وزارة التعليم العالي والبحث العلمي

BADJI MOKHTAR-ANNABA UNIVERSITY
UNIVERSITE BADJI MOKHTAR-ANNABA

جامعة باجي مختار- عنابة

Faculté des Sciences de l'ingéniorat

Département électronique

MEMOIRE

Présenté en vue de l'obtention du diplôme de MASTER

Option : Automatique et Informatique Industriel

Intitulé :

Communication entre station S7-1200 et plateforme Arduino via Modbus TCP/IP.

Domaine : Scientifique et Technologie.

Filière : Automatique.

Spécialité : Automatique et Informatique Industriel.

Par :

ZEGHIDA Abdenour

Devant le jury :

Président : ARBAOUI	**Grade : Dr**	**UBMA-ANNABA**
Directeur de mémoire : BENOUARET	**Grade : Pr**	**UBMA-ANNABA**
Examinateurs : RAMDANI	**Grade : Pr**	**UBMA-ANNABA**
Examinateurs : BEKAIK	**Grade : Dr**	**UBMA-ANNABA**

REMERCIEMENT

Je tiens tour d'abord à exprimer mes profondes gratitudes et tous mes remerciements à mon promoteur, Pr .BENOUARET MOHAMED, pour son patience, ses conseils et ses précieuses Directives, qui m'ont permis de progresser et de réaliser ce mémoire.

Aux membres de jury nous adressons également nos remercîments pour avoir accepté de Juger notre travail.

Aussi, je tiens à exprimer mes gratitudes à Mon père, pour son aide et ses précieux conseils

Enfin, A toutes les personnes qui ont contribué, de près ou de loin à la réalisation de ce mémoire

DÉDICACE

Je dédie ce travail en premier à mes chers parents, pour leur patience,

Leurs sacrifices et leurs encouragements,

A mes sœurs,

Toute ma famille et toute personne qui m'a aidé et donné le courage

De continuer

RÉSUMÉ :

Dans ce projet on a essayé de donner des notions de réseau, les modèles de communication (OSI, TCP, MODBUS TCP/IP), aussi établir une communication Modbus TCP IP entre plateforme Arduino avec un sheilde et automate seimes S7-1200 (server/ client), aussi fabriquer une machine outil qui contient 3 moteurs pas a pas, avec des drivers type TB6600 pour les commander.

Et pour cela on a utilisé TIA PORTAL V13 (Programmation automate) et enivrement d'Arduino.

ABSTRACT:

In this project we've tried to give smattering about network, (OSI, TCP , MODBUS TCP/IP) communication model, ainsi set up in Modbus TCP/IP communication between Arduino platform a shield and automate seimes S7- 1200 (server/client), also made a machine tool which contain 03 step by step engine with TB6600 drivers for order.

And, for that, we've used TIA PORTAL V13 (Automat programmation) and an arduino.

SOMMAIRE

LISTE DES FIGURES :

INTRODUCTION GENERAL

Introduction générale :

Modbus est un standard de protocole industriel créé par Modicon en 1979.
Il est généralement utilisé avec les automates programmables ou les équipements de types industriels. Il est maintenant devenu une norme "Open Protocol" dans ce domaine, il est le moyen le plus couramment utilisé pour faire communiquer des équipements industriels. IL existe des versions avec des modifications mineures ou adaptées à d'autres environnements.

La plupart des appareils et dispositifs embarqués comme les microcontrôleurs, les automates, les capteurs intelligents etc.…sont équipés d'interface Modbus et sont capables de communiquer en Modbus. Bien sure il existe plusieurs mode communication (mode OSI, mode TCP …)

Le protocole de communication Modbus base sure la notion maitre / esclave ou server/client
En dit un protocole de communication est un ensemble de règles permettant à plusieurs périphériques de dialoguer entre eux. A la manière des humains, les périphérique (ordinateur, automate), doivent parler le même langage afin de se comprendre.
Pour dialoguer entre les plateformes ou les périphériques, les informations sont passé par des différentes transmissions (c'est une voie de communication entre deux machines, la communication peut s'effectuer de différentes manières).
Un automate est un dispositif reproduisant en autonomie une séquence d'actions prédéterminées sans l'intervention humaine, le système fait toujours la même chose, ou s'adapte à des conditions environnementales perçues par ses capteurs.

Dans le domaine industriel, on nomme automate un appareil qui exécute une séquence déterminée d'opérations de manière séquentielle en utilisant des technologies mécaniques, électriques, gêné-électrique. Autrefois, le métier d'automatisme consistait en la conception et fabrication d'automates, et il s'est beaucoup développé avec l'avancement de l'horlogerie.

La programmation est réalisée par des dispositifs variés : cartes perforées, rouleaux à picots, arbres à cames,… etc. La lecture et la transmission des informations sont assurées par des systèmes mécaniques et physiques.

Arduino est une marque qui couvre des cartes électroniques matériellement libres sur lesquelles se trouve un microcontrôleur (d'architecture Atmel AVR comme l'Atmega328p, et d'architecture ARM comme le Cortex-M3 pour l'Arduino Due).

Les schémas de ces cartes électroniques sont publiés en licence libre. Cependant, certaines composantes, comme le microcontrôleur par exemple, ne sont pas sous licence libre.

Le microcontrôleur peut être programmé pour analyser et produire des signaux électriques, de manière à effectuer des tâches très diverses comme le contrôle des appareils domestiques, éclairage, chauffa, le pilotage d'un robot, de l'informatique embarquée, etc.

Dans notre projet, nous allons présenter une communication entre deux plateformes différentes (Automate s7 1200 et une carte Arduino).

Cette communication permette de commander les 3 axes (3 moteurs pas à pas) d'une machine industrielle.

- Le chapitre 1 : constitue des généralités sure le réseau industriel (Modbus TCP/IP) et les différent registres.
- Le chapitre 2 : porte sur des définitions de l'automate s7 1200 et ses caractéristiques, ainsi le logiciel de programmation et la méthode de configuration matérielle, en passant sur des généralités sur l'Arduino et leur environnement de programmation.
- Le chapitre 3 : qui représente notre cas pratique qui porte sur une réalisation d'une commande de 3 axes (machine à outil) via Arduino maitre et automate slave.

CHAPITRE 01

Chapitre 1 : Protocol de communication Modbus TCP/ IP :

Introduction :

Modbus est un standard de protocole industriel créé par Modicon en 1979.

Au début, le Modbus a été initialement conçu pour fonctionner avec les lignes de communication filaires série mais il existe aujourd'hui des extensions à la norme pour les communications sans fil et les réseaux TCP / IP. Le protocole Modbus permet la communication entre plusieurs équipements connectés sur un même réseau, par exemple un système qui mesure la température et l'humidité d'un four peut communiquer ses résultats à un ordinateur de traitement via Modbus.

Section 1 : Généralité sur le réseau :

Un protocole de communication est un ensemble de règles permettant à plusieurs périphériques de dialoguer entre eux. A la manière des humains, les périphérique (ordinateur, automate), doivent parler le même langage afin de se comprendre.

Dans cette section nous allons présenter : notion de base de réseau communication TCP/IP et les couches de protocole.

1. Historique :

- 1979 : Création de MODBUS par MODICON (Modular Digital Controller).
- 1994 : Modicon fusionne avec Schneider (Télémécanique / April / Square D).
- 2003 : Transfert de compétences Schneider à MODBUS-IDA.
- 2004 : Pré-Standard international IEC62030.
- 2004 : MODBUS/TCP leader mondial (840000 nœuds).
- 2005 : MODBUS adopté en tant que standard chinois.

2. Le modèle OSI:

Le modèle OSI (de l'anglais Open System Interconnexion) a été créé par l'organisation internationale de standardisation (International Organization for Standardization ISO) en tant que modèle de référence pour une communication ouverte via divers systèmes techniques.

Le modèle OSI est le résultat d'une tentative de standardisation. Il dessine un cadre conceptuel pour la base de design des standards de communication entre différents ordinateurs. Le modèle ISO divise ce processus complexe de communication en 7 couches (de l'anglais layer). On parle alors de modèle en couches. La communication entre deux systèmes exige que chaque couche respecte une tâche.[1]

2.1. Les couches d'un modèle OSI :[2]

Développé en 1978 par l'ISO (International Organization of Standards) afin que soit défini un standard utilisé dans le développement de systèmes ouverts.

Chaque couche, excepté la 1ere et la dernière, utilise les services de la couche inférieure et propose des services à la couche supérieure.

Il est composé de 7 couches représentées sur la figure :

Figure 1: Composition du modèle OSI :

7	Application *Application*	échanges de données d'application
6	Présentation *Presentation*	mise en forme des données pour la transmission
5	Session *Session*	synchronisation de processus
4	Transport *Transport*	transfert de blocs d'octets entre processus
3	Réseau *Network*	transfert de blocs d'octets entre systèmes
LLC	Contrôle de Lien Logique *Logical Link Control*	transfert fiable d'octets entre systèmes raccordés au même médium
MAC	Contrôle d'accès au médium *Medium Access Control*	transfert fiable d'octets entre systèmes raccordés au même médium, avec contrôle du droit d'émettre
1	Physique *Physical*	transfert de blocs d'octets entre systèmes raccordés au même médium

Sources : Cours génie électrique 2eme année de Madame Marie-Claude Vialatte.

La caractérisation donnée ici est tirée du chapitre 7 d'ISO 7498-1. La description originelle donne en plus, pour chaque couche, les fonctions de manipulation de commandes ou de données significatives parmi celles décrites plus bas [3]:

[1]https://www.ionos.fr/digitalguide/serveur/know-how/le-modele-osi-reference-pour-les-standards/ publié 27/12/16 consulter 10/03/19.
[2]Cours génie électrique 2eme année de Madame Marie-Claude Vialatte.
[3]https://loufida.com/les-7-couches-du-modele-osi/

- **La couche physique :** est chargée de la transmission effective des signaux entre les interlocuteurs. Son service est limité à l'émission et la réception d'un bit ou d'un train de bit continu.
- **La couche liaison de données :** gère les communications entre 2 machines directement connectées entre elles.
- **La couche réseau :** gère les communications de proche en proche, généralement entre machines : routage et adressage des paquets.
- **La couche transport :** gère les communications de bout en bout entre processus (programmes en cours d'exécution). Le rôle principal de la couche transport est de prendre les messages de la couche session, de les découper s'il le faut en unités plus petites et de les passer à la couche réseau, tout en s'assurant que les morceaux arrivent correctement de l'autre côté. Cette couche effectue donc aussi le réassemblage du message à la réception des morceaux.
- **La couche session :** gère la synchronisation des échanges et les « transactions », permet l'ouverture et la fermeture de session. Elle réalise le lien entre les adresses logiques et les adresses physiques des tâches réparties.
- **La couche présentation** c'est elle qui traite l'information de manière à la rendre compatible entre tâches communicantes. Elle va assurer l'indépendance entre l'utilisateur et le transport de l'information. Typiquement, cette couche peut convertir les données, les reformater, les crypter et les compresser.
- **La couche application :** Cette couche est le point de contact entre l'utilisateur et le réseau. C'est donc elle qui va apporter à l'utilisateur les services de base offerts par le réseau, comme par exemple le transfert de fichier, la messagerie...

Les couches basses (1, 2, 3 et 4) sont nécessaires à l'acheminement des informations entre les extrémités concernées et dépendent du support physique. Les couches hautes (5, 6 et 7) sont responsables du traitement de l'information relative à la gestion des échanges entre systèmes informatiques. Par ailleurs, les couches 1 à 3 interviennent entre machines voisines, et non entre les machines d'extrémité qui peuvent être séparées par plusieurs routeurs. Les couches 4 à 7 sont au contraire des couches qui n'interviennent qu'entre hôtes distants.[4]

[4] https://www.frameip.com/osi/

3. Modèle TCP/IP :

TCP/IP a également un modèle de réseau comme le modèle OSI. TCP/IP était sur la voie du développement lorsque la norme OSI a été publiée et il existait une interaction entre les concepteurs des normes OSI et TCP/IP.[5]

3.1 Description du modèle TCP/IP :

TCP/IP a été historiquement créé à la demande du ministère de la Défense des États-Unis c'est pourquoi le modèle garde ce nom.TCP/IP désigne communément une architecture réseau, mais cet acronyme désigne en fait 2 protocoles étroitement liés : un protocole de transport, TCP (Transmission Control Protocol) qu'on utilise « par-dessus » un protocole réseau, IP (Internet Protocol). Ce qu'on entend par « modèle TCPIP », c'est en fait une architecture réseau en 4 couches dans laquelle les protocoles TCP et IP jouent un rôle prédominant, car ils en constituent l'implémentation la plus courante. Par abus de langage, TCP/IP peut donc désigner deux choses : le modèle TCP/IP et la suite de deux protocoles TCP et IP.

3.2. Les couches d'un modèle :

Le modèle TCP/IP peut en effet être décrit comme une architecture réseau à 4 couches [6] :

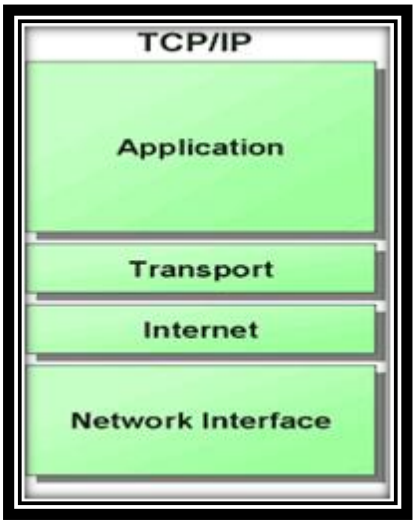
- La couche hôte réseau : Cette couche est assez « étrange ». En effet, elle semble « regrouper » les couches physiques et liaison de données du modèle OSI. En fait, cette couche n'a pas vraiment été spécifiée ; la seule contrainte de cette couche, c'est de permettre un hôte d'envoyer des paquets IP sur le réseau. L'implémentation de cette couche est laissée libre. De manière plus concrète, cette implémentation est typique de la technologie utilisée sur le réseau local. Par exemple, beaucoup de réseaux locaux utilisent Ethernet ; Ethernet est une implémentation de la couche hôte-réseau.

[5] https://loufida.com/les-4-couches-du-modele-tcp-ip/
[6] https://www.frameip.com/tcpip/#2-8211-description-du-modele-tcpip

- La couche internet : Cette couche est la clé de voûte de l'architecture. Cette couche réalise l'interconnexion des réseaux (hétérogènes) distants sans connexion. Son rôle est de permettre l'injection de paquets dans n'importe quel réseau et l'acheminement de ces paquets indépendamment les uns des autres jusqu'à destination.
- La couche transport : Son rôle est le même que celui de la couche transport du modèle OSI.
- La couche application : Contrairement au modèle OSI, c'est la couche immédiatement supérieure à la couche transport, tout simplement parce que les couches présentation et session sont apparues inutiles. On s'est en effet aperçu avec l'usage que les logiciels réseau n'utilisent que très rarement ces 2 couches, et finalement, le modèle OSI dépouillé de ces 2 couches ressemble fortement au modèle TCP/IP.

