République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
UNIVERSITE MOULOUD MAMMERI, TIZI –OUZOU

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

Mémoire de Fin d'Etudes

De MASTER PROFESSIONNEL

Spécialité : Electronique industrielle

Filière : Electronique

Thème

Conception et Réalisation d'un Automate Programmable Industriel à base d'une Carte Arduino Méga (ADK).

Réalisé par : dirigé par :

M^r: ALILECHE Abdenour **M^r. HAMICHE**

M^elle: AIT DRIS Ouiza

Mémoire soutenu publiquement le …… devant le jury composé de :

Promoteur : M^r HAMICHE Hamid

Président:

Examinateur :

Année Universitaire : 2017-2018

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
UNIVERSITE MOULOUD MAMMERI, TIZI –OUZOU

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

Mémoire de Fin d'Etudes

De MASTER PROFESSIONNEL

Spécialité : Electronique industrielle

Filière : Electronique

Thème

Conception et Réalisation d'un Automate Programmable Industriel à base d'une Carte Arduino Mega ADK.

Réalisé par : dirigé par :

M$^{r:}$ ALILECHE Abdenour **M^{r}. HAMICHE**

M^{elle}: AIT DRIS Ouiza

Mémoire soutenu publiquement le …… devant le jury composé de :

Promoteur : M^{r} HAMICHE Hamid

Président:

Examinateur :

Année Universitaire : 2017-2018

Remerciement

Avant tout, nous remercions le bon dieu le tout puissant qui nous a donné la volonté le courage et la patience durant notre parcours d'étude afin de réaliser ce modeste travail.

Nos vifs remerciement s'adresse à notre encadreur ***Mr.HAMICHE Hamid*** *qui a été toujours présent pour nous orienter sur le bon chemin, et a qui nous somme reconnaissons pour tout les moyens mis à notre disposition pour l'élaboration de ce mémoire.*

Nous remercions également ***Mme RAHMANI*** *ingénieur et responsable du laboratoire maquette de nous avoir consacré de son temps afin de faciliter l'avancement de notre de réalisation.*

Nous souhaitons également exprimer notre profonde gratitude et remerciements aux membres de jury qui nous font l'honneur de juger notre travail. Nous souhaitons être à la hauteur de leurs attentes.

Nous tenons aussi à remercier tous les enseignants qui ont veillé au bon déroulement de notre formation tout le long de notre cursus.

Un grand merci a toute personne qui a contribué de prés ou de loin pour réaliser ce modeste travail.

Dédicace

Il me tient à cœur de dédier ce modeste travail à ceux qui m'ont donné la vie source de mon courage et mon inspiration.

A mon père

A ma mère

A mes frères Kamel et Sid ali

A la mémoire de ma grand-mère que dieu l'accueille dans son vaste paradis

A toute la famille ALILECHE

A mon binôme Ouiza

A toutes les personnes que je porte dans mon cœur et qui se reconnaitront car elles en font autant.

Je vous dédie ce travail en guise de reconnaissance car vous m'êtes si chers que je ne peux que vous offrir ce que j'ai appris de mieux dans ma vie

ABDENOUR…

Dédicace

A la mémoire de ma grande mère

A mes chères parents qui m'ont éclairé le chemin et m'ont encouragé et soutenu tout le temps

A mes grands parents

A mes sœurs (Amel, Sinta et Amira).

A tout mes oncles et toutes mes tantes et leurs enfants.

A tous mes amis surtout (Sonia, Anissa et Samira) et leurs familles.

A mon binôme Abdenour

Ouiza…

Liste des abréviations

[A]

ADK: Androïde développement kit.

API: Automate programmable industriel.

AREF: analogue reference.

AVR: désigne le cœur du processeur.

AVRC: microcontrôleurs utilisant C comme langage.

[C]

CDT: C&C++ developement tool.

CPU: (Central Processing Unit), unité centrale de traitement.

[D]

DIY: (Do it yourself), fait maison.

DFU: device firmware upgrade.

[F]

FBD: fonction block diagram**.**

FTDI: circuit intégré qui adapte le signal entre le port USB et le port série de l'ATmega.

[G]

GPS: (global positioning system) assistant de navigation personnel.

[I]

ICSP: in circuit serial programming.

IDE: integrated development environment.

IL: (instruction List) liste d'instruction**.**

IOREF: input/output reference.

[L]

LD: ladder diagram.

[O]

OSX: système d'exploitation développé par Appel**.**

[P]

PCB: (printed circuit board) circuit imprimé.

PLC: programmable logique contrôleur.

PWM: (Pulse Width Modulation), modulation de largeur d'impulsions.

[R]

RESET: remettre à zéro.

[S]

SCADA: (Supervisory Control And Data Acquisition) système d'acquisition et de contrôle de données.

SCL: Serial clock.

SDA: Serial data.

SFC: (séquentielle fonction chart), langage graphique de programmation.

SPDT: single pole, double throw.

ST: (structured text) littéral structuré.

[U]

UART: (universal Asynchronous Receiver Transmitter), émetteur-récepteur asynchrone universel.

USB: (Universal Serial Bus), bus universel en série.

Liste des figures

Figure I.1 : Automate compact d'Allen-Bradley .....-2-

Figure I.2 : Automate modulaire d'Allen-Bradley.....-3-

Figure I.3 : Structure d'un API.....-3-

Figure I.4 : Schéma d'un photocoupleur.....-6-

Figure I.5 : Niveaux des entrées.....-7-

Figure I.6 : Niveaux des sorties.....-7-

Figure I.7 : Exemple d'une carte de sortie à relais typique d'un API.....-8-

Figure I.8 : Symboles usuels en langage LD.....-12-

Figure II.1 : Carte Arduino Méga (ADK) .....-13-

Figure II.2 : Schéma des broches de l'Arduino Méga(ADK) .....-17-

Figure II.3 : Les différentes parties que comporte un programme Arduino .....-18-

Figure II.4 : L'interface de l'IDE d'Arduino .....-23-

Figure II.5 : les différentes parties de l'interface de l'IDE d'Arduino .....-24-

Figure II.6 : La barre d'outils de l'IDE d'Arduino .....-25-

Figure II.7 : La fenêtre de Slober .....-26-

Figure II.8 : Quelques blocs de codage .....-27-

Figure II.9: Interface hote USB .....-28-

Figure II.10: Shield Ethernet Arduino .....-28-

Figure II.11: Module XBee wifi .....-29-

Figure II.12: Shield écran TFT tactil .....-29-

Figure III.1: Schéma synoptique du système .....-31-

Figure III.2: Schéma de conception de la carte d'alimentation.....-33-

Figure III.3: Conception du module de l'Arduino .....-34-

Figure III.4: Schéma conçu pour le module d'entrée .....-35-

Figure III.5: Schéma conçu pour le module de sortie .....-36-

Figure III.6: Schéma conçu pour le module à relais .....-37-

Figure III.7 : Simulation du module d'entrée sur Proteus .....-39-
Figure III.8 : Simulation du module de sortie à transistor de puissance sur Proteus .....-40-
Figure III.9 : Simulation du module de sortie à relais sur Proteus .....-41-
Figure IV.1 : Module d'entrée .....-46-
Figure IV.2 : Module de sortie .....-46-
Figure IV.3 : Module à relais .....-47-
Figure IV.4 : Typon module CPU .....-47-
Figure IV.5 : Typon module entré .....-48-
Figure IV.6 : Typon module sortie à transistor .....-48-
Figure IV.7 : Typon module sortie à relais .....-49-
Figure IV.8 : Schéma d'implantation de la CPU .....-50-
Figure IV.9 : Schéma d'implantation du module d'entrée .....-50-
Figure IV.10 : Schéma d'implantation du module de sortie .....-51-
Figure IV.11 : La carte du module à relais .....-51-
Figure IV.12 : La carte de la CPU .....-52-
Figure IV.13 : La carte d'alimentation. .....-53-
Figure IV.14 : Photo du module d'entrée .....-54-
Figure IV.15 : Photo du module de sortie .....-55-
Figure IV.16 : Photo du module à relais .....-56-

Liste des abréviations

Liste des figures

SOMMAIRE

Sommaire

Introduction générale .....- 1 -

Chapitre I : Généralités sur les Automates Programmables Industriels

I.1 Introduction ..... - 2 -
I.2 Définition ..... - 2 -
I.3 Structure d'un API ..... - 3 -
I.4 Architecture interne d'un API ..... - 4 -
I.4.1 L'alimentation ..... - 4 -
I.4.2 La CPU ..... - 4 -
I.4.3 Les BUS ..... - 4 -
I.4.4 La mémoire [1] ..... - 5 -
I.4.5 Unité d'entrées /Sorties ..... - 5 -
I.5 Objectifs à atteindre : ..... - 8 -
I.6 Choix d'un API ..... - 9 -
I.7 Fonctionnement et dysfonctionnement d'un API ..... - 9 -
I.8 Caractéristique d'un automate programmable ..... - 10 -
I.9 Les avantages qu'apportent les API aux systèmes automatisés. ..... - 10 -
I.10 Les langages de programmation des automates programmables ..... - 11 -
I.10.1 Les langages littéraux ..... - 11 -
I.10.2 Les langages graphiques ..... - 11 -
I.11 Conclusion ..... - 12 -

Chapitre II : Etude de la carte arduino-Méga (ADK)

II.1 Introduction .....- 13 -
II.2 Description de la carte Arduino Méga (ADK) .....- 13 -
II.3 Spécifications techniques de la carte Arduino Méga (ADK) [8] .....- 14 -
II.4 Alimentation de la carte Arduino Méga (ADK) .....- 14 -
II.5 Avantages de la carte Arduino Méga (ADK) .....- 15 -
II.6 Répartition des broches de l'Arduino Méga (ADK) .....- 17 -
II.7 Généralités sur le langage Arduino .....- 18 -
II.7.1 La structure d'un programme [6], [8] ..... - 18 -

II.7.2 Coloration syntaxique ..... - 19 -
II.7.3 La syntaxe du langage Arduino ..... - 19 -
II.7.3.1 La ponctuation [6], [10] ..... - 19 -
II.7.3.2 Les variables ..... - 20 -
II.7.3.3 Les fonctions [10] ..... - 20 -
II.7.3.4 Les structures de contrôle [6], [8] ..... - 21 -
II.8 Logiciel de programmation l'IDE d'Arduino ..... - 23 -
II.8.1 Généralités sur L'IDE d'Arduino ..... - 23 -
II.8.1.1 Présentation du logiciel [6], [10] ..... - 24 -
II.8.1.2 Les différents boutons de l'interface de l'IDE d'Arduino ..... - 25 -
II.9 Autres logiciels de programmation d'Arduino : ..... - 26 -
II.9.1 Les logiciels de programmation à langages littéraux : ..... - 26 -
II.9.2 Les logiciels de programmation à langages graphiques : ..... - 27 -
II.10 Connectivité et communication ..... - 27 -
II.10 Conclusion ..... - 30 -

Chapitre III : Conception et simulation-

III.1 Introduction ..... - 31 -
III.2 Schéma synoptique du système ..... - 31 -
III.3 Principe de fonctionnement du système ..... - 32 -
III.4 Etude et conception des différents modules de l'API ..... - 32 -
III.4.1 Alimentation stabilisée ..... - 33 -
III.4.2 La CPU ..... - 34 -
III.4.3 Module d'entrée ..... - 35 -
III.4.4 Module de sortie à transistor de puissance ..... - 36 -
III.4.5 Module de sortie à relais ..... - 37 -
III.5 Simulation sur Proteus ..... - 38 -
III.5.1 Présentation du logiciel proteus [11] ..... - 38 -
III.5.1.1 ISIS ..... - 38 -
III.5.1.2 ARES ..... - 38 -
III.5.2 Simulation du module d'entrée ..... - 39 -
III.5.3 Simulation du module de sortie à transistor de puissance ..... - 40 -
III.5.4 Simulation du module de sortie à relais ..... - 41 -

III.6 Schémas électriques des différents modules ..... - 41 -
III.7 Conclusion ..... - 45 -

Chapitre IV : Test sur lab d'essai et réalisation

IV.1 Introduction ..... - 46 -
IV.2 Test des différents circuits sur lab d'essai ..... - 46 -
IV.3 Fabrication des circuits imprimés ..... - 47 -
IV.3.1 Les étapes de la fabrication des plaques ..... - 49 -
IV.3.2 Implantation des composants ..... - 50 -
IV.4 Concrétisation de l'API ..... - 52 -
IV.4.1 Carte de la CPU ..... - 52 -
La photo de la carte de la CPU est donnée ci dessous : ..... - 52 -
IV.4.2 Carte d'alimentation ..... - 53 -
La photo de la carte d'alimentation est donnée ci dessous : ..... - 53 -
IV.4.3 Module d'entrée ..... - 54 -
La photo de la carte d'entrée est donnée ci dessous : ..... - 54 -
IV.4.4 Module de sortie ..... - 55 -
La photo de la carte de sortie est donnée ci dessous : ..... - 55 -
IV.4.5 Module de sortie à relais ..... - 56 -
La photo de la carte du module à relais est donnée ci dessous : ..... - 56 -
IV.5 Tableau comparatif ..... - 57 -
IV.6 Conclusion ..... - 57 -

Conclusion générale ..... - 58 -

Introduction générale

Introduction générale

Dans le domaine industriel où la concurrence est rude, les entreprises sont confrontées à une exigence d'améliorer la qualité et la quantité de leurs produits en minimisant les prix et les dépenses. Pour cela, elles doivent disposer de chaînes de production souples et performantes, avec moins d'intervention humaine. C'est ainsi que l'idée des procédés automatisés dans l'industrie a immergé [1], [2]. Pour réaliser cette tâche, un Automate Programmable Industriel dit API a été introduit. C'est une machine électronique programmable, capable de piloter un système qui modifie dans le sens souhaité un quelconque processus technologique [2].

Notre travail consiste à réaliser un outil didactique utilisant comme partie de commande un système à l'aide d'une carte Arduino Méga ADK. On parle alors d'automate programmable conçu à base de la logique programmée. Ce dernier sera capable de réaliser la plupart des applications qui lui seront demandées par simple modification du logiciel en gardant la même structure matérielle.

L'automate que nous réaliserons répondra globalement aux mêmes exigences attendues d'un automate classique, notamment la possibilité d'être programmé et reprogrammé directement via un port parallèle d'un micro-ordinateur.

Le présent mémoire est organisé en quatre chapitres et une conclusion et quelques perspectives.

Dans le premier chapitre, nous allons donner des généralités sur les Automates Programmables industriels.

Le deuxième chapitre sera consacré à l'étude de la carte Arduino Méga ADK autour duquel est architecturé notre système.

Dans le troisième chapitre, nous allons présenter la phase conception et simulation des différents module de l'Automate conçu à base de la carte Arduino-Méga ADK.

Dans le dernier chapitre, nous allons présenter la partie réalisation, des différents modules de notre automate.

Nous clôturons notre mémoire par une conclusion générale et quelques perspectives.

Chapitre I

Généralités sur les Automates Programmables Industriels

I.1 Introduction

L'automatisme prend un élan considérable dans le domaine de l'industrie. Grâce à l'automatisation des processus industriels de plus en plus complexe [1], nous assistons à la naissance de produits de plus en plus performants et précis.

L'objectif de ce premier chapitre est de donner un aperçu sur la notion de l'automatisme ainsi que les automates programmables industriels (API).

I.2 Définition

Un API est une forme particulière de contrôleur à microprocesseur qui utilise une mémoire programmable pour stocker les instructions et qui implémente différentes fonctions, qu'elles soient logiques, de séquencement, de temporisation, de comptage ou arithmétiques, pour commander les machines et les processus. Il est conçu pour être exploité par des ingénieurs, dont les connaissances en informatique et langages de programmation peuvent être limitées [2].

Les automates peuvent aller de petits blocs de construction (compact) avec des dizaines d'entrées et sorties (E/S), dans un boîtier intégré au processeur, à de grands dispositifs modulaires montés en rack avec un nombre de milliers d'E/S, et qui sont souvent en réseau avec d'autres systèmes PLC et SCADA. (Voir figures I.1 et 2).

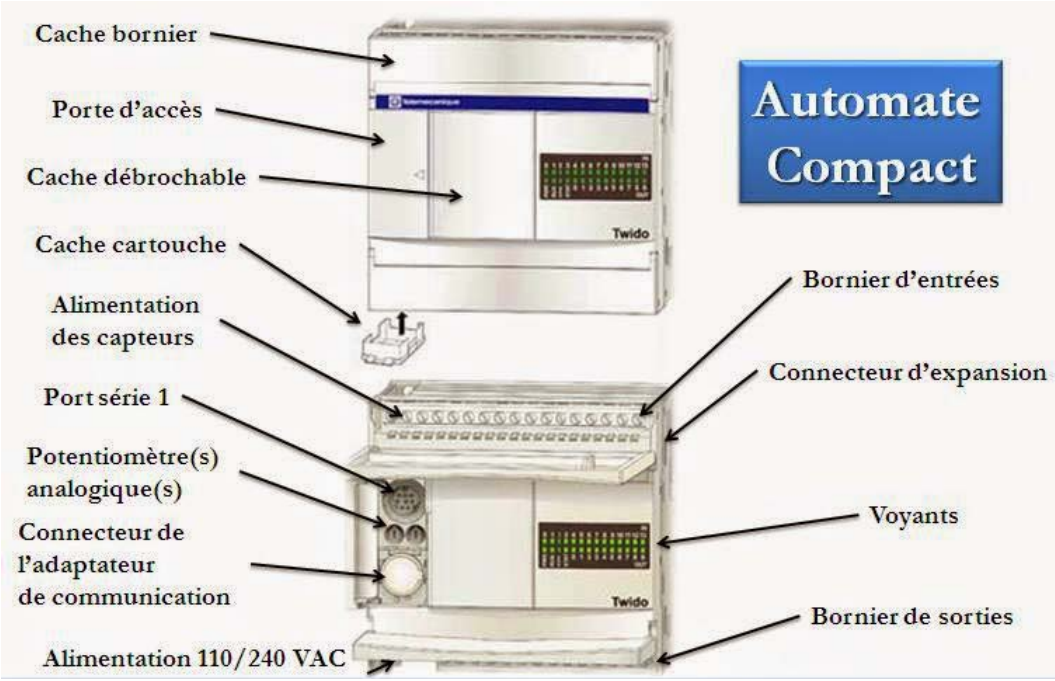


Figure I.1 : Automate compact d'Allen-Bradley

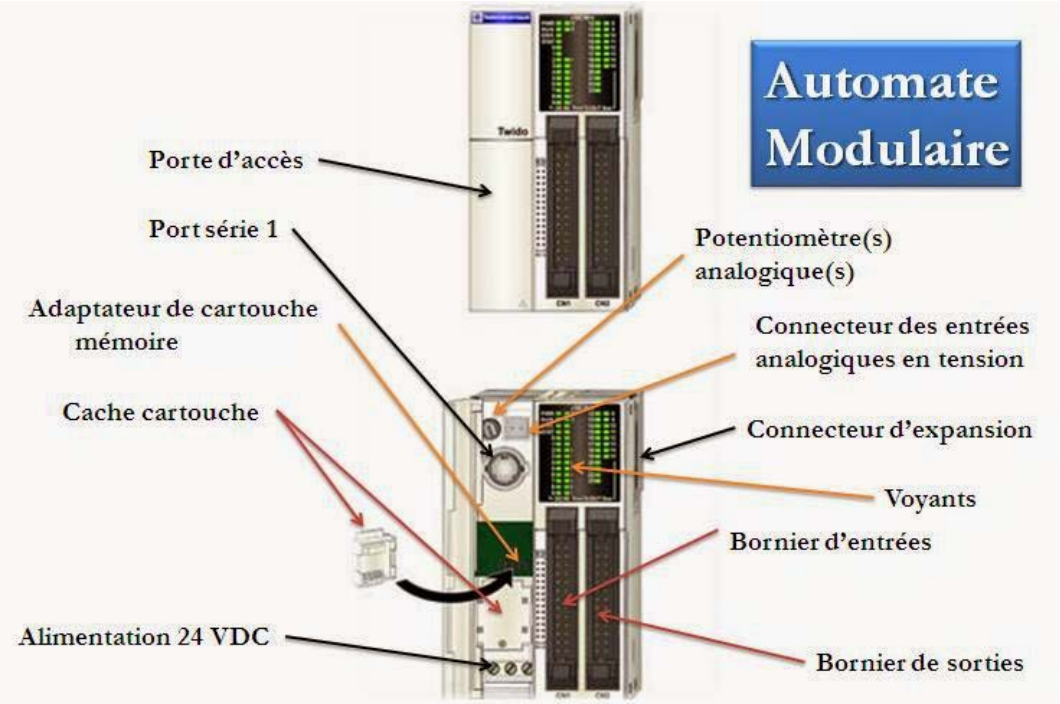


Figure I.2 : Automate modulaire d'Allen-Bradley

I.3 Structure d'un API

De manière générale, un API est structuré autour de plusieurs éléments de base qui sont l'unité de traitement, la mémoire, l'unité d'alimentation, les interfaces d'entrées-sorties l'interface de communication et le périphérique de programmation. Ceci est illustré par la figure ci-dessous [2].

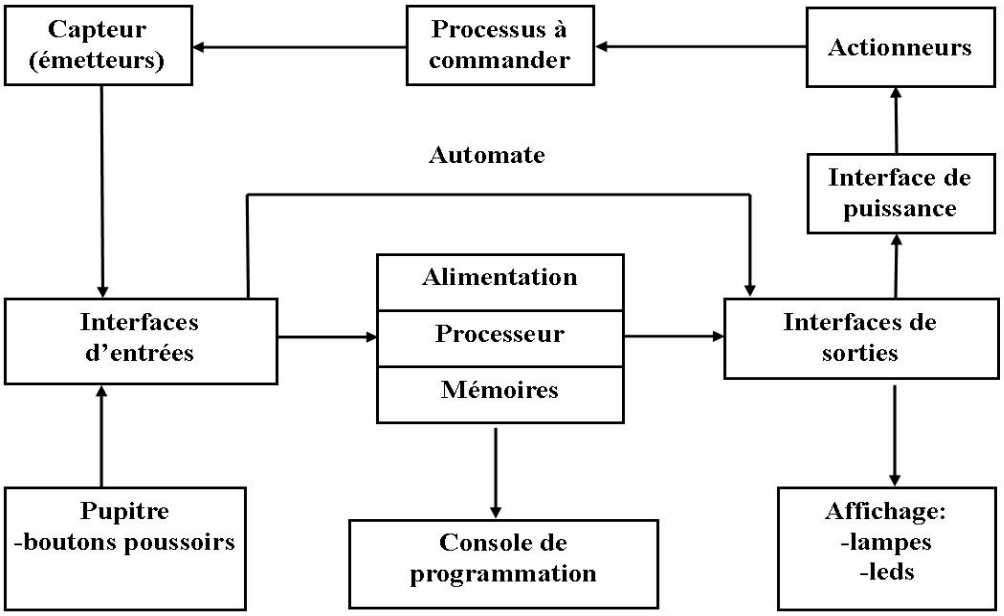


Figure I.3 : Structure d'un API

I.4 Architecture interne d'un API

L'automate programmable reçoit les informations relatives à l'état du système et puis commande les préactionneurs suivant le programme inscrit dans sa mémoire, il est constitué d'une unité centrale de traitement qui comprend le microprocesseur, la mémoire et les entrées-sorties du système [1].

I.4.1 L'alimentation

Elle est élaborée à partir d'un réseau 220V en courant alternatif, ou d'une source 24V en courant continu, elle fournit les tensions internes distribuées aux modules de l'automate [1].

Afin d'assurer le niveau de sûreté requis, celle-ci comporte des dispositifs de détection de baisse ou de coupure de la tension réseau, et de surveillance des tensions internes. En cas de défaut, ces dispositifs peuvent lancer une procédure prioritaire de sauvegarde [2].

I.4.2 La CPU

La CPU (l'unité centrale de traitement), contrôle et exécute toutes les opérations de l'automate. Il est muni d'une horloge dont la fréquence se situe généralement entre 1 et 8 mhz [1].

Cette fréquence détermine la rapidité de fonctionnement de l'API et sert de base au minutage et à la synchronisation pour tous les éléments du système au sein de l'API. Toutes les informations sont transmises au moyen de signaux numériques. Les chemins par lesquels passent ces signaux sont appelés bus.

L'architecture interne de la CPU dépend du microprocesseur employé [3]. En général, les CPU sont constitués des éléments suivants :

- L'unité arithmétique et logique.
- La mémoire.
- Une unité de commande.

I.4.3 Les BUS

Au sens physique, un bus n'est qu'un ensemble de conducteurs sur lesquels circulent les signaux électriques. Il peut s'agir de pistes sur un circuit imprimé ou de fils dans un câble plat qui réalise la liaison entre les différents éléments de l'automate [1]. Dans un API, il se présente sous forme d'un circuit imprimé situé à l'intérieur de l'API et supporte des connecteurs sur lesquels viennent se brancher les différents modules : Processeur, extension mémoire, interfaces et coupleurs.

Le bus est organisé en plusieurs sous ensembles destinés chacun à véhiculer un type d'informations bien défini :

- Bus de données : Il transporte les données utilisées dans les traitements effectués par la CPU. Il peut donc réaliser des opérations entre des nombres et fourni des résultats sous forme de valeurs sur un nombre de bits bien défini.
- Bus d'adresses : Celui-ci transporte les adresses des emplacements mémoire. Pour que chaque mot puisse être localisé en mémoire, chaque emplacement possède une adresse unique que la CPU utilise pour accéder aux données enregistrées à cet emplacement que ce soit pour les lire ou pour y écrire.
- Bus de contrôle : Il transporte les signaux utilisés par la CPU pour le contrôle. Il sert, par exemple, à informer les dispositifs mémoires s'ils vont recevoir des données à partir d'une entrée ou s'ils vont envoyer des données.
- Bus système : sert aux communications entre les ports d'entrées/sorties et l'unité d'entrées/sorties.

I.4.4 La mémoire [1]

L'API accède aux données à traiter et aux instructions, c'est-à- dire au programme, qui lui explique comment traiter ces données. Ces informations sont stockées dans la mémoire de l'API qui est composée :

- d'une mémoire morte (ROM, Read Only Memory) du système qui représente un espace de stockage permanent pour le système d'exploitation et les données figées utilisées par la CPU.
- d'une mémoire vive (RAM, Random Access Memory), utilisée pour le programme de l'utilisateur, et également pour les données. C'est à ce niveau que sont stockées les informations sur l'état des entrées et des sorties, ainsi que les valeurs des temporisateurs, des compteurs et des autres dispositifs internes. Une partie de cet espace est réservée aux adresses d'entrées et de sorties, ainsi qu'à leurs états.
- d'une mémoire morte reprogrammable (EPROM, Erasable and Programmable Read Only Memory). Elle est parfois employée pour stocker de manière permanente les programmes.

