REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A. MIRA – BEJAIA

Faculté de Technologie

Département de Génie électrique

# *MEMOIRE DE FIN D'ETUDE*

En vue de l'obtention du diplôme de Master en Génie électrique

**Option**

*Automatismes industriels*

Thème

## Plateforme d'automatisation et de supervision basée sur Arduino, Proteus et Easybuilder pro avec applications diverses

**Présenté par:**

**Mr. KENNOUCHE Samir**

**Mr. KADRI Nabil**

**Encadré par :**

**Mr. A. MELAHI**

**Membres de jury :**

**Mr. N.TAIB**

**Mr. K.DJERMOUNI**

*Année universitaire 2019-2020*

# Dédicace .....

Je dédie ce modeste travail pour mes chers parents, un grand merci pour eux de m'avoir soutenue durant tout ce temps et de m'avoir orienté et guider dans le droit chemin, mon grand frère Kousseila, ma petite sœur sans oublier mes cousins et cousines Slimane, Djamel, Tarik, Lynda et ma précieuse famille et la famille KADRI en général, bien sure sans oublier mon cher binôme Samir, mes meilleurs amis en général.

NABIL

Mes dédicaces sont destinés particulièrement à mes très chers parents, à ma sœur et deux grands frères tous mes oncles et tantes et grand-mère ; cousins et cousines leurs famille et à toute la famille KENNOUCHE, sans oublier mes amis, toute personne qui me connait ou ma aider dans mon parcours pédagogique ou dans la vie quotidiennes .à l'âme de toute personne qui nous a laissé particulièrement ma grand-mère yemma Zineb.

SAMIR

# Remerciements .....

Par cette occasion, on remercie toutes personnes venue en aide durant la réalisation de notre modeste travail notamment notre cher promoteur Mr A .MELAHI, et tout les autres enseignants qui nous ont aidé à réussir notre cursus universitaire à l'université de A/Mira de Bejaia.

Sans oublier les membres de jury, un grand merci pour :

Mr N.TAIB et Mr K.DJERMOUNI.

# TABLE DES MATIERES……

## CHAPITRE 3: EXEMPLES D'APPLICATION



## CHAPITRE 4 :PROGRAMMATION D'UN BRAS MANIPULATEUR PAR ARDUINO

# Liste des figures .....

# Liste des tableaux .....

# Liste des symboles et d'abréviations .....

**SAP** : Système Automatisé de Production

**PO** : Partie opérative

**PC** : Partie commande

**PR** : Partie dialogue (relationnelle)

**API** : Automate programmable industriel

**AFNOR** : Association Française de Normalisation

**MAS (SMA) :** Système d'articulation mécanique

**D.D.L** : Degré de liberté

**FBD :** Function Block Diagram ( Logigrammes )

**IL** : Instruction List

**ST** : Structured Text

**GRAFCET** : Graphe Fonctionnel de Commande Etape Transition

**PC** : Personal Computer

**E/S** : Entrées et sorties

**MTU :**Master Terminal Unit

**IHM ou HMI ou MMI :** Human Machine Interface

**RTU** : Remote Terminal Unit

**IDE :** Integrated développement environnement

**LED :** Light-emitting Diode. (diode électroluminescente)

**OSI :** Open Systems Interconnection)

**RLI :** Réseau local industriel

**TCP/IP :** Transmission Control Protocol/Internet Protocol

**ASCII :** American Standard Code for Information Interchange

**COM :** Port série

**USB :** Universal Serial Bus (Bus série universel).

**WIFI :** Wireless Fidelity (fidélité sans fil)

**VSPD :** Virtual Serial Port Driver

**Carte SD :** Secure Digital

**CAO** : Construction assistée par ordinateur

**PWM:** Pulse Width Modulation.

**PCB** : Circuits imprimé

**PLC :** Programmable Logic Controller,

**DCY :** Départ cycle

**HEX:** Hexadécimale

**Ms:** Millisecondes

**ID:** Identifiant

**MHz:** Méga hertz

**UART:** Universal Asynchronous Receiver Transmitter

**GND :** Ground ( masse )

**Vcc:** Voltage common collector

**kΩ :** Kilos ohm

**Ω:** Ohm

**TOR :** Tout ou rien

**MLI:** Modulation en Largeurs d'Impulsions

**V :** Volt

**Moteur CC:** Moteur à courant continue

**εr :** Constante diélectrique

**PET:** Poly (téréphtalate d'éthylène)

# Introduction générale :

Auparavant l'être humain effectue ses tâches d'une manière manuelle voir traditionnelle qui a atteint ses limites dans quelques domaines ; sur tout en terme de rapidité et de précision ; cela provoquais l'insécurité dans son environnement de travail mais aussi affaiblit la rentabilité des entreprises en influençant sur leurs chiffres d'affaires.

C'est pourquoi l'homme s'est opté vers l'automatisation en fondant des usines intelligentes afin de réussir une communication continue et instantanée entre les différents outils et postes de travail intégrés dans les chaînes de production et d'approvisionnement ,qui est le concept de l'industrie 3.0 .

L'objet de notre étude est de rechercher comment pouvoir automatiser et superviser une chaîne de production et de manutention du verre en lui intégrant un robot manipulateur par l'intermédiaire des outils Software et Hardware .

En étant des personnes passionnées par la programmation et l'innovation technologique, on a décidé donc d'approfondir nos recherches sur ce sujet, en fixant un plan qui sont réparties en plusieurs chapitres.

Nous verrons dans un premier temps qu'il est nécessaire d'avoir des connaissances générales sur l'automatisation et la structure des systèmes automatisés de production, mais aussi la robotique et la supervision industrielle.

Nous devrons également chercher des informations sur les différents logiciels et outils envisagés pour atteindre l'objectif tracé puis les appliquer sur des petites applications afin de terminer par la suite avec une grande où on exploitera toutes les notions acquises dans l'automatisme, la robotique, la supervision et la programmation.

# CHAPITRE 1 :

## GENERALITES SUR L'AUTOMATISATION ET LA SUPERVISION

# 1.1.Introduction

Grace à l'avancement technologique, l'homme fait à présent un grand pas vers l'avant en bénéficiant d'une grande sécurité dans l'environnement industriel là où il travaille , en remplaçant les procédures manuelles par des procédés automatiques, notamment pour viser le facteur économique qui est devenu primordial pour les entreprises, mais aussi pour la qualité et quantité du produit.

De ce fait, pour automatiser un système ou une chaîne de production on fait appel aux robots industriels, qui sont des corps ressemblent à l'être humain en traduisant les mouvements naturels de l'homme par des mouvements à caractère mécanique, électrique et électronique, modifiable en terme de programmation.

Toutefois, pour assurer une bonne gestion de la chaîne de production afin de réduire les risques et les défaillances, un système de supervision doté de logiciels spécifiques est conçu pour ça.

# 1.2.Automatisation

## 1.2.1. Définition de l'automatisation

Automatiser un processus c'est de faire évoluer la sortie rapidement et précisément d'un processus en fonction de son entrée, qui est un progrès technique des dispositifs techniques secondent l'homme, dans ses efforts musculaires et dans son travail intellectuel de surveillance et de contrôle.[1]

## 1.2.2. Objectifs de l'automatisation

Comme objectifs de l'automatisation, on peut citer :

- Les objectifs concernant le personnel comme l'amélioration des conditions de travail et de la sécurité, la suppression des tâches pénibles et répétitives et la réalisation d'opération impossible pour l'humain.

- Les objectifs concernant la production comme l'amélioration de la qualité du produit par rapport au cahier de charges, et sa fiabilité dans le temps, la croissance de la cadence de la production et pouvoir économiser de la matière première et de l'énergie.

- Objectif concernant l'entreprise pour la diminution des couts par réduction des frais de la main-d'œuvre, de matières et d'énergie et l'obtention de plus de bénéfices, plus de compétitivité et de cotes sur le marché.

### 1.2.3. Définition d'un système automatisé

Un système est un ensemble d'éléments permettant de répondre à un besoin qui est la nécessité ou le désir éprouvé par un utilisateur.

Un système automatisé est un ensemble d'éléments en interaction organisés dans un but précis : agir sur la matière d'œuvre entrante afin de lui donner une valeur ajoutée

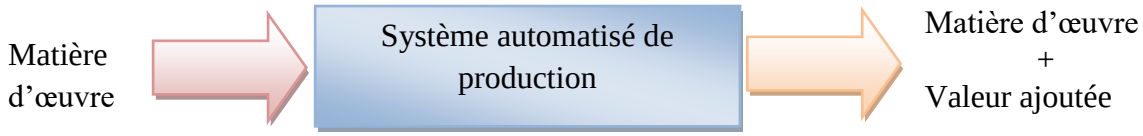



Figure 1.1 Système automatisé de production

« L'automatisation industrielle » est un terme lié directement à la troisième révolution industrielle autrement dit l'industrie 3.0.En celle-ci l'automatisation touche principalement les processus de fabrication, de contrôle de qualité et de manutention. En remplaçant la prise de décision humaine et l'activité manuelle avec l'utilisation d'équipements mécanisés et de commande de programmation logique. [2]

Chaque processus industriel de fabrication est composé d'un ensemble de machines conçu pour la fabrication voir la transformation. Chaque machine ou partie opérative se compose d'actionneurs, ensemble de moteurs, vérins, vannes et autres dispositifs assurant son fonctionnement, pilotés par la partie commande. Cette dernière, élabore les ordres transmis aux actionneurs à partir des informations fournies par la machine au moyen de capteurs et instrumentation; interrupteurs de position, thermostats, manostats et autres dispositifs.

Elle reçoit également des informations transmises par un opérateur en fonctionnement normal, ou un dépanneur en cas de réglage ou de mauvais fonctionnement (défaillance) de la partie commande ou de la partie opérative. Entre celles-ci et l'homme se trouve la partie dialogue qui est la partie relationnelle qui permet à ce dernier de transmettre des informations au moyen de dispositifs adaptés (boutons poussoirs, commutateurs, etc.).

Désormais, il y a un retour d'informations de la partie commande vers l'homme sous des formes compréhensibles (voyant, afficheurs, cadrans, etc.). [3]

### 1.2.4. Description d'un Système Automatisé de Production (SAP)

Un système de production est dit automatisé lorsqu'il peut gérer d'une manière autonome un cycle de travail préétabli qui se décompose en séquences ou en étapes.[4]

### 1.2.5. Structure d'un système automatisé de production (SAP)

Ce système est constitué généralement de trois parties : la partie opérative(PO), la partie commande (PC) et la partie dialogue ou partie relation (PR).

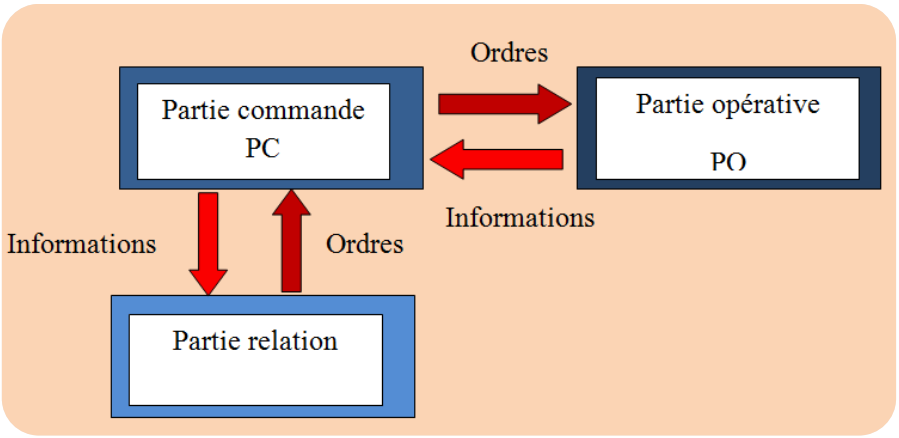



Figure 1.2 Structure d'un système automatisé de production

#### 1.2.5.1. La partie opérative (PO)

Généralement ce sont les parties qui opèrent sur la matière d'œuvre entrante et le produit en effectuant des actions physiques (déplacement, émission de lumière...), mesure des grandeurs physiques (température, humidité, luminosité...). La partie opérative se constitue de :

- **Effecteurs :** sont des dispositifs terminaux agissant directement sur la matière d'œuvre pour lui donner sa valeur ajoutée (outils de coupe, pompes, têtes de soudure, etc.) [3]

- **Les actionneurs** : effectuent la convection d'énergie pour l'adaptation au besoin de la partie opérative. Cette énergie est ensuite consommée par les effecteurs (moteur, vérin, électroaimant, résistance de chauffage, etc.)
- **Les pré-actionneurs** : Le rôle des pré-actionneurs est de distribuer, sur ordre de la P.C., l'énergie utile aux actionneurs. Les pré-actionneurs les plus utilisés sont : les contacteurs pour les moteurs électriques et les distributeurs pour les vérins pneumatiques ou hydrauliques

#### 1.2.5.2. La partie commande (PC)

C'est un sous ensemble de composants (opérateurs logiques, relais électromagnétique) et constituants (microcontrôleur, API, carte.) effectuant le traitement des informations émises par les organes de commande de la PO et PR.

Désormais il y'aura un retour d'information aux pré-actionneurs de la PO et aux composants de signalisation de la PR pour indiquer à l'opérateur l'état et la situation du système.

#### 1.2.5.3. La partie relation (PR)

Elle dispose du pupitre du dialogue, qui est relationnelle entre l'opérateur (l'homme) et la machine, appelé interface homme machine, permettant la sélection des modes de marche, arrêt d'urgence, mettre en /hors énergie de l'installation …

### 1.2.6. Principales technologies utilisées en automatisation

Pour mesurer, détecter et actionner on fait appel à plusieurs technologies parmi lesquelles on cite :

- Technologie électromécanique
- Technologie électrique
- Technologie pneumatique
- Technologie hydraulique

## 1.3. Généralité sur les robots

### 1.3.1. Introduction

D'après l'association Française de Normalisation (AFNOR):

« Un robot est un manipulateur commandé en position, reprogrammable, polyvalent, à plusieurs degrés de libertés, capable de manipuler des matériaux, des pièces, des outils et des dispositifs spécialisés, au cours du mouvements variables et programmés pour l'exécution d'une variété de tâches. Il a souvent l'apparence d'un ou plusieurs bras se terminant par un poignet. Son unité de commande utilise, notamment un dispositif de mémoire et éventuellement de perception et d'adaptation à l'environnement et aux circonstances. Ces machines polyvalentes sont généralement étudiées pour effectuer la même fonction de façon cyclique et peuvent être adaptées à d'autres fonctions sans modification permanente du matériel ».

### 1.3.2. Type des robots

Généralement il existe deux groupes de robots : *les robots fixes* et *les robots mobiles*. Les robots fixes sont souvent utilisés dans l'industrie pour réaliser des tâches dangereuses voir pénible pour l'être humain (soudure du châssis ou peinture de la carrosserie dans une usine automobile), les robots mobiles pour transporter des charges (depuis les chaînes de fabrication jusqu'aux zones de stockage) ou encore pour transporter le courrier dans les bureaux, ou pour intervenir dans des milieux hostiles. [5]

### 1.3.3. Relation Homme-Tâche

La *Figure1.3* représente schématiquement quatre modes de relations entre l'homme et la tâche qu'il doit effectuer à l'aide d'un outil (outil à prendre dans un sens très général), on va se préoccuper de la façon dont l'homme agit sur l'outil et de la façon de contrôler la bonne exécution de la tâche. [6]

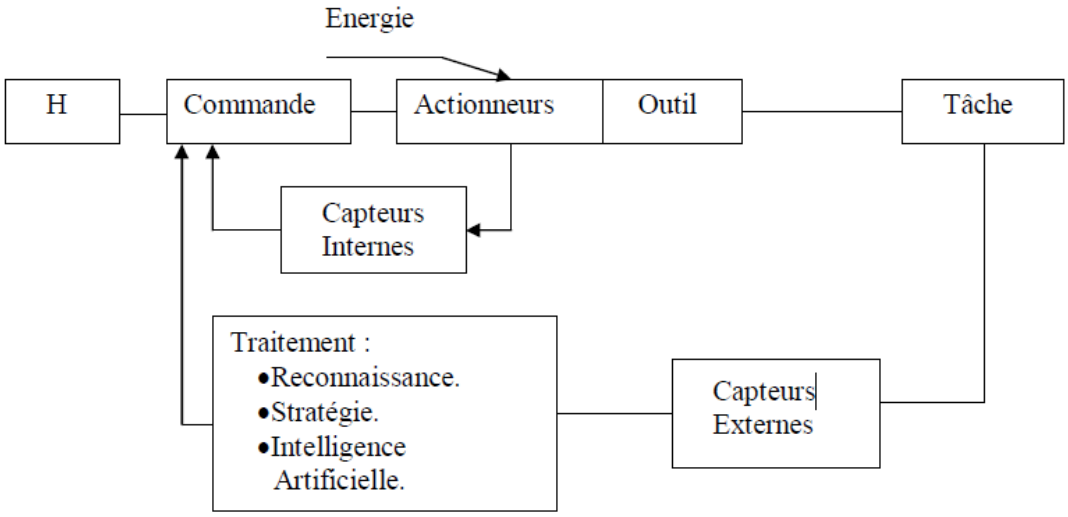



Figure1.3 Relation Homme-Tâche

### 1.3.4. Architecture mécanique

#### 1.3.4.1. Liaisons mécaniques usuelles

Le système d'articulation mécanique (MAS) est un assemblage solide lié entre eux par des liaisons schématisées sur la *Figure 1.4 (a)* par des points à définir une connexion par exemple entre S1 et S2, il est nécessaire d'isoler S1 et S2 de l'ensemble selon *Figure 1.4 (b)* Ensuite, nous définissons d.d.l (degré de liberté) de la connexion comme un nombre mouvements indépendants possibles de S2 par rapport à S1.

Désormais il existe trois types d'architectures de MAS en robotique, on distingue :

- Architecture série (ou chaine cinématique ouverte)
- Architecture parallèle (ou chaine cinématique multi boucle)
- Architecture mixte

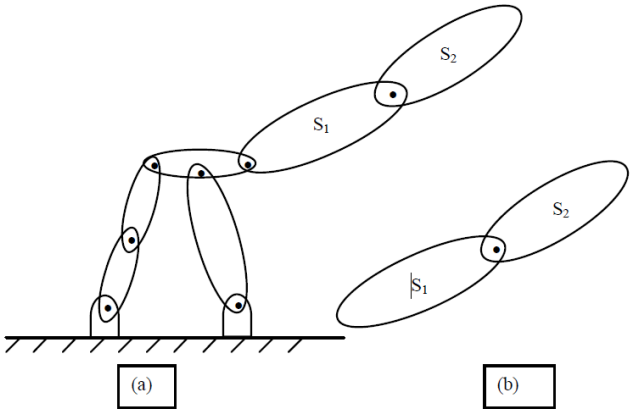



Figure 1.4 Schématisation d'un MAS

#### 1.3.4.2. Principe de fonctionnement :

Un système robotique est décrit comme étant un télémanipulateur, qui est essentiel de bien comprendre le principe de réalisation d'une tâche complexe par un robot.

### 1.3.5. Critères du choix de robots

Le choix s'effectue à base de plusieurs critères, à savoir le domaine d'application du robot ainsi que les caractéristiques du propre robot, tel que :

- La charge maximale du robot ;
- La flexibilité / précision ;
- Nombre d'axes ;
- Rayon d'action ;
- Programmation utilisée ;
- Rapidité/vitesse ;

## 1.4.Langages de programmation des automates programmable industriels

Chaque automate dispose de son propre langage de programmation cependant il existe des langages universels à tous types d'automates. On distingue les langages graphiques (Ladder, FBD et Grafcet) et les langages textuels (IL et ST). [7]

Dans ce qui suit, on va s'intéresser au Grafcet.

Le GRAFCET est un graphe constitué de séquences d'étapes et de transitions reliées par des liaisons orientées.

L'étape représente un état dans lequel l'automatisme est invariant vis à vis de ses entrées/sorties. Elle peut être active ou inactive. L'état du GRAFCET est défini, à un instant donné, par l'ensemble de ses étapes actives.

La transition traduit la possibilité d'évolution d'un état vers un autre. Cette évolution est la conséquence du franchissement de la transition. Une transition est validée si toutes ses étapes immédiatement amont sont actives.

Une liaison orientée relie une étape à une transition et inversement. Elle indique les configurations atteignables à partir d'un état donné.[8]

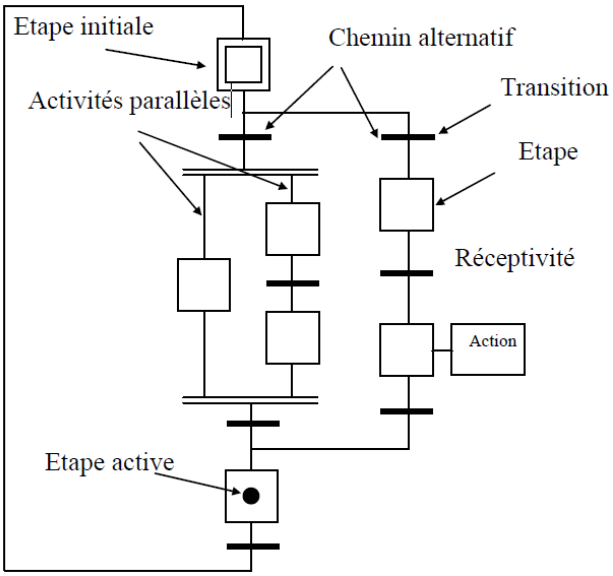



Figure 1.5 Concepts de base d'un GRAFCET

Le grafcet suit des règles bien précises. Ces règles sont les suivantes :

- **Situation initiale***:* la situation initiale correspond aux étapes actives au début du fonctionnement. C'est donc le comportement au repos.

- **Franchissement d'une transition :** une transition est dite validée lorsque toutes les étapes amont de cette transition sont actives. Le franchissement d'une transition est effectif lorsque la transition est validée et lorsque la réceptivité associée est vraie.
- **Activation des étapes**: le franchissement d'une transition entraîne immédiatement l'activation des étapes aval de cette transition et la désactivation de ses étapes amont.
- **Evolutions simultanées** : plusieurs transitions simultanément franchissables sont effectivement franchies simultanément.
- **Activation et désactivation simultanée d'une étape** : si, au cours du fonctionnement, une étape est simultanément activée et désactivée, alors elle reste active.

## 1.5. Généralités sur la supervision industrielle

La supervision a comme fonction d'estimer l'état du procédé en fonction de son état présent et des actions de contrôle passées ainsi de détecter et diagnostiquer les dysfonctionnements, qui aide à l'évaluation de l'état du procédé et à la détection et au diagnostic des pannes. [9]

Un système de supervision donne de l'aide à l'opérateur dans la conduite du processus, son but est de présenter à l'opérateur des résultats expliqués et interprétés et son avantage principal est

- Surveiller le processus à distance ;
- La détection des défauts ;
- Le diagnostic et le traitement des alarmes ;
- Traitement des données ;

La majorité des systèmes de supervisions se composent d'un support à microprocesseurs (ordinateur, pupitre) doté d'un logiciel de supervision en communication via un réseau local industriel avec une ou plusieurs parties commande.

La conception de tels systèmes de supervision demande l'utilisation de logiciels conçus à cet effet. Parmi les plus intéressant nous citerons le logiciel EasyBuilder Pro qu'on va ensuite voir par détails. [10]

Généralement le moteur central (logiciel) est la partie essentielle dans les systèmes de supervisions, à qui se rattachent des données provenant des équipements (automates).

Toutefois ce logiciel de supervision assure l'affichage, le traitement des données, l'archivage et la communication avec d'autres périphériques, la figure ci-dessous indique les modules de visualisation .[11]

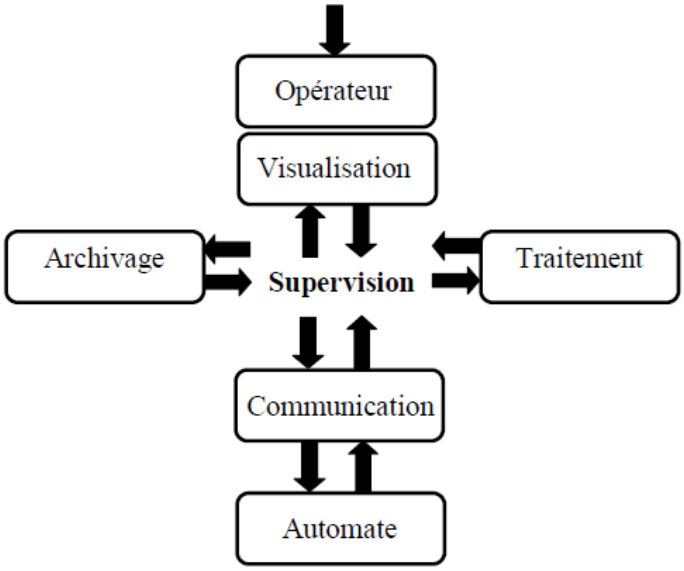



Figure 1.3.5.6 Schéma synoptique d'un système de supervision

Il y a deux types de critères pour bien choisir le système de supervision, afin de facilité son intégration au sein de l'installation industrielle :

Le premier critère concerne la compatibilité Opérationnelle qui se base sur l'intégration du Système Hard au sein d'une installation et les caractéristiques du hardware d'où il est impératif de connaître le type du PC et ses caractéristiques et le type de communication à utiliser.

Le deuxième critère concerne la compatibilité Fonctionnelle qui dépend des possibilités des fonctions du système à répondre aux besoins en plus des caractéristiques du software. [12]

## 1.6. Conclusion

A partir de ce chapitre, on a réussi à bien comprendre les notions de l'automatisation et décrire ses principaux objectifs, que ça soit sur l'être humain ou bien pour l'entreprise.

Généralement les systèmes automatisés de production sont constitués de trois parties majeurs : la partie opérative, la partie commande et la partie relation.

La technologie électromécanique, électrique, pneumatique et hydraulique, sont les trois technologies utilisées pour procéder à une automatisation.

Les robots industriels sont destinés à remplacer les efforts manuels de l'homme par un procédé automatisé. Le contrôle de ces robots est gérer par des programmes injecté dans des cartes électroniques, dont le contenu est rechargeable et modifiable en fonction des taches préétablies.

