République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A. MIRA-BEJAIA

Faculté de Technologie

Département de Génie Electrique

Mémoire de Fin d'étude

En vue de l'obtention du diplôme de Master en Automatique

Spécialité : Automatique et Informatique Industrielle

Thème

Commande d'un ascenseur à base d'Arduino

Présenté par :

Melle. MADI Kenza
Melle. ASLOUN Katia

Encadré par :

Mr. HADJI Slimane

Examinateurs:
Mr. TAFININE
Mr. HANFOUG

Année universitaire 2018/2019

Remerciement

En préambule à ce mémoire nous remercions ALLAH le tout puissant et miséricordieux, qui nous a donné le courage, la force et la foi de mener à terme ce modeste travail.

À notre encadreur Monsieur HADJI.S Nous tenons d'abord à le remercier très chaleureusement d'avoir accepté de suivre notre projet et pour son attention particulière qu'il nous a donnée au courant de l'année, pour ses conseils indispensables, sa disponibilité, son aide, ses orientations, le temps consacré et son enthousiasme envers ce mémoire, qui ont constitués un apport considérable grâce auquel ce travail a pu être mené à bon port.

Nos vifs remerciements vont également aux membres du jury Mr.TAFININE et Mr.HANFOUG , qui ont bien voulu nous honorer par leur précieuse présences parmi nous, afin d'examiner et d'évaluer ce modeste travail ;

Veuillez trouver ici l'expression de notre respectueuse considération et notre profonde admiration pour toutes vos qualités scientifiques et humaines. Ce travail est pour nous l'occasion de vous témoigner notre profonde gratitude.

Dédicaces

Je dédié ce modeste travail a ceux qui, quels que soient les termes embrassés, je, n'arriverai jamais à leur exprimer mon amour sincère :

A l'homme de ma vie mon père,

Aucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours pour toi papa .Tu es mon exemple éternel, mon soutien moral et source de joie et de bonheur, tu s'est toujours sacrifié pour me voir réussir,

A ma chère mère,

Tu es la lumière de mes jours, la source de mes efforts .Par les inestimables sacrifices que tu as consentis pour moi, tu as tant souhaité que je parvienne à ce but. Je te serai reconnaissante toute ma vie, qu'Allah t'accorde longue vie et santé.

A mes grandes sœurs, leurs maries et ma nièce lina,

je dédie ce travail dont le grand plaisir leurs revient en premier lieu pour leurs conseils, aides, et encouragements.

A ma petite sœur Fatima,

Qui sait toujours comment procurer la joie et le bonheur pour toute la famille.

A mouloud,

Tes sacrifices, ton soutien moral et matériel m'ont permis de réussir mes études. Ce travail soit témoignage de ma reconnaissance et de mon amour sincère et fidèle.

A tata Hasina, tonton Kaci et soussou,

je ne peux pas trouver les mots justes et sincères pour vous Exprimer mon affection, mon amour et mes pensées, vous êtes pour moi un père une mère et une sœur sur qui je peux compter. Je vous dédie Ce travail et je vous souhaite une vie pleine de santé et de Bonheur.

A ma chère copine dida malgré son absence,

Mon complice, c'est un vrai bonheur de l'avoir dans ma vie. On peut se confier, sans jamais se juger. Les paroles sont inutiles entre nous, notre complicité étant évidente, j'ai hâte de la revoir.

A toutes mes cousines (hania et sihem, anisssa, amel),

Mes chères copines (Nacera , sarah et Warda),

Mes amis, collègues d'études,

Qui m'ont toujours aidé et encouragé, qui étaient toujours à mes côtés, merci.

Sans oublier ma binôme Kenza,

Pour son soutien moral, sa patience et sa compréhension tout au long de ce projet

Katia

Je dédie ce travail à ma chère famille considérée comme ma force et mon courage.

Je commence tout d'abord avec mes parents

À ma très chère mère;

Elle est la lumière de mes jours, la source de mes efforts, la flamme de mon cœur, ma vie et mon bonheur les mots ne seront jamais suffisant pour décrire l'amour que j'ai pour elle, c'est une mère, une sœur, une copine elle est tout pour moi.

À la mémoire de mon défunt Père,

Qui a quitté le monde mais pas mon cœur, je garde toujours sa photo dans ma tête son sourire, son charme et sa force au fond de moi. Et je sais qu'il serait fier de moi, il a toujours été ma source de force que Dieu bénisse son âme.

A l'homme de ma vie Amine,

Quoi que je fasse quoi que je dise je ne saurais point de te remercier comme il se doit, Ton affection me couvre, ta bien vaillance me guide et ta présence à mes cotes été toujours ma source de force pour affronter les différents obstacles.

À mes chères copines Lynda, Kenza, Imene,Nacera ,Warda et Sarah,

En souvenir de notre sincère et profonde amitié et des moments agréables que nous avons passés ensemble. Veuillez trouver dans ce travail l'expression de mon respect le plus profond et mon affection la plus sincère.

À tous mes frères mes sœurs et mes belles sœurs,

À qui je dédie ce travail dont le grand plaisir leurs revient en premier lieu pour

Leurs conseils, aides, et encouragements.

À mes chers petits neveux et nièces Barddine, Yassemine, Bilal, Malak, Narimane Sarah et foufou,

Aucune dédicace ne saurait exprimer tout l'amour que j'ai pour vous, Votre joie et votre gaieté me comblent de bonheur. Puisse Dieu vous garder, éclairer votre route et vous aider à réaliser à votre tour vos vœux les plus chers.

Et à mon binôme Katia

Pour son soutient Morel et patience Durant ce travail.

Aux personnes qui m'ont toujours aidé et encouragé, qui étaient Toujours à mes côtés, et qui m'ont accompagnés durant mon chemin d'études, mes aimables amis, collègues d'étude,

Merci.

Kenza

Sommaire

Liste des Figures

Chapitre I :

Chapitre II :

Chapitre III :

Liste des abréviations

C

CMOS==Complementary Metal Oxide Semiconductor

CAO=computer aided design

D

DC=direct current

E

E/S = Entrée / Sortie

EEPROM = Electrically Erasable Programmable Read-Only Memory

G

GUI=Graphical user interface

I

IDE = Integrated Development Environment

ICSP=In-Circuit Serial Programming

L

LED = Light-Emitting Diode

M

MCC = Moteur à Courant Continu

MIPS=Million instructions **p**er **s**econd

MCU =Microcontrôleur

N

NO =Normalement fermé

NF=Normalement fermé

NPN =négatif positif négatif

P

PWM = Pulse With Modulation

PNP= positifnegatifpositif

R

RISC==Reduced Instruction Set Computer)

S

SRAM = Static Random-Access Memory

SPI=Serial Peripheral Interface

T

TWO==Two Wire Interface

U

UART =universal asynchronous receiver transmitter

USB=Universal serial bus

Introduction générale

Avec la montée de l'urbanisation de notre société, l'ascenseur est devenu désormais un système indispensable pour répondre aux exigences modernes de notre vie en matière d'autonomie, de mobilité, d'accessibilité et de rapidité. Il est ainsi un élément essentiel des immeubles résidentiels, des bureaux, des musées, des aéroports, des centres de soins, des bâtiments publics, etc.…. Il contribue ainsi à gagner du temps, faciliter les déplacements, le transport et les courses.

En effet, ce projet consiste à étudier, à concevoir et à réaliser une carte de commande d'un ascenseur à base d'une carte Arduino Méga2560.

Ce rapport présente le déroulement du projet et permet de suivre la progression de notre travail ainsi que les résultats obtenus et les améliorations possibles.

L'objectif initial de ce projet était de commander une maquette d'ascenseurs à quatre étages. Et pour cela notre projet sera composé de trois chapitres :

- Le premier chapitre sera consacré à une présentation sur l'ascenseur. L'historique et les différents types et modes de fonctionnent, aussi nous avons montré les avantages et les inconvenants de chaque type, ensuite nous avons abordé les différentes parties de l'ascenseur de traction qui sera notre intérêt à ce projet, et à la fin, nous avons fait une présentation de la maquette (prototype).
- Le deuxième chapitre présente une étude théorique sur la carte Arduino Méga2560 et son environnement de programmation. Avant de nous lancer dans la commande de notre ascenseur, nous avons donné quelques applications sur des différents composants qui le constituent, après les avoir définis.
- Enfin le troisième chapitre présente une étude conceptuelle et étude de commande de notre projet, en premier lieu nous avons fait une présentation globale de notre ascenseur qu'on va commander avec Arduino, ensuite on va décrire la maquette où nous avons traité un petit cahier de charge ainsi qu'un organigramme de son fonctionnement. Avec bien sûr des tests sur le programme de commande développé.

Chapitre I : Généralités sur les ascenseurs

I.1. Introduction

Au cours de ce XXème siècle et surtout à compter des années 50 l'ascenseur passe d'un produit artisanal et architectural de luxe à un équipement s'industrialisant progressivement favorisant ainsi la démocratisation de l'ascenseur y compris dans les immeubles d'habitations. La forte urbanisation des années 1960-70 stimule la demande, entraînant la standardisation des produits.

Ce chapitre présente quelques catégories des ascenseurs et les différentes parties qui les constituent.

I.2. Définition d'un ascenseur

Appareil élévateur installé à demeure, desservant des niveaux définis, comportant une cabine, dont les dimensions et la constitution permettent manifestement l'accès des personnes, se déplaçant, au moins partiellement, le long de guides verticaux, ou dont l'inclinaison sur l'horizontale est supérieure à 15 degrés.[1]

I.3. Les catégories d'ascenseurs

On distingue deux grandes familles d'ascenseur :

- Les ascenseurs à traction à câble.
- Les ascenseurs hydrauliques.

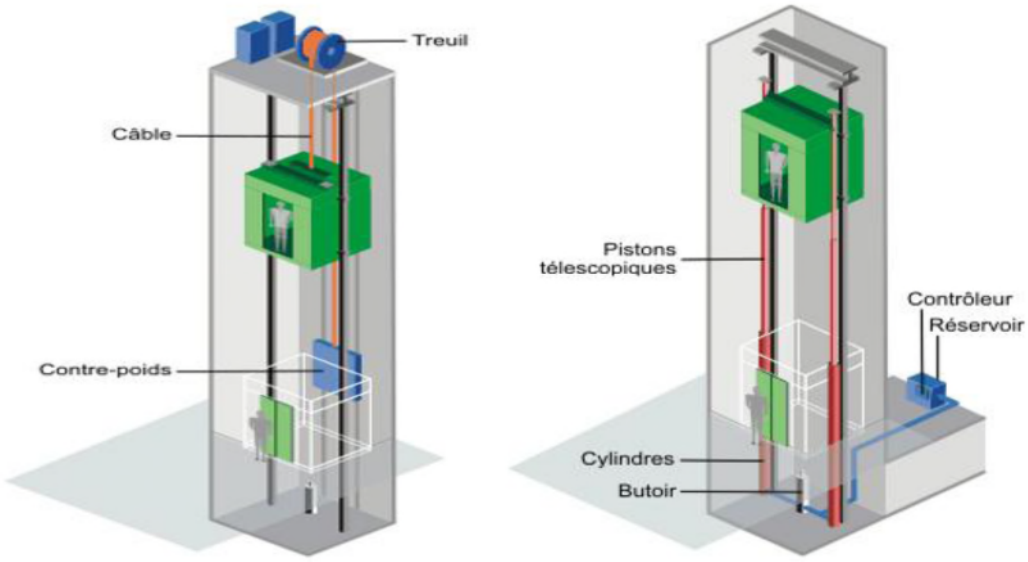


Figure I.1 : Les deux types d'ascenseur [2]

En général, ces deux types utilisent l'énergie électrique pour déplacer verticalement la cabine d'ascenseur, cependant, les ascenseurs hydrauliques sont nettement moins utilisés que les ascenseurs à treuil. [2]

I.4. Les ascenseurs hydrauliques

Principe

Comme toute machine hydraulique la pompe met sous pression l'huile qui pousse le piston hors du cylindre vers le haut. Lorsque la commande de descente est programmée, le by-pass (vanne) de la pompe permet de laisser sortie l'huile du cylindre vers le réservoir.