I.4.5 Unité d'entrées /Sorties

L'unité d'entrées /sorties apporte l'interface entre le système et le monde extérieur. Au travers de canaux d'entrées/ sorties, elle permet d'établir des connexions avec des dispositifs d'entrée, comme des capteurs, et des dispositifs de sortie, comme des moteurs et des solénoïdes (bobine) [2]. C'est également par l'intermédiaire de cette unité que se fait la saisie des programmes depuis un terminal. Chaque point d'entrée/sortie dispose d'une adresse unique que la CPU peut utiliser.

Puisque les canaux d'entrées/sorties mettent en place les fonctions d'isolation et de traitement des signaux, il est possible de connecter directement des capteurs et des actionneurs aux canaux, sans passer par un autre circuit d'interface.

L'isolation électrique avec le monde extérieur est généralement réalisée par des photocoupleurs (également appelés optocoupleurs). (Voir figure I.4). Lorsque la diode électroluminescente est traversée par une impulsion numérique, elle produit un rayonnement infrarouge [1].

Ce rayonnement est détecté par le phototransistor, qui fait naitre une tension dans son circuit. L'espace qui sépare la LED et le phototransistor crée une isolation électrique, mais une impulsion numérique dans le premier circuit permet néanmoins de produire une impulsion numérique dans le second circuit.

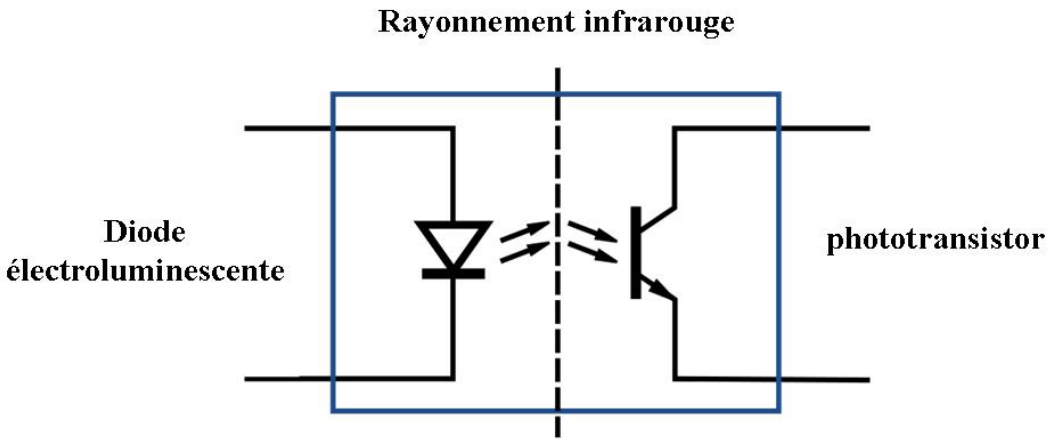


Figure I.4 : Schéma d'un photocoupleur

Pour être compatible avec le microprocesseur de l'API, le signal numérique utilise généralement une tension continue de 5 V. Toutefois, le traitement du signal réalisé au niveau du canal d'entrée, avec l'isolation, permet de manipuler une grande diversité de signaux d'entrée. Un API élaboré peut ainsi accéder à des entrées dont les signaux numériques/discrets (c'est-à-dire Tout Ou Rien) utilisent des tensions de 5 V, 24 V, 110 V et 240 V [1], ceci est illustré par la figure I.5. Un API de base sera généralement en mesure d'utiliser une seule forme d'entrée, par exemple des signaux de 24 V.

Une unité d'entrée-sortie peut produire des sorties numériques avec un niveau de 5 V. toutefois, après traitement du signal par des relais, des transistors ou des triacs. Il est possible d'obtenir en sortie un signal de commutation 24 V et 100 mA, un courant continu de 110 V ou un courant alternatif de 240 V et 1 A ou 240 V et 2 A [1]. Avec un API de base, toutes les sorties seront probablement d'un seul type, par exemple un courant alternatif de 240 V et 1 A,

ceci est illustré par la figure I.6. En revanche, un API modulaire pourra disposer de différentes sorties en installant les modules correspondants.

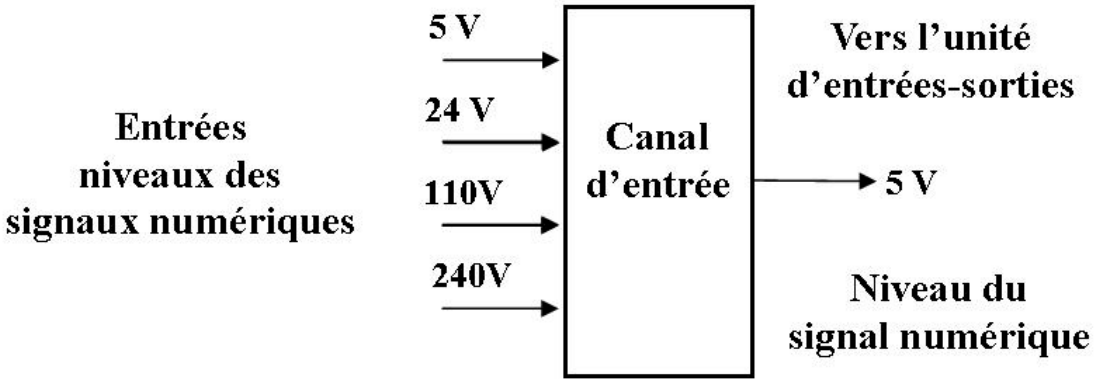


Figure I.5: niveaux des entrées [1]

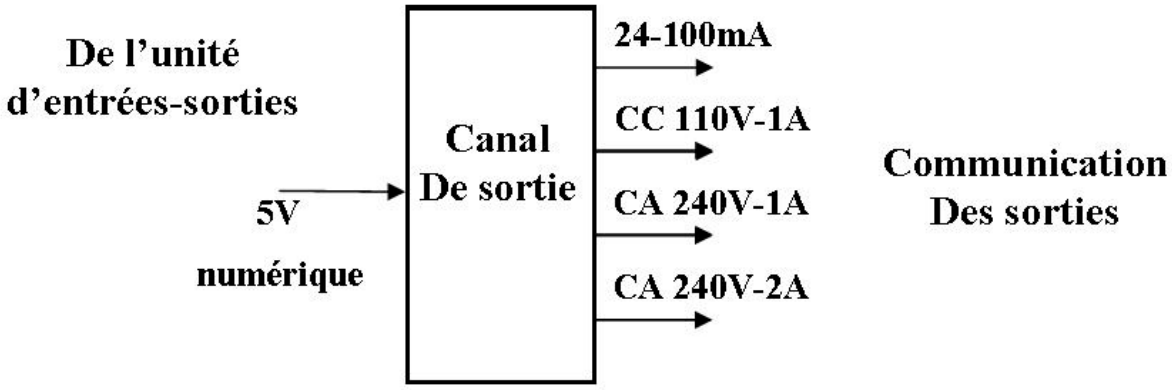


Figure I.6 : Niveaux des sorties [1]

Les sorties sont de type relais, transistor ou triac :

- Avec le type relais, le signal issu d'une sortie de l'API est utilisé pour déclencher un relais et peut activer des courants de quelques ampères dans le circuit externe (Figure I.7). Grâce au relais, non seulement des courants faibles peuvent commuter des courants forts, mais l'isolation entre l'API et le circuit externe est également assurée. En revanche, les relais sont relativement lents. Les sorties à relais sont adaptées à la commutation de courants alternatifs et continus. Ils peuvent supporter des surtensions transitoires et des courants de choc élevés.

- Avec le type transistor, la sortie se fonde sur un transistor pour commuter le courant dans le circuit externe. L'opération de commutation est donc extrêmement rapide. Toutefois, les sorties de ce type sont adaptées uniquement à la commutation d'un courant continu et sont détruites par les surintensités et les tensions inverses élevées. Pour leur protection, il faut utiliser un fusible ou un système électronique. Les photocoupleurs sont employés à des fins d'isolation.

- Avec le type triac, et des photocoupleurs pour l'isolation, les sorties peuvent servir à contrôler les charges externes connectées à une alimentation en courant alternatif. Elles prennent en charge uniquement les courants alternatifs et sont très facilement détruites par les surintensités. De manière générale, les sorties de ce type sont toujours protégées par des fusibles.

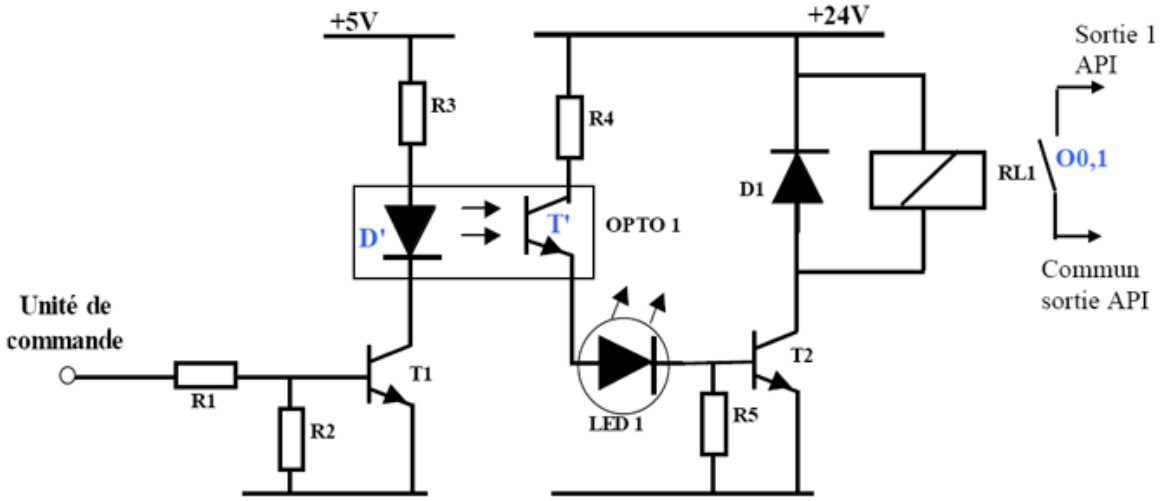


Figure I.7: Exemple d'une carte de sortie à relais typique d'un API [5]

I.5 Objectifs à atteindre :

La vocation d'un automate programmable est d'être un outil proche de l'utilisateur, il doit donc accomplir des fonctions telles que [3] :

- Le dialogue avec l'opérateur.
- Le traitement des informations.
- La liaison avec la partie opérative.

I.6 Choix d'un API

Les paramètres essentiels à prendre en compte pour choisir un automate sont [3] :

- Le nombre d'entrées/sortie et leur nature.
- Le choix de l'unité centrale (microcontrôleur) et la taille de la mémoire utilisateur.
- Temps de traitement (scrutation).
- Capacité de la mémoire.
- Nombre de compteurs.
- Nombre de temporisateurs

I.7 Fonctionnement et dysfonctionnement d'un API

Les programmes des API sont traités selon un cycle précis, le plus souvent :

- Diagnostic (autotest).
- Acquisition de toutes les entrées (recopie dans une mémoire image).
- Traitement du programme.
- Mise à jour des sorties.

L'automate programmable reçoit des données par ses entrées, celles-ci sont ensuite traitées par un programme défini, le résultat obtenu étant délivré par ses sorties. La lecture des capteurs et commande des actionneurs sont réalisées par scrutation. Ce cycle de traitement est toujours le même, quel que soit le programme, néanmoins le temps d'un cycle d'API varie selon la taille du programme, la complexité des calculs, le nombre d'entrées/sorties [1].

La puissance de l'API et les besoins du procédé piloté varient d'une à quelques dizaines de millisecondes.

C'est l'unité centrale qui gère l'automate programmable : Elle reçoit, mémorise et traite les données entrantes et détermine l'état des données sortantes en fonction du programme établi [2].

Un dysfonctionnement peut avoir pour origine :

- Un composant mécanique défaillant (pré-actionneur, actionneur, détecteur,…).
- Un câblage incorrect ou défaillant (entrées, sorties).
- Un composant électrique ou électronique défectueux (interface d'entrée ou de sortie).
- Une erreur de programmation (affectation d'entrées-sorties, ou d'écriture).
- Un système non initialisé (étape, conditions initiales…).

I.8 Caractéristique d'un automate programmable

L'automate programmable se substitue à un processus automatique complexe à base de différentes technologies.

- Il se raccorde directement aux capteurs et préactionneurs via les interfaces.
- Il est conçu pour fonctionner dans des ambiances industrielles qui peuvent être très sévères.
- Il est apte à gérer un grand nombre d'entrées /sorties.
- Il peut contenir des modules complémentaires (analogique, communication,..).
- Il est programmable par plusieurs langages.

I.9 Les avantages qu'apportent les API aux systèmes automatisés.

L'utilisation de l'automate programmable dans le domaine industriel présente l'avantage d'être composé d'éléments particulièrement robustes et possède d'énormes capacités d'exploitation. L'automatisation permet d'apporter des éléments supplémentaires à la valeur ajoutée par le système, ses éléments sont exprimés en termes d'objectifs par [4] :

- L'accroissement de la productivité du système sous forme :
- d'une meilleure rentabilité.
- d'une meilleure compétitivité.
- L'amélioration de la flexibilité de production.
- L'amélioration de la qualité du produit.
- L'adaptation a des contextes particuliers.
- L'adaptation a des environnements hostiles pour l'homme.
- L'utilisation dans les taches physiques ou intellectuelles pénibles pour l'homme.
- L'augmentation de la sécurité, etc…

Cependant, ils sont plus chers que des solutions informatiques classiques à base de microcontrôleurs par exemple mais demeurent à l'heure actuelle les seules plateformes d'exécution considérées comme fiables en milieu industriel (avec les ordinateurs industriels). Le prix est notamment dépendant du nombre d'entrées/sorties nécessaires, de la mémoire dont on veut disposer pour réaliser le programme, de la présence ou non des modules.

De plus ils nécessitent la maîtrise de langages spécifiques qui reprennent dans leur forme la logique d'exécution interne de l'automate. Ces langages apparaissent toutefois à beaucoup d'utilisateurs plus accessibles et plus visuels que les langages informatiques classiques.

I.10 Les langages de programmation des automates programmables

Un langage est un ensemble de caractères, de convention et de règles employés pour communiquer, traiter et échanger des informations.

Ils existent deux grandes familles de langages : Les langages littéraux et les langages graphiques.

I.10.1 Les langages littéraux

C'est un ensemble de langages dont les instructions s'écrivent sous forme d'expressions littérales utilisant des parties textuelles et des mots réservés.

On distingue les deux langages [2], [4] :

- **IL** : (**Instruction List**) liste d'instructions, le langage List est très proche du langage assembleur. On travaille au plus près du processeur en utilisant l'unité arithmétique et logique, ses registres et ses accumulateurs.
- **ST** : (**Structured Text**) littéral structuré, tels que les langages informatiques. Ce langage structuré ressemble aux langages de haut niveau utilisés pour les ordinateurs tels que (le C, le C++).

I.10.2 Les langages graphiques

Parmi les langages graphiques les plus utilisés on distingue [2] **:**

- **LD : (Ladder Diagram**), le langage Ladder (échelle en anglais) ressemble aux schémas électriques et permet de transformer rapidement une ancienne application faite de relais électromécaniques en un programme. (Voir figure I.8).
- Cette façon de programmer exploite une approche visuelle du problème longtemps appréciée en industrie, mais qui s'appuie sur une logique de moins en moins adaptée mais toujours utilisée (2013). On parle également de langage à contacts ou de schéma à contacts pour désigner ce langage Ladder.

- **Le grafcet ou SFC** : (**Séquentielle Fonction Chart**), est un outil graphique de définition de l'automatisme séquentiel, en un nombre défini d'étapes, séparées par des conditions de transition. Il utilise une représentation graphique claire.

- **FBD** : (**Function Block Diagram**) langages en blocs fonctionnels le FBD se présente sous forme diagramme : Suite de blocs, connectables entre eux, réalisant des opérations, simples ou très sophistiquées.

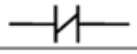
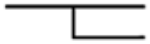
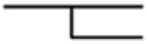
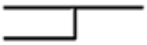
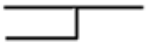
Fonction	Symbole	
	Européen	Américain
Contact ouvert au ropos	---o o---	
Contact fermé au ropos	---o̅ o---	
Début de branchement		
Fin de branchement		
Affectation	---( )---	---( )

Figure I.8: Symboles usuels en langages LD [5]

I.11 Conclusion

Dans ce présent chapitre, nous avons essayé de présenter quelques notions de base sur les automates programmables industriels. A travers l'étude menée, nous concluons qu'un automate programmable est un constituant capable de réaliser le fonctionnement d'un automatisme au moyen d'un programme qui est écrit et modifié à partir d'un terminal de programmations et de réglage.

L'objectif du prochain chapitre sera consacré à l'étude de la carte Arduino Méga ADK qui sera utilisée pour la conception d'un API à base de microprocesseur.

Chapitre II

Etude de la carte

Arduino Méga (ADK)

II.1 Introduction

Issu d'une équipe d'enseignants et d'étudiants de l'école de design d'Interaction d'Ivrea en (Italie), le projet Arduino est sorti en 2005 comme un modeste outil pour de nombreux artistes, passionnés, étudiants, et tous ceux qui rêvaient d'un tel gadget. Il a initié une révolution DIY dans l'électronique à l'échelle mondiale [6].

Pour cela une équipe de professeurs et d'étudiants (David Mellis, Tom Igoe, Gianluca Martino, David Cuartielles, Massimo Banzi ainsi que Nicholas Zambetti) avait pour objectif la conception d'une carte à un prix particulier de 30$, accessible aux étudiants. L'environnement Arduino est particulièrement adapté à la production artistique ainsi qu'au développement de conceptions qui peuvent trouver leurs réalisations dans la production industrielle [7].

Vu que l'Arduino Méga (ADK) est l'élément de base de notre réalisation, dans ce présent chapitre, nous allons approfondir nos connaissances en donnant un aperçu globale sur la carte sa structure interne et son schéma électrique.

II.2 Description de la carte Arduino Méga (ADK)

L'Arduino Méga (ADK) est une carte microcontrôleur basée sur l'ATmega2560. Elle dispose d'un ATmega 8U2 programmé comme un convertisseur USB-série, et d'une interface hôte USB pour se connecter avec les téléphones basés sur Androïde [8], basé sur le MAX3421e IC. Ceci est illustré par la figure ci-dessous

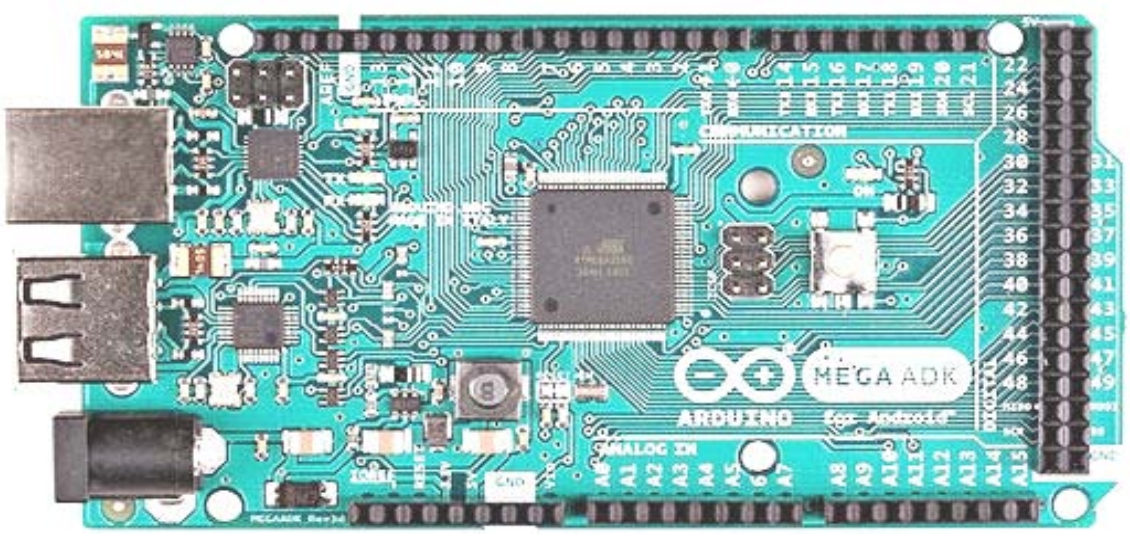


Figure II.1: Carte Arduino Méga (ADK) [8]

Les projets réalisés avec cette carte peuvent être autonomes, comme ils peuvent communiquer avec d'autres logiciels sur l'ordinateur tel que Flach, Procesing, ou Matlab.

Elle dispose de 54 broches d'entrées / sorties numériques (dont 15 peuvent être utilisées comme sorties PWM), 16 entrées analogiques, 4 UART (ports série matériels), un oscillateur à quartz 16 MHz, une connexion USB, une prise d'alimentation, un connecteur ICSP, et un bouton de réinitialisation [8].

L'Arduino Méga (ADK) est basée sur l'ATmega 2560, elle dispose d'un ATmega8U2 programmé comme un convertisseur USB-série.

II.3 Spécifications techniques de la carte Arduino Méga (ADK) [8]

- Microcontrôleur : ATmega2560
- Tension de fonctionnement : 5 V
- Gamme de tension d'entrée (recommandée) : 7-12 V
- Gamme de tension d'entrée (limite) : 6-20 V
- Pins digitaux E /S : 54
- Pins digitaux E/S PWM : 15
- Pins d'entrée analogique : 16
- Courant direct par pin E/S : 40 mA
- Mémoire flash : 256 KB
- Mémoire Flash du Boot loader : 8 KB
- SRAM : 8 KB
- EEPROM : 4 KB
- Fréquence d'horloge de l'oscillateur à quartz : 16 MHz
- Puce hôte USB : MAX3421E
- Dimensions : 101,52 x 53,3 mm
- Poids : 37 g

II.4 Alimentation de la carte Arduino Méga (ADK)

La carte Arduino Méga ADK peut-être alimentée soit par la connexion USB (qui fournit 5V jusqu'à 500mA) ou à l'aide d'une alimentation externe. La source d'alimentation est sélectionnée automatiquement par la carte [8].

L'alimentation externe (non-USB) peut être soit un adaptateur secteur (pouvant fournir de 7V à 12V) ou des piles (accumulateurs). L'adaptateur secteur peut être connecté en branchant une prise 2.1mm positif au centre dans le connecteur jack de la carte. Les fils en provenance d'un bloc de piles ou d'accus peuvent être insérés dans les connecteurs des broches de la carte appelées Gnd (masse ou 0V) et Vin (Tension positive en entrée) du connecteur d'alimentation [6].

La gamme de fonctionnement de la carte est comprise entre 6 à 20 volts. Cependant, si la carte est alimentée avec moins de 7V, la broche 5V pourrait fournir moins de 5V et la carte pourrait être instable. Si on utilise plus de 12V, le régulateur de tension de la carte pourrait chauffer et endommager la carte. Aussi, la plage idéale pour alimenter la carte est entre 7V et 12V [6], [8].

Les broches d'alimentation sont les suivantes [6] :

- VIN. La tension d'entrée positive lorsque la carte Arduino est utilisée avec une source de tension externe (à distinguer du 5V de la connexion USB ou autre source 5V régulée).

- 5V. La tension régulée utilisée pour faire fonctionner le microcontrôleur et les autres composants de la carte. Le 5V régulé fourni par cette broche peut donc provenir soit de la tension d'alimentation VIN via le régulateur de la carte, ou bien de la connexion USB (qui fournit du 5V régulé) ou de tout autre source d'alimentation régulée.

- Une alimentation de 3.3V fournie par le circuit intégré FTDI (circuit intégré faisant l'adaptation du signal entre le port USB de l'ordinateur et le port série de l'ATmega) de la carte est disponible : Ceci est intéressant pour certains circuits externes nécessitant cette tension au lieu du 5V). L'intensité maximale disponible sur cette broche est de 50mA

- GND : Broche de masse (ou 0V).

II.5 Avantages de la carte Arduino Méga (ADK)

Grâce à son utilisation simple et accessible, Arduino a été utilisé dans des milliers de projets et applications. Son logiciel est facile à utiliser pour les débutants, mais suffisamment flexible pour les utilisateurs avancés. Il est utilisé pour construire des instruments scientifiques à faible coût, ou pour commencer avec la programmation et la robotique.

L'Arduino Méga (ADK) est un outil clé qui permet d'apprendre de nouvelles choses, elle permet l'utilisation des appareils Androïde qui peuvent être des stations d'accueil audio, des appareils d'exercice, des appareils de test médical personnel, des stations météorologiques ou tout autre périphérique externe qui ajoute à la fonctionnalité d'Androïde, et ce si grâce à la fonction (ADK) (kit de développement d'accessoires Android) [6].

Il offre plusieurs avantages par rapport à d'autres cartes utilisé dans d'autres systèmes :

- **Peu coûteuse :**

La carte Arduino Méga (ADK) est relativement peu coûteuse par rapport aux autres plates-formes de microcontrôleurs notamment les API. La version la moins chère du module Arduino peut être assemblée à la main, et même les modules pré-assemblés sont peu coûteux.

- **Multiplateformes :**

Le logiciel Arduino (IDE) fonctionne sur les systèmes d'exploitation Windows, Macintosh OSX et Linux. La plupart des systèmes à microcontrôleur se limite à Windows.

- **Environnement de programmation simple et clair :**

Le logiciel Arduino (IDE) est facile à utiliser pour les débutants, et suffisamment flexible pour que les utilisateurs avancés puissent en profiter. Il est basé sur un environnement de programmation qui est une variante du C et C++, allégé et restreint à l'utilisation de la carte.

- **Logiciels open source et extensibles :**

Le logiciel Arduino est publié en tant qu'outil open source, disponible pour extension par des programmeurs expérimentés. Le langage peut être étendu à travers des bibliothèques C++.

- **Matériel Open Source et matériel extensible** :

Les plans de la carte Arduino sont publiés, de sorte que les concepteurs de circuits expérimentés peuvent créer leur propre version du module, l'étendre et l'améliorer. Même les utilisateurs relativement inexpérimentés peuvent construire la version maquette du module afin de comprendre comment cela fonctionne et économiser de l'argent.