La figure (2) suivant représente les 4 couches de modelé TCP/IP [7]:

Figure 2: 4 couches de modelé TCP/IP



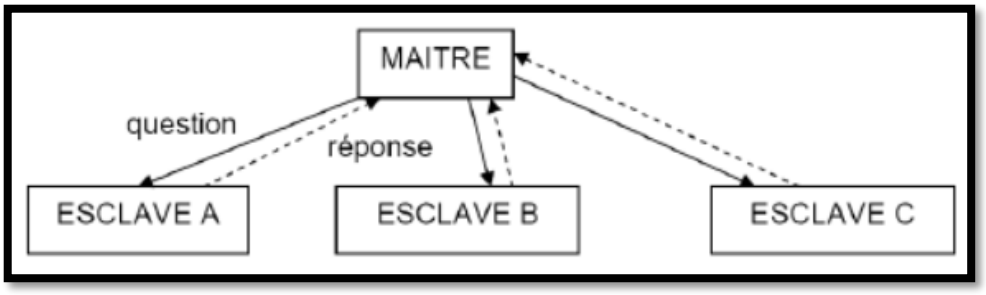
Section 2 : Communication Modbus TCP/IP Fonction et Registre :

Modbus (marque déposée par Modicon) est un protocole de communication utilisé pour des réseaux d'automates programmables (API). Il fonctionne sur le mode maître / esclave. Il est

[7]Network Architectures: **Layers of OSI model and TCP/IP model** , publié le13/04/2011.

constitué de trames contenant l'adresse de l'automate concerné, la fonction à traiter (écriture, lecture), la donnée et le code de vérification d'erreur appelé contrôle de redondance cyclique.

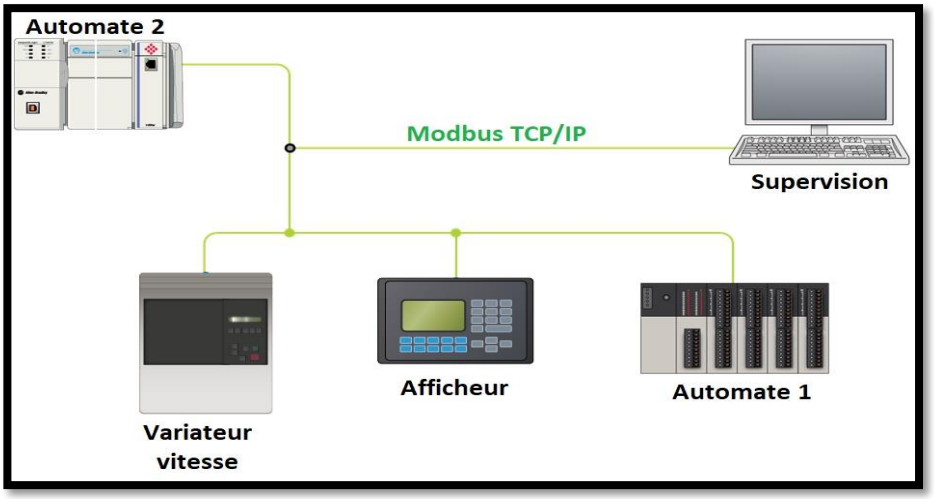
Figure 3 : Construction MODBUS TCP/IP



1. Modbus TCP/IP :

MODBUS TCP / IP est une variante de la famille MODBUS de protocoles de communication simples et indépendants du fournisseur destinés à la supervision et au contrôle des équipements d'automatisation. Plus précisément, il couvre l'utilisation de la messagerie MODBUS dans un environnement« Internet » à l'aide des protocoles TCP / IP. À l'heure actuelle, les protocoles sont principalement utilisés pour la connexion Ethernet d'API.[8]

Figure 4: Système communication MODBUS TCP/IP



[8]**Source:** https://www.automation-sense.com/blog/automatisme/qu-est-ce-que-le-modbus-tcp-ip.html.

2. Transmission des données [9]:

2.1. Les canaux de transmission :

Un canal de transmission ou ligne de transmission est une liaison entre deux machines. On désigne généralement le terme émetteur la machine qui envoie les données et récepteur celle qui les reçoit.

2.2. Caractéristiques d'une transmission :

Pour une transmission de donnée sur une voie de communication entre deux machines, la communication peut s'effectuer de différentes manières. La transmission est caractérisée par :

- Le sens des échanges
- Le mode de transmission : il s'agit du nombre de bit envoyé simultanément.
- La synchronisation: il s'agit de la synchronisation entre émetteur et récepteur.

2.3. Les modes de transmission :

Selon le sens des échanges, on distingue 3 modes de transmission:

- **Mode Simplex ou unidirectionnel :** il caractérise une liaison dans laquelle les données circulent dans un seul sens, c'est-à-dire de l'émetteur vers le récepteur.
- **Mode Half-Duplex ou bidirectionnel alterné:** caractérise une liaison dans laquelle les données circulent dans un sens ou dans l'autre mais pas les deux en même temps. Ce type de liaison permet d'avoir une liaison bidirectionnelle utilisant la capacité totale de la ligne.
- **Mode full duplex ou duplex intégral :** caractérise une liaison dans laquelle les données circulent de façon directionnelle et simultanée. Chaque extrémité de la ligne peut émettre et recevoir en même temps, ce qui signifie que la bande passante est divisée par deux pour chaque sens d'émission des données si un même support de transmission est utilisé pour les deux transmissions.

3. Les liaisons séries :

Les liaisons série dans une liaison de type série, les données sont envoyées bit par bit sur la voie de transmission. Toutefois, étant donné que la plupart des processeurs traitent
Les informations de façon parallèle (transmission simultanée de N bits), les données parallèle arrivant au niveau de l'émetteur et inversement au niveau du récepteur sont transformées en

[9] www.automation-sense.com – Juin 2016/COMPRENDRE ET METTRE EN OEUVRE FACILEMENT LE BUS INDUSTRIEL MODBUS.

série par un contrôleur de transmission appelé UART (Universal Asynchrones Receiver Transmitter).

3.1. Transmission série asynchrone :

En environnement industriel on préfère utiliser la transmission Série asynchrone plus simple mettre en œuvre et moins coûteuse. La ligne peut ne comporter qu'un fil; on en utilise en général 3: émission; réception masse.

Les éléments binaires d'informations (bits) d'un mot ou caractère sont alors envoyés successivement les uns après les autres (sérialisation) au rythme d'un signal d'horloge. Le récepteur effectue l'opération inverse: transformation Série / parallèle à partir de son horloge ayant la même fréquence que celle de l'émetteur. Les informations peuvent être transmises de manière irrégulière, cependant, l'intervalle de temps entre 2 bits est fixe. Des bits de synchronisation (Start, Stop) encadrent les informations de données.

3.2. Transmission série synchrone :

Une transmission synchrone est une transmission dans laquelle, l'émetteur et le récepteur sont cadencés à la même horloge.

4. Les supports physiques de transmission du protocole Modbus :

Les communications Modbus peuvent s'effectuer via les supports physiques suivants :

- RS-232.
- RS-485.
- RS-422.
- Ethernet TCP/IP (Modbus Ethernet).

Les communications de type Modbus sont caractérisées par leur vitesse de transmission qui s'exprime en bits/s .Typiquement, cette vitesse de transmission est souvent comprise entre 9600 et 19 200 bits/s, mais on peut avoir des vitesses supérieures.

4.1.Le Modbus via liaison RS-232/RS-422/RS-485 :

La communication Modbus via(RS-232), (RS-422) et(RS-485) fonctionne en mode maitre/esclave. Cela signifie qu'un dispositif fonctionnant comme maître va interroger un ou plusieurs dispositifs fonctionnant comme esclave.

Un dispositif esclave ne peut donc pas fournir volontairement des informations au maître, il doit attendre une sollicitation.

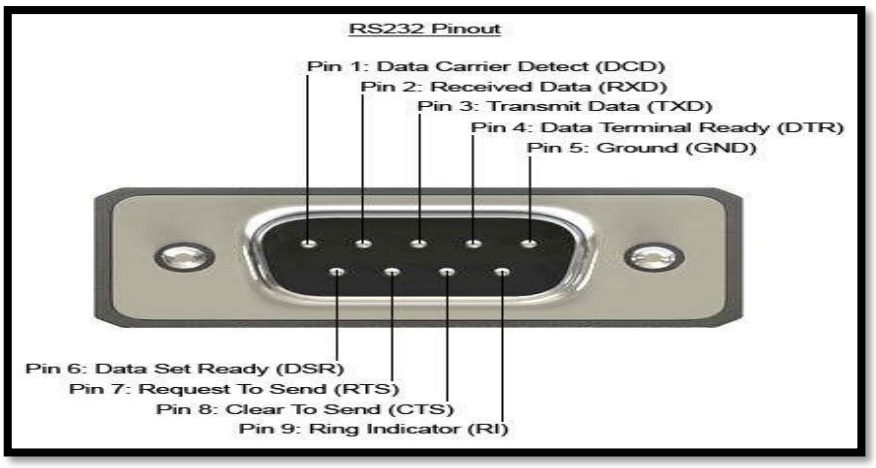
Le maître peut écrire des données dans les registres d'un périphérique esclave ou lire les données à partir des registres de celui-ci.

4.1.1. RS232 :

C'est le plus connu des standards de communication série. Les ports série RS232 sont présents sur la plupart des PCs standards. Il est de type point to point et est composé des lignes Rx, Tx et GND. Il a été couramment nomme (le port série).

La figure suivant représente la prise Rs232 ou DB 9 :

Figure 5:la prise RS232 ou DB 9

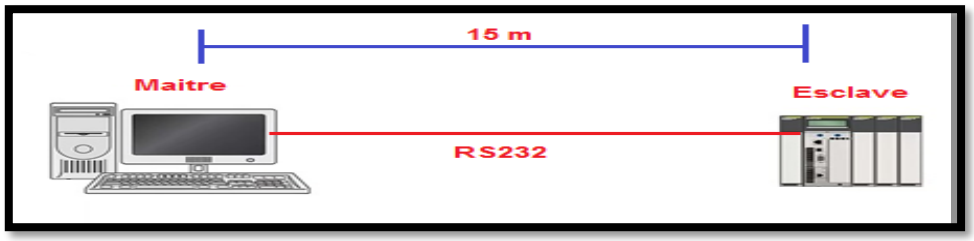


Source : https://www.stratusengineering.com/rs232-9-pin-pinout/

Le RS232 permet de faire communiquer uniquement un maitre et un esclave sur chaque ligne. Il fonctionne en full duplex et sa vitesse de communication peut aller jusqu'à 115 Kbits/s.

En RS232, la distance séparant les deux équipements ne dépasse pas généralement 15 m. Il a comme inconvénients d'être inadapté dans les environnements où il y'a beaucoup de bruits ou parasites (risque perturbation transmission).

Figure 6:exemple de liaison Rs232 entre Pc et Automate

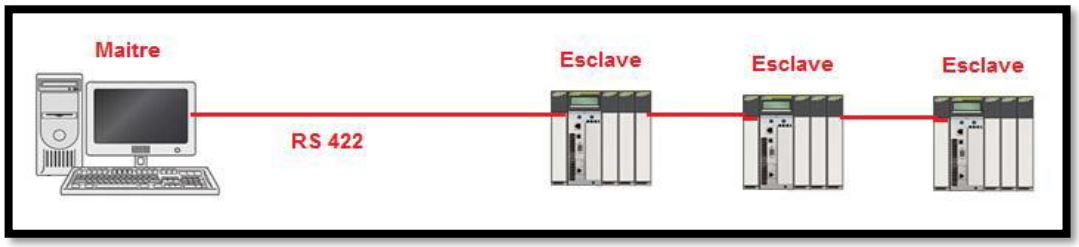


Source : guide-du-Modbus-pour –les nuls-PDF.

4.1.2. Le RS422

Il est full duplex et est utilisé sur les ordinateurs Apple, sa vitesse de transmission peut aller jusqu'à 10 Mbits/s. Les signaux sont envoyés sur 2 fils afin d'augmenter la fréquence de transmission. Il peut supporter jusqu'à 10 récepteurs par ligne (on dit alors qu'il est multi-drop ou multipoints).

Figure 7: le RS422



4.1.3. RS485 :

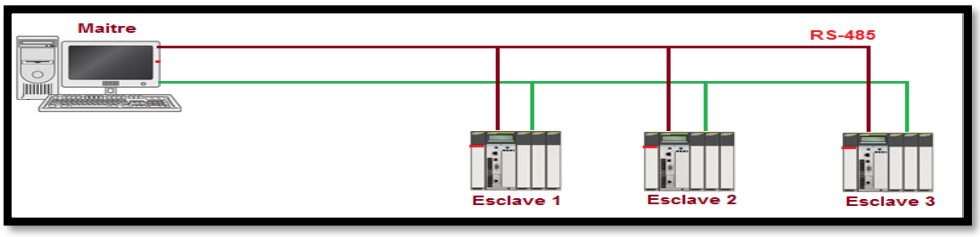
Les médias de type RS485 sont souvent en half duplex c'est-à-dire la transmission s'effectue via 2 fils.

Ils permettent de faire communiquer jusqu'à 32 périphériques sur la même ligne de données et sur une distance pouvant aller jusqu'à 1200 m sans répéteurs.

A noter que l'on peut obtenir du full duplex en utilisant 4 fils de transmission au lieu de 2. Cela permet d'avoir un débit de transmission plus rapide.

Chaque périphérique esclave peut aussi communiquer avec les 32 autres périphériques. Les protocoles de communication RS422 et RS485 sont multi-drop c'est à dire plusieurs périphériques peuvent communiquer sur la même ligne de données. Le RS485 a comme avantages d'être immunisé contre les bruits ou parasites.

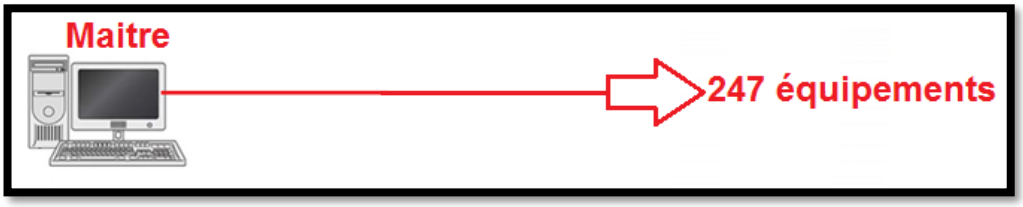
Figure 8: Le RS485



4.2.Les spécificités du Modbus via interface série RS-xxx :

En Modbus série, seul le maître est actif, les esclaves sont complètement passifs. C'est le maître qui doit lire et écrire dans chaque esclave. Le maitre peut communiquer avec un nombre d'esclaves allant jusqu'à 247 (cas du Modbus via RS-485 avec l'utilisation de répéteurs) sur le même réseau. Les adresses allant de 248 à 255 sont des adressées réservées.

