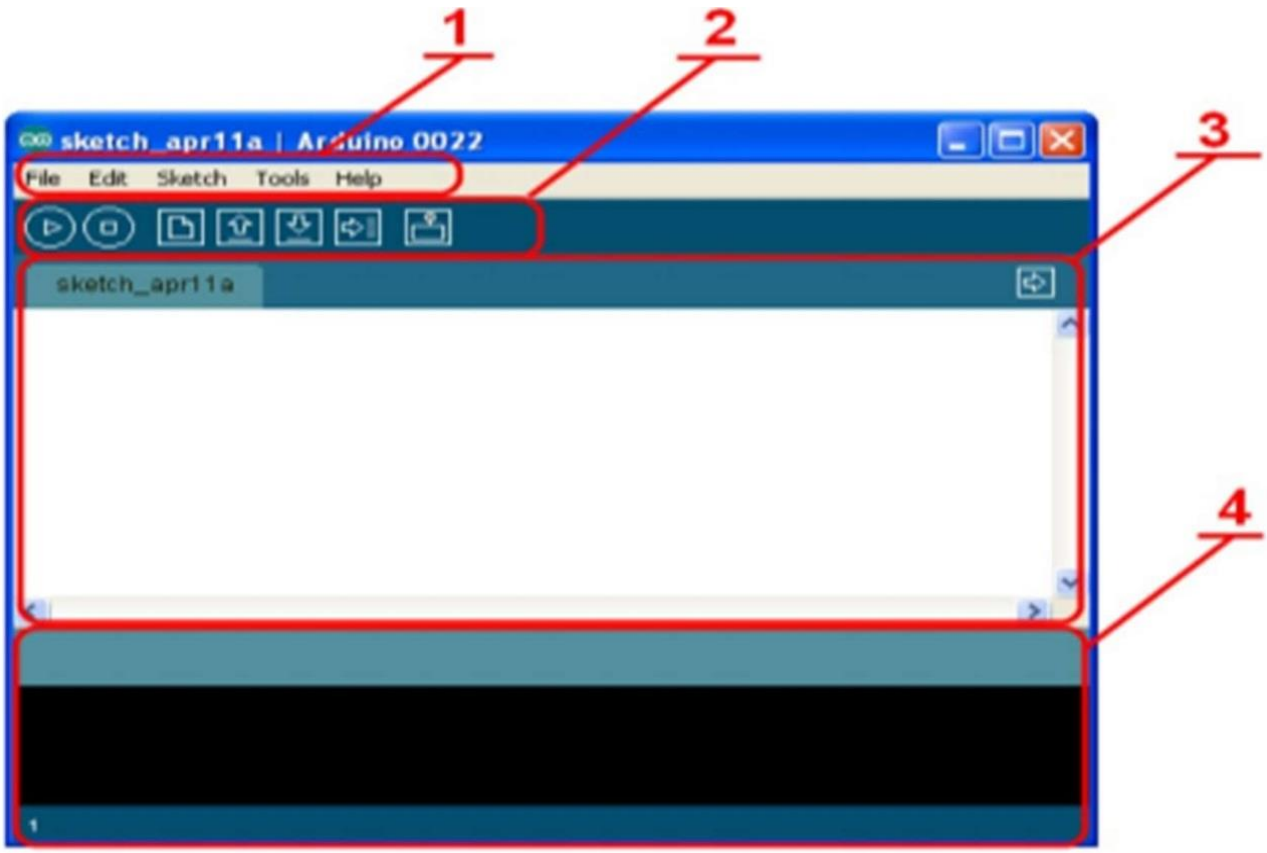


Figure II.12 : Présentation IDE Arduino

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

Le cadre numéro 2 : Il contient les boutons nécessaires pour la programmation des cartes.

Le cadre numéro 3 : C'est le bloc qui contiendra le programme crée.

Le cadre numéro 4 : Il aide à corriger en signalant les erreurs commises dans le programme. C'est le débogueur. [9]

Figure II.13: La barre d'outils de l'IDE d'Arduino

La barre d'outils de l'IDE d'Arduino comporte six boutons, encadrés en rouge et numérotés dans la figure précédente.

Bouton 1 : Ce bouton permet de vérifier le programme, il actionne un module qui cherche les erreurs dans le programme écrit.
Bouton 2 : Charge (téléverse) le programme dans la carte Arduino sélectionné dans (outils).

Bouton 3 : Il permet de créer un nouveau fichier.

Bouton 4 : Il permet d'ouvrir un fichier déjà enregistré, ou présent dans la liste des exemples fournie par Arduino.
Bouton 5 : C'est le bouton de sauvegarde, il permet d'enregistrer le fichier ou d'enregistrer des modifications apportées à ce dernier.
Bouton 6 : Ce bouton ouvre le moniteur série, qui permet de communiquer avec la carte Arduino à travers la liaison série. [16]

II.6.1 La structure d'un programme

Un programme Arduino comporte trois parties illustrées dans la figure Suivants :

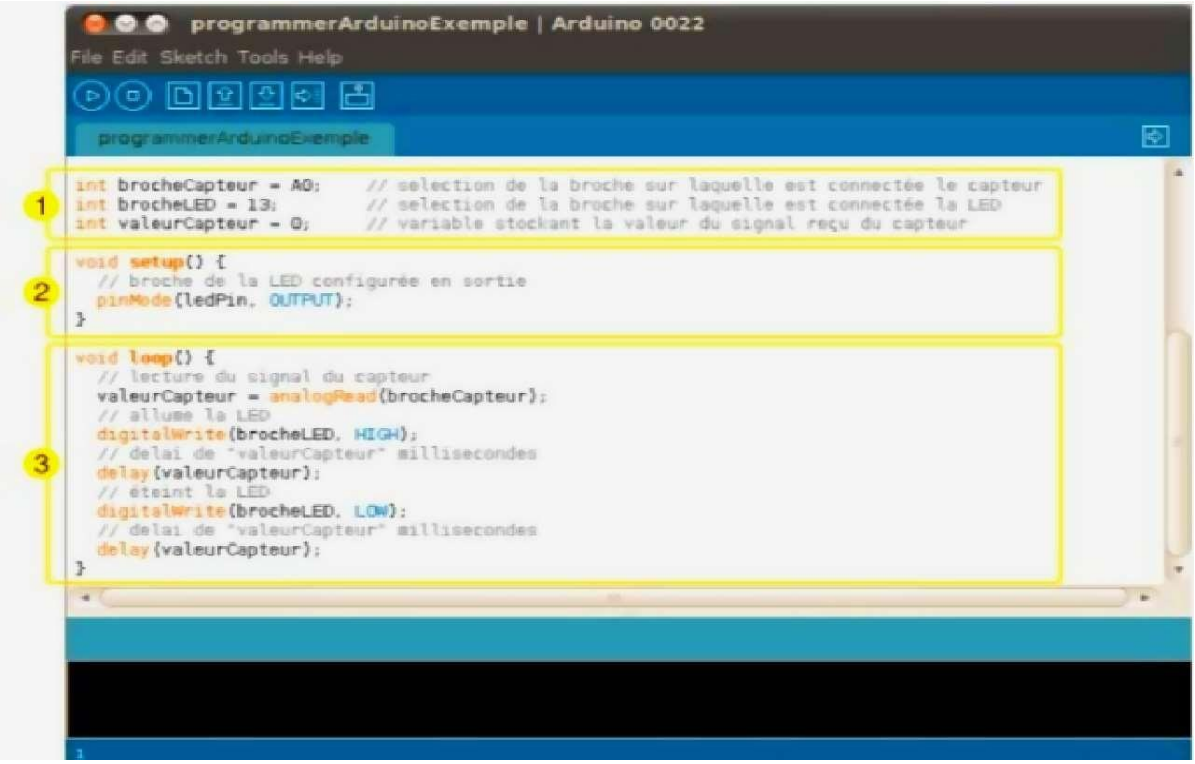


Figure II.14: Structure d'un programme

1. La partie déclaration des variables (optionnelle)
2. La partie initialisation et configuration des entrées/sorties : la fonction setup ()
3. La partie principale qui s'exécute en boucle : la fonction Loop()

Dans chaque partie d'un programme sont utilisées différentes instructions issues de la syntaxe du langage Arduino. [16]

Conclusion

Nous avons parlé dans ce chapitre sur la langage Ladder et ce programmation et bine sur les principes de ce langage, puis parlé sur la Grafcet en générale et après nous avons parlé un peu sur L'IDE .

Chapitre III

Carte Arduino

III.1 Introduction

Arduino est une plate-forme open source utilisée pour la construction de projets électroniques (allant de projets simples comme un thermomètre à des projets complexes comme des robots et des imprimantes 3D) La plate-forme Arduino se compose de deux parties principales : la partie matérielle et la partie logicielle La partie matérielle est l'Arduino carte et ses composants électroniques associés et autres composants matériels, tandis que la partie logicielle se compose de l'environnement de développement Arduino (Arduino IDE), qui représente l'environnement d'incubation Pour écrire le code du programme dans le langage Arduino et le télécharger sur les cartes Arduino pour contrôler le partition matérielle.

De plus, l'IDE Arduino utilise une version simplifiée de C et C++, ce qui facilite l'apprentissage de la programmation.

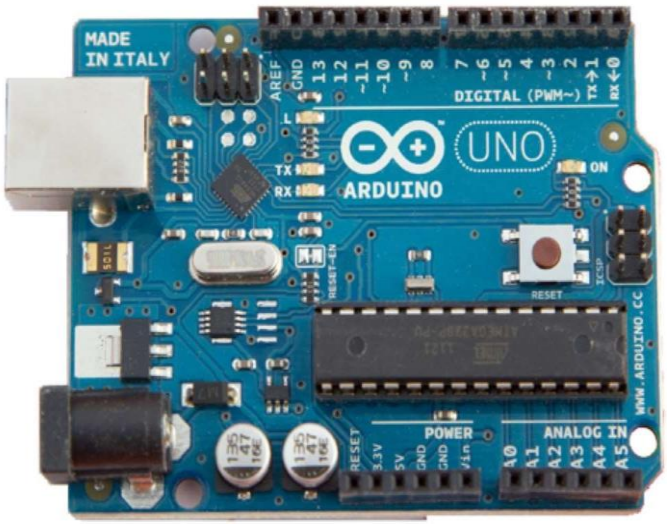


Figure III.1 : Description de la carte Arduino.

III.2 Définition de Arduino

Arduino est un ensemble matériel et logiciel qui permet d'apprendre l'électronique (en s'amusant) tout en se familiarisant avec la programmation informatique. Arduino est en source libre ; vous pouvez donc télécharger le schéma d'origine et l'utiliser pour élaborer votre propre carte et la vendre sans payer des droits d'auteur.[17]

III.3 Les gammes de la carte Arduino

Actuellement, il existe plus de 20 versions de module Arduino, nous citons quelques-unes afin d'éclaircir l'évaluation de ce produit scientifique et académique **:**

III.3.1 La carte Arduino UNO

La carte Arduino Uno est basée sur un ATMega328 cadencé à 16 MHz. C'est la plus simple et la plus économique carte à microcontrôleur d'Arduino. Des connecteurs situés sur les bords

extérieurs du circuit imprimé permettent d'enficher une série de modules complémentaires. Cette carte peut se programmer avec le logiciel Arduino disponible gratuitement en téléchargement à cette adresse. Le microcontrôleur ATMega328 contient un *bootloader* qui permet de modifier le programme sans passer par un programmateur. Cette carte est livrée avec un support en plastique mais sans cordon USB (voir articles conseillés).

Figure III.2: La carte Arduino UNO.

III.3.2 La carte Arduino Leonardo

Cette Carte Arduino Leonardo il présente de grandes similitudes avec l'Uno, même en apparence. Mais il ne faut pas les confondre, car il existe des différences notables entre les deux . [18]

Figure III.3 : carte Arduino Leonardo.

III.3.3 La carte Arduino Nano

La carte Arduino Nano est basée sur un ATMega328. Sa mémoire de 32 kB et ses E/S font de ce circuit un élément idéal pour les systèmes embarqués ou pour des applications nécessitant du multitâches.

La Nano peut se programmer avec le logiciel Arduino. Le contrôleur ATMega328 contient un *bootloader* qui permet de modifier le programme sans passer par un programmateur. Le logiciel est téléchargeable gratuitement. [19]

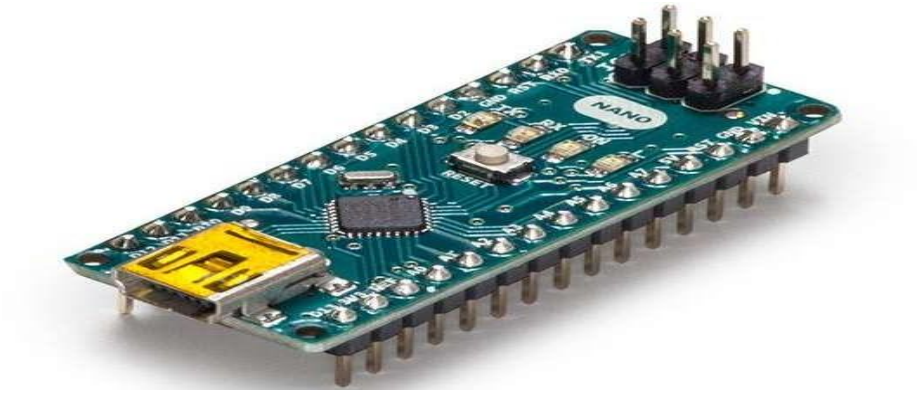


Figure III.4 : Carte Arduino Nano

III.3.4 La carte Arduino Yun

L'Arduino Yun est une carte Arduino mais qui possède quelques fonctionnalités avancées pour réaliser des prototypes d'objets connectés. Là où les Arduino Uno (cartes Arduino basiques) connectent des capteurs et peuvent les faire communiquer entre eux ou avec un ordinateur, l'Arduino Yun a le grand avantage de pouvoir les faire communiquer directement avec un réseau local ou internet, qu'il soit filaire ou wifi. Elle simplifie considérablement la communication distante, via internet, entre un ordinateur et des capteurs. [20]

Figure III.5 : La carte Arduino Yun

III.3.5 La carte Arduino Méga

La carte Arduino Méga est la carte la plus diffusée après la carte Arduino Uno. Elle offre un nombre d'entrées/sorties beaucoup plus important, un processeur plus puissant doté d'une mémoire plus vaste qui permet d'exploiter des algorithmes plus complexes. [21]

Figure III.6: Carte Arduino Méga.