II.6 Répartition des broches de l'Arduino Méga (ADK)

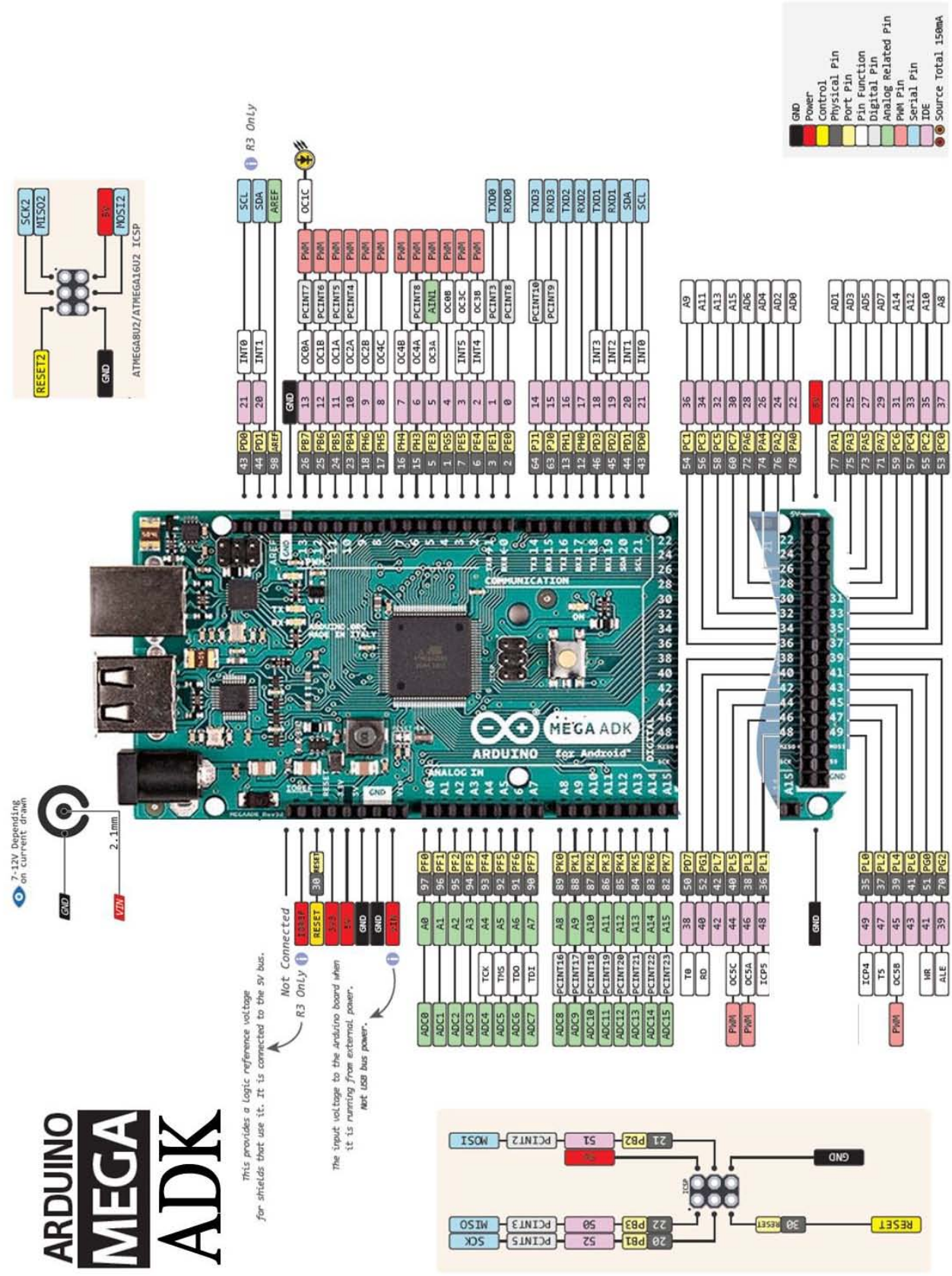


Figure II.2 : Schéma des broches de l'Arduino Méga (ADK)

II.7 Généralités sur le langage Arduino

II.7.1 La structure d'un programme [6], [8]

Un programme Arduino comporte trois parties illustrées dans la figure II.3 :

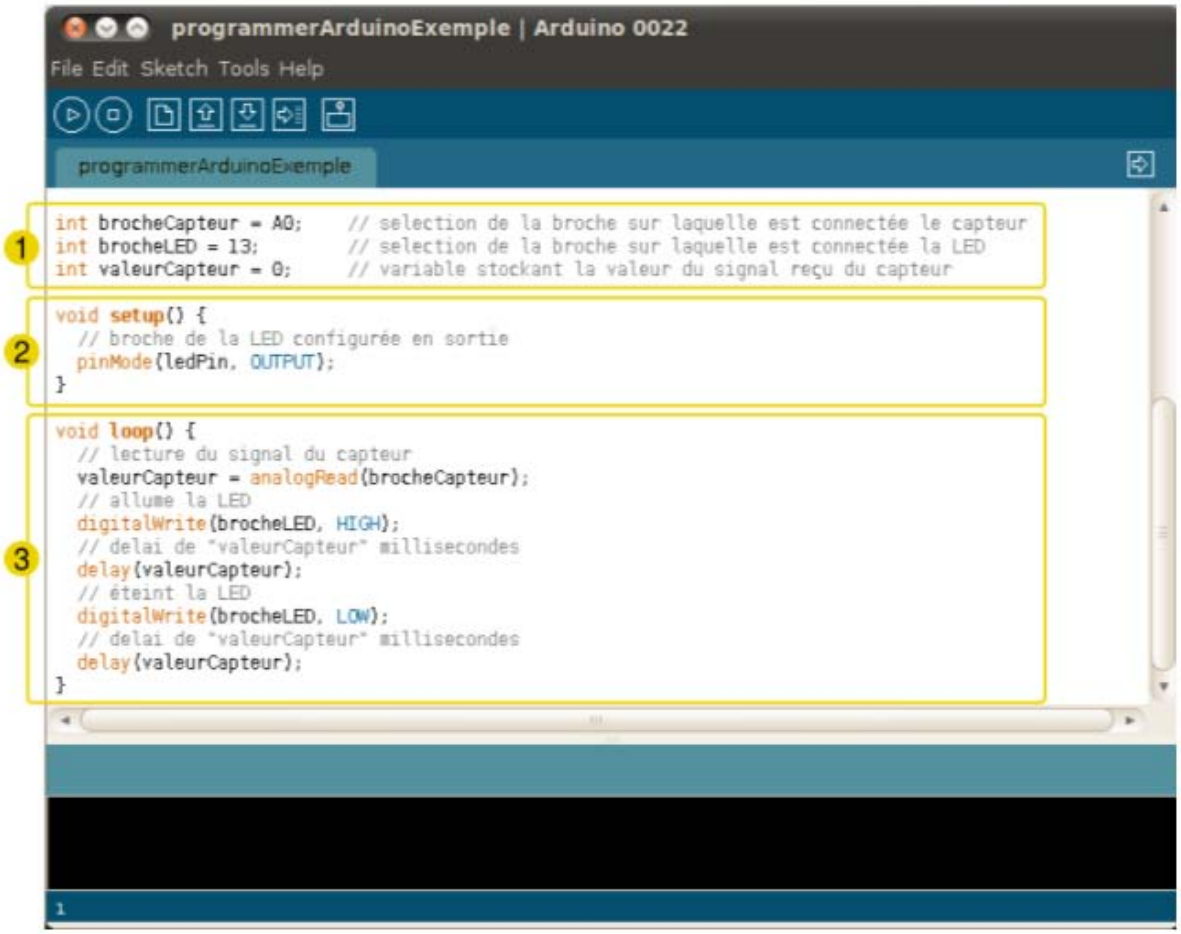


Figure II.3: Les différentes parties que comporte un programme Arduino

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

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

II.7.2 Coloration syntaxique

Dans un programme écrit en langage Arduino, on trouve certains mots apparaissent en différentes couleurs dans l'interface de programmation [6], [10], ceci permet d'indiquer le statut des différents éléments :

En orange, apparaissent les mots-clés reconnus par le langage Arduino comme des fonctions existantes.
Exemple : digitalWrite.

En bleu, apparaissent les mots-clés reconnus par le langage Arduino comme des constantes.
Exemple : int.

En gris, apparaissent les commentaires qui ne seront pas exécutés dans le programme. Il est utile de bien commenter son code pour s'y retrouver facilement ou pour le transmettre à d'autres personnes.

Exemple : // LED pin est une sortie.

La déclaration d'un commentaire s'effectue de deux manières différentes :

- dans une ligne de code, tout ce qui se trouve après « **//** » sera un commentaire.
- l'on peut encadrer des commentaires sur plusieurs lignes entre « **/*** » et « ***/** ».

II.7.3 La syntaxe du langage Arduino

Le langage Arduino utilise une syntaxe simple non compliquée, ce qui rend la carte Arduino facile à programmer pour les débutants, mais suffisamment flexible pour les programmeurs, cette syntaxe se compose des éléments suivants :

II.7.3.1 La ponctuation [6], [10]

Le code est structuré par une ponctuation stricte :

- toute ligne de code se termine par un point-virgule « **;** »
- le contenu d'une fonction est délimité par des accolades « **{** » et «} »
- les paramètres d'une fonction sont contenus entre des parenthèses « **(** » et «) ».

II.7.3.2 Les variables

Une variable est un espace réservé dans la mémoire de la carte. C'est comme un compartiment d'une taille bien définie contenant un seul type d'information. Elle est caractérisée par un nom qui permet d'y accéder facilement [10].

Il existe différents types de variables identifiés par un mot-clé dont les principaux sont [6] :

- nombres entiers (int)
- nombres à virgule flottante (float)
- texte (String)
- valeurs vrai/faux (boolean).

Exemple le nombre à décimales 3.14159, peut se stocker dans une variable de type float. On utilise un point et non une virgule pour les nombres à décimales.

Dans un programme Arduino, il est nécessaire de déclarer les variables pour leur réserver un espace mémoire adéquat. On déclare une variable en spécifiant son type, son nom puis en lui assignant une valeur initiale (optionnelle).

Exemple :
int ma variable = 45;

// int est le type, ma variable le nom et = 45 assigne une valeur.

II.7.3.3 Les fonctions [10]

Une fonction (également désignée sous le nom de procédure) est un bloc d'instructions que l'on peut appeler à tout endroit du programme.

Le langage Arduino est constitué d'un certain nombre de fonctions, par exemple analogRead (), digitalWrite () ou delay ().

Il est possible de déclarer ses propres fonctions par exemple :

```
Void clignote () {
   digitalWrite (brocheLED, HIGH);
   delay (1000);
   digitalWrite (brocheLED, LOW);
   delay (1000) ;
}
```

Pour exécuter cette fonction, il suffit de taper la commande : clignote ();

On peut faire intervenir un ou plusieurs paramètres dans une fonction :

```
   void clignote (int broche, int vitesse) {
   digitalWrite (broche, HIGH);
   delay (1000/vitesse);
   digitalWrite (broche, LOW);
   delay (1000/vitesse);
}
```

Dans ce cas, l'on peut moduler leurs valeurs depuis la commande qui l'appelle :

```
clignote(5,1000); //la sortie 5 clignotera vite
clignote(3,250); //la sortie 3 clignotera lentement
```

II.7.3.4 Les structures de contrôle [6], [8]

Les structures de contrôle sont des blocs d'instructions qui s'exécutent en fonction d'un certain nombre de conditions.

Il existe quatre types de structure :

- **if...else :**

  La boucle if exécute un code si certaines conditions sont remplies et exécutera un autre code si les conditions ne le sont pas.

Exemple :

```
//si la valeur du capteur dépasse le seuil
if (valeurCapteur>seuil) {
   //appel de la fonction clignote
   clignote() ;}
```

❖ while :

Tant que certaines conditions sont remplies, l'instruction exécute un code bien défini.

Exemple :

```
//tant que la valeur du capteur est supérieure à 250
while (valeurCapteur>250){
   //allume la sortie 5
   DigitalWrite (5, HIGH);

}

DigitalWrite (5, LOW);
```

❖ for :

La boucle for exécute un code pour un certain nombre de fois, elle est utilisée plus fréquemment pour les compteurs.

Exemple :

```
//pour i de 0 à 255, par pas de 1
for (int i=0; i <= 255; i++){
   analogWrite (PWMpin, i);
   delay(10);   }
```

❖ switch/case :

L'instruction switch/case permet de faire un choix entre plusieurs codes parmi une liste de possibilités.

Exemple :

```
// fait un choix parmi plusieurs messages reçus
switch (message) {

   case 0: //si le message est "0"
      //allume que la sortie 3
      digitalWrite(3,HIGH);
      digitalWrite(4,LOW);
      break;
```

```
case 1: //si le message est "1"
    //allume que la sortie 4
    digitalWrite(3,HIGH);
    digitalWrite(4,LOW);
    digitalWrite(5,LOW);
    break;

case 2: //si le message est "2"
    //allume que la sortie 5
    digitalWrite(3,LOW);
    digitalWrite(4,LOW);
    digitalWrite(5,HIGH);
    break; }
```

II.8 Logiciel de programmation l'IDE d'Arduino

II.8.1 Généralités sur L'IDE d'Arduino

L'IDE d'Arduino est un environnement de développement open source gratuit, téléchargeable sur le site officiel Arduino, son interface (figure II.4) se compose d'un ensemble d'outils développés qui rend les cartes Arduino facilement programmables [6], [7].

Figure II.4 : L'interface de l'IDE d'Arduino [8]

II.8.1.1 Présentation du logiciel [6], [10]

L'IDE Arduino se compose de 4 parties illustré par la (Figure II.5) :

- Le cadre numéro 1 : Ce sont les options de configuration du logiciel.
- Le cadre numéro 2 : Il contient les boutons nécessaires pour la programmation des cartes.
- Le cadre numéro 3 : C'est le bloc qui contiendra le programme crée.
- Le cadre numéro 4 : Il aide à corriger en signalant les erreurs commises dans le programme. C'est le débogueur.

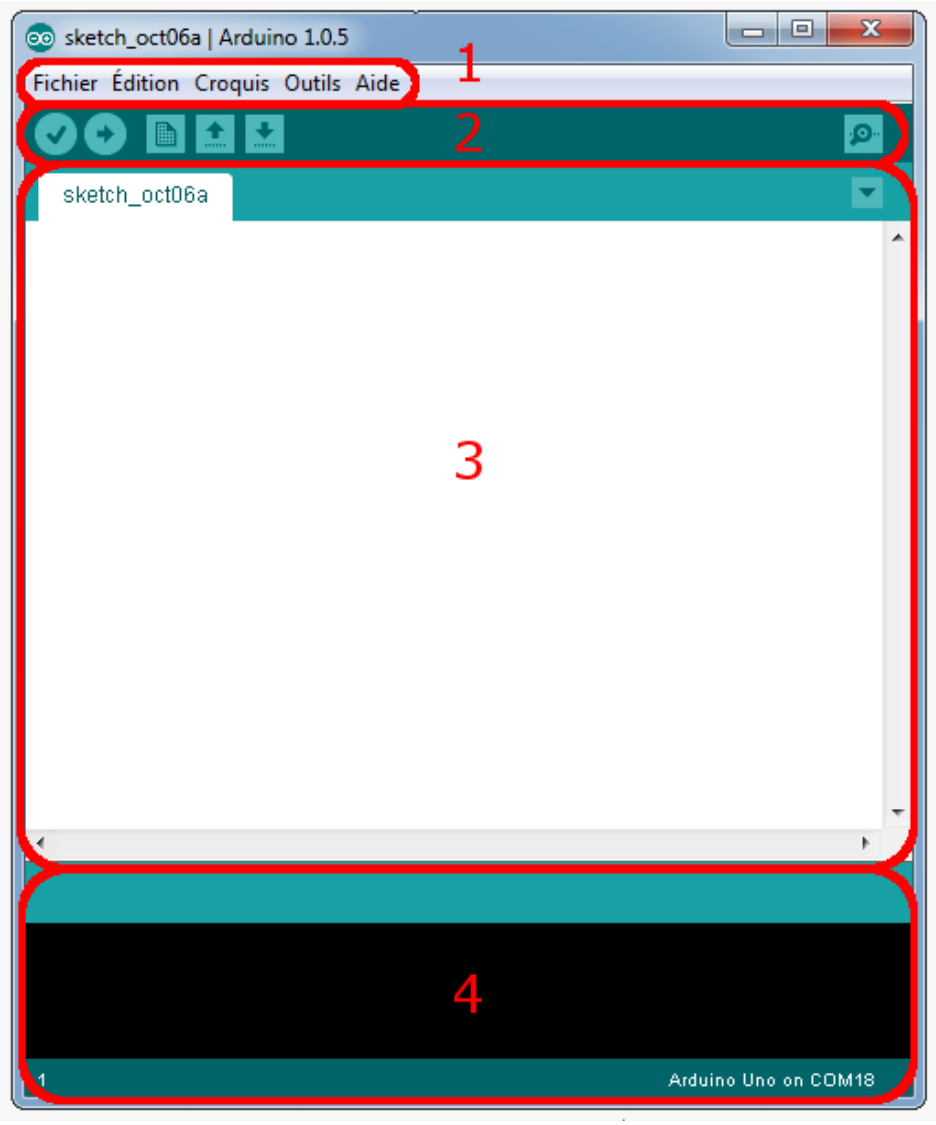


Figure II.5 : Les différentes parties de l'interface de l'IDE d'Arduino

II.8.1.2 Les différents boutons de l'interface de l'IDE d'Arduino [6], [7]

La barre d'outils de l'IDE d'Arduino comporte six boutons, encadrés en rouge et numérotés dans la figure II.6 :

- Bouton 1 : Ce bouton permet de vérifier le programme, il actionne un module qui cherche les erreurs dans le programme écrit.
- Bouton 2 : Charge (téléverse) le programme dans la carte Arduino sélectionné dans (outils).
- Bouton 3 : Il permet de créer un nouveau fichier.
- Bouton 4 : Il permet d'ouvrir un fichier déjà enregistré, ou présent dans la liste des exemples fournie par Arduino.
- Bouton 5 : C'est le bouton de sauvegarde, il permet d'enregistrer le fichier ou d'enregistrer les modification apporté a ce dernier.
- Bouton 6 : Ce bouton ouvre le moniteur série, qui permet de communiquer avec la carte Arduino à travers la liaison série.

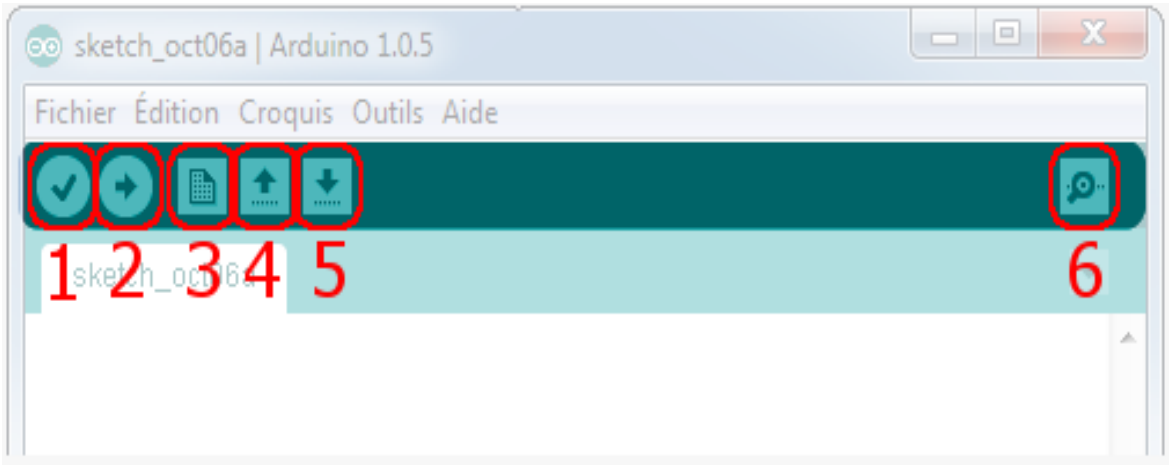


Figure II.6 : La barre d'outils de l'IDE d'Arduino

II.9 Autres logiciels de programmation d'Arduino :

Plusieurs autres logiciels que l'IDE Arduino ont été développés pour faciliter la programmation des cartes Arduino. De formes diverses, ils permettent tous une programmation plus simplifiée des cartes. Ces différents logiciels permettent de programmer les cartes Arduino avec des langages littéraux ou graphiques [9].

II.9.1 Les logiciels de programmation à langages littéraux :

Les logiciels de programmation à langages littéraux sont ceux qui utilisent un ensemble de langages dont les instructions s'écrivent sous forme d'expressions littérales utilisant des parties textuelles et des mots réservés, et parmi ces logiciels on trouve :

- **Slober** : qui est un outil de développement C et C ++, le logiciel **Slober** est l'IDE Arduino conçu par éclipse, il est disponible en open source. (Voir son interface dans la figure II.7).

- **AsmEditor** : qui est un Environnement de Développement Intégrés (E.D.I.) permettant la création et la compilation de projets en langage assembleur.

- **LARP** : Logiciel d'Algorithmes et de Résolution de Problèmes.

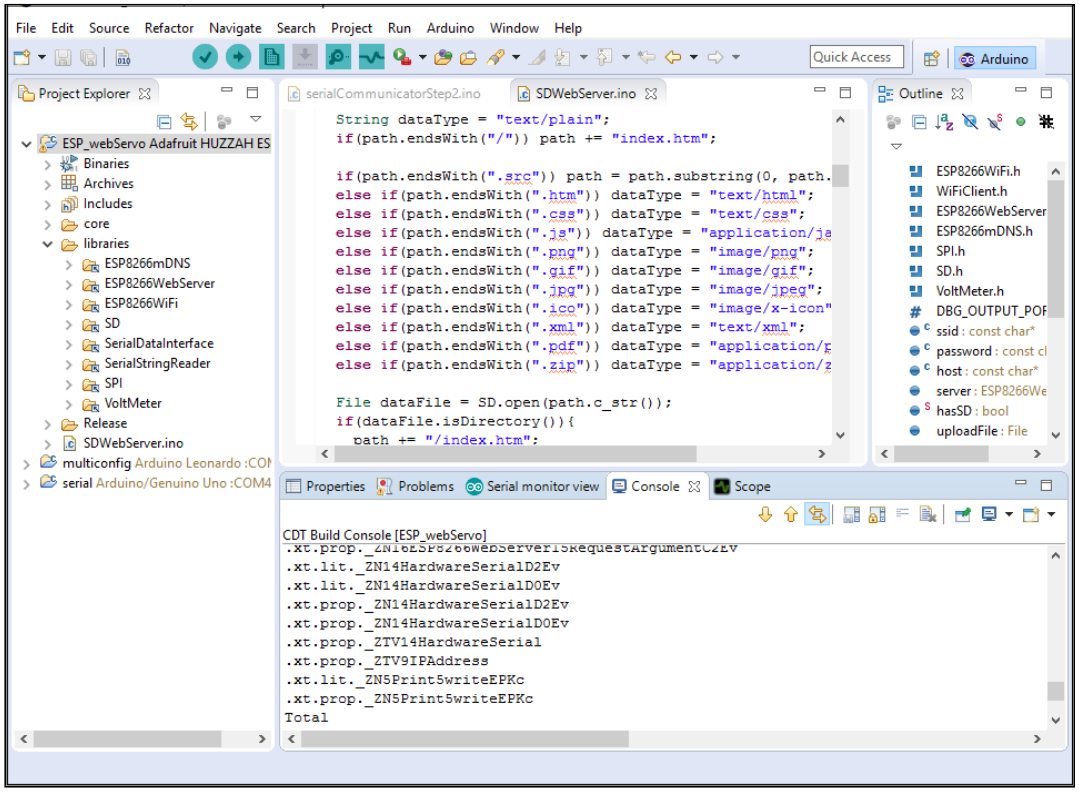


Figure II.7 : La fenêtre de Slober

II.9.2 Les logiciels de programmation à langages graphiques :

Les logiciels de programmation à langage graphique permettent une programmation par assemblage de blocs fonctionnels. Distribués sous forme de plug-ins, ils permettent une programmation simple de la carte. Et parmi ces logiciels, on peut citer :

- **Blockly** : développé par Google à partir d'App Inventor.
- **Ardublock :** C'est un environnement de développement qui permet la création et la compilation de projets, et ceci par assemblage de blocs fonctionnels. (Voir figure II.8).
- **S4A :** Scratch for Arduino.
- **Simulink** : basé sur MATLAB et orienté sur les systèmes multi-physiques.

Figure II.8 : Quelques blocs de codage

II.10 Connectivité et communication

L'utilisation de la carte Arduino Méga (ADK) dans des projets complexes requière plus de flexibilité. Pour cela les cartes ont besoin de connectivité et de communication, et ceci est assuré par différents shield [10], disponibles sous forme de cartes modulaires qui se greffent sur l'Arduino pour ajouter des fonctionnalités supplémentaires, telle que le contrôle de servomoteur, l'ajout de mémoire et les différentes connectivités (Wifi, Bluetooth, GPS) [6].

Tous ces shield rendent l'Arduino plus qu'une simple carte de développement, mais un outil professionnel très puissant. Parmi ces shield, nous pouvons citer :

- **Interface hôte USB :**

  L'Arduino Méga (ADK) dispose d'une interface hôte USB pour se connecter avec les téléphones basés sur Android [8], et ceci à l'aide de l'ATmega8U2 programmé comme un convertisseur USB-série. (Voir figure II.9).

  Cette interface permet aux appareils sous Android de prendre complètement le contrôle de la carte Arduino.

Elle permet aussi à l'Arduino d'utiliser tous les capteurs et interfaces de l'appareil connecté.

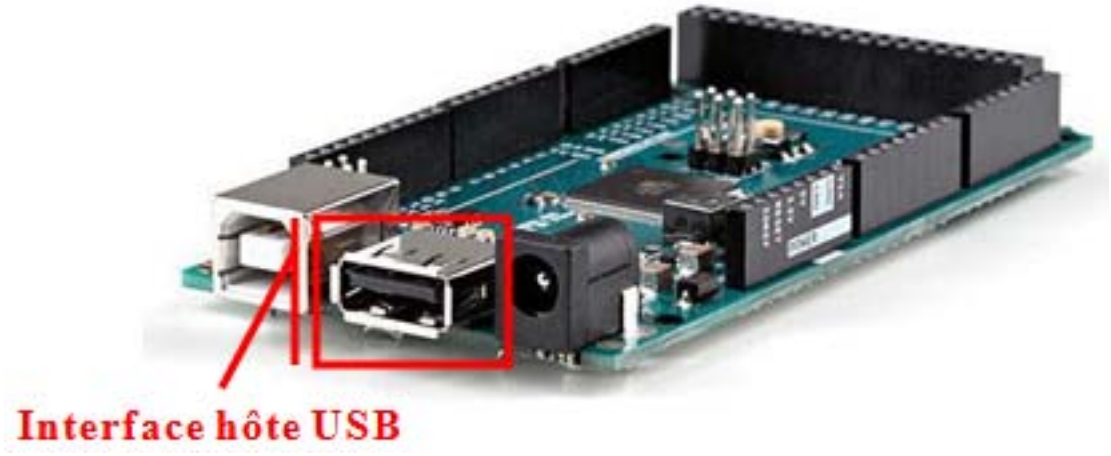


Figure II.9: Interface hôte USB

❖ **Arduino Ethernet Shield :**

C'est une carte très utilisée par les développeurs. (Voir figure II.11). L'Ethernet Shield fournit une capacité de connexion au World Wide Web [3]. Ce module est équipé d'un port micro-SD permettant d'ajouter un espace de stockage, il possède aussi une excellente bibliothèque pour le soutenir disponible en open source.