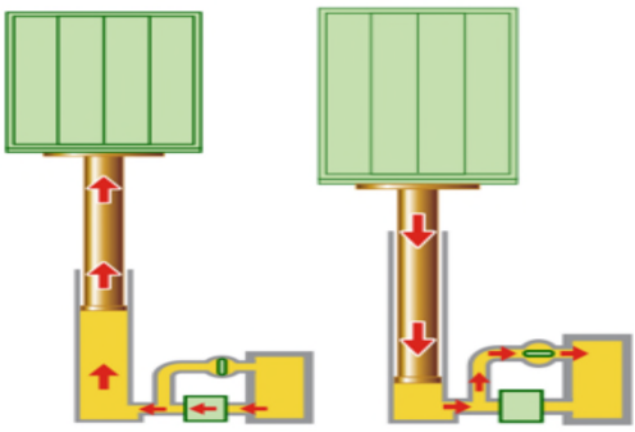


Figure I.2 : principe de fonctionnement d'un ascenseur hydraulique [2]

Description

Les ascenseurs hydrauliques sont utilisés en général pour satisfaire des déplacements relativement courts de l'ordre de 15 à 18 m maximums.

Plusieurs modèles existent sur le marché. Nous citerons les ascenseurs hydrauliques :

- À cylindre de surface.
- À cylindre enterré.
- Télescopiques à cylindre de surface.

Ce type d'ascenseur n'est pas très présent sur le marché.

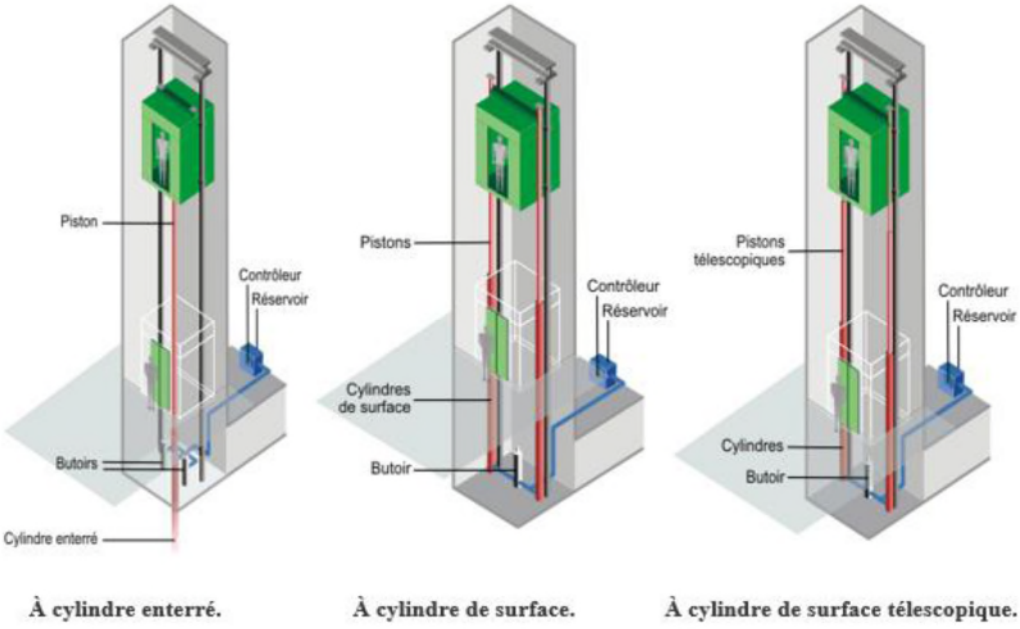


Figure I.3 : Les différents modèles des ascenseurs hydrauliques [2]

Les ascenseurs hydrauliques se composent principalement de :

- D'une cabine.
- De guides.
- D'un ensemble pistons-cylindres hydrauliques placé sous la cabine de l'ascenseur.
- D'un réservoir d'huile.
- D'un moteur électrique accouplé à une pompe hydraulique.
- D'un contrôleur.

Énergie

Énergétiquement parlant les ascenseurs hydrauliques posent un problème dans le sens où il n'y a pas de contrepoids qui équilibre la cabine comme dans les systèmes à traction à câble par exemple. [2]

Avantages et inconvénients

Ci-dessous, on trouvera les principaux avantages et inconvénients des ascenseurs hydrauliques :[2]

Avantage

- Précision au niveau du déplacement (mise à niveau).
- Réglage facile de la vitesse de déplacement.
- Ne ne nécessite pas de cabanon de machinerie.
- Implantation facile dans un immeuble existant.

Inconvénients

- Course verticale limitée à une hauteur entre 15 et 18 m.
- Risque de pollution du sous-sol.
- Consommation énergétique importante.
- Nécessiter de renforcer la dalle de sol.

I.5. Les ascenseurs à traction à câble

Description

Les ascenseurs à traction à câbles sont les types d'ascenseurs les plus fréquemment utilisés, notamment dans les bâtiments tertiaires. Ils se différencient entre eux selon le type de motorisation :

- À moteur-treuil à vis sans fin.
- À moteur-treuil planétaire.
- À moteur à attaque directe (couramment appelé "Gearless" ou sans treuil).

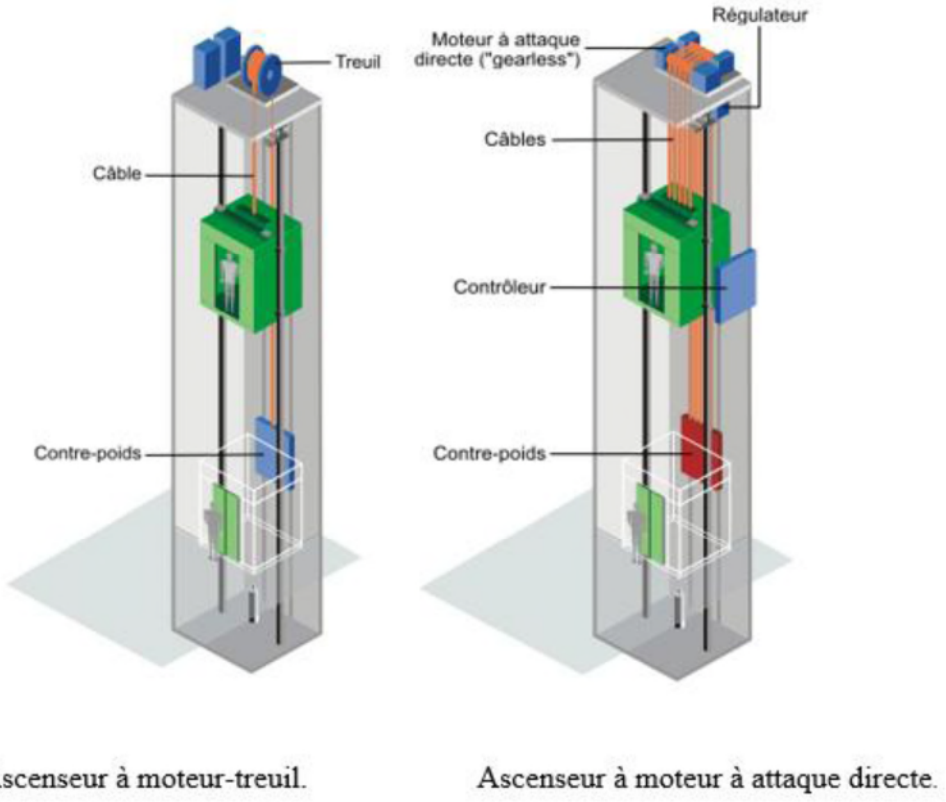


Figure I.4 : Les deux types d'ascenseur à traction à câble [2]

Quel que soit le type, les ascenseurs à traction à câbles comprennent généralement :

- Une cabine.
- Un contre-poids.
- Des câbles reliant la cabine au contre-poids.
- Des guides.
- Un système de traction au-dessus de la cage de l'ascenseur.

Énergie

Énergétiquement parlant les ascenseurs à traction à câbles sont plus intéressants que les ascenseurs hydrauliques dans le sens où le contrepoids réduit fortement la charge quelle que soit le type de motorisation. Les consommations et les courants de démarrages sont réduits par rapport aux ascenseurs hydrauliques.[2]

Avantages et inconvénients

Ci-dessous, on trouvera les principaux avantages et inconvénients des ascenseurs à câbles :[2]

Avantages

- Course verticale pas vraiment limitée.
- Suivant le type de motorisation précision au niveau de la vitesse et du déplacement.
- Rapidité de déplacement
- Efficacité énergétique importante.
- Pas de souci de pollution.

Inconvénients

- En version standard, nécessite un cabanon technique en toiture.
- Exigence très importante sur l'entretien.

Différentes parties d'un ascenseur à traction

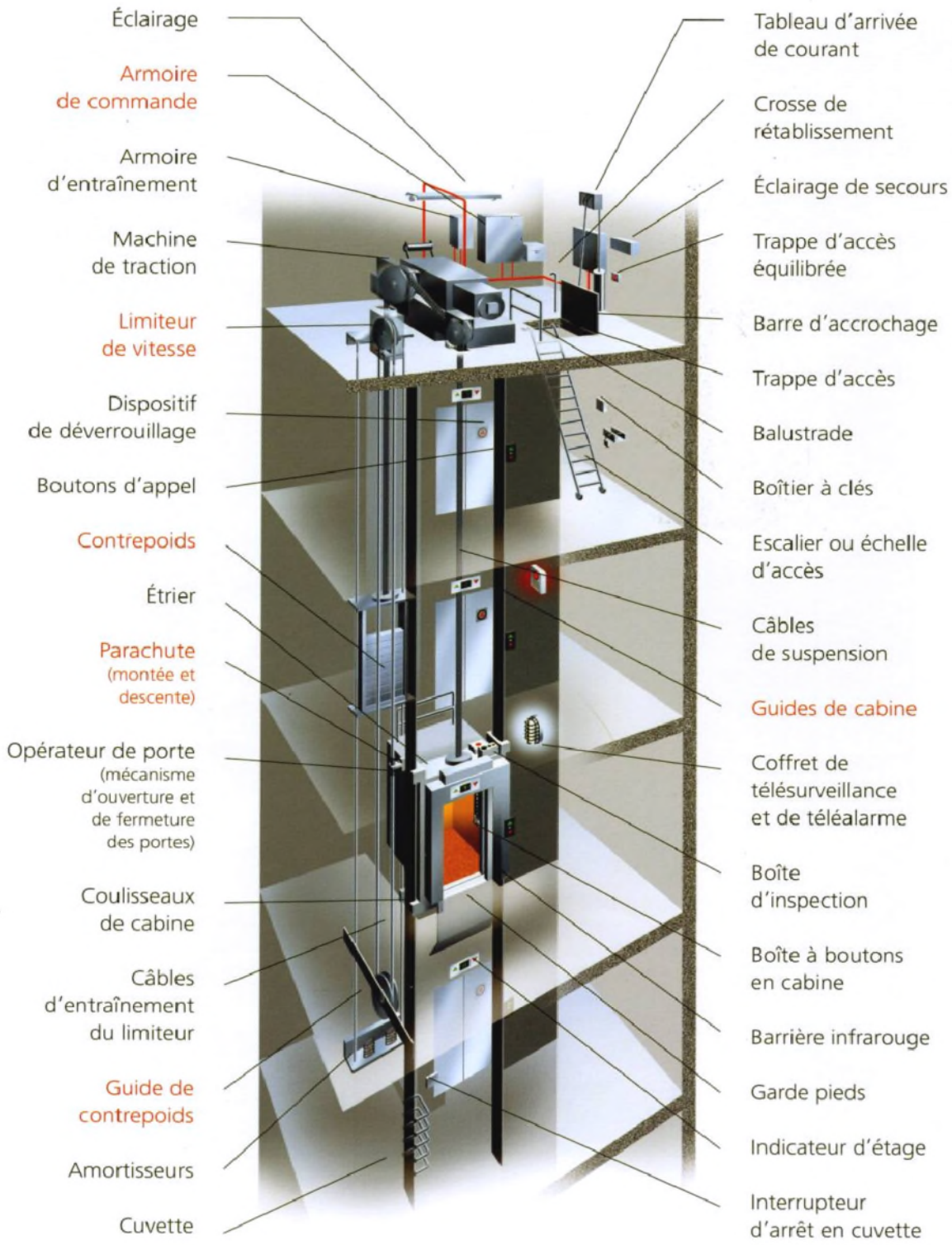


Figure I.5 : Les différentes parties d'un ascenseur à traction[12]

Nous nous limitons aux composants d'un ascenseur à traction, car il représente la majorité des ascenseurs qui existent sur le marché.