III.4 Le langage d'Arduino

Le programme est lié à une série d'instruction élémentaires sous forme de texte, donc la carte litaprès exécute les instructions suivant un ordre un après l'autre.

- Un ordinateur.
- Une carte Arduino.
- Programme lié à l'Arduino. [22]

Syntaxe du langage : C et C++ qui est le suivant :

- **Code minimal** : son rôle de devisé le programme en deux parties La fonction :
- **Setup ()** : est considéré comme fonction d'initialisation on l'appelle une seul fois au début duProgramme.
- **Loop () :** c'est pour écrire le cœur du programme. Elle est appelée en permanence : en boucleinfinie.
- **Les instructions** : sont des lignes contenant des codes, exemple : « Fait ceci » « Fait cela »
- **Les points-virgules (;)** : pour finir les instructions
- **Les accolades []** : sont utilisées pour les fonctions ; les boucles. Elles sont obligatoires.
- **Les commentaires:** // cette ligne a un commentaire. /* */ pour plusieurs lignes
- **Les variables :** Les variables booléennes peuvent prendre deux valeurs soit vraie ou faux donc,si une variable vaut (0) on la considère comme variable booléennes fausse et si une variable prend n'importe quelle valeur différente de zéro on la considère comme variable booléenne vraie.
  - **char** (variable 'caractère')
  - **Int** (variable 'nombre entier')
  - **long** (variable 'nombre entier de très grande taille')
  - **string** (variable 'chaine de caractères')
  - **Array** (tableau de variables). [22]

III.5 Elément de la carte Arduino

La carte Arduino Uno est constituée de 14 broches d'entrées/sorties digitales, dont six sont utilisables en PWM, de 6 broches d'entrées analogiques, d'une connectique USB, d'une connectique d'alimentation, d'un port ICSP et d'un bouton RESET.

Aujourd'hui, Arduino est l'un des éléments les plus précieux pour débuter dans le domaine de l'électronique**,** puisqu'il a été conçu pour les artistes, les designers, les amateurs et toute personne intéressée par la création de projets électroniques ; aussi bien au niveau logiciel qu'au niveau matériel. Cela est principalement dû au fait que l'**Arduino** ne nécessite pas de matériel séparé pour télécharger un nouveau code sur le microcontrôleur car il utilise un câble USB**.**

Ajouté à cela, grâce au fait qu'il utilise une version simplifiée de C ++, l'IDE Arduino se caractérise par sa facilité de programmation. De plus, ladite carte a la capacité d'interagir avec des boutons, des moteurs, des haut-parleurs, des caméras, des LED, des unités GPS, des smartphones et Internet. Ce qui en fait un logiciel et un matériel flexibles pour mener à bien des projets éducatifs ou tout projet électronique.

Mais, au-delà de ses caractéristiques utiles pour cela, il est important de savoir quels sont chacun des composants d'une carte Arduino**.** Depuis, ces éléments sont indispensables pour pouvoir partir de zéro et ainsi, programmer correctement vos premiers projets *open source*. Donc, ensuite, vous connaîtrez les modules qui composent une carte Arduino [23]. Et pour le coté matériel exister :

III.5.1 Le Microcontrôleur

C'est un circuit intégré qui est défini comme la partie qui a le pouvoir de traiter toutes les informations et, par conséquent, est la zone où le code appelé «Sketch» est enregistré dans le logiciel Arduino**.** Par conséquent, il est estimé comme le cerveau d'une carte Arduino et se caractérise par être un élément noir avec des pieds métalliques. [24]

En ce sens, le microcontrôleur sur une carte Arduino fonctionne comme un circuit intégré programmable qui, par défaut, a la capacité d'effectuer des opérations mathématiques complexes à grande vitesse .

III.5.2 Broches numériques

Ce sont ceux qui s'occupent de détecter s'il y a un ZERO ou UN logique et sont utilisés à la fois pour l'entrée numérique et la sortie numérique . Soit, par exemple, d'appuyer sur un bouton et d'allumer une LED , respectivement. Cela signifie que les broches numériques sont utilisées pour appuyer sur des boutons ou des appareils qui envoient et / ou reçoivent des informations numériques . Ces broches vont de 0 à 13.

III.5.3 Broches analogiques

Ils correspondent à la surface des broches dans le cadre du «analogique**»** et grâce à cela, ils sont capables de détecter des signaux analogiques (température ou lumière, par exemple). Pour ce

faire, ils disposent d'un segment de tension de fonctionnement qui oscille entre 0 et 5 volts . En conséquence, les broches analogiques sont des broches qui peuvent lire le signal d'un capteur analogique et le transformer en une valeur numérique qui peut être affichée . Ce qui signifie que, grâce aux broches analogiques, il est possible de mesurer des choses dans le monde réel . [24]

III.5.4 Broches d'alimentation du capteur

Ils fournissent une alimentation entre 6 et 12 volts qui, directement, va au régulateur de tension. Grâce à ces broches, il existe également la possibilité d'alimenter l'Arduino avec des piles ou des piles . Cependant, de cette manière, il n'y a pas de diode de protection de polarité ou de protection contre les surintensités.

III.5.5 Broches PWM

Ce sont des broches spécifiques qui agissent comme des broches numériques normales et, en même temps, peuvent être utilisées pour la «modulation de largeur d'impulsion» ou ce que l'on appelle «PWM» en anglais. Physiquement, ces broches se distinguent en contenant un tilde (~) à côté de certaines des broches numériques, telles que: 3, 5, 6, 9, 10 et 11 . [24]

III.5.6 Broche AREF

Il s'agit d'une seule broche qui, la plupart du temps, peut être laissée seule en place. Fondamentalement, il s'agit d'un élément qui a un support de référence analogique et est parfois utilisé pour pouvoir établir une tension de référence externe entre 0 et 5 volts. Qui fonctionne comme la limite supérieure des broches d'entrée analogiques .

III.5.6 Broche GND

Sur une carte Arduino, il y a toujours plusieurs broches GND et l'une d'entre elles peut être utilisée pour connecter le circuit à la terre . D'où son nom qui est l'abréviation de «land» en anglais. Pour sa part, lorsqu'il s'agit de le localiser, il se trouve juste à côté du pin AREF.

III.5.7 Bouton de réinitialisation

Il se réfère au bouton de réinitialisation qui est connecté à la broche numéro un du microcontrôleur , directement. Ce qui est mieux connu sous le nom de «Clear» ou «Master Clear» . Maintenant, une fois que ce bouton est enfoncé, la broche de réinitialisation est temporairement connectée à la terre, afin de réinitialiser tout code chargé sur la carte Arduino

Il convient de noter que cela nécessite 5 volts pour exécuter le programme, tandis que le bouton interrompt la tension à 0 volts pour arrêter le même programme qu'Arduino est en cours d'exécution , afin de le redémarrer.

III.5.8 Port USB

C'est un composant également identifié comme «Connecteur USB» et c'est celui qui vous permet de connecter la carte Arduino à un certain ordinateur. De cette manière, il offre la possibilité de charger le code requis et d'alimenter correctement la carte .

III.5.9 Controle USB

Fondamentalement, le contrôle USB consiste en un modérateur qui opère entre le microcontrôleur et le logiciel. Raison pour laquelle, cela se concentre sur la conversion des informations transmises par le microcontrôleur en données qui vont à l'équipement utilisé.

III.5.10 Câble d'alimentation

Comme son nom l'indique, c'est un élément qui offre la possibilité d'alimenter la carte avec une tension en courant continu pouvant aller de 7 à 12 volts. Compte tenu de cela, chaque fois que la carte Arduino est connectée à une prise électrique, le voyant d'alimentation LED devra s'allumer. Sinon, quelque chose ne va probablement pas. [24]

III.5.11 Variateur

Dans une carte Arduino, le régulateur de tension est chargé de contrôler la quantité de tension restante sur la carte. En d'autres termes, il agit pratiquement comme un chien de garde qui empêchera l'entrée d'une tension supplémentaire avec laquelle il pourrait endommager le circuit. Généralement, ce régulateur de tension prend en charge une sortie stable de 5 volts sans dépendre de la tension d'entrée.

III.5.12 Réception (RX) et transmission (TX) série

Ce sont des sortes de marques qui, en électronique, sont utilisées pour indiquer les broches responsables de la communication série. En conséquence, il se compose d'un composant de carte Arduino Uno qui se concentre sur l'externalisation lorsque la carte reçoit ou transmet des données, visuellement. Compte tenu de cela, il consiste en une transmission qui est effectuée via les broches zéro et un.

III.5.13 Cristal

Plus précisément, cet élément a la capacité de régler l'impulsion ou le temps de travail du microcontrôleur afin de fonctionner parfaitement sur une carte Arduino. Ainsi, un cristal est une sorte de circuit oscillateur électronique qui crée un signal électrique avec une fréquence précise afin de contrôler le temps de travail exact. [24]

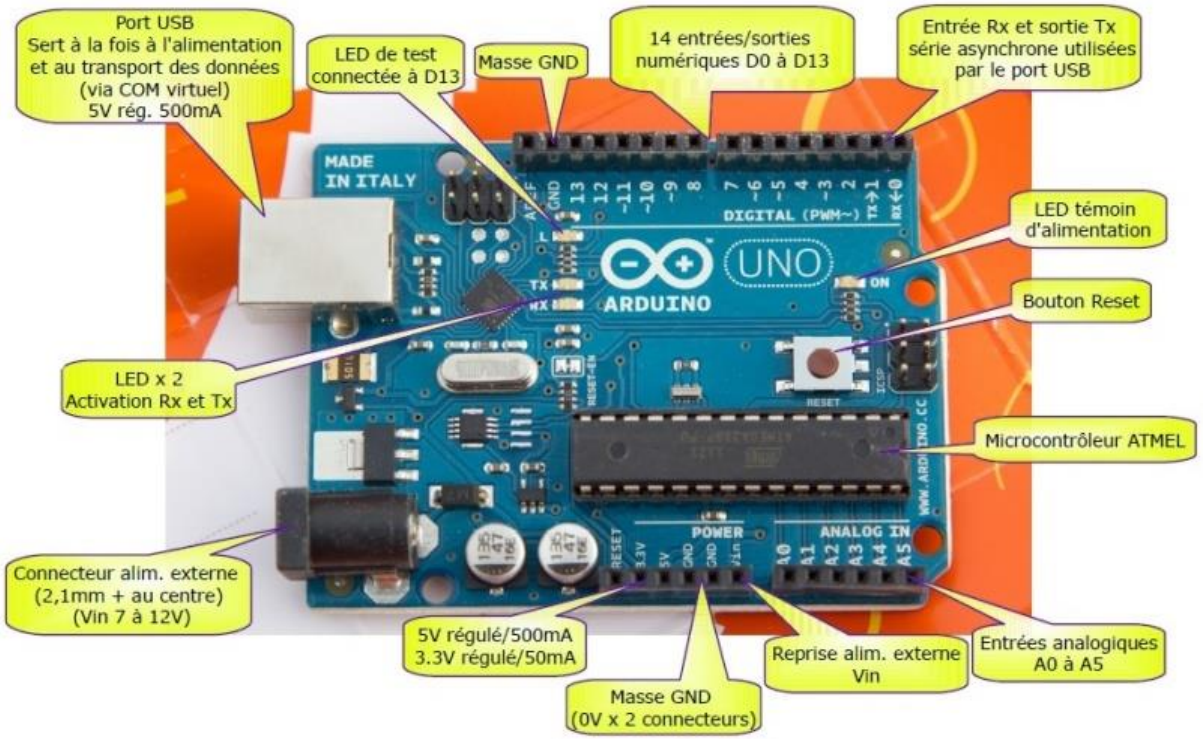


Figure 7: Représentation des éléments d'une carte Arduino**.**

II.6 Présentation de Arduino méga 2560

La carte Arduino Mega 2560 est basée sur un ATMega2560 cadencé à 16 MHz. Elle dispose de 54 E/S dont 14 PWM, 16 analogiques et 4 UARTs. Elle est idéale pour des applications exigeant des caractéristiques plus complètes que la Uno.

Des connecteurs situés sur les bords extérieurs du circuit imprimé permettent d'enficher une série de modules complémentaires.

Cette carte peut se programmer avec le logiciel Arduino disponible gratuitement en téléchargement sur le site officiel. **www.Arduino.cc** .

Le contrôleur ATMega2560 contient un *bootloader* qui permet de modifier le programme sans passer par un programmateur.