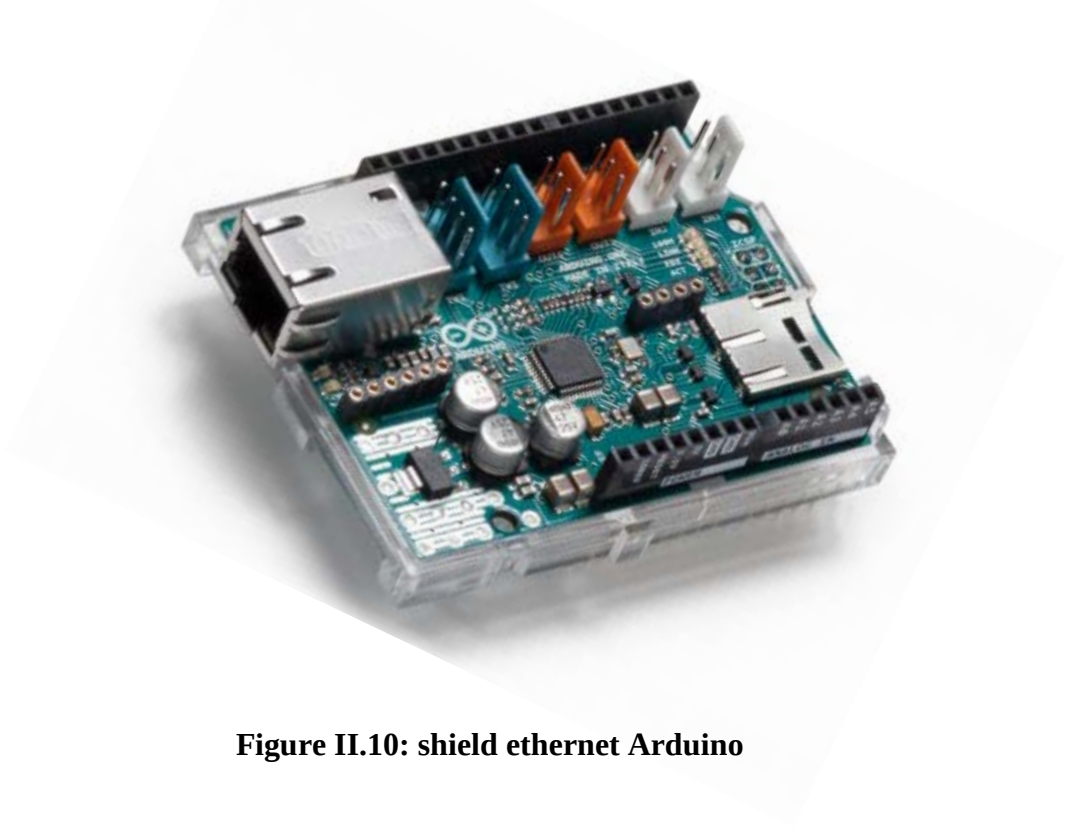


Figure II.10: shield ethernet Arduino

❖ **Shield XBee :**

Le module XBee illustré par la figure II.11 permet à une carte Arduino de communiquer sans fil. Il peut communiquer jusqu'à 100 pieds à l'intérieur ou 300 pieds à l'extérieur (avec une visibilité directe). Il peut aussi être utilisé comme un remplacement série / USB [10].

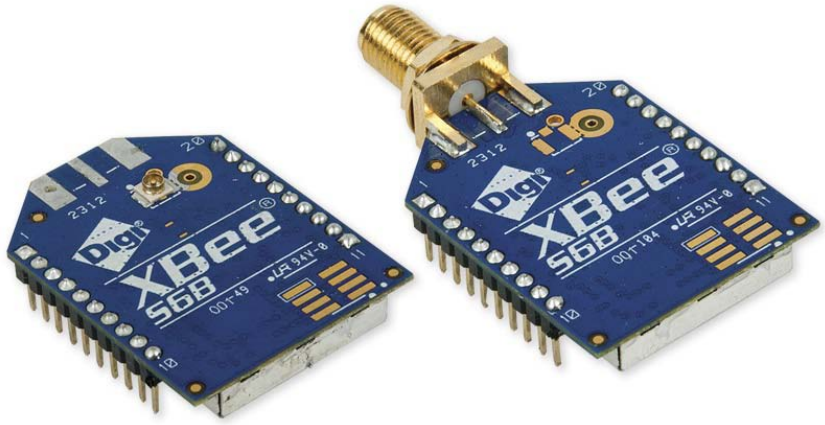


Figure II.11: Module XBee wifi

❖ **Shield écran TFT tactile:**

Cet écran dispose d'une résolution bien supérieure aux affichages noir et blanc. Ce module est équipé d'un port micro-SD permettant d'ajouter un espace de stockage, et il est équipé d'un écran tactile résistif, qui permet de détecter la pression d'un doigt partout sur cet écran [10].

Ce shield est complètement assemblé, il se branche simplement sur l'Arduino ce qui facilite son utilisation. (Voir figure II.12)

Figure II.12: Shield écran TFT tactile

II.10 Conclusion

Dans ce chapitre, nous avons retracé l'histoire intéressante du projet Arduino. Ici, nous nous sommes basés sur la carte Arduino Méga (ADK) où on a présenté sa structure, son logiciel et son langage de programmation, ainsi que d'autres logiciels utilisant d'autres langages. Ensuite, nous avons présenté les aspects matériels et logiciels de la carte Arduino Méga ADK.

A travers cette étude, nous avons conclu que la carte Arduino Méga possède beaucoup d'avantages comme sa souplesse d'utilisation dans plusieurs domaines, tels que l'automatisme et la robotique, etc…

Le prochain chapitre sera consacré à la conception et à la simulation des différents modules d'un API qui sera construit autour de la carte Arduino Méga (ADK).

Chapitre III

Conception et simulation

III.1 Introduction

Après les généralités sur les API et l'étude de la carte Arduino Méga (ADK), respectivement aux premier et deuxième chapitres, nous consacrons ce chapitre à la conception et la phase simulation de notre automate programmable.

Pour concevoir cette API, nous proposons d'abord d'exposer le schéma synoptique général du système, ensuite nous détaillerons les différents blocs constituant le dispositif.

III.2 Schéma synoptique du système

Le système proposé est construit autour d'une carte Arduino Méga. Il est constitué de plusieurs modules d'entrées sorties avec une alimentation générale du dispositif.

Le schéma synoptique du système proposé est présenté ci-dessous :

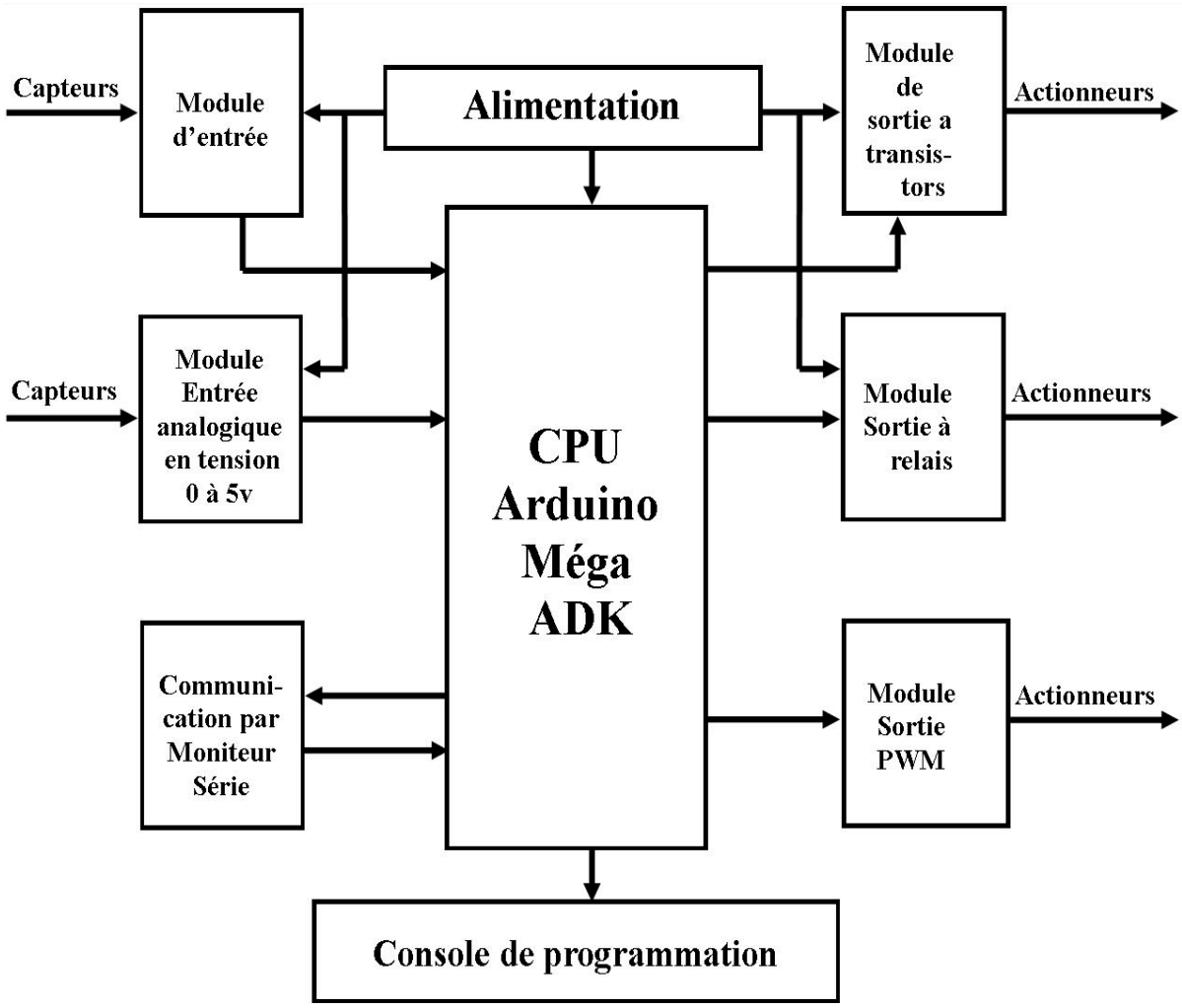


Figure III.1 Schéma synoptique du système

III.3 Principe de fonctionnement du système

Le système que nous proposons est un API dont son principe de fonctionnement repose sur la CPU conçu à base de la carte Arduino Méga (ADK).

L'API reçoit des données par ses entrées (analogiques/numériques), à travers les modules d'entrées élaborés, celles-ci sont ensuite traitées par un programme défini installé à l'aide d'un PC ou d'un appareil androïde. Le résultat obtenu après traitement sera délivré par ses sorties à travers les différents modules de sortie conçus.

Le temps de cycle de l'API varie selon la taille du programme et la tâche assignée à ce dernier.

III.4 Etude et conception des différents modules de l'API

En suivant la structure de l'API qu'on a mentionnée dans le chapitre I, nous sommes parvenus à déterminer les différents modules à concevoir. Ces derniers sont les suivants :

- Module Alimentation.
- La CPU.
- Module d'entrée numérique.
- Module de sortie à transistor.
- Module de sortie à relais.

III.4.1 Alimentation stabilisée

Cette alimentation doit permettre de fournir l'énergie nécessaire au bon fonctionnement de la CPU, ainsi qu'aux autres modules constituant le dispositif. L'alimentation fournira trois tensions continues de 24V-8A, 12V-800mA, et 5V-600mA et cela à partir du 220V du secteur.

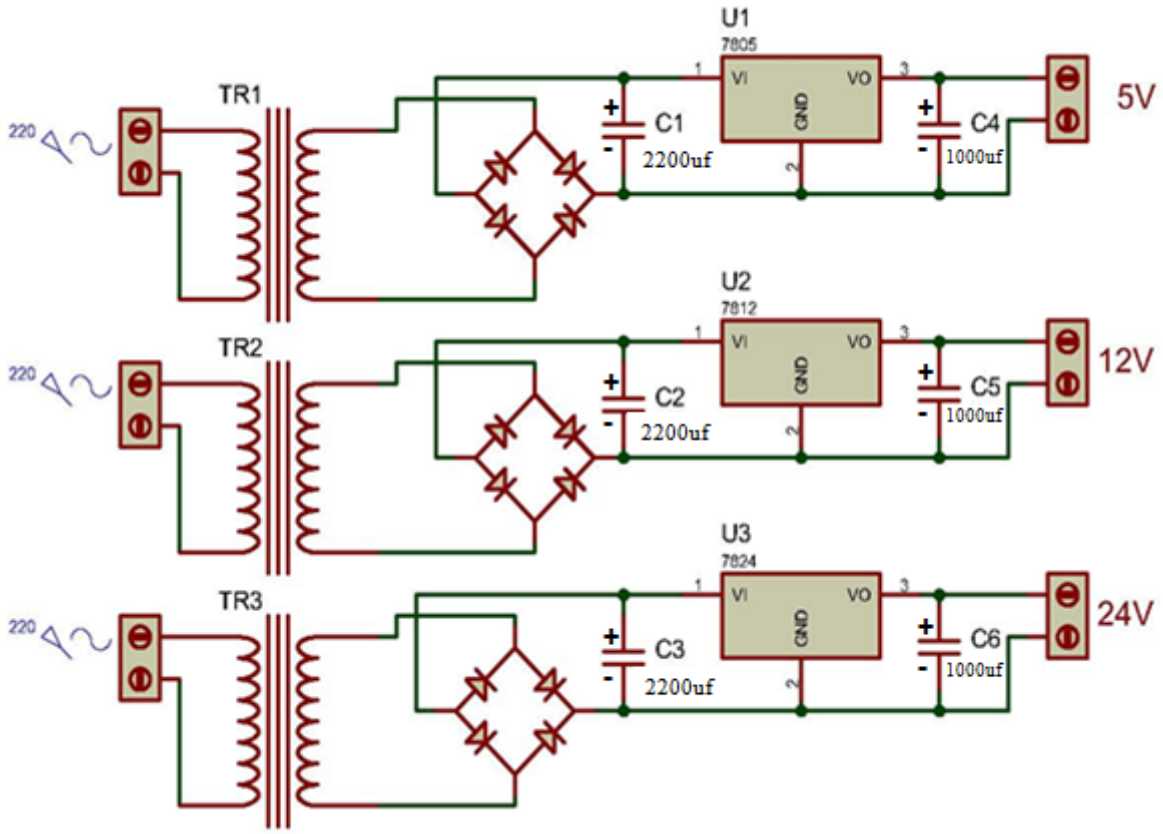


Figure III.2 Schéma de conception de la carte d'alimentation.

III.4.2 La CPU

Le module CPU va contenir la carte Arduino qui est l'organe essentiel de notre API, et lui permettra de communiquer avec les autres modules par le biais de connexion filaire (câble RJ 45). Il servira aussi d'interface d'entrées/sorties pour les entrées analogiques et les sorties PWM. Ceci est illustré par la figure ci-dessous :

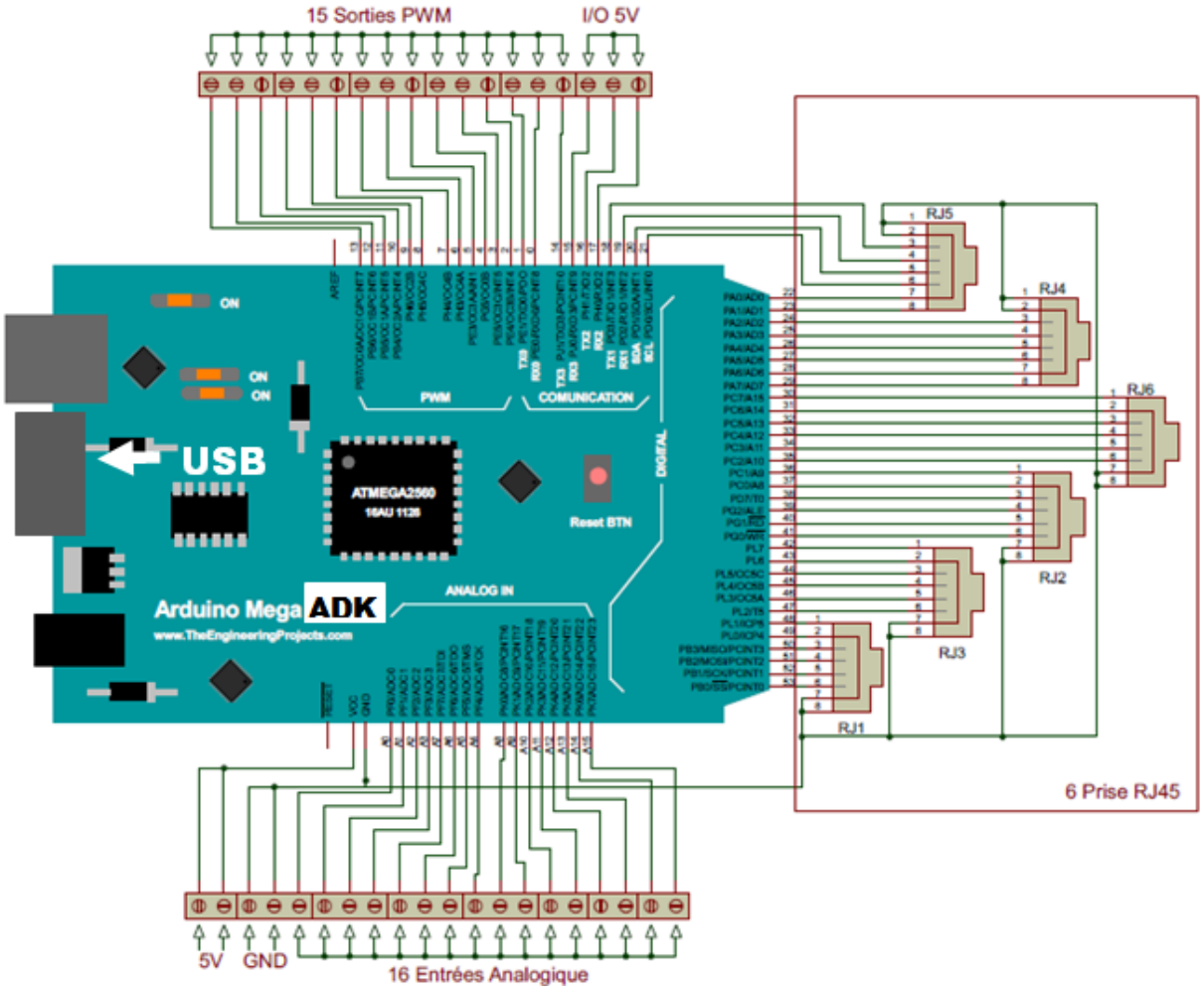


Figure III.3 Schéma de conception du module de l'Arduino (CPU)

III.4.3 Module d'entrée

Il sera composé de 12 entrées, ce module servira à convertir les tensions fournies par les capteurs présents sur la machine (tel que les boutons poussoirs et les capteurs de fin de course) vers la tension supportée par la CPU, donc convertir du 24V en 5V. Pour cela, nous avons utilisé un régulateur LM7805 et un photocoupleur PC817 qui permettra d'assurer l'isolation galvanique, entre la puissance et la commande (voir figure III.4).

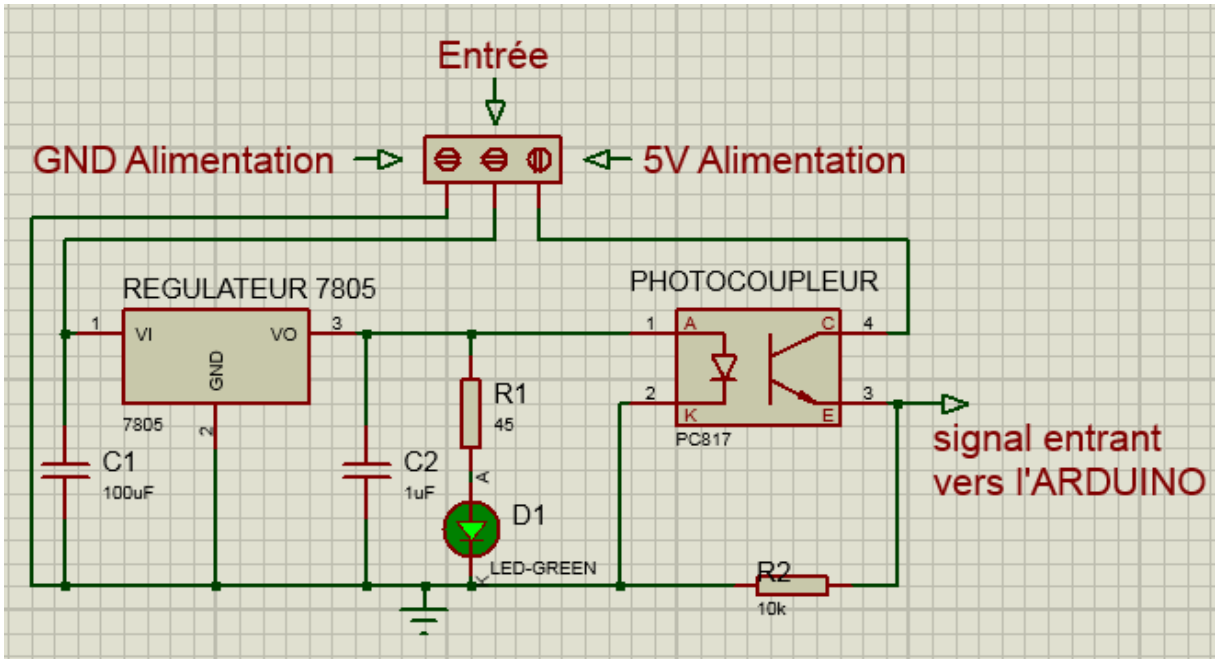


Figure III.4 Schéma conçu pour le module d'entrée

III.4.4 Module de sortie à transistor de puissance

Le module de sortie est en réalité un amplificateur qui permet de délivrer (24V ,8A) à partir du (5V, 20mA) généré par les sorties de l'Arduino. Et ce, en utilisant un transistor de puissance de référence « IRFI1010N ». Tout le circuit est protégé par un photocoupleur de même référence que le module d'entrée. Ce module sera composé de 12 sorties. Ceci est illustré par la figure ci-dessous :

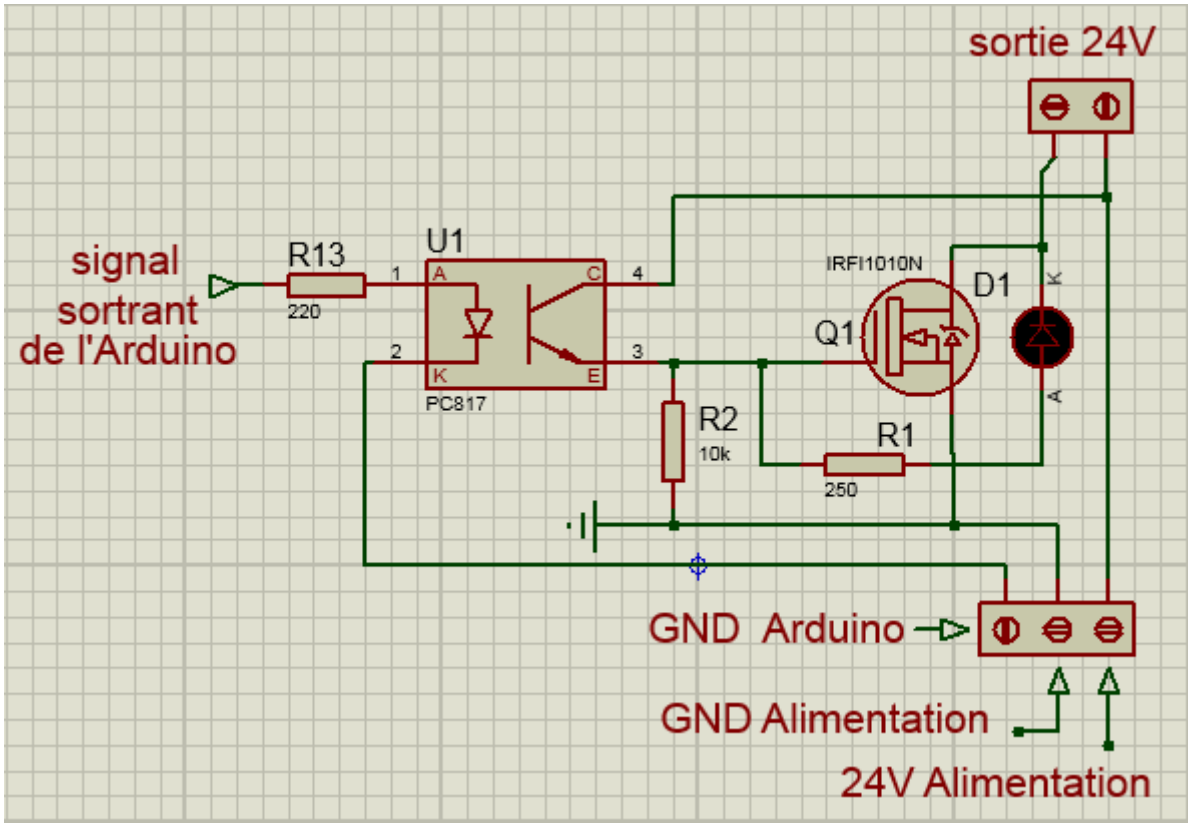


Figure III.5 Schéma conçu pour le module de sortie

III.4.5 Module de sortie à relais

Ce dernier module comporte 12 relais **(**SPDT). Ces organes électromécaniques permettent de dissocier la partie puissance de la partie commande. Elle permet l'ouverture et la fermeture d'un circuit électrique par un second circuit complètement isolé. (Voir figure III.6). Ces commandes s'effectuent en recevant des impulsions provenant de la carte Arduino en passant par le circuit d'isolation galvanique PC817.