Figure 9: Modbus via RS-485



Le RS485 ne peut pas comporter plus de 32 périphériques sur le même nœud, on utilise alors des répéteurs afin de pouvoir ajouter d'autres périphériques sur la ligne.

Figure 10: RS484- Périphérique

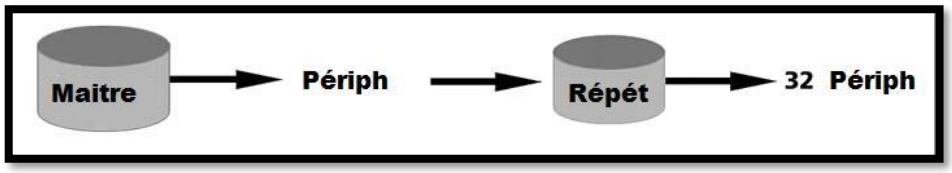
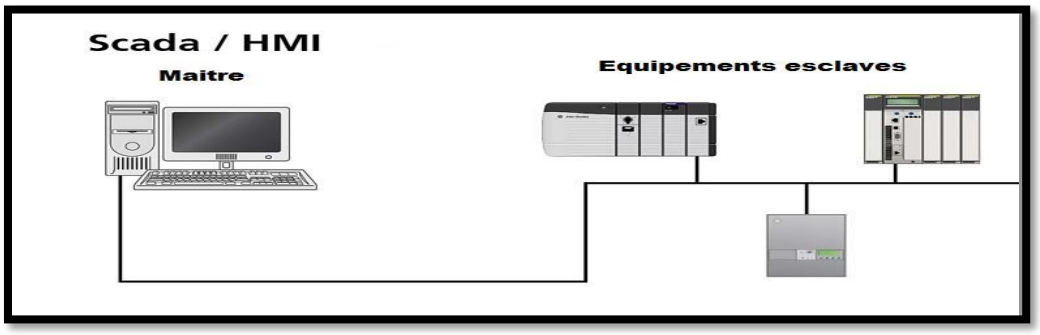


Figure 11:Topologie d'équipements connectes via le Modbus série Rs-xxx



4.3.Les messages de Broadcaste :

Aussi appelé message de diffusion est une communication unidirectionnelle initiée par le maitre et envoyé à tous les esclaves. Ce type de message n'obtient pas de réponse de la part des esclaves, il est utilisé pour envoyer des commandes communes à tous les esclaves par exemple les commandes de configuration ou de réinitialisation.

5. Fonctions Modbus :

Modbus est un protocole de requête / réponse qui offre des services définis par codes de fonction. 127 codes de fonction publics et définis par l'utilisateur sont disponibles et peuvent être utilisés pour l'accès aux données (lecture écriture), le diagnostic et autres services.

5.1. Registre Modbus :[10]

Les registres Modbus d'un appareil sont organisés autour des quatre types de référence de données de base indiqués ci-dessus et ce type de données est en outre identifié par le numéro de début de l'adresse de référence.

Le caractère «x» représente un emplacement d'adresse à quatre chiffres dans la mémoire de données de l'utilisateur.

Le caractère de début est généralement impliqué par le code de fonction et omis du spécificateur d'adresse pour une fonction donnée. Le caractère principal identifie également le type de données d'E / S.

[10]http://www.acromag.com/ Copyright 2005, Acromag, Inc., Printed in the USA.Data and specifications are subject to change without notice.

Figure 12: Description des caractères de référence.

Référence	Description
0xxxx	Lecture / écriture de sorties ou bobines discrètes. Une adresse de référence 0x est utilisée pour piloter les données de sortie sur un canal de sortie numérique.
1xxxx	Lecture d'entrées discrètes. L'état ON / OFF d'une adresse de référence 1x est contrôlé par le canal d'entrée numérique correspondant.
3xxxx	Lecture des registres d'entrée. Un registre de référence 3x contient un nombre de 16 bits reçu d'une source externe, par exemple. Un signal analogique
4xxxx	Lecture / écriture sortie ou registres de maintien. Un registre 4x est utilisé pour stocker 16 bits de données numériques (binaires ou décimales) ou pour envoyer les données de la CPU à un canal de sortie.

5.2.Architecture d'un service de messagerie Modbus TCP:

Évidemment la communication Modbus TCP est basée sur le modèle client/serveur. Un équipement Modbus peut donc intégrer à la fois un module client et un module serveur, mais cela n'est pas obligatoire. Un équipement peut très bien n'intégrer qu'un seul de ces deux rôles.

Parmi les principales fonctions implémentées par un service de messagerie Modbus TCP figurent l'établissement et la terminaison des communications, ainsi que la gestion des flots des données (contrôle de flux) parcourant les connexions TCP établies.

La communication entre un client et un serveur Modbus requiert la mise en place d'un système de gestion des connexions TCP. Deux options sont envisageables :

- Soit c'est l'application qui se charge de cette tâche (programmation par sockets et gestion des mécanismes TCP/IP).
- Soit cette gestion est réalisée au travers d'un module dédié, baptisé TCP connexion Management, inclus au niveau de la couche TCP Management de l'architecture composant Modbus. Dans ce cas, la gestion des connexions devient totalement

transparente pour l'application, qui se contente d'envoyer et recevoir les messages Modbus.

Pour permettre l'établissement des connexions et l'échange de données entre équipements, le service de messagerie Modbus TCP doit fournir un socket d'écoute sur le port 502. Il est important de noter que le port d'écoute 502 TCP est réservé aux communications Modbus.

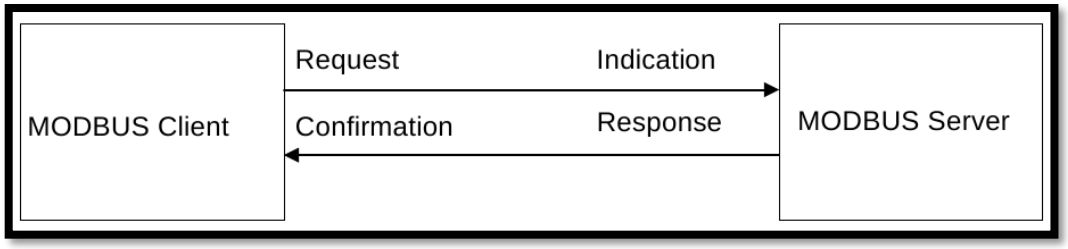
Dans certains contextes critiques, l'accès aux données internes des équipements doit être interdit aux hôtes indésirables. C'est pourquoi un module de contrôle d'accès (Access CTL) peut être implémenté si nécessaire.

Si le nombre de connexions client et serveur est supérieur au nombre de connexions autorisées (de1 à 16), la plus ancienne connexion non utilisée est fermée. Les mécanismes de contrôle d'accès peuvent par ailleurs être activés pour vérifier que les adresses IP des clients distants sont bien autorisées. Par défaut, lorsque le mode de sécurité est activé, les adresses IP non-configurées par l'utilisateur sont interdites d'accès.

Le module client Modbus construit une requête sur la base des informations transmises au travers de l'interface client Modbus. Cette interface fournit une API (Application Programming Interface) permettant à l'application de construire des requêtes pour accéder à différents services Modbus.

Un module serveur Modbus est, quant à lui, chargé de recevoir les requêtes et de mettre en œuvre des actions (de lecture et d'écriture notamment) afin d'y répondre. L'exécution de ces actions est réalisée de façon totalement transparente pour le programmeur de l'application. Les fonctions principales d'un serveur Modbus sont l'attente de requêtes sur le port 502 TCP, ainsi que le traitement de ces requêtes et la construction de réponses Modbus en fonction du contexte dans lequel se trouve l'équipement. Entre le module serveur Modbus et l'application on peut également trouver une interface baptisée Modbus Backend, qui permet un accès indirect aux objets de l'application.

Figure 13:schéma block de communication Modbus TCP



Source : exercice mode bus TCP.

5.3.Description d'un échange :

C’est le client (qui initie la transaction Modbus) qui construit l’ADU. Le champ Function Code du PDU indique au serveur le type d’action à mener.

A la réception d’une requête du client, le serveur analyse l’entête MBAP de l’ADU Modbus. Si celle-ci correspond bien à un entête Modbus, une transaction Modbus est instanciée.

Dans le cas où le nombre maximum de transactions simultanée autorisé est dépassé, le serveur construit une réponse d’exception (Exception Code 6 : Server Busy).

Dans le cas contraire, une transaction est instanciée et initialisée avec les informations suivantes : l’identifiant de la connexion TCP utilisée (fournie par le module TCP Management), l’ID de la transaction Modbus (contenu dans l’entête MBAP) et l’Unit Identifier (contenu également dans l’entête MBAP). Ensuite, c’est au tour du champ PDU d’être analysé. Si celui-ci est reconnu comme un champ valide, le serveur est prêt à exécuter le service demandé.

Une fois que la requête a été traitée, le serveur Modbus construit une réponse qu’il doit ensuite envoyer au module TCP Management. En fonction du résultat du traitement, deux types de réponses sont envisageables : une réponse positive ou une réponse d’exception, dont l’objectif est de fournir au client des informations concernant les erreurs détectées durant le traitement de la requête. Le PDU de la réponse Modbus doit être préfixé d’une entête MBAP construite à partir des informations mémorisées lors de la réception de la requête : Unit Identifier, Protocol Identifier et Transaction Identifier. Le serveur renseigne également le champ « Length », indiquant la longueur des champs PDU+Unit Identifier. Enfin, la réponse Modbus est envoyée au client.

A la réception de la réponse, le client Modbus utilise le Transaction Identifier pour retrouver la requête correspondante, précédemment envoyée sur la connexion TCP. Si celui-ci ne correspond à aucune requête connue du client, la réponse est ignorée et mise au rebut.

Dans le cas contraire, la réponse est analysée et utilisée par le client Modbus pour construire la confirmation qui sera ensuite envoyée à l’application.

L’analyse de la réponse consiste à vérifier l’entête MBAP ainsi que le PDU de la réponse Modbus. Si celle-ci provient d’un serveur Modbus directement connecté au réseau TCP/IP, l’identification de la connexion TCP utilisée est suffisante pour identifier le serveur distant sans aucune ambiguïté. Dans ce cas l’Unit Identifier n’est pas significatif et doit être ignoré.

Par contre, si le serveur est connecté à un sous-réseau sur liaison série, et que la réponse provient d’un routeur, l’Unit Identifier (qui dans ce cas ne sera pas égal à 0xFF) devra être pris en compte pour l’identification du serveur par le client. Pour ce qui est de l’analyse du

PDU de la réponse, le client vérifie la conformité du Function Code ainsi que du format de la réponse. Si le Function Code est le même que celui contenu dans la requête, et si le format de la réponse est correct, le client envoie une confirmation positive à l'application. Si au contraire le Function Code contenu dans la réponse est différent de celui contenu dans la requête originale, ou si le format de la réponse est incorrect, le client signale une erreur à l'application en lui envoyant une confirmation négative. Enfin, si le Function Code correspond à un code d'exception, une réponse d'exception Modbus est transmise à l'application.

Il est important de noter qu'une confirmation positive indique seulement à l'application que le serveur a bien reçu la requête et qu'il y a répondu. Une telle confirmation ne la renseigne en aucun cas sur le succès ou l'échec des actions menées, ceci étant du ressort des réponses d'Exception Modbus.

Conclusion :

Dans ce chapitre, nous avons présenté des notions de basse sur le réseau et les différentes couches de communications, et aussi les model de communication (OSI, TCP, Modbus TCP/IP).

CHAPITRE 03

Chapitre 2 : Automatisation Et Arduino

Un système automatisé est un système technique qui permet de passer d'une situation à une autre sans l'intervention humaine et exécuter toujours le même cycle de travail pour lequel il a été programmé.

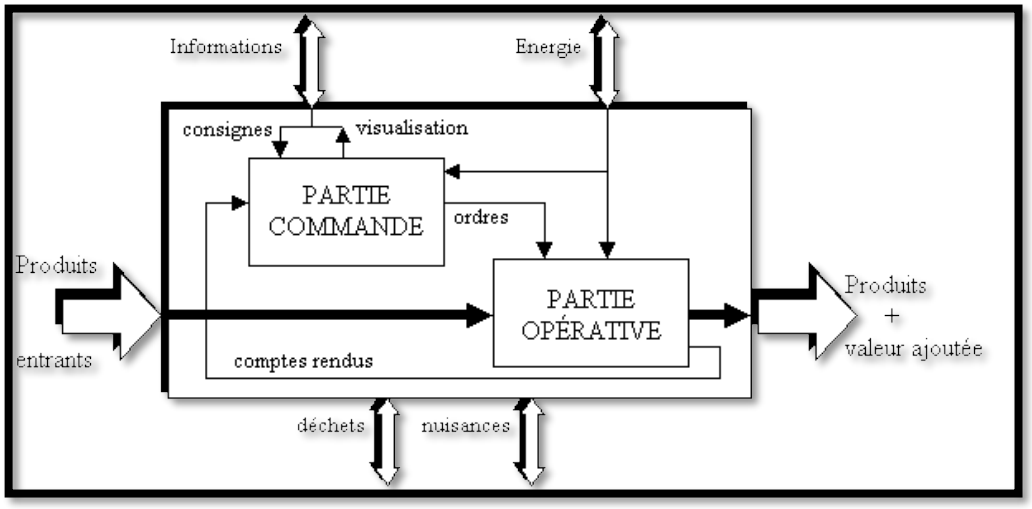
En d'autres termes, c'est un système qui à partir des informations qui lui est fourni, effectue des actions prédéfinies sur son environnement. Ces actions sont mises en oeuvre selon une procédure précise qui dépend des informations fournies et des paramètres calculés ou prédéfinis.

Section 1 : Description d'un système automatisé :

L'objectif de l'automatisation est de fournir des produits de qualité pour un coût le plus faible en moins de temps possible de façon automatique, sans faire intervenir l'homme en tant de moyens de production.

Un système automatisé est composé de deux parties comme présenter sur la figure suivant :

Figure 14: Composition du système automatisé



1. La Partie Commande :

Elle est en général composée d'un automate qui contient un programme qui gère le fonctionnement du Système. Elle sélectionne les ordres nécessaires au fonctionnement de la partie opérative en fonction des consignes qu'elle reçoit du dialogue homme-machine (par l'opérateur) ou par acquisition des données (les informations reçues par des capteurs).

2. La Partie Opérative :

Cette partie exécute les ordres reçus de la partie commande, elle transforme les signaux de commande en énergie électrique, pneumatique ou hydraulique pour réaliser le fonctionnement du système. En même temps, elle transmet l'état du système à la partie commande à travers les capteurs.