On distingue deux types de robots : les robots mobiles et les robots fixes. Le choix s'effectue à base de plusieurs critères, à savoir le domaine d'application du robot ainsi que ces caractéristiques intrinsèques.

La supervision industrielle est destinée pour gérer les procédés automatisés en créant les interfaces homme-machines pour pourvoir communiquer entre les machines et l'être humain afin de rendre l'environnement plus confortable loin des défaillances et risques, ainsi et pour surveiller la partie opérative et la partie commande.

# CHAPITRE 2 :

## ELEMENTS DE LA PLATEFORME D'AUTOMATISATION ET DE SUPERVISION

## 2.1.Introduction

La plateforme d'automatisation et de supervision doit s'appuiera sur des outils logiciels et matériels, ce que on va détailler dans ce chapitre .on verra un logiciel et une carte pour programmation et automatisation des systèmes, un logiciel pour supervision des processus et un autre pour la simulation des circuits électroniques.

## 2.2.Arduino

Auparavant la programmation d'un microcontrôleur était quelque chose de complexe, de lent, et qui coute un peut cher ; du coup dans le but de réaliser facilement les circuits électroniques une équipe hispano-italiano-américaine composée Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, David Mellis et Nicholas Zambetti ont conçu une petite carte électronique programmable et un logiciel multiplateforme, appelés Arduino. [15]

### 2.2.1. Carte électronique Arduino

Est une carte électronique programmable open source flexible ou rigide composées d'époxy ou de fibre de verre. Elles sont dotées de processeur et de mémoire et de différentes bronches interconnectées permettant de réaliser des circuits électroniques intelligents et des multitudes d'opérations désirées.[15]

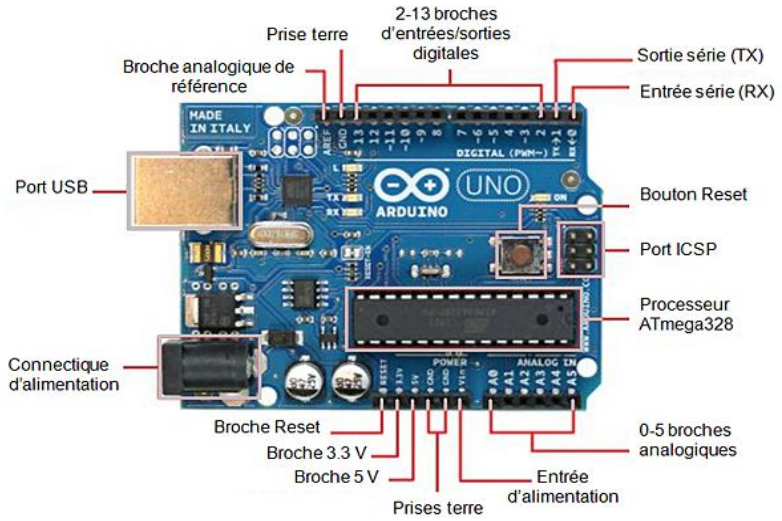



Figure 2. 1 Carte électronique Arduino UNO

Jusqu'à présent plus de 17 versions de cartes Arduino ont été commercialisé, la plus part d'entre elles sont fabriquées par la société italienne Smart Projects et quelques autres par la société américaine SparkFun Electronics.

Chaque une de ces cartes a ses propres spécificités et surtout son propre microcontrôleur ; qui sont généralement des µc de la société Atmel AVR.

Dans le tableau ci-dessous des exemples sur cartes Arduino avec quelques chiffres techniques :

| VERSION DE CARTE | ANNEE DE SORTIE | MICRO-CONTROLLEUR | BROCHES E/S TOR | BROCHES ANALOGIQUES | DIMENSIONS EN mm |
|---|---|---|---|---|---|
| Diecimila | 2007 | ATmega168V | 14 | 6 | 68.6 x 53.3 |
| LilyPad | 2007 | ATmega168V/ATmega328V | 14 | 6 | r=50 |
| Nano | 2008 | ATmega328/ATmega168 | 14 | 8 | 43 x 18 |
| Mini | 2008 | ATmega168 | 14 | 8 | 30 x 18 |
| Mini Pro | 2008 | ATmega328P | 14 | 6 | 33 x18 |
| Duemilanove | 2008 | ATmega168/ATmega328 | 14 | 6 | 68.6 x 53.3 |
| Ega | 2009 | ATmega1280 | 54 | 16 | 101.6 x53.3 |
| Fio | 2010 | ATmega328P | 14 | 8 | 40.6 x 27.9 |

| | | | | | |
|---|---|---|---|---|---|
| Mega 2560 | 2010 | ATmega2560 | 54 | 16 | 101.6 x53.3 |
| Uno | 2010 | ATmega328P | 14 | 6 | 68.6 x 53.3 |
| Ethernet | 2011 | ATmega328 | 14 | 6 | 68.6 x 53.3 |
| Mega ADK | 2011 | ATmega2560 | 54 | 16 | 101.52x53.3 |
| Leonardo | 2012 | ATmega32U4 | 20 | 12 | 68.6 x 53.3 |
| Esplora | 2012 | ATmega32U4 | / | / | 165.1x60.96 |
| Micro | 2012 | ATmega32U4 | 20 | 12 | 48 x 18 |
| Yún | 2013 | ATmega32U4 +Linino | 20 | 12 | 73 53 |

Tableau 1 Chiffres sur les cartes Arduino disponibles

### 2.2.2. Logiciel Arduino

Les créateurs ont développé un logiciel de programmation qui porte la même appellation.

L'IDE multiplateforme écrit en Java et inspire des processing, permet de programmer d'une façon simple et visuelle les cartes Arduino.

L'IDE affiche une fenêtre graphique qui contient un éditeur de texte et tous les outils nécessaires à l'activité de programmation.
Vous pouvez donc saisir votre programme, l'enregistrer, le compiler, le vérifier, le transférer sur une carte Arduino. La fenêtre suivante montre l'environnement d'un logiciel Arduino.[13]

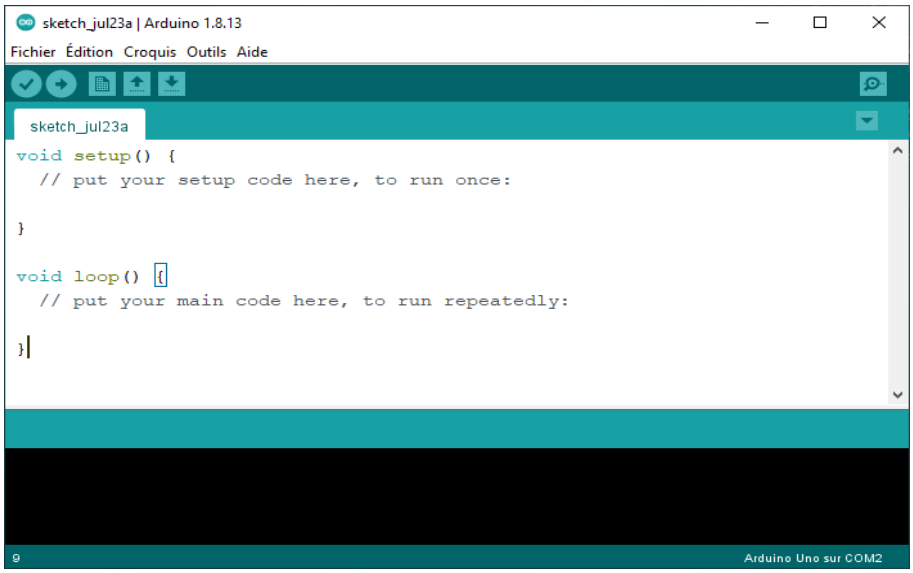



Figure 2. 2 Environnement d'un logiciel Arduino

Le programme Arduino écrit en C doit avoir au minimum les deux fonctions qui apparaissent dans l'IDE de démarrage, la fonction « setup () » et la fonction « loop () ». Il est obligatoire de les écrire, même si elles ne contiennent aucun code.

-Cette fonction setup () ou la fonction d'initialisation est appelée une seule fois lorsque le programme commence. C'est pour ca qu'on va écrire là-dedans le code qui n'a besoin de s'exécuter qu'une seule fois. . On y retrouvera d'habitude la mise en place des différentes sorties et quelques autres réglages.

- Dans la fonction loop () où l'on va écrire le contenu du programme. Cette fonction est appelée en permanence, elle est exécuté plusieurs fois, c'est une sorte de boucle infini.[14]

Avant les deux fonctions, au début on peut inclure les librairies spécifiques dont on a besoin ou déclarer les variables globales.

Donc dans les deux fonctions précédentes on met un jeu d'instructions Arduino, qu'on peut trouver au le site Arduino, en allant sur Aide>> Référence.

Voici un simple programme pour allumer une LED à la branche 13 avec lecture de la position de l'interrupteur à la branche12 :

```
int switchState ;
void setup() {
pinMode(13,OUTPUT);
pinMode(12,INPUT);
}
void loop() {
  switchState=digitalRead(12) ;
   if (switchState==HIGH)  digitalWrite(13,LOW);

 if (switchState==LOW )digitalWrite(13,HIGH);

  }
```

Figure 2. 3 Programme pour allumer une LED

## 2.3. Communication via Modbus

La communication consiste à transmettre des informations. Pour que la communication s'effectue correctement, les interlocuteurs doivent parler le même langage et avoir une certaine maitrise des règles minimales d'émission et de réception de données ; brièvement, ils doivent avoir un protocole de communication commun.

### 2.3.1. Protocole de communication

Un protocole est un ensemble de règles et de procédures à respecter pour émettre et recevoir des données sur un réseau pour permettre la communication entre des processus ou stations. Les protocoles sont hiérarchisés en couches, pour décomposer et ordonner les différentes tâches, plusieurs modèles existe parmi eux on trouve le modèle OSI et le TCP/IP.

Dans l'industrie, on trouve les RLI (réseau local industriel) , des systèmes de communication entre plusieurs équipements de type industriel dans une zone géographique limitée.

Il existe plus de 2000 bus de terrain différents, les technologies les plus connues et utilisées sont : MODBUS, PROFIBUS…[16]

### 2.3.2. Modbus [17]

Modbus est un protocole de communication industriel en 1979 aujourd'hui Schneider Electric pour la transmission des données entre les instruments de contrôle et l'automate, Capteurs et IHM. Maintenant c'est devenu une norme « open protocole » dans les domaines de l'automatisation et des communications industrielles. Il est le plus habituellement utilisé pour les équipements industriels de communication.il fonction sur le mode mono-maitre/esclave contrairement au PROFIBUS. Le MODBUS est flexible et facile pour la mise en œuvre. Il existe trois variantes du MODBUS .

- Le Modbus RTU (8bits) ;

- Le Modbus ASCII (7 bits) ;

-Le Modbus TCP/IP (Ethernet) ;

Le support physique utilisé pour ModBus peut être l'un des suivants:

- RS-232;

- RS-485;

- RS-422;

- Ethernet TCP/IP (Modbus Ethernet);[18]

### 2.3.3. Le ModBus RTU [19]

Modbus RTU (Remote Terminal Unit) est l'un des deux modes de transmission définis dans la spécification Modbus d'origine. Modbus RTU et ASCII sont conçus pour les périphériques série prenant en charge les protocoles RS232, RS485 et RS422. L'une des fonctions spécifiques de Modbus RTU est son utilisation du codage binaire et sa méthode avancée de recherche d'erreurs CRC. Modbus RTU est l'implémentation de protocole Modbus la plus utilisée pour les applications industrielles et les systèmes de production automatisés.

Le maître Modbus RTU est l'appareil central qui demande des informations à l'appareil esclave qui lui est connecté. Le contrôleur central peut servir de station maître Modbus RTU. Une implémentation Modbus ne peut avoir qu'un seul maître. Le périphérique maître collecte

des informations sur le périphérique esclave et peut également écrire dans le registre du périphérique esclave.

L'esclave Modbus RTU est le périphérique répondant aux ordres envoyés par le maître. Il ne peut pas initier les communications, il est en phase d'attente tant qu'il n'a pas à répondre à une requête du maître.

Il y a un seul périphérique maître dans le protocole Modbus RTU et il peut y avoir jusqu'à 247 périphériques esclaves. Chaque esclave prend une adresse de 1 à 247.

La trame de MODBUS RTU est constituée d'une suite d'informations écrites en hexadécimal, elles sont classe comme la figure ci-dessus le montre :

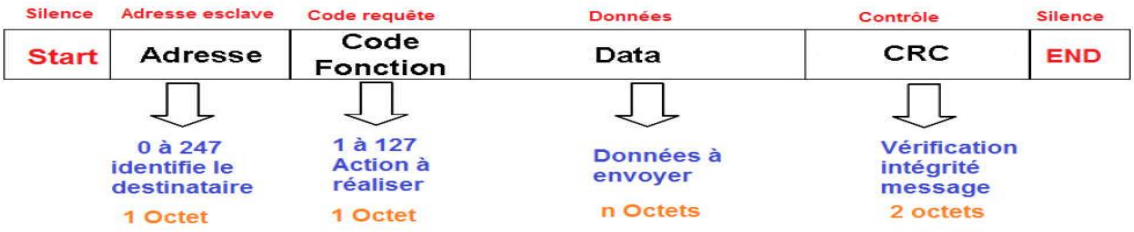



Figure 2. 4 La Trame de MODBUS RTU

### 2.3.4. RS-232 [20]

Le RS 232 est une norme qui standardise les voie de communication de type série, elle est appelée aussi « port série » ou « port COM » puisque dans quelque système d'exploitation comme Windows cette liaison est désigné par COM1, COM2…

De nos jours on utilise les USB/RS-232 Avec l'aide d'un émulateur de port série virtuel (vspd), car les interfaces physiques n'existent plus dans les nouveau PC depuis le milieu des années 2000.

La transmission des données s'effectue suivant un ensemble de paramètres similaire entre l'émetteur et le récepteur autrement dit un protocole commun. Ces paramètres sont :

- Le bit de Start : il indique le début de l'émission de la trame, il correspond à la transition du niveau Haut au niveau Bas.

- La donnée : l'émission de donnée commence par le bit de poids le plus faible, le Nombre de bits arrivent jusqu'à 8 bits (un octet de données).

- Le bit de parité : ce bit est facultatif, il permet de vérifier les erreurs de transmission parle récepteur.

- Le bit de Stop : indique la fin de la trame, il est toujours au niveau Haut.

## 2.4.L'interface Homme-Machine (IHM ou HMI ou MMI)

L'IHM signifie interface homme machine et fait référence à un tableau de bord qui permet aux utilisateurs de communiquer avec des machines, des programmes informatiques ou des systèmes. Techniquement, vous pouvez appliquer le terme IHM à tout écran qui interagit avec l'appareil, mais il est généralement utilisé pour décrire l'écran utilisé dans un environnement industriel. L'IHM affiche des données en temps réel et permet aux utilisateurs de contrôler la machine via une interface utilisateur graphique.

On distingue trois types d'IHM, de point vue organique :Les interfaces d'acquisition comme les boutons, les molettes, les joysticks, les claviers, … et les interfaces de restitution comme les écrans, les témoins à LED, … en plus des interfaces combinées qui se présentent sous forme d'écrans tactiles et Multi-touch, des écrans de type Nano Mod non tactiles et des commandes à retour d'effort.[21]

### 2.4.1. Logiciel Easybuilder

C'est un logiciel destiné à la programmation des IHM.L'interface ergonomique d'Easybuilder Pro fournit aux développeurs tous les outils nécessaires pour créer des fenêtres, des boutons, des indicateurs, des alarmes, des recettes, historiques, des droits d'accès, des mots de passe et déclarer des utilisateurs. Tous les automates, lecteurs, lecteurs de codes-barres et autres consoles réseau.

La couleur, la forme, l'animation et l'étiquette de chaque objet sont entièrement configurables pour optimiser l'utilisation et afficher les résultats.[22]

Pour la création d'un nouveau projet, le logiciel propose à l'utilisateur de différents modèles de pupitres et interfaces, chacun a sa propre résolution ainsi que quelques propriétés disponibles, comme le types de liaison sur le COM, le wifi, Ethernet, la carte SD, USB…[23]

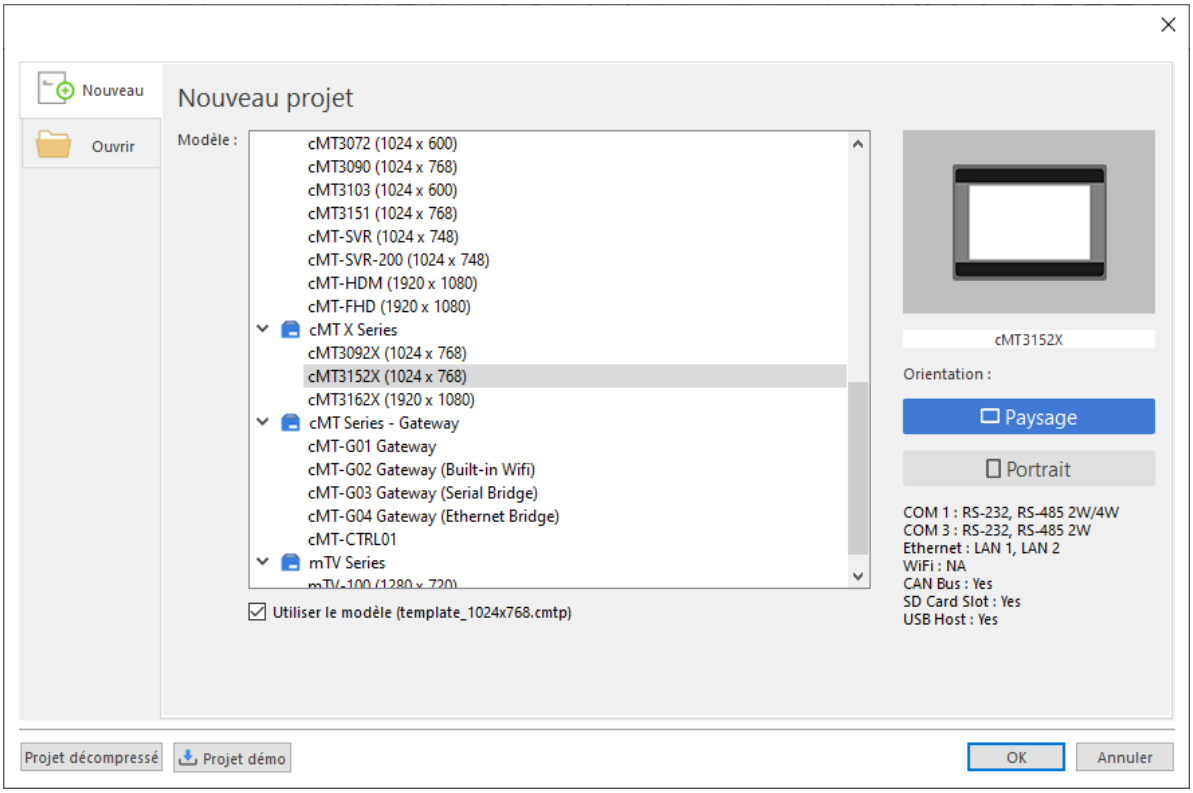



Figure 2. 5 Fenêtre nouveau projet

Après le lancement de nouveau projet une boite de dialogue « paramètres système » s'ouvre pour ajouter et configurer le nouveau périphérique.

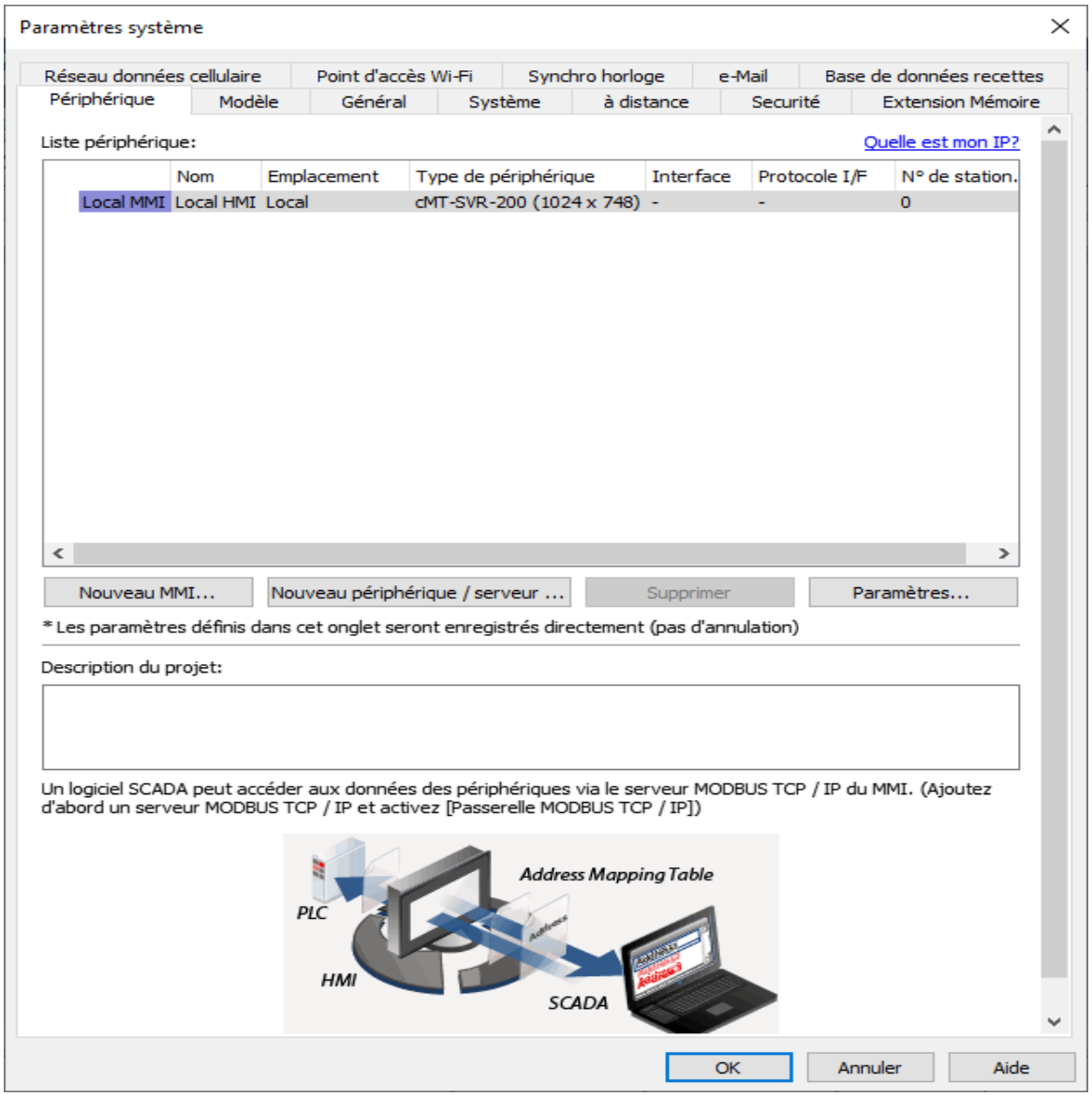



Figure 2. 6 Paramètres système.

En cliquant sue le « nouveau périphérique/serveur » sur la fenêtre intitulée « propriétés de périphérique », on choisit le type de périphérique, le type de support physique utilisé, et les paramètres de COM en occurrence son numéro, le débit, le nombre de bits des données, le type de parité et le nombre de bits de stop.

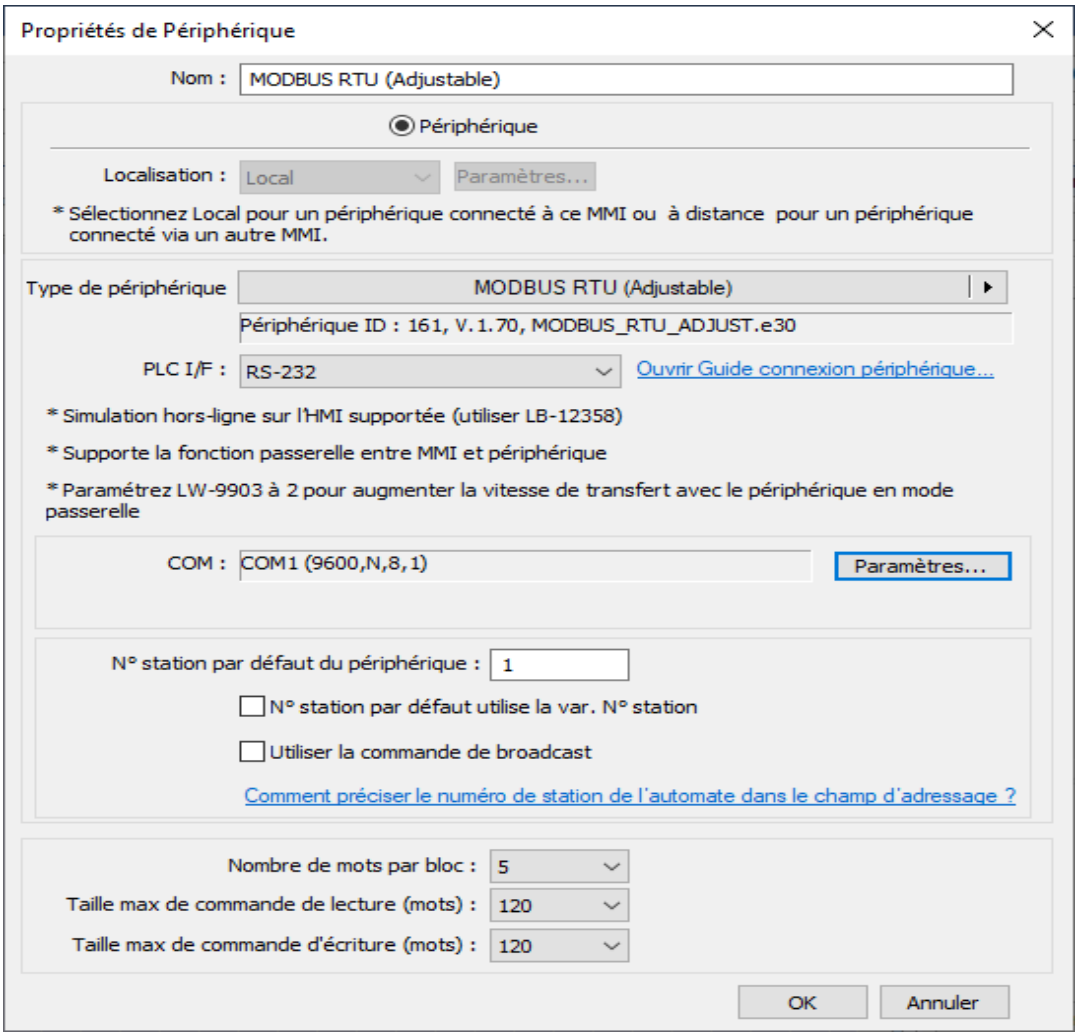



Figure 2. 7 Propriétés de périphérique

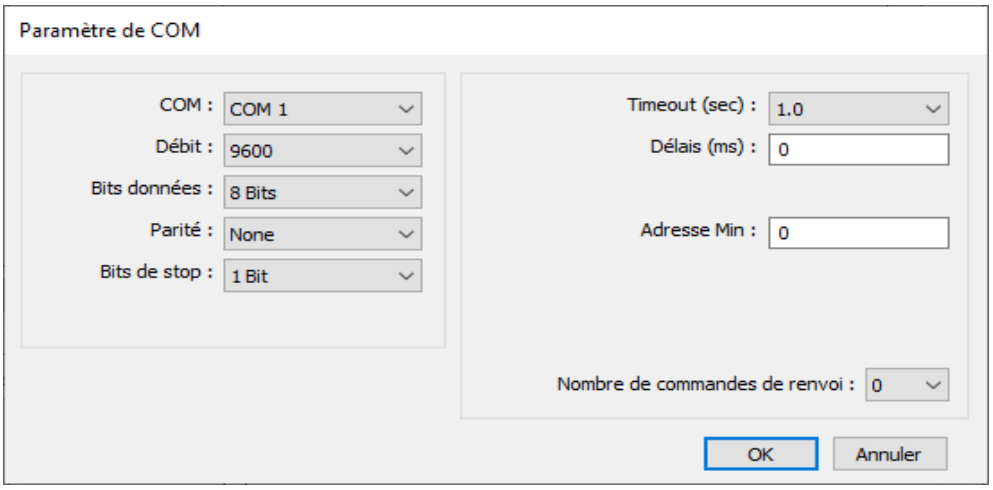



Figure 2. 8 Paramètre du COM

Une fois la configuration matérielle terminée, la fenêtre d'édition apparait où on trouve les différentes objets et icones souhaitable afin de créer son pupitre.