Machine de traction : ensemble des organes moteurs assurant le mouvement et l'arrêt de l'ascenseur. [12]

Limiteur de vitesse : Organe mécanique, Si la vitesse dépasse anormalement la vitesse maximale autorisée, les masselottes se lèvent et coupent un contact de sécurité. [12]

Dispositif de déverrouillage.

Boutons d'appel: On nomme boutons d'appels les boutons installés aux paliers. [12]

Contrepoids : Elément destiné à contre balancer le poids de la suspension cabine augmenté de la moitié de la charge utile. [12]

Étrier : Ossature métallique portant la cabine ou le contrepoids, attelée aux organes de suspension. Cette ossature peut faire partie intégrante de la cabine elle-même. [18]

Parachute (montée et descente) : Organe mécanique placé sur la suspension de cabine et commandé par un câble de limiteur. En cas de rupture des câbles de traction ou de survitesse exagérée en descente, le mécanisme du parachute assure un blocage mécanique de la suspension dans les guides évitant la chute libre de la cabine. [12]

Opérateur de porte (mécanisme d'ouverture et de fermeture des portes) : Porte à fermeture généralement automatique destinée à confiner l'utilisateur dans la cabine pendant le déplacement de celle-ci, lui interdisant tous contact avec les parties extérieures à la cabine.[16]

Coulisseaux de cabine : Eléments fixés à la suspension, garnis d'une fourrure épousant la forme des guides et destinés à guider celle-ci dans la gaine. [16]

Câble de sélecteur d'étage : Généralement, les sélecteurs d'étage mécaniques sont entraînés par le treuil ou le limiteur de vitesse. Cependant, certains sélecteurs d'étage ont leur propre câble d'entraînement. Celui-ci, relié entre la cabine d'ascenseur et le contrepoids entraîne un petit tambour qui actionne le sélecteur d'étage. [16]

Guide de contrepoids : Profilés en acier, généralement en forme de T, destinés à guider la cabine et le contrepoids dans la gaine. [16]

Amortisseurs : Ressorts puissants placés en cuvette et destinés a ralentir la suspension cabine ou le contrepoids en cas de dépassement des "fin de course" de sécurité. Dans le cas d'un ascenseur à grande vitesse, on utilise des amortisseurs à huile. [16]

Cuvette : Partie la plus basse de la gaine de l'ascenseur contenant les poulies de renvoi et les amortisseurs. [16]

Eclairage de secours : éclairage efficace de tous les locaux contenant machines et poulies. [17]

Trappe d'accès équilibrée.

Balustrade : système de fixation sur la cabine est assuré par deux pieds.

Boitier à clés : permettant de garder à disposition les clés des locaux techniques.

Escalier ou échelles d'accès: Pour faciliter l'accès aux fosses d'ascenseur.

Câble de suspension : L'espace libre entre la cabine et le sol de la fosse doit être assez grand et entièrement disponible sur toute la hauteur de la boucle du câble. Les câbles doivent être suspendus sous la cabine en conservant leur courbe naturelle. .

Guides de cabine : Rails en acier en forme de T sur lesquels coulisse la cabine.[12]

Coffret de télésurveillance et de télé-arme : Coffret principal d'alimentation se trouvant en machinerie. Avant toute intervention en machinerie, il va de la sécurité de l'utilisateur de déclencher le levier de ce coffret pour couper l'alimentation de l'appareil. [16]

Boite d'inspection: Le boiter d'inspection est un boiter électrique situé sur le toit de la cabine. Son usage est réservé aux intervenants. Il permet de prendre le contrôle de l'appareil et de se déplacer (avec la cabine), en étant sur le dessus de la dites cabine. Son bon fonctionnement doit être testé avant de monter sur la cabine ; chose qui, si elle n'est pas faite, peut générer un accident pouvant aller jusqu'à la mort.[16]

Boite à boutons en cabine : boutons d'envois sont installés dans la cabine.

Barrière infrarouge : Une protection fiable et sûre de la porte de l'ascenseur.

Garde pieds : C'est une tôle fixe ou rétractable, destinée à protéger les chutes en gaines lorsque la cabine est immobilisée en dehors de la zone de déverrouillage. [16]

Indicateur d'étage :

- Dans la cabine :

Indique la position de l'ascenseur, ainsi que dans la plupart des cas, la direction. Les premiers indicateurs furent "analogiques", les numéros des étages étant tout d'abord un cadran avec une aiguille, puis alignés sur le mur, et chaque nombre s'allumait lorsque l'ascenseur y passait, ainsi qu'une flèche lumineuse qui indiquait la direction de déplacement.

Puis vint l'indicateur LCD numérique, le plus souvent rouge sous une plaque noire. La flèche fut intégrée à l'afficheur.

- Depuis l'extérieur

L'étage est souvent indiqué uniquement au réez de chaussée, dans ce cas l'utilisateur attendant depuis un étage supérieur ne peut pas savoir à quel niveau la cabine est située, excepté le sens de déplacement qui est parfois indiqué par une flèche qui clignote lorsque la cabine est en déplacement ou qui reste continue lorsque la cabine est à l'arrêt.

L'étage de positionnement est toutefois indiqué sur quelques ascenseurs depuis l'extérieur sur tous les étages.

L'arrivée d'une cabine est signalée par un petit retentissement de sonnerie.

Interrupteur d'arrêt en cuvette : Dispositif qui permet l'arrêt de la cabine d'ascenseur à chaque niveau. [18]

Principe de fonctionnement d'unascenseur à traction

Un ascenseur à contrepoids se compose d'une cabine qui se déplace dans un couloir vertical nommé gaine et qui est guidée par des rails afin d'éviter une collision avec le contre poids. Un frein situé dans la machinerie du moteur permet de stopper la cabine à l'étage demandé. Le déplacement en translation de la cabine est permis par un système de transmission de mouvement. Des câbles, actionnés par un treuil permettent de mettre en mouvement la cabine et le contrepoids. Le moteur du treuil permet la mise en mouvement. Le contrepoids est une charge

lourde qui sert à équilibrer la charge de la cabine et à diminuer l'énergie à fournir par le moteur. Lorsque la cabine monte, le contrepoids descend. Le système comprend aussi des organes de commande pour enregistrer les appels des usagers et optimiser les déplacements de la cabine afin de répondre le plus rapidement possible aux différents appels.

Enfin, l'ascenseur est équipé d'organes assurant la sécurité des passagers. Des freins d'urgence ou parachutes sont placés de chaque côté de la gaine et se déclenchent en cas de rupture du câble tracteur pour éviter la chute de la cabine. Ils sont déclenchés par un limiteur de vitesse lorsque la vitesse de la cabine est supérieure à la vitesse de déplacement normale (de 2 à 9 km/h selon les ascenseurs). Les parachutes bloquent alors de façon brutale la cabine sur les guides. [13]

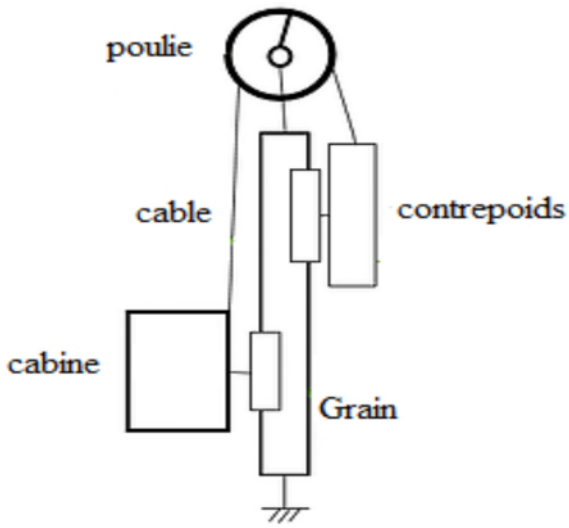


Figure I.6 : Principe de fonctionnement d'un ascenseur à traction

Le contrepoids est un peu plus lourd que la cabine : il sert à contrebalancer le poids de la cabine. Celle-ci est suspendue par des câbles grâce à des poulies, l'effort du moteur pour élever la cabine est réduit. Poids effectif = contrepoids moins poids de la cabine et de sa charge.

I.6. Les critères du choix du type d'ascenseur

En général, les dépenses énergétiques des ascenseurs ne sont pas la priorité des gestionnaires de bâtiments tertiaires. En effet, la préoccupation première reste

avant tout : emmener un maximum de monde en toute sécurité et avec un maximum de confort.

On retrouve des critères de choix :

- **Constructifs** : hauteur de bâtiment, espace disponible au niveau des étages, possibilité de placer une salle des machines au sommet de la gaine, stabilité du terrain de sécurité.
- **Organisationnels** : comme le type de fonction du bâtiment, son occupation et son type de fonctionnement en garantissant une performance de confort et de trafic (rapport vitesse/charge)
- **Energétiques** : basées essentiellement sur la consommation et les appels de puissance de la motorisation. [2]

I.7. Conclusion

Dans ce chapitre nous avons tout d'abord fait une présentation générale des ascenseurs, l'historique et leurs différents types et modes de fonctionnent. Les ascenseurs hydrauliques sont plus lents et consomment plus que les ascenseurs électriques, à notre projet on s'intéresse au deuxième type, nous nous concluons qu'il y a plusieurs solutions pour commander l'ascenseur parmi ces solutions on utilise la commande par carte **Arduino Méga 2560** qui sera l'objectif du chapitre suivant.

Chapitre II : Présentation de la carte Arduino et son environnement de programmation

II.1. Introduction

Après un aperçu global sur le fonctionnement des ascenseurs, nous allons apprendre à programmer sous Arduino les différents composants qui sont utilisés dans le système de control de conçu. Mais avant ca on doit dans ce chapitre les règles qui régissent l'écriture du langage Arduino.

Arduino a été conçu pour être accessible à tous par sa simplicité. Mais il peut également être d'usage professionnel, tant les possibilités d'application sont nombreuses. Ces cartes polyvalentes sont donc parfaites pour, ceux qui ne demandent qu'apprendre et progresser. Grace à cette carte, ces fonctions sont réalisées par des capteurs, des actionneurs, et plus généralement par des équipements électriques et/ou électroniques.

II.2. Définition d'Arduino

Arduino désigne un écosystème libre comprenant des cartes (Arduino Uno, Arduino Leonardo, Arduino Méga, Arduino Nano...), des logiciels (notamment l'IDE Arduino), ou encore des librairies.

La syntaxe du langage de programmation sous Arduino est l'ensemble des règles d'écriture liées à ce langage.