Elle comporte en général :

- Des actionneurs qui transforment l'énergie reçue en énergie utile : moteur, vérin, lampe.
- Des capteurs qui transforment la variation des grandeurs physiques liée au fonctionnement du système en signaux électriques : capteur de position, de température, bouton poussoir…

3. Définition de l'automate :

L'Automate Programmable Industriel (API) est un dispositif électrique de traitement logique d'informations dont le programme de fonctionnement est effectué à partir d'instructions établies en fonction du processus à réaliser. Il est adapté à l'environnement industriel. Il génère des ordres vers les pré-actionneurs de la partie opérative à partir de données d'entrées (capteurs) et d'un programme.

Figure 15 : Model API.

4. Choix de l'automate :

Pour choisir un automate, il faut prendre en considération les besoins comme :

- Le type de processeur : La taille mémoire, la vitesse de traitement et les fonctions spéciales offertes par le processeur permettront le choix souvent très étendu dans la gamme.
- Le nombre d'entrée / sortie : Le nombre de cartes peut avoir une incidence sur le nombre de racks, dès que le nombre d'entrée / sortie nécessaire devient élevé.

5. Structure des API :

5.1.Structure général des API :

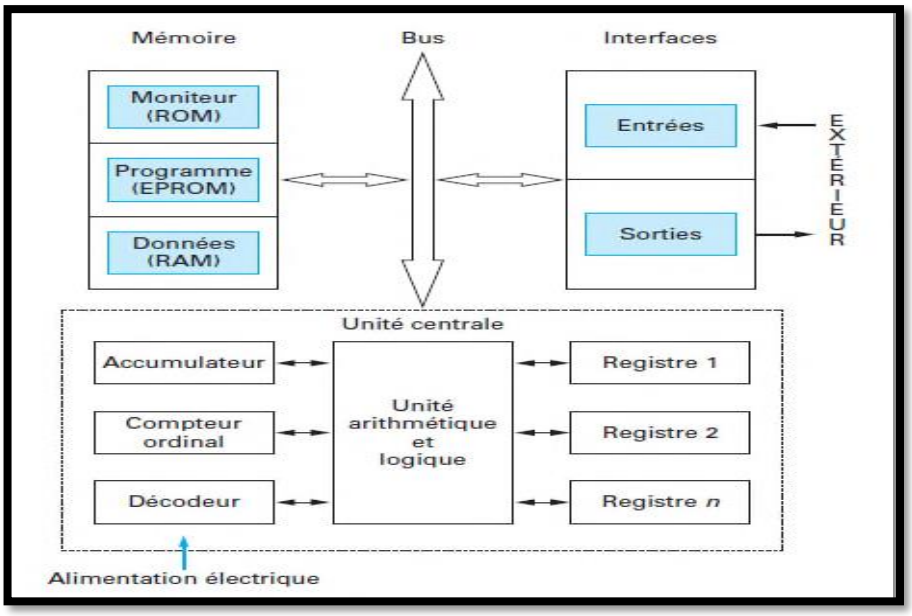
Les composants principaux d'un automate programmable industriel (API) sont :

- Coffret, rack, baie ou cartes.
- Compact ou modulaire.
- Tension d'alimentation.
- Taille mémoire.
- Sauvegarde (EPROM, EEPROM, pile, …).
- Nombre d'entrées / sorties.
- Modules complémentaires (analogique, communication...).
- Langage de programmation.

5.2. Structure interne d'un API :

Les API comportent principalement les parties suivantes :

Figure 16 : Structure Interne.

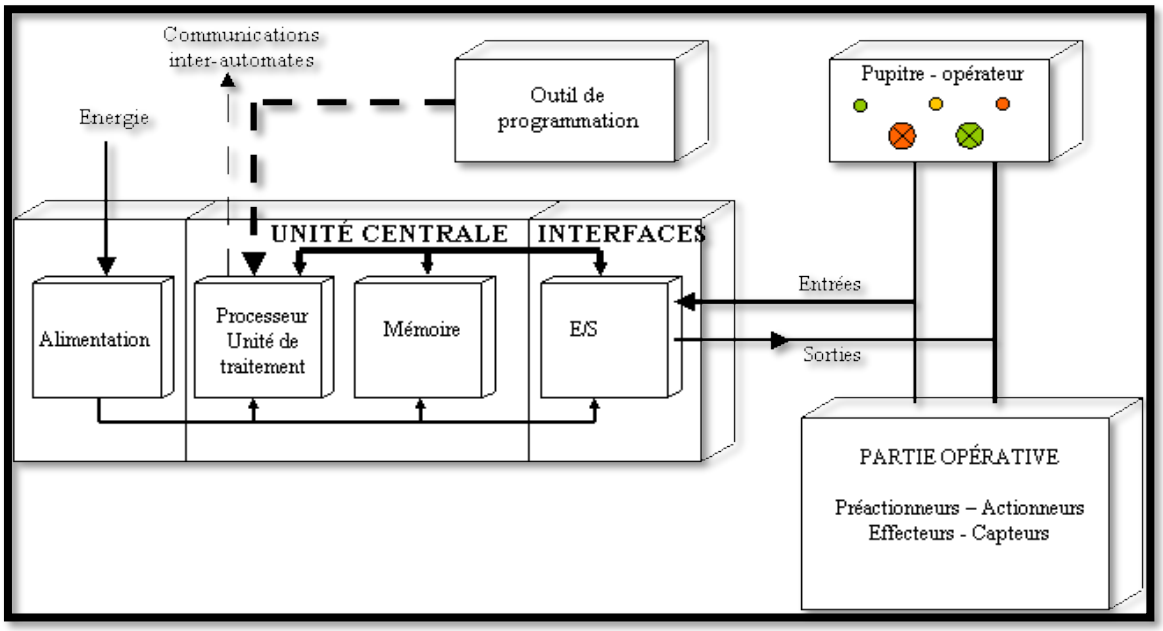


- Une unité de traitement (un processeur CPU)
- Une mémoire
- Des modules d'entrées-sorties
- Des interfaces d'entrées-sorties
- Une alimentation 230 V, 50/60 Hz (AC), 24 V (DC)

La structure interne d'un automate programmable industriel (API) est assez voisine de celle d'un système informatique simple, l'unité centrale est le regroupement du processeur et de la mémoire centrale. Elle commande l'interprétation et l'exécution des instructions programme.

6. Description des éléments d'un API[11] :

Figure 17: schéma représentatif des éléments d'un API

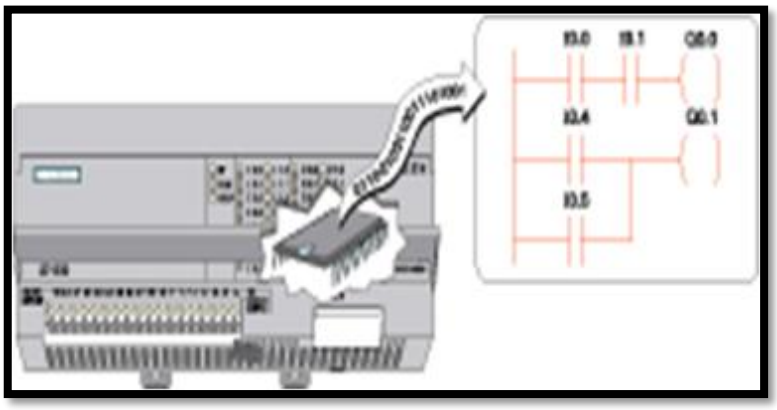


[11]http://jacques.boudier.pagesperso-orange.fr/c_online/Informatique_industrielle/automate/cours%20008.htm

6.1. La mémoire[12] :

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

Figure 18: La mémoire



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

- **La mémoire Langage** où est stocké le langage de programmation. Elle est en général figée, c'est à dire en lecture seulement. (ROM : mémoire morte)
- **La mémoire Travail** utilisable en lecture-écriture pendant le fonctionnement c'est la RAM (mémoire vive). Elle s'efface automatiquement à l'arrêt de l'automate (nécessite une batterie de sauvegarde

6.2.Le processeur :

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

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

- L'interface d'entrée comporte des adresses d'entrée.
- L'interface de sortie comporte de la même façon des adresses de sortie.
- Le nombre de ces entrées et sorties varie suivant le type d'automate.
- Les cartes d'E/S ont une modularité de 8, 16 ou 32 voies.

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

- Les tensions disponibles sont normalisées (24, 48, 110 ou 230V continu ou alternatif).

6.3.1. Cartes d'entrées :

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

6.3.2. Cartes de sorties :

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

7. Critères de choix de l'automate :

Il revient à nous d'établir le cahier des charges de notre système et de chercher sur le marché l'automate le mieux adapté à nos besoins. Cela est fait en considérant un certain nombre de critères importants :

- Avoir les compétences et l'expérience nécessaire pour programmer la gamme d'automate.
- Le nombre et le type d'entrées et de sorties nécessaires.
- La communication envisagée avec les autres systèmes.
- Les capacités de traitement de la CPU.
- Les moyens de sauvegarde.
- La fiabilité et la robustesse.
- Le cout d'investissement, de fonctionnement, de maintenance de l'équipement.
- La qualité du service après-vente.

Section 2 : Automate s7 1200 :

L'automate SIMATIC S7-1200 fabriqué par SIEMENS est un automate de conception modulaire et compact, polyvalent, destiné à des taches d'automatisation simple mais d'une précision extrême, il constitue donc, un investissement sûr et une solution parfaite à une grande variété d'applications.

Une conception modulaire et flexible, une interface de communication répondant aux exigences les plus sévères dans l'industrie et une large gamme de fonctions technologiques performantes et intégrées, font de cet automate, un composant à part entière d'une solution d'automatisation complète.

Figure 19: Automate S7 1200

Les CPU du système SIMATIC S7-1200 se déclinent en trois classes de performances : CPU 1211 C, CPU1212 C et CPU1214 C, chacune d'elles pouvant être étendue en fonction des besoins de la station. Sur chaque CPU, il est possible de greffer une platine d'extension pour ajouter des E/S TOR ou analogiques supplémentaires sans modification de l'encombrement de l'automate. Des modules d'E/S supplémentaires peuvent être ajoutés du côté droit de la CPU pour étendre la capacité d'E/S TOR ou analogiques.

1. Défirent CPU S7-1200 :

Les CPU de l'automate S7-1200 sont données dans le tableau suivant :

Figure 20: Tableau des CPU de l'automate S7-1200

CPU	**Mémoire de travail**	**E/S TOR**	**Modules E/S extensible**	**Prix**
CPU 1211C	50 Ko	6 entrées / 4 sorties	Aucune	**25 873.85 DA**
CPU 1212C	75 Ko	8 entrées / 6 sorties	2 modules	**34 763.08 DA**
CPU 1214C	100 Ko	14 entrées/ 10 sorties	8 modules	**43 868.37 DA**
CPU 1215C	125 Ko	14 entrées/ 10 sorties	8 modules	**66 135.91 DA**
CPU 1217C	150 Ko	14 entrées	8 modules	**91 603.72 DA**

2. Présentation des différents modules :

SIMATIC S7-1200 est un système d'automatisation modulaire et offre la gamme de modules suivants :

- Unités centrales (CPU) avec différentes capacités, entrées/sorties intégrées et interfaces PROFINET (par exemple CPU 1214C)

Figure 21:CPU S71200.

- Module de puissance PM avec une tension d'alimentation de 120/230V CA, 50Hz/60Hz, 1.2A/0.7A, et une tension de sortie 24V CC/2.5A

Figure 22: Alimentation.

- Signal Boards SB permet d'ajouter des entrées ou des sorties analogiques ou TOR sans pour autant modifier la taille de la CPU. (Les Signal Boards peuvent être intégrés dans les CPU 1211C/1212C et 1214C).

- Les modules signaux SM pour les entrées et sortie digitales et analogiques (2 SM max. possible pour les CPU 1212C et 8 SM max. pour la 1214C).

- Les modules de communication CM pour une communication sérielle RS 232 / RS 485

  (jusqu'à 3 CM son possible pour les CPU (1211C/1212C et 1214C).

- Les cartes mémoires SIMATIC 2 Mo jusqu'à 32 Mo pour enregistrer les données du programme et pour le remplacement aisé des CPU en cas de maintenance.

3. Présentation la CPU 1214C DC/DC/DC :

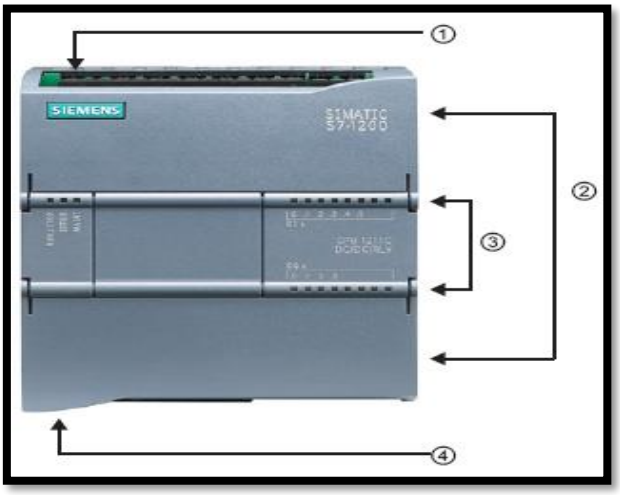
Avec une alimentation intégrée de 24V et des entrées et sorties TOR intégrées, la CPU 1214C DC/DC/DC est prête à l'emploi, sans que des composants supplémentaires soient nécessaires.

Pour communiquer avec une console de programmation, la CPU est équipée d'un port TCP/IP intégré.

Au travers d'un réseau ETHERNET, la CPU est capable de communiquer avec des pupitres opérateurs IHM ou avec d'autres CPU.

- Alimentation 24V
- Borniers enfichables pour un câblage utilisateur (derrière les caches plastiques)
- LED d'état des E/S intégrées et pour les modes de fonctionnement de la CPU
- Port TCP/IP (sous la CPU).

Figure 23: CPU S7 1200.

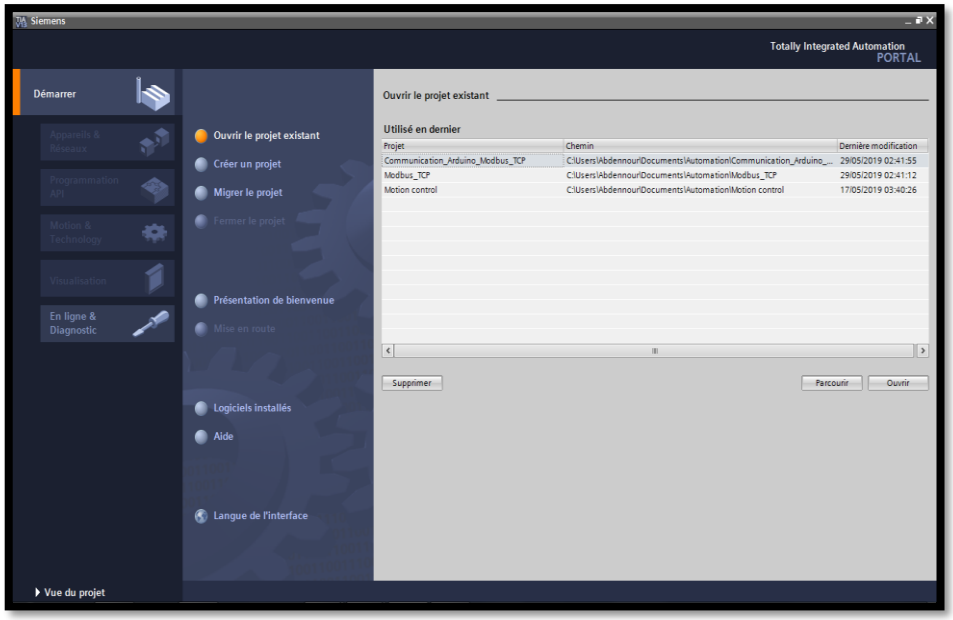


4. Logiciel de programmation STEP 7 Basic V13 (TIA Portal V13) :

Pour la programmation, on a utilisé le logiciel de Siemens STEP 7 Professional V13 (TIA Portail V13.