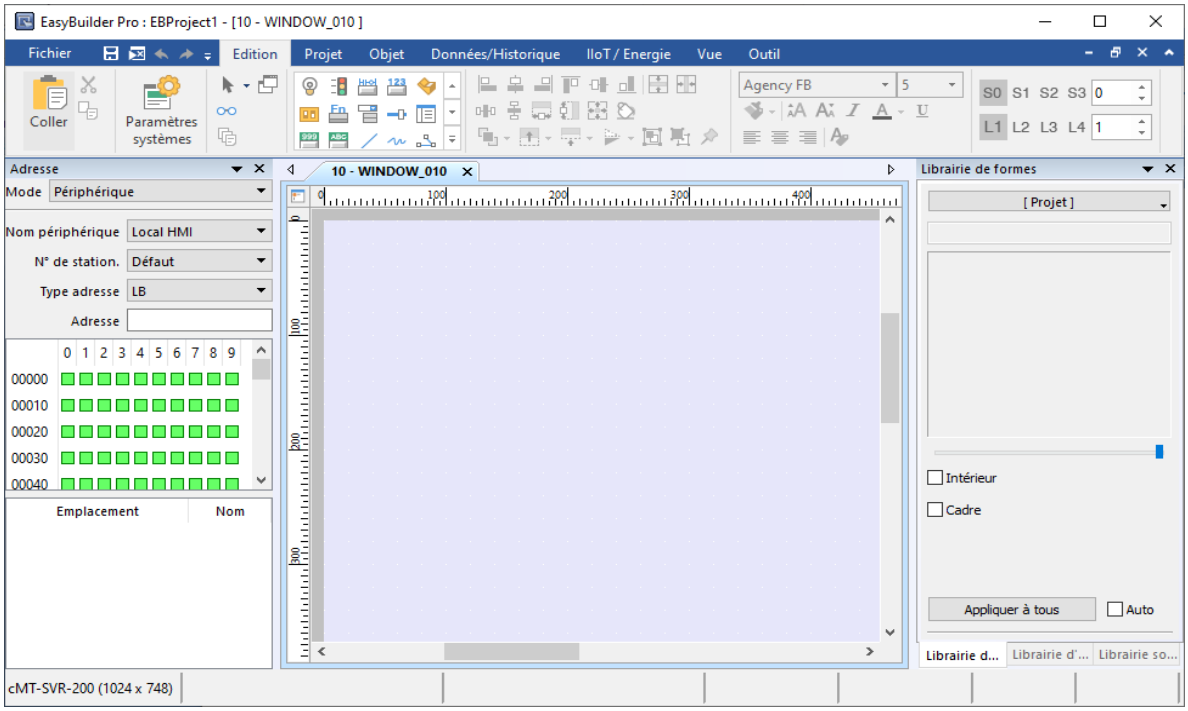



Figure 2. 9 Fenêtre d'édition de projet

Dans la bibliothèque d'objet on clique sur l'objet désiré, puis dans la boite de dialogue, on définit le type de registre d'adresse, l'adresse et la forme et un commentaire.

La définition d'adresse est primordiale car à travers se fait la communication avec la carte ou l'automate en générale.

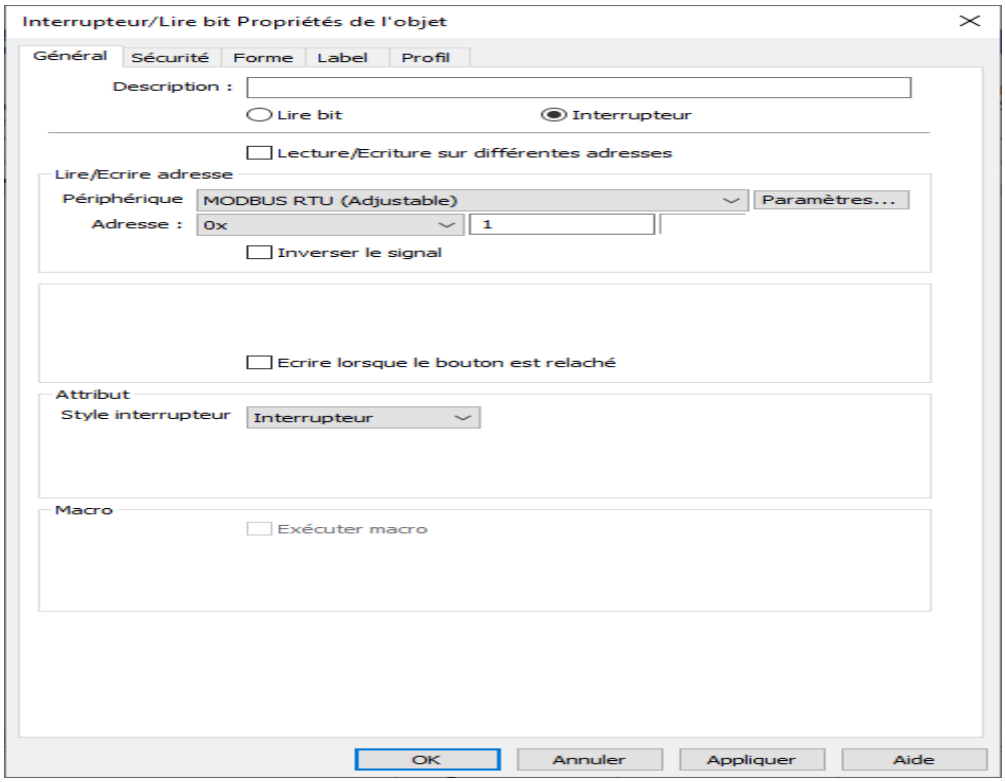



Figure 2.10 configuration d'adresse d'objet

Le projet doit être compilé et vérifié, puis simulé soit par une simulation hors ligne ou par une simulation en ligne.

Le programme sera chargé sur l'IHM une fois toutes les vérifications terminées.

## 2.5.Proteus

Proteus Professional est une suite logicielle destinée à la CAO en Electronique. Développé par Labcenter Electronics, le logiciel inclus dans Proteus Professional est le seul qui permet actuellement une CAO complète (construction assistée par ordinateur) dans le domaine de l'électronique. La suite logicielle se compose des logiciels principaux (ISIS, ARES, PROSPICE) et VSM.[24]

Proteus Professional présente des avantages comme :

- Pack de logiciels faciles à comprendre et à utiliser ;

- Support technique efficace ;
- Les outils de prototypage virtuel aident à réduire les coûts matériels et logiciels lors de la conception de projets ;

Le logiciel Proteus ISIS est utilisé particulièrement pour éditer des schémas électriques. En outre, le logiciel peut simuler ces schémas dans le but de détecter certaines erreurs dès la conception. A travers, on peut simuler également un circuit avec microcontrôleur puisque il a la puissance de suivre pas à pas les instructions contenus dans les programmes (code source). [25]

Les circuits produits par ce logiciel peuvent être intégrés dans des documents scientifiques car le logiciel peut contrôler la plupart des aspects graphiques des circuits.

Le logiciel ARES complète parfaitement ISIS puisque il est destiné à l'édition et au routage. Les schémas électriques générés sur ISIS peuvent être facilement importés dans ARES pour produire des PCB (circuits imprimés) pour cartes électroniques. Bien qu'il soit plus efficace d'éditer le PCB manuellement, le logiciel vous permet de placer automatiquement les composants et d'effectuer automatiquement le routage.[25]

L'environnement ISIS Proteus est comme toute autre environnement classique type Windows, on constate directement qu'il se constitue d'une fenêtre principale pour le travail et d'ensemble de barres d'outils et de gestion de projets.

On trouve :

-**Zone de travail :** pour le développement des schémas électriques et leur simulation.

-**Barre des menus :** pour la gestion du projet en occurrence ouverture, sauvegarde, impression, mode d'affichage, l'aide…

-**Barre d'outils de commande :** Cette barre fournit un accès équivalent à la barre des menus. Elle peut être masquée par la commande "Barre d'outils" du menu "Affichage». Elle inclut les parties suivantes : de gauche à droite, commandes fichier/projet, commandes d'affichage, commandes d'édition et bibliothèque et commandes outils.

-**Barre d'orientation d'outil :** pour contrôle d'affichage en terme rotation et réflexion

**-Vue d'ensemble :** qui montre une représentation simplifiée de la totalité du dessin. A travers on peut focaliser le cadre sur une partie.

**-Barre des touches magnétoscope :** constitué des raccourcis permettant le lancement de la simulation, la mise en pause, l'exécution pas à pas et l'arrêt de la simulation.

**-Bibliothèque d'objets :** sorte de sélecteur d'objets qui liste les différents éléments précédemment utilisés .Les types d'objets qui peuvent y apparaître sont les composants, les marqueurs, les graphes, les terminaux, les symboles graphiques…

**-Barre d'outils principales et sélection du mode :** la barre verticale de raccourcis et de boutons sert au développement rapide d'applications.[26]

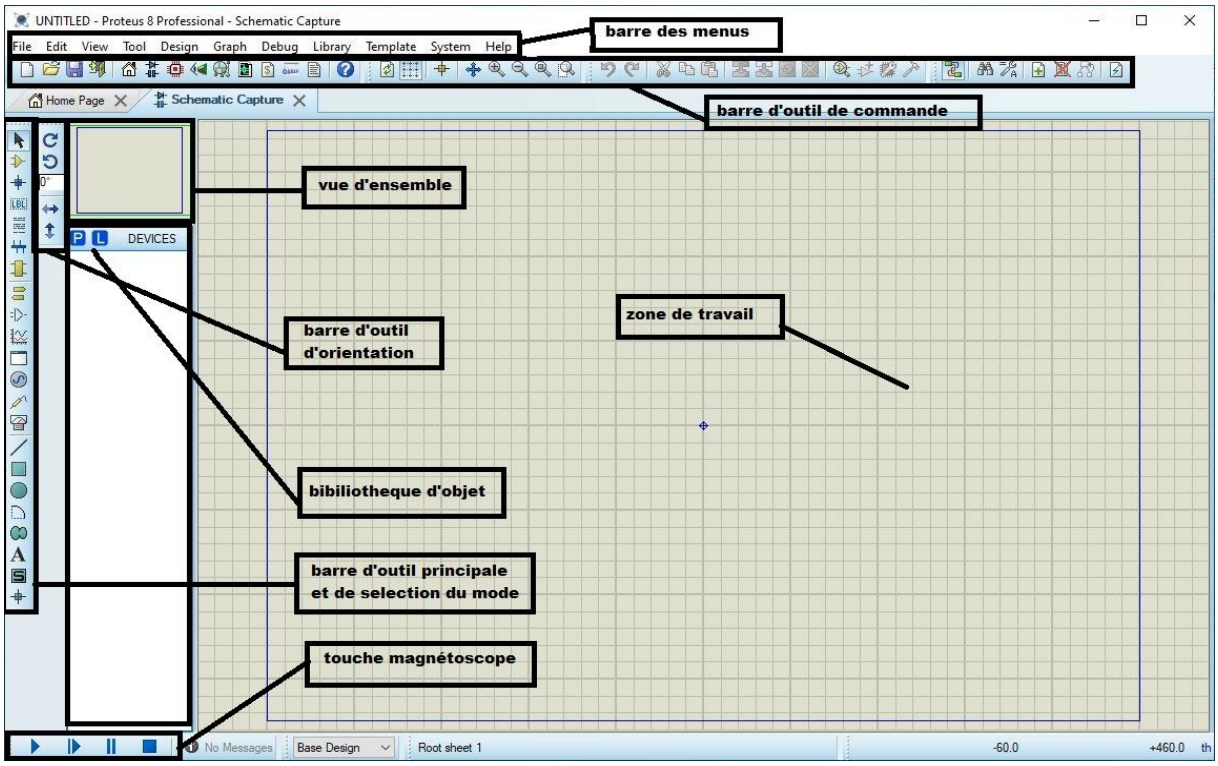



Figure 2. 11 Environnement ISIS proteus

On distingue trois modes dans la barre d'outils :

- Mode principal : on trouve dedans les différents composants et compris la bibliothèque d'objet.

-Mode gadgets : donne accès aux différents types de générateurs, mise a la terre et différents appareils de mesures.

-Mode graphique : permet de dessiner une forme ou écrire des textes.

| MODE PRINCIPAL | | MODE GADGETS | | MODE GRAPHIQUE | |
|---|---|---|---|---|---|
| | Composants | | Terminal | | Ligne |
| | Point de jonction | | Patte de composant | | Rectangle |
| | Label de fil | | Graphe | | Cercle |
| | Script de texte | | Cassette | | Arc |
| | Bus | | Générateurs | | Chemin |
| | Sous circuit | | Sonde de tension | | Texte |
| | Édition | | Sonde de courant | | Symbole |
| | | | Appareils | | Marqueur origine |

Figure 2. 12 Modes et outils de la barre d'outils

Pour ajouter un quelconque composant au schéma, il faut :

- Sélectionner le mode composant dans la barre d'outils et de modes.

- Cliquer sur le l'icône « P » qui se trouve à la barre bibliothèque et sélecteur d'objet.

Une fenêtre intitulée « pickdevices », la grande bibliothèque qui rassemble la majorité du composant électrique et électronique disponible sur le marché.

Les composants sont classés en premier temps sous forme catégories, puis sous forme de sous-catégories et finalement par manufacturier ou producteur. Mais il suffit de taper le nom du composant afin que le résultat s'affiche.

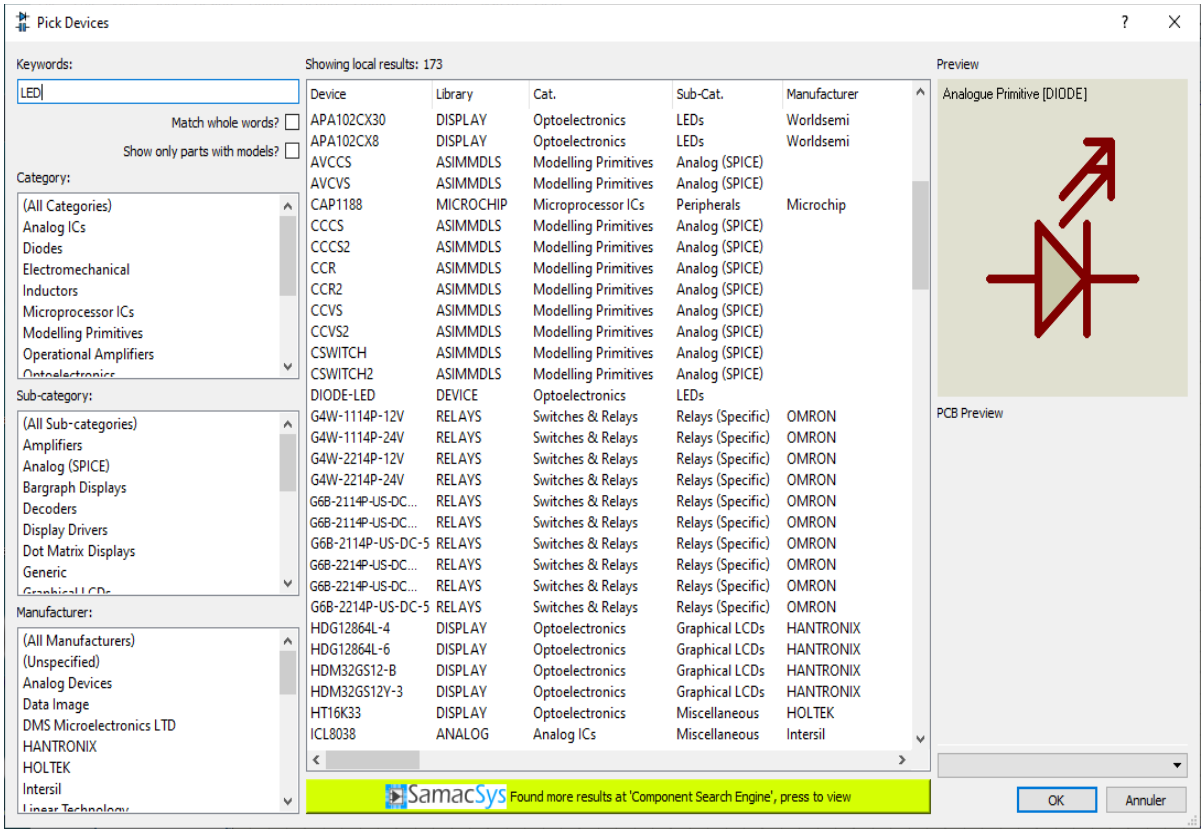



Figure 2. 13 Fenêtre de la bibliothèque des composants

Une fois la recherche faite, il suffit de sélectionner l'objet et de cliquer sur OK pour le charger a la zone du travail et à la barre de sélectionneur d'objet. Cette procédure se répète jusqu'à que tous les composants voulus seront chargés.

Finalement il reste que de mettre des connexions entres les composant et cela se fait en deux façons :

Soit manuellement avec le curseur, par clique il se transforme à un crayon qu'à travers on se déplace de l'extrémité d'un objet à un autre.
Soit automatiquement, par sélection de l'option auto-routeur dans le menu d'outils, puis Placer le curseur sur l'extrémité de la patte à connecter. Le pointeur se transforme en un crayon. Cliquer pour valider le point de départ. Ensuite Placer directement le curseur sur le point d'arrivée, puis cliquer comme ça la liaison est faite automatiquement.

Quand l'édition de projet est finalisée totalement en termes d'objet, de générateurs, de connexions et de programmes, il sera prêt à la simulation qui s'effectue simplement par un simple clique sur le bouton « Run simulation ».

## 2.6. Conclusion

Les logiciels vus dans ce chapitre sont fortement utilisés dans les domaines liés à l'électronique, l'informatique et l'automatisme industriel.

Premièrement l'Arduino est l'une des cartes électroniques les plus faciles à programmer puisque elles disposent de différentes fonctionnalités et librairies d'exploitation plus simple à utiliser contrairement à ces paires.

Par la suite on a découvert un logiciel de programmation des pupitres de commande easybuilder, qui est riche en bibliothèque d'objets qu'on trouve d'habitude dans des industries, comme on a présenté la manière de communiquer entre l'IHM et l'automate avec protocole modbus.

L'outil proteus est très célèbre aussi dans la conception et simulation des circuits électroniques, avant de les mettre sur le terrain.

Tous ce qui a été vue dans ce chapitre seront utilisés dans les applications prochaines pour démontrer la façon de les mettre en œuvre.

# CHAPITRE 3 :

## EXEMPLES D'APPLICATION

## 3.1.Introduction

Ce chapitre sera dévoué à la réalisation et la programmation de simples Grafcets par le langage Arduino. Chaque exemple sera accompagné par son circuit électrique édité par proteus et pour quelques-uns par un pupitre de commande élaboré par le logiciel Easybuilder.

Ces exemples seront un support d'explication et d'application des logiciels vus dans le chapitre précédent et seront aussi une introduction parfaite pour le dernier chapitre.

## 3.2.Programmation d'un grafcet par la librairie PlcLib

Dans cet exemple on verra la manière de programmer un grafcet par une libraire particulière appelé plcLib élaborée pour Arduino.

### 3.2.1. La librairie PlcLib

Comme son nom l'indique, c'est une librairie qui sert à transformer une carte Arduino en un API. D'une façon simple et facilement compréhensible, elle permet de programmer des fonctions séquentielles et donc un grafcet.

Avant de commencer la programmation, il faut tout d'abord télécharger la librairie et la copier dans le dossier librairie Arduino et cela est similaire pour toutes les autres librairies.

Le tableau suivant montre quelques instructions de la librairie PlcLib utiles pour coder un grafcet ou n'importe quelle autre opération en relation avec l'API.[27]

| Instruction | Rôle |
|---|---|
| in() | Lire et charger l'entrée entre parenthèses |
| inNot() | Lire et charger l'inverse de l'entrée entre parenthèses |
| out() | Envoyer et écrire dans la sortie entre parenthèses |
| outNot() | Ecrire et envoyer l'inverse à la sorties entre parenthèses |

| | |
|---|---|
| andBit()<br><br>andNotBit() | Faire une opération logique ET ou le NON ET entre le résultat en amont et l'entrée entre parenthèses |
| orBit()<br><br>orNotBit() | Faire une opération logique OU ou le NON OU entre le résultat en amont et l'entrée entre parenthèses |
| xorBit() et xorNotBit() | Faire une opération logique XOR ou le NON XOR entre le résultat en amont et l'entrée entre parenthèses |
| timerOn() | Lancer une temporisation de retard a la montée |
| timerOff () | Lancer une Temporisation avec retard a la retombée |
| Set () | Mise à un de la sortie entre parenthèses |
| Reset () | Mise à zéro de la sortie entre parenthèses |
| Pulse1.**inClock** () | connecter l'impulsion surnommée « pulse1 » à l'entrée d'horloge |
| Pulse1.**rising**() | Lire le front montant |
| Pulse1.**falling**() | Lire le front descendant |
| ctr.**clear**() | Initialiser le compteur surnommé « ctr » (le mettre à la limite inferieur) |
| ctr.**countUp** ()<br><br>ctr.**countDown** () | Compter ou décompter |
| ctr.**preset**() | Mettre le compteur surnommé « ctr » à la valeur limite supérieure |
| inAnalog() | Lire l'entrée analogique |
| compareGT() | Comparer et vérifier si l'entrée en amont est supérieure à l'entrée entre parenthèses |
| compareLT() | Comparer et vérifier si l'entrée en amont est inférieure à l'entrée entre parenthèses |

Tableau 2 Instructions de la librairie PlcLib utiles pour coder un grafcet

### 3.2.2. La configuration matérielle par défaut de la bibliothèque PlcLib

Un ensemble de base d'entrées et de sorties est activé par défaut. Cette configuration par défaut est sélectionnée en incluant tout d'abord le fichier de bibliothèque PLC (#include <plcLib.h>) et deuxièmement en appelant la fonction setupPLC () depuis la section setup() du programme Arduino dit aussi sketch.

Au minimum, le logiciel définit quatre entrées X0, X1, X2 et X3 (entrées analogiques A0 – A3) et quatre sorties Y0, Y1, Y2 et Y3 (broches 3, 5, 6 et 9).[27]

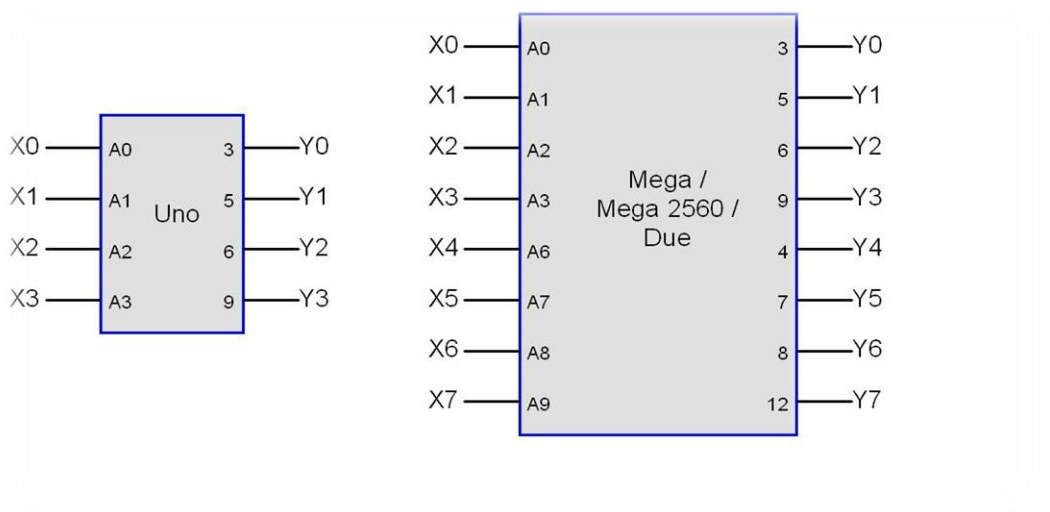



Figure 3. 1 La configuration matérielle par défaut de la bibliothèque PlcLib

### 3.2.3. Cahier des charges

Dans un Grafcet linéaire de trois étapes, on représentera le cahier des charges suivant :

- Par un appui sur le bouton « dcy », le départ de cycle s'effectue et une LED bleue s'allume
- Par un clic sur le bouton «bt », une LED rouge s'allume après 3secondes.
- Quand 10 secondes s'achèvent le système revient à l'état initial avec l'extinction des deux LEDs.