Cette carte est livrée avec un support en plastique mais sans cordon USB. [25]

III.6.1 Caractéristiques technique de la carte Arduino Méga 2560

- Alimentation: - via port USB ou - 7 à 12 V sur connecteur alim
- Microprocesseur: ATMega2560
- Mémoire flash: 256 kB
- Mémoire SRAM: 8 kB
- Mémoire EEPROM: 4 kB
- 54 broches d'E/S dont 14 PWM.
- 16 entrées analogiques 10 bits
- Intensité par E/S: 40 mA

- Cadencement: 16 MHz
- 3 ports série
- Bus I2C et SPI
- Gestion des interruptions
- Fiche USB B
- Version: Rev 3
- Dimensions: 107 x 53 x 15 mm

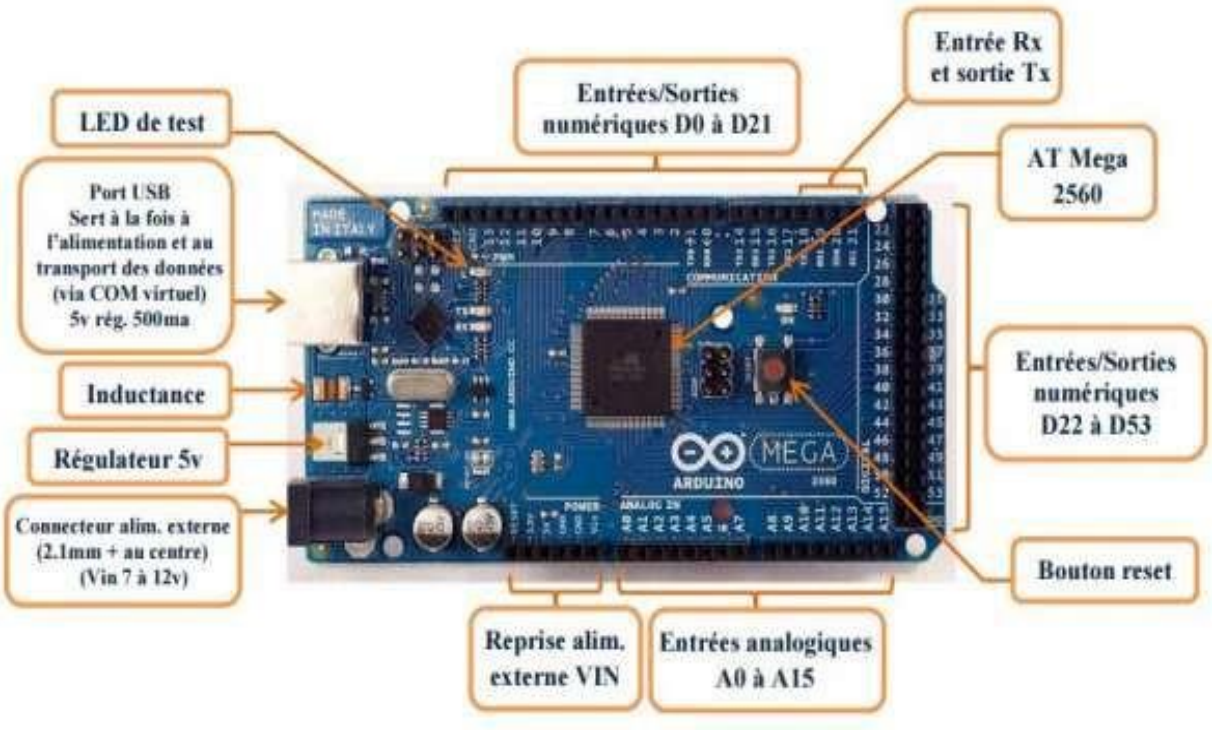


Figure 8 : Présentation de Arduino méga 2560

III.6.2 Alimentation

La carte Arduino Mega 2560 peut-être alimentée soit via la connexion USB (qui fournit 5V jusqu'à 500mA) ou à l'aide d'une alimentation externe. La source d'alimentation est sélectionnée automatiquement par la carte.

L'alimentation externe (non-USB) peut être soit un adaptateur secteur (pouvant fournir typiquement de 3V à 12V sous 500mA) ou des piles (ou des accus). L'adaptateur secteur peut être connecté en branchant une prise 2.1mm positif au centre dans le connecteur jack de la carte. Les fils en provenance d'un bloc de piles ou d'accus peuvent être insérés dans les connecteurs des broches de la carte appelées GND (masse ou 0V) et Vin (Tension positive en entrée) du connecteur d'alimentation.

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

La carte Arduino Mega2560 diffère de toutes les cartes précédentes car elle n'utilise par le circuit intégré FTDI usb-vers-série. A la place, elle utilise un Atmega8U2 programmé en convertisseur USB-vers-série. [26]

Les broches d'alimentation sont les suivantes :

- VIN. La tension d'entrée positive lorsque la carte Arduino est utilisée avec une source de tension externe (à distinguer du 5V de la connexion USB ou autre source 5V régulée). Vous pouvez alimenter la carte à l'aide de cette broche, ou, si l'alimentation est fournie par le jack d'alimentation, accéder à la tension d'alimentation sur cette broche.
- 5V. La tension régulée utilisée pour faire fonctionner le microcontrôleur et les autres composants de la carte (pour info : les circuits électroniques numériques nécessitent une tension d'alimentation parfaitement stable dite "tension régulée" obtenue à l'aide d'un composant appelé un régulateur et qui est intégré à la carte Arduino). Le 5V régulé fourni par cette broche peut donc provenir soit de la tension d'alimentation VIN via le régulateur de la carte, ou bien de la connexion USB (qui fournit du 5V régulé) ou de tout autre source d'alimentation régulée. [26]
- 3V3. Une alimentation de 3.3V fournie par le circuit intégré FTDI (circuit intégré faisant l'adaptation du signal entre le port USB de votre ordinateur et le port série de l'ATmega) de la carte est disponible : ceci est intéressant pour certains circuits externes nécessitant cette tension au lieu du 5V). L'intensité maximale disponible sur cette broche est de 50mA
- GND. Broche de masse (ou 0V).The power pins are as follows:

III.6.3 Mémoire

L'ATmega 2560 a 256Ko de mémoire FLASH pour stocker le programme (dont 8Ko également utilisés par le bootloader). L'ATmega 2560 a également 8 ko de mémoire SRAM (volatile) et 4Ko d'EEPROM (non volatile - mémoire qui peut être lue à l'aide de la librairie EEPROM) .

Pour info : Le bootloader est un programme préprogrammé une fois pour toute dans l'ATméga et qui permet la communication entre l'ATmega et le logiciel Arduino via le port USB, notamment lors de chaque programmation de la carte. [26]

III.6.4 Entrées et sorties numériques

Chacune des 54 broches numériques de la carte Mega peut être utilisée soit comme une entrée numérique, soit comme une sortie numérique, en utilisant les instructions pinMode(), digitalWrite() et digitalRead() du langage Arduino. Ces broches fonctionnent en 5V. Chaque broche peut fournir ou recevoir un maximum de 40mA d'intensité et dispose d'une résistance interne de "rappel au plus" (pull-up) (déconnectée par défaut) de 20-50 KOhms. Cette résistance interne s'active sur une broche en entrée à l'aide de l'instruction digitalWrite(broche, HIGH).

De plus, certaines broches ont des fonctions spécialisées :

- **Communication Serie**: Port Serie Serial : 0 (RX) and 1 (TX); Port Serie Serial 1: 19 (RX) and 18 (TX); Port Serie Serial 2: 17 (RX) and 16 (TX); Port Serie Serial 3: 15 (RX) and 14 (TX). Utilisées pour recevoir (RX) et transmettre (TX) les données séries de

niveau TTL. Les broches 0 (RX) and 1 (TX) sont connectées aux broches correspondantes du circuit intégré ATmega8U2 programmé en convertisseur USB-vers-série de la carte, composant qui assure l'interface entre les niveaux TTL et le port USB de l'ordinateur.

- **Interruptions Externes**: Broches 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), et 21 (interrupt 2). Ces broches peuvent être configurées pour déclencher une interruption sur une valeur basse, sur un front montant ou descendant, ou sur un changement de valeur. Voir l'instruction attachInterrupt() pour plus de détails.
- **Impulsion PWM (largeur d'impulsion modulée)**: Broches 0 à 13. Fournissent une impulsion PWM 8-bits à l'aide de l'instruction analogWrite().
- **SPI (Interface Série Périphérique)**: Broches 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). Ces broches supportent la communication SPI (Interface Série Périphérique) disponible avec la librairie pour communication SPI. Les broches SPI sont également connectées sur le connecteur ICSP qui est mécaniquement compatible avec les cartes Uno, Duemilanove et Diecimila. [26]
- **I2C**: Broches 20 (SDA) et 21 (SCL). Supportent les communications de protocole I2C (ou interface TWI (Two Wire Interface - Interface "2 fils"), disponible en utilisant la librairie Wire/I2C (ou TWI - Two-Wire interface - interface "2 fils") . Noter que ces broches n'ont pas le même emplacement que sur les cartes Uno, Duemilanove ou Diecimila. [26]
- **LED**: Broche 13. Il y a une LED incluse dans la carte connectée à la broche 13. Lorsque la broche est au niveau HAUT, la LED est allumée, lorsque la broche est au niveau BAS, la LED est éteinte.

III.6.5 Broches analogiques

La carte Mega2560 dispose de 16 entrées analogiques, chacune pouvant fournir une mesure d'une résolution de 10 bits (càd sur 1024 niveaux soit de 0 à 1023) à l'aide de la très utile fonction analogRead() du langage Arduino. Par défaut, ces broches mesurent entre le 0V (valeur 0) et le 5V (valeur 1023), mais il est possible de modifier la référence supérieure de la

plage de mesure en utilisant la broche AREF et l'instruction analogReference() du langage Arduino. [26]

Note : les broches analogiques peuvent être utilisées en tant que broches numériques.

III.6.6 Autres broches

Il y a deux autres broches disponibles sur la carte :

- **AREF** : Tension de référence pour les entrées analogiques (si différent du 5V). Utilisée avec l'instruction analogReference().
- **Reset** : Mettre cette broche au niveau BAS entraîne la réinitialisation
- (= le redémarrage) du microcontrôleur. Typiquement, cette broche est utilisée pour ajouter un bouton de réinitialisation sur le circuit qui bloque celui présent sur la carte.

Conclusions

Larduino c'est un mélange d'électronique et de programmation donc c'est de l'électronique embarquée qui nous facilite la réalisation de beaucoup de projets .

Chapitre IV
Réalisation

IV.1 Introduction

Depuis les temps anciens, les hommes ont cherché un moyen de renforcer. La production est plus confortable et serrée pour réduire le stress et permettre une assistance en toutes circonstances. Il est également plus respectueux de l'environnement et fait partie de notre vie économique productive quotidienne, et en conséquence, nous avons réalisé ce modèle, que nous n'avons épargné aucun effort ni argent pour compléter afin de fournir le bénéfice public.

IV. 2 Exemple d'une machine de remplissage des bouteilles de jus

IV.2.1 Fonctionnement

Lorsque la machine est lancée. Le convoyeur convoie la bouteille vide jusqu'à ce que la bouteille soit détectée par un capteur placé sur le bord du convoyeur.
Le convoyeur s'arrêté de bouger et la pompe à jus se met en marche pendant un certain temps jusqu'à ce que la bouteille soit remplie de jus.
Lorsque le temps de remplissage du jus est écoulé, la pompe s'arrête et le convoyeur se remet en marche.
Le détecteur (C2) détecte la bouteille quand elle arrive ensuite un vérin fait tourner le bouchon pour le fermer avec un moteur qui tourne à une vitesse rapide,
Le détecteur (C3) détecte l'arrivée de la bouteille et donne un signal à un vérin qui prendra en charge l'attribution d'un cachet au-dessus du bouchon fermé et donc le processus est repeter pour les bouteilles qui suivent.

IV.2.2 Cahier de charge

- Impulsions sur le bouton (BM) pour que le convoyeur (M1) démarre.
- Le capteur de position (CPOS) détecte l'arrivé de la bouteille qui demande un verrouillage du convoyeur (M1).
- Démarrage de l'un des pompes à jus (M2), (M3), (M4) ou de toutes les pompes en même temps (mélange).
- Après certain temps arrêt de la pompe à jus (M2) ,(M3) ,(M4).
- Le deuxième capteur (IR1) détecte l'arrivé de la bouteille qui demande un verrouillage du convoyeur (M1).
- Démarrage du vérin (V1).
- Après un certain temps l'arrêt du vérin (V1) et déverrouillage du convoyeur (M1).
- Le Troisième capteur (IR2) détecte l'arrivé de la bouteille et demande un verrouillage du convoyeur (M1).
- Démarrage du vérin (V2).
- Après un certain temps, arrêt du vérin (V2).
- Déverrouillage du convoyeur (M1).

- Le deuxième capteur (IR1) détecte l'arrivé de la bouteille qui demande un verrouillage du convoyeur (M1).
- -Démarrage du Vérin (V1).
- -Après un certain temps l'arrêt du vérin (V1) et déverrouillage du convoyeur (M1)
- Le Troisième capteur (IR2) détecte l'arrivé de la bouteille et demande un verrouillage du convoyeur (M1).
- Démarrage du Vérin (V2).
- Après un certain temps, arrêt du vérin (V2).
- Déverrouillage du convoyeur (M1).