Ces systèmes d'électroniques programmables permettent de construire des projets facilement, et d'aborder tant l'approche électronique de l'approche logicielle.[3]

II.3. Architecture de la carte Arduino

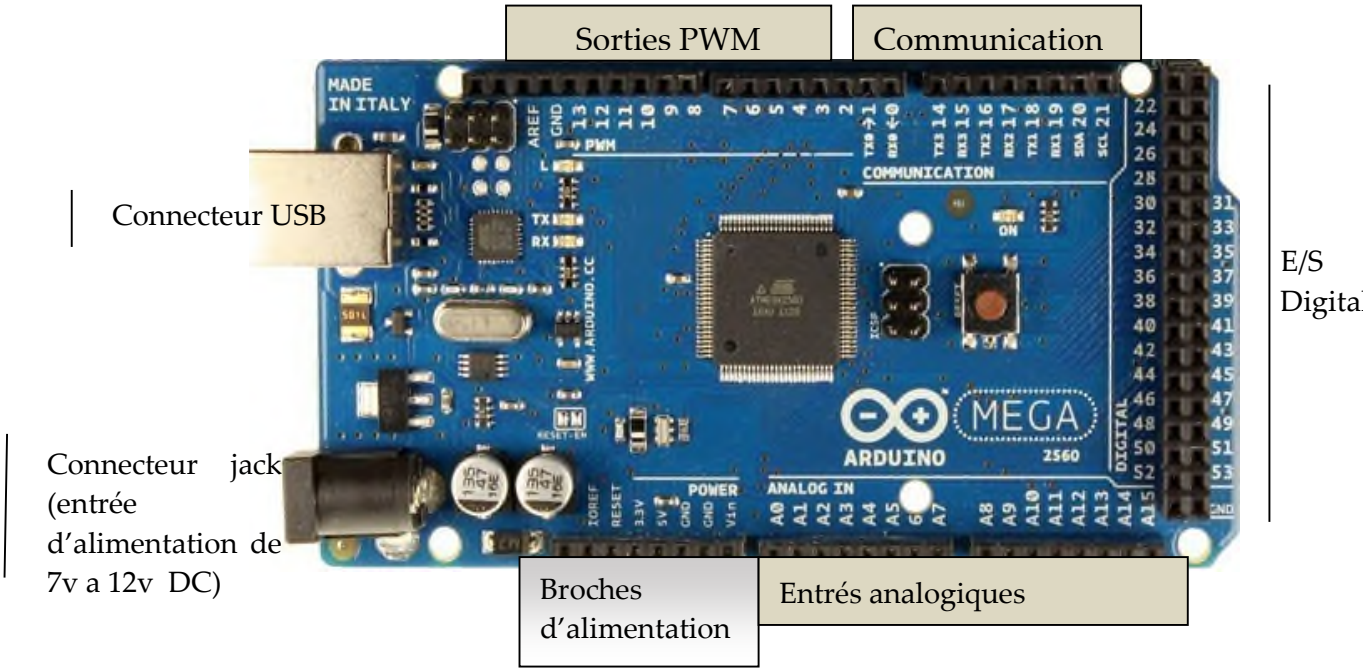


Figure II.1 : Carte Arduino Méga 2560[3]

II.3.1. Définition de la carte Arduino

La carte Arduino Méga 2560 est une carte à microcontrôleur basée sur un ATmega2560. Elle contient tout ce qui est nécessaire pour le fonctionnement du microcontrôleur; Pour pouvoir l'utiliser et se lancer, il suffit simplement de la connecter à un ordinateur à l'aide d'un câble USB (ou de l'alimenter avec un adaptateur secteur ou une pile, mais ceci n'est pas indispensable, l'alimentation étant fournie par le port USB.[3]

II.3.2. Définition du microcontrôleur

Le ATmega640 / 1280/1281 / 2560/2561 est un CMOS à faible puissance microcontrôleur 8 bits basée sur l'AVR amélioré l'architecture RISC. En exécutant des instructions puissantes dans un seul cycle d'horloge, le ATmega640 / 1280/1281 / 2560/2561 atteint des débits approchant 1 MIPS par MHz permettant au concepteur du système pour optimiser la consommation d'énergie par rapport à la vitesse de traitement. [3]

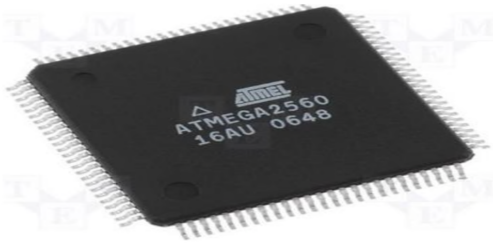


Figure II.2 : Microcontrôleur AtMéga 2560[3]

II.3.3. Les principales caractéristique dela carte Arduino Mega

Les principales caractéristiques de la carte Mega sont :

- Microcontrôleur : ATmega2560
- Tension de fonctionnement : 5 V
- alimentation: via port USB ou 7 à 12 V sur connecteur alimentation
- Ports digitaux I/O : 54 dont 14 PWM
- Ports d'entrée analogique : 16
- Courant direct par pin I/O : 40 Ma
- Mémoire flash : 256 KB dont 8 KB réservé au Bootloader (il permet de télécharger les programmes entre l'IDE Arduino (interface de développement).
- Mémoire SRAM : 8 KB.
- Mémoire EEPROM : 4 KB
- Fréquence d'horloge de l'oscillateur à quartz : 16 MHz
- Ports série UART: 4.
- Bus I2C et SPI.
- 6 entrées à interruptions
- Port USB (fiche B) géré par le MCU ATmega 16U
- Un connecteur ICSP (programmation "in-circuit").
- Un bouton de réinitialisation (Reset) .[3]

II.3.4. Visualisation

- LED 'ON' : indique que la carte est sous tension.
- LED 'Tx' et LED 'Rx': servent à visualiser l'activité sur la voie série (pendant l'émission et la réception de données ou le chargement du programme dans le microcontrôleur par leport USB).
- LED 'L': elle est connectée à la broche 26 du microcontrôleur (port 13 d'Arduino) et sert pour tester le matériel. Cette LED clignote quelques

secondes aprèsl'initialisation de la carte (et après le branchement de la carte au PC). [3]

II.3.5. Les connecteurs d'entrées/sorties :

- Les ports E/S numérique 0-53 peuvent être configurés en entrée ou en sortie
- Les ports PWM 0-13 codée sur 8 bits (0 à 255)
- Les ports d'entrée analogiques A0-A15
- Ports de communication : UART1/USB 0(RX0), 1(TX0), UART2 18(TX1), 19(RX1), UART3 utilise les broches: 16(TX2), 17(RX2), UART4 14(TX3), 15(RX3).
- SPI est accessible via les broches ICSP ou les broches: 50(MISO), 51(MOSI), 52(SCK), 53(SS).
- I²C accessible via les broches: 20(SDA), 21(SCL). [3]

II.4. Environnement de programmation :

II.4.1. Description :

L'environnement de développement intégré, dédié au langage Arduino et à la programmation des cartes Arduino, permet :

- d'écrire et compiler des programmes pour la carte Arduino
- de programmer la carte Arduino (y transférer les programmes)
- de communiquer avec la carte Arduino (débogage…)

Comme pour tout logiciel une interface graphique (GUI), l'IDE Arduino comporte les éléments suivants :

- une Barre de Menus
- une Barre de Boutons qui donne un accès direct aux fonctions essentielles.
- un Editeur (à coloration syntaxique) pour écrire le code des programmes, avec onglets de navigation, Le programme écrit avec le logiciel Arduino est appelé sketch (ou croquis - en français), le fichier correspondant est d'extension « .ino ».
- une Zone de messages qui affiche indique l'état des actions en cours (compilation, transfert du programme vers la carte...).
- une Console d'affichage qui affiche les messages concernant le résultat de la compilation du programme (taille programme compilé, erreurs …).

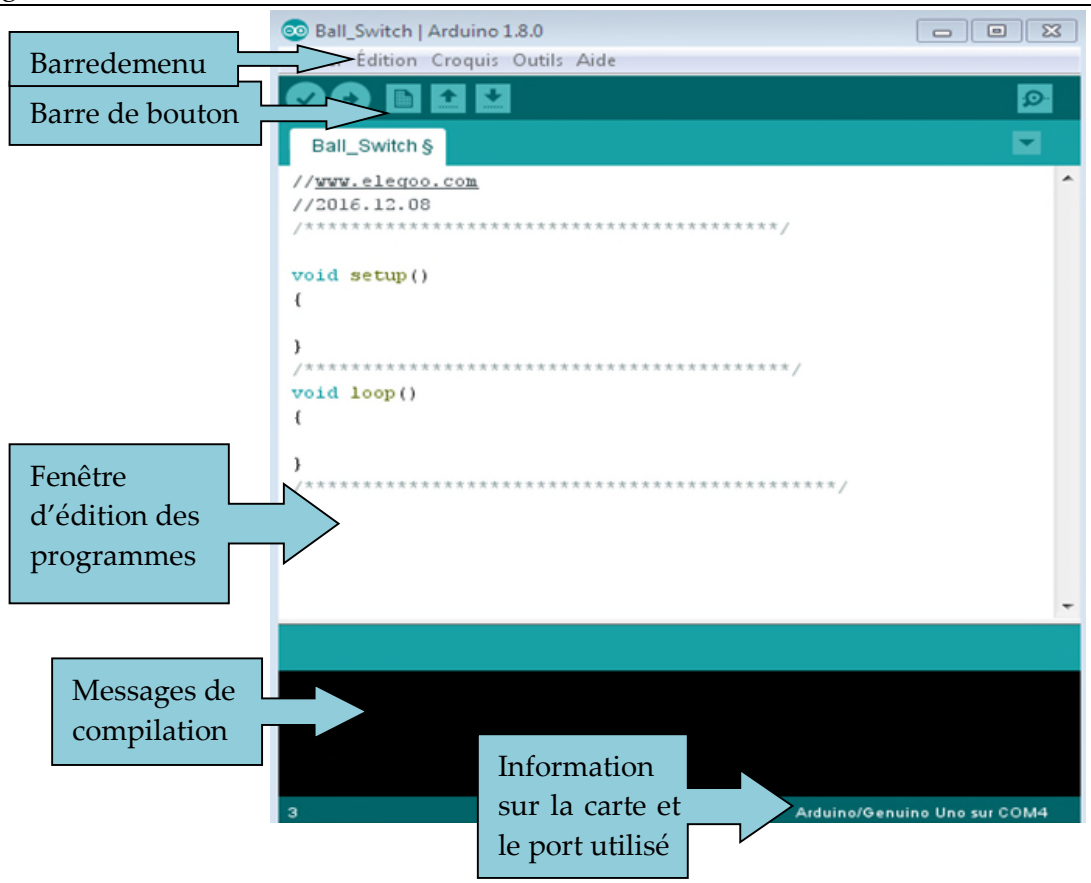


Figure II.3 : Les éléments de l'interface d'Arduino IDE

II.4.1.1. Description de la barre de boutons

La barre de boutons permet un accès rapide aux fonctions usuelles, elle est représentée par la Figure suivante :

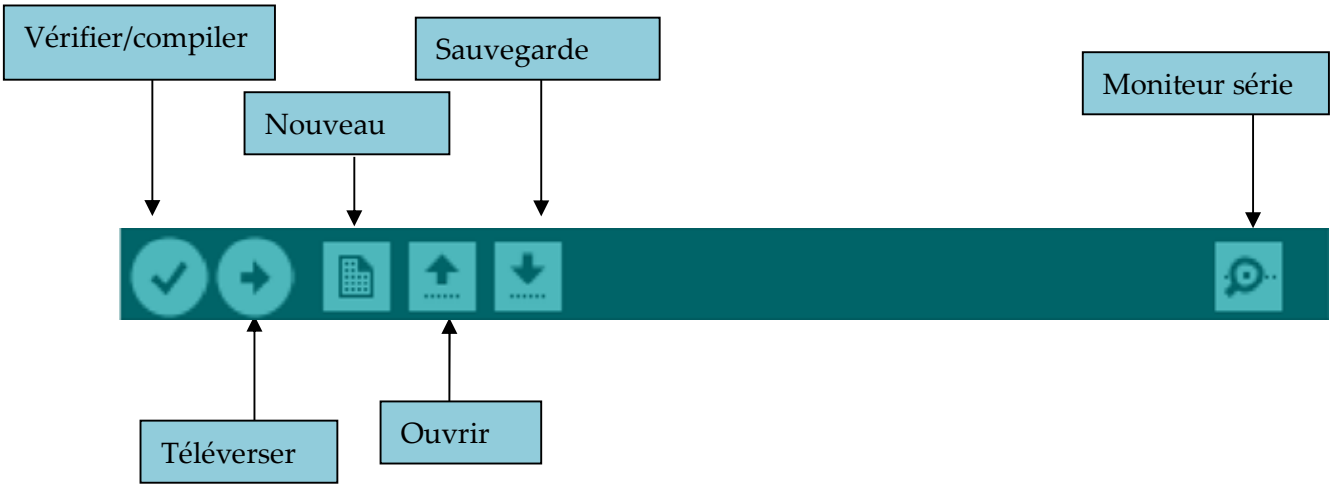


Figure II.4 : Barre de boutons

- **Vérifier/compiler** : Vérifie le code à la recherche d'erreur de syntaxe.
- **Nouveau** : Crée un nouveau code (ouvre une fenêtre d'édition vide).