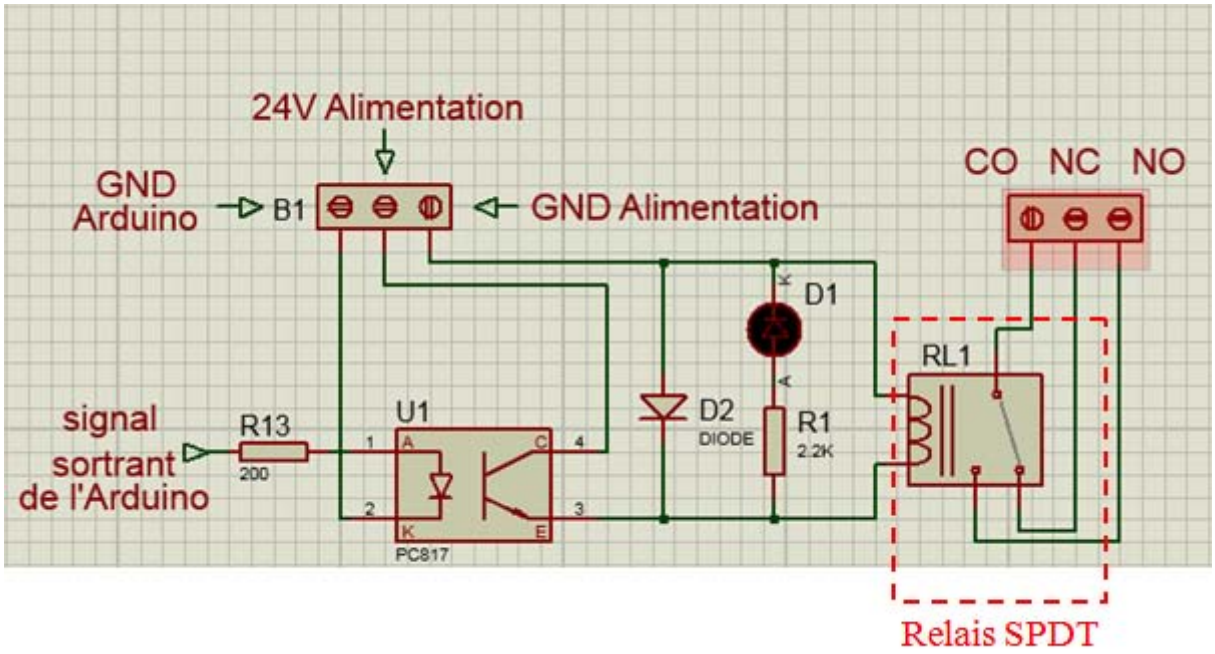


Figure III.6 Schéma conçu pour le module à relais

III.5 Simulation sur Proteus

III.5.1 Présentation du logiciel proteus [11]

Avant la réalisation pratique de nos cartes, nous avons réalisé la simulation de ces dernières en utilisant le logiciel proteus.

Proteus est un outil qui permet de dessiner des schémas électroniques, de les simuler et de réaliser le circuit imprimé correspondant. Ce logiciel comprend deux principaux modules ISIS et ARES.

III.5.1.1 ISIS

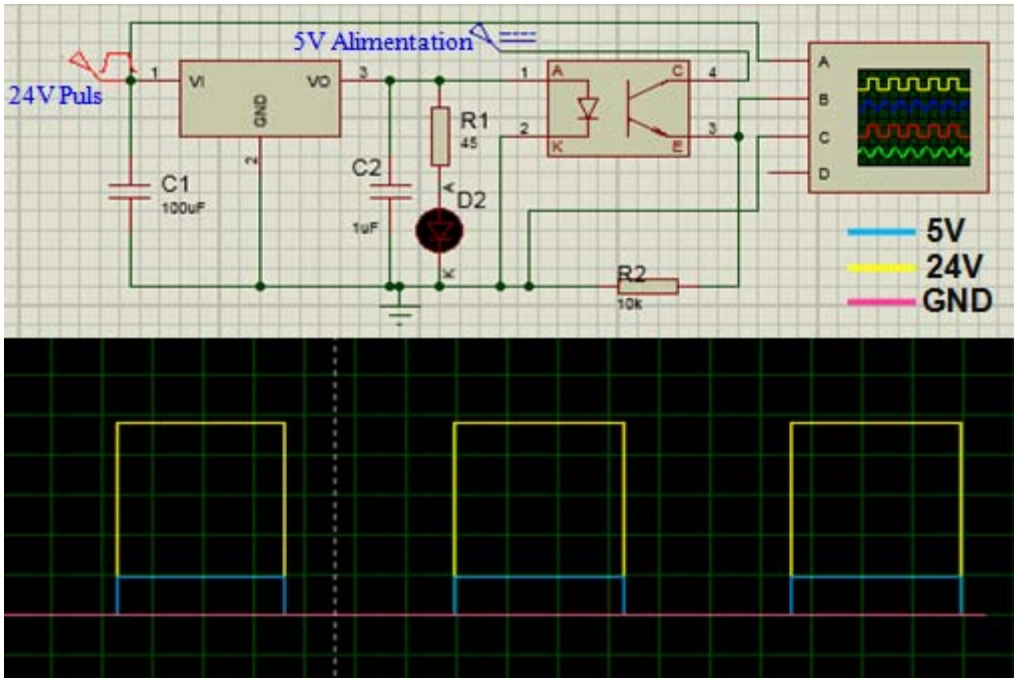
Connu comme éditeur de schémas électriques, ISIS de proteus simule ces derniers, pour déceler les erreurs des différentes étapes de conception. Les circuits électriques conçus grâce à ce logiciel peuvent être utilisés dans des documentations car ce dernier permet d'enregistrer les circuits sur différents formats graphiques.

III.5.1.2 ARES

Le module ARES est un outil d'édition et de routage qui permet de réaliser le PCB de la carte à partir d'un circuit réalisé sur ISIS. De plus, il offre la possibilité de placer automatiquement les composants et réalise automatiquement le routage.

III.5.2 Simulation du module d'entrée

Après avoir conçu le circuit électrique du module d'entrée (en assemblant les différents composants électronique dont les valeurs et les références sont fixées au préalable), on passe à l'étape simulation dans laquelle on simule le fonctionnement de ce module. Pour cela, on génère une impulsion d'amplitude 24V par un générateur d'impulsion et on visualise le signal sur l'oscilloscope du logiciel Proteus. Cette impulsion doit avoir la même période que l'impulsion d'entrée mais d'amplitude différente qui est 5V, (signal supporté par l'Arduino). Cette simulation est illustrée par la figure ci dessous :

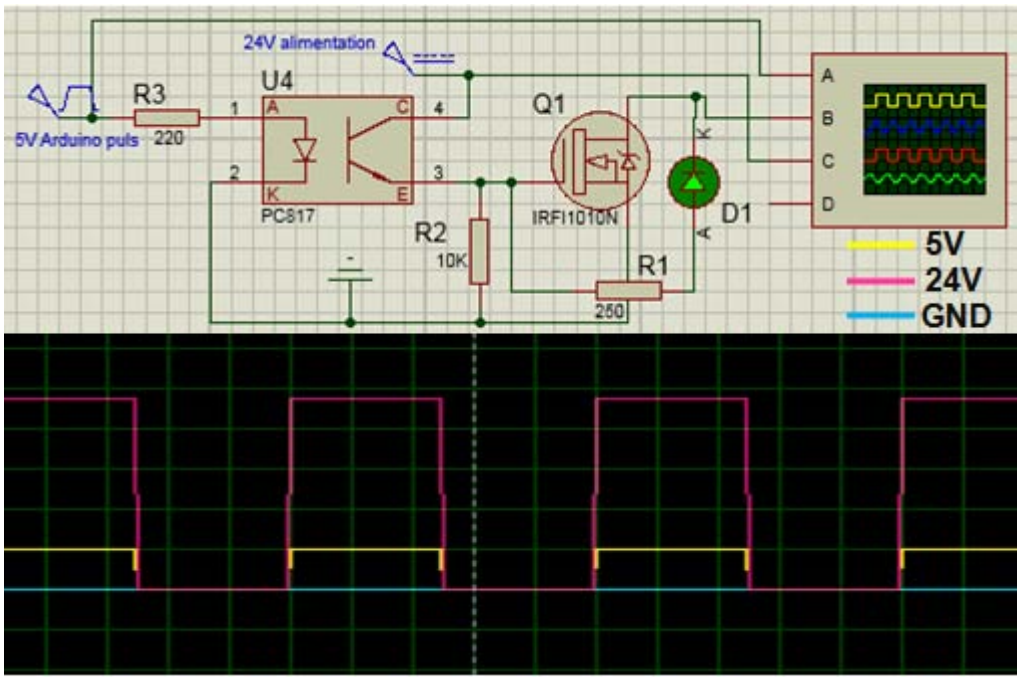


III.7 : Simulation du module d'entrée sur proteus

III.5.3 Simulation du module de sortie à transistor de puissance

Avant d'entamer la réalisation pratique du module de sortie, on passe par l'étape simulation du circuit électrique réalisé sur ISIS de proteus. La simulation de ce module suit les étapes suivantes :

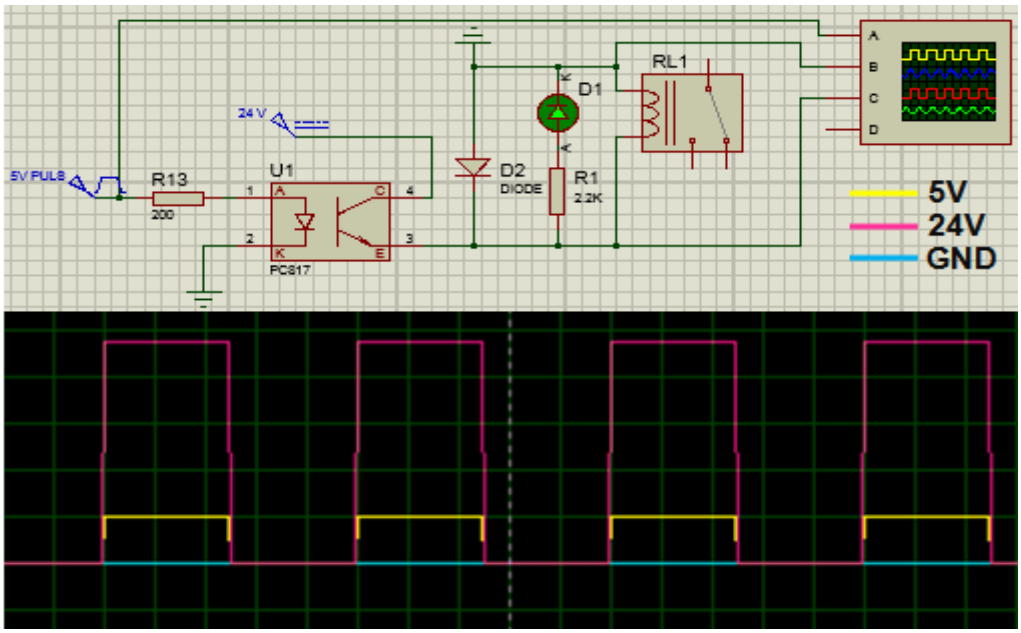
On commence par générer une impulsion d'amplitude 5V (signal émis par l'Arduino), puis on visualise le signal de sortie à l'aide de l'oscillateur intégré dans ce logiciel (Voir figure III.8). Cette impulsion doit avoir la même période que le signal d'entrée et une amplitude de 24V. (Signal permettant d'activer une électrovanne).



III.8 : Simulation du module de sortie à transistor de puissance sur proteus

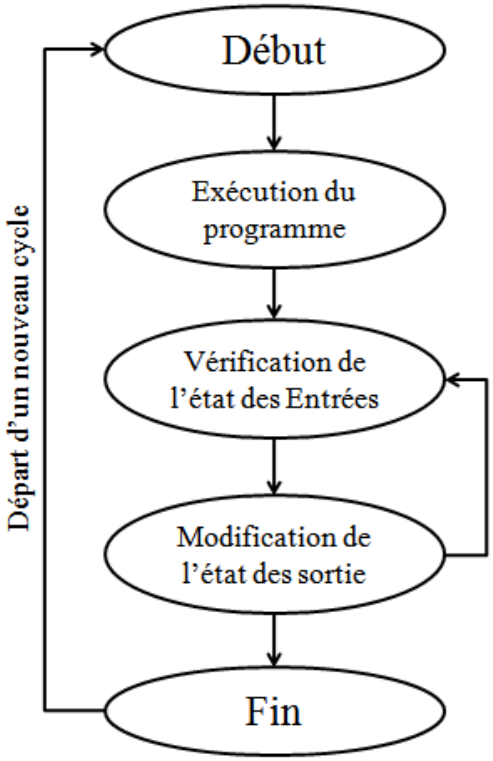
III.5.4 Simulation du module de sortie à relais

Comme il a été mentionné précédemment dans la simulation du module de sortie à transistor, on procède de la même manière pour le module à relais. Dans ce dernier, on visualise le signal qui active la bobine du relais correspondant à un signal d'amplitude 24V.



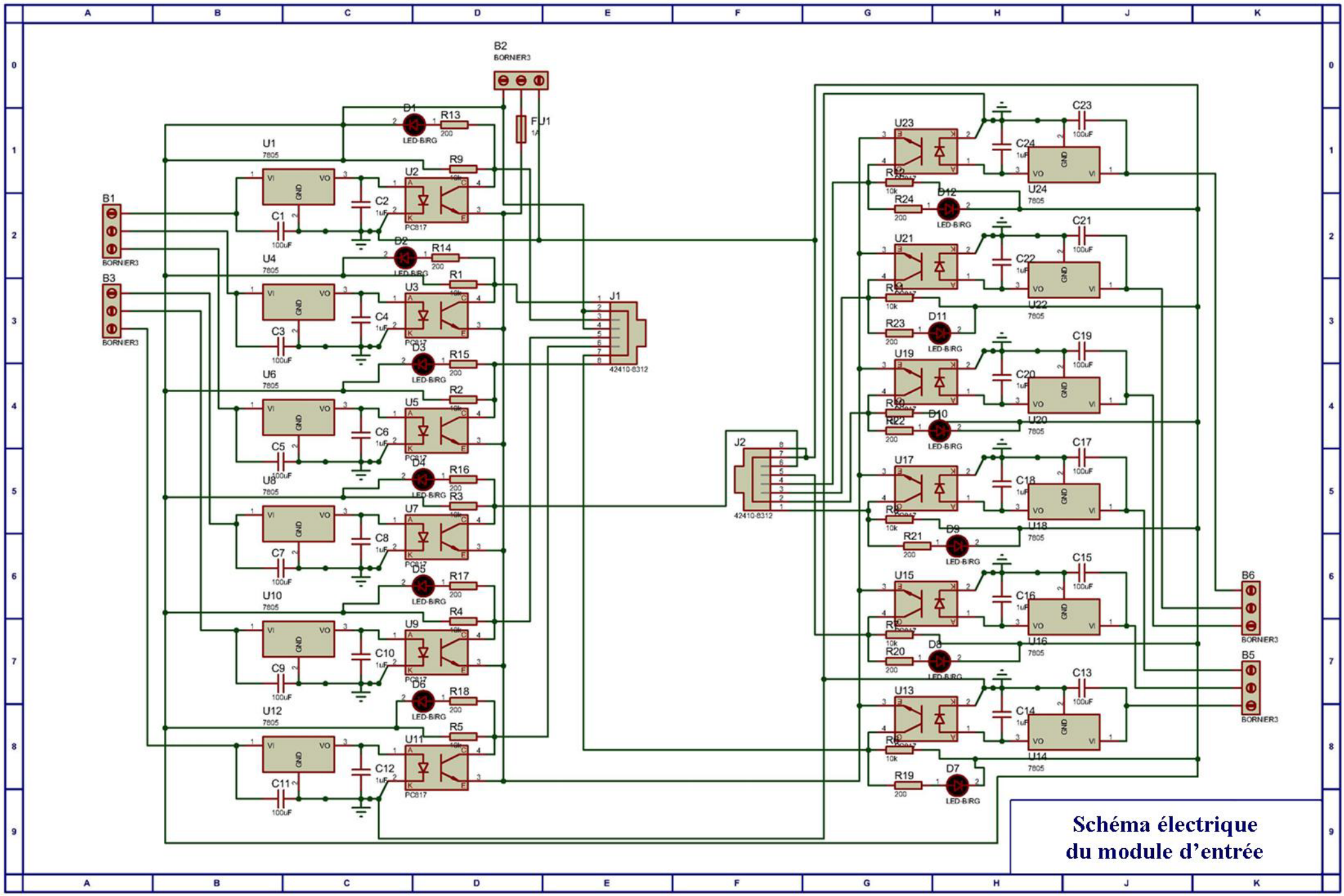
III.9 : Simulation du module de sortie à relais sur proteus

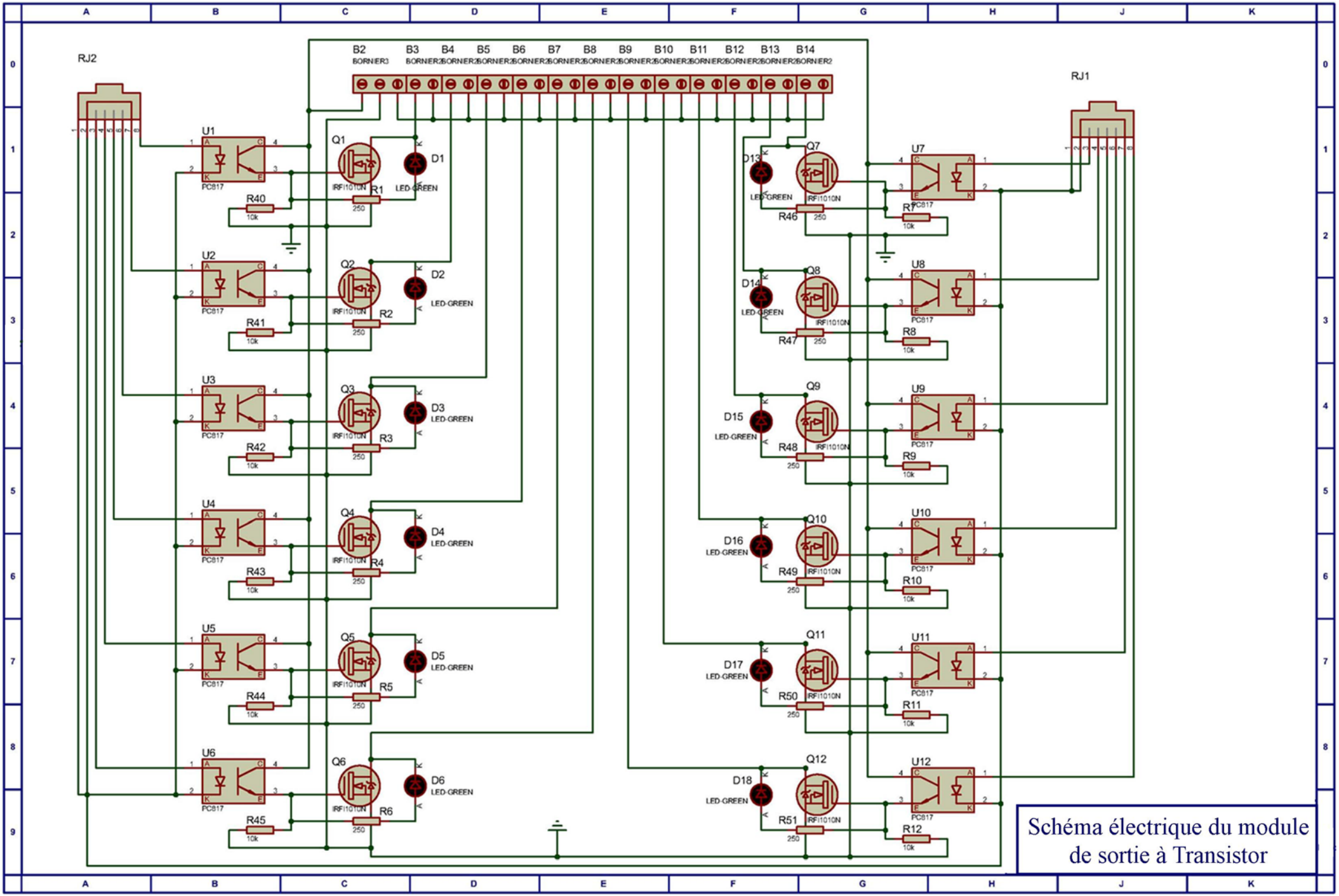
III.6 Organigramme de fonctionnement



III.7 Schémas électriques des différents modules

Les schémas électriques des différents modules sont illustrés par les figures suivantes :


B2
BORNIER3
F J1
1A
D1
R13
200
LED-BIRG
U1
7805
R9
10k
U2
PC817
C2
1uF
C1
100uF
B1
BORNIER3
D2
R14
200
U4
7805
R1
U3
C4
C3
B3
BORNIER3
D3
R15
U6
7805
R2
U5
C6
C5
U8
7805
D4
R16
R3
U7
C8
C7
D5
R17
U10
7805
R4
U9
C10
C9
D6
R18
U12
7805
R5
U11
C12
C11
J1
42410-8312
J2
42410-8312
U23
C23
100uF
C24
1uF
GND
U24
7805
R24
D12
LED-BIRG
U21
C21
C22
U22
R23
D11
U19
C19
C20
U20
R22
D10
U17
C17
C18
U18
R21
D9
U15
C15
C16
U16
R20
D8
U13
C13
C14
U14
R19
D7
B6
BORNIER3
B5
BORNIER3
Schéma électrique
du module d'entrée


RJ2
RJ1
B2 B3 B4 B5 B6 B7 B8 B9 B10 B11 B12 B13 B14
U1 U2 U3 U4 U5 U6
U7 U8 U9 U10 U11 U12
PC817
Q1 Q2 Q3 Q4 Q5 Q6
Q7 Q8 Q9 Q10 Q11 Q12
IRFI1010N
D1 D2 D3 D4 D5 D6
D13 D14 D15 D16 D17 D18
LED-GREEN
R1 R2 R3 R4 R5 R6
R46 R47 R48 R49 R50 R51
R40 R41 R42 R43 R44 R45
R7 R8 R9 R10 R11 R12
250
10k
Schéma électrique du module de sortie à Transistor

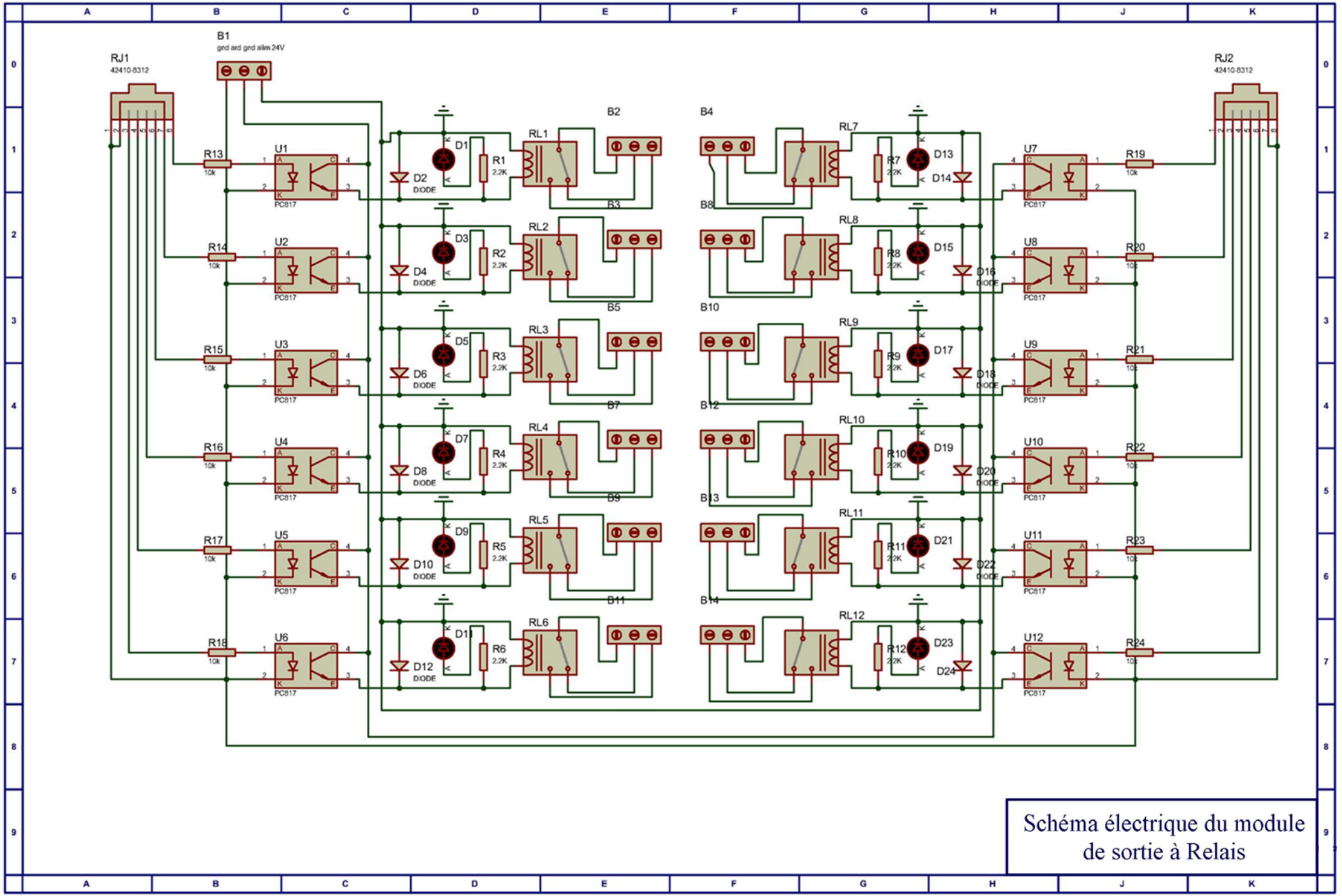


Schéma électrique du module de sortie à Relais

III.8 Conclusion

Dans ce chapitre, nous avons développé une étude théorique des différents modules constituant notre automate. Nous avons présenté les différents schémas de conception et de simulations avec Proteus ainsi que le schéma électrique de chaque module.

Le chapitre suivant sera consacré à la réalisation pratique de notre API.

Chapitre IV

Test sur lab d’essai et réalisation

IV.1 Introduction

Après l'étude théorique des différents modules de notre système, nous passons aux tests sur lab d'essai puis à la réalisation de notre API. Dans la première partie, nous allons donner les photos des différents blocs de notre réalisation sur des Labs d'essais. Ensuite, les différents typons ainsi que les schémas d'implantation seront donnés.