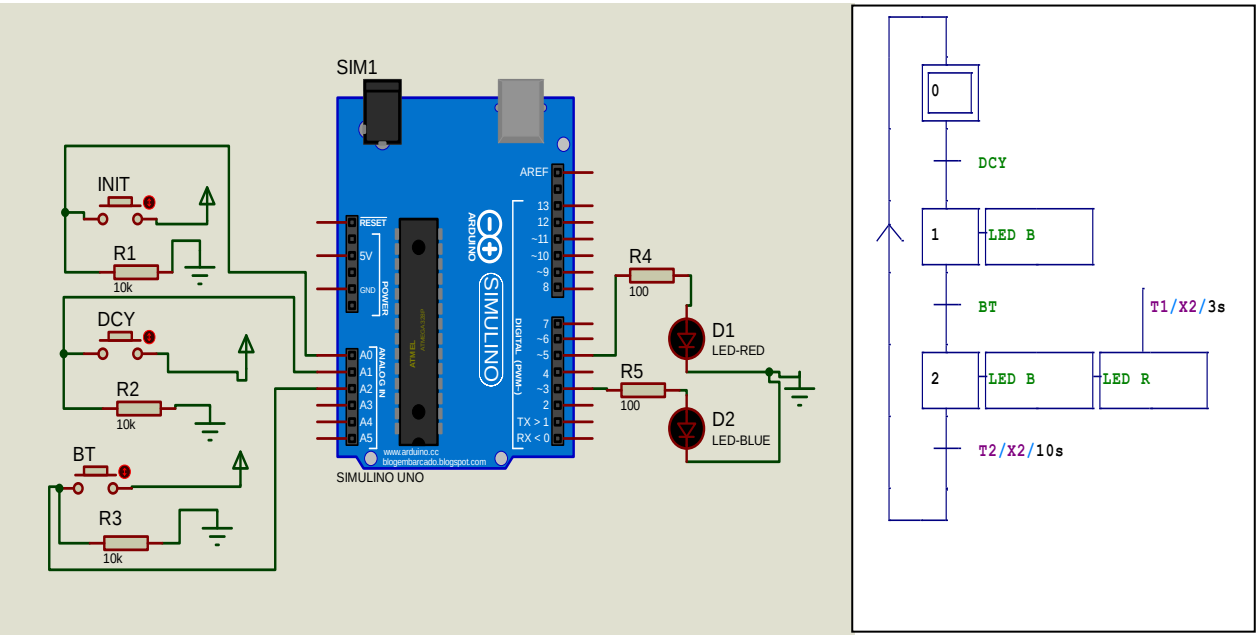



Figure 3. 2 Circuit et grafcet du système

### 3.2.4. Le programme Arduino

Le programme avec « plcLib Library » qui réalise ce grafcet est le suivant :

```
#include<plcLib.h>
/*
la configuration des entrees sorties sont comme suite:
LES ENTREES:
X0 c'est INIT              branché à la pin A0
X1 c'est DCY               branché à la pin A1
X2 c'est BT                branché à la pin A2

LES SORTIES:
Y0 c'est LED BLUE          branché à la pin 3
Y1 c'est LED ROUGE         branché à la pin 5
*/
unsigned int start=1;
unsigned int step1=0;
unsigned int step2=0;

unsigned long tempo1=0;
unsigned long tempo2=0;

void setup() {

setupPLC();
}
```

```
void loop(){

 in(X0);set(start);reset(step1);reset(step2);

 in(start);andBit(X1);set(step1);reset(start);

 in(step1); andBit(X2);set(step2);reset(step1);

 in(step2); timerOn(tempo1,10000);set(start);reset(step2);

 in(step1); orBit(step2);out(Y0);

 in (step2); timerOn(tempo2,3000);out(Y1);
}
```

## 3.3.Réalisation d'un grafcet par la librairie PlcGrafcet

### 3.3.1. La librairie PlcGrafcet

Cette bibliothèque est conçue exclusivement pour programmation des grafcet avec Arduino. Elle se constitue de trois fichiers d'en-tête et de source ; ensemble elles assurent la réalisation de toutes les fonctions fondamentales d'un grafcet.

La bibliothèque Arduino contient au moins deux fichiers, un fichier d'en-tête se terminant par <.h> et un fichier source se terminant par <.ccp>.

Le fichier d'en-tête contient la définition des fonctions disponibles et le fichier source contient l'implémentation du code. Autrement dit, le code des fonctions définies dans le fichier d'en-tête.

Pour utiliser n'importe quelle bibliothèque, il suffit de l'inclure dans le sketch en écrivant : #include <le nom de la bibliothèque .h> c'est- à- dire inclure le fichier d'en-tête seulement.[28]

Pour l'inclusion et l'exploitation des fonctions de cycle d'automate, on utilise les fonctions suivantes :

- updateOutputs() : écriture dans les sorties (pré-actionneur, pins, temporisation, compteur).
- readInputs() : la lecture des entrées.
- computeTrans() : les équations des transitions.
- updateEtapes() : activations et désactivations d'étapes.

Pour la programmation et la commande des temporisations se réalisent par :

- TimerTimer1() : déclaration et initialisation de Timer1 et définition du nombre de cycle d'automates ;
- Timer1.updateTimer() : définition de l'étape dans laquelle la temporisation s'active ;
- Timer1.getTimerOutput() : vérification de fin de temporisation ;

Pour la programmation et la commande de compteurs, on utilise les instructions suivantes :

- Counter Counter1(A, B) : Déclaration et initialisation de Counter1 pour compter de la valeur A jusqu'à la valeur B ;
- Counter1.updateCounter(vEA, vEB) : mise à jour du compteur et définition de l'étape vEB comme étape d'initialisation et l'étape vEA comme l'étape d'incrémentation ou de décrémentation du compteur ;
- Counter1.getCounterOutput() : fin de comptage. [29]

### 3.3.2. Structure de sketch

Le sketch sera organisé de la façon suivante :

- Inclure les librairies
- Définir les pins
- Déclarer des variables (images des entrées et des sorties, les étapes et les transitions)
- Déclarer les temporisations et les compteurs
- Déclarer les fonctions de cycles d'automates vus précédemment
- Dans la fonction setup (),on configure le cycle d'automate avec l'instruction Grafcet1.setCycle(AL).
- Lancer le grafcet avec l'instruction Grafcet.run () dans la fonction loop () du sketch
- Configurer les fonctions du grafcet.

### 3.3.3. Cahier des charges

Cette fois, on verra un exemple de codage d'un grafcet OU exclusif avec une action temporisée.

Avec appui sur le bouton DCY le cycle commence ; en premier temps, deux vérins doubles effets sortent (vérin1 et vérin2), et deux LED s'allume rouge et vert respectivement. Après 5 secondes, le vérin1 rentre et la LED rouge s'éteint. Après le passage de 8 secondes on aura deux cas, soit en cycle unique et donc le vérin2 rentre à l'étape initiale et le LED verte s'éteint, ou en cycle continu ou le vérin2 restes toujours sortis. Chaque fois que le cycle continu est sélectionné une LED bleue s'allume pour 3 secondes quelle que soit l'étape.

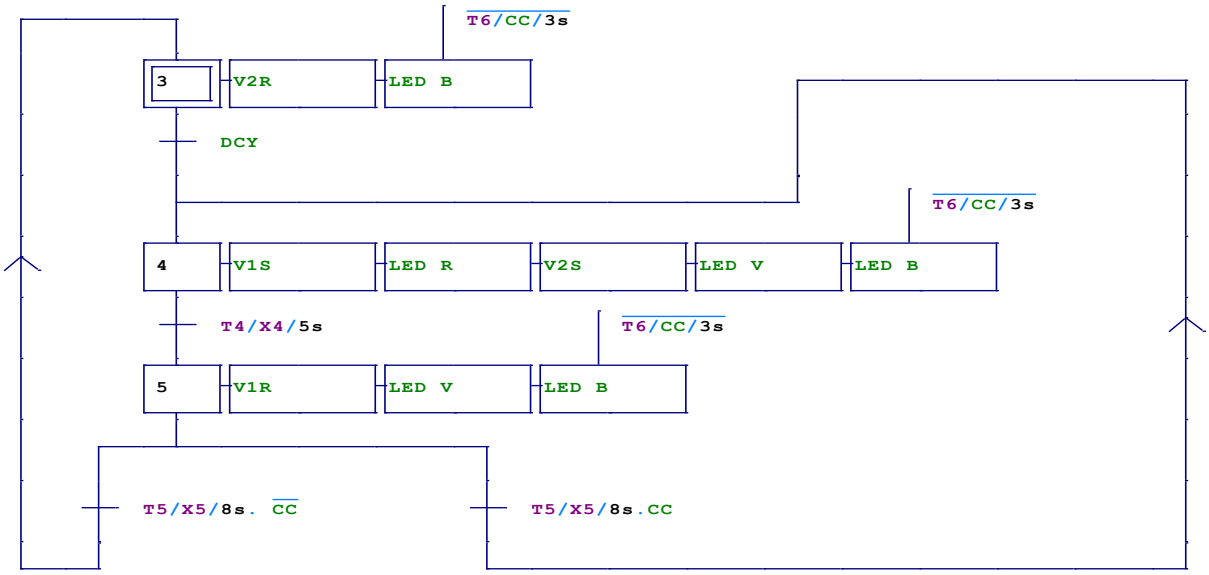



Figure 3. 3 grafcet OU exclusif avec une action temporisée.

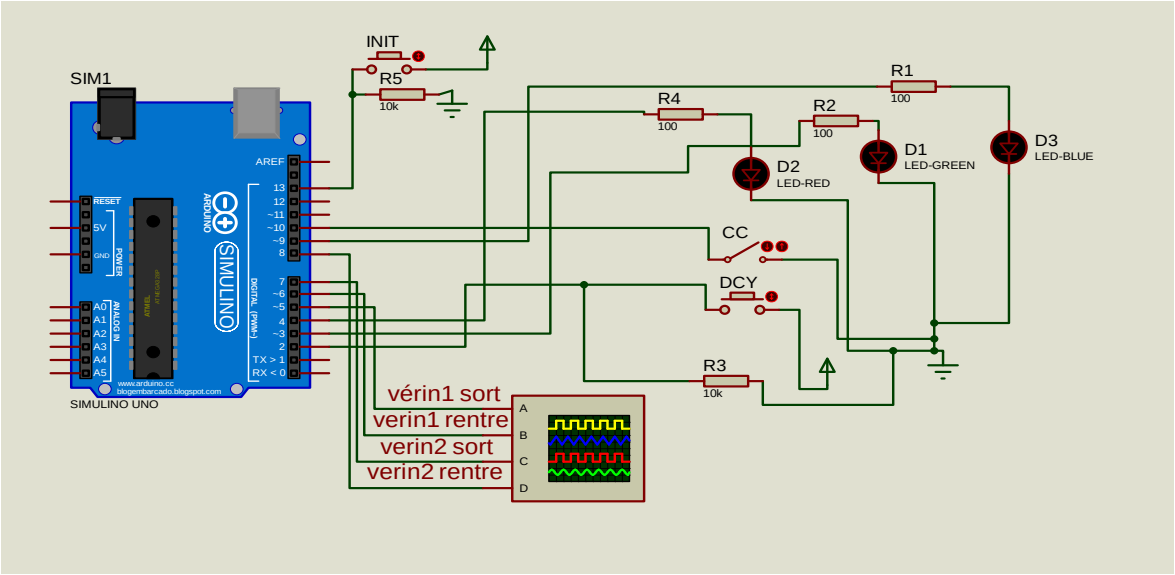



Figure 3. 4 Circuit proteus du système

### 3.3.4. Le programme

Le programme suivant est un exemple de codage de grafcet avec une librairie PlcGrafcet

```
#include "TimerTON.h"
#include "plcGRAFCET.h"
// Définition des E/S
#define  DCY            2
#define  GREENLED       3
#define  REDLED         4
#define  VERIN1SORT     5
#define  VERIN1RENTRE   6
#define  VERIN2SORT     7
#define  VERIN2RENTRE   8
#define  BLUELED        9
#define  CC             10
#define  INIT           13
// Déclaration des variables
// Les variables image des entrées
bool vDCY,vCC,vINIT;
// Les variables des etapes
bool vE0, vE1, vE2;

// les variables des transitions
bool vT0, vT1, vT21,vT22;
// Les variables image des sorties
bool vGREENLED, vREDLED,vVERIN1SORT,vVERIN1RENTRE,vVERIN2SORT,vVERIN2RENTRE,vBLUELE
// Déclaration et initialisation de Timer1
Timer Timer1(500);Timer Timer2(800);Timer Timer3(300);
// Déclaration des fonctions de cycle automate
void updateOutputs(); void readInputs();void computeTrans();void updateEtapes();
GRAFCET Grafcet1(updateOutputs, readInputs, computeTrans, updateEtapes);
void setup() {
// Configuration des E/S
pinMode(INIT,INPUT);  pinMode(DCY, INPUT);
pinMode(GREENLED, OUTPUT);pinMode(REDLED, OUTPUT);
pinMode(VERIN1SORT,OUTPUT);pinMode(VERIN1RENTRE,OUTPUT);
pinMode(VERIN2SORT,OUTPUT);pinMode(VERIN2RENTRE,OUTPUT);
pinMode(BLUELED,OUTPUT);pinMode(CC,INPUT);
// Initialisation des étapes
vE0 = 1;  vE1 = 0;  vE2 = 0;
```

```
// configuration de cycle automate
Grafcet1.setCycle(10L);}
void loop() {
Grafcet1.run(); }
void updateOutputs(){
vGREENLED=vE1;
vREDLED=vE1 or vE2;
vVERIN1SORT=vE1;
vVERIN2SORT=vE1 or vE2;
vVERIN1RENTRE=vE2;
vVERIN2RENTRE=vE0;
vBLUELED= !vCC && !Timer3.getTimerOutput();
digitalWrite(GREENLED,vGREENLED);
digitalWrite(REDLED,vREDLED);
digitalWrite(VERIN1SORT,vVERIN1SORT);
digitalWrite(VERIN1RENTRE,vVERIN1RENTRE);
```

```
digitalWrite(VERIN2SORT,vVERIN2SORT);
digitalWrite(VERIN2RENTRE,vVERIN2RENTRE);
digitalWrite(BLUELED,vBLUELED);
Timer1.updateTimer(vE1);
Timer2.updateTimer(vE2);
Timer3.updateTimer(!vCC);}
void readInputs(){
vINIT=digitalRead(INIT);
vDCY=digitalRead(DCY);
vCC=digitalRead(CC); }
void computeTrans(){
vT0=vE0 && vDCY;
vT1=vE1 && Timer1.getTimerOutput();
vT22=vE2 && Timer2.getTimerOutput() && vCC;
vT21=vE2 && Timer2.getTimerOutput() && !vCC; }
void updateEtapes(){
if(vINIT==1){vE0=1;vE1=0;vE2=0;}
if(vT0==1){vE1=1;vE0=0;}
if(vT1==1){vE1=0;vE2=1;}
if(vT21==1){vE1=1;vE2=0;}
if (vT22==1){vE2=0;vE0=1;}}
```

## 3.4.Communication avec l'IHM par l'utilisation de la librairie ModBus

Le codage d'un grafcet avec Arduino ne nécessite pas forcément une libraire particulière comme celles vues précédemment, puisque avec des simples commandes de gestion des E/S une telle opération devient possible.

On utilisera également dans cette partie la bibliothèque Modbus pour la communication entre les périphériques ou entre le maître et l'esclave.

### 3.4.1. Étapes à suivre

Pour réaliser le projet, on suit les étapes suivantes :

- Tracer le grafcet de système selon le cahier de charges.
- Editer le circuit électrique du système par logiciel proteus.
- Programmer le grafcet par Arduino suivant le brochage effectué dans le circuit au proteus et configurer le COM.
- Réaliser le pupitre de commande par le logiciel easybuilder et configurer les adresses, les périphériques et le COM.
- Copier le fichier HEX du programme Arduino et coller le dans le fichier programme (program file) de la carte Arduino dans proteus.
- Créer une paire de ports séries virtuels via vspd entre les deux COM précédemment configurés.
- Lancer la simulation de proteus puis celle d'easybuilder.

### 3.4.2. Commandes E/S Arduino [30]

Le tableau suivant regroupe quelques instructions Arduino largement utilisées pour la gestion et commande des E/S :

| Instruction | Description |
|---|---|
| pinMode() | Configurer une broche comme entrées ou sortie |
| digitalRead() | Lire la valeur de l'entrée numérique soit HIGH ou LOW |
| digitalWrite() | Ecrire une valeur HIGH ou LOW dans une sortie numérique |

| | |
|---|---|
| analogReference() | Configurer la tension de référence utilisée par les entrées analogique |
| analogWrite() | Envoyer la valeur analogique à la sortie spécifiée (PWM) |
| analogRead() | Lire la valeur de l'entrée analogique spécifiée |
| delay() | Suspendre le programme pendant une durée (en millisecondes) |
| millis() | Renvoyer le nombre de (ms) écoulées depuis que la carte Arduino a commencé à exécuter le programme |
| *Serial*.begin() | Définir le débit de données en bits par seconde (bauds) pour la transmission de données série. |
| Serial.print()<br><br>Serial.println() | Imprimer les données sur le port série sous forme de texte ASCII lisible par l'homme. |

Tableau 3 instructions Arduino pour la gestion et commande des E/S

### 3.4.3. Librairie Modbus RTU [31]

Les bibliothèques modbus, modbusDevice, modbusRegbank et modbusSlave permettent de communiquer avec les appareils Modbus Master en utilisant les interfaces série RS232, RS485 et USB, via le protocole de codage des messages RTU. Ce protocole fournit la structure de requête qui permet à l'esclave de répondre à la requête du maître conformément à la spécification Modbus.

Les paramètres de communication Modbus tels que le débit en bauds série et l'ID d'adresse de l'esclave Modbus sont définis dans le programme Arduino.

#### 3.4.3.1. ID d'adresse esclave Modbus [32]

Les registres modbus suivent le format suivant :

- 00001-09999 : Sorties numériques où un appareil maître peut lire et écrire.
- 10001-19999 : Entrées numériques où un appareil maître ne peut que lire les valeurs de ces registres.
- 30001-39999 : Entrées analogiques où un appareil maître ne peut que lire les valeurs de ces registres.

- 40001-49999 : Sorties analogiques où un appareil maître peut lire et écrire dans ces registres.

#### 3.4.3.2. Instruction bibliothèque Modbus RTU [32]

Dans la bibliothèque Modbus RTU slave on trouve les instructions suivantes :

| **Instruction** | **Description** |
|---|---|
| modbusDevice regBank | Toutes les données accumulées seront stockées ici. |
| modbusSlave slave | Pour créer le gestionnaire de protocole esclaveModbus. |
| regBank.setId() | Pour attribuer un identifiant au dispositif Modbus. |
| regBank.add() | Ajouter un registre à la banque des registres |
| slave_device=&regBank | Affecter l'objet d'appareil modbus au gestionnaire protocole C'est ici que le gestionnaire de protocole lira et écrira. |
| slave.setBaud() | Initialisez le port série pour les communications à un débit en bauds (de 300 à 2000000 bauds) |
| regBank.set() | Mettre des données dans le registre |
| regBank.get() | obtenir une valeur d'un registre |
| slave.run() | Lancer la communication |

Tableau 4 Instruction bibliothèque Modbus

### 3.4.4. Cahier des charges

Appuyant sur le bouton DCY deux vérins1 et 2 sortent, et deux LED verte et rouge s'allument.

A la fin de sortie des deux vérins détectée par les capteurs fsv1 et fsv2, le vérin1 rentre directement et la LED verte s'éteint. Juste que le vérin 1 rentre complètement (fev1) le vérin2 rentre aussi et la LED verte s'éteint. Le cycle revient à l'étape initiale après la fin d'entrées du verin2 Les deux vérins sont double effets.

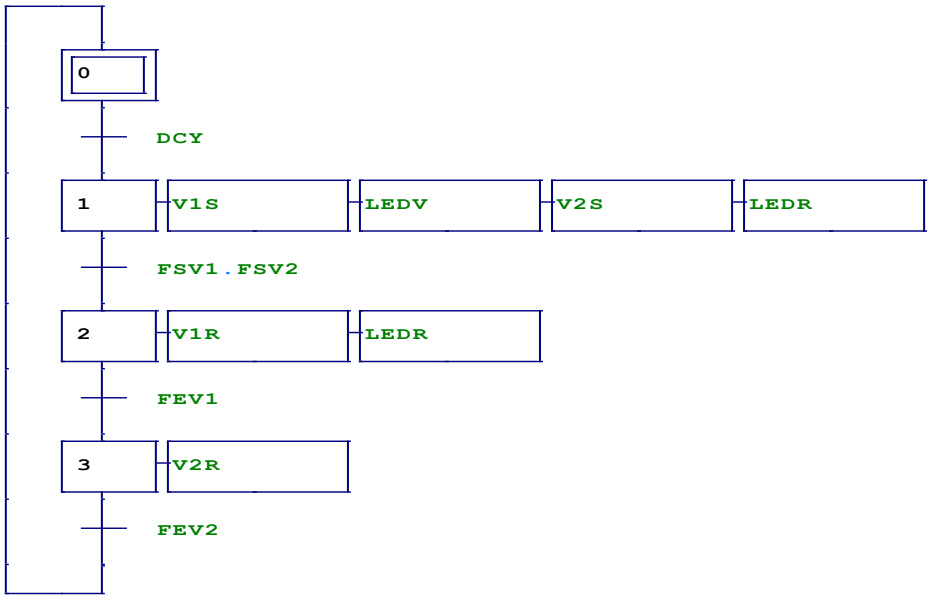



Figure 3. 5 Grafcet du système

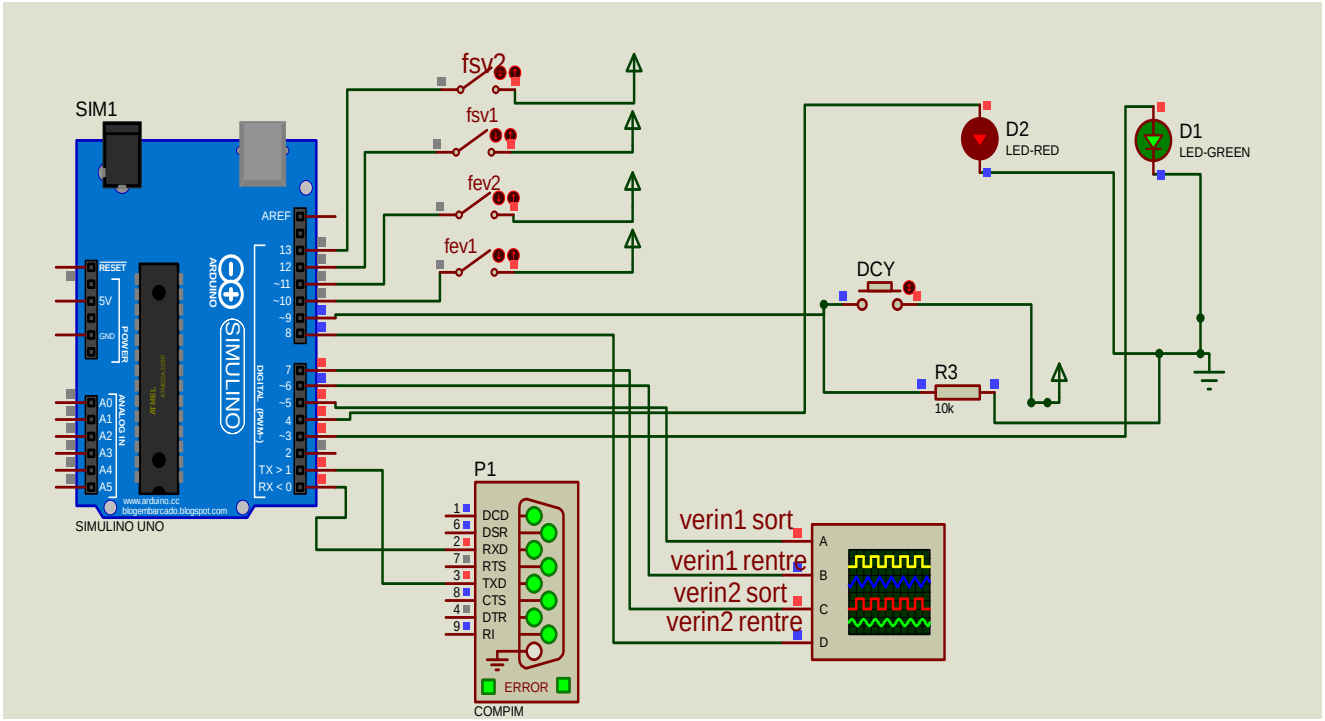



Figure 3. 6 Circuit sous Proteus

### 3.4.5. Le programme Arduino

Ce programme représente un grafcet linaire sans utilisation d'aucune librairie particulière de grafcet mais avec ce programme permet de communiquer avec l'IHM grâce aux librairies modbus RTU

```
#include <modbus.h>
#include <modbusDevice.h>
#include <modbusRegBank.h>
#include <modbusSlave.h>

modbusDevice regBank;
modbusSlave slave;

//les etapes:
int step0; int step1; int step2;int step3;
//les entrees:
int dcy = 9; int fev1 = 10; int fev2 = 11;int fsv1=12;int fsv2=13;
//les sorties:
int v1s = 5; int v1r = 6; int v2s = 7; int v2r = 8;
int ledv = 3; int ledr = 4;

void setup() {

regBank.setId(1);
    regBank.add(1); //dcy
    regBank.add(2);//fev1
    regBank.add(3); //fev2
    regBank.add(4);//fsv1
    regBank.add(5);//fsv2

     regBank.add(10001);//ledv
     regBank.add(10002);//ledr



  //affectations des pins:
  pinMode(dcy, INPUT);
  pinMode(fev1, INPUT);
  pinMode(fev2, INPUT);
  pinMode(fsv1, INPUT);
  pinMode(fsv2, INPUT);
  pinMode(v1s, OUTPUT);
  pinMode(v1r, OUTPUT);
  pinMode(v2s, OUTPUT);
  pinMode(v2r, OUTPUT);
  pinMode(ledv, OUTPUT);
  pinMode(ledr, OUTPUT);



  //etats des sorties a 0v ou etat initial:
  digitalWrite(v1s, LOW);
  digitalWrite(v1r, LOW);
  digitalWrite(v2s, LOW);
  digitalWrite(v2r, LOW);
  digitalWrite(ledv, LOW);
  digitalWrite(ledr, LOW);
```

```
  //initialisation des etapes:
  step0 = HIGH;
  step1 = LOW;
  step2 = LOW;
  step3=LOW;

 slave._device = &regBank;
  slave.setBaud(9600);


}

void loop() {


  //lecture des pins:
  dcy = digitalRead(9);
  fev1 = digitalRead(10);
  fev2 = digitalRead(11);
  fsv1 = digitalRead(12);
  fsv2 = digitalRead(13);


   dcy=regBank.get(1);
 fev1=regBank.get(2);
 fev2=regBank.get(3);
fsv1=regBank.get(4);
fsv2=regBank.get(5);

  //allumage des LED  dans l'ihm:

 regBank.set(10001,step1);
 regBank.set(10002,step1 | step2);


  //equations logiques des etapes:
  step0 = (step0 | (step3 && fev2 )) && (!step1);
  step1 = ((step1 | (step0 && dcy ))) && (!step2);
  step2 = ((step2 | (step1 && fsv1 && fsv2))) && (!step3);
  step3 = ((step3 | (step2 && fev1))) && (!step0);

  //les actions:
  if (step0 == HIGH) {
    digitalWrite(v1r, LOW);
    digitalWrite(v2r, LOW);
    digitalWrite(ledr, LOW);
    digitalWrite(ledv, LOW);
    digitalWrite(v1s, LOW);
    digitalWrite(v2s, LOW);


  }
  if (step1 == HIGH) {
   digitalWrite(v1r, LOW);
    digitalWrite(v2r, LOW);
    digitalWrite(ledr, HIGH);
    digitalWrite(ledv, HIGH);
    digitalWrite(v1s, HIGH);
    digitalWrite(v2s, HIGH);
  }
  if (step2 == HIGH) {
```

```
    digitalWrite(v1r, HIGH);
    digitalWrite(v2r, LOW);
    digitalWrite(ledr, HIGH);
    digitalWrite(ledv, LOW);
    digitalWrite(v1s, LOW);
    digitalWrite(v2s, LOW);

  }
    if (step3 == HIGH) {
     digitalWrite(v1r, LOW);
    digitalWrite(v2r, HIGH);
    digitalWrite(ledr, LOW);
    digitalWrite(ledv, LOW);
    digitalWrite(v1s, LOW);
    digitalWrite(v2s, LOW);

  }
slave.run();
}
```

### 3.4.6. Pupitre de commande

Notre pupitre consiste à effectuer la commande par les entrées et à la visualisation de l'état des sorties.