- **Ouvrir** : Ouvre la liste de tous les programmes exemples.
- **Sauvegarder** : Enregistre le programme.
- **Téléverser** : Compile le code et le transfert vers la carte Arduino.
- **Moniteur Série** : Ouvre la fenêtre du moniteur série.

Moniteur Série

Le logiciel Arduino intègre également un Moniteur et un Traceur série (fenêtres séparées) qui permettent principalement d'afficher des messages textes et des valeurs reçus de la carte Arduino sous formes de liste ou de graphe. Ces fonctionnalités permettent d'afficher sur l'ordinateur l'état de variables, de résultats de calculs ou de conversions analogiques numériques, ce qui permettra de tester et corriger les programmes (débogage).

Le moniteur série assure une communication entre l'ordinateur et la carte Arduino via les ports 0 (RX) et 1 (TX) du connecteur USB. Ainsi, si l'on utilise ces fonctions, on ne peut pas utiliser la broche 0 et 1 pour E/S numérique. . Le lancement du Moniteur série se fait simplement à partir de la barre d'outils en sélectionnant le même débit en bauds que celui configuré dans le programme vérifié [3]

Figure II.5 : L'interface du moniteur série

II.4.1.2. Barre de menu

Des commandes complémentaires sont disponibles dans cinq menus :

- Fichier
- Edition
- Croquis
- Outils
- Aide

II.4.2. Structure du programme

Le logiciel permet de programmer la carte Arduino, Il offre une multitude de Fonctionnalités. Le langage Arduino est inspiré de plusieurs langages. Le langage impose une structure particulière typique de l'informatique embarquée. Le programme est lu par le microcontrôleur de haut vers le bas. Une variable doit être déclarée avant d'être utilisée par une fonction.

La structure minimale est constituée :

- **En tête** : déclaration des variables, des constantes, indication de l'utilisation de Bibliothèques.
- **Un setup** (= initialisation) : cette partie n'est lue qu'une seule fois, elle comprend les fonctions devant être réalisées au démarrage (utilisation des broches en entrées ou en sortie,..)
- **Une loop** (boucle) : cette partie est lue en boucle, C'est ici que les fonctions sont réalisées. En plus de cette structure minimale, on peut ajouter : Des « sous-programmes » ou « routines » qui peuvent être appelées à tout moment dans la boucle, très pratiqué pour réaliser des morceaux de codes répétitifs.

Exemple :

Dans cet exemple on va faire clignoter une LED:

L'instruction de configuration du port par pinMode() est insérée dans la fonction setup() parce qu'il est nécessaire qu'une fois. Cela fonctionnerait toujours si nous déplacions l'appel dans la boucle, mais il serait inutile de répéter cette action de configuration. Les instructions qui agissent sur le port pour faire clignoter la LED sont insérées dans Loop (exécution répétée).

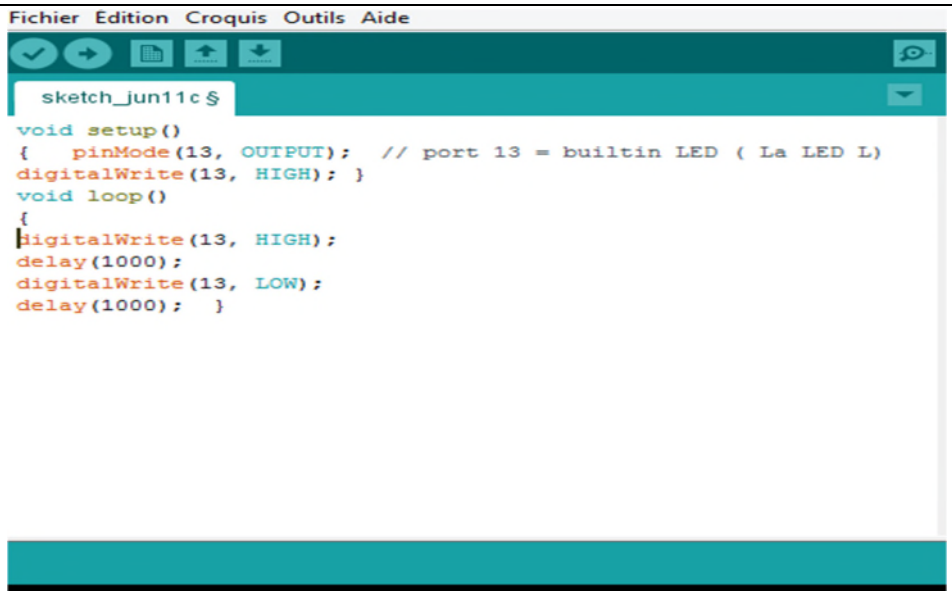


Figure II.6 : Code clignotement d'une LED.

II.5. Représentation de la plateforme Fritzing

Avant de passer à la réalisation pratique, nous avons utilisé un CAO: Il s'agit de Fritzing, c'est un logiciel libre de conception de circuit imprimé qui permet de concevoir de façon entièrement graphique le circuit et d'en imprimer le typon. [4]

Ce logiciel comporte trois vues principales :

- La platine d'essai : ou l'on voit les composants tel qu'ils sont dans la réalité et où l'on construit le montage.
- La vue schématique : représente le schéma fonctionnel du circuit.
- Le circuit imprimé : représente ma vue de circuit imprimé tel qu'il sera sorti en PDF pour être imprimé.

Parmi les composants proposés par défaut sur Fritzing, on peut citer :

- Les composants électroniques standards (résistance, diode, transistor, etc…).
- Les circuits intégrés logiques simples.
- Les capteurs les plus courants
- Les composants de sorties les plus courants (LEDs, écran LCD, servomoteur, relais, etc…)
- Différents types d'alimentations.
- Les connecteurs les plus courants (USB, jack, microSD, etc…).
- Les différents types des cartes Arduino et microcontrôleurs.

- Quelque platine d'essai. [4]

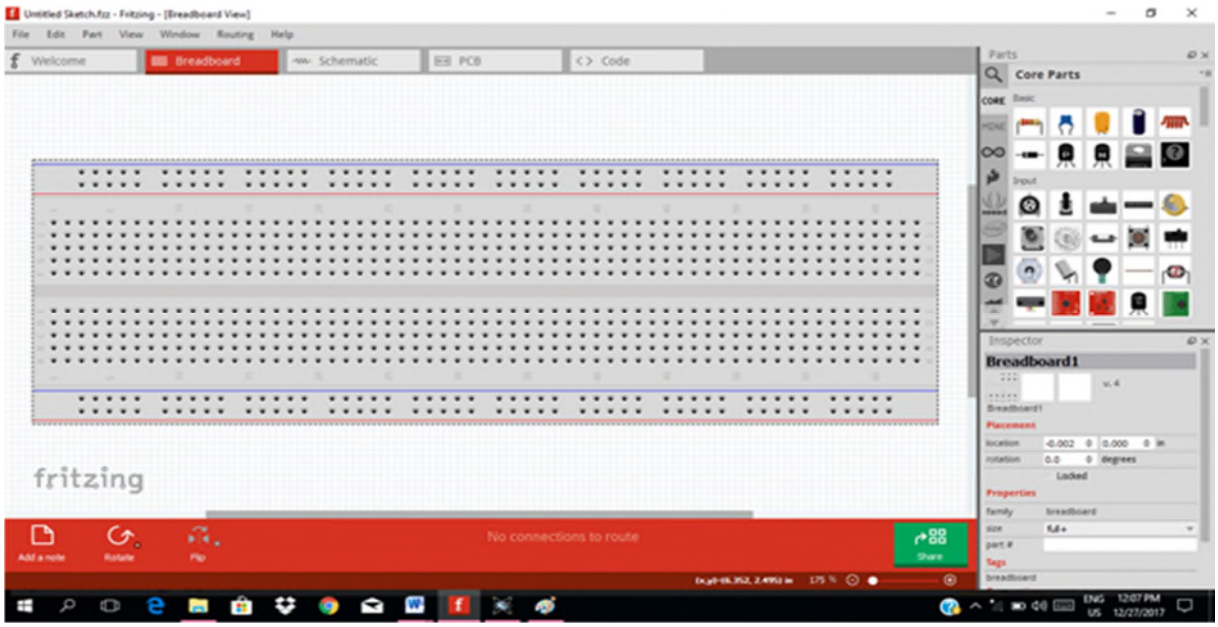


Figure II.7 : Plateforme de Fritzing

II.6. Description des composants

II.6.1. LED et boutton poussoir

II.6.1.1. Les LED

Description :

Les LEDs sont des diodes qui produisent de la lumière lorsque le courant y passe en polarisation directe de l'anode vers la cathode. Tout comme toute diode, la polarisation inverse bloque le courant et la LED dans ce cas n'émet aucune lumière. [5]

La LED est caractérisée par des propriétés électriques, optiques, thermiques et géométriques.

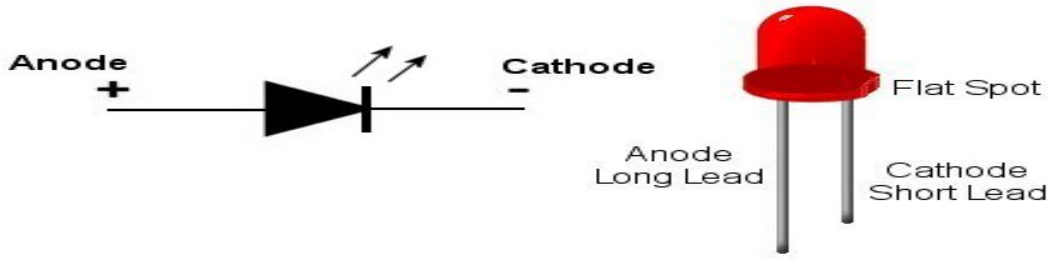


Figure II.8 : Représentations d'une LED [5]

II.6.1.2. Bouton poussoir

Description

Les boutons poussoirs sont des composants très simples. Lorsque vous pressez le bouton, un contact électriques fait et laisse passer le courant. Les boutons poussoirs ont 4 pattes ce qui peut créer une certaine confusion. [6]

Fonctionnement :

- Lorsqu'il est ouvert, la tension à ses bornes ne peut être nulle (ou alors c'est que le circuit n'est pas alimenté). En revanche, lorsqu'il est fermé cette même tension doit être nulle. En effet, aux bornes d'un fil la tension est de 0V.
- Ensuite, lorsque le bouton est ouvert, aucun courant ne peut passer, le circuit est donc déconnecté. Par contre, lorsqu'il est fermé, le courant nécessaire au bon fonctionnement des différents composants le traverse.

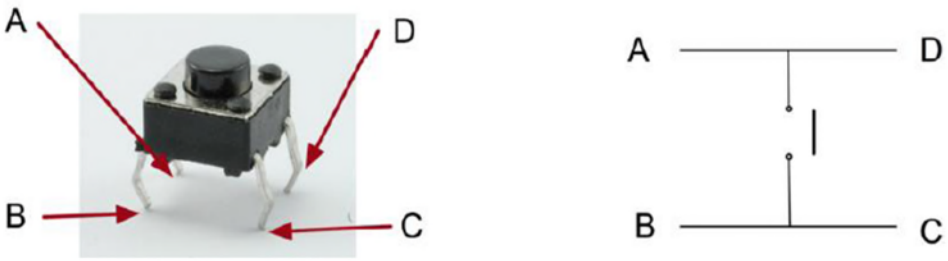


Figure II.9 : Bouton poussoir [6]

II.6.1.2.1. Bouton poussoir normalement ouvert (NO) :

Ils ont deux positions :

- **Relâché** : le courant ne passe pas, le circuit est déconnecté ; on dit que le circuit est **ouvert**.
- **Appuyé** : le courant passe, on dit que le circuit est **fermé**.