Le portail Totally Integrated Automation, ci-après appelé portail TIA, offre la fonctionnalité complète pour réaliser notre tâche d'automatisation, regroupée dans une plateforme logiciel globale. Le portail TIA permet également de disposer, au sein d'un cadre, d'un environnement de travail commun pour une ingénierie transparente avec différents systèmes SIMATIC. Tous les progiciels requis, de la configuration matérielle à la visualisation du processus en passant par la programmation, sont intégrés dans un cadre complet d'ingénierie.

Figure 24: STEP 7 Professional V13 (TIA Portail V13) :



Le logiciel STEP 7 Professional (TIA Portal V13) est l'outil de programmation des nouveaux automates comme :

- SIMATIC S7-1500
- SIMATIC S7-1200
- SIMATIC S7-300
- SIMATIC S7-400

Avec STEP 7 Professional (TIA Portal), les fonctions suivantes peuvent être utilisées pour automatiser une installation :

- Configuration et paramétrage du matériel.
- Paramétrage de la communication.
- Programmation.
- Test, mise en service et dépannage avec les fonctions d'exploitation et de diagnostic
- Documentation.
- Génération d'écrans de visualisation pour les Basic Panels SIMATIC avec Win CC Basic intégré.

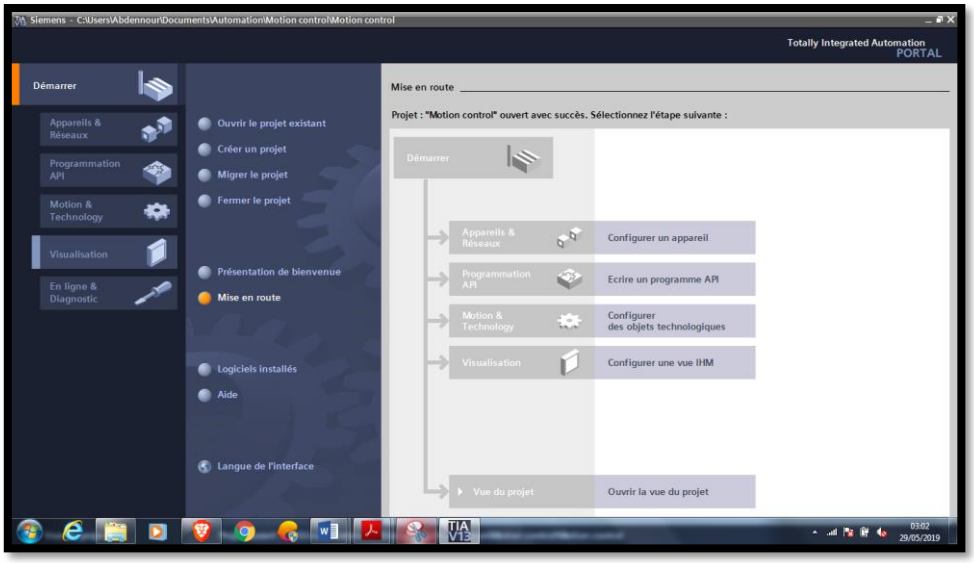
4.1.TIA Portal – Vue du projet et vue du portail :

La vue du projet, représentée dans la **Erreur ! Source du renvoi introuvable.**, est utilisée pour la c onfiguration matérielle, la programmation, la création de la visualisation et pour nombre d'autres tâches avancées.

La barre de menu avec les barres de fonction est située, comme le veut la norme, en haut de la fenêtre, le navigateur du projet et tous les éléments du projet sont sur la gauche, et les menus associés aux différentes tâches (avec les instructions et les bibliothèques, par exemple) sur la droite.

Si un élément (par exemple ici la configuration de l'appareil) est sélectionné dans le navigateur du projet, il est affiché au centre et peut y être édité.

Figure 25: Vue du portail



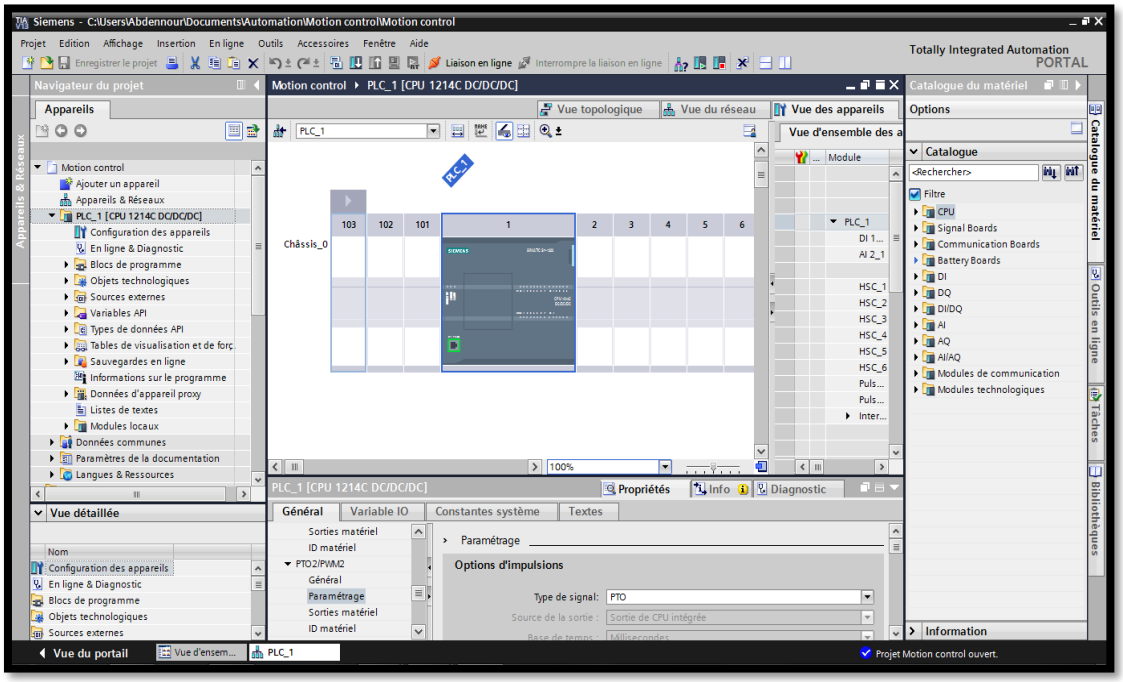
4.2.La configuration matérielle :

La configuration matérielle est une étape qui correspond à l'arrangement des modules et de la périphérie décentralisée. Ces modules sont fournis avec des paramètres définis par défaut en usine. Elle est nécessaire pour :

- Configurer les paramètres ou les adresses préréglées d'un module.
- Configurer les liaisons de communication.

L'analyse de la configuration de la station existante a conduit au choix de la configuration illustrée dans la figure suivante :

Figure 26:La configuration matérielle du projet



4.3.Définir l'adresse IP sur la console de programmation :

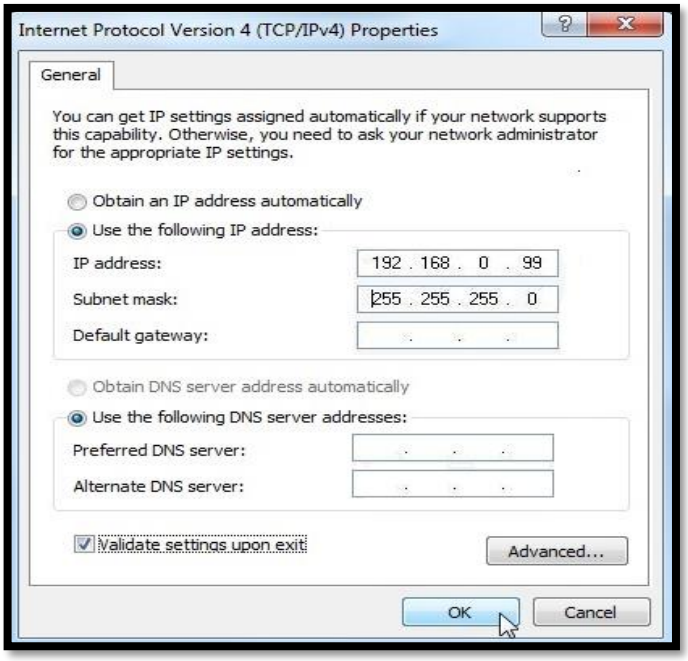
Pour programmer le SIMATIC S7-1200 à partir d'un PC, d'une PG, vous avez besoin d'une connexion TCP/IP ou d'une connexion PROFIBUS.

Pour que le PC et SIMATIC S7-1200 puissent communiquer via TCP/IP, il est important que leurs adresses IP correspondent.

Il s'agit ici premièrement de montrer comment l'adresse IP du PC peut être paramétrée par le système d'exploitation Windows 7.

- Repérez le symbole représentant le réseau sur la barre d'outils  et cliquez ensuite sur "Open Network and Sharing Center" (Ouvrir le Centre Réseau et partage).
- Dans la fenêtre du centre Réseau et partage, cliquez sur "Change adapter settings" (Modifier les paramètres de la carte).
- Sous "local area connection" (Connexion au réseau local), choisissez celle avec laquelle vous voulez vous connecter à l'automate et cliquez sur "Properties" (Propriétés).

- Choisissez maintenant les "Properties" (Propriétés) de « Internet Protocol (TCP/IPv4)".
- Vous pouvez ensuite paramétrer, par ex., l'adresse IP suivanteIP Address (Adresse IP) : **192.168.0.99** Subnet mask (Masque de sous-réseau) **255.255.255.0** et les appliquer en cliquant sur OK « "**OK**" ».



4.4.Le programme :

Chaque automate possède son propre langage. Mais par contre, les constructeurs proposent tous une interface logicielle répondant à la norme CEI1 1131-3. Cette norme définit cinq langages de programmation utilisables.

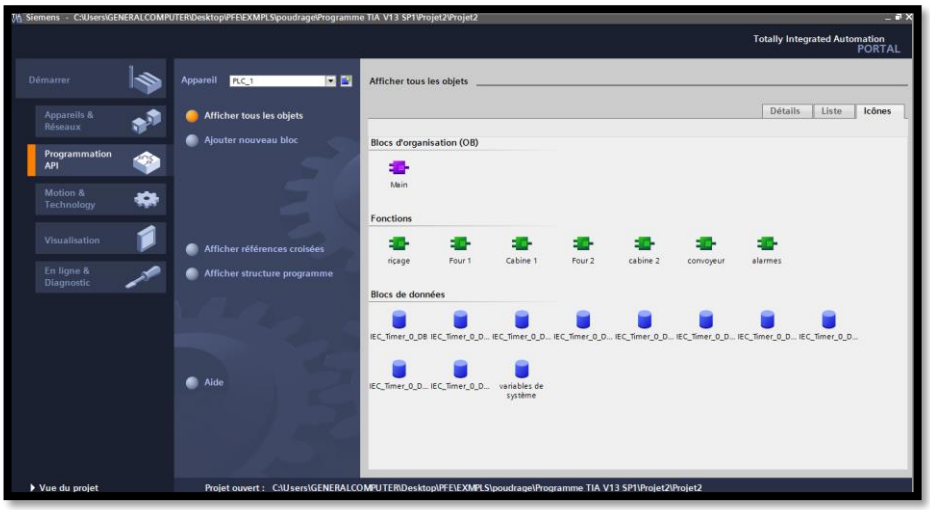
Dans ce projet, on choisit le langage de programmation CONT car les automates S7-1200 ne peuvent être programmés que par les langages CONT, LOG ou SCL.

- **GRAFCET (SFC) :** Ce langage de programmation de haut niveau permet la programmation de tous les procédés séquentiels.
- **Schéma blocs (FBD) :** Ce langage permet de programmer graphiquement à l'aide de blocs, représentant des variables, des opérateurs ou des fonctions. Il permet de manipuler tous les types de variables.
- **Schéma à relais (LD) :** Ce langage graphique est essentiellement dédié à la programmation d'équations booléennes.
- **Texte structuré (ST) :** Ce langage est un langage textuel de haut niveau. Il permet la programmation de tout type d'algorithme plus ou moins complexe.

- **Liste d'instructions (IL) :** Ce langage textuel de bas niveau est un langage à une instruction par ligne. Il peut être comparé au langage assembleur.

La structure générale du projet est représentée par la Figure ci-dessous :

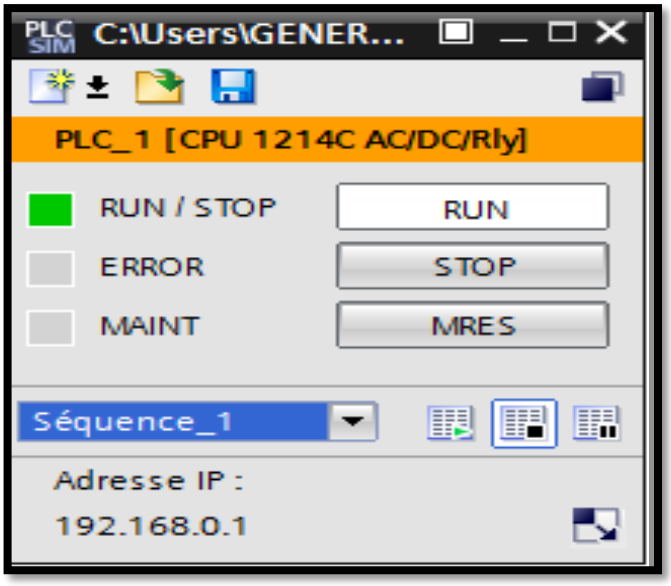
Figure 27 : La structure du notre projet.



4.5.Simulation du programme :

Pour la CPU, la simulation est complètement réalisée au sein du logiciel TIA Portal V13. En effet, S7-PLCSIM dispose une interface comportant une CPU S7-1214C virtuelle et des modules d'entrées/sorties qui permettent de visualiser et forcer les différents états du programme.