IV.2 Test des différents circuits sur labd'essai

Avant de procéder au routage et à l'impression des typons, on commence par le test du fonctionnement des différents modules sur Lab d'essai.Pour cela, on implante les composants constituants chaque module sur le Lab d'essai, puis on effectue les liaisons entre les composants en suivant le schéma de simulation présent dans le chapitre précédent.

Le module d'entrée est donné par la figure ci-dessous :

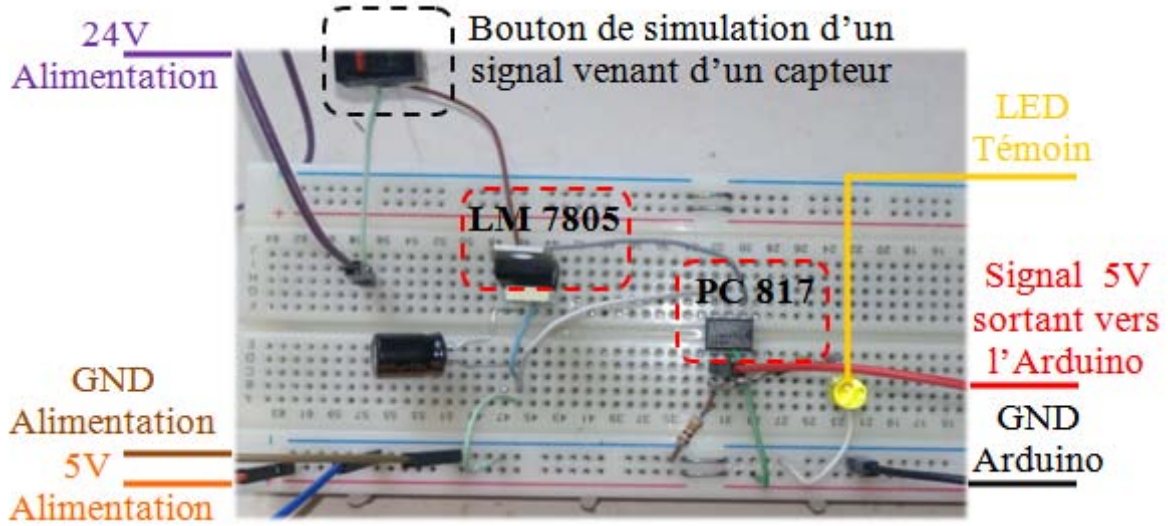


Figure IV.1 : Module d'entrée

La figure suivante montre le module de sortie :

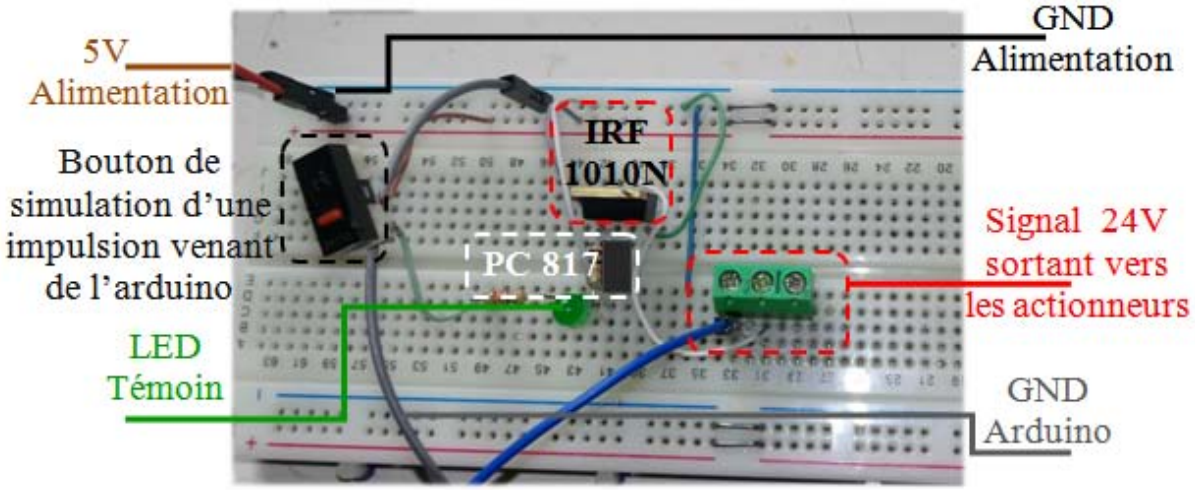


Figure IV.2 : Module de sortie

Le module à relais est illustré dans la figure ci-dessous :

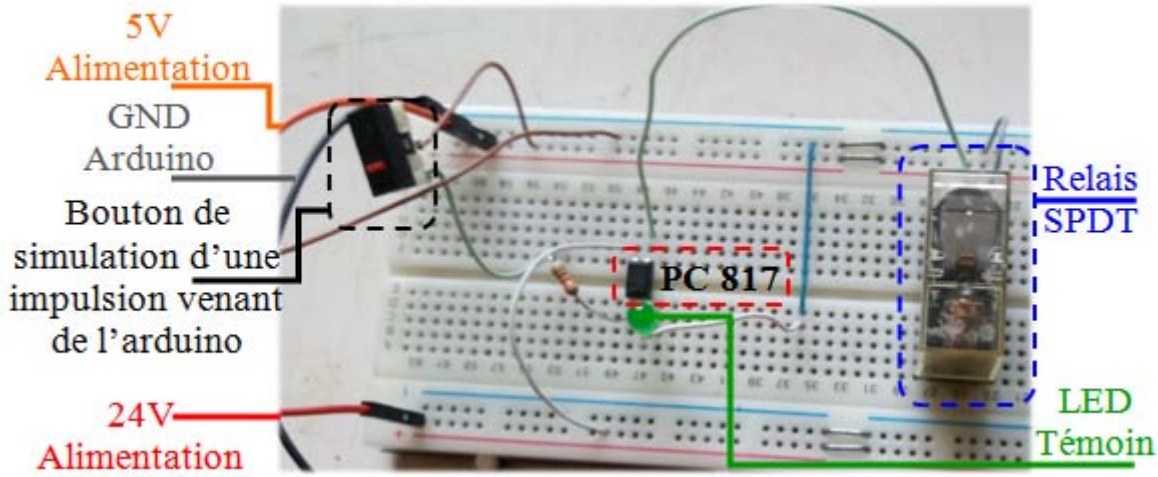


Figure IV.3 : Module à relais

IV.3 Fabrication des circuits imprimés

Après les tests sur lab d'essai, on passe à la réalisation du circuit imprimé qui est en général une plaque permettant de relier électriquement un ensemble de composants électroniques entre eux. Ce circuit imprimé ne peut être réalisé que si on possède un typon. Ce dernier est un dessin des pistes et des pastilles réalisés à l'aide du module ARES de Proteus, puis imprimé sur un support transparent (calque), il est utilisé à la première étape de la fabrication du circuit imprimé.La figure ci-dessous représente le typon du module CPU.

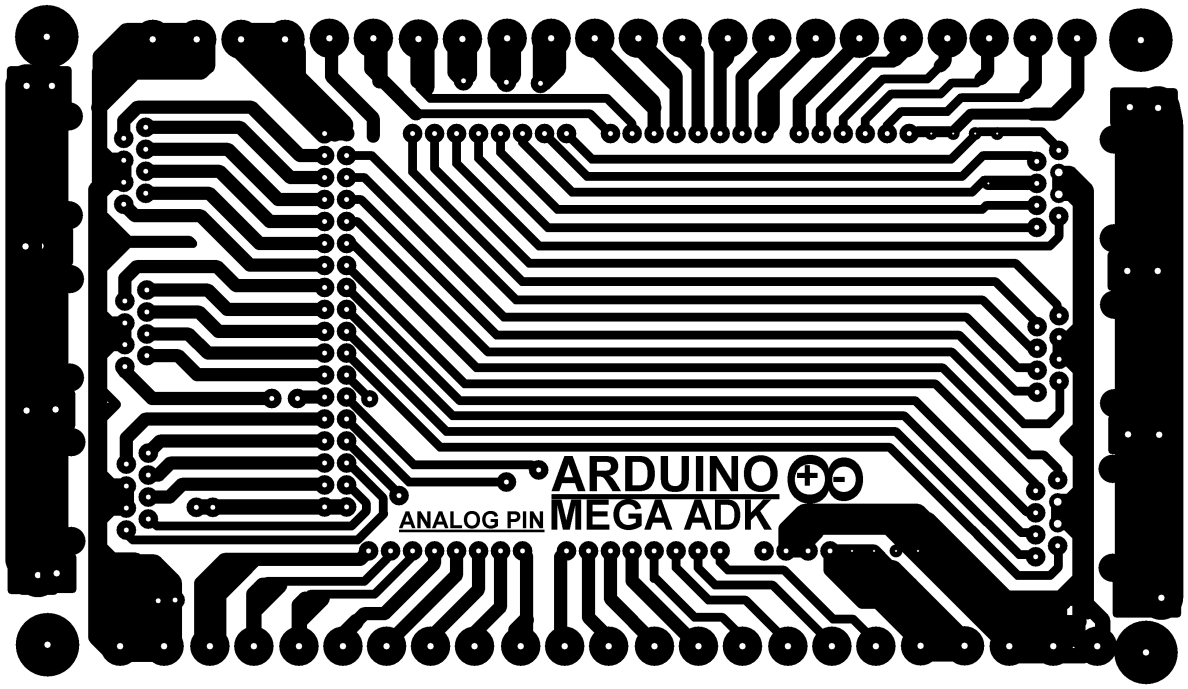


Figure IV.4 : Typon module CPU

Le typon du module d'entrée est donné ci-dessous :

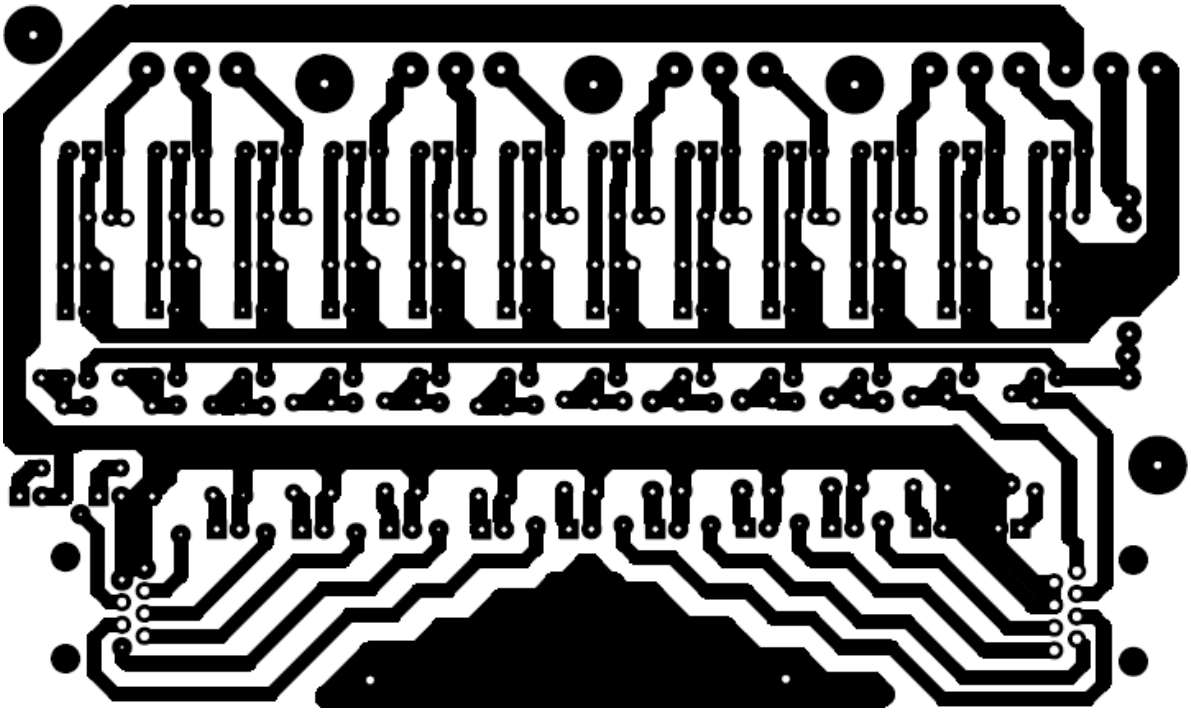


Figure IV.5 : Typon module entrée

Le typon du module de sortie est donné ci-dessous :

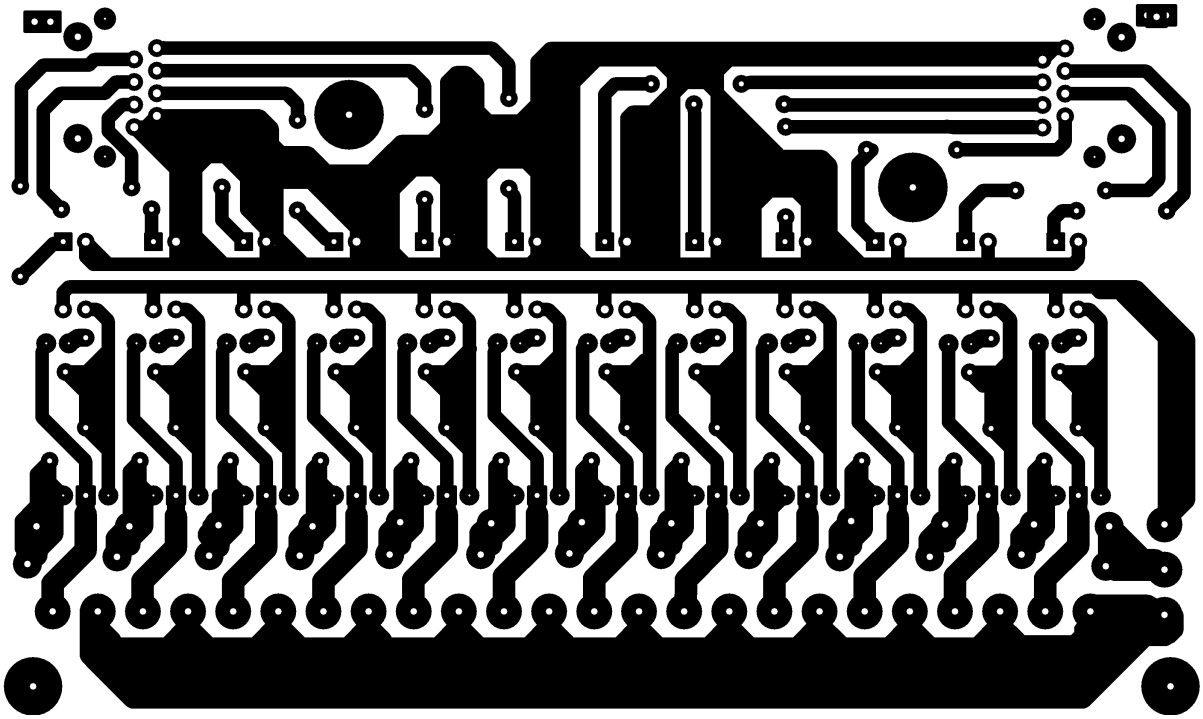


Figure IV.6 : Typon module sortie à transistor

Le typon du module à relais est donné ci-dessous :

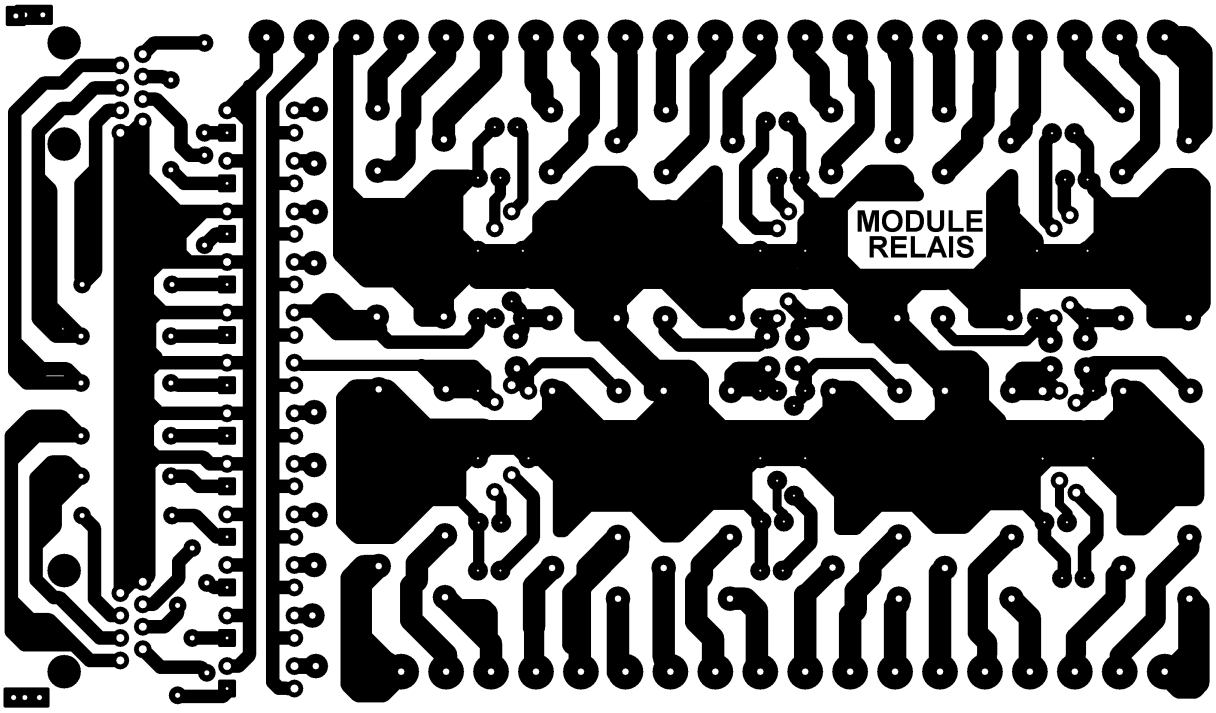


Figure IV.7 : Typon module sortie à relais

IV.3.1 Les étapes de la fabrication des plaques

- **Insolation**

On place le typon sur la plaque pré sensibilisée au contact de la résine photosensible. L'ensemble est placé dans une insoleuse UV pour une exposition pendant quelques minutes à une lumière ultraviolette.

- **la révélation**

En second lieu, on met la plaque insolée dans un bac contenant du révélateur. Les zones de résine fragilisées par la lumière ultraviolette seront détruites.

- **la gravure**

Après l'étape révélation, on place la plaque dans un bac contenant du perchlorure de fer qui va dissoudre les parties de la couche de cuivre non protégées par la résine.

IV.3.2 Implantation des composants

Une fois les plaques sont prêtes, on passe au perçage sur les pastilles présentes dans chaque module. Pour pouvoir ensuite implanter les composantsen suivant les schémas donnés par les figures ci-dessous, et on effectue les soudures.

Pour éviter tout problème, il est nécessaire de procéder à une vérification de l'orientation des composants polarisés et support.

La figure ci dessous représente le schéma d'implantation de la CPU :

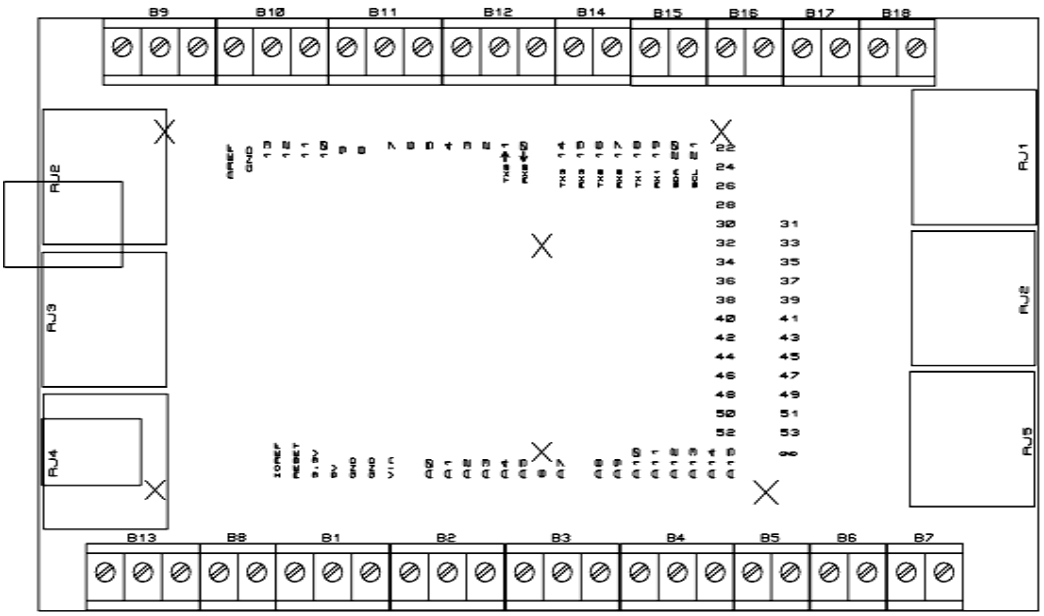


Figure IV.8 : Schéma d'implantation de la CPU

La figure ci dessous représente le schéma d'implantation du module d'entrée :

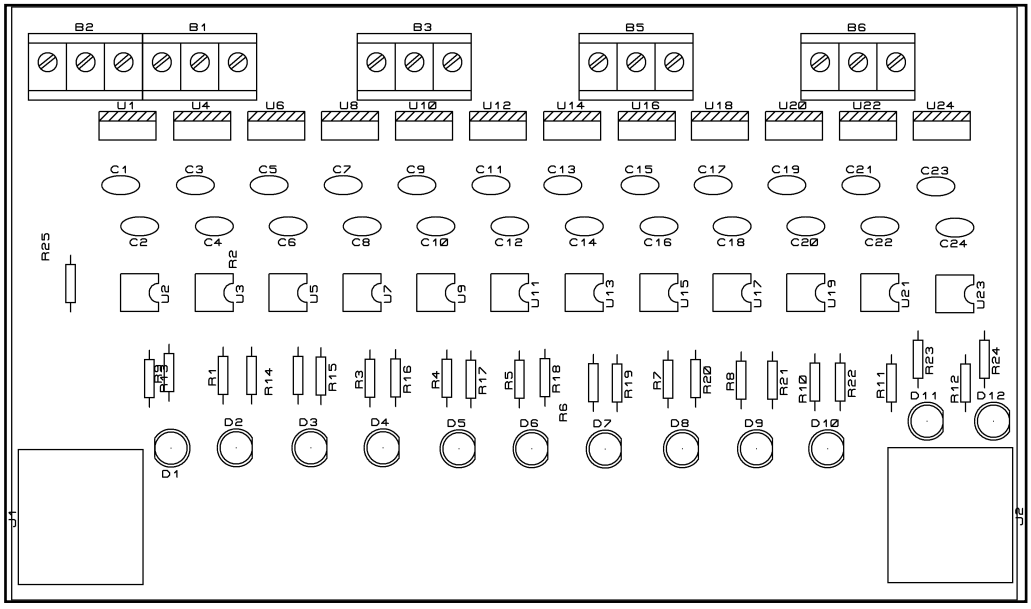


Figure IV.9 : Schéma d'implantation du module d'entrée

La figure ci dessous représente le schéma d'implantation du module de sortie :

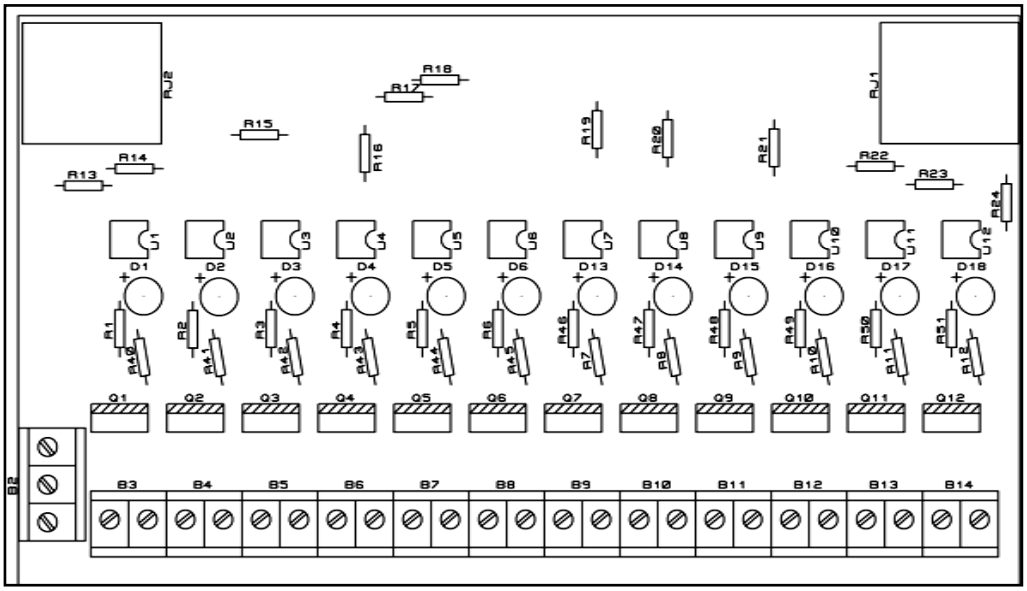


Figure IV.10 : Schéma d'implantation du module de sortie

La figure ci dessous représente le schéma d'implantation du module à relais:

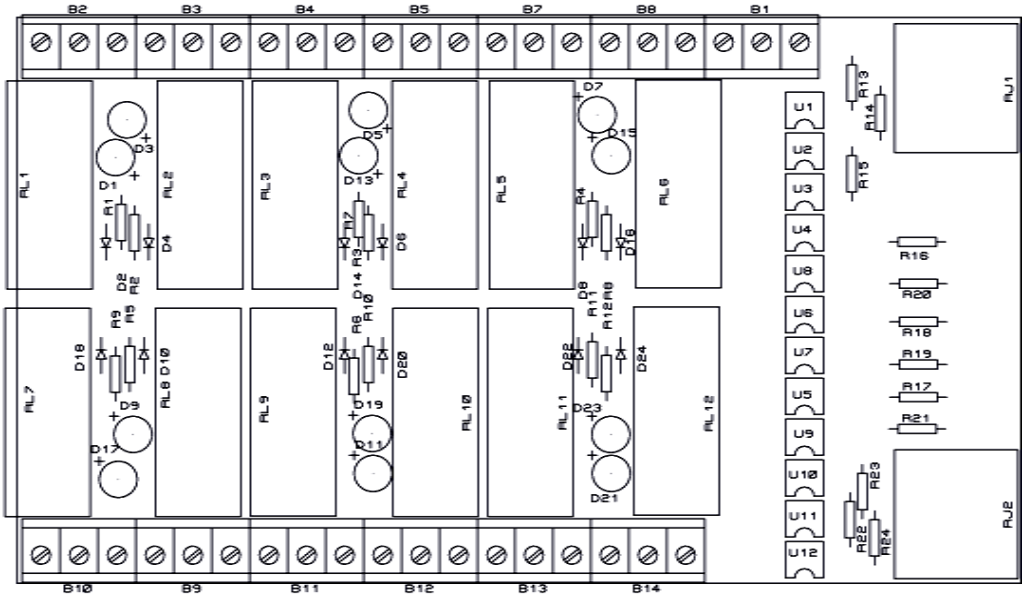


Figure IV.11 : Schéma d'implantions du module à relais

IV.4Concrétisation de l'API

Une fois les plaques sont prêtes, on passe au perçage sur les pastilles présentes dans chaque module. Pour pouvoir ensuite implanter les composants, et effectuer les soudures.