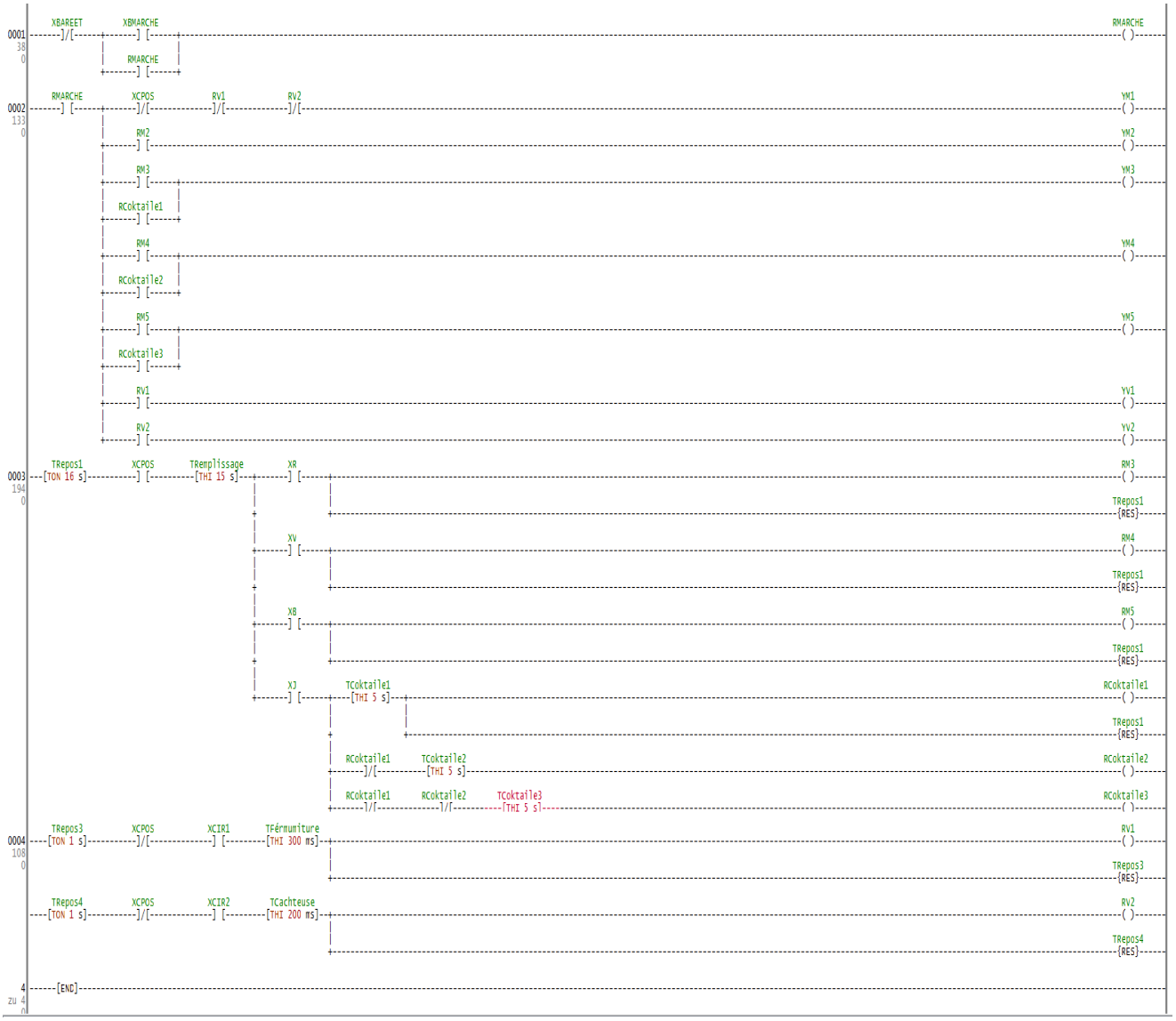


Figure IV. 1 : Programme Ladder de Réalisation

Tableau IV. 1 : les variables

PIC Adresse In	**Nom en LDMICRO**	**Les Appareils**
RA1	XCPOS	Capteur de couleurs
RA2	XR	Capteur de couleurs
RA3	XV	Capteur de couleurs
RA4	XB	Capteur de couleurs
RE0	XJ	Capteur de couleurs
RA5	X BMARCHE	Bouton poussoir Marche
RE1	X BAREET	Bouton poussoir Areet
RE2	XCIR1	Capteur infrarouge 1
RC0	XCIR2	Capteur infrarouge 2

Tableau IV.2 : les variables

PIC Adresse Out	**Nom en LDMICRO**	**Les Appareils**
RB6	YM1	Moteur 1
RB7	YM2	Moteur 2
RB2	YM3	Moteur 3
RD4	YM4	Moteur 4
RD5	YM5	Moteur 5
RB3	YV1	Vérin électrique 1
RB1	YV2	Vérin électrique 2

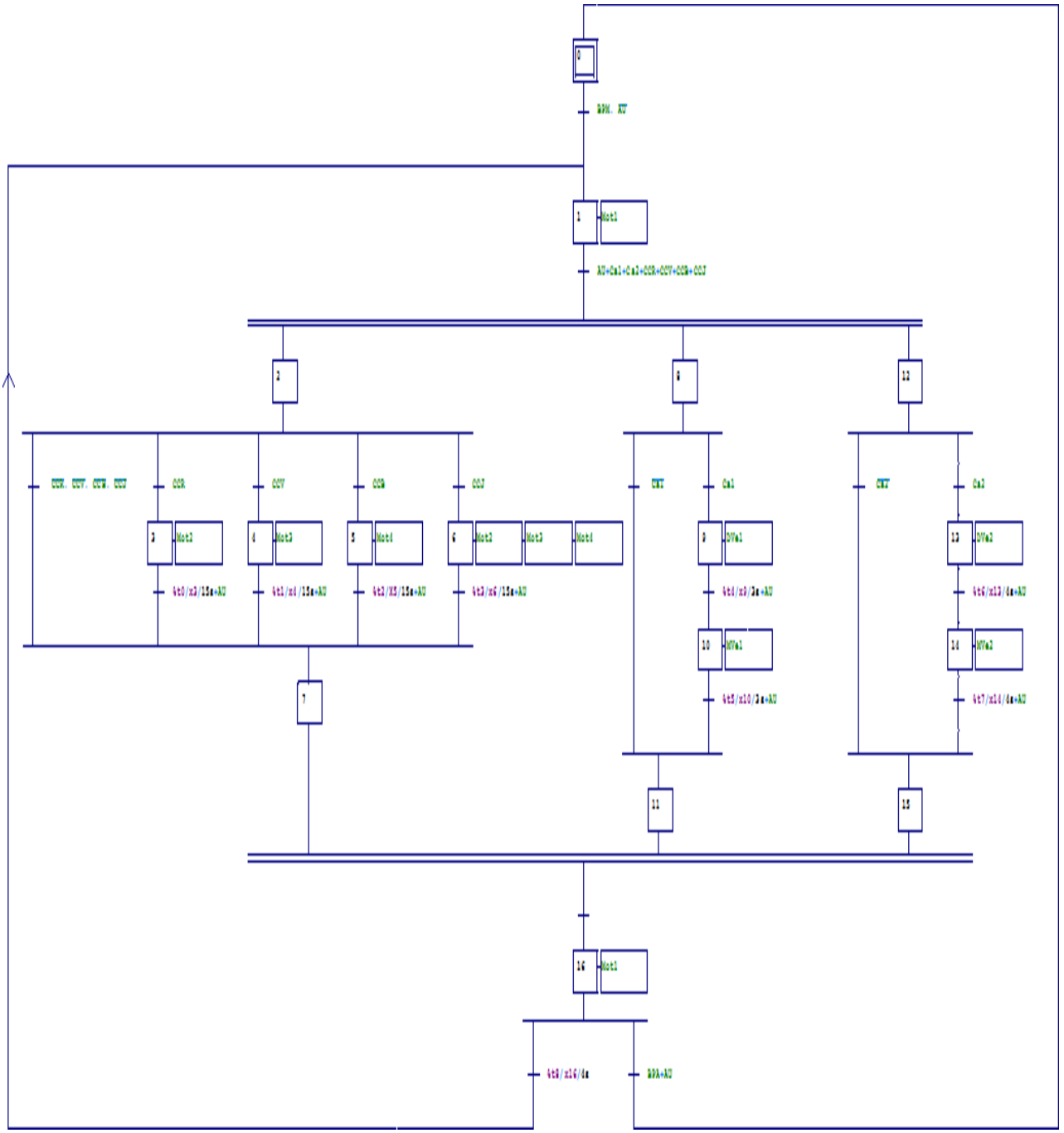


Figure IV. 2 : GRAFCET du système

IV. 3 Moteur pas a pas

Un moteur pas à pas permet de transformer une impulsion électrique en un mouvement angulaire. Deux moteurs pas à pas. On trouve trois types de moteurs pas à pas :

- le moteur à réluctance variable ;
- le moteur à aimants permanents ;
- le moteur hybride, qui est une combinaison des deux technologies précédentes.

Le moteur pas à pas fut inventé en 1936 par Marius Lavet, un ingénieur français des Arts et Métiers, pour l'industrie [27]. Et voila les avantages et les inconvénients du moteur pas à pas.

Moteurs pas à pas

- **Avantages**
  - Asservissement de position ou de vitesse en boucle ouverte
  - Fort couple à basse vitesse
  - Simplicité de mise en œuvre
  - Positionnement statique
  - Fiabilité
  - Faible prix
- **Inconvénients**
  - Positionnement discret
  - Faible vitesse maximale
  - Bruyant, source d'oscillations
  - Faible puissance
  - Faible rendement

Figure IV.3 : les avantages et les inconvénients de moteur pas à pas

IV.3.1 les caractéristiques de moteur NEMA 23

C'est le moteur pas a pas que nous avons utilisé dans notre travaille

- Norme du moteur pas à pas: NEMA 23.
- Dimensions: 56x56x54.5mm.
- Poids: 720g.
- Diamètre de l'axe: Ø6. 35 x ~19.1 mm.
- Nombre de phase: 2.
- Nombre de pas: 200.
- Pas angulaire: 1,8° (+/-5%)
- Couple de maintien: 12,6 Kg.cm (175oz.in)

Figure IV.4 **:** Moteur pas a pas NEMA 23

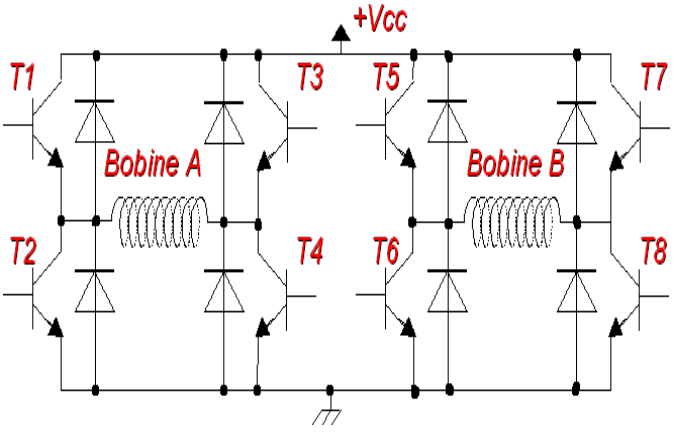


Figure IV. 5 : commande moteur pas a pas

IV.4 Servomoteur

Dans ce chapitre, nous allons parler d'un moteur que nos amis modélistes connaissent bien : le Servomoteur (abrégé : "servo"). C'est un moteur un peu particulier, puisqu'il confond un ensemble de mécanique et d'électronique, mais son principe de fonctionnement reste assez simple. Les parties seront donc assez courtes dans l'ensemble car les servomoteurs contiennent dans leur "ventre" des moteurs à courant continu que vous connaissez à présent. Cela nous évitera des explications supplémentaires [28].

IV .4 .1 Composition d'un servomoteur

Les servomoteurs ont donc l'avantage d'être *asservis* en position angulaire. Cela signifie, nous expliquais, que l'axe de sortie du servomoteur respectera une consigne d'orientation que vous lui envoyez en son entrée. En plus, tenez-vous bien, si par malheur les roues venaient à changer d'orientation en passant sur un caillou par exemple, l'électronique interne du servomoteur essaiera tant bien que mal de conserver cette position ! Et quelle que soit la force que l'on exerce sur le bras du servomoteur, il essayera de toujours garder le même angle (dans les limites du raisonnable évidemment). En quelque sorte vous ne pilotez pas directement le moteur, mais plutôt vous imposez le résultat que vous voulez avoir en sortie.

IV.4.2 Apparence

On en trouve de toutes les tailles et de toutes les puissances. La plupart du temps la sortie peut se positionner entre 0 et 180°. Cela dit, il en existe également dont la sortie peut se débattre sur seulement 90° et d'autres, ayant un plus grand débattement, sur 360°. Ceux qui ont la possibilité de faire plusieurs tours sont souvent appelés servo-treuils. Enfin, les derniers, qui peuvent faire tourner leur axe sans jamais se buter, sont appelés servomoteurs à rotation continue. Les servomoteurs sont très fréquemment employés dans les applications de modélisme pour piloter le safran d'un bateau, le gouvernail d'un avion ou bien même les roues d'une voiture téléguidée dont on a parlé jusqu'à présent. Maintenant que les présentations sont faites, mettons-le à nu ! Il est composé de plusieurs éléments visibles … :

- Les fils, qui sont au nombre de trois (nous y reviendrons).
- L'axe de rotation sur lequel est monté un accessoire en plastique ou en métal.
- Le boitier qui le protège.