Figure 28 : Le simulateur de S7-1214C



Section 3 : Généralité sur Arduino

Les cartes Arduino font partie de la famille des microcontrôleurs. Un microcontrôleur est une petite unité de calcul accompagné de mémoire, de ports d'entrée/sortie et de périphériques permettant d'interagir avec son environnement. Parmi les périphériques , ont recense généralement des Timers, des convertisseurs analogique-numérique, des liaisons Séries, etc. On peut comparer un micro contrôleurs à un ordinateur classique, mais système d'exploitation et avec une puissance de calcul considérablement plus faible.

Les microcontrôleurs sont inévitables dans les domaines de l'informatique embarquée, de l'automatique et de l'informatique industrielle. Ils permettent de réduire le nombre de composant et de simplifier la création de cartes électroniques logiques.

1. Définition Arduino :

Arduino est une plate-forme de prototypage d'objets interactifs à usage créatif constituée d'une carte électronique et d'un environnement de programmation.

Sans tout connaître ni tout comprendre de l'électronique, cet environnement matériel et logiciel permet à l'utilisateur de formuler ses projets par l'expérimentation directe avec l'aide de nombreuses ressources disponibles en ligne.

2. Caractéristiques techniques de l'Arduino UNO :

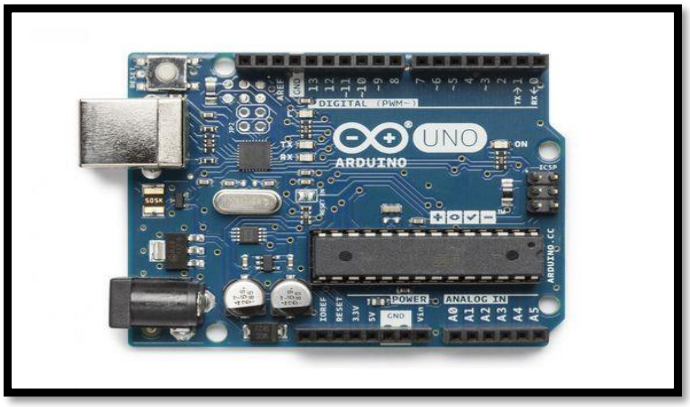
Un des modèles les plus répandu de carte Arduino est l'Arduino UNO .C'est la première version stable de carte Arduino.

Elle possède toutes les fonctionnalités d'un microcontrôleur classique en plus de sa simplicité d'utilisation. Elle utilise une puce ATmega328P cadencée à 16Mhz. Elle possède 32ko de mémoire flash destinée à recevoir le programme, 2 ko de SRAM (mémoire vive) et 1 ko d'EEPROM (mémoire morte destinée aux données). Elle offre 14 pins (broches) d'entrée/sortie numérique (données acceptée 0 ou 1) dont 6 pouvant générer des PWM (Puls e Width Modulation, détaillé plus tard).

Elle permet aussi de mesurer des grandeurs analogiques grâce à ces 6 entrées analogiques.

Chaque broche est capable de délivré un courant de 40mA pour une tension de 5V. Cette carte Arduino peut aussi s'alimenter et communiquer avec un ordinateur grâce à son port USB. On peut aussi l'alimenter avec unes alimentions comprise en 7V et 12V grâce à sa connecteur Power Jack.

Figure 29 : Arduino UNO.



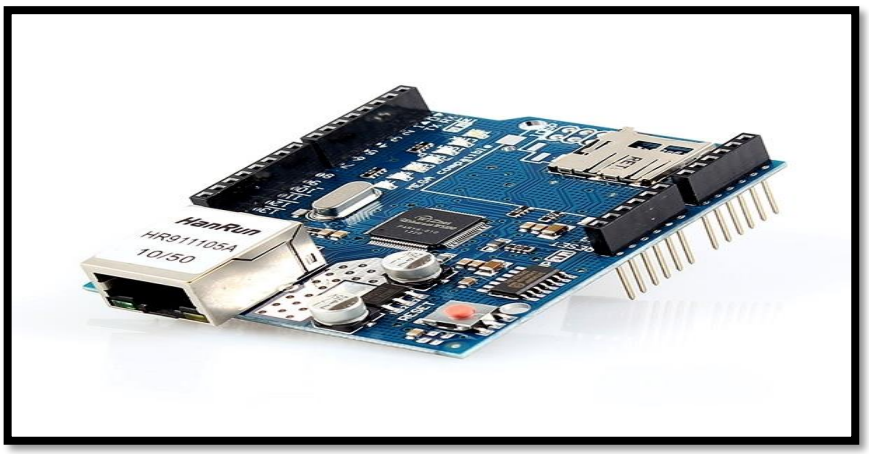
3. Les Shields :

Pour la plupart des projets, il est souvent nécessaire d'ajouter des fonctionnalités aux cartes Arduino. Plutôt que d'ajouter soit même des composants extérieurs (sur une platine d'essai, circuit imprimé, etc.), il est possible d'ajouter des shields.

Un shield est une carte que l'on connecte directement sur la carte Arduino qui a pour but d'ajouter des composants sur la carte. Ces shields viennent généralement avec une librairie permettant de les contrôler. On retrouve par exemple, des Shields Ethernet, de contrôle de moteur, lecteur de carte SD, etc.

Le principal avantage de ces shields est leurs simplicités d'utilisation. Il suffit des les emboiter sur la carte Arduino pour les connecter, les circuits électronique et les logiciel sont déjà faits et on peut en empiler plusieurs. C'est un atout majeur pour ces cartes pour pouvoir tester facilement de nouvelles fonctionnalités. Cependant il faut bien garder à l'esprit que les shield sont un prix. Suivant les composants qu'ils apportent, leurs prix peuvent aller de 5 à 100€ ! Exemple Shield Ethernet Utiliser dans notre application :

Figure 30 : Shield Ethernet.



4. Présentation du logiciel :

Le logiciel Arduino est un environnement de développement (IDE) open source et gratuit, téléchargeable sur le site officiel Arduino.

L'IDE Arduino permet :

- d'éditer un programme des croquis (sketch en Anglais),
- de compiler ce programme dans le langage « machine » de l'Arduino,
- de télé verser le programme dans la mémoire de l'Arduino,
- de communiquer avec la carte Arduino grâce au terminal.

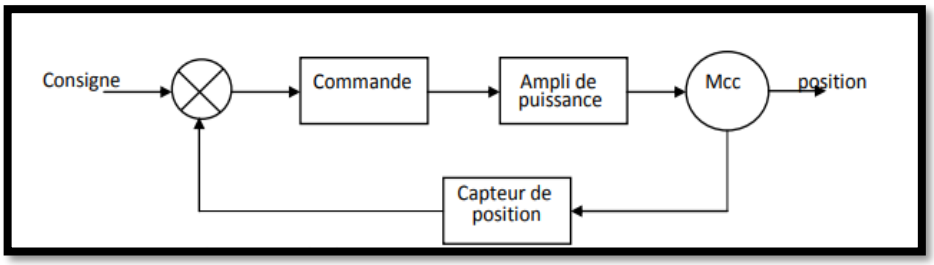
Figure 31: IDE Arduino.

5. Moteur pas à pas :

Le moteur pas à pas, ou encore stepper -motor, est un moteur particulier pour lequel sa rotation s'effectue par pas. C'est à dire, qu'à l'inverse d'un petit moteur à courant continu comme on trouve dans nombreux jouets et qui tourne sans arrêt un fois alimenté, le moteur pas à pas tourne d'un angle spécifique à chaque fois qu'il reçoit une impulsion. Mais comme nous allons le découvrir, ces impulsions ont besoin d'un formatage bien particulier.

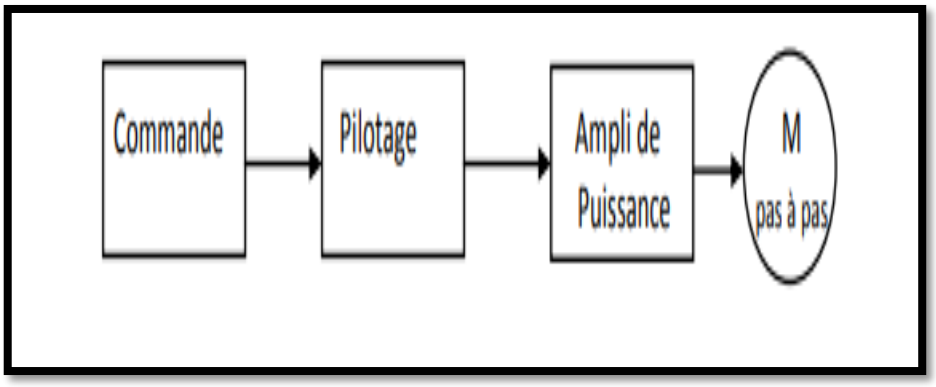
5.1. Principe de commande en position d'un moteur à courant continu :

Figure 32:Principe de commande en position d'un moteur à courant



5.2. Principe de commande d'un moteur pas à pas

Figure 33: Principe de commande d'un moteur pas à pas



On constate que le système est beaucoup plus simple. En effet, à chaque impulsion du signal de commande correspond au niveau du rotor un déplacement angulaire bien défini appelé « pas ». Un moteur pas à pas est caractérisé par sa résolution ou encore son nombre de pas par tour. Il peut avoir une valeur comprise entre 0,9°et 90°. Les valeurs les plus couramment rencontrées sont :

- 0,9° : soit 400 pas par tour
- 1,8° : soit 200 pas par tour
- 3,6° : soit 100 pas par tour
- 7,5° : soit 48 pas par tour
- 15° : soit 24 pas par tour

La vitesse de rotation est fonction de la fréquence des impulsions. On distingue 3 groupes de moteur pas à pas :

- les moteurs à aimant permanent (bipolaire, unipolaire) .
- les moteurs à reluctance variable.
- les moteurs hybrides.

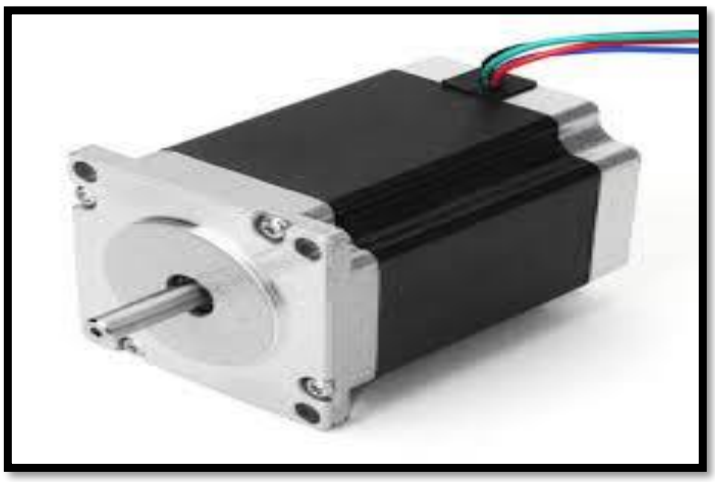
5.2.1. Moteur bipolaire :

Moteur bipolaire a deux enroulements, correspondant chacune à une phase. Il ya quatre fils, deux pour chaque enroulement. Le contrôle est effectué est nécessairement bipolaire, généralement avec un cavalier.

5.2.2. Moteur unipolaire:

La bobine pour chaque phase est double, ainsi que l'intérieur et placés en série nous donne 6 fils, groupées par trois, pour chaque phase (dont l'un est commun). Le contrôle est unipolaire, mais vous pouvez faire un numéro de contrôle bipolaire laissant le fil conducteur dans l'air, et non conçus à cette fin

Figure 34 : Moteur pas a pas



6. Caractéristiques :

Taille: 56,4 mm × 76 mm, arbre non compris (NEMA 23)
Poids: 1 kg (35 oz)
Diamètre de l'arbre: 6,35 mm (0,25 ″) “D”
Pas par tour: 200
Courant nominal: 2,8 A par bobine
Tension nominale: 3,2 V
Résistance: 1,13 Ω par bobine
Couple de maintien: 19 kg-cm (270 oz-in)
Inductance: 3,6 mH par bobine
Longueur du fil: 30 cm (12 ″)

7. Contrôleurs de Moteurs Pas-à-Pas tb6600 :

7.1. Caractéristiques:

Tension de fonctionnement: DC 10V-45V.

Il est recommandé d'utiliser l'alimentation du commutateur d'alimentation DC32V.

L'utilisation du couplage optique haute vitesse 6N137 garantit une vitesse élevée sans perte de vitesse.

L'utilisation de la nouvelle puce d'origine haute tension TB6600HG haute tension actuelle, il existe un circuit de protection contre la basse pression, le stationnement en surchauffe et la protection contre les surintensités, la protection contre les courts-circuits et l'augmentation par rapport au TB6560.

Figure 35: TB6600

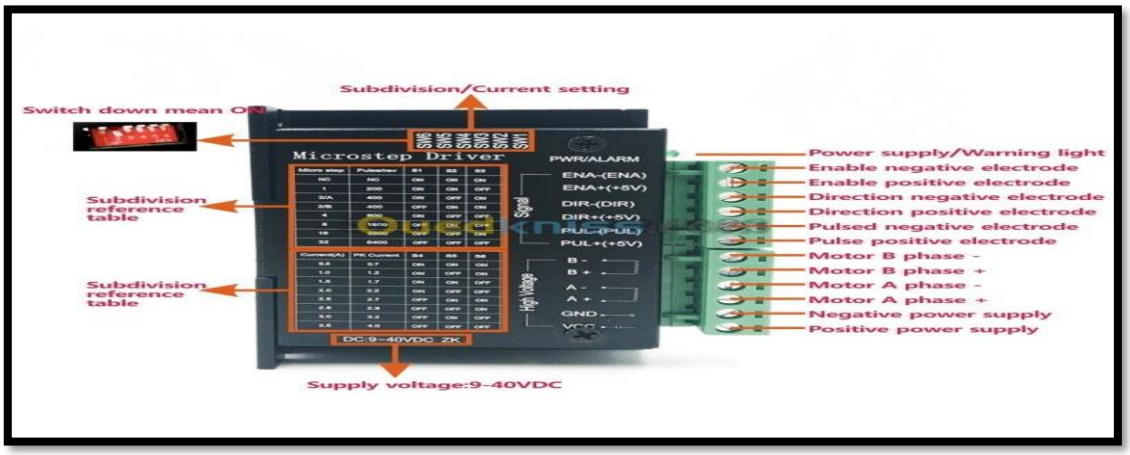
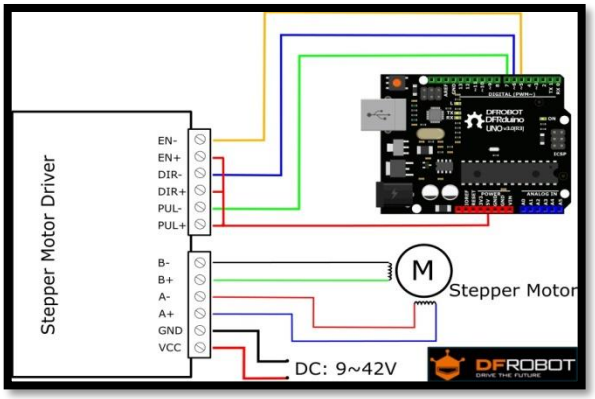


Figure 36: Cablage arduino par moteur



Conclusion :

Dans ce chapitre, nous avons présenté la description de l'automate et la structure générale des API et les différents matériels de siemens, ainsi logiciel de programmation S7 1200 TIA PORTAL V13.