Où :

-dcy, fev1, fev2, fsv1 et fev2 sont des interrupteurs utilisés comme des entrées

-Etat du vérin 1 et Etat du vérin 2 sont des voyants utilisés comme des sorties.

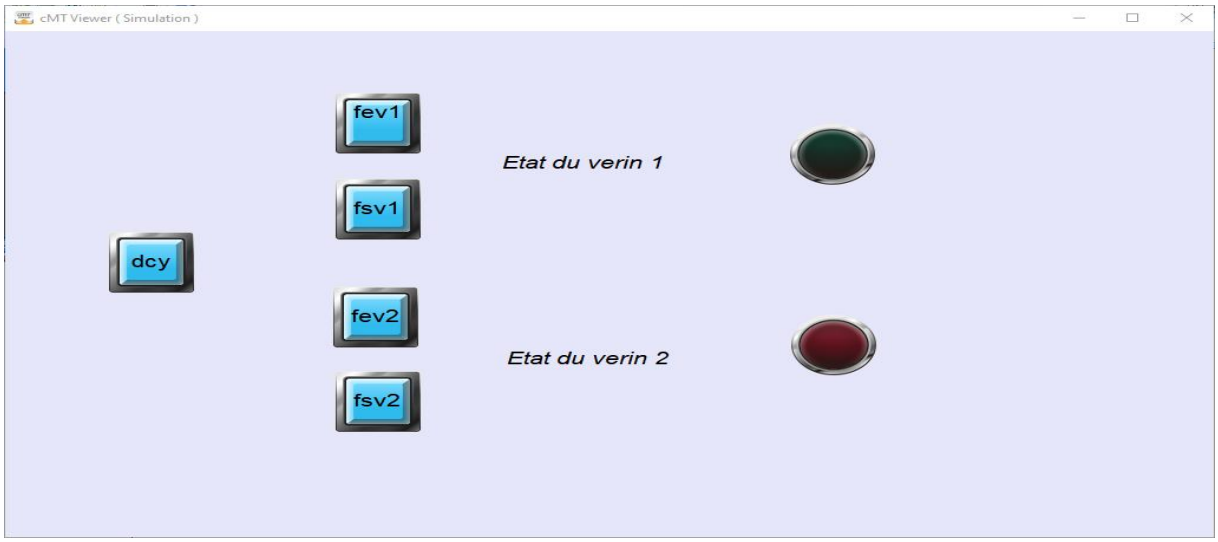



Figure 3. 7 Pupitre de commande

## 3.5.Conclusion :

Dans ce chapitre on a expoité de differentes librairies pour simplifier la programmation des grafcets,chaque librairie a ses avantages et sa façon de coder le grafcet.

Comme on peut aussi coder un grafcet sans inclure des librairies partuculiéres dans le programme mais seulement avec utilisation des équations logiques d'etats des étapes grafcets, en tout cas cette façon nous donne une possibilité d'exploiter les registres d'adresses de la librairie modbus,contrairement au autres librairies utilisés.

La librarie modbus permet une communication avec IHM grace à ces registres d'adresses , qu'à travers on peut lire et écrire ou écrire dans des données numeriques ou anlogiques.

# CHAPITRE 4 :

## PROGRAMMATION D'UN BRAS MANIPULATEUR PAR ARDUINO

## 4.1.Introduction

Aujourd'hui l'intégration des robots dans l'industrie est devenue primordiale grâce à leurs atouts, notamment en termes de sécurités, productivité, rapidité d'exécution et précision.

Un robot industriel est un système ayant plusieurs axes à l'image d'un bras humain souvent composé de six degrés de liberté, trois axes destinés au positionnement et trois axes à l'orientation permettant de déplacer et d'orienter un outil (organe effecteur) dans un espace de travail donné. [33]

Le choix de ces robots et leurs nombres d'axes se réfère à l'espace de travail et la tâche voulue, en tenant compte de l'optimisation des robots.

La robotique est un concentré de technologies complexes et variées faisant appel à différents domaines tel que la programmation, la mécanique, la mécatronique, la cognatique, le design…, toutefois ils sont doté d'actionneurs et préhenseurs tels que les : moteurs, servo-moteurs, les vérins, les pinces et ventouses, mais aussi les capteurs jouant un rôle d'émission et réception d'informations. ces robots et servomoteurs sont pilotés par des logiciels de programmation grâce à une carte électronique qui lui ont été insérer tel que la carte Arduino. [34]

## 4.2.Cahier des charges

Dans un système composé de deux postes A et B pour le perçage de plaques de verre. Un robot fixe organise à lui seul les opérations chargement de pièces d'un convoyeur à l'un des deux postes ; ainsi que le déchargement à la fin de traitement des pièces de l'un des deux postes vers un autre convoyeur conçu pour l'évacuation.

Les opérations et les taches du robot citées précédemment sont soumis à respecter quelques priorités afin ordonner et gérer la procédure de production et de traitement.

Ces priorités sont comme suite :

- L'évacuation des plaques de verre de l'un des deux postes est prioritaire.

- Le poste A est prioritaire par rapport au poste B dans les deux opérations soit pour lui ramener les plaques ou pour les évacuer.

Il faut savoir que le poste A et B sont similaires, ils font le même traitement et passe la même durée dans le perçage des plaques du verre.

Dans la figure suivante on expose la répartition des postes et des convoyeurs ainsi que la position du robot.

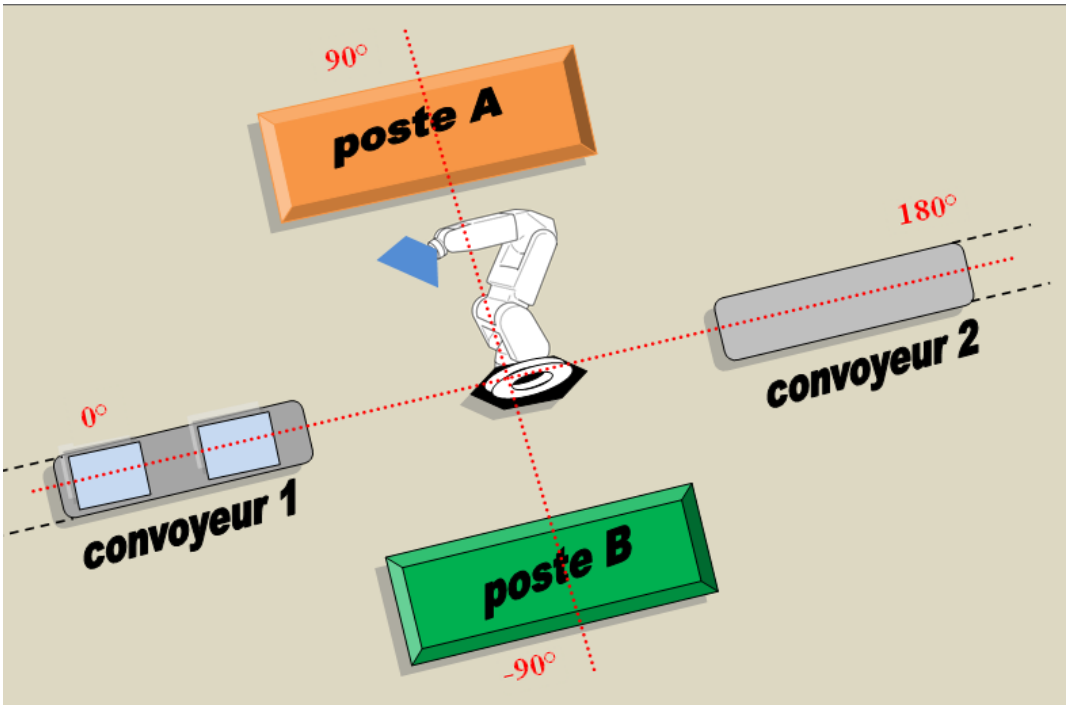



Figure 4. 1 Positions robot, convoyeurs et postes de traitement

## 4.3.Décortication des composants du système

### 4.3.1. Les postes

Les postes A et B sont similaires, les deux ont le même outil de perçage et les mêmes types de capteurs.

La tâche d'un poste est d'effectuer un perçage avec scie cloche dans les plaques de verre afin de permettre d'installer les systèmes d'aération. La perceuse est en marche toujours et elle est portée par un vérin double effets doté des fin de courses dans ses extrémités et la fin d'entrée de ce vérin est considéré ici comme le capteur de fin de traitement dans le poste.

L'opération s'effectue sur une table située en dessous des systèmes de perçage, le traitement commence dès que le capteur de présence de plaque de verre installé sur la table détecte un front montant la pièce.

### 4.3.2. Les convoyeurs

Dans ce système existent deux convoyeurs :

Le premier ramène les plaques de verre une par une .il est entrainé par un moteur pas à pas qui change de position à chaque fois que le capteur de plaque de verre installé à l'extrémité du convoyeur signale une absence de pièce.

Le deuxième convoyeur évacue les plaques après la fin de leur traitement dans l'un des deux postes. Il est entrainé par un moteur asynchrone qui en marche d'une façon continue.

### 4.3.3. Le robot manipulateur

Afin que le robot puisse atteindre les quatre positions en occurrence le convoyeur 1 supposé à la positon 0°, le convoyeur 2 à la position 180°, les deux poste A et B aux position 90° et -90° respectivement, on aura besoin d'un robot rotatif composé de deux servomoteurs qui tourne de -90° à +90°.un pour l'axe et l'autre pour le bras.

Le bras du robot est porte un vérin pneumatique en plus d'un servomoteur tournant de 0° à +90 pour remonter et redescendre le support des ventouses afin d'assurer la sécurité des opérations chargement et déchargement et alléger la charge sur les ventouses.

Les ventouses sont les effecteurs de ce robot ; elles tirent et relâchent les plaques en utilisant l'effet venturi.

L'automatisme précédent permet d'atteindre toutes les positions ; de descendre à la hauteur des postes et des convoyeurs, puis tirer la plaque de verre par les ventouses ; la remonter pour ensuite la déplacer en toute sécurité.

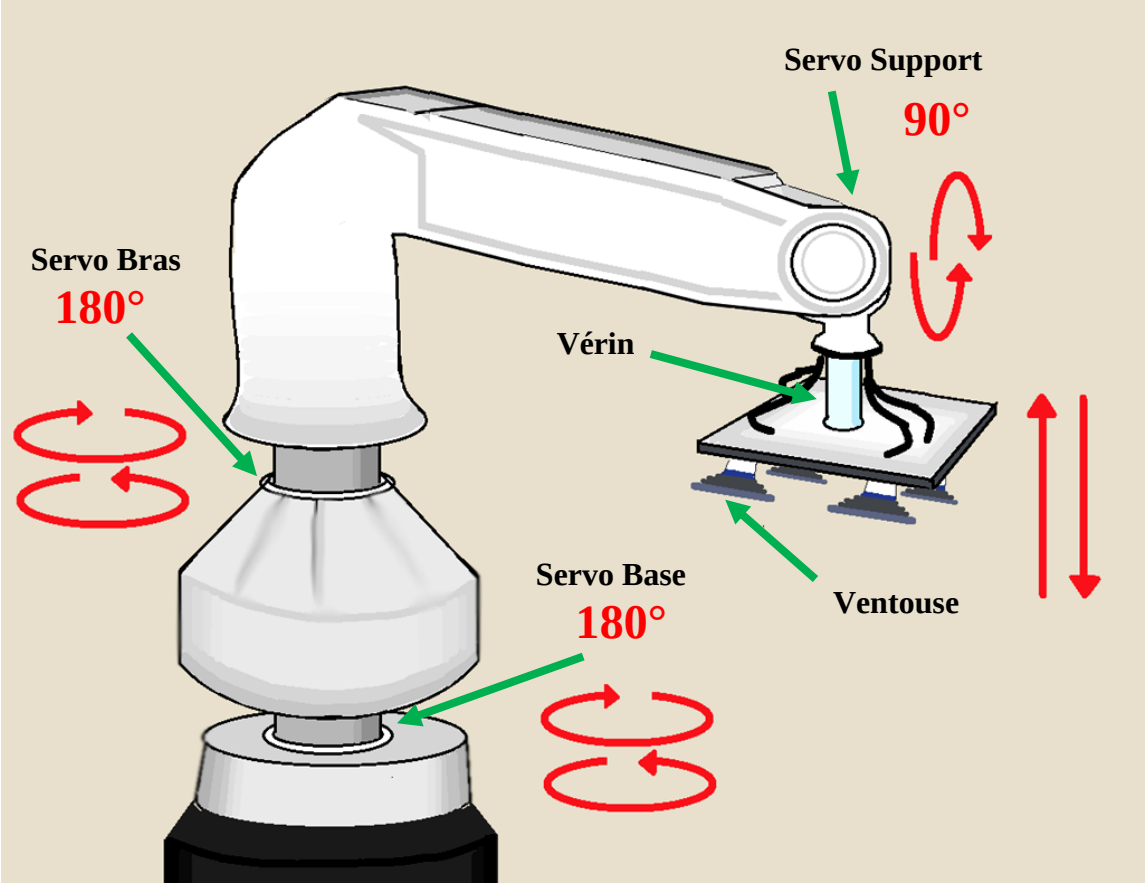



Figure 4. 2 Robot manipulateur utilisé dans le système

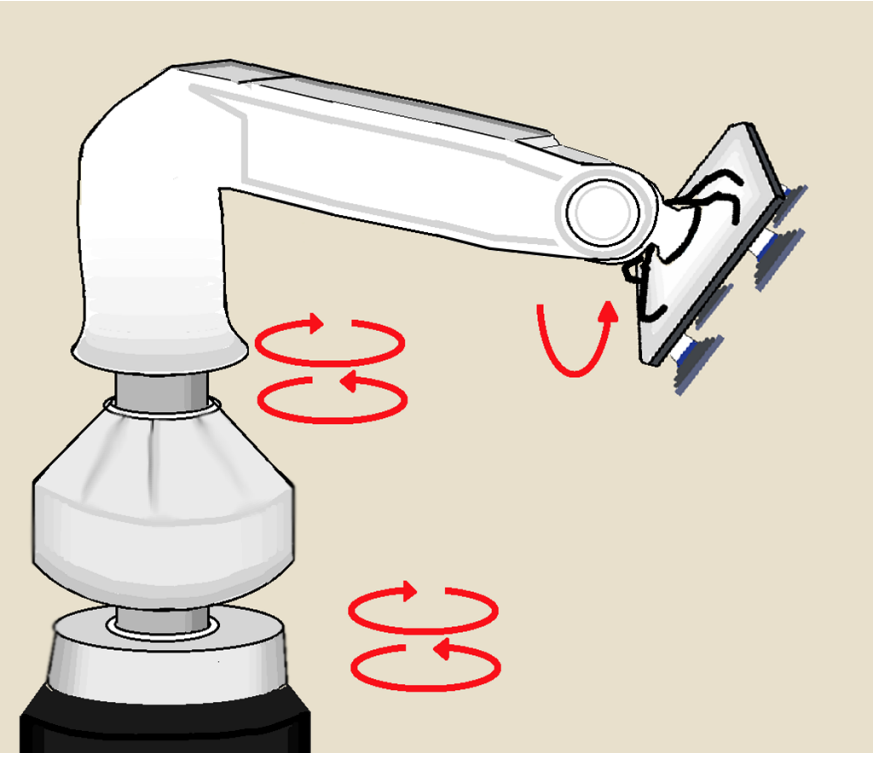


Figure 4. 3 Robot du système support en position

## 4.4. Grafcet du système

Le fonctionnement du système proposé sera traduit sous forme grafcet, pour ensuite le coder en programme Arduino toute en respectant le cahier des charges exigé, trois grafcets sont proposés ; un grafcet maître pour diriger les actions de positionnement et deux autres pour effectuer les opérations chargement et déchargement de plaques.

### 4.4.1. Les entrées

Le tableau suivant présente les différentes entrées utilisées :

| Entrées | Numéro de pin | Description |
|---|---|---|
| **FEV** | 2 | Fin d'entrée du vérin |
| **FSV** | 3 | Fin de sortie du vérin |
| **PPB** | 4 | Présence de pièce au poste B |
| **B** | 5 | Fin de traitement au poste B |
| **PPA** | 6 | Présence de pièce au poste A |
| **A** | 7 | Fin de traitement au poste A |

Tableau 5 Entrées du système

### 4.4.2. Les sorties

Le tableau suivant présente les différentes sorties utilisées :

| Sorties | Numéro de pin | Description |
|---|---|---|
| **AC** | 25 | Alimentation du clapet |
| **AGV** | 24 | Alimentation du générateur du vide |
| **ServoBase** | 10 | Servomoteur du tronc robot |
| **ServoBras** | 11 | Servomoteur du bras |
| **ServoSupport** | 12 | Servomoteur du support |
| **EV** | 23 | Bobine d'entrée du vérin |
| **SV** | 22 | Bobine de sortie du vérin |

Tableau 6 Sorties du système

### 4.4.3. Grafcet principale du robot

Le grafcet principal suivant répond totalement au cahier des charges puisque il étudie tout les cas possible et permet la succession logique des opérations de chaque partie constituante du robot en occurrence les servomoteurs, le vérin qui porte le support des ventouses et les ventouses elles-mêmes.

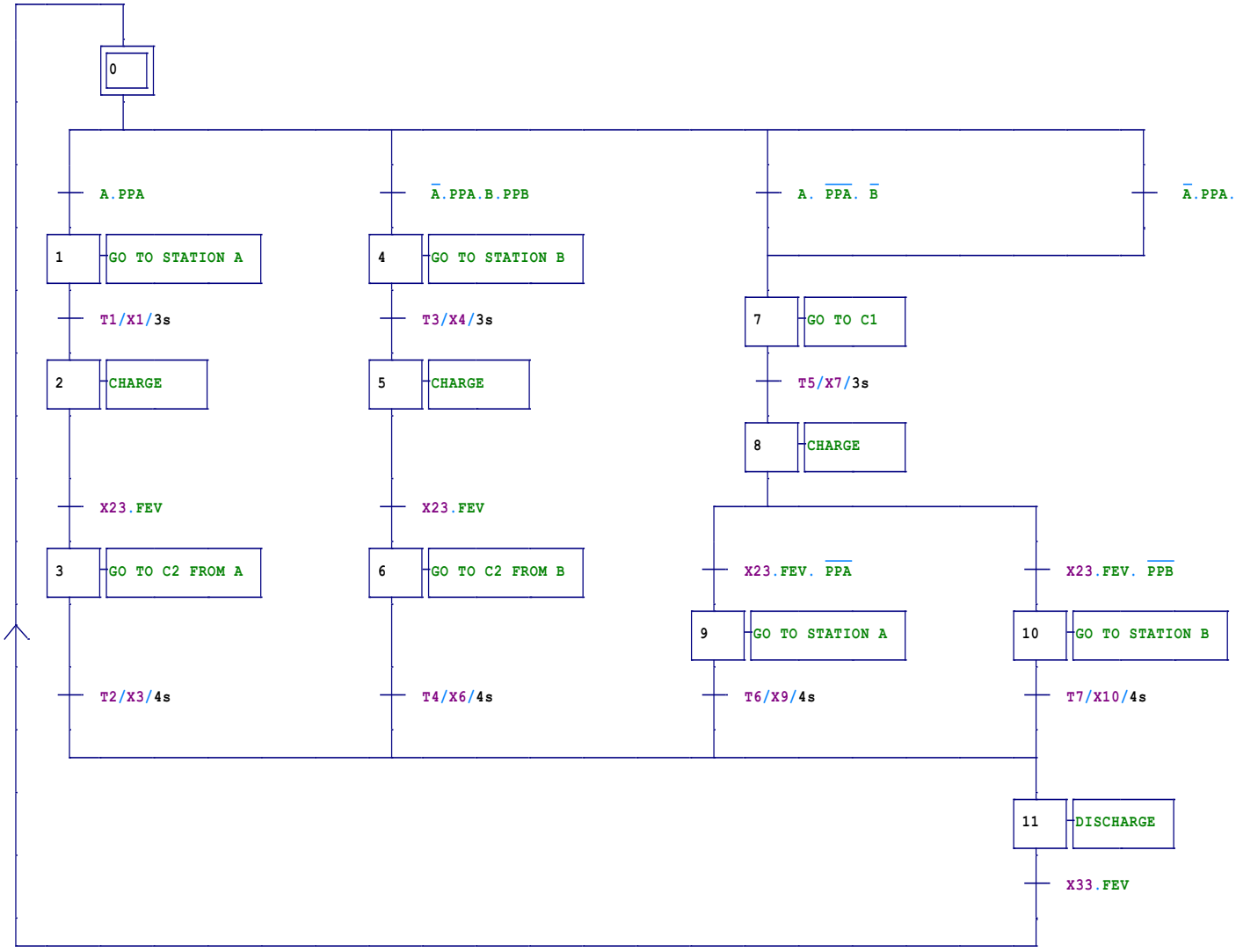



Figure 4. 4 Grafcet principale du robot

### 4.4.4. Grafcet chargement

Le sous-grafcet de chargement est sollicité à chaque fois que le grafcet maître active l'action CHARGE.

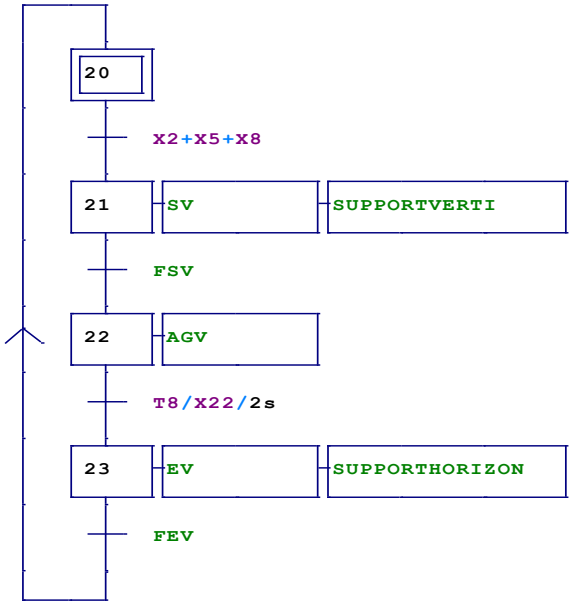



Figure 4. 5 Grafcet de chargement

### 4.4.5. Grafcet déchargement

Le sous-grafcet de chargement est sollicité à chaque fois que le grafcet maître active l'action DISCHARGE.