… mais aussi de plusieurs éléments que l'on ne voit pas :

- un moteur à courant continu
- des engrenages pour former un réducteur (en plastique ou en métal)
- un capteur de position de l'angle d'orientation de l'axe (un potentiomètre bien souvent)
- une carte électronique pour le contrôle de la position de l'axe et le pilotage du moteur à courant continu [28]

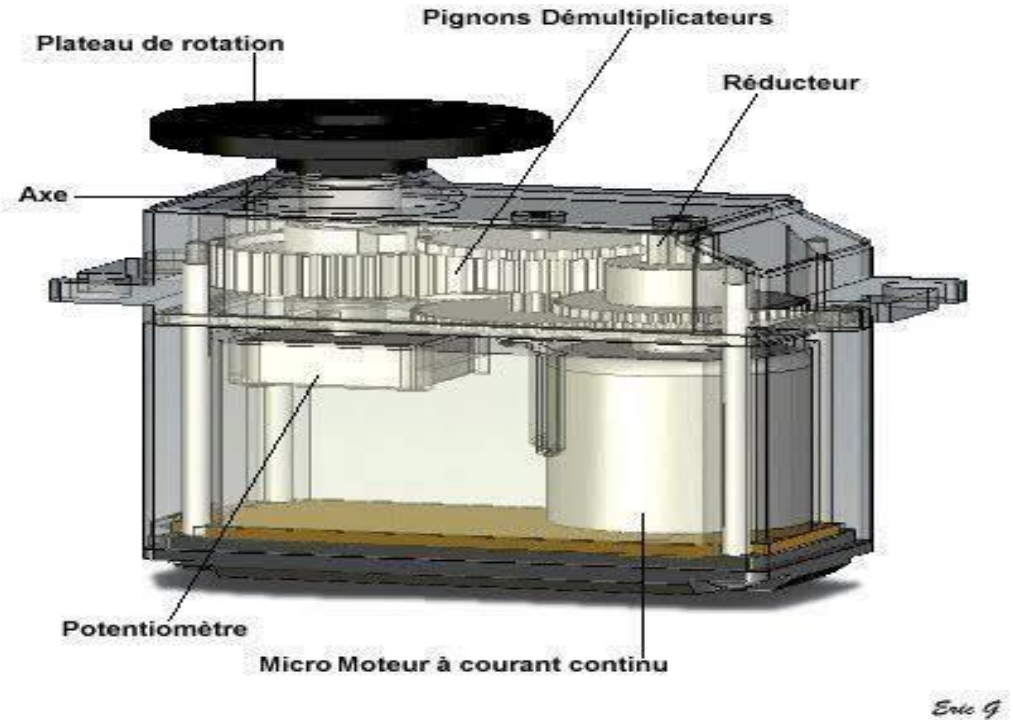


Figure IV. 6 : Schémas servomoteur

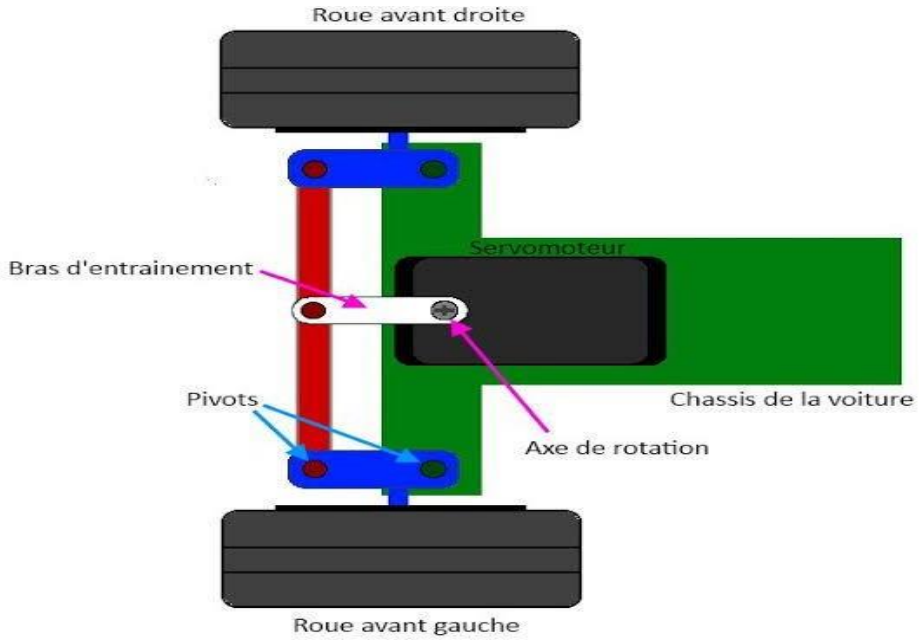


Figure IV. 7 . Exemple de jeux d'enfants avec servomoteur

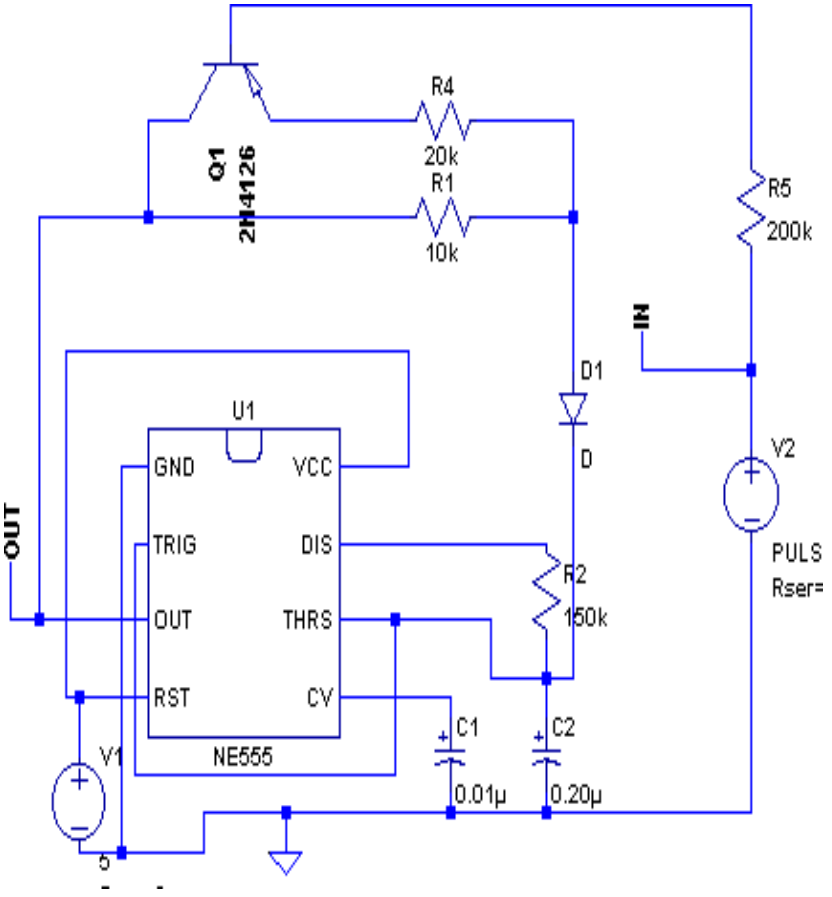


Figure IV. 8 schéma électrique de servomoteur

IV. 5 Servomoteur MG6600R

c'est le servomoteur pas a pas que nous avons utilisé dans notre travaille

Le MG996R est un servomoteur à pignons métalliques à fort couple. Il est équipé de pignons métalliques et de deux roulements lui assurant une précision et une longévité sans égal. Même avec un couple allant jusqu'à 10kg/cm ,avec caractéristiques dont les suivant :

- Dimension: 40mm x 19mm x 43mm.
- Poids: 56g.
- Vitesse de rotation: 0,17s / 60 degrés (4,8V sans charge)
- Vitesse de rotation: 0,13s / 60 degrés (6,0V sans charge)
- Couple: 13 kg.cm à 4,8V.
- Couple de décrochage: 15 kg.cm à 6V.
- Tension de fonctionnement: 4,8 - 7,2V.
- Type de roues : Toutes roues en métal.

Figure IV. 9 : Servomoteur MG6600R

IV. 6 Définition moteur DC

Figure IV 10 : moteur DC

Le moteur DC, appelé aussi moteur à courant continu, fait partie de la classe des moteurs électriques et sert essentiellement à transformer de l'énergie électrique en énergie mécanique. La plupart des formes de moteurs DC reposent, dans ce contexte, sur les forces magnétiques et disposent de mécanismes internes de types électronique ou électromécanique. Les moteurs à courant continu conventionnels se caractérisent d'abord par le collecteur qui change périodiquement le sens du flux électrique à l'intérieur du moteur. Une des plus puissantes versions du moteur DC classique que l'on retrouve dans de nombreuses applications est le moteur DC sans balais ; celui-ci fonctionne sans frottement et offre donc notamment une durée de vie plus longue.

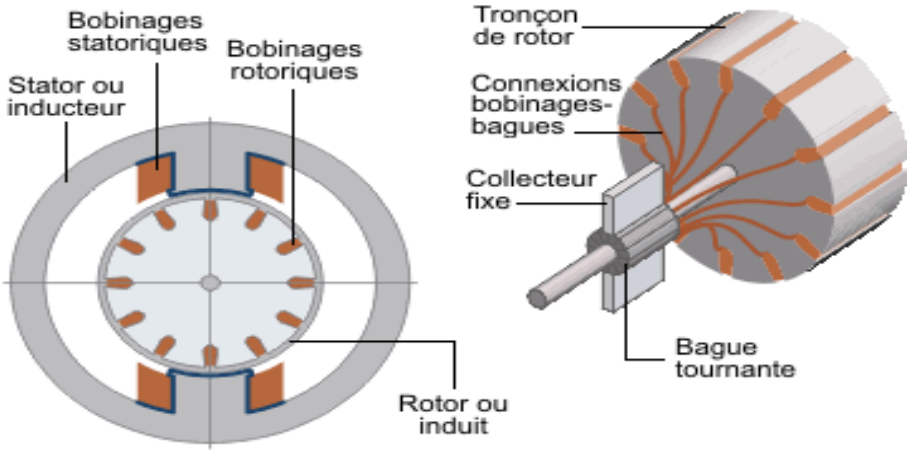


Figure IV. 11 : schémas de moteur DC

Malgré l'utilisation croissante et la grande rivalité avec les moteurs AC, lesquels, en tant que moteurs triphasés, font également partie des moteurs électriques, les moteurs DC jouent aujourd'hui encore un rôle très important. Ils sont souvent utilisés dans les applications industrielles, en raison de leurs propriétés — notamment la possibilité qu'ils offrent d'ajuster leur couple et leur vitesse de manière précise et extrêmement variable —. C'est par exemple le cas des servo-actionneurs de la série PMA de Harmonic Drive AG qui sont composés d'un moteur à courant continu à haute dynamique et d'une mini-cartouche de réduction PMG avec codeur incrémental. Ils sont parfaitement adaptés aux applications de l'industrie des semi-conducteurs, de la technique médicale et des machines de mesure et de tests. [29]

IV.6.1 Structure et fonctionnement du moteur DC

Le moteur DC classique présente une forme épurée et ne compte que quelques composants. Les éléments porteurs sont en première ligne le stator et le rotor. Le stator, qui est fixe et statique, est la plupart du temps composé d'un électroaimant ou principalement sur les petites machines d'un aimant permanent. À l'intérieur du stator, se trouve le rotor, nommé également induit, et un composant rotatif, lequel sur les moteurs DC conventionnels est aussi fabriqué à partir d'un électroaimant. Les moteurs à courant continu, composés tels que décrits précédemment d'un stator et d'un rotor, sont appelés induits intérieurs, alors qu'une structure inversée s'apparente à un induit extérieur. [29]

Les bobines du rotor sont branchées à l'aide d'un collecteur. Celui-ci sert d'inverseur de polarité et abrite les balais, lesquels ont une forme de brosse et sont fabriqués dans un matériau conducteur. Les matériaux privilégiés sont le graphite et selon l'application en question du moteur — des matériaux enrichis de métal. Les balais sont cruciaux dans le fonctionnement du moteur DC conventionnel. Car lorsque le courant continu circule au travers de la bobine du rotor, c'est-à-dire à travers du rotor, celui-ci se transforme en électroaimant et produit des forces magnétiques grâce aux propriétés du stator. Comme les pôles de même polarité se repoussent et les pôles contraires s'attirent, cela induit un mouvement de rotation du rotor qui se terminerait en principe dans une zone neutre. Pour garantir toutefois une rotation continue, une inversion périodique du sens du courant est nécessaire. À l'aide des balais, le collecteur du moteur à courant continu effectue à intervalles réguliers cette inversion de polarité.

La différence de construction des moteurs DC à balais s'étend par ailleurs selon le type de commutation de l'induit et du bobinage du stator. Sur la machine à courant continu, aussi appelée moteur en série, la bobine d'excitation et la bobine du rotor sont montées en série, c'est-à-dire couplées les unes après les autres, ce qui crée la base de l'alimentation de courant alternatif. Le moteur shunt, sur lequel les deux bobines sont couplées en parallèle, est un équivalent. Le moteur compound quant à lui combine la machine à courant continu et le moteur shunt. Selon sa conception, il permet différents modes de fonctionnement et profite des avantages des deux types de moteurs.

IV. 7 Calcul de puissance pour le moteur DC

Le moteur utilisé dans ce travail est un moteur à courant continu. Voici quelque exemple de calcule qui sont appliqué pour réaliser un ascenseur avec des paramètres réel.

IV.7.1 Exemple :

La puissance, le travail et la force sont calculés comme suit :
Puissance = Travail / Temps
Travail = Force × Distance Force, F = mg
Ce qui implique : Puissance = Force × $distance$/Temps

Puissance = Force × Vitesse

En va prendre les coordonnées suivantes pour avoir une meilleure idée,Si le poids de la cabine vide = 100 kg donc Le Contrepoids = 100 kg Et en supposent que 10 personnes de 65 kg chacune = 650 kg

Pour un fonctionnement à vitesse constante de 1 m/s ou 60 m/min la Force = 650 × 9,8 = 6370 N

Puissance = 6370N × 1m/s = 6370W
(Avec 1cv = 745 W)Donc 6370 W sera
6370 / 745 = 8.55cv

Cela peut être approximatif à 9hp ou 6705 WVitesse de rotation = 1500 tr/min

Le couple du moteur peut être calculé en utilisant l'expression suivante :

Couple du moteur (Nm) = puissance en $Watts$ / $2\pi\omega$ = 6705 / $2\pi x$ 1500 = 31,5 Nm

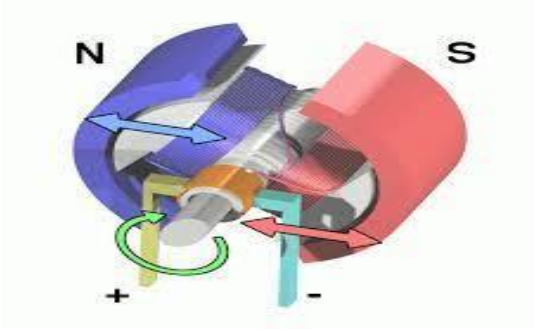


Figure IV. 12 rotation de moteur

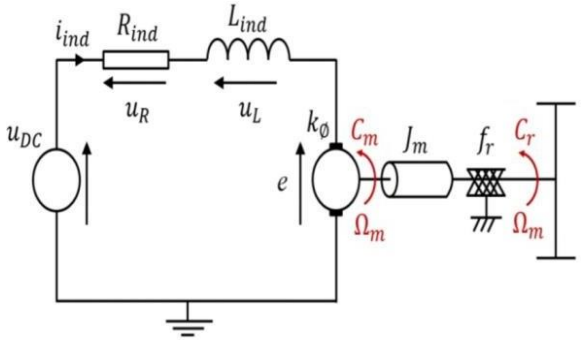


Figure IV.13 Schéma électrique de moteur dc

IV.8 récepteur laser

Figure IV.14 : laser

Un laser est un appareil qui émet de la lumière grâce à un processus d'amplification optique basé sur l'émission stimulée de rayonnement électromagnétique. Le mot "laser" est un acronyme pour "amplification de la lumière par émission stimulée de rayonnement.