Ensuit nous avons signale procédure à suivre pour la création de nouveau projet sur TIA PORTAL V13 et sa configuration du matériel et communication.

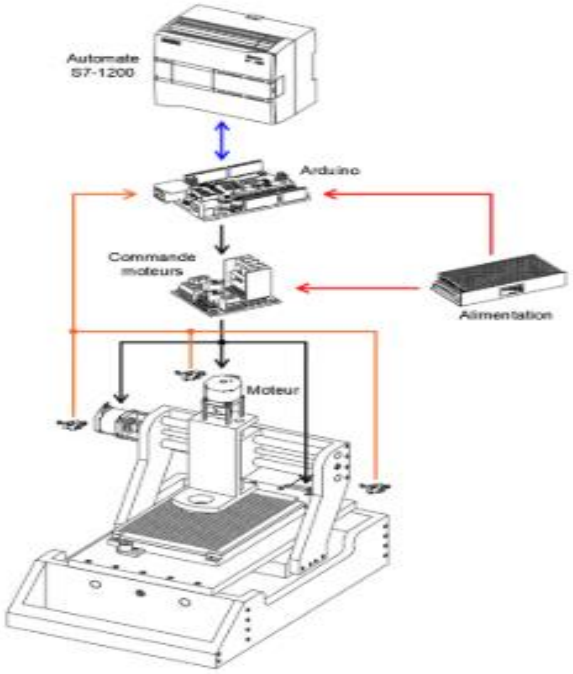
Puis on a abordé une généralité sure l'Arduino et ses caractéristiques, plus le fonctionnement du moteur pas a pas.

CHAPITRE 3

Chapitre 03 : Réalisation Pratique :

Après avoir décrit la partie théorique, nous allons maintenant entamer la partie pratique qui va commander tout le système.

Commençant tout d'abord par une représentation des différents matériels qu'on a utilisés et la programmation effectuer.



1. Partie programmation API :

1.1. motion contrôle :

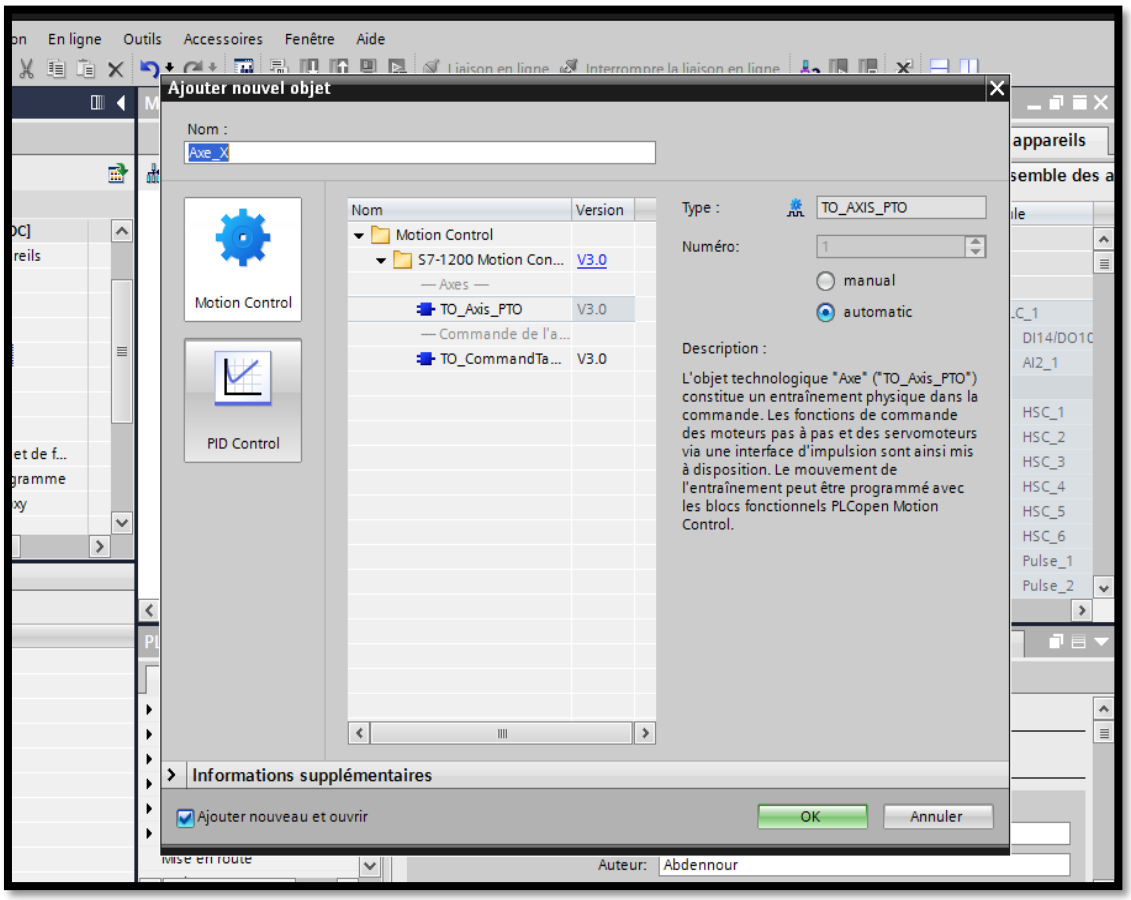
Le TIA Portal nous assiste avec la fonctionnalité "Motion Control" de la CPU S7 -1200 lors de la commande de moteurs pas à pas et de servomoteurs.

Il faut configurer les objets technologiques "Axe de positionnement" dans le TIA Portal. La CPU S7-1200 commande les sorties pour la commande des entraînements à l'aide de ces objets technologiques.

Le paramétrage de l'automate s'effectue dans TIA Portal sous "Objet technologique > Configuration, Paramètres de base > Entraînement/codeur".

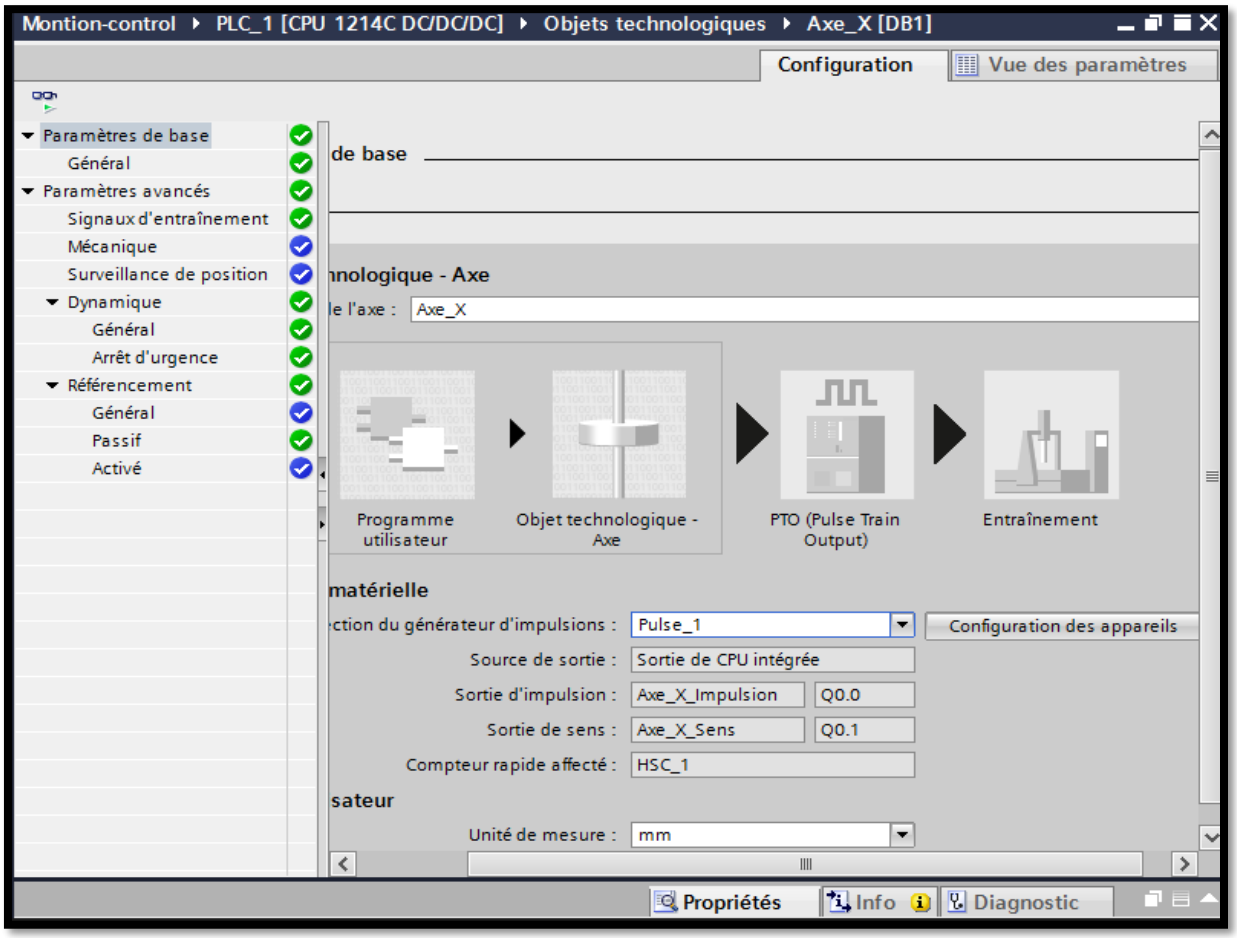
Le paramétrage de l'entraînement et du codeur s'effectue lors de la configuration du matériel Correspondant.

Figure 37: Objet technologies Axes X.



Ensuit, répéter de même manière l'objet technologies pour les axes Y et Z.

Apes ça, On fixe les paramètres de chaque moteur pas à pas. (Nombre d'impulsions par seconde, degré de rotation, la vitesse de rotation et accélération).



1.2. Partie Commande d'un moteur :

Dans le portail TIA, l'entraînement physique est représenté mécanique comprise en tant qu' objet technologique "Axe de positionnement". Pour cela, configurez l'objet technologique "Axe de positionnement" avec les paramètres suivants :

● Choix du PTO à utiliser (Pulse Train Output) .

● Paramètres touchant la mécanique et le rapport de démultiplication de l'entraînement (ou de la machine ou de l'installation).

●Paramètres des limites de position et de la surveillance de positionnement.

●Paramètres de dynamique et de référencement.

La configuration de l'objet technologique "Axe de positionnement" est enregistrée dans l'objet technologique (bloc de données). Ce bloc de données est aussi l'interface entre le programme utilisateur et le firmware de la CPU. Au temps d'exécution du programme utilisateur, les données actuelles de l'axe sont enregistrées dans le bloc de données de l'objet technologique.

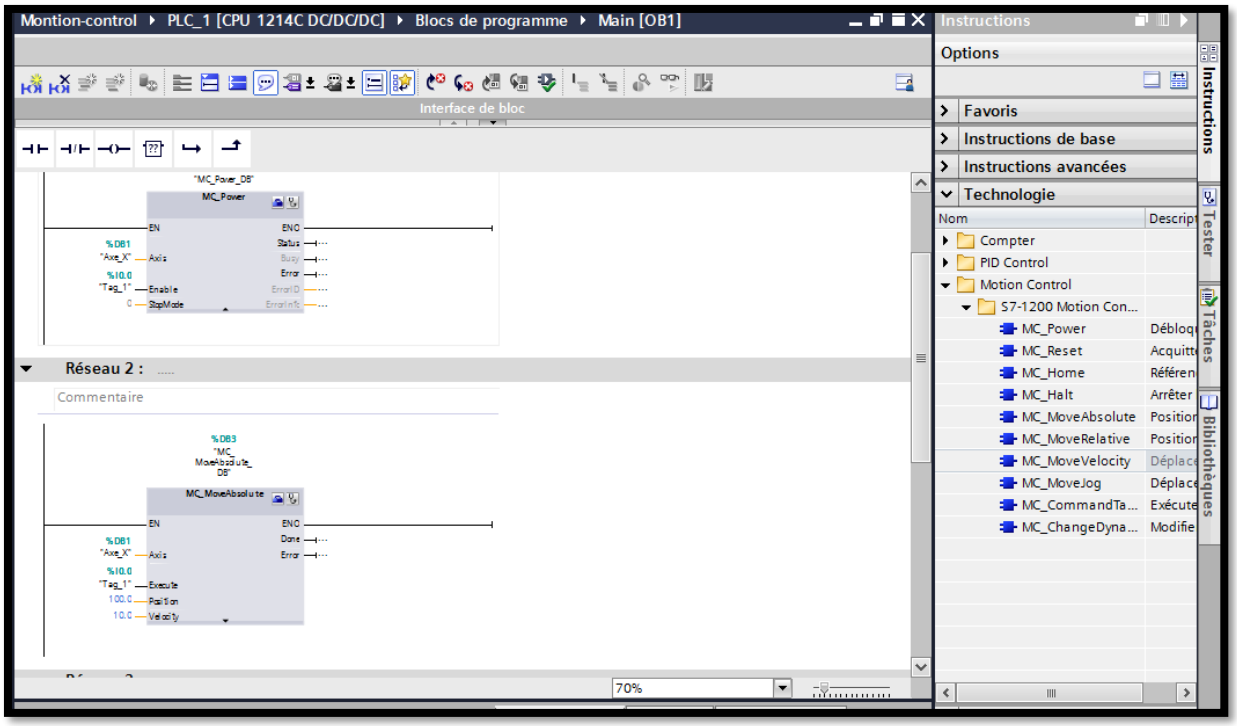
Le programme utilisateur permet de lancer des commandes dans le firmware de la CPU au moyen d'instructions Motion Control. Les commandes suivantes sont possibles pour commander l'axe :

●Libérer et bloquer l'axe

●Positionnement absolu de l'axe

●Positionnement relatif de l'axe

●Déplacer l'axe à une vitesse prédéfinie



En compilant tous les blocs et télécharger dans l'automate.

1.3.Configuration Modbus TCP/IP :

L'instruction « mb client » communique en tant que client Modbus TCP via la connexion PROFINET de la CPU S7-1200. Pour utiliser l'instruction, vous n'avez pas besoin de module matériel supplémentaire. L'instruction « mb client »vous permet d'établir une connexion entre le client et le serveur, d'envoyer des requêtes et de recevoir des réponses et de commander la coupure de la liaison du serveur Modbus TCP.

1.3.1. Paramètre de Modbus TCP/IP :

- **Reg : Requête de communication avec le serveur Modbus TCP :**

Le paramètre REQ est commandé par niveau. Cela signifie que tant que l'entrée est à 1 (REQ=true), l'instruction envoie des requêtes de communication.