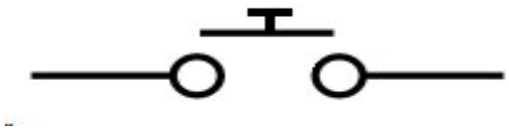


Figure II.10 :Schéma d'un BP (NO) [6]

II.6.1.2.2. Bouton poussoir normalement fermé :

Ce type de bouton est l'opposé du type précédent, c'est-à-dire que lorsque le bouton est relâché, il laisse passer le courant. Et inversement :

- **Relâché** : le courant passe, le circuit est connecté ; on dit que le circuit est "**fermé**".
- **Appuyé** : le courant ne passe pas, on dit que le circuit est **ouvert**.

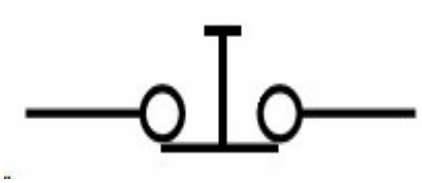


Figure II.11 : Schéma d'un BP (NF) [6]

Exemple :

Commande d'une LED avec deux boutons poussoirs :

Dans cette application, nous allons utiliser les entrées digitales pour allumer et éteindre une LED.

Presser un premier bouton allumera la LED, presser un autre bouton l'éteindra.

Matériel nécessaire :

- Une Carte Arduino MEGA 2560.
- Une Planche prototype.
- Une LED rouge 5mm.
- Une Résistance 220 ohm.
- deux Boutons poussoirs
- Sept câbles male-male

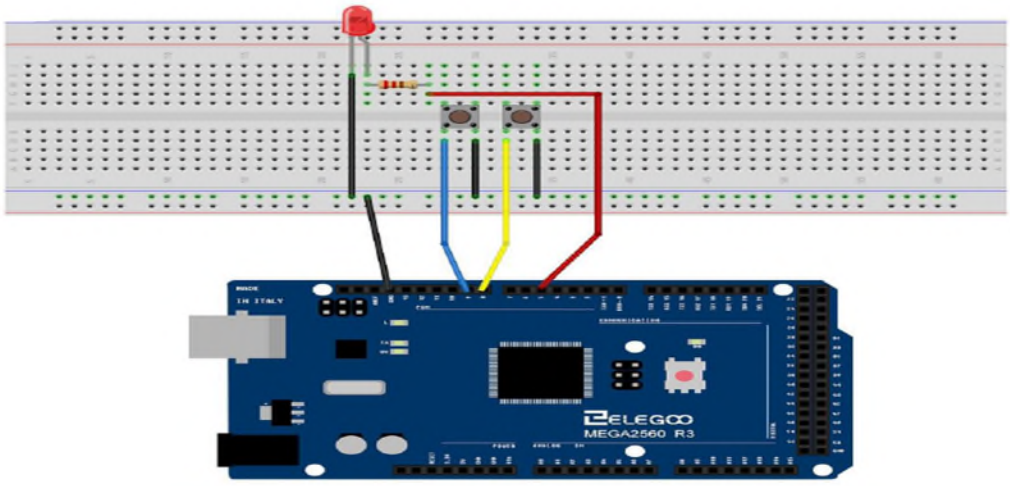


Figure II.12 : Vue prototype du câblage d'une LED avec deux boutons poussoirs

Programmer sur Arduino :

La première partie du sketch est consacré à la définition des trois variables pour les trois pins qui seront nécessaires au fonctionnement du montage.

Dans le bloc **(setup),** les pins sont affectés en tant qu'entrée ou sortie.

Ledpin : est défini en tant que sortie

ButtonApin et buttonBpin : en tant qu'entrée.

pinMode(buttonApin, INPUT_PULLUP);

pinMode(buttonBpin, INPUT_PULLUP);

Les entrées sont définies en tant que **INPUT_PULLUP** et non pas définis comme **INPUT simple** parce que :

Le fait de presser un bouton met la pin a la masse, mais lorsque le bouton n'est pas presser il peut subsister des signaux parasites qui peuvent être interpréter par la carte comme une mise a la masse, alors que le bouton n'est pas pressé.

Le bloc **loop** prend la mesure de la position de chaque bouton.

led_boutton

```
int ledPin = 5;
int bouttonApin = 9;
int bouttonBpin = 8;

void setup()
{
  pinMode(ledPin, OUTPUT);
  pinMode(bouttonApin, INPUT_PULLUP);
  pinMode(bouttonBpin, INPUT_PULLUP);
}

void loop()
{
  if (digitalRead(bouttonApin) == LOW)
  {
    digitalWrite(ledPin, HIGH);
  }
  if (digitalRead(bouttonBpin) == LOW)
  {
    digitalWrite(ledPin, LOW);
  }
}
```

Enregistrement terminé.

Figure II.13 : Code entrées digitaux

Après l'exécution : On regarde si le bouton A est pressé (LOW). Si oui on allume la LED, si non, on ne fait rien.

On regarde si le bouton B est pressé. Si oui on éteint la LED, si non, on ne fait rien.

II.6.2. Buzzer

Un buzzer est un élément électromécanique ou électronique qui produit un son quand on lui applique une tension. Certains nécessitent une tension continu (buzzers électromécaniques), d'autres nécessitent une tension alternative (transducteurs piézo-électrique). [7]

Il existe 2 types de Buzzer : Actif et passif

II.6.2.1. Buzzer actif

Peut recevoir une tension continue. On le différencie du Buzzer passif, car l'électronique n'est pas apparente sur la face inférieure du buzzer. De plus le Buzzer actif possède souvent une étiquette sur le dessus. [5]

Figure II.14 : Buzzer actif [7]

II.6.2.2. Buzzer passif

Le principe de fonctionnement d'un buzzer passif est d'utiliser un signal PWM (alternatif). Modulé à la bonne fréquence, la vibration génère différents sons selon la variation de la fréquence qui est comprise entre 500 HZ et 5 KHZ. [5]

Figure II.15 : Buzzer passif [7]

La différence entre les deux réside dans le fait qu'un buzzer actif possède un oscillateur intégré, il va donc générer un son lorsque le courant passe. Un buzzer passif utilise un signal en entrée pour générer le son (généralement signal carré en2 KHZ et 5 KHZ).

Exemple :

Dans cet exemple on va générer des sons avec un buzzer actif.

Matériel utilisé :

- une carte Arduino Méga 2560.

- Un buzzer actif.
- Deux câbles males-femelles.

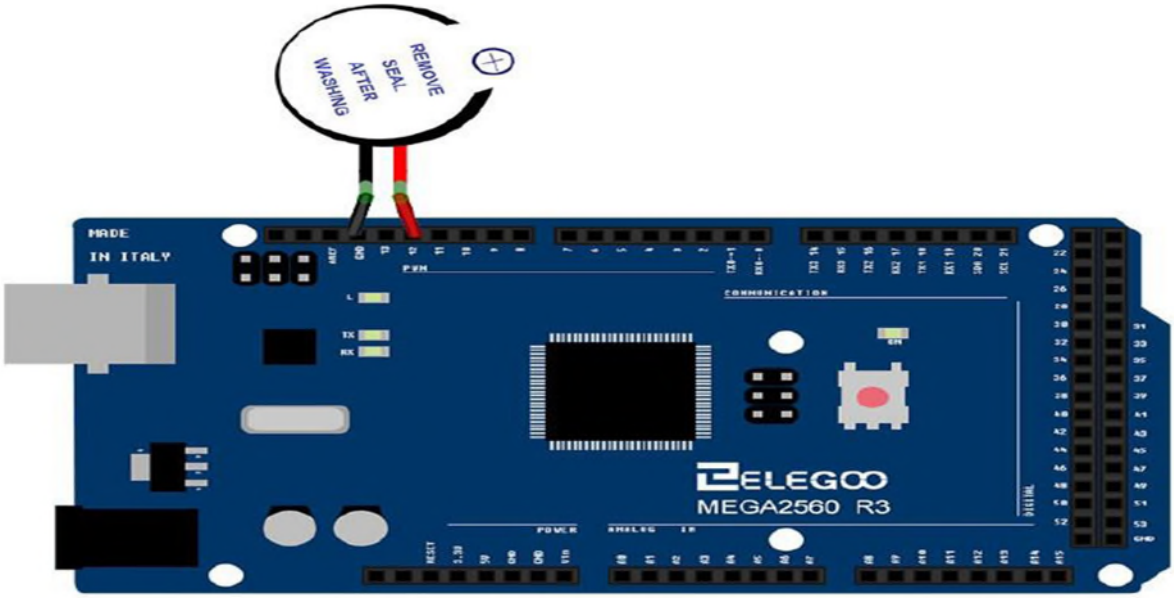


Figure II.16 : Vue prototype d'un buzzer actif.

Application sous Arduino :

- Au niveau de la partie déclarative : on déclare le pin de buzzer.
- Dans de la fonction initialisation setup() : on initialise le buzzer comme sortie.
- Au niveau de la boucle principale, la fonction loop ( ) : générer deux différents sons avec deux différentes fréquences.

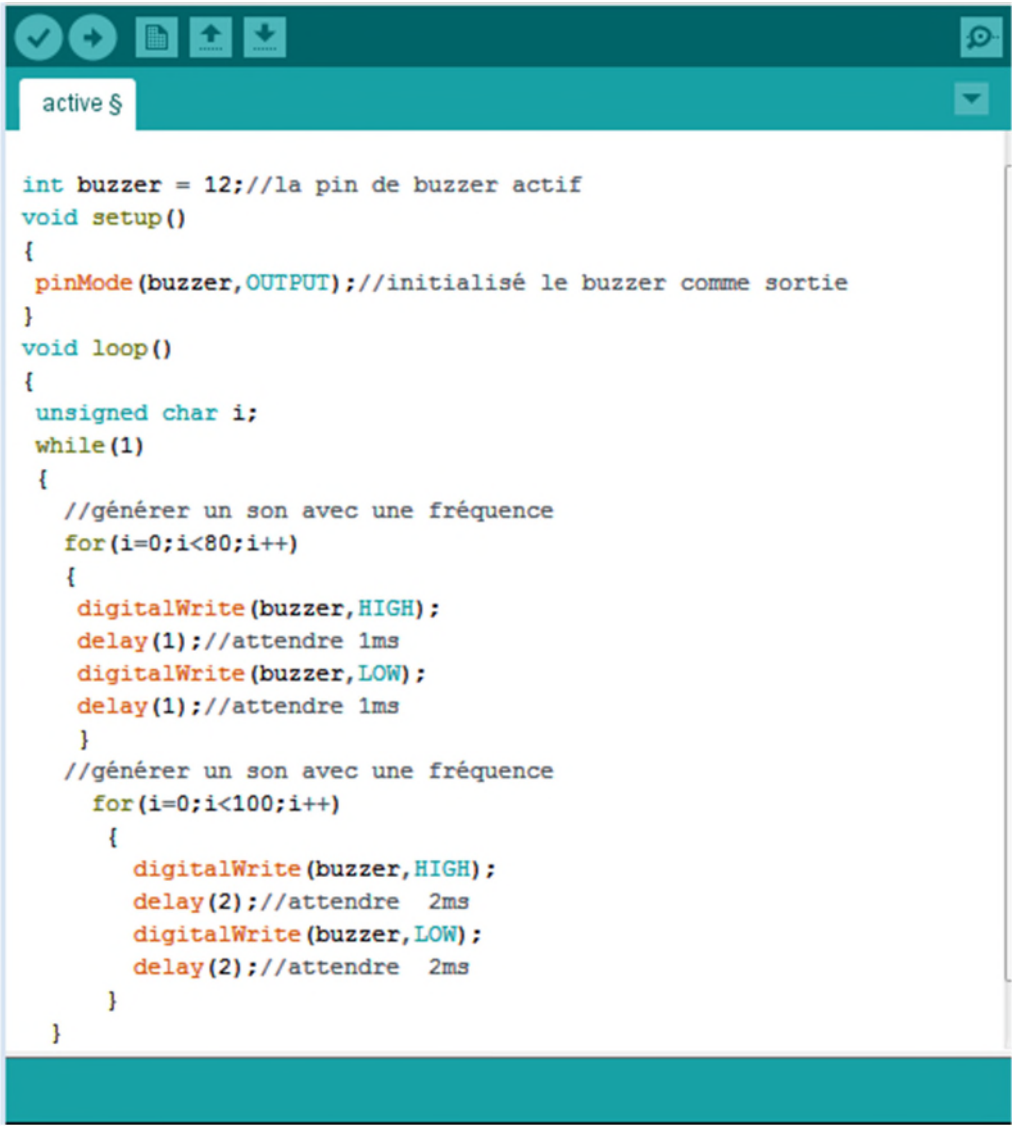


Figure II.17 : Code d'un buzzer actif.