À ce jour, on compte cinq principaux types de lasers sur le marché : le laser CO^2, le laser fibre, le laser vert, le laser UV, et le laser à cristal (YAG ou VANADATE). Ces lasers se différencient par la longueur d'onde de leur faisceau, ainsi que par la nature des matériaux qu'ils peuvent marquer.

IV.9 .Interfaçage du pilote de moteur pas à pas TB6600 avec Arduino

IV.9. 1 Caractéristiques du pilote de moteur pas à pas TB6600

Le TB6600 est un pilote de moteur pas à pas professionnel facile à utiliser dont vous pouvez régler les micro-pas. Ce module pourrait contrôler un moteur pas à pas biphasé. Une caractéristique clé de ce module est que vous pouvez modifier les paramètres de micro-pas à l'aide des commutateurs intégrés sur le pilote.

Ces modules ont plusieurs fonctions de sécurité comme suit :

- Protection contre les surintensités
- arrêt en cas de sous-tension
- protection contre la surchauffe

Il existe 2 types de ce module : 4 ampères et 4,5 ampères. Les deux ont des fonctions similaires. Vous pouvez voir ces modules dans l'image ci-dessous.

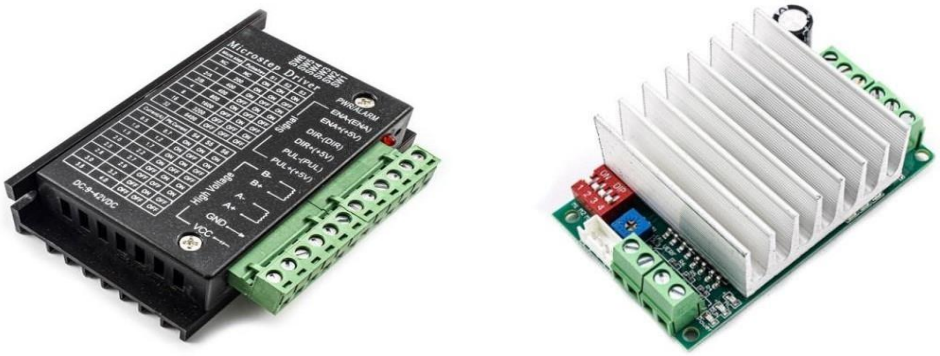


Figure IV 15 pilote de moteur pas à pas TB6600

IV.9. 2 Brochage du pilote de moteur pas à pas TB6600

Ce module a les broches suivantes :

a) Haute tension

- **VCC** : Alimentation moteur – 9-42V pour le type 4A et max 32V pour le type 4.5A
- **GND** : Masse
- **A+** : Broche positive de la bobine 1
- **A- :** Broche négative de la bobine 1
- **B+ :** Broche positive de la bobine 2
- **B- : Broche** négative de la bobine 2

b) Signal

- **PUL(CLK) :** Broches pour contrôler les pas de rotation
- **DIR(CW) :** Broches pour contrôler les pas de rotation
- **ENA :** Activer la broche du pilote
- **5V** : Tension – 5V

Noter :

Il existe 2 façons de commander les broches PUL, DIR et ENA en type 4A :

1. Connectez les broches négatives à la masse et effectuez le contrôle par les broches positives. (Actif -ÉLEVÉ)
2. . Connectez les broches positives à 5 volts et faites le contrôle par les broches négatives. (Actif - BAS)

Noter :

Dans le type 4.5A, parce qu'il y a une broche 5V dans les broches de contrôle, les autres broches sont activées avec une tension BASSE.

Commutateurs de commande de pilote de moteur pas à pas TB6600

Ces commutateurs sont utilisés pour contrôler la résolution des micropas et limiter le courant du pilote.

Vous pouvez modifier la résolution du micropas de pas complet à 1/32 pas en basculant les commutateurs S1 à S3.

S1	S2	S3	Microstep resolution
ON	ON	ON	NC
ON	ON	OFF	Full step
ON	OFF	ON	1/2 step
OFF	ON	ON	1/2 step
ON	OFF	OFF	1/4 step
OFF	ON	OFF	1/8 step
OFF	OFF	ON	1/16 step
OFF	OFF	OFF	1/32 step

Figure IV .16. Commutateurs de commande de pilote

Noter :

Il n'y a pas de pas de 1/32 dans le type 4.5A.

Noter :

En type 4.5A, vous pouvez régler et limiter le courant par le potentiomètre.

Par le voltage de référence : C'est une méthode un peu plus complexe, mais plus recommandable. Nous devons d'abord déterminer la tension de référence requise à l'aide de la formule :

$Vref = I_{max} \cdot 8 \cdot Rs$

Où I_{max} est le courant maximal auquel le moteur sera alimenté (généralement au maximum 90 % du maximum spécifié par le fabricant) et Rs est la résistance de détection du driver.

Pour le régler sur le driver, il suffit de mettre le driver sous tension, de mesurer la tension entre la broche Vref (généralement le potentiomètre lui-même) et une broche de masse (généralement la broche d'alimentation) et drégler la valeur appropriée à l'aide du potentiomètre.

Et voici le Bloc d'Alimentation 12V, 15A

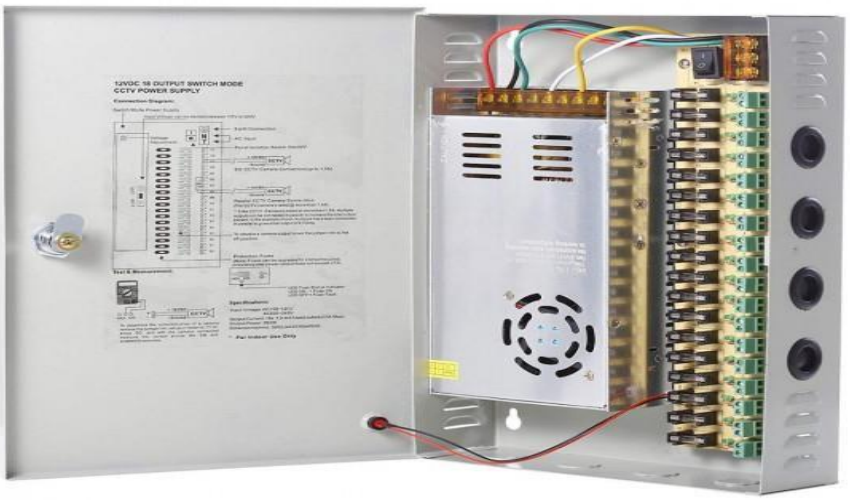


Figure .IV .17 Bloc d'Alimentation

CARACTERISTIQUES : BLOC D'ALIMENTATION 12V, 15A

Tension d'entrée : AC110 / 220V 50/60Hz

Canaux de sortie : 18 canaux

Tension de sortie : DC12V

Courant de sortie : 15A Max

Puissance de sortie : 120W

Taille : 205* 235 * 53mm (L * W * H)

Poids : 1,75 kg

IV.10 carte Arduino Méga 2560

La MEGA 2560 est conçu pour des projets plus complexes. Avec 54 broches E /S dont 14 PWM, 16 entrées analogiques et d'un plus grand espace pour les croquis. Cela donne à notre projet beaucoup d'espace et d'opportunités. En effet la carte Arduino Méga 2560 offre toutes les fonctionnalités de la carte UNO, mais avec des fonctionnalités supplémentaires. [30]

Figure IV. 18 carte Arduino Méga 2560

Tableau IV .2: tableau regroupant les caractéristiques de la Mega 2560**.**

Les caractéristiques de la Méga 2560	
Version : Rev.3 ;	Alimentation : via port USB ou 7 à12 V
Microprocesseur : ATMega2560 ;	Mémoire flash : 256 KB ;
Mémoire SRAM : 8 KB ;	Mémoire EEPROM : 4 KB ;
54 broches d'E/S dont 14 PWM ;	16 entrées analogiques 10 bits ;
Intensité par E/S : 40 mA ;	Cadencement : 16 MHz ;
3 ports séries ;	Bus 12C et SPI ;
Gestion des interruptions ;	Fiche USB B ;
Dimensions : 107 x 53 x15 mm	

IV.11 Branchement de la machine :

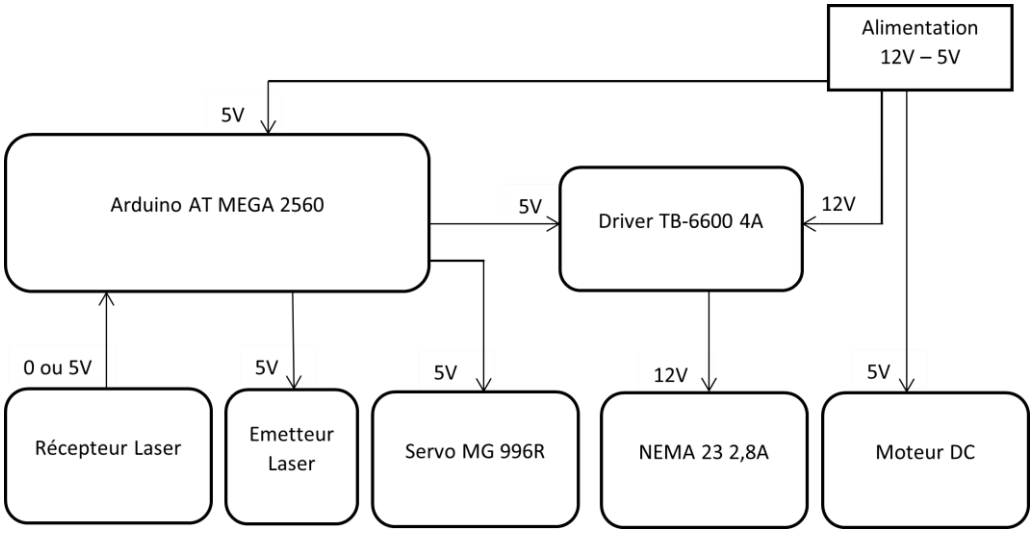


Figure IV.19 branchement de la machine

IV. 12 Fonctionnement de la machine :

- Mise en marche de la machine, Allumage de l'alimentation
- Rotation du moteur DC, Allumage du Tapis
- Alimentation de la machine en colis
- Détection laser de la taille du colis (3 niveaux soit 7cm – 14cm- 21cm)
- Triage des colis automatiquement par l'Arduino
- Sortie du colis à travers le bloc de triage (servo MG996R + moteur DC) dans la direction appropriée