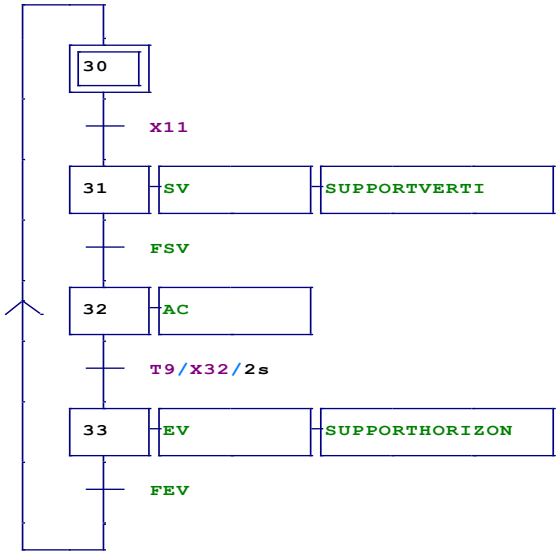



Figure 4. 6 Grafcet déchargement

### 4.4.6. Les composants utilisés :

Dans le circuit du robot élaboré avec proteus, on a utilisé les composants suivant :

-Une carte Arduino MEGA 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.

-Des servomoteurs : Ce sont des servomoteurs PWM ils ont trois broches, l'une d'elles va vers Vcc, l'autre à GND tandis que la broche centrale est la broche de contrôle et va à n'importe quelle broche numérique PWM de la carte Arduino. Dans la boite de paramètres et de configuration des servomoteurs, on règle les angles Min et Max ainsi que la vitesse de rotation et les largeurs d'impulsions Min et Max.

-Des résistances : Des Résistances 10kΩ pour les interrupteurs et d'autres à 100Ω pour allumage des LED.

-Des LEDs : Elles sont utilisées pour indiquer l'état des sorties, elles s'allument si l'action est active et elles s'éteignent dans le cas contraire, donc elle sert juste comme voyant indiquant l'excitation des bobines de commande des distributeurs.

-Des interrupteurs : Elles prennent la place des capteurs TOR.

-Des alimentations et des mises à la terre :

-Le COMPIM : Afin de faire une communication série virtuelle entre la carte et le pupitre de commande, on indique le COM et la vitesse de communication, le nombre de bits de données et de bits de parité.

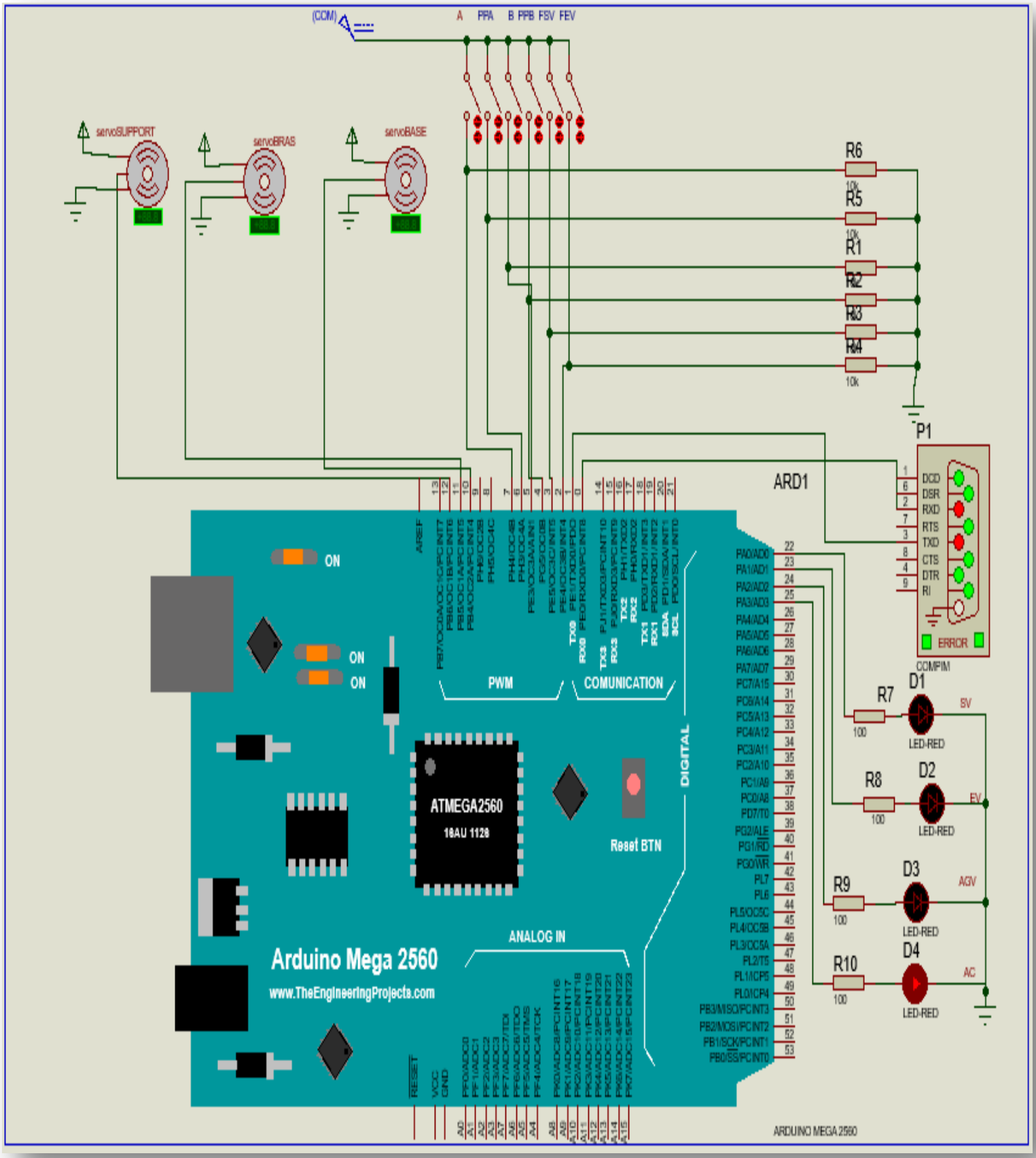



Figure 4. 7 Circuit sous Proteus

## 4.5.Le pupitre de commande

Le pupitre de commande du système est élaboré avec le logiciel Easybuilder Pro. A travers trois fenêtres on visualise l'état d'avancement du processus, l'état des capteurs et les actions effectuées à chaque instant.

### 4.5.1. Fenêtres des entrées sorties

Cette fenêtre montre les états des capteurs et des actions effectuées par le robot.

- Les cases vertes et bleues sont des afficheurs format numérique avec registres d'adresses de lecture des entrées analogiques, afin d'afficher les numéros des étapes et les chronomètres des temporisations.
- Les voyants verts indiquent les états d'actions, ont des adresses de lecture et d'écritures des valeurs numériques.
- Les voyants rouges affichent les états des capteurs, ils portent des adresses de lecture et d'écritures.
- Les cases jaunes sont des touches fonctions pour permettre à l'utilisateur de se déplacer entre fenêtres.

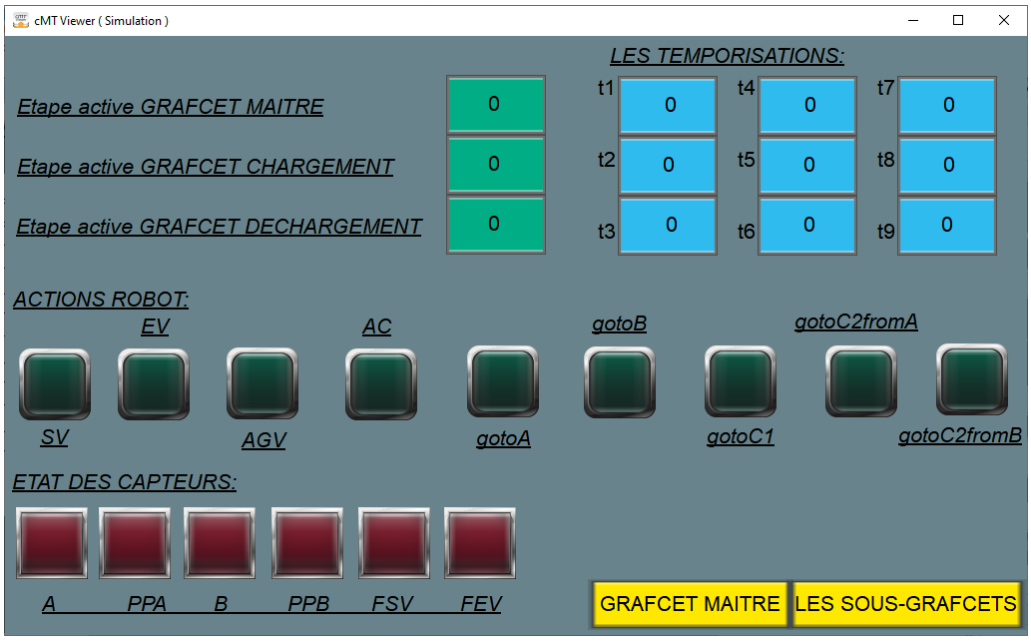



Figure 4. 8 Fenêtres des entrées sorties

### 4.5.2. Fenêtres du grafcet maître

Cette fenêtre représente seulement les opérations concernant le grafcet maître, on y trouve l'image du grafcet maître élaboré en Automgen, le numéro d'étape active du grafcet maître et ses propres les temporisations.

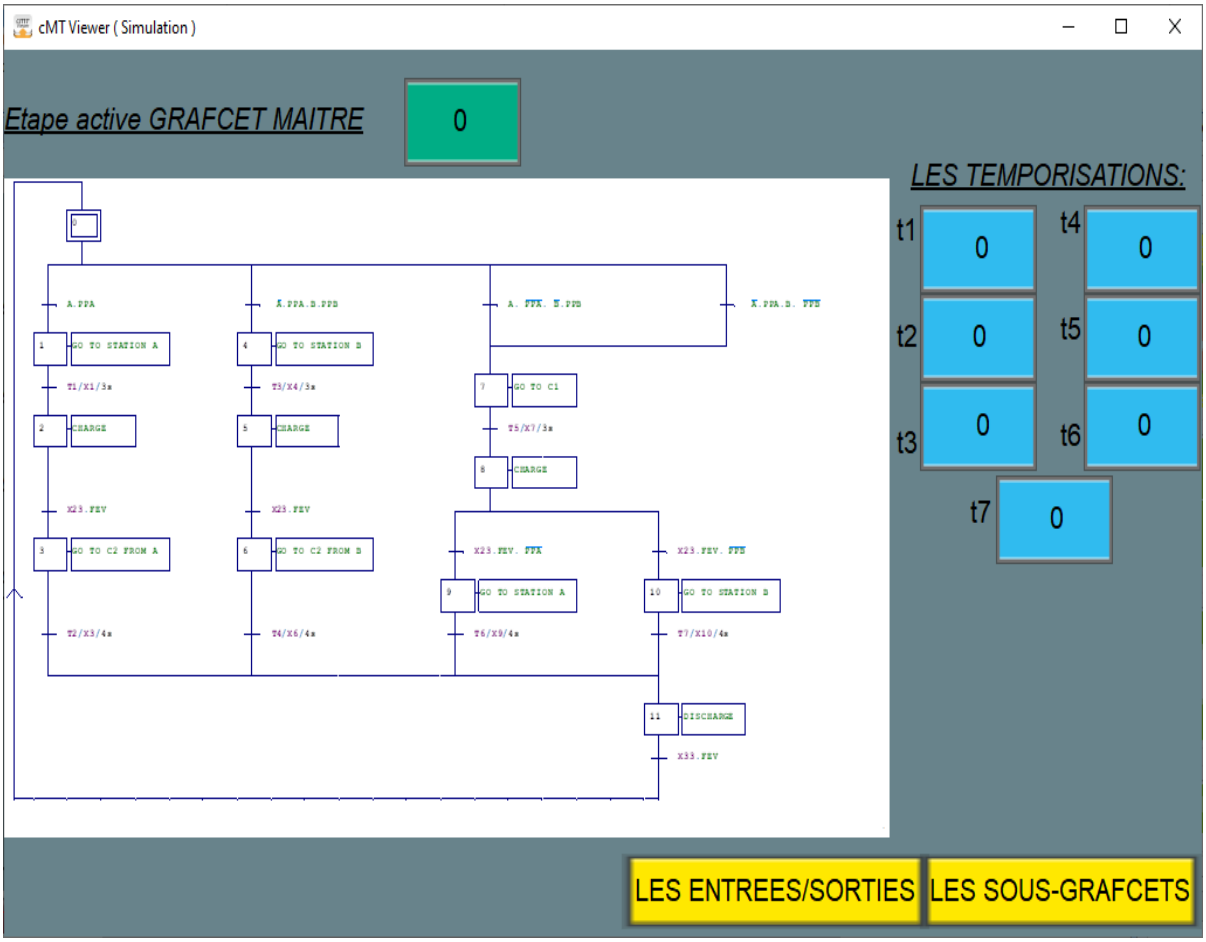



Figure 4. 9 Fenêtres du grafcet maître

### 4.5.3. Fenêtres des sous-grafcets

Cette fenêtre représente les opérations concernant les sous-grafcets on occurrence le grâce chargement et déchargement, on y trouve les images des deux grafcets élaboré en Automgen, le numéro des étapes actives et la temporisation de chaque un.

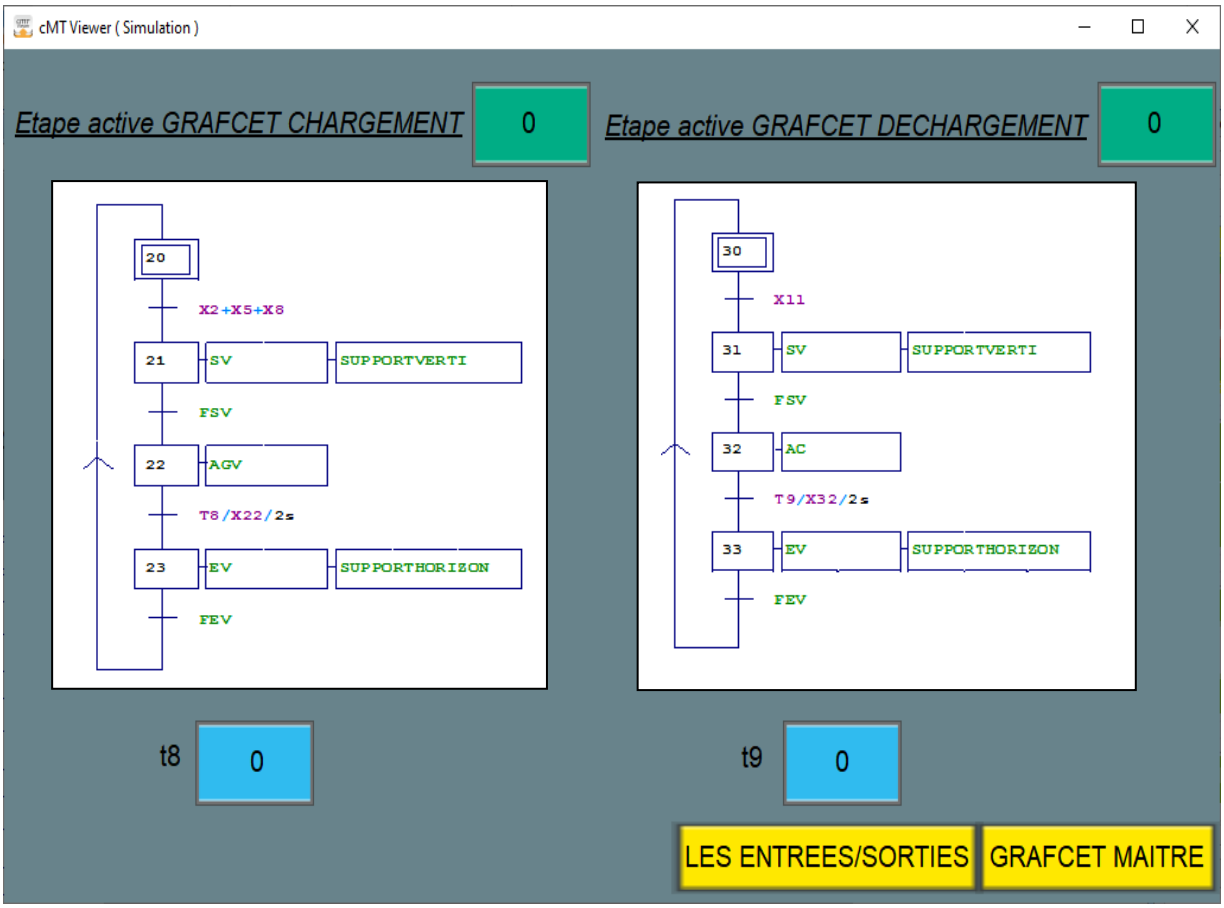



Figure 4. 10 Fenêtres des sous-grafcets

## 4.6.Programmes Arduino du système

Le programme Arduino est un code en langage C qui permettra d'automatiser les taches du robot telle quelles sont définie dans les grafcets.

Le programme se basera essentiellement sur quatre fonctions principales :

- Lecture des entrées de la carte Arduino ;
- Calcule et traitement des informations entrantes ;
- Délivrance des résultats sous forme sorties ;
- Communication avec l'interface homme machine ;

### 4.6.1. Étude du programme par partie

Comme la majorité des programmes Arduino il sera reparti en cinq parties :

- Inclusion des librairies ;
- Déclaration des variables ;
- Initialisation et configuration (la fonction setup()) ;
- Programme principal en boucle (la fonction loop()) ;
- Création des fonctions ;

#### 4.6.1.1. Inclusion des librairies

Pour que faciliter la programmation en aura besoin d'utiliser deux librairies : un pour librairie modbus RTU et l'autre pour les servomoteurs .c'est deux librairies feront appelle à l'exécution d'autre programme externe elles donnent l'accès à l'utilisation de ses propre variables.

```
#include <modbus.h>
#include <modbusDevice.h>
#include <modbusRegBank.h>
#include <modbusSlave.h>
#include <Servo.h>
```

#### 4.6.1.2. Déclaration des variables :

Dans le programme on utilisera des différentes variables, pour les calcules logiques ont utilisera des variables booléennes pour uniformiser les types des calculs et des résultats.

```
bool X0,X1,X2,X3,X4,X5,X6,X7,X8,X9,X10,X11;// grafcet maitre
bool XP0,XP1,XP2,XP3,XP4,XP5,XP6,XP7,XP8,XP9,XP10,XP11;
bool A, PPA, B, PPB, FSV, FEV;
bool SV, EV, AGV, AC;
```

Les variables de type entier sont utilisées pour indiquer les numéros des pins de brochage des entrées et sorties.

```
int A_pin   = 7;//capteur de fin de traitement au poste A
int PPA_pin = 6;//capteur de presence de piece au poste A
int B_pin   = 5;//capteur de fin de traitement au poste B
int SV_pin  = 22;//alimentation de la bobine de distributeur pour sortir le verin
int EV_pin  = 23;//alimentation de la bobine de distributeur pour entrer le verin
```

Les valeurs des temporisations sont des variables longues non signées .elles sont des variables de taille étendue pour le stockage des nombres de 32 bits.

```
unsigned long T1_value = 3000, T2_value = 4000, T3_value = 3000, T4_value = 4000, T5_value = 3000;
unsigned long T6_value = 4000, T7_value = 4000, T8_value = 2000, T9_value = 2000;
unsigned long T1_begin, T2_begin, T3_begin, T4_begin, T5_begin, T6_begin, T7_begin, T8_begin, T9_begi
```

Déclaration des servomoteurs se fait aussi avec appel des variables locales librairie précédemment inclus.

```
Servo servoBase ;
Servo servoBras ;
```

#### 4.6.1.3. Initialisation et configuration (la fonction setup())

Dans cette fonction on initialise tous les paramètres du système :

-Brochage des servomoteurs et initialisation des postions.

```
servoBase.writeMicroseconds(1500);//0°
servoBras.writeMicroseconds(1500);//0°
```

-Configurations des entrées et sorties

```
pinMode(A_pin,    INPUT);
pinMode(PPA_pin,  INPUT);
pinMode(SV_pin,   OUTPUT);
pinMode(EV_pin,   OUTPUT);
```

-Rajout des registres modbus selon le format souhaité, soit de point de vue types de signaux (numérique ou analogique) et de point de vue instructions (lecture et/ou écriture)

```
regBank.add(30001); // numéro étape active dans le grafcet maitre
regBank.add(30002); // numéro étape active dans le grafcet chargement
regBank.add(30003);  //  numéro  étape  active  dans  le  grafcet  déchargement

regBank.add(30011);//temporisation t1
regBank.add(30012);//temporisation t2
regBank.add(30013);//temporisation t3

regBank.add(10001);//capteur A
regBank.add(10002);//capteur PPA
regBank.add(10003);//capteur B
regBank.add(1);//voyant SV
regBank.add(2);//voyant EV
regBank.add(3);//voyant AGV
```

-Définition de la vitesse de communication en baud entre les entités.

```
//vitesse de communication:
slave.setBaud(19200);
```

#### 4.6.1.4. Programme principal en boucle (la fonction loop())

C'est le programme principal qui va s'exécuter en boucle infinie. Il permettra à la carte de se comporter en automate programmable dans l'évolution séquentielle du grafcet.

-Comme toute automate, on commence par lecture des entrées pour en suite utiliser les valeurs obtenu dans la suite du programme.

```
A   = digitalRead(A_pin);
PPA = digitalRead(PPA_pin);
```

-A partir des entrées, on calcule les réceptivités telle quelles sont classées dans le grafcet, à chaque cycle d'exécution l'automate vérifie si ces réceptivités sont vraies ou fausses pour ensuite utiliser les résultats obtenu dans l'activation de l'étape qui suit l'une de ces réceptivités.

A titre d'exemple la réceptivité qui précède l'étape 0 du grafcet maître est appelé r0 et son équation est comme suite :

```
r0  = XP33 && FEV;
r1  = A    && PPA;
```

- Ensuite calcul d'activation d'étape. L'activation d'une étape dans un grafcet est obtenue à partir du résultat d'un ET logique entre l'étape précédente, la réceptivité et l'inverse de l'initialisation du grafcet.

Dans le grafcet maître, l'activation de l'étape1 dépend de l'état de l'etape0, de l'état de la réceptivité r1 et le INIT comme l'indique l'exemple suivant.

```
S1  = XP0 && r1 && !INIT;
```

-Dans un grafcet une seule étape est activée le reste est désactivée, donc dans le programme on doit préciser pour chaque étape les conditions de sont désactivation.

La désactivation d'une étape correspond toujours soit à l'activation de ou des étapes suivante ou activation de l'étape initiale par INIT, comme exemple dans l'étape2 du grafcet maître se désactive si l'étape3 ou INIT est vrai.

```
R2  = XP3  || INIT;
```

-A partir des résultats obtenu précédemment notamment les réceptivités, l'activation et la désactivation d'étape ont obtient l'équation finale de l'état de l'étape.

L'équation de l'étape est sa valeur logique correspond soit à l'état logique précédent ou au état logique de son activation multiplié logiquement par l'état de son désactivation.

La désactivation d'étapes est prioritaire par rapport à son activation pour sécurité du système et pour éviter l'infraction l'une des règles importante des grafcet si on aura plus d'une étape activée dans un même grafcet.

```
X0  = (XP0  || S0)  && !R0;
X1  = (XP1  || S1)  && !R1;
```

-L'activation de l'état de sortie est tout simplement équivalent à l'activation de ou des étapes dont cette sortie est associées.

```
SV  = X21 || X31;
EV  = X23 || X33;
```

-Les états des bits de temporisation sont considérés aussi comme entrées mais pas comme les états d'entrées obtenues de la partie opérative du système mais se sont des états calculés intérieurement dans l'automate.

Les temporisations utilisées dans ce système sont des temporisations de retard à la montée. C'est-à-dire le bit correspondant est faux initialement, il sera vrai après l'arrivé de la valeur de comptage au seuil prédéfini.

La procédure suivie dans le programme Arduino est de mémoriser la valeur du chronomètre Arduino dés l'activation de l'étape associée, pour ensuite la comparer avec la valeur final de la temporisation.

```
if (X1 && !XP1) T1_begin = millis() ;
if (X1) t1 = millis()-T1_begin;
T1 = X1 && (t1 > T1_value);
```

-Après la fin de calcul des états dans un cycle d'automate on affecte l'état de l'étape à l'état précédent pour ensuite refaire les calculs en boucle.

```
XP0   = X0;
XP1   = X1;
```

-La phase d'affichage se fait par communication avec l'IHM à travers la librairie modbus, ou chaque objet dans le pupitre correspond à un registre d'adresse ou on lit ou/et on écrit.

```
regBank.set(10001,A);//affichage de l'etat du capteur A
regBank.set(10002,PPA);//affichage de l'etat du capteur PPA
regBank.set(10003,B);//affichage de l'etat du capteur B
regBank.set(10004,PPB);//affichage de l'etat du capteur PPB
regBank.set(10005,FSV);//affichage de l'etat du capteur FSV
regBank.set(10006,FEV);//affichage de l'etat du capteur FEV


regBank.set(1,SV);//adresse du voyant SV
regBank.set(2,EV);//adresse du voyant EV
regBank.set(3,AGV);//adresse du voyant AGV
regBank.set(4,AC);//adresse du voyant AC
```

#### 4.6.1.5. Création des fonctions

Pour éviter de répéter une série d'instructions à chaque fois, on crée des fonctions sou on met ces instruction dedans, leur exécution s'effectue avec appel par le nom de la fonction.

```
void GOTOC2FROMA() {
  servoBase.writeMicroseconds(2000);//+90°
  delay(20);
  servoBras.writeMicroseconds(2000);//+90°
  delay(20);}
```

## 4.7. Conclusion :

Le chapitre est une application récapitulative du travail présenté toute au long des précédents chapitres. À partir d'un cahier de charge imposé au début, on a tracé un grafcet qui résume tout les trajets et opérations du système robot, par la suite il a été codé par un programme Arduino qui permet aussi de communiquer avec le système par un IHM.

L'exemple donné concrétise l'utilisation des logiciels vus, et aussi la façon de les mettre en œuvre ensemble pour commander, simuler et superviser l'évolution d'une chaine.

Le type de chaine traité aussi ouvre les portes aussi sur le domaine de la robotisation, ses avantages et alternatives qu'elle donne pour l'humain dans divers domaine, de l'industrie vers la médecine, l'espace … l'utilisation de la robotique prend de plus en plus d'importance puisqu'elle peut réaliser des taches qui semble impossible pour l'homme.

Le progrès de ce domaine est croissant tant que l'intelligence artificielle et les logiciels de programmation gardent leurs courbes de progression ascendantes.