IV.4.1Carte de la CPU

La photo de la carte de la CPU est donnée ci dessous :

Figure IV.12 : Photo de la carte de la CPU

Les composants constituants ce module sont :

- Carte Arduino Méga ADK 2560.
- Des prises RJ45.
- Des borniers à vis.
- Barrette de contact pour l'Arduino.
- Radiateursventilé de refroidissement.

IV.4.2 Carte d'alimentation

La photo de la carte d'alimentation est donnée ci dessous :

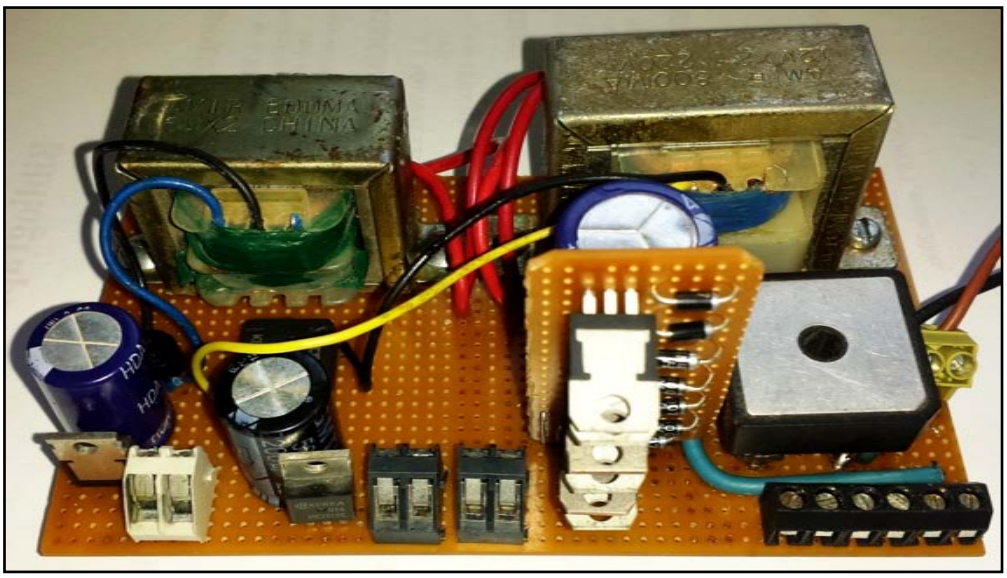


Figure IV.13 : Photo de la carte d'alimentation.

La carte d'alimentation est constituée des éléments suivants :

- Des transformateurs (24v, 12v, 5v).
- Des ponts de diodes pour le redressement.
- Des condensateurs de filtrage.
- Des régulateurs (LM7805, LM7812, LM7824).
- Des borniers à vis.

IV.4.3 Module d'entrée

La photo de la carte d'entrée est donnée ci dessous :

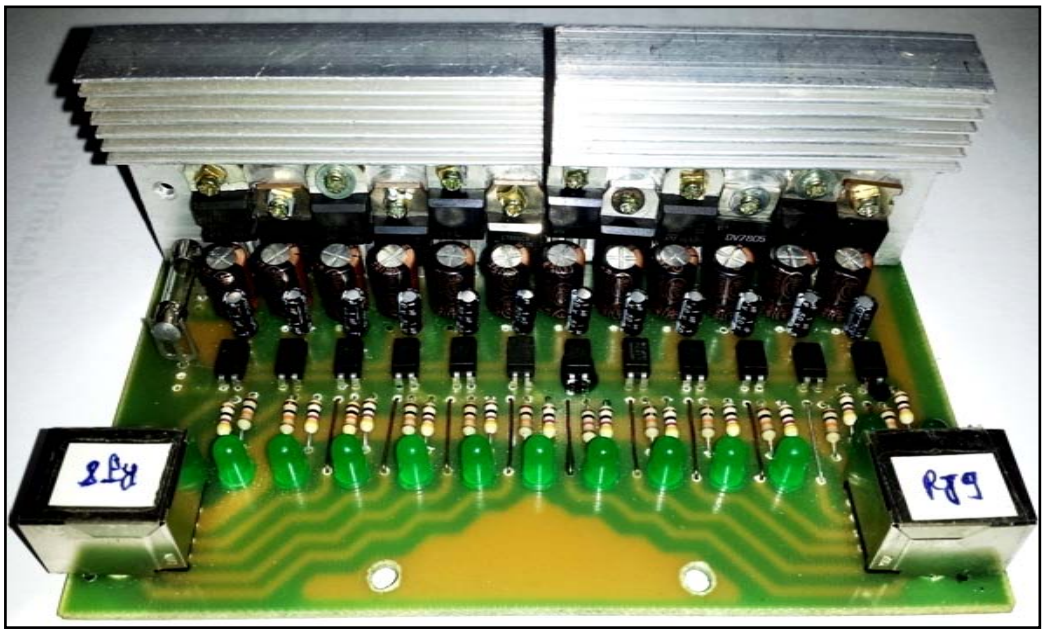


Figure IV.14 : Photo du module d'entrée

Le module d'entrée est constitué des éléments suivants :

- Régulateurs (LM7805).
- Photocoupleurs (PC817).
- Condensateurs (100µf, 1µf).
- Résistances (10KΩ ,200Ω).
- Ledstémoin.
- Prises RJ45.
- Radiateurs de refroidissement.
- Borniers à vis

IV.4.4 Module de sortie

La photo de la carte de sortie est donnée ci dessous :

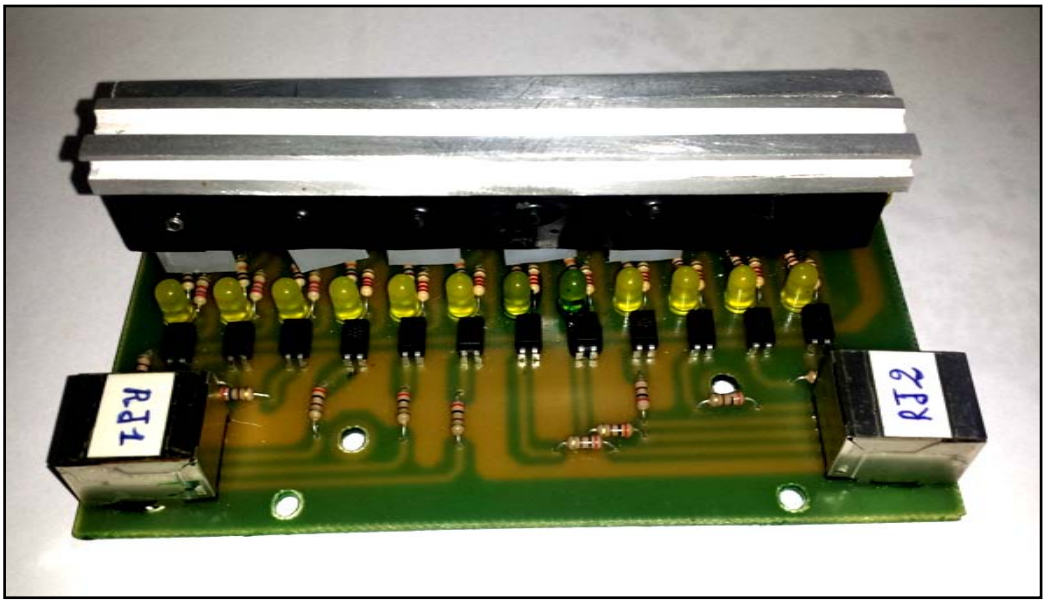


Figure IV.15 : Photo du module de sortie

Les composants constituant ce module sont les suivants :

- Transistors de puissance (IRF1010N)
- Photocoupleurs (PC817)
- Résistances (10KΩ ,200Ω ,2.2Ω).
- Ledstémoin.
- Prises RJ45.
- Radiateurs de refroidissement.
- Borniers à vis

IV.4.5Module de sortie à relais

La photo de la carte du module à relais est donnée ci dessous :

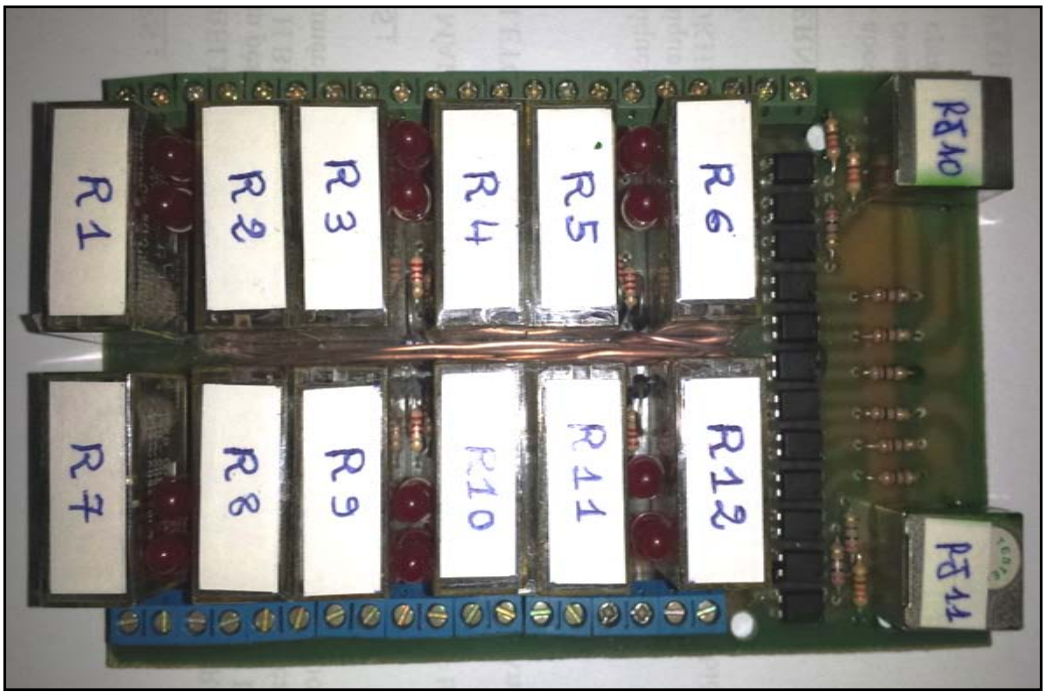


Figure IV.16 : Photos du module à relais

Les composants constituants la carte relais sont les suivants :

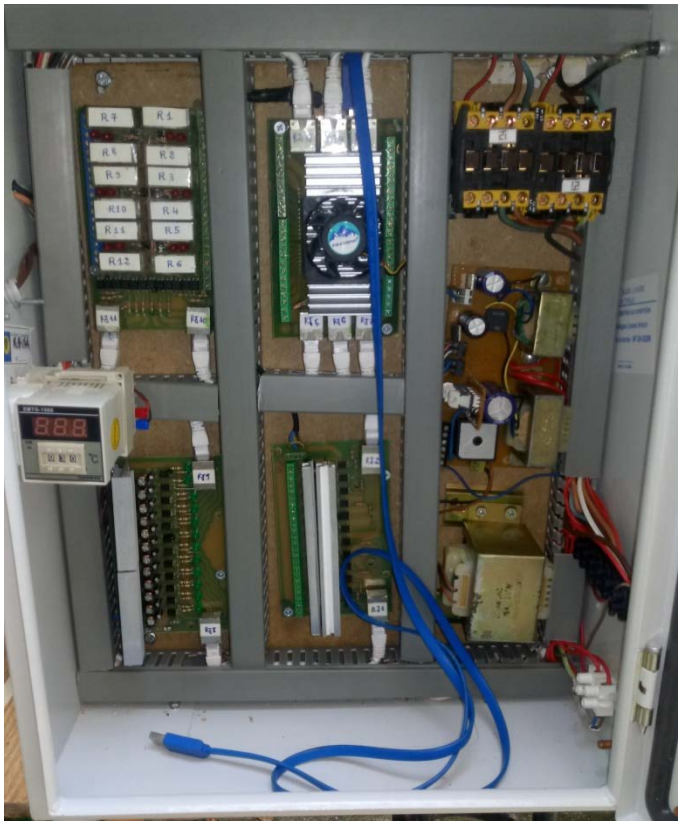
- Relais de type SPDT.
- Photocoupleurs (PC817)
- Des prises RJ45.
- Des diodes.
- Des résistances (2.2kΩ,200Ω).
- Des leds témoin.
- Borniers à vis.

IV.5 Tableau comparatif

Dans le tableau suivant, on a fait une comparaison entre l'API qu'on à réalisé et l'API S7-300 de siemens :

	Siemens S7-300	**Notre API**
Fréquence d'horloge	1 MHz	16 MHz
RAM	Optionnelle (de 8Kbit à 2 Mo) Non extensible	8 Kbit extensible
Mémoire flash	Optionnelle	256 Kbit extensible
EEPROM	20 Ko	4 Kbit extensible
Tension de fonctionnement de la CPU	20.4v à 28.8v DC	7v à 12v DC
Logiciel de programmation	Step 7	-Arduino IDE -Appareil androïde -plusieurs autres logiciels
Nombre d'entrées	Dépend du module	-16 entrées analogiques -12 entrées numériques extensibles
Nombre de sorties	Dépend du module	-12 sorties relais -12 sorties analogiques extensibles
Signal d'entrée	Une seule tension qui dépend du type de module.	-De 5v à 24v DC
Signal de sortie	Une seule tension qui dépend du type de module.	-De 0v à 120v DC -De 0v à 220v AC

IV.6 L'API et l'application finale :

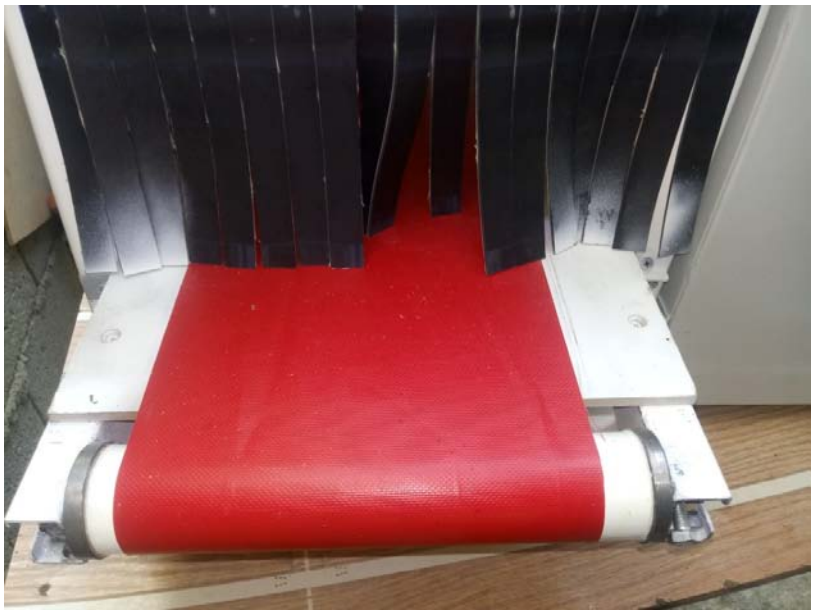


Installation de l'API dans l'armoire

Le procédé piloté par l'automate s'exécute comme suit :

On pose le paquet sur le tapis, puis on appuis sur le bouton départ cycle, le moteur se met en marche pour déplacer le paquet a l'intérieur du tunnel.

Il s'arrête pour permettre au film plastique de se rétracté pendant un laps de temps, défini dans le programme, en suite le moteur se met en marche pour évacuer la pièce finie du tunel.



IV.7 Conclusion

Dans ce chapitre, nous avons effectué un test afin de vérifier le bon fonctionnement et les performancesde chaque module. Ensuite, nous avons abordé les différentes phases d'élaboration de notre API.

En ce qui concerne notre montage, nous pouvons remarquer bien sa simplicité et sa facilité de mise en œuvre. Ainsi, nous tenons à préciser que notre API possède les mêmes caractéristiques que les automates de siemens, et il présente beaucoup d'avantages :

- Faible coût par rapport aux automates disponible sur le marché.
- Bonne stabilité.
- Grande précision.

Conclusion générale

Conclusion générale

Au cours de ce projet de fin d'études, nous avons pu concrétiser notre idée qui consiste à réaliser un automate programmable industriel à base d'une carte Arduino Méga (ADK).

De nos jours, l'utilisation des API est généralisée en industrie, en domotique, en automatisme et dans les systèmes embarqués.Les exigences de production, ont fait des automates des éléments essentiels des chaines automatisées, surtout en milieu industriel. L'automate que nous avons conçu et réalisé nous a permis de comprendre plus finement les concepts de l'automatisation.

Les conceptions des schémas électroniques ont été faites et simulées sur le logiciel Proteus avant la mise en œuvre. Ensuite, nous avons validé ce travail par des tests sur Labd'essai.

Malgré les difficultés rencontrées lors de la réalisation de ce modeste projet, le résultat final est un succès étant donné que l'objectif visé est atteint.

En perspective, ce projet de conception peut éventuellement être élargi en terme d'application, et ce, en ajoutant quelques options pour la maquette. A titre d'exemple, nous proposons les améliorations suivantes:

- ✓ Ajouter d'autres cartes Arduino branchées en esclave, ce qui permettra d'ajouter d'autres modules d'E/S.
- ✓ Utiliser des plaques pré-sensibilisées doubles couches dans la réalisation des différents modules pour minimiser l'espace occupé par l'API.
- ✓ Créer une application androïde spécialisée dans la programmation et le contrôle de l'API en utilisant la fonction (ADK).
- ✓ Ajouter une commande a distance.
- ✓ Créer un logiciel de programmation spéciale pour Arduino qui va la programmer en LADER.
- ✓ Ajouter un module d'entrée pour les capteurs en 4-20 mA.
- ✓ Visualisation des processus commandé par l'automate via LabVIEW.

Enfin, nous espérons que ce modeste travail, servira de document appréciable pour les promotions à venir.

Références bibliographiques

Références bibliographiques

[1] : W. Bolton ; « Les automates programmables industriels », Edition Dunod, 2010**.**

[2] : G. Michel, « Les API. Architectures et applications des automates programmables industriels », Edition Dunod, 1988.

[3] : S. Ben khemou , « Etude et réalisation d'un automate programmable industriel à base d'un pic 16F84 », mémoire d'Ingénieur, Département d'électronique, UMMTO, 2004.

[4] : H. Oubabas, A.Kader, A. Sersour, « Etude et réalisation d'un automate programmable industriel conçu à base d'un microcontrôleur pic 16F84 », mémoire d'Ingénieur, Département automatique, UMMTO, 2002.

[5] : www. Techno logune pro.com, consulté le 20/04/2018.

[6] : Erik Bartman : « Le grand livre Arduino 2ème édition », Edition Eyrolles 2015.

[7]: M. Banzi, M. Shilah « Getting started with Arduino 3ème edition», Edition Maker Media, Décembre 2014.

[8] : www.arduino.cc. Consulté le 27/05/2018.

[9] :www.mon-club-elec.fr consulté le 18/05/2018.

[10] : C.Tavernier, « Arduino, maitrisez sa programmation et ses cartes d'interfaces (Schields) », Edition Dunod, mars 2014.

[11] : Guide d'utilisateur du logiciel proteus, consulté le 12/06/2018**.**

ANNEXES

ANNEXE N° 01 Programme de l'API

```
// 12 Entrées de E1 à E12
const int E1 = 35 ;    const int E7 = 47  ;
const int E2 = 37 ;    const int E8 = 49  ;
const int E3 = 39 ;    const int E9 = 51  ;
const int E4 = 41 ;    const int E10 = 53 ;
const int E5 = 43 ;    const int E11 = 52 ;
const int E6 = 48 ;    const int E12 = 50 ;

// 12 Sorties à transistors de S1 à S12
const int S1 = 14 ;    const int S7 = 20  ;
const int S2 = 15 ;    const int S8 = 21  ;
const int S3 = 16 ;    const int S9 = 22  ;
const int S4 = 17 ;    const int S10 = 24 ;
const int S5 = 18 ;    const int S11 = 26 ;
const int S6 = 19 ;    const int S12 = 28 ;

// 12 Sorties à Relais de R0 à R12
const int R1 = 30 ;    const int R7 = 31  ;
const int R2 = 32 ;    const int R8 = 25  ;
const int R3 = 34 ;    const int R9 = 29  ;
const int R4 = 36 ;    const int R10 = 27 ;
const int R5 = 38 ;    const int R11 = 23 ;
const int R6 = 33 ;    const int R12 = 40 ;

void setup() {
  pinMode(R1 , OUTPUT);  pinMode( R7 , OUTPUT);
  pinMode(R2 , OUTPUT);  pinMode( R8 , OUTPUT);
  pinMode(R3 , OUTPUT);  pinMode( R9 , OUTPUT);
  pinMode(R4 , OUTPUT);  pinMode(R10 , OUTPUT);
```

```
  pinMode(R5 , OUTPUT);  pinMode(R11 , OUTPUT);
  pinMode(R6 , OUTPUT);  pinMode(R12 , OUTPUT);

  pinMode(S1 , OUTPUT);  pinMode( S7 , OUTPUT);
  pinMode(S2 , OUTPUT);  pinMode( S8 , OUTPUT);
  pinMode(S3 , OUTPUT);  pinMode( S9 , OUTPUT);
  pinMode(S4 , OUTPUT);  pinMode(S10 , OUTPUT);
  pinMode(S5 , OUTPUT);  pinMode(S11 , OUTPUT);
  pinMode(S6 , OUTPUT);  pinMode(S12 , OUTPUT);

  pinMode(E1 , INPUT);  pinMode( E7 , INPUT);
  pinMode(E2 , INPUT);  pinMode( E8 , INPUT);
  pinMode(E3 , INPUT);  pinMode( E9 , INPUT);
  pinMode(E4 , INPUT);  pinMode(E10 , INPUT);
  pinMode(E5 , INPUT);  pinMode(E11 , INPUT);
  pinMode(56 , INPUT);  pinMode(E12 , INPUT);
}
void loop() {
}
```

I Les différents logiciels de programmation en langages littéraux

Il existe plusieurs logiciels de programmation qui utilisent les langages littéraux, ces logiciels de développement a langages de haut niveau sont utilisés pour faciliter la tâche aux utilisateurs des cartes Arduino, en simplifiant sa programmation, et parmi ces logiciels on trouve :

- Slober, l'IDE Arduino pour ECLIPSE.
- LARP Logiciel d'Algorithmes et de Résolution de Problèmes.
- AsmEditor est un Environnement de Développement Intégrés (E.D.I.) permettant la création et la compilation de projets en langage assembleur.

I.1Echantillon de logiciels de programmation en langages littéraux (Slober) d'Éclipse

I. 1.1 A propos de Slober

Slober (Software Logic to program Open Electronic Boards) IDE d'Eclipse une alternative à l'IDE Arduino, un environnement de développement utilisé par les développeurs professionnels et amateurs. C'est un logiciel open-source et extensible via des plugins. De nombreux développeurs ont contribué à son développement, y compris certains avec le soutien de l'entreprise. Ce logiciel fonctionne sur MS Windows, Mac OSX et Linux.

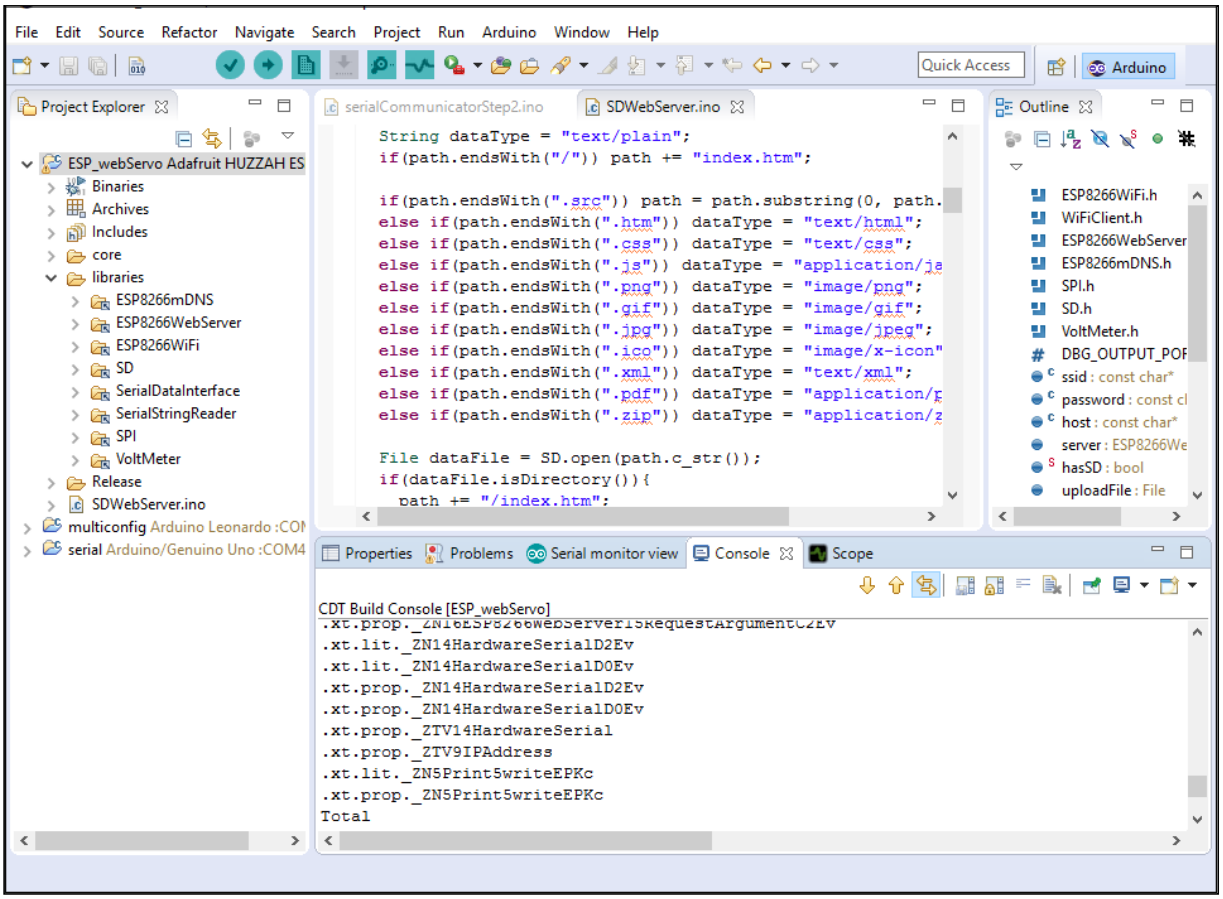
L'Eclipse IDE (environnement de développement intégré) est un éditeur de programmation complet avec de nombreuses fonctionnalités qui aident à coder plus rapidement et plus facilement. L'IDE Arduino est parfait pour ce qu'il fait - mais il ne fait pas grand-chose pour écrire, naviguer et comprendre le code. Slober comble cet écart et aide à évoluer vers un environnement de développement plus puissant tout en conservant le matériel et les bibliothèques Arduino.