IV. 13 Câblage de la machine et schéma de la machine

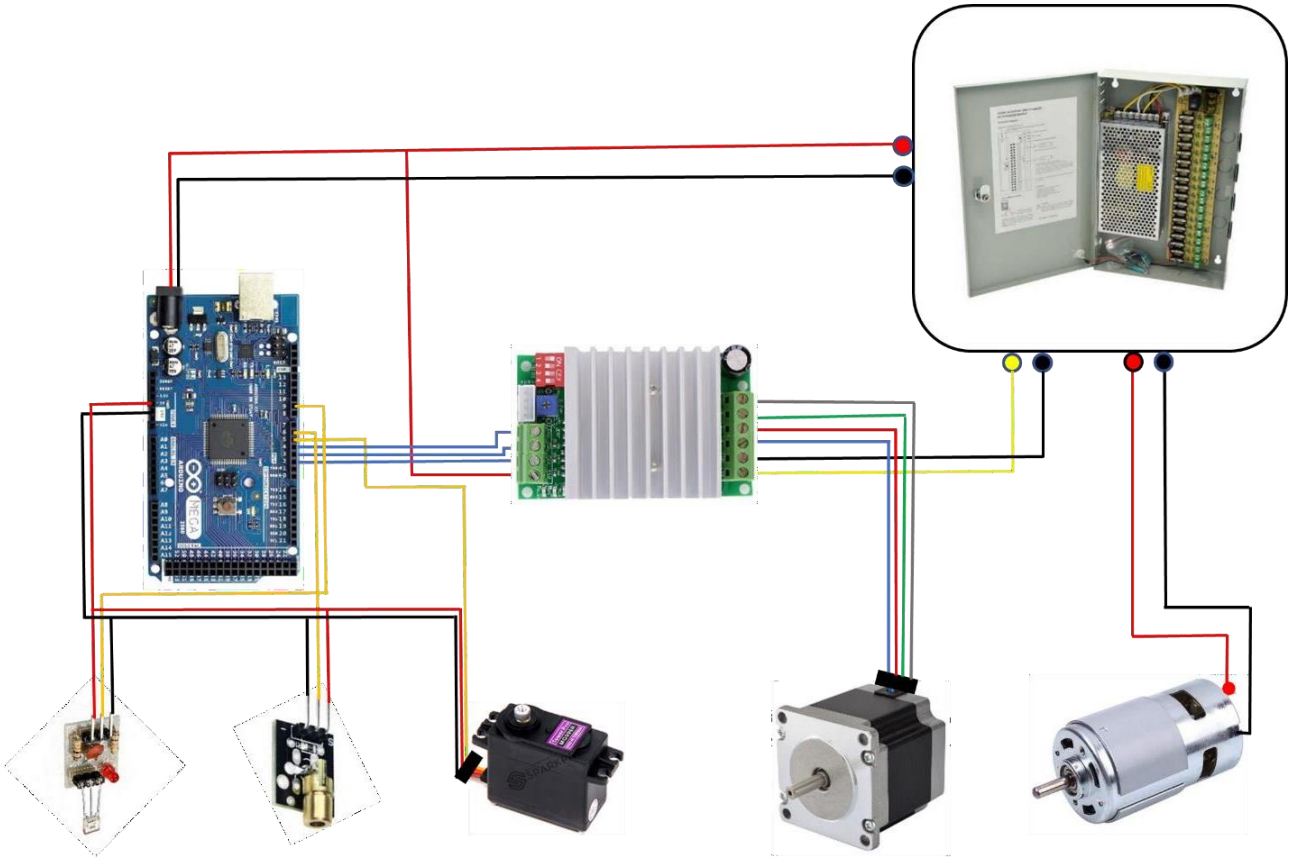


Figure IV.20 Câblage de la machine

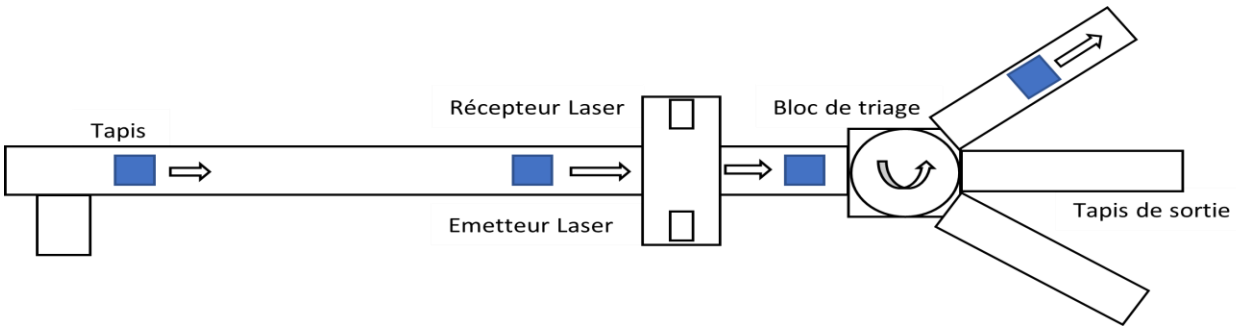


Figure IV.21 *Schéma de la machine*

IV. 14 Aperçu du prototype:

IV. 14. 1 **Composants:**

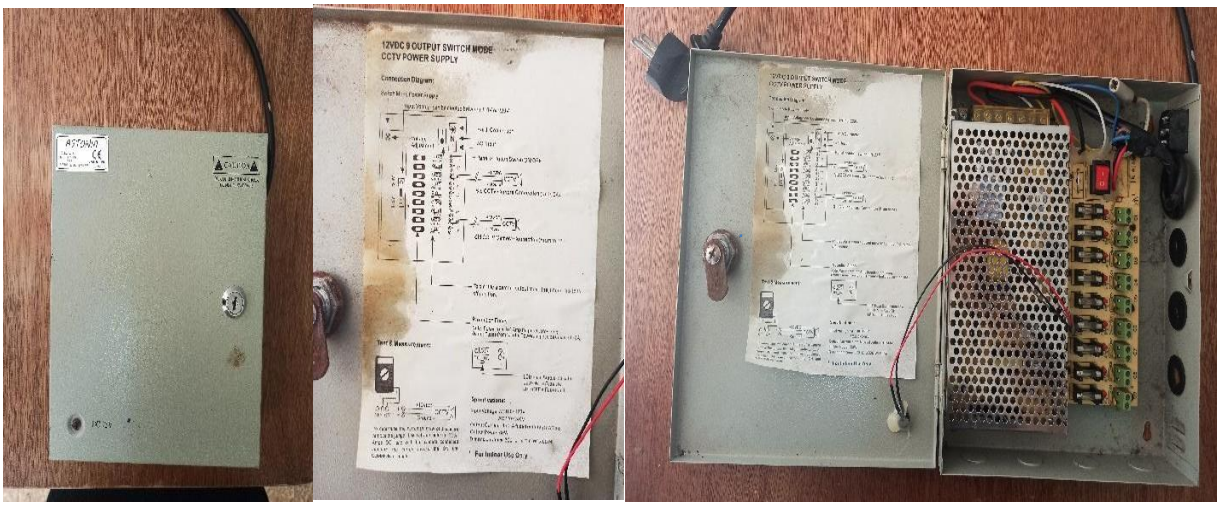


Figure IV .22 Alimentation

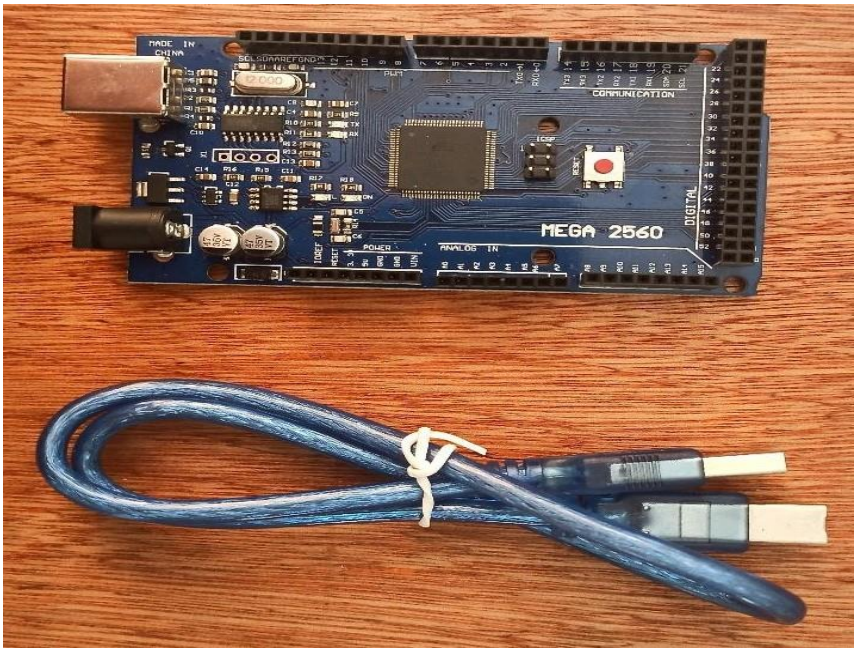


Figure IV.23 Arduino AT MEGA

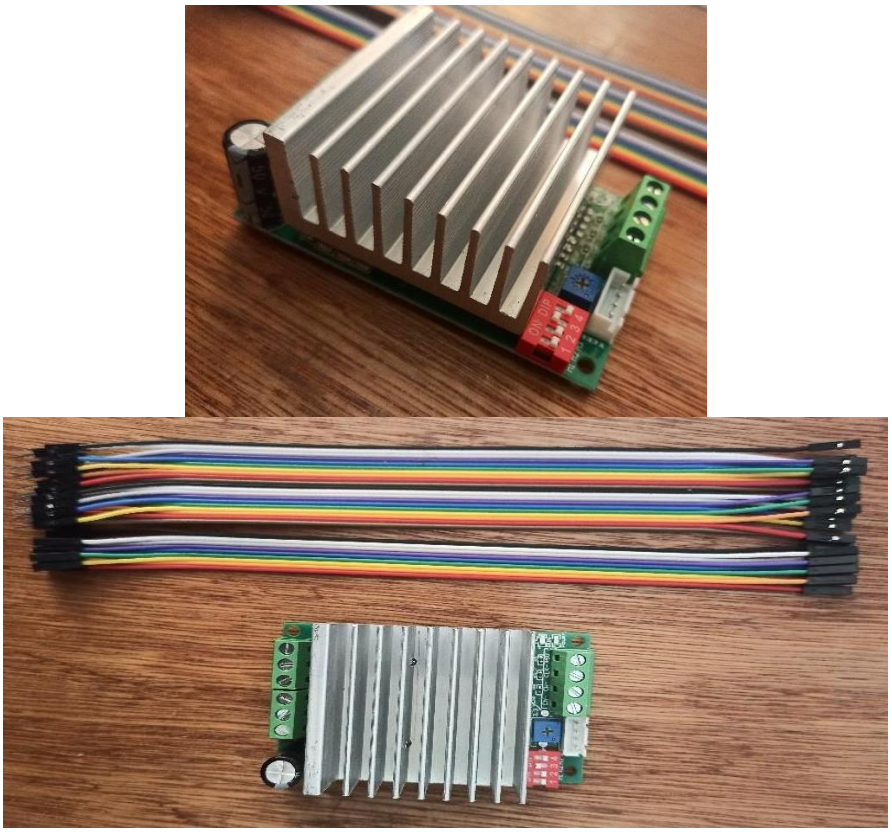


Figure.IV 24 Driver Nema

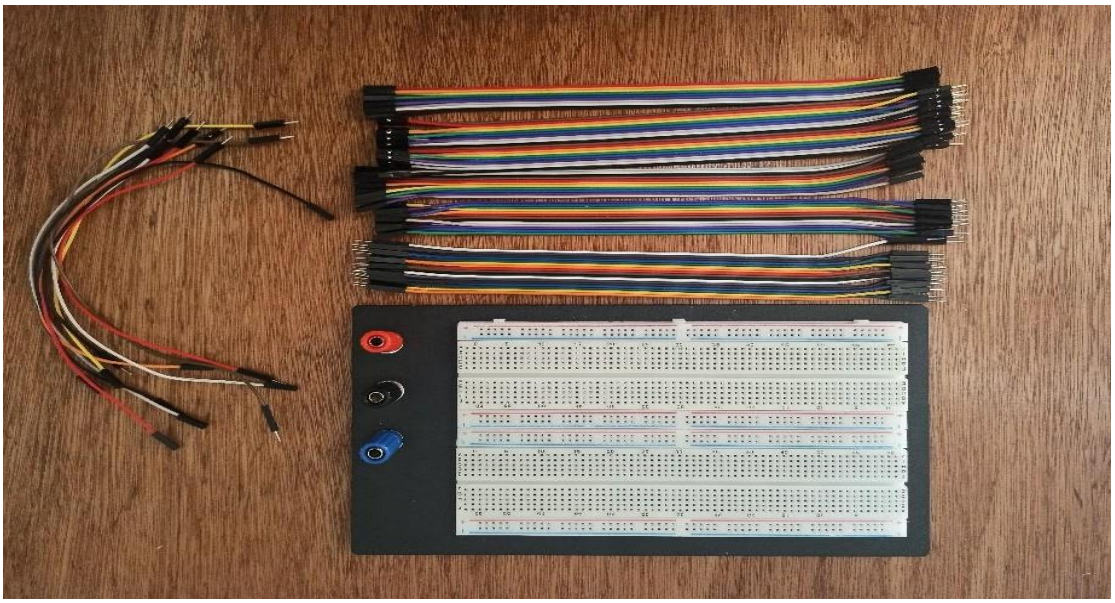


Figure IV.25 Plaque d'essaies et cables

Figure IV.26 Capteurs laser

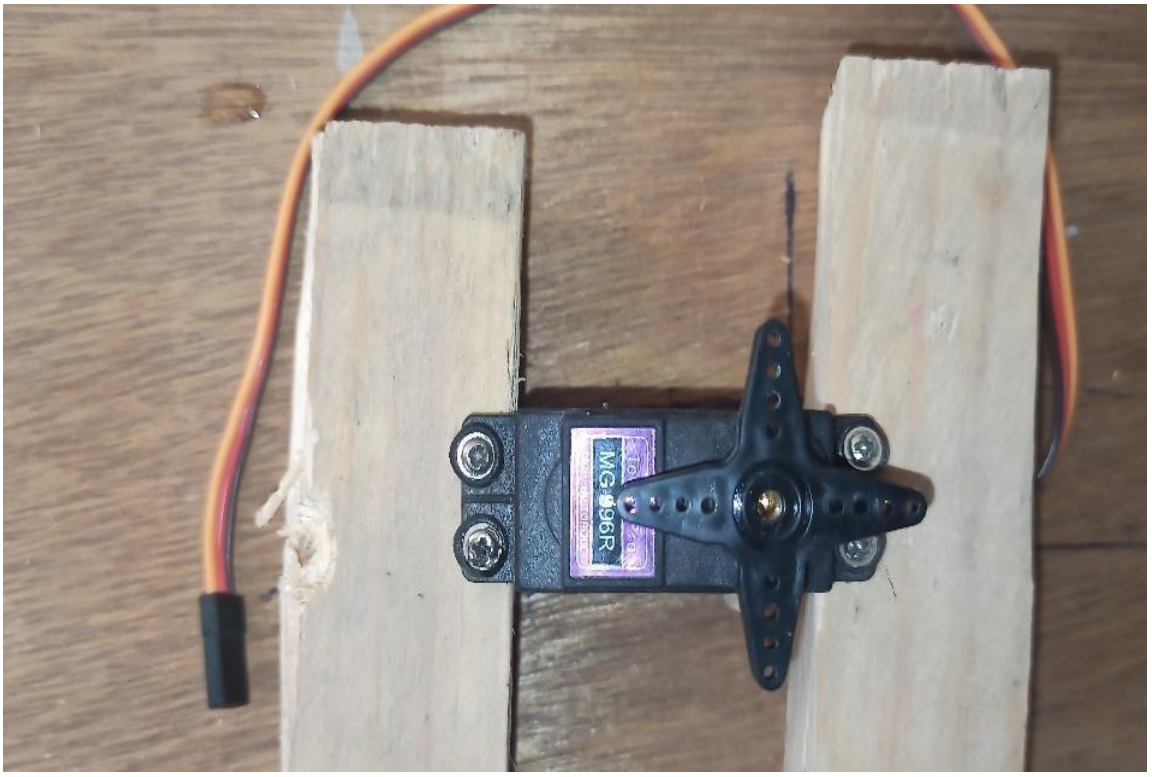


Figure IV.27 Servo moteur

Figure IV.28 Moteur pas à pas Nema

IV.14 .2 **Assemblage et structure du prototype:**

Figure IV.29 Tapis de transport principale

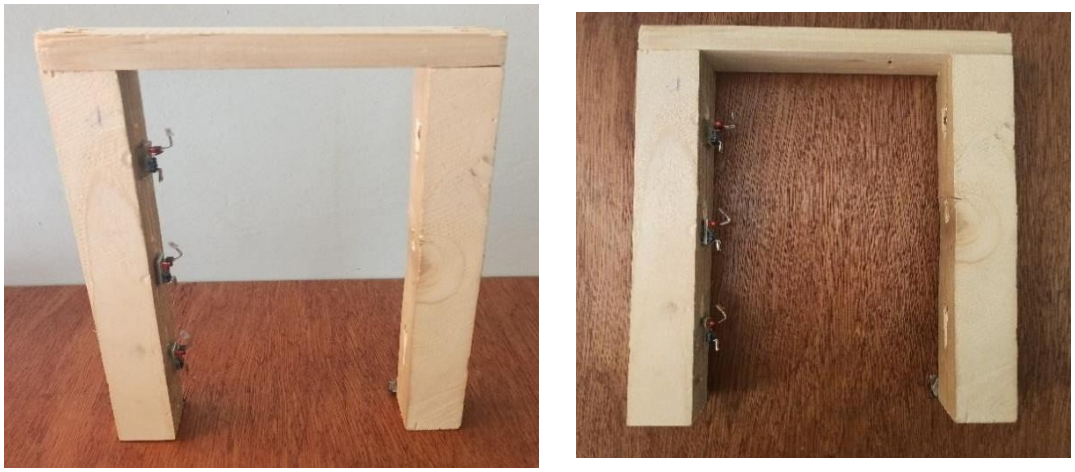


Figure IV.30 Arche de détection (support des capteurs lasers)