# Conclusion générale :

La problématique posée à l'introduction générale était de créer une plateforme exemplaire pour automatiser et superviser un système en utilisant les moyens conçus pour résoudre se genre de problèmes.

Toute au long de notre développement on a essayé d' entourer le problème de notre travail en partant des notions générales de l'automatisme et supervision arrivant à une application réelle d'un système robot qui concrétise l'ensemble des outils matériels et logiciels cités dans le corps des chapitres, une chose qu'a été réalisé par utilisation de carte Arduino comme automate , cerveau et chef d'orchestre de l'automatisme , l'easybuilder pro pour programmation des pupitres de commande ainsi que le logiciel proteus pour simuler les différents éléments et circuits électriques et électroniques.

Pour arriver à un résultat satisfaisant on a fait appel au premiers temps à un travail de recherche de chaque partie séparément, pour finalement les mettre efficacement ensemble au service du projet .La manière la plus réputé pour automatiser un système est : à partir du matériel disponible et l'objectif tracé, on adresse un cahier de charge qui caractérise de la manière la plus parfaite le système puis le traduire à un algorithme ou grafcet afin de le programmer en langage compréhensible par les machines .

Pour pouvoir contrôler et visualiser des chaînes de productions, on aura besoin d'installer un système de supervision comme à titre d'exemple un pupitre de commande qui nous donne la possibilité de communiquer avec les entités et d'intervenir quand il est nécessaire, comme on vu dans l'exemple traité.

Ce travail peut ouvrir la porte à d'autres approfondissements dans divers domaines surtout dans celui de la robotique et des langages de programmation des robots manipulateur ;ou aussi l'amélioration de la supervision des système à temps réel pour mieux accompagner des telles chaines de productions ,est cela par utilisation à titre d'exemple de

l’outil matlab v-realm builder pour créer une bibliothèque d’objet similaire au matériel réel présent et de les animer tout en respectant le cahier des charge et le programme d’ automate.

On peut dire qu’à travers ce projet de fin d’étude on a utilisé pas mal de notions et de compétences d’automatismes et de génie électrique globalement, malgré qu’on n’a pas été en stage, mais l’application abordée est une opportunité pour découvrir  d’autres connaissances qu’on n’aurait pas dû voir dans une entreprise, à cause de rareté des robots manipulateurs actuellement dans les industries algériennes, des choses qu’on verra peut être beaucoup plus dans le futur.

# Références bibliographiques .....


[1]. **GUERRAS, LAHCEN.** commande d'un système automatisé de production par automate programmable industriel. [En ligne] 2012-2013. http://dspace.univ-msila.dz:8080/xmlui/handle/123456789/10664.

[2]. **MIKLSTEM, DANIEL.** livre de l'automatisation et technologies émergentes. [En ligne] https://books.google.dz/books?id=RR7NDwAAQBAJ&pg=PT31&dq=livre+de+l%E2%80%99automatisation+et+technologies+%C3%A9mergentes&hl=fr&sa=X&ved=2ahUKEwiYpva5YvrAhVHTxUIHY2JA2cQ6AEwAHoECAAQAg#v=onepage&q=livre%20de%20l%E2%80%99automatisation%20et%20technologies.

[3]. **LAIFAOUI, ABDELKRIM.** Technologies des automatismes. [En ligne] 2016-2017.

[4]. **ATMIMOU LYES, BELFAKED HASSINA.** Place de l' API dans le système automatisédeproduction(S.A.P).[Enligne] https://dl.ummto.dz/bitstream/handle/ummto/7917/AtmimouLyes_BelfakedHassina.pdf?sequence=1&isAllowed=y.

[5]. **UNIV SAIDA.** CHAPITRE I Généralité sur la robotique. [En ligne] https://www.univ-saida.dz/butec/doc_num.php?explnum_id=366.

[6]. **BENSAHAL.** Généralités sur la robotique- introduction aux structures rigides et flexibles. [En ligne] 2011. [Citation : 15 JUIILLET 2020.] Adresse URL: http://thesis.univ-biskra.dz/2421/2/chapitre1.pdf.

[7]. Automation & PLC knowledge center . [En ligne] Adresse URL: https://www.automation-sense.com/pages/les-langages-de-programmations.html.

[8]. **BASSAM, KATTAN.** Synthèse structurelle d'un contrôleur basée sur le Grafcet. Automatique / Robotique. [En ligne] Université Joseph-Fourier - Grenoble I, 2004. Français.

[9]. **AHMED, SKAF.** Etude d'un système de supervision et de commande d'un procédé complexe comme élément de base d'une organisation distribuée comprenant des machines et des hommes. Automatique / Robotique. [En ligne] Université Joseph-Fourier - Grenoble I,, 2001. Français. tel-00198491f.

[10]. **SOUTTOU Chafik, MOULAI Aomer,.** Automatisation et supervision d'un mixeur de produit fini de la boisson gazeuse de l'usine Fruital Coca-Cola. [En ligne] Université Mouloud Mammeri De Tizi-Ouzou, 2017.

[11]. **BELHADJ, BRAHIM CHIKH.** Automatisation et supervision d'une station de Thermolaquage par un automate S7-1200,. [En ligne] université M'hamed Bougara - Boumerdes, Juin 2017.

[12]. **AUTOMATE-PRO.** Cours-supervision-industrielle-Format PPT. *automate-pro.blogspot,*. [Enligne] 2013. Adresse URL :http://automate-pro.blogspot.com/2013/03/cours-supervision-industrielle-ppt.html.

[13]. **DULEX, FREDERIC GENEVEY et JEAN PIERRE.** arduino-a-l-ecole. *arduino.developpez.com.* [En ligne] 9 décembre 2019. https://arduino.developpez.com/tutoriels/arduino-a-l-ecole/.

[14]. **LANDRAULT, SIMON et WEISSLINGER, HIPPOLITE.** Le langage Arduino. *zestedesavoir.com.* [En ligne] 12 juillet juillet 2020. https://zestedesavoir.com/tutoriels/686/arduino-premiers-pas-en-informatique-embarquee/742_decouverte-de-larduino/3418_le-langage-arduino-12/.

[15]. **LANDRAULTt, SIMON et HIPPOLITE, WEISSLINGER et.** Arduino-Premiers-Pas-en-Informatique-Embarquee. *fr.scribd.com.* [En ligne] 19 juin 2014. https://fr.scribd.com/document/263174613/Arduino-Premiers-Pas-en-Informatique-Embarquee.

[16]. **LLP, HOAREAU et LJF, L.DESCHAMPS.** dialogue_reseau_encapsulation.pdf. *ressource.electron.free.fr*.[Enligne] http://ressource.electron.free.fr/tp/reseaux/dialogue_reseau_encapsulation.pdf.

[17]. **SID, ALI LAFIOUNE.** Etude et réalisation d'un banc de démonstration pour la supervision industrielle en utilisant le protocole Modbus RTU. *PROJET DE FIN D'ETUDE.* jijel : UNIVERSITE Mohamed Seddik Ben Yahia, 2019.

[18]. **AUTOMATION-SENSE.** guide-du-modbus-pour-les-nuls-extrait.pdf. *automation-sense.* [En ligne] juin 2016. https://www.automation-sense.com/medias/files/guide-du-modbus-pour-les-nuls-extrait.pdf.

[19]. **OLGA, WEIS.** modbus-software. *virtual-serial-port.* [En ligne] Eltima, 21 juillet 2020. https://www.virtual-serial-port.org/fr/articles/modbus-software/.

[20]. **THIERRY, VAIRA.** Mise en oeuvre d'un port série. *tvaira.free.* [En ligne] 24 mai 2018. http://tvaira.free.fr/projets/activites/activite-port-serie.html.

[21]. **COPADATA.** quest-ce-qu-une-ihm-interface-homme-machine-copa-data. *copadata.* [En ligne] 2020. https://www.copadata.com/fr/contact/.

[22]. **KEPFRANCE SAS.** Environnement de programmation IHM. *kepfrance.* [En ligne] 2019. [Citation : https://www.kepfrance.fr/pupitres-tactiles-multiprotocoles-easybuilder-pro--a1104.html juillet 2020.]

[23]. **WEINTEK.** *EasyBuilder Pro User Manual v6.04.01*. 2013.

[24]. **LAMINE, M. TOURE MOHAMED.** Cours de Proteus Professionel. Guinée : s.n.

[25].**ELEKTRONIQUE.** Proteus (ISIS et ARES). *elektronique.* [En ligne] http://www.elektronique.fr/logiciels/proteus.php.

[26]. **UNIV USTO.** Introduction_Isis_Proteus. *univ-usto.* [En ligne] 2016. https://www.univ-usto.dz/images/coursenligne/Introduction_Isis_Proteus.pdf.

[27]. **W, DITCH.** PlcLib(Arduino) User Guide. *electronics-micros.* [En ligne] 3 janvier 2017. http://electronics-micros.com/resources/arduino/plclib-docs/plcLib-arduino-guide-v1pt2.pdf.

[28]. **RX, PHILLIPE.** Tutoriel bibliothèque Arduino. *robot-maker.* [En ligne] 14 avril 2016. https://www.robot-maker.com/forum/tutorials/article/30-tutoriel-bibliotheque-arduino/.

[29]. **ATOUI, HAMZA.** *test des fonctions Timer et Counter par un GRAFCET et la conversion de GRAFCET en C++ OOP.* [programme] 2020.

[30].**ARDUINO.**Language Reference. *arduino.* [En ligne] 2020. https://www.arduino.cc/reference/en/.

[31]. **RYAN, BECARELLI.** Arduino Modbus Simulator. [En ligne] Octobre 2016. [Citation : 27 juillet 2020.] https://eprints.usq.edu.au/31372/1/Beccarelli_R_Hills.pdf.

[32]. **SAXENA, ABHINAV.** MobusRTUSlaveExample.ino. *gist.github.* [En ligne] avril 2020. https://gist.github.com/xandfury/626e7849422e8757e8fa1c16403e22ee.

[33].**WIKIPEDIA.**wikipédia.[Enligne] https://fr.wikipedia.org/wiki/Robotique_industrielle#:~:text=Un%20robot%20industriel%20est%20un,un%20espace%20de%20travail%20donn%C3%A9..

[34]. **BAPTISTE, GAULTIER.** D-clics Numérique découvrir décrypter diffuser. *PARCOURS robotique version 1.0.* [En ligne] janvier 2017. http://www.centredeloisirseducatif.net/sites/default/files/dclics-parcours-robotique.pdf.

# ANNEXES :

# Annexe1 :

```
#include<modbus.h>
#include <modbusDevice.h>
#include <modbusRegBank.h>
#include <modbusSlave.h>
#include <Servo.h>

//declaration des servomoteurs:
Servo servoBase ;
Servo servoBras ;
Servo servoSupport;

modbusDevice regBank;
modbusSlave slave;

//variable d' affichage:
int N1;// numero d'etape du grafcet maitre
int N2;// numero d'etape du grafcet chargement
int N3;// numero d'etape du grafcet dechargement

//états des étapes
bool X0,X1,X2,X3,X4,X5,X6,X7,X8,X9,X10,X11;// grafcet maitre
bool    X20,X21,X22,X23;//grafcet      chargement

bool X30,X31,X32,X33;//grafcet déchargement

//valeurs précédentes des états des étapes
bool XP0,XP1,XP2,XP3,XP4,XP5,XP6,XP7,XP8,XP9,XP10,XP11;
bool XP20,XP21,XP22,XP23;
bool XP30,XP31,XP32,XP33;

// variables d'entrées
bool A, PPA, B, PPB, FSV, FEV;

// vairables de sorties et actions
Bool    SV,EV,AGV,AC;


bool gotoA, gotoB, gotoC1, gotoC2_fromA, gotoC2_fromB;
int moteur_base, moteur_bras;//commande des moteur en PWM(0 Ã  255)(rapport
cyclique de 0 Ã  1)

// variables des réceptivités
bool  r0,r1,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11_3,r11_6,r11_9,r11_10;//  grafcet
maitre
bool r20,r21,r22,r23;//grafcet chargement
bool r30,r31,r32,r33;//grafcet déchargement


// variables activation des étapes
bool S0,S1,S2,S3,S4,S5,S6,S7,S8,S9,S10,S11;// grafcet maitre
bool S20,S21,S22,S23;//grafcet chargement
bool S30,S31,S32,S33;//grafcet déchargement

// variables désactivation des étapes
bool R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11;// grafcet maitre
bool R20,R21,R22,R23;//grafcet chargement
bool R30,R31,R32,R33;//grafcet déchargement
```

```
// variable initialisation des grafcets
bool INIT;
// pins des entrees
int A_pin   = 7;//capteur de fin de traitement au poste A
int PPA_pin = 6;//capteur de presence de piece au poste A
int   B_pin     =   5;//capteur   de   fin   de   traitement   au   poste   B

int PPB_pin = 4;//capteur de fin de presence de piece au poste B
int FSV_pin = 3;//capteur de fin de sortie du verin
int FEV_pin = 2;//capteur de fin de entree du verin

// pins des sorties
int SV_pin  = 22;//alimentation de la bobine de distributeur pour sortir le
verin
int EV_pin  = 23;//alimentation de la bobine de distributeur pour entrer le
verin
int AGV_pin = 24;//alimentation des generateur du vide afin de tirer le
verre
int AC_pin  = 25;//alimentation du clapet pour lâcher le verre

int moteur_base_pin = 10;
int moteur_bras_pin = 11;
int moteur_support_pin=12;

// variables Temporitsations
bool T1 = false, T2= false, T3= false, T4= false, T5= false, T6= false, T7
= false, T8= false, T9= false;
unsigned long T1_value = 15000, T2_value = 20000, T3_value = 15000,
T4_value = 20000, T5_value = 15000;
unsigned long T6_value = 10000, T7_value = 10000, T8_value = 10000,
T9_value = 10000;
unsigned long T1_begin, T2_begin, T3_begin, T4_begin, T5_begin, T6_begin,
T7_begin, T8_begin, T9_begin;
unsigned long t1, t2, t3, t4, t5, t6, t7, t8, t9;//valeur de comptage des
temporisations

void setup() {
  INIT = true;
//les pins réservées au servomoteurs:
  servoBase.attach(moteur_base_pin);
  servoBras.attach(moteur_bras_pin );
  servoSupport.attach(moteur_support_pin);

//les positions initiales des servomoteurs:
servoBase.writeMicroseconds(1500);//0°
servoBras.writeMicroseconds(1500);//0°
servoSupport.writeMicroseconds(1000);// 0° position horizontal du verin


//-------------
  pinMode(A_pin,   INPUT);
  pinMode(PPA_pin, INPUT);
  pinMode(B_pin,   INPUT);
  pinMode(PPB_pin, INPUT);
  pinMode(FSV_pin, INPUT);
  pinMode(FEV_pin, INPUT);

  pinMode(SV_pin,  OUTPUT);
  pinMode(EV_pin,  OUTPUT);
  pinMode(AGV_pin, OUTPUT);
```

```
  pinMode(AC_pin,  OUTPUT);

  pinMode(moteur_base_pin,OUTPUT);

  pinMode(moteur_bras_pin, OUTPUT);

  //librarie modbus identifiant(adresse d'esclave):
  regBank.setId(1);
  //Affecter l'objet d'appareil modbus au gestionnaire protocole :
  slave._device = &regBank;

  regBank.add(30001); // numéro étape active dans le grafcet maitre
  regBank.add(30002); // numéro étape active dans le grafcet chargement
  regBank.add(30003); // numéro étape active dans le grafcet déchargement

  regBank.add(30011);//temporisation t1
  regBank.add(30012);//temporisation t2
  regBank.add(30013);//temporisation t3
  regBank.add(30014);//temporisation t4
  regBank.add(30015);//temporisation t5
  regBank.add(30016);//temporisation t6
  regBank.add(30017);//temporisation t7
  regBank.add(30018);//temporisation t8
  regBank.add(30019);//temporisation t9

  regBank.add(10001);//capteur A
  regBank.add(10002);//capteur PPA
  regBank.add(10003);//capteur B
  regBank.add(10004);//capteur PPB
  regBank.add(10005);//capteur FSV
  regBank.add(10006);//capteur FEV

  regBank.add(1);//voyant SV
  regBank.add(2);//voyant EV
  regBank.add(3);//voyant AGV
  regBank.add(4);//voyant AC
  regBank.add(5);//voyant gotoA
  regBank.add(6);//voyant gotoB
  regBank.add(7);//voyant gotoC1
  regBank.add(8);//voyant gotoC2_fromA
  regBank.add(9);//voyant gotoC2_fromB

  //vitesse de communication:
  slave.setBaud(19200);


}
//////////////////////////////////////////////////////////////////

void loop() {

  // lecture des entrées
  A   = digitalRead(A_pin);
  PPA = digitalRead(PPA_pin);
  B   = digitalRead(B_pin) ;
  PPB = digitalRead(PPB_pin);
  FSV = digitalRead(FSV_pin);
  FEV = digitalRead(FEV_pin);

  // calcul des réceptivités
  r0  = XP33 && FEV;
```

```
  r1  = A   && PPA;
  r2  = T1;
  r3  = XP23 && FEV;
  r4  = !A && PPA && B && PPB;
  r5  = T3;
  r6  = XP23 && FEV;
  r7    =   (A   &&   !PPA   &&   !B)||(!A   &&   PPA   &&   B   &&   !PPB);

  r8  = T5;
  r9  = XP23 && FEV && !PPA;
  r10 = XP23 && FEV && !PPB;

  r11_3  = T2;
  r11_6  = T4;
  r11_9  = T6;
  r11_10 = T7;

  r20 = FEV;
  //r21 = (X2&& !XP2) || (X5&& !XP5) || (X8&& !XP8);
  r21 = X2 || X5 || X8;
  r22 = FSV;
  r23 = T8;

  r30 = FEV;
  //r31 = X11&& !XP11;
  r31 = X11;
  r32 = FSV;
  r33 = T9;


  // calcul activation des étapes
  S0  = INIT || XP11 && r0;
  S1  = XP0 && r1 && !INIT;
  S2  = XP1 && r2 && !INIT;
  S3  = XP2 && r3 && !INIT;
  S4  = XP0 && r4 && !INIT;
  S5  = XP4 && r5 && !INIT;
  S6  = XP5 && r6 && !INIT;
  S7  = XP0 && r7 && !INIT;
  S8  = XP7 && r8 && !INIT;
  S9  = XP8 && r9 && !INIT;
  S10 = XP8 && r10 && !INIT;
S11 = (XP3 && r11_3||XP6 && r11_6||XP9 && r11_9 || XP10 && r11_10)&& !INIT;

  S20 = INIT || XP23 && r20 ;
  S21 = XP20 && r21 && !INIT && ! XP3 && !XP6 && !XP9 && !XP10;
  S22 = XP21 && r22 && !INIT;
  S23 = XP22 && r23 && !INIT;

  S30 = INIT || XP33 && r30;
  S31 = XP30 && r31 && !INIT && !XP0;
  S32 = XP31 && r32 && !INIT;
  S33 = XP32 && r33 && !INIT;



  // calcul désactivation des étapes
  R0  = (XP1 || XP4 || XP7) && !INIT;
  R1  = XP2  || INIT;
  R2  = XP3  || INIT;
  R3  = XP11 || INIT;
```

```
R4  = XP5  || INIT;
R5  = XP6  || INIT;
R6  = XP11 || INIT;
R7  = XP8  || INIT;
R8  = XP9  || XP10 || INIT;
R9  = XP11 || INIT;
R10 = XP11 || INIT;
R11 = XP0  || INIT;

R20 = XP21 && !INIT;
R21 = XP22 || INIT;
R22 = XP23 || INIT;
R23 = XP20 || INIT || XP3 || XP6 || XP9 || XP10

R30 = XP31 && !INIT;
R31 = XP32 || INIT;
R32 = XP33 || INIT;
R33 = XP30 || INIT || XP0  ;



// calcul des états des étapes
X0  = (XP0  || S0)  && !R0;
X1  = (XP1  || S1)  && !R1;
X2  = (XP2  || S2)  && !R2;
X3  = (XP3  || S3)  && !R3;
X4  = (XP4  || S4)  && !R4;
X5  = (XP5  || S5)  && !R5;
X6  = (XP6  || S6)  && !R6;
X7  = (XP7  || S7)  && !R7;
X8  = (XP8  || S8)  && !R8;
X9  = (XP9  || S9)  && !R9;
X10 = (XP10 || S10) && !R10;
X11 = (XP11 || S11) && !R11;

X20 = (XP20 || S20)  && !R20;
X21 = (XP21 || S21) && !R21;
X22 = (XP22 || S22) && !R22;
X23 = (XP23 || S23) && !R23;

X30 = (XP30 || S30) && !R30;
X31 = (XP31 || S31) && !R31;
X32 = (XP32 || S32) && !R32;
X33 = (XP33 || S33) && !R33;



//visualisation des états
if (X0) N1=0;
if (X1) N1=1;
if (X2) N1=2;
if (X3) N1=3;
if (X4) N1=4;
if (X5) N1=5;
if (X6) N1=6;
if (X7) N1=7;
if (X8) N1=8;
if (X9) N1=9;
if (X10) N1=10;
if (X11) N1=11;
```

```
  if (X20) N2=20;
  if (X21) N2=21;
  if (X22) N2=22;
  if (X23) N2=23;

  if (X30) N3=30;
  if (X31) N3=31;
  if (X32) N3=32;
  if (X33) N3=33;


  //           calcul           des           sorties           (actions):

  SV  = X21 || X31;
  EV  = X23 || X33;
  AGV = X22;
  AC  = X32;

  gotoA        = X1 || X9;
  gotoB        = X4 || X10;
  gotoC1       = X7;
  gotoC2_fromA = X3;
  gotoC2_fromB=X6;


  //affectation des positions des servomoteurs:
  if( gotoA)          GOTOSTATIONA() ;
  if(gotoB)         GOTOSTATIONB();
  if(gotoC1)        GOTOC1() ;
  if(gotoC2_fromA)  GOTOC2FROMA() ;
if(  gotoC2_fromB) GOTOC2FROMB();
if(X21 || X31) SUPPORTVERTICAL();
if (X23 || X33)SUPPORTHORIZONTAL();
  //calcul des transitions des temporisations:
  if (X1 && !XP1) T1_begin = millis() ;
  if (X3 && !XP3) T2_begin = millis();
  if (X4 && !XP4) T3_begin = millis();
  if (X6 && !XP6) T4_begin = millis();
  if (X7 && !XP7) T5_begin = millis();
  if (X9 && !XP9) T6_begin = millis();
  if (X10 && !XP10) T7_begin = millis();
  if (X22 && !XP22) T8_begin = millis();
  if (X32 && !XP32) T9_begin = millis();

  if (X1) t1 = millis()-T1_begin;
  if (X3) t2 = millis()-T2_begin;
  if (X4) t3 = millis()-T3_begin;
  if (X6) t4 = millis()-T4_begin;
  if (X7) t5 = millis()-T5_begin;
  if (X9) t6 = millis()-T6_begin;
  if (X10) t7 = millis()-T7_begin;
  if(X22)t8=millis()-T8_begin;

  if (X32) t9 = millis()-T9_begin;

  T1 = X1 && (t1 > T1_value);
  T2 = X3 && (t2 > T2_value);
  T3 = X4 && (t3 > T3_value);
  T4 = X6 && (t4 > T4_value);
  T5 = X7 && (t5 > T5_value);
  T6 = X9 && (t6 > T6_value);
```

```
  T7 = X10 && (t7 > T7_value);
  T8 = X22 && (t8 > T8_value);
  T9 = X32 && (t9 > T9_value);

//  T1 = X1 && ((millis()-T1_begin) > T1_value);
//  T2 = X3 && ((millis()-T2_begin) > T2_value);
//  T3 = X4 && ((millis()-T3_begin) > T3_value);
//  T4 = X6 && ((millis()-T4_begin) > T4_value);
//  T5 = X7 && ((millis()-T5_begin) > T5_value);
//  T6 = X9 && ((millis()-T6_begin) > T6_value);
//  T7 = X10 && ((millis()-T7_begin) > T7_value);
//  T8 = X22 && ((millis()-T8_begin) > T8_value);
//  T9 = X32 && ((millis()-T9_begin) > T9_value);


  // Ã©criture des sorties
  digitalWrite(SV_pin,SV);
  digitalWrite(EV_pin,EV);
  digitalWrite(AGV_pin,AGV);
  digitalWrite(AC_pin,AC);

  //        mémorisation        des        états        des        étapes

  XP0  = X0;
  XP1  = X1;
  XP2  = X2;
  XP3  = X3;
  XP4  = X4;
  XP5  = X5;
  XP6  = X6;
  XP7  = X7;
  XP8  = X8;
  XP9  = X9;
  XP10 = X10;
  XP11 = X11;

  XP20 = X20;
  XP21 = X21;
  XP22 = X22;
  XP23 = X23;

  XP30 = X30;
  XP31 = X31;
  XP32 = X32;
  XP33 = X33;


 // actualisation de INIT
  INIT = false;


 ////////////////////////////////////////////////////////////////
  regBank.set(30001,N1); // numéro d'étape active dans le grafcet maitre
  regBank.set (30002,N2); // numéro étape active dans le grafcet chargement
  regBank.set(30003,N3); // numéro    étape    active    dans    le    grafcet
déchargement

regBank.set(30011, (int) t1);//affichage de la valeur de la temporisation
t1
  regBank.set(30012, (int) t2);//affichage de la valeur de la temporisation
t2
```

```
  regBank.set(30013, (int) t3);//affichage de la valeur de la temporisation
t3
  regBank.set(30014, (int) t4);//affichage de la valeur de la temporisation
t4
  regBank.set(30015, (int) t5);//affichage de la valeur de la temporisation
t5
  regBank.set(30016, (int) t6);//affichage de la valeur de la temporisation
t6
  regBank.set(30017, (int) t7);//affichage de la valeur de la temporisation
t7
  regBank.set(30018, (int) t8);//affichage de la valeur de la temporisation
t8
  regBank.set(30019, (int) t9);//affichage de la valeur de la temporisation
t9

  regBank.set(10001,A);//affichage de l'etat du capteur A
  regBank.set(10002,PPA);//affichage de l'etat du capteur PPA
  regBank.set(10003,B);//affichage de l'etat du capteur B
  regBank.set(10004,PPB);//affichage de l'etat du capteur PPB
  regBank.set(10005,FSV);//affichage de l'etat du capteur FSV
  regBank.set(10006,FEV);//affichage de l'etat du capteur FEV



  regBank.set(1,SV);//adresse du voyant SV
  regBank.set(2,EV);//adresse du voyant EV
  regBank.set(3,AGV);//adresse du voyant AGV
  regBank.set(4,AC);//adresse du voyant AC
  regBank.set(5,gotoA);//adresse du voyant gotoA
  regBank.set(6,gotoB);//adresse du voyant gotoB
  regBank.set(7,gotoC1);//adresse du voyant gotoC1
  regBank.set(8,gotoC2_fromA);//adresse du voyant gotoC2_fromA
  regBank.set(9,gotoC2_fromB);//adresse du voyant gotoC2_fromB

 //lancer          la          communication          avc          l'ihm:

 slave.run();
}
////////////////////////////////////////////////////////////////////
//fonctions rotation des servomoteurs:
////////////////////////////////////////////////////////////////////
void GOTOSTATIONA() {

  servoBase.writeMicroseconds(1500);//0Â°
  delay(20);
  servoBras.writeMicroseconds(2000);//+90Â°
  delay(20);
}
////////////////////////////////////////////////////////////////////
void GOTOSTATIONB() {
  servoBase.writeMicroseconds(1500);//0Â°
  delay(20);
  servoBras.writeMicroseconds(1000);//-90Â°
  delay(20);
}
////////////////////////////////////////////////////////////////////
void GOTOC1() {
  servoBase.writeMicroseconds(1500);//0Â°
  delay(20);
  servoBras.writeMicroseconds(1500);//0Â°
  delay(20);
```

```
}
////////////////////////////////////////////////////////////////////////
void GOTOC2FROMA() {
  servoBase.writeMicroseconds(2000);//+90Â°
  delay(20);
  servoBras.writeMicroseconds(2000);//+90Â°
  delay(20);}

////////////////////////////////////////////////////////////////////////
void GOTOC2FROMB() {
  servoBase.writeMicroseconds(1000);//-90Â°
  delay(20);
  servoBras.writeMicroseconds(1000);//-90Â°
  delay(20);
}
////////////////////////////////////////////////////////////////////////
void SUPPORTHORIZONTAL()
{
  servoSupport.writeMicroseconds(1000);//0Â°
  delay(20);
}
////////////////////////////////////////////////////////////////////////
void SUPPORTVERTICAL(){
  servoSupport.writeMicroseconds(2000);//+90Â°
}
```

# Annexe2:

## Définition :

**Robots manipulateur** est en forme d'un bras et se composent d'un certain nombre de segments qui est conçu pour manipuler ou déplacer des matériaux, outils et pièces sans contact humain direct. Ils sont des dispositifs qui permettent aux humains d'interagir avec des objets dans un environnement en toute sécurité. Les robots manipulateurs sont utilisés dans des applications industrielles, installé sur le lieu de travail afin de d'exécuter des travaux programmés.