- La requête de communication verrouille l'accès au DB d'instance pour les autres clients.
- Les modifications aux paramètres d'entrée ne s'appliquent qu'à partir du moment où il y a une réponse du serveur ou un message d'erreur a été émis.
- Si le paramètre REQ est mis à nouveau à "1" pendant une requête Modbus en cours, aucune autre transmission ne sera exécutée directement après.

- **DISCONNECT :**

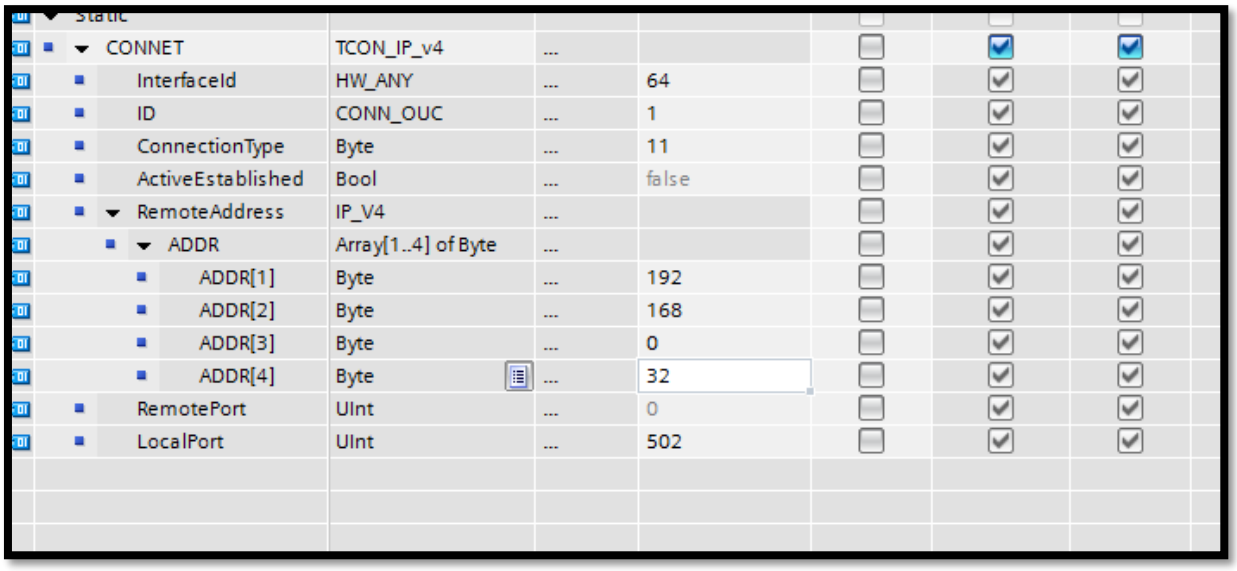
Le paramètre vous permet de commander l'établissement et la coupure de la liaison au serveur Modbus :

- **0:** Établissement de la communication à l'adresse IP et au numéro de port indiqué.
- **1:** Suspendre la connexion de communication. Aucune autre fonction n'est exécutée durant la coupure de la liaison. Après avoir réalisé la coupure de la liaison avec succès, le paramètre STATUS affiche la valeur 7003.

Si le paramètre REQ est mis à "1" lors de l'établissement de la liaison, la requête est émise immédiatement.

- **CONNECT_ID :**

ID univoque pour l'identification de la connexion. A chaque instance des instructions "Mb_client » et « Mb_server » doit être assignée une ID de liaison univoque.



Static						
CONNET	TCON_IP_v4	...		☐	☑	☑
InterfaceId	HW_ANY	...	64	☐	☑	☑
ID	CONN_OUC	...	1	☐	☑	☑
ConnectionType	Byte	...	11	☐	☑	☑
ActiveEstablished	Bool	...	false	☐	☑	☑
RemoteAddress	IP_V4	...		☐	☑	☑
ADDR	Array[1..4] of Byte	...		☐	☑	☑
ADDR[1]	Byte	...	192	☐	☑	☑
ADDR[2]	Byte	...	168	☐	☑	☑
ADDR[3]	Byte	...	0	☐	☑	☑
ADDR[4]	Byte	...	32	☐	☑	☑
RemotePort	UInt	...	0	☐	☑	☑
LocalPort	UInt	...	502	☐	☑	☑

2. Programme Arduino :

Modbus_PLC_V3

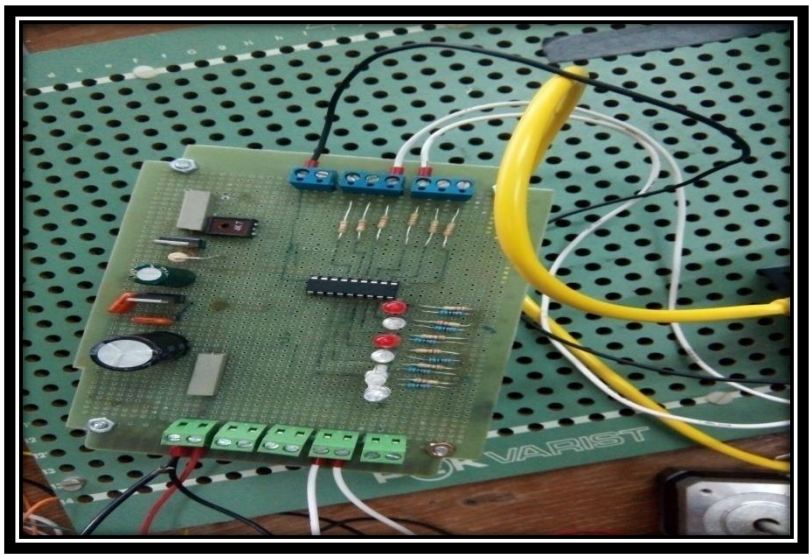
```
#include <SPI.h>
#include <Ethernet.h>
//#include <Modbus.h>
#include <ModbusIP.h>

//Modbus Registers Offsets (0-9999)
const int SXP=22;
const int SXN=23;
const int LSXP=24;
const int LSXN=25;

const int SYP=26;
const int SYN=27;
const int LSYP=28;
const int LSYN=29;

const int SZP=30;
const int SZN=31;
const int LSZP=32;
const int LSZN=33;

unsigned int data_seq=0x0000;
const unsigned data_HREG=100; //40101

//ModbusIP object
ModbusIP mb;
unsigned long ts=0;
unsigned long sp=0;
```

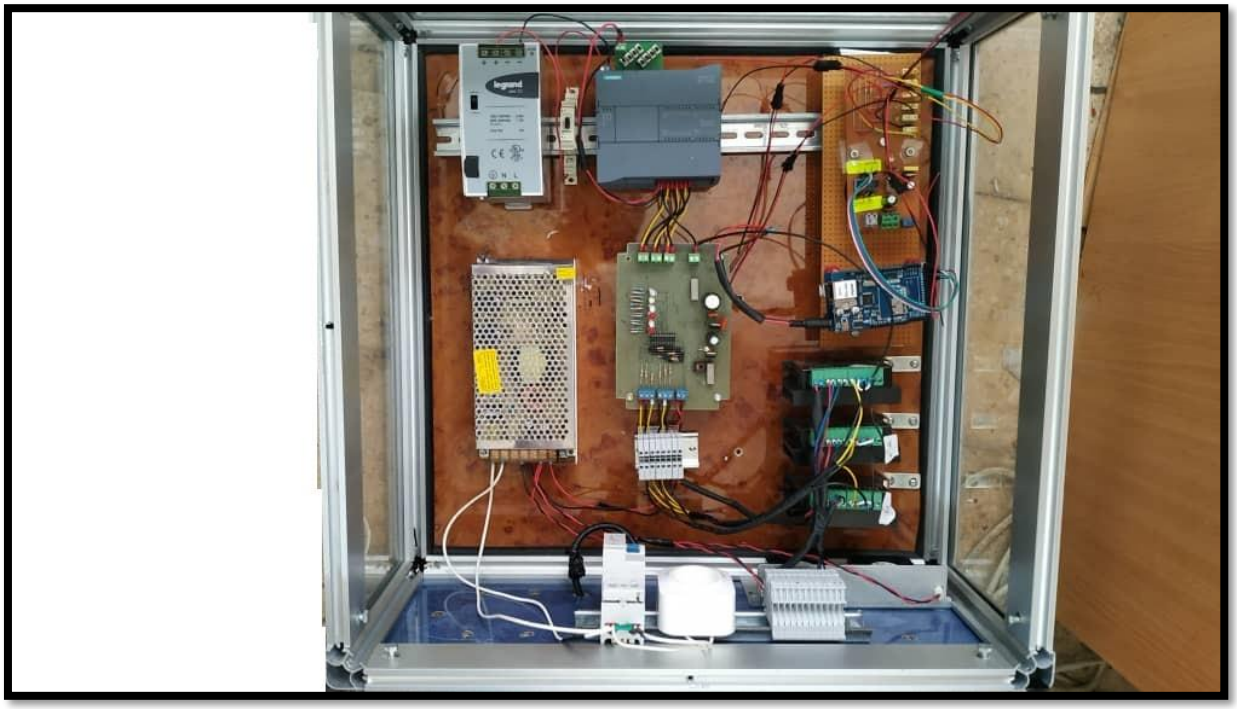
Modbus_PLC_V3

```
//ModbusIP object
ModbusIP mb;
unsigned long ts=0;
unsigned long sp=0;


static byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };//{ 0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02};
static byte ip[]={192,168,1,5};


void setup() {
    Serial.begin(9600);
    pinMode(SXP,INPUT);
    //Config Modbus IP
    //Ethernet.begin(mac,ip);
    mb.config(mac, ip); //Ethernet.begin(mac,ip,gateway,subnet);
    mb.addHreg(data_HREG,0);
   ts=millis();
   sp=millis();
}


void loop() {
   data_seq=0x0000;
  if(digitalRead(SXP)==HIGH){ data_seq|= 0x0001;} else {data_seq &=0xFFFE;}
  if(digitalRead(SXN)==HIGH){ data_seq|= 0x0010;} else {data_seq &=0xFFFD;}
  if(digitalRead(LSXP)==HIGH){ data_seq|= 0x0011;} else {data_seq &=0xFFFE;}
  if(digitalRead(LSXN)==HIGH){ data_seq|= 0x0100;} else {data_seq &=0xFFFE;}
```

```
 if(digitalRead(SYP)==HIGH){ data_seq|= 0x0001;} else {data_seq &=0xFFFE;}
 if(digitalRead(SYN)==HIGH){ data_seq|= 0x0001;} else {data_seq &=0xFFFE;}
 if(digitalRead(LSYP)==HIGH){ data_seq|= 0x0001;} else {data_seq &=0xFFFE;}
 if(digitalRead(LSYN)==HIGH){ data_seq|= 0x0001;} else {data_seq &=0xFFFE;}

 if(digitalRead(SZP)==HIGH){ data_seq|= 0x0001;} else {data_seq &=0xFFFE;}
 if(digitalRead(SZN)==HIGH){ data_seq|= 0x0001;} else {data_seq &=0xFFFE;}
 if(digitalRead(LSZP)==HIGH){ data_seq|= 0x0001;} else {data_seq &=0xFFFE;}
 if(digitalRead(LSZN)==HIGH){ data_seq|= 0x0001;} else {data_seq &=0xFFFE;}


 //data_seq |
  //&0xFFFE
Serial.print("data sequence : ");
Serial.println(data_seq);
//---------------------------------------
 mb.task();
 mb.Hreg(data_HREG,data_seq);

 if (millis()> ts+300){
  mb.Hreg(data_HREG,data_seq);
  ts=millis();
 }
/   if(millis() >sp+200){
/    Serial.print(data_seq, BIN);
/    sp=millis();
```

L'alimentation d'automate avec une tension continue 24V, les sorties sont câblés par des filles vers une carte pour adapter la tension de 24v vers 5v de Arduino.





Armoire électriques.

Conclusion :

Dans ce chapitre on a réalisé une machine outil Puis on a commander cette dernier avec un automate s7 1200 « objet technologie » qui nous permettre de commander les 3 axes de machine en mode slave, car le plateforme Arduino donne des informations par réseaux Modbus TCP/IP

Conclusion General :

L'élaboration de ce travail dans le cadre du projet fin d'étude, nous a permis d'approfondir nos connaissances théoriques en électronique et d'acquérir une bonne expérience au niveau de la réalisation pratique.

Lors de cette manipulation, on a essayé de fournir l'automatisation de commande de trois moteurs pas à pas et applique à une machine utile. Notre projet de fin d'étude consiste à détailler ainsi qu'à réaliser une carte électronique adaptateur de tension de 24 V 5V.

Ce projet m'a donné une meilleure idée sur la complémentaire entre le volet théorique et le volet pratique.

Dans la première partie, nous avons d'abord entamé la première étape qui est basé sur la présentation de MODBUS TCP/IP. Cette partie est essentiellement au fonctionnement de communication entre périphérique industrielles.

Dans la deuxième partie qui consiste sur l'automate et l'Arduino ainsi le matériel utilisé (moteur, driver..) dont nous avons fait une description et une mise en œuvre de la carte Arduino.

Dans la troisième partie on a utilisé l'Arduino comme un maitre dans notre application dont l'automate fonctionne âpres reçoit l'information depuis l'Arduino.

En fait, ce projet a été une source de découverte de plusieurs domaines d'étude tel que l'informatique pour la programmation embarqué sans oublier le savoir-faire dans le domaine automatique et informatique industriel.

En perspective, plusieurs points sont à approfondir et ouvrent la voie à de nouveaux axes de travail :

- Manipule des axes avec dessins et formes précis.
- L'utilisation de la communication sans fils entre le système d'acquisition et le PC.

Table des matières

BIBLIOGRAPHIE

- Cours génie électrique 2eme année de Madame Marie-Claude Vialatte
- guide-du-Modbus-pour –les nuls-PDF.
- http://jacques.boudier.pagesperso-orange.fr/c_online/Informatique_industrielle/automate/cours%20008.htm
- http://www.acromag.com/, Printed in the USA. Data and specifications are subject to change without notice.
- https://loufida.com/les-4-couches-du-modele-tcp-ip/
- [1]https://loufida.com/les-7-couches-du-modele-osi/
- https://www.automation-sense.com/blog/automatisme/qu-est-ce-que-le-modbus-tcp-ip.html
- https://www.frameip.com/osi/
- [1]https://www.frameip.com/tcpip/#2-8211-description-du-modele-tcpip
- https://www.ionos.fr/digitalguide/serveur/know-how/le-modele-osi-reference-pour-les-standards/
- [1]https://www.technologuepro.com/cours-automate-programmable-industriel/Les-automates-programmables-industriels-API.htm
- Network Architectures: Layers of OSI model and TCP/IP model
- www.automation-sense.com COMPRENDRE ET METTRE EN OEUVRE FACILEMENT LE BUS INDUSTRIEL MODBUS.