II.6.3. Clavier matriciel :

Description :

Ce clavier comprend 16 touches10 numériques (0-9) et 6 touches marquées # * A B C D, Disposées en 4 lignes et 4 colonnes. L'appui sur une touche fait communiquer une ligne avec une colonne.

Les lignes sont des sorties, les colonnes sont des entrées maintenues au niveau haut par une résistance interne a Arduino.

Le système envoie par balayage un niveau bas sur chaque ligne (1 seule à la fois) et balaye les colonnes en lecture. Quand il lit un niveau bas, c'est que la colonne est reliée par une touche appuyée à la ligne qui est basse à ce moment. Ce balayage est fait par des bibliothèques.

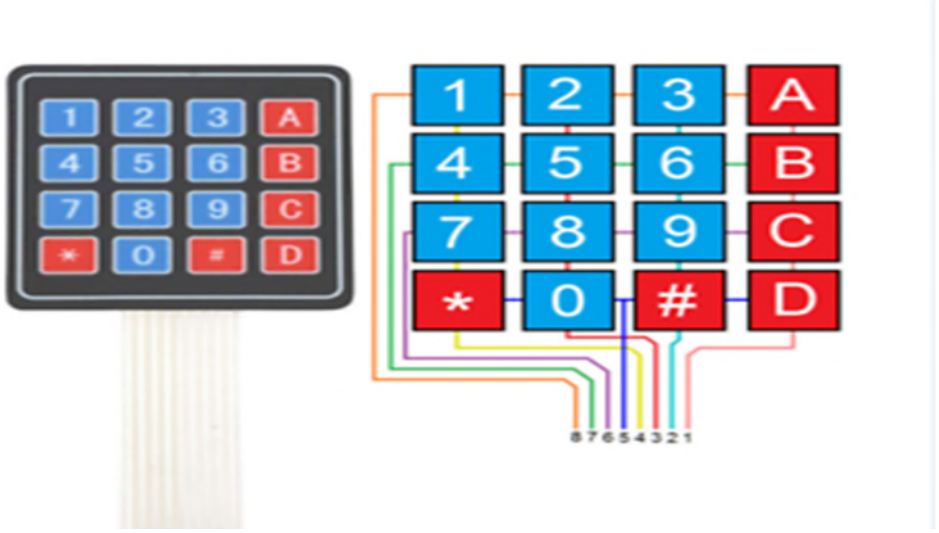


Figure II.18 : Un clavier matriciel [5]

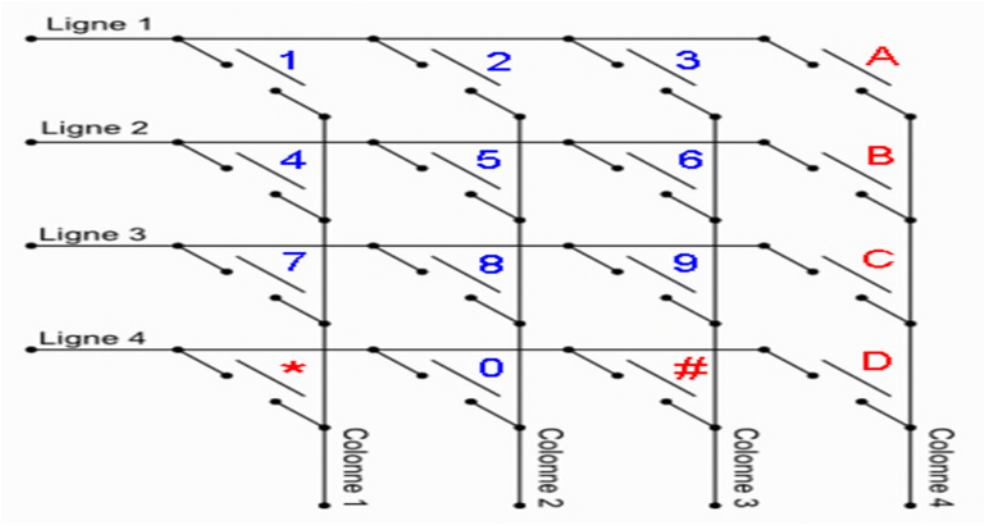


Figure II.19 : Architecture d'un clavier matriciel [5]

Exemple :

Dans cette application nous allons afficher sur le moniteur série les 16 caractères du clavier.

Matériel utilisés

- Une carte Arduino MEGA2560.
- Clavier matriciel.
- Huit câbles male-male.

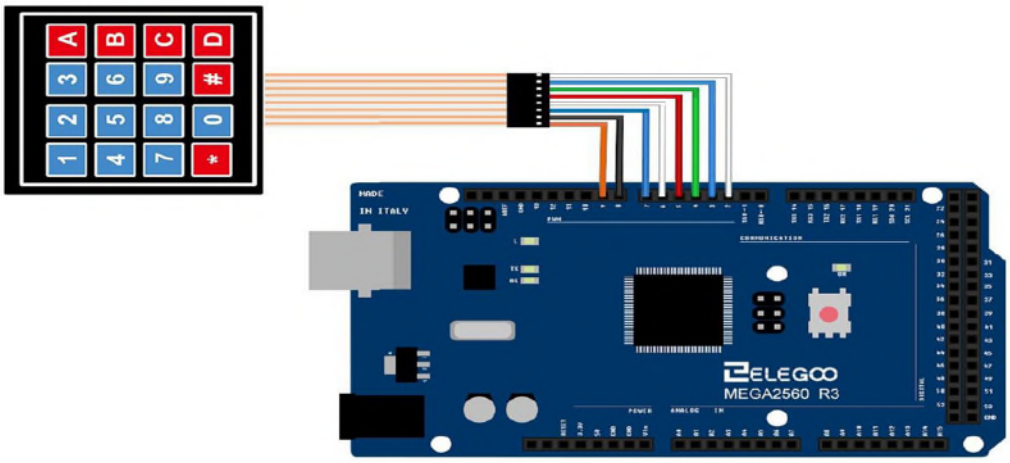


Figure II.20 : Vue prototype d'un clavier matriciel.

Programmation de l'Arduino :

Au niveau de la partie déclarative :

- On inclut les librairies des fonctionnalités utilisées :**<Keypad.h>.**
- Déclaration des constantes utiles pour l'utilisation du clavier 4x4 : nombre de lignes et nombre de colonnes.
- Déclaration des variables globales des touches du clavier 4x4 : les symboles des touches de clavier (1 2 3 4 5 6 7 8 9 * 0 # A B C D).
- Déclaration des variables globales de lignes et de colonnes du clavier 4x4 et de la variable de mémorisation de la touche appuyées :
  customkeypad = Keypad(makeKeymap(matrice),lignesPins, colpins, lignes, col)
  les broches de lignes sont automatiquement configurées en entrée avec pullup interne activé .
  les broches de colonnes sont automatiquement configurées en sortie..

Au niveau de la fonction d'initialisation setup( ) :

- Initialisation de l'utilisation de moniteur série.

Au niveau de la boucle principale, la fonction loop ( ) :

- On commence par lire la touche appuyée :
  demand = customkeypad.getKey() .
- Ensuite, on teste si une touche a été appuyée et on réalise l'affichage de la touche appuyée

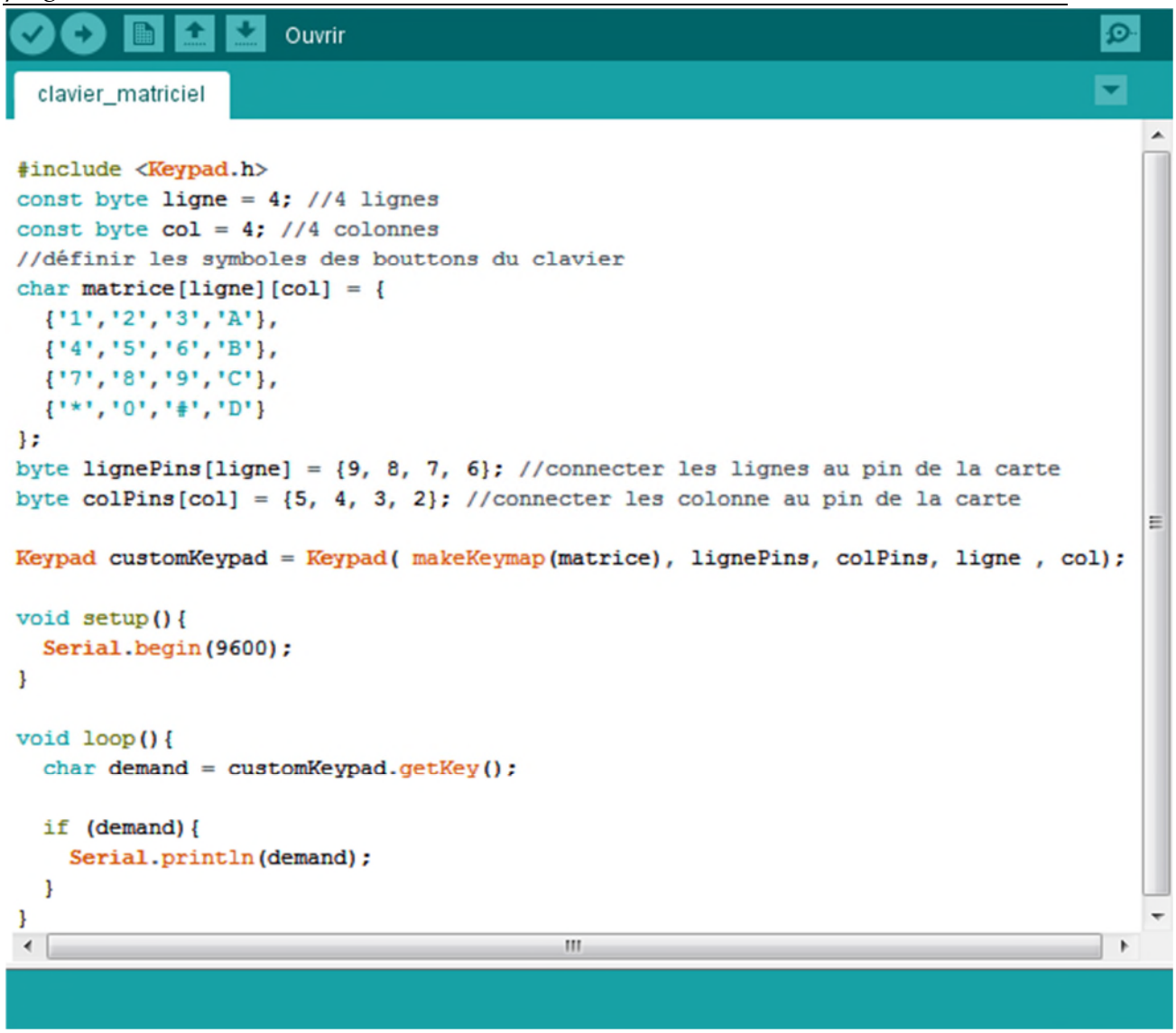


Figure II.21 : Code d'un clavier matriciel.