**Un robot industriel** est un « manipulateur multi-application reprogrammable commandé automatiquement, programmable sur trois axes ou plus, qui peut être fixé sur place ou mobile, conçu à être utilisé dans des applications d'automatisation industrielle ».

**La base:**

La base du manipulateur est fixée sur le lieu du travail.

**Segment :** principalement c'est le corps du robot comprenant les jonctions, les articulations, et d'autres éléments de structure du robot.

**Axe (Articulation):**

Une articulation lie deux corps successifs en limitant le nombre de degré de liberté de l'un par rapport à l'autre.

- Articulation rotoïde (rotation)
- Articulation prismatique (translation)

**Capteurs :**

L'utilisation de capteurs permet en étant combinée à des algorithmes pour traiter les données et à une structure mécanique adaptée, d'accroitre l'autonomie du robot pour réaliser des tâches. Cela peut aller aussi bien dans le sens d'un travail complètement automatisé, et donc sans humain, que dans le sens d'une collaboration plus fine ou plus naturelle avec les hommes.

Les capteurs pour la robotique sont variés, ils permettent au robot de connaître son état interne (par exemple les encodeurs permettent de connaître les positions articulaires)

**Actionneur:**

Sont les «muscles» de manipulateurs. Le contrôleur envoie des signaux aux actionneurs, qui, à son tour, déplacent les articulations du robot et des jonctions, les types communs des actionneurs sont les servomoteurs, les moteurs pas à pas, les actionneurs pneumatiques et les vérins hydrauliques. Les actionneurs sont sous le contrôle du contrôleur.

**Effecteur:**

Ce sont les outils placés au bout des robots industriels et qui permettent la réalisation de la tâche (soudage, préhension, peinture, inspection, etc.). Cette partie est reliée à la dernière jonction (main) d'un manipulateur.

**Les circuits électroniques**

Les microprocesseurs ou les microcontrôleurs sont des éléments primordiaux d'un robot, car ils permettent l'exécution de logiciels informatiques donnant son autonomie au robot.

**Le pupitre de programmation**

Appelé aussi « teach pendant » ou boitier opérateur, permet d'effectuer la programmation par apprentissage. Il comporte généralement un écran d'affichage, des boutons de commande et un dispositif de mise en mouvement de robot (arrêt, départ, mouvement manuels, clavier, etc.)

## Les servomoteurs:

• Les servomoteurs sont des moteurs capables de maintenir une position ( à un effort statique et dont la position est vérifié en continue (grâce au potentiomètre) et corrigé en fonction de la mesure .C'est un système motorisé capable d'atteindre des positions prédéterminées puis de les maintenir (peuvent tourner avec une liberté d'environ 180°). Le servomoteur intègre un système électronique qui convertie un signal numérique en un angle qui sera reproduit grâce au moteur électrique à courant continu présent dans le servomoteur.

• Ce signal numérique est une dérivée de la technique PWM ou MLI (Modulation en Largeurs d'Impulsions). Le servo est alimenté avec 3 fils: une entrée 5V (ou plus), une masse et une

entrée d'impulsion (la commande du servo). C'est dans cette entrée d'impulsion qu'est envoyé le signal numérique modulé en impulsions.

• Ces impulsions sont des créneaux à rapport cyclique variable (5 à 10% pour le bon fonctionnement du servomoteur), et la période de ce créneau est fixée à 20 ms:

**Composition du servomoteur :**

- un micromoteur à courant continu ;
- un axe de rotation ;
- une boite de réduction de la vitesse ;
- un potentiomètre (un capteur de position de l'angle d'orientation de l'axe) ;
- une électronique de contrôle (position de l'axe+ pilotage du moteur CC) ;

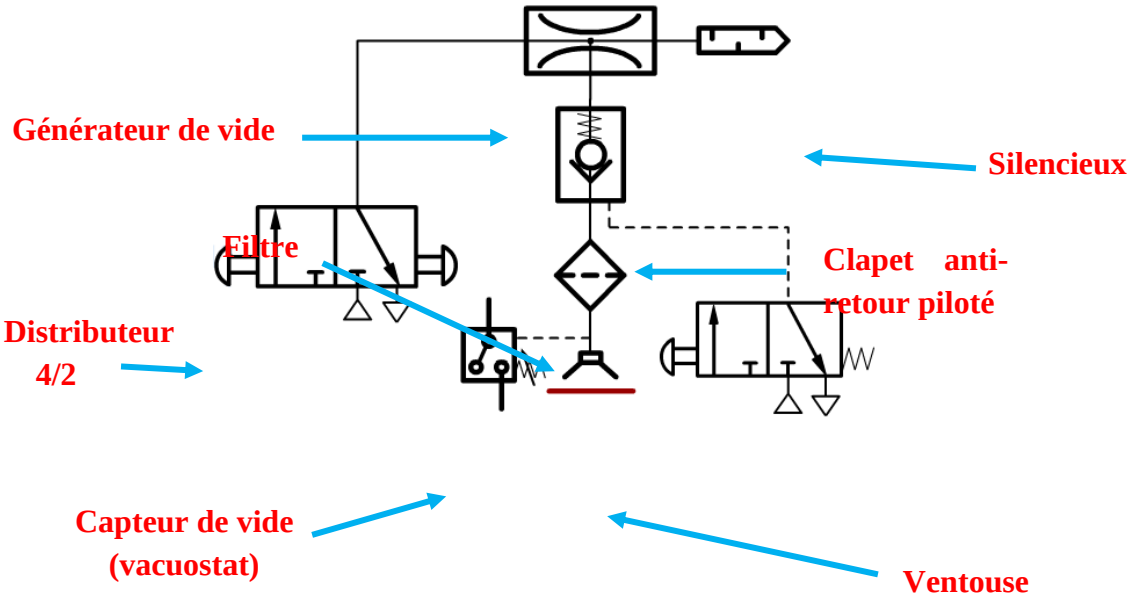
**Technique de préhension par le vide :**

Une chaine d'action pneumatique est généralement constituée : D'un distributeur et d'un vérin, Il existe aussi la chaîne de *préhension par le vide.*

Le physicien italien Giovanni Battista Venturi avait déjà découvert et analysé au XVIIIème siècle l'interaction de l'étranglement et du débit.

De nos jours, cette technique de manipulation d'objet est très fréquemment utilisée dans l'industrie en se basant sur l'effet Venturi. Assez simple à mettre en œuvre, cette technique est souvent plus économique que d'utiliser des pinces adaptées aux pièces à manipuler.

Aujourd'hui la technologie du vide est un élément essentiel dans la production automatisée (pièces, outils, plaques en verres..) quelque soit leur matières et leurs formes, elles peuvent être manipulées à l'aide du vide avec toute sûreté et sans risque d'une manière continue.




Technique de préhension par le vide

**Constitution d'une chaîne de préhension par le vide :**

Cette chaîne se constitue de :

- Deux distributeurs 4/2 (l'un pour l'alimentation du générateur du vide, l'autre pour le pilotage du clapet anti-retour) ;
- Un générateur de vide (aspirer l'air présent sur une canalisation et créer une dépression) ;
- Un silencieux (pour minimiser le bruit) ;
- Un clapet anti-retour (bloquer totalement le débit d'air dans un sens) ;
- Un filtre (filtré l'air en bloquant des particules) ;
- Un vacuostat (capteur de vide détectant la dépression sur l'installation) ;
- Une ventouse (permettent de déplacer des pièces avec grande douceur sans les détériorés) ;

**Principe de fonctionnement de cette chaîne :**

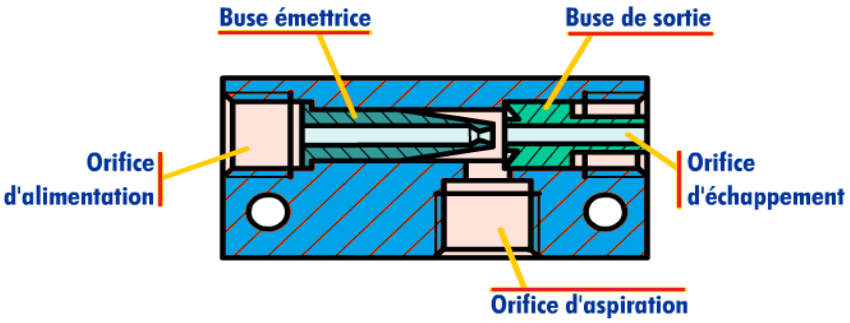
- Initialement le système est au repos. La bille du clapet est plaquée sur le siège par le ressort, l'objet à manipuler est donc pas aspirer ;
- Lorsque il y'aura actionnement sur le distributeur, il y'aura donc un changement de chambre du distributeur ce qui fait l'alimentation du générateur de vide, l'air sera aspiré à travers le clapet (la bille sera décollée de son siège par le courant d'aspiration) , le filtre et la ventouse. L'objet sera plaqué sur la ventouse ;
- Une fois le distributeur sera dis-actionner la chambre sera déplacer également et le générateur de vide ne sera plus alimenté. La bille est plaquée sur son siège par le ressort et la dépression présente dans la ventouse : la pièce est maintenue aspirée par la ventouse ;
- Lorsque il y'aura actionnement sur l'autre distributeur la bille sera décollée de son siège temporairement et elle sera plaquer à nouveau sur le siège après le dis-actionnement du distributeur par le ressort : il permettra le relâchement de la pièce par la ventouse et le retour à l'état initiale ;

**L'effet venturi :**

C'est un effet d'aspiration provoqué par le passage d'un courant d'air, mis en application dans les générateurs de vide.

Le générateur de vide est alimenté en air comprimé, l'air est injecté au travers d'une buse de petite dimension, l'air est accéléré grâce à cette buse émettrice.

Dans le volume qui entour la buse, les molécules d'air sont aspirées et fusent par la buse de sortie avec l'alimentation d'air qui s'échappe.




effet venturi

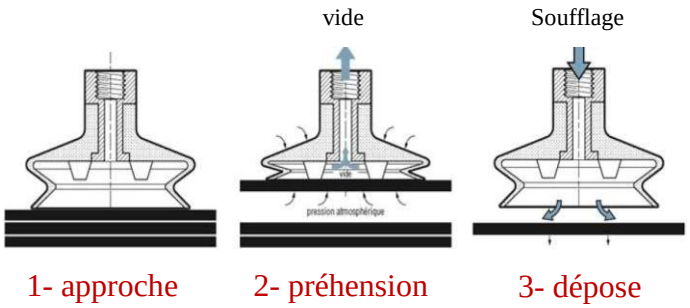
**Les ventouses :**

Les ventouses permettent de manipuler des pièces avec une grande douceur et sans détériorer les matériaux.

La surface de contact entre la ventouser et la pièce doit être lisse et le matériau non poreux pour atteindre la force de préhension nécessaire. La précision de positionnement de l'objet sur la ventouse n'est pas trop élevée.

La ventouse est reliée à un circuit d'aspiration appelé générateur de vide.

Lors de l'aspiration de l'air à l'intérieur de la ventouse, il se produit une différence de pression par rapport à la pression atmosphérique à l'extérieur. La pression atmosphérique étant alors plus élevée que la pression dans la ventouse, la pièce est plaquée contre la ventouse et peut être déplacée.




Les ventouses

**Famille des capteurs :**

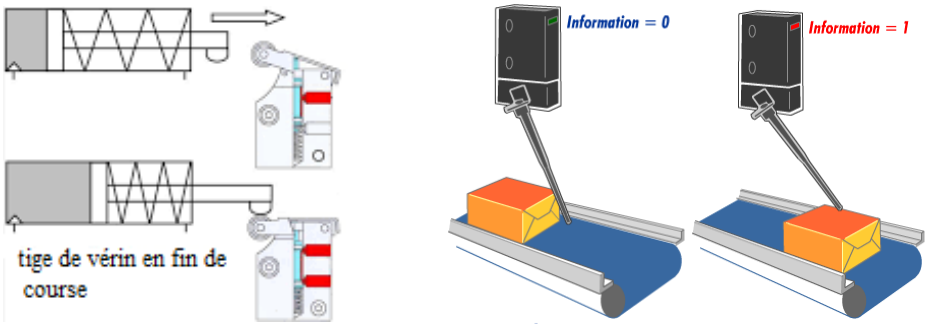
Il existe un très grand nombre de capteurs, adapté en fonction du type d'application, de mesure ou d'actionneur.

Parmi les détecteurs de présence on distingue *les détecteurs par contact* (action mécanique) et *les détecteurs de proximité.*

Concernant les détecteurs de position ou mesure de déplacement on utilise principalement les *codeurs optiques*.

***Les détecteurs par contact (action mécanique)***

Exemple de détection de passage d'un carton dans un tapis roulant, ou détection des fins de cours d'un vérin par capteur à galet ou interrupteur de position.




Les détecteurs par contact (action mécanique)

***Les détecteurs  sans contact (de proximité) :***

On distingue  4  familles de détecteurs de proximité électrique :

- Détecteurs inductifs ;
- Détecteur capacitifs ;
- Détecteurs  magnétiques ;
- Détecteurs Photo-électriques ;

## Détecteurs pour applications spécifiques -Détection de verre et de film transparent :

Les détecteurs de proximité sont des produits essentiels dans tous les automatismes. Ils sont basés sur des lois physiques connues depuis des décennies mais mises en applications, aujourd'hui, grâce à de nouveaux matériaux et surtout à l'électronique. Ces capteurs ou détecteurs peuvent utiliser les ultrasons, le rayonnement infrarouge, les courants de Foucault, l'effet Hall, ils peuvent être fluidiques, magnétiques, inductifs ou capacitifs. A la suite des divers guides d'achats concernant les détecteurs inductifs et capacitifs et les détecteurs de proximité optiques, nous nous bornerons aujourd'hui aux détecteurs d'objets transparents. Il existe trois technologies pour détecter la présence d'objets transparents et d'objets réflexifs elles sont basées sur :

- la variation de capacité d'un condensateur (capacitif) ,
- l'optoélectronique (optique)
- les ultrasons.

### Détecteurs capacitifs :

Ils détectent les matériaux de toute nature (verre, plastique, métaux, liquide, poudre..).
Ils sont employé généralement pour détecter les éléments non conducteurs (non détecté par des capteurs inductifs), ils sont sensible la saleté et la poussière.
La distance de détection reste faible (quelques millimètres) par rapport à celle des capteurs optiques et à ultrasons.
Les détecteurs de proximité capacitifs sont conçus pour fonctionner par création d'un champ électrostatique et par détection des modifications de ce champ créées par une cible approchant(en fonction de la constante diélectrique $\varepsilon r$ ) de la face de détection.

### Détecteurs à Ultrasons :

Un capteur à ultrasons émet à intervalles réguliers de courtes impulsions sonores à haute fréquence. Ces impulsions se propagent dans l'air à la vitesse du son. Lorsqu'elles rencontrent un objet, elles se réfléchissent et reviennent sous forme d'écho au capteur. Celui-ci calcule alors la distance le séparant de la cible sur la base du temps écoulé entre l'émission du signal et la réception de l'écho.

Pratiquement tous les matériaux qui reflètent le son peuvent être détectés et ce, quelle que soit leur couleur. Même les matériaux transparents, feuilles minces sans problèmes à détecter.

**Détecteurs optoélectroniques:**

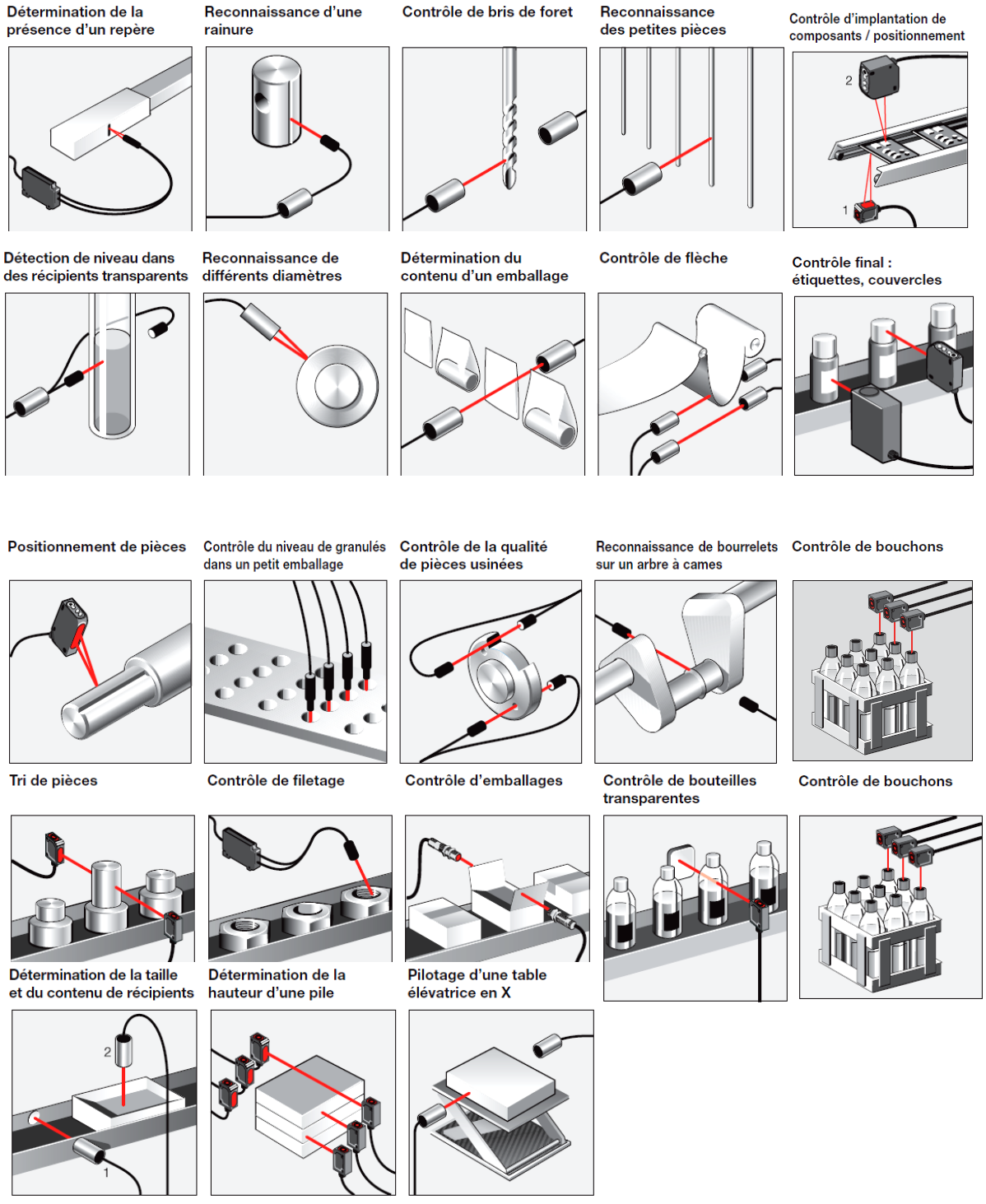
Les détecteurs d'objets transparents sont constitués d'un système réflex à filtre de polarisation et d'un réflecteur « nid d'abeille » très fin. Le verre, les films transparents, les bouteilles en PET et les emballages transparents sont détectés de manière fiable. Cela permet de compter des bouteilles et des verres ou de surveiller la rupture des films. Pour cela, ces détecteurs sont principalement utilisés pour des applications dans l'agroalimentaire, les boissons et la pharmaceutique.

Les détecteurs travaillent en lumière rouge visible, ce qui facilite leur orientation lors de la mise en service, caractérisés par une haute fréquence de commutation.

Parmi ces détecteur optoélectroniques les plus utilisés on cite :

- Cellule en mode barrage ;
- Cellule en mode reflex ;
- Détecteurs de contraste et analyseurs de couleurs ;
- Rideaux optoélectroniques ;

## Quelques applications des capteurs optoélectroniques :




Quelques applications des capteurs optoélectroniques

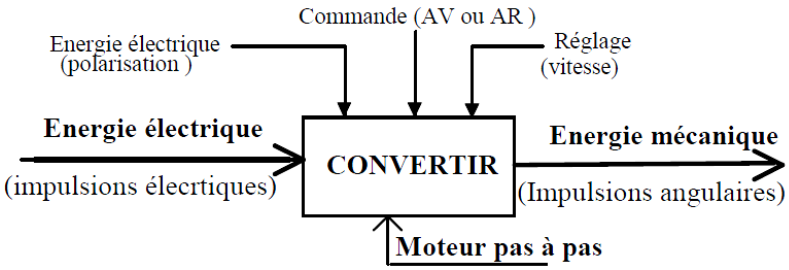
## Les moteurs pas à pas :

**Description :** des moteurs spéciaux destinés pour la commande avec une grande précision du déplacement et le positionnement d'un objet.

Un moteur pas à pas est un actionneur qui transforme une information numérique sous forme de train d'impulsions en un nombre équivalent de pas angulaire de caractère incrémental.

Ils sont utilisés pour les positionnements angulaires précis (imprimantes, scanners, disques durs ...). Contrairement aux moteurs à courant continu, il ne nécessite pas de boucle d'asservissement et sont plus simples à commander, il existe trois types de moteurs pas à pas :

- moteurs à aimant permanent.
- moteurs à réluctance variable.
- moteurs hybrides.

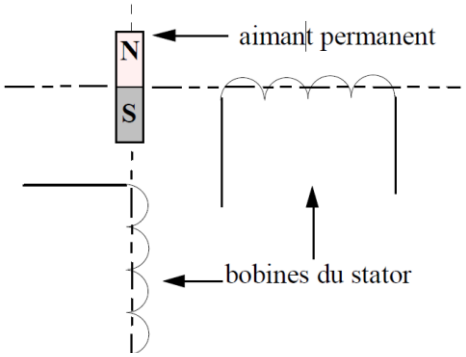



## Principe de fonctionnement d'un moteur pas à pas :

Si on place, sur un axe de rotation, entre deux bobines à axes perpendiculaires un aimant permanent on constate que :

- Si une seule bobine est alimentée l'aimant se positionne parallèlement à son axe.
- Si on inverse le courant dans la bobine, l'aimant fait un 1/2 tour (90°) et reste parallèle à l'axe de la bobine.
- Si les deux bobines sont alimentées, l'aimant se positionne suivant la bissectrice des deux axes.

On dit que l'aimant se positionne de façon qu'il soit traversé par le maximum de flux : règle de flux maximal.

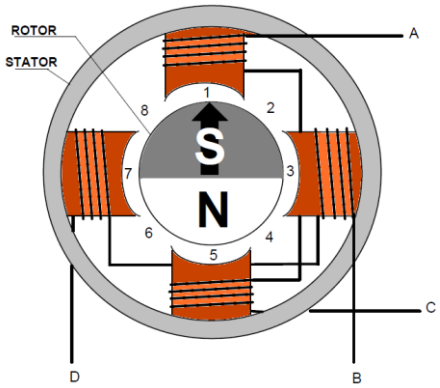



Moteur à aiment permanent

**Constitution d'un moteur pas à pas :**

Comme tous les moteurs, il est constitué par :

- Une parie fixe : C'est le stator, formé d'un circuit magnétique et des bobines (phases) dont le rôle est de créer un flux magnétique à directions multiples.
- Une partie mobile : C'est le rotor, placé dans le flux du stator il se positionne suivant le flux maximum.




Moteur pas à pas bipolaire 2 phases au stator ,2 pôles au rotor