Le développement Arduino basé sur Eclipse utilise deux ajouts à l'IDE Eclipse de base. L'un est l'outil de développement C / C ++ (CDT). Le CDT ajoute non seulement la capacité de développement C / C ++, mais aussi des outils pour l'achèvement et l'insertion automatique du code, ainsi que du refactoring de code.

L'autre ajout est un plug-in développé par [Jantje Baeyens]. Le plug-in est gratuit et open-source. Cette configuration fonctionne en combinaison avec l'environnement de construction de l'IDE Arduino.

I. 1.2 L'interface de Slober

Slober possède une interface de programmation complète avec de nombreuses fonctionnalités qui aident à coder plus rapidement et plus facilement. Grâce à ses nombreux outils et sa boite de dialogue ainsi que sa fenêtre graphique.



La fenêtre graphique de Slober

I.1. 3 L'utilité de Slober

Slober se caractérise par sa simplicité et par les ajouts apportés a l'environnement Arduino, par ses nombreux outils et sa boite de dialogue ainsi qu'une fenêtre graphique <<oscilloscope>>.Tout ce qui suit est ce que l'IDE apporte.

❖ **Onglets de l'éditeur**

Lorsque les projets prennent de l'ampleur, ils ont évidemment plus de lignes de code. Avoir des centaines ou des milliers de lignes de code dans un seul fichier. Faire défiler ce gros fichier pour trouver une seule ligne de code prend du temps. C'est pourquoi Slober

prend en charge le fractionnement du code en plusieurs fichiers. Passer d'une fenêtre d'édition à une autre est beaucoup plus facile que de faire défiler les fenêtres.

Il supporte environ 18 onglets. Les onglets pour les fichiers supplémentaires défilent vers la droite, ou à travers une liste déroulante sur la droite. Il permet également d'accéder à plusieurs projets en même temps. C'est utile pour obtenir des extraits de code pour un projet actuel d'un ancien, ou dans le cas ou il est utilisé deux cartes Arduino qui coopèrent entre elle.

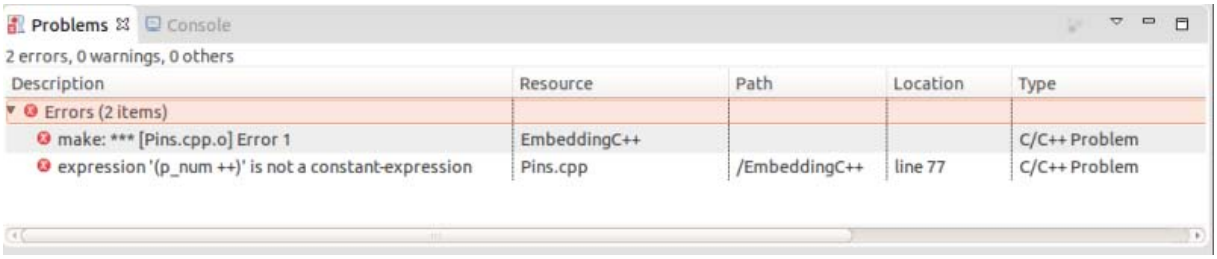
- **Vitesse de compilation**

Sous Eclipse, la construction ne déplace pas les fichiers. La chaîne d'outils reconnaît qu'une fois qu'un fichier est compilé, il n'a pas besoin d'être compilé à nouveau jusqu'à ce qu'une modification soit apportée à la source. Dans des projets commerciaux extrêmement importants, cela peut littéralement économiser des heures. Même dans les grands projets de passe-temps, les économies de temps peuvent être substantielles.

- **Détections des erreurs**

Eclipse signale des erreurs de deux manières. La première est une fenêtre de console similaire à celle de l'Arduino. La différence est que vous pouvez cliquer sur une erreur et être pris à la ligne de code. Eclipse ouvrira même le fichier s'il n'est pas actif dans un éditeur. Un économiseur de temps réel.

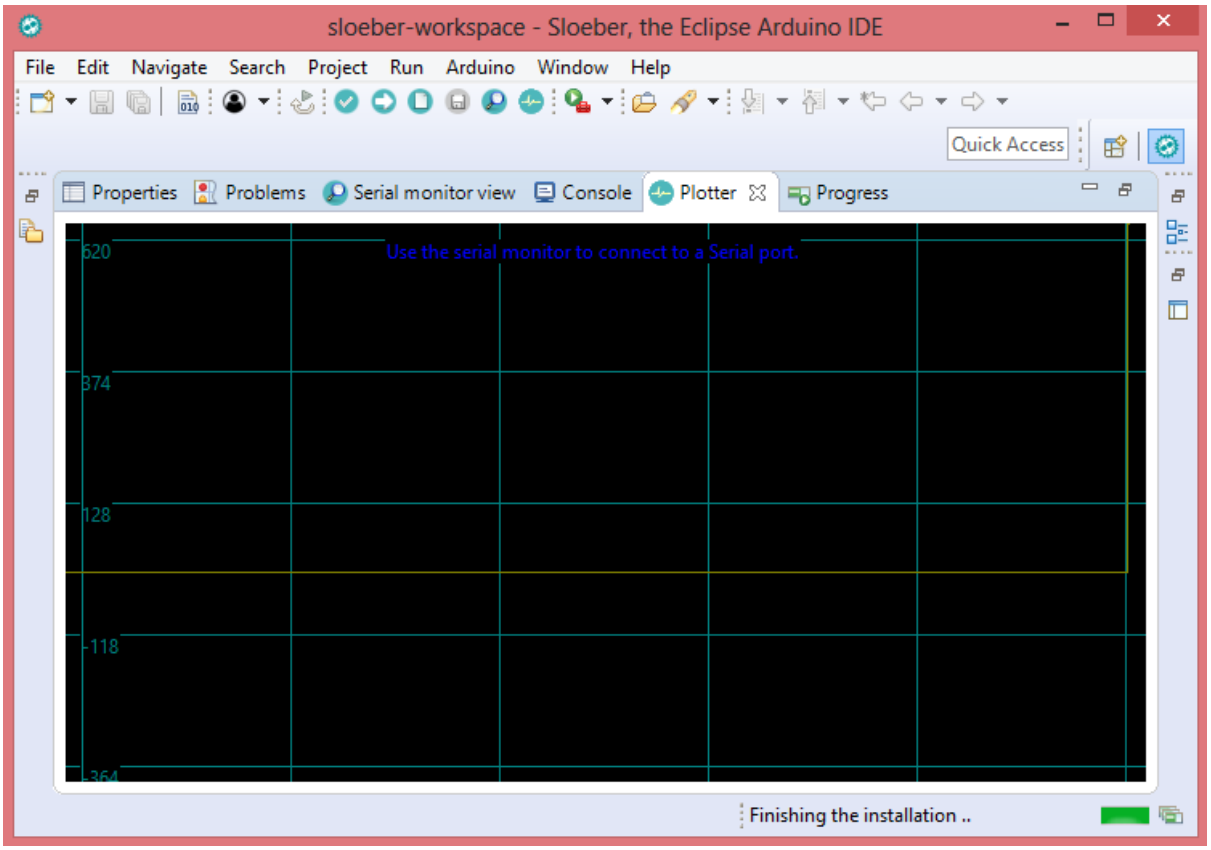
La seconde est une liste d'erreurs dans une fenêtre "Problèmes" . La lecture de cette liste est beaucoup plus rapide que la fenêtre de console d'IDE d'Arduino. Cette console est toujours importante car elle fournit les informations supplémentaires parfois nécessaires pour comprendre exactement ce qui cause le problème.



La fenêtre "Problèmes" de Slober

❖ **Fenêtre graphique**

Un ajout apporté par le plug-in est une fenêtre graphique «oscilloscope» qui affiche des données correctement formatées en tant que courbe .Utilisé pour voir comment les capteurs réagissent à l'environnement.



La fenêtre graphique «oscilloscope» de Slober

II Les logiciels de programmation a langage graphiques

Un outil de programmation graphique ou visuel est un logiciel de programmation dans lequel les programmes sont écrits par assemblage d'éléments graphiques. Sa syntaxe concrète est composée de symboles graphiques et de textes, qui sont disposés spatialement pour former des programmes. De nombreux langages visuels se basent sur les notions « de boîtes et de flèches » : les boîtes (ou d'autres d'objets) sont traités comme des entités, reliées par des flèches ou des lignes qui représentent des relations.

Plus précisément, un langage est défini par une syntaxe abstraite, à laquelle sont associées une ou plusieurs syntaxes concrètes, parmi lesquelles une ou plusieurs peuvent être graphiques.

Généralement ces langages sont associés à un environnement graphique de programmation. Il n'est pas toujours possible de les dissocier. Il faut également faire la distinction entre le langage au sens "normalisé" et son implémentation au sens "logiciel".

ANNEXE N°02 Autres logiciels de programmation

II.1 Les différents logiciels graphiques de programmation

Il existe plusieurs logiciels de programmation qui utilisent la solution graphique, ces logiciels de développement facilitent la tache aux utilisateurs des cartes Arduino en simplifiant sa programmation, et parmi ces logiciels on trouve :

- **Blockly** développé par Google à partir d'App Inventor.
- **Scratch** : implémentation libre et graphique du langage Smalltalk.
- **Simulink** : basé sur MATLAB et orienté sur les systèmes multi-physiques.
- **Grafcet / SFC.**
- **Ladder.**
- **Ardublock.**
- **MiniBloq.**
- **S4A** (Scratch for Arduino).

II.2 Exemple de logiciels graphiques de programmation (S4A)

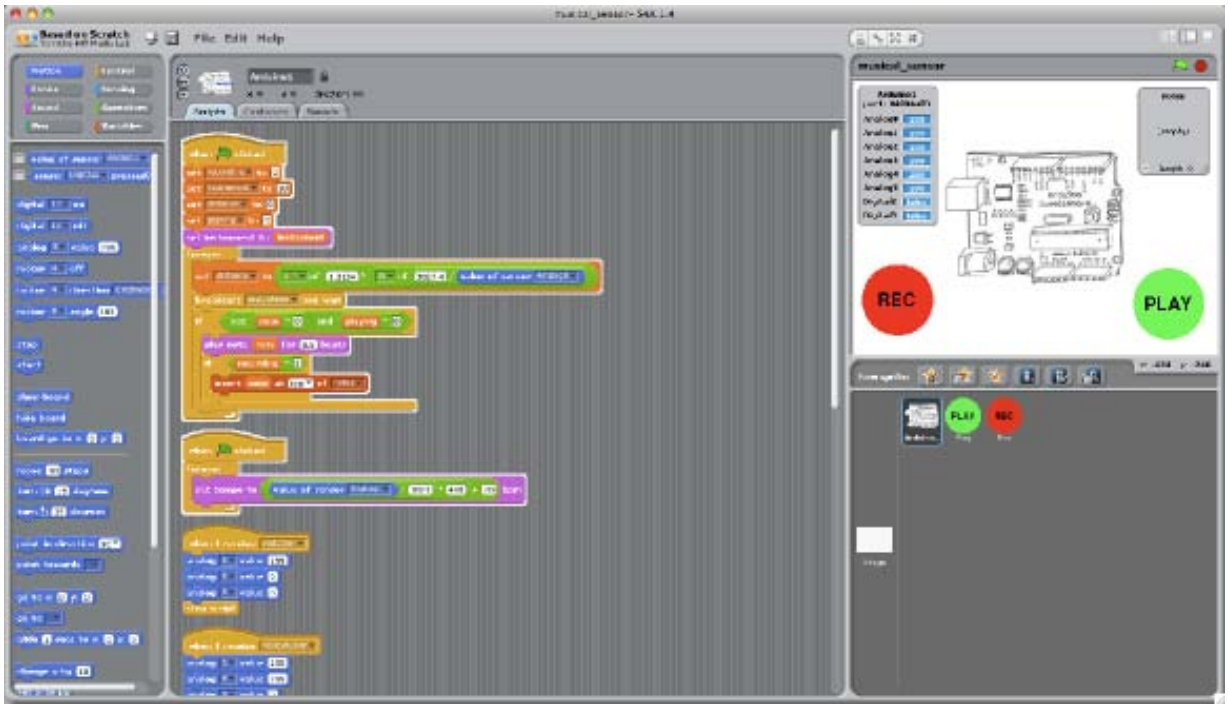
II.2.1 À propos de S4A

S4A est une modification de Scratch qui permet une programmation simple de la plate-forme matérielle open source Arduino. Il fournit de nouveaux blocs pour gérer les capteurs et les actionneurs connectés à Arduino. Il existe également un tableau de bord des capteurs similaire à celui du PicoBoard.

L'objectif principal du projet est d'attirer des gens dans le monde de la programmation. L'objectif est également de fournir une interface de haut niveau aux programmeurs Arduino avec des fonctionnalités telles que l'interaction avec un ensemble de cartes à travers des événements utilisateur.

II.2.2 L'interface de S4A

L’utilisateur de l’interface de Scratch for Arduino n’a pas à écrire la moindre ligne de code, ce qui rend ce logiciel particulièrement approprié par les novices en programmation, voire son interface.



L'interface de S4A

Les objets Arduino offrent des blocs pour les fonctionnalités de base du microcontrôleur, les écritures et lectures analogiques et numériques, ainsi que pour les niveaux supérieurs. Vous pouvez trouver des blocs pour gérer les servomoteurs à rotation standard et continue.

Quelques blocs de codage

En S4A, une carte Arduino est représentée sur l’interface. L'image-objet Arduino trouve automatiquement le port USB auquel la carte est connectée.

Il est possible de se connecter à plusieurs cartes en même temps en ajoutant simplement une nouvelle image-objet Arduino.

II.2.3 Les différentes briques de codage

Le langage Scratch est constitué de multiples briques permettant d'exécuter une action précise. Il existe 10 catégories de briques différentes (classées par couleurs).

Couleur										
Catégorie	Mouvement	Contrôle	Événements	Apparence	Capteurs	Sons	Opérateurs	Stylo	Données	Ajouter blocs

Les différentes briques de codage du S4A

ANNEXE N° 03 Les datasheets

I Les régulateurs (LM7805, LM7812, LM7824)

Les figures ci-dessous illustres les caractéristiques des régulateurs LM 7805, LM7812, LM7824

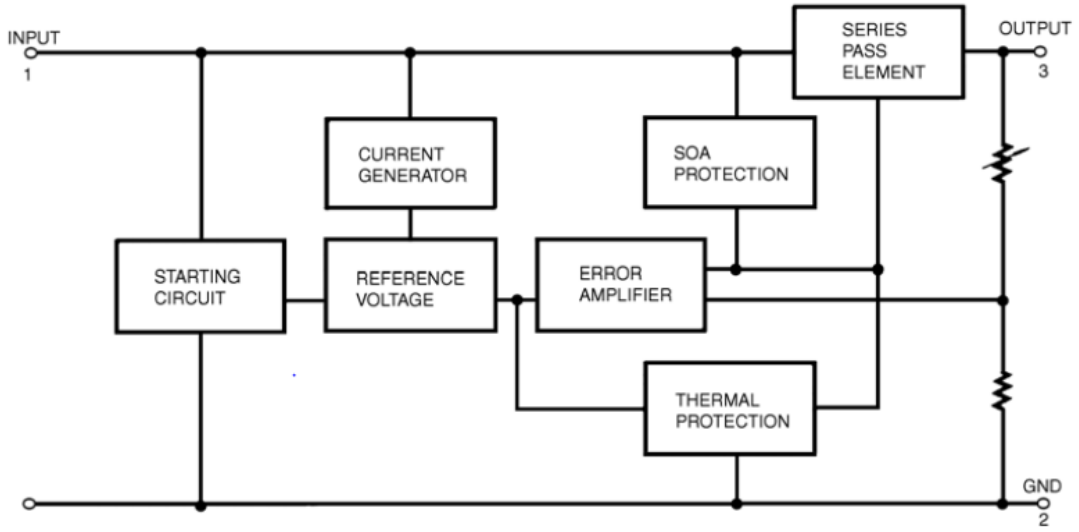


Schéma interne des régulateurs

Tableau des caractéristiques électriques du régulateur LM7805 :

Parameter	Symbol	Conditions		Min	Typ	Max	Unit
Output Voltage	V_O	T_J = +25°C		4.8	5.0	5.2	V
		5mA ≤ I_O ≤ 1A, P_O ≤ 15W, V_I = 7V to 20V		4.75	5.0	5.25	
Line Regulation (Note 2)	Regline	T_J = +25°C	V_O = 7V to 25V	–	4.0	100	mV
			V_I = 8V to 12V	–	1.6	50.0	
Load Regulation	Regload	T_J = +25°C	I_O = 5mA to 1.5mA	–	9.0	100	mV
			I_O = 250mA to 750mA	–	4.0	50.0	
Quiescent Current	I_Q	T_J = +25°C		–	5.0	8.0	mA
Quiescent Current Change	ΔI_Q	I_O = 5mA to 1A		–	0.03	0.5	mA
		V_I = 7V to 25V		–	0.3	1.3	
Output Voltage Drift (Note 3)	$\Delta V_O/\Delta T$	I_O = 5mA		–	-0.8	–	mV/°C
Output Noise Voltage	V_N	f = 10Hz to 100KHz, T_A = +25°C		–	42.0	–	µV/V_O
Ripple Rejection (Note 3)	RR	f = 120Hz, V_O = 8V to 18V		62.0	73.0	–	dB
Dropout Voltage	V_{DROP}	I_O = 1A, T_J = +25°C		–	2.0	–	V
Output Resistance (Note 3)	rO	f = 1KHz		–	15.0	–	mΩ
Short Circuit Current	I_{SC}	V_I = 35V, T_A = +25°C		–	230	–	mA
Peak Current (Note 3)	I_{PK}	T_J =+25°C		–	2.2	–	A

ANNEXE N° 03 Les datasheets

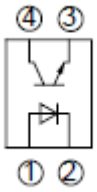
Tableau des caractéristiques électriques du régulateur LM7812 :

Parameter	Symbol	Conditions		Min	Typ	Max	Unit
Output Voltage	V_O	T_J = +25°C		11.5	12.0	12.5	V
		5mA ≤ I_O ≤ 1A, P_O ≤ 15W, V_I = 14.5V to 27V		11.4	12.0	12.6	
Line Regulation (Note 12)	Regline	T_J = +25°C	V_I = 14.5V to 30V	–	10.0	240	mV
			V_I = 16V to 22V	–	3.0	120	
Load Regulation (Note 12)	Regload	T_J = +25°C	I_O = 5mA to 1.5mA	–	11.0	240	mV
			I_O = 250mA to 750mA	–	5.0	120	
Quiescent Current	I_Q	T_J = +25°C		–	5.1	8.0	mA
Quiescent Current Change	ΔI_Q	I_O = 5mA to 1A		–	0.1	0.5	mA
		V_I = 14.5V to 30V		–	0.5	1.0	
Output Voltage Drift (Note 13)	$\Delta V_O/\Delta T$	I_O = 5mA		–	-1.0	–	mV/°C
Output Noise Voltage	V_N	f = 10Hz to 100KHz, T_A = +25°C		–	76.0	–	$\mu V/V_O$
Ripple Rejection (Note 13)	RR	f = 120Hz, V_I = 15V to 25V		55.0	71.0	–	dB
Dropout Voltage	V_{DROP}	I_O = 1A, T_J = +25°C		–	2.0	–	V
Output Resistance (Note 13)	rO	f = 1KHz		–	18.0	–	mΩ
Short Circuit Current	I_{SC}	V_I = 35V, T_A = +25°C		–	230	–	mA
Peak Current (Note 13)	I_{PK}	T_J = +25°C		–	2.2	–	A

Tableau des caractéristiques électriques du régulateur LM7824 :

Parameter	Symbol	Conditions		Min	Typ	Max	Unit
Output Voltage	V_O	T_J = +25°C		23.0	24.0	25.0	V
		5mA ≤ I_O ≤ 1A, P_O ≤ 15W, V_I = 27V to 38V		22.8	24.0	25.25	
Line Regulation (Note 18)	Regline	T_J = +25°C	V_I = 27V to 38V	–	17.0	480	mV
			V_I = 30V to 36V	–	6.0	240	
Load Regulation (Note 18)	Regload	T_J = +25°C	I_O = 5mA to 1.5mA	–	15.0	480	mV
			I_O = 250mA to 750mA	–	5.0	240	
Quiescent Current	I_Q	T_J = +25°C		–	5.2	8.0	mA
Quiescent Current Change	ΔI_Q	I_O = 5mA to 1A		–	0.1	0.5	mA
		V_I = 27V to 38V		–	0.5	1.0	
Output Voltage Drift (Note 19)	$\Delta V_O/\Delta T$	I_O = 5mA		–	-1.5	–	mV/°C
Output Noise Voltage	V_N	f = 10Hz to 100KHz, T_A = +25°C		–	60.0	–	$\mu V/V_O$
Ripple Rejection (Note 19)	RR	f = 120Hz, V_I = 28V to 38V		50.0	67.0	–	dB
Dropout Voltage	V_{DROP}	I_O = 1A, T_J = +25°C		–	2.0	–	V
Output Resistance (Note 19)	rO	f = 1KHz		–	28.0	–	mΩ
Short Circuit Current	I_{SC}	V_I = 35V, T_A = +25°C		–	230	–	mA
Peak Current (Note 19)	I_{PK}	T_J = +25°C		–	2.2	–	A

II Photocoupleur PC 817



① Anode
② Cathode
③ Emitter
④ Collector

Tableau des caractéristiques électriques du photocoupleur PC817:

Parameter		Symbol	Rating	Unit
Input	Forward current	I_F	50	mA
	*1Peak forward current	I_{FM}	1	A
	Reverse voltage	V_R	6	V
	Power dissipation	P	70	mW
Output	Collector-emitter voltage	V_{CEO}	35	V
	Emitter-collector voltage	V_{ECO}	6	V
	Collector current	I_C	50	mA
	Collector power dissipation	P_C	150	mW
Total power dissipation		P_{tot}	200	mW
*2Isolation voltage		V_{iso}	5 000	V_{rms}
Operating temperature		T_{opr}	- 30 to + 100	°C
Storage temperature		T_{stg}	- 55 to + 125	°C
*3Soldering temperature		T_{sol}	260	°C

III Transistor de puissance IRF 1010N

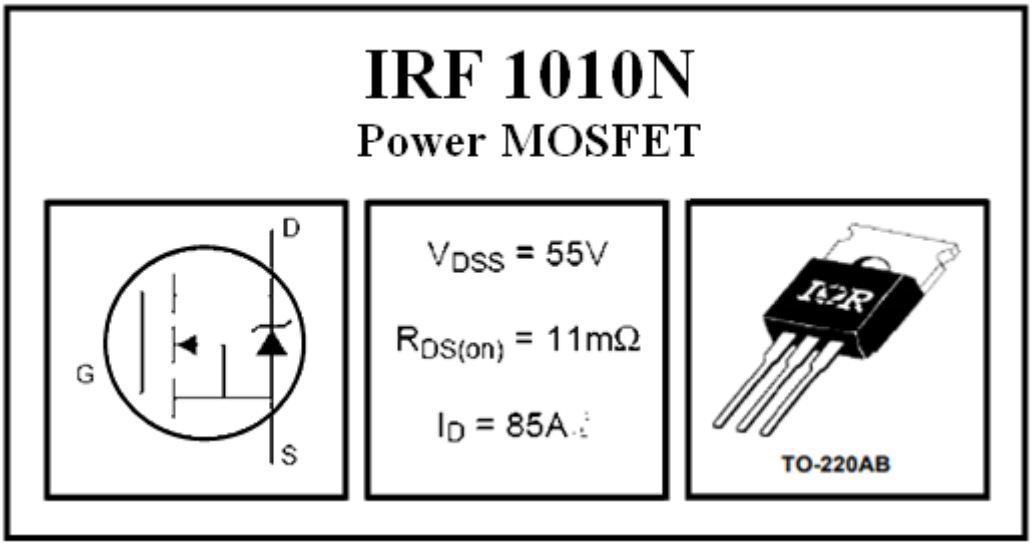


Tableau des caractéristiques électriques du Transistor de puissance IRF 1010N:

	Parameter	Max.	Units
I_D @ T_C = 25°C	Continuous Drain Current, V_{GS} @ 10V	85⑦	A
I_D @ T_C = 100°C	Continuous Drain Current, V_{GS} @ 10V	60	
I_{DM}	Pulsed Drain Current ①	290	
P_D @T_C = 25°C	Power Dissipation	180	W
	Linear Derating Factor	1.2	W/°C
V_{GS}	Gate-to-Source Voltage	± 20	V
I_{AR}	Avalanche Current①	43	A
E_{AR}	Repetitive Avalanche Energy①	18	mJ
dv/dt	Peak Diode Recovery dv/dt ③	3.6	V/ns
T_J T_{STG}	Operating Junction and Storage Temperature Range	-55 to + 175	°C
	Soldering Temperature, for 10 seconds	300 (1.6mm from case)	
	Mounting torque, 6-32 or M3 srew	10 lbf•in (1.1N•m)	

Résumé

Nous avons réalisé un automate programmable industriel à base d'une carte Arduino Méga (ADK).cet automate que nous avons conçu nous a permis de bien comprendre les concepts de l'automatisation.

Dans ce mémoire nous avons cité quelques généralités sur les automates programmables industriels et on a fait une étude sur la carte Arduino Méga (ADK) en présentant shilds. On a conçu et simulé les différents modules sur le logiciel protéus, ensuite on a passé aux tests sur lab d'essai et à la réalisation.

L'automate que nous avons réalisé à répondu globalement aux même exigences attendues d'un automate classique, notamment la possibilité d'être programmé et reprogrammé directement via un port parallèle d'un micro-ordinateur.