Le résultat sur le moniteur série est donné par la Figure suivante :

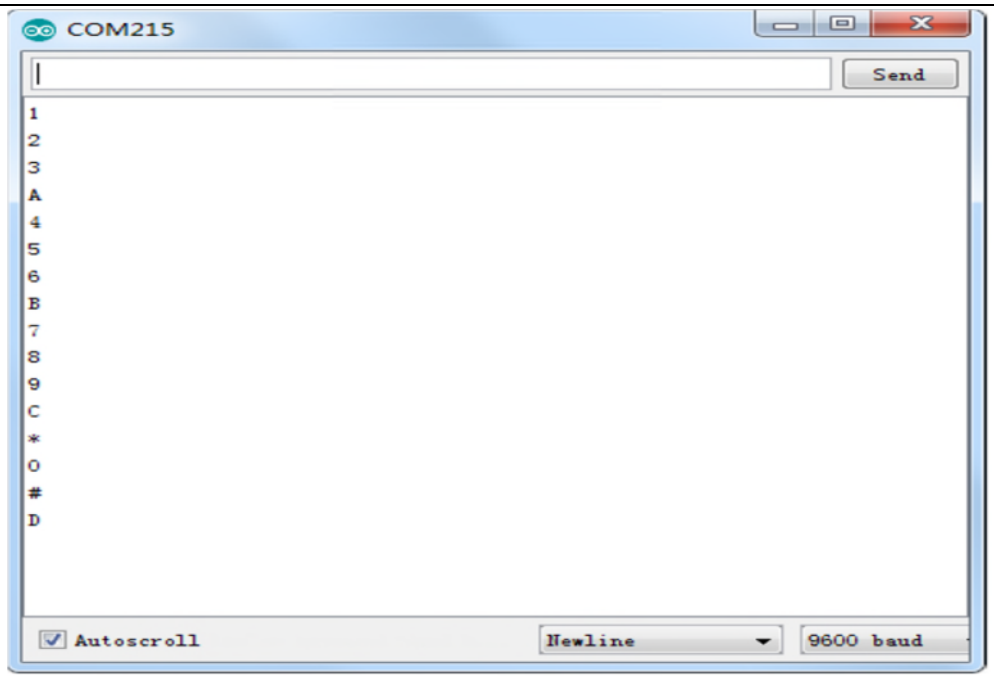


Figure II.22 : Affichage sur le moniteur série.

II.6.4. Ecran LCD

Description :

L'afficheur LCD est en particulier une interface visuelle entre un système (projet) et l'homme (utilisateur). Son rôle est de transmettre les informations utiles d'un système à un utilisateur. Il affichera donc des données susceptibles d'être exploitées par l'utilisateur d'un système. [8]

Fonctionnement :

L'afficheur dispose d'un rétro- éclairage LED et peut afficher deux lignes de 16 caractères, chaque caractère est un rectangle pixel. Il est possible de contrôler chaque pixel de chaque rectangle pour créer des caractères spécifiques.

Figure II.23 : Afficheur LCD16*2. [5]

Exemple :

Dans cette application nous allons afficher sur un écran LCD deux messages sur chaque ligne sous Arduino on utilisant un potentiomètre pour régler le contraste de l'écran.

Matériels utilisés :

- Une carte Arduino méga 2560.
- Une plaque prototype.
- Un potentiomètre 10K.
- Un écran LCD
- Des câbles de connexions.

En partant de la gauche, voici à quoi servent les pins :

- Les deux premiers pins tout à gauche servent à l'alimentation de l'écran.
- La troisième pin est connecté à un potentiomètre et sert pour régler l'affichage (le contraste de l'écran).
- La quatrième, noté RS, est connectée au pin 12 de l'Arduino dans notre exemple. Elle sert à sélectionner la zone mémoire de l'écran LCD dans laquelle nous allons écrire (Register Select).
- La cinquième doit toujours être connectée au ground. C'est un sélecteur de mode lecture ou écriture. On peut la connecter à un pin, mais dans notre cas c'est inutile. Comme il doit recevoir un signal à 0V, on la connecte au ground (état R/W).
- La sixième, noté E, est connectée a la pin 11 de l'Arduino dans notre exemple. elle permet de lancer ou non l'écriture dans les zones mémoires (Enable).
- Les qutre suivantes (reliées au ground) servent pour la communication 8 bits. Pour la communication 4 bits, il est conseillé de les relier au ground. elles représentent les bits de poids fort.
- Les quatre qui suivent, notés D4, D5, D6, D7 se connectent dans notre exemple sur les pins 2, 3, 4, 5 de l'Arduino. elles servent pour la communication (8 bits ou 4 bits) et doivent toujours être connectées. elles représentent les bits de poids faible (ou servent pour envoyer d'abord les bits de poids faible, puis les bits de poids fort)
- Les deux pins tout à droite servent pour alimenter la LED du rétro-éclairage.

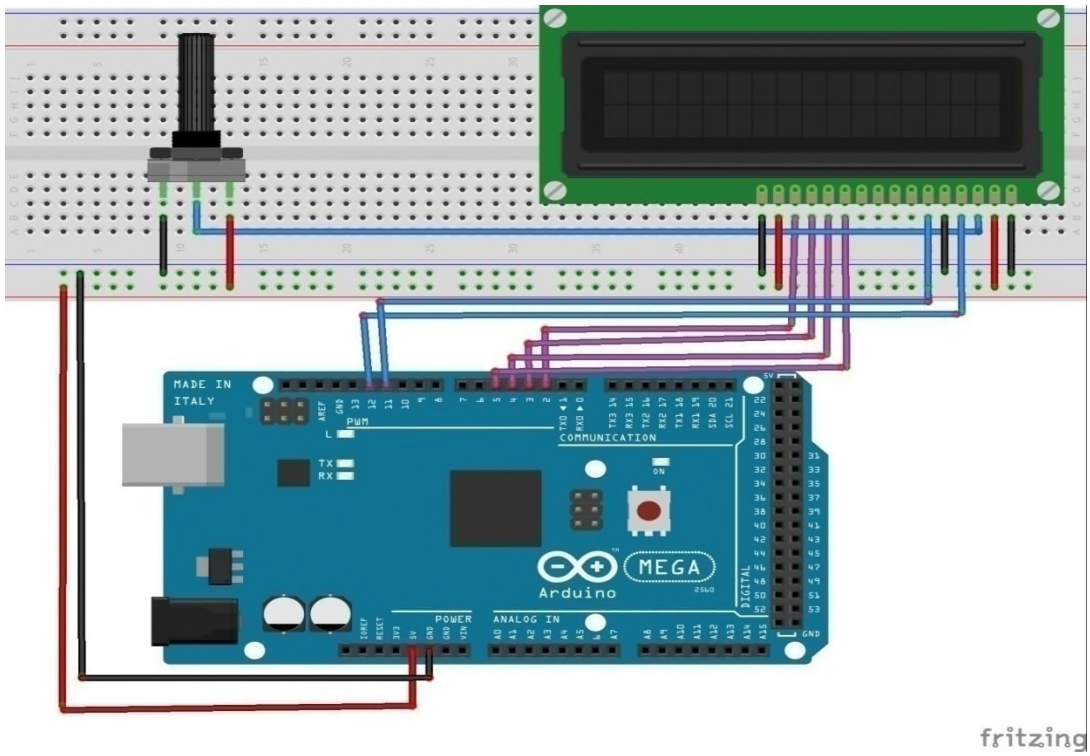


Figure II.24 : Vue prototype du câblage d'un écran LCD

Programmation de l'Arduino :

Nous commençons notre programme en incluant la bibliothèque : **#include <LiquidCrystal.h>.**

Comme pour toute utilisation de bibliothèque, nous allons commencer par définir un objet avec quelques paramètres : **LiquidCrystal lcd(12, 11, 5, 4, 3, 2).**

On initialise la communication entre l'Arduino et l'objet lcd avec une matrice de 16 colonnes et 2 lignes : **lcd.begin(16,2).**

Voici un programme simple qui affiche "**monté, 1er étage**" sur deux lignes :

ecran_LCD

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup(){
lcd.begin(16,2);
}
void loop(){
lcd.setCursor(0,0);
lcd.print("monté");
lcd.setCursor(0,1);
lcd.print("1er étage ");
delay(1000);
}
```

Figure II.25 : Code d'affichage sur un écran LCD

II.6.5. Les détecteurs mécaniques (type fin de course)

Les interrupteurs de positions mécaniques peuvent aussi être appelés (Détecteur de position) et (Interrupteur de fin de course). Ils coupent ou établissent un circuit lorsqu'ils sont actionnés par un mobile. La détection s'effectue par contact d'un objet extérieur sur le levier ou un galet.[9]

Figure II.26 : Interrupteur fin de course. [9]

Utilisation :

Les détecteurs mécaniques de position, appelés aussi interrupteurs de position, sont surtout employés dans les systèmes automatisés pour assurer la fonction détecter les positions. On parle aussi de détecteurs de présence.

Ils sont réalisés à base de microcontacts placés dans un corps de protection et muni d'un système de commande ou tête de commande.

Principe de fonctionnement :

C'est un commutateur, commandé par le déplacement d'un organe de commande (corps d'épreuve).

Lorsque le corps d'épreuve est actionné, il ouvre ou ferme un contact électrique.

De nombreux modèles peuvent être associés au corps : tête à mouvement rectiligne, angulaire ou multi direction associée à différents dispositifs d'attaque (à poussoir, à levier, à tige).

Avantages :

- sécurité de fonctionnement élevée : fiabilité des contacts et manœuvre positive d'ouverture
- bonne fidélité sur les points d'enclenchement (jusqu'à 0,01 mm)
- séparation galvanique des circuits
- bonne aptitude à commuter les courants faibles, combinée à une grande endurance électrique
- tension d'emploi élevée
- mise en œuvre simple, fonctionnement visualisé
- grande résistance aux ambiances industrielles
- détection de tout objet solide
- facilité d'installation

Inconvénient :

- En contact avec le produit à détecter.

II.6.6. Moteur à courant continu :

Description :

Un moteur à courant continu est constitué de deux parties électriques : le stator et le rotor. Lorsqu'on alimente le moteur, il se crée une interaction magnétique qui met le moteur en mouvement. Lorsqu'on inverse le sens de la tension qui alimente le moteur, il tourne en sens inverse. [10]

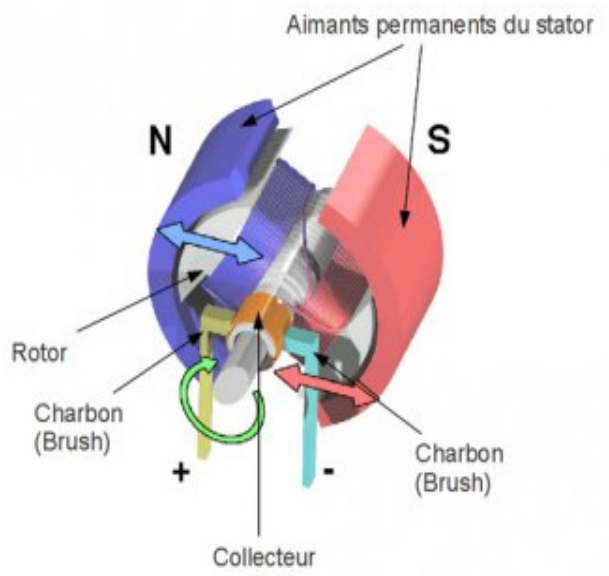


Figure II.27 : Moteur à courant continu. [15]

Principe de fonctionnement :

Le stator :

Le stator d'un moteur à courant continu est la partie fixe du moteur (statique). Le stator est aussi nommé l'inducteur ou l'excitation : on fait passer un courant dans le bobinage du stator et c'est lui qui crée (qui induit) un champ magnétique. Le stator pose le décor pour le rotor qui se retrouve ainsi plongé dans ce champ magnétique. Le stator (inducteur) crée un champ magnétique B appelé champ inducteur, ou encore champ statorique. [10]