Figure IV .31 Bloc de triage rotatif

IV.15 Le programme principale

```
      #include "Moteur_Servo.h"     // On inclut les fichiers en tête
#include "Moteur_pas_a_pas.h"
#include "emetteur_recepteur.h"

void setup () {
    digitalWrite(EN, LOW); // On met la pin EN en LOW (0V) afin que le moteur
                           // soit contrôlé par l'Arduino
   Direction(1); // On sélectionne le sens horaire
}
void loop () {
   PAS(vdr); // On met en marche le moteur
   Etat Val_1 = digitalRead(detecteur_1); // On enregistre l'état du recepteur
                                          // dans la variable "Val" équivalente
   Etat Val_2 = digitalRead(detecteur_2);
   Etat Val_3 = digitalRead(detecteur_3);
   Tri(); // On fait le triage en faisant tourner le Servo-moteur
```

Le programme principal inclut trois sous-programmes afin de faciliter la programmation et la correction d'éventuelle dysfonctionnement, chaque sous-programme est responsable d'une partie de la machine, on cite : Le tapis, le détecteur-laser, le bloc de triage

Les fichiers en tête sont détaillés dans l'annexe A, B et C afin de ne pas surcharger cette partie du chapitre

IV.15 Conclusions :

Dans ce dernier chapitre, en première approche, nous avons abordé les différentes étapes de mise en place d'une carte de contrôle. Nous avons ensuite effectué un test pour vérifier le fonctionnement et les performances de notre modèle.

Il est intéressant et sa réalisation nécessite de nombreux domaines technologiques, de plus, c'est un moyen de production très utilisé et plus répandu, remplissant un rôle important pour la maintenance en améliorant la sécurité des utilisateurs et la disponibilité des appareils. J'ai eu l'occasion d'étudier ce modèle qui m'a été très utile, mis à part mes difficultés à me procurer les composants nécessaires pour me rendre compte que nous voulions qu'il soit complet.

Conclusion générale

Grâce à notre projet, nous avons pu on apprendre davantage sur la vie industrielle d'une manière qui est à la mesure du développement atteint par la science avancée. Machine de tri programmable a base d'Arduino AT Mega 2560.

Ce qui nous a permis de programmer le projet nous garantit la qualité, la rapidité de production et la classification des produits.

Où nous avons transféré le programme dans l'Arduino. L'objectif principal est de manipuler le langage de programmation afin de construire une application capable de transmettre les commandes de l'utilisateur directement aux moteurs et équipements électriques existants. Comprendre les différents protocoles de communication et extraire le port série (USB) comme port pour transférer le programme sur la carte Arduino.

Nous avons tout accompli selon des spécifications de haute précision et utilisé des outils modernes pour effectuer un travail avancé qui suit le rythme des progrès de manière intelligente.

Ce travail est une combinaison de nos énergies et de nos efforts afin de le réaliser sur le terrain.

En fin de compte, nous voyons que ce travail que nous avons fait a commencé à se répandre très rapidement dans le domaine industriel car il réduit la main-d'œuvre et réalise de gros profits en peu de temps, car les pays développés sont ainsi devenus toutes leurs usines et se sont répandus même en Algérie et dans les pays africains en raison de leur forte croissance économique.

Nous conseillons à toutes nos usines de travailler avec cette machine car c'est le titre du progrès et le fruit de l'invention

ANNEXES

ANNEXE A (MOTEUR PAS A PAS)

(fichiers en tête)

```
#ifndef Moteur_pas_a_pas
#define Moteur_pas_a_pas

/*
  Ce programme a pour but de contrôler un moteur pas à pas
  avec le driver TB6600 4amps sans utiliser la bibliotheque
  "stepper.h"
*/

#define STEP 2 // On renomme les pins
#define DIR 3 // de manière
#define EN 4 // plus lisible

int vdr = 5000; // on crée la variable qui va déterminer
                // la vitesse de rotation du moteur (30 tr/min)

void PAS (int x);          // On déclare les noms des fonctions
void Direction (bool sens); // qui seront utilisées plus tard

pinMode(STEP, OUTPUT); // On initialise
pinMode(DIR, OUTPUT); // les pins comme
pinMode(EN, OUTPUT); // sorties
```

```
/*
  Envoie un signal au driver sur la pin DIR
  pour définir le sens de rotation.
  Si sens = 1 alors le moteur tourne dans le sens horaire.
  Si sens = 0 alors le moteur tourne dans le sens antihoraire
  (Si ce n'est pas le cas, on inverse le brachement du moteur)
*/

void Direction (bool sens){
   if (sens == 1)
      digitalWrite(DIR, HIGH);
   else
      digitalWrite(DIR, LOW);
  }
/*
  Fais avancer le moteur de 1 pas
*/
void PAS(int x){
      digitalWrite(STEP, HIGH);
      delayMicroseconds(x); // Un délai de 5 millisecondes
                            // est requis par le driver
      digitalWrite(STEP, LOW);
      delayMicroseconds(x); // Ce délai est là pour s'assurer que pour 2 appels
                            // consécutifs de Step, le délai entre les 2
                           // appels ne soit pas trop court pour le driver.
  }

#endif
```

ANNEXE B (SERVOMOTEUR)

(fichiers en tête)

```
#ifndef Moteur_Servo
#define Moteur_Servo

#define Imp 5 // On renomme la pin

typedef enum { // On définit les trois angles usuelles.
 D = 0,      // 0 pour la droite
 M = 60,     // 60 pour le milieu
 G = 120  // 120 pour la gauche
 }Ang;

void Serv(char x); // On déclare le nom de la fonction

void Serv(Ang x) // NB : Le cerveau moteur n'est capable de lire qu'une seule et unique
            // impulsion chaque 20ms (0,02s) soit 1/0,02 = 50HZ
{
   if(x == 0)
   {
      digitalWrite(Imp, HIGH); // on donne une impulsion de largeur 0,001s,
      delayMicroseconds(1000); // puis on cree un delais ou la pin Imp
      digitalWrite(Imp, LOW); // est en low de duree de 0,019s
      delayMicroseconds(19000);
   }
   else if(x == 60)
   {
```

```
        digitalWrite(Imp, HIGH);
        delayMicroseconds(1500);
        digitalWrite(Imp, LOW);
        delayMicroseconds(18500);
    }
    else
    {
        digitalWrite(Imp, HIGH);
        delayMicroseconds(2000);
        digitalWrite(Imp, LOW);
        delayMicroseconds(18000);
    }
}

#endif
```

ANNEXE C (emetteur_Récepteur)

(fichiers en tête)

```
#ifndef emetteur_recepteur
#define emetteur_recepteur

#define laser_1  6 // On renomme les pins des détecteurs lasers
#define laser_2 7
#define laser_3  8
#define detecteur_1  9
#define detecteur_2  10
#define detecteur_3  11

typedef enum { // On crée une variable d'état
   faux,     // pour présence d'un colis
   vrai     // pour abscence de colis
}Etat;

void Tri(void); // On crée la fonction qui fait le triage

   pinMode(laser_1, OUTPUT);  // On déclare les transmetteurs comme sorties
   pinMode(laser_2, OUTPUT);
   pinMode(laser_3, OUTPUT);
   pinMode(detecteur_1, INPUT); // On déclare les recepteurs comme entrées
   pinMode(detecteur_2, INPUT);
   pinMode(detecteur_3, INPUT);

   digitalWrite(laser_1, HIGH);  // On active les transmetteurs
   digitalWrite(laser_2, HIGH);
   digitalWrite(laser_3, HIGH);

void Tri(void) {
    // On met en place les conditions
    If (Val_1 == faux && Val_2 == vrai && Val_3 == vrai){ // Si il n'y a que le recepteur 1 qui
est traversé
                                                          // alors le servo tourne à droite
        Serv(D);
```

```
        delay(10000); // le temps nécessaire au clois afin d'être transféré
        }
        If (Val_1 == faux && Val_2 == faux && Val_3 == vrai) {
                    Serv(M);
             delay(10000);
        }
            If (Val_1 == faux && Val_2 == faux && Val_3 == faux) {
                Serv(G);
                delay(10000);
            }
}
#endif
```

Résumé

Nous avons réalisé un automate programmable industriel à base d'un Arduino, cet automate que nous avons conçu nous a permis de bien comprendre les concepts de l'automatisation.

Dans ce mémoire nous avons cité quelques généralités sur les automates programmables industriels et on a fait une étude sur la carte Arduino, puis On a conçu et des différents exemples à l'aide de IDE

ensuite on a passé aux tests sur les plaques d'essai et à la réalisation final de notre système machine de tri

ملخص

وقد سمحت لنا وحدة التحكم هذه التي أونتوماتيكية Arduino لقد أنجزنا وحدة تحكم مطورة قابلة للبرمجة تعتمد على مفاهيم

مبنية على الأمثلة

Arduino هي هذه الذاكرة ، اقتبسنا بعض العموميات حول الآلات الصناعية القابلة للبرمجة وقمنا بإجراء دراسة على بطاقة

ثم قمنا بتصميم ومحاكاة أمثلة مختلفة باستخدام

IDE

ثم انتقلنا إلى الاختبارات على لوحات الاختبار والتحقيق النهائي لنظام آلة الفرز لدينا

Summariy

We have produced a programmable logic controller based on Arduino, and this controller we designed has allowed us to understand the concepts of

Automation.

In this memory, we quote some generalities about industrial programmable machines and do an Arduino card study, then design and simulate different examples using IDE

Then we moved on to the tests on test boards and the final investigation of our sorting machine system

Bibliographiques

[1] https://www.clicours.com/structure-dun-systeme-industriel-automatisee/

[2] Programmable Logic Controller '*/ibrahim6060.weebly.com/introduction.html*'

[3] Automates programmables industriels Mr Philippe LE BRUN (*Décembre 1999*)

[4] ANDRIANIRINA Mamy Harifetra Eddy '*Etudes de liaison dans un système d'équipements interconnectés intégrant en particulier un API (Automate Programmable Industriel) dans un Réseau Local Industriel (RLI)*' Ecole Supérieure polytechnique université d'Antananarivo-mémoire 2017.

[5] Etude des automates programmables industriels (API) Mr ROIZOT Sébastien

[|6]https://www.technologuepro.com/cours-automate-programmable-industriel/Les-automates-programmables-industriels-API.htm

[7] https://www.startimes.com/f.aspx?t=36114937

[8] Mémoire de Master "Système de Contrôle Distribué (DCS) avec l'exploitation de l'automate programmable AC800 F (ABB) ", Université Mohamed Khider Biskra , Juin 2012.

[9] mcourscom@gmail.com / Résumé de Théorie et Guide de travaux pratiques / UTILISATION DEL'AUTOMATE PROGRAMMABLE

[10]http://fasoeducation.net/espace_eleves/secondaire/eftp/bac_technologique/automates_programmables_industriels/co/grainDomaine_utilisation.html

[11]https://www.clicours.com/cours-electromecanique-utilisation-de-lautomate-programmable/

[12]https://public.iutenligne.net/informatique/informatique-industrielle/deprez_maillefert/Autom/2Pupitre/programmation_en_langage_ladder.html

[13] https://m.nearbyme.io/search/?search_term=LE LANGAGE LADDER &brand=gc1

[14] http://www.est-usmba.ac.ma/GRAFCET/co/module_cours_grafcet_2.html

[15] https://www.technologuepro.com/cours-automate-programmable-industriel/Cours-Grafcet-notions-de-base.htm Cours GRAFCET Mr Robert Valette

[16] Par Astalaseven , Eskimon et olyte Arduino 'Pour bien commencer en électronique et en programmation'

[17] https://www.positron-libre.com/electronique/arduino/arduino.php Arduino Uno Rev3 A000066

[18] https://www.gotronic.fr/art-carte-arduino-uno-12420.htm

[19] https://www.gotronic.fr/art-carte-arduino-uno-12420.htm

[20] Les différents type d'Arduino (2019) *'https://www.academia.edu/39339945'*

[21]https://www.gotronic.fr/art-carte-arduino-mega-2560-12421.htm

[22] https://www.developpez.com/actu/166397/Quels-langages-de-programma
[23] https://www.editions-eni.f/open › mediabook

[24] https://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.MaterielUno

[25] Référence Arduino français ' *http://www.mon-club* elec.fr/pmwiki_reference_arduino/ pmwiki.php?n=Main.MaterielMega2560 '

[26]https://www.researchgate.net/figure/Carte-Arduino-Mega-2560-Le-microcontroleur-est-preprogramme-avec-un-bootloader-de_fig3_310759430

[27] https://www.omega.fr/prodinfo/Moteur-pas-a-pas.html

[28] https://eskimon.fr/tuto-arduino-602-un-moteur-qui-a-de-la-t%C3%AAte-le-servomoteur

[29] https://harmonicdrive.de/fr/glossaire/moteur-dc

[30] Référence Arduino français ' *http://www.mon-club* elec.fr/ pmwiki_reference_arduino/ pmwiki.php?n=Main.MaterielMega2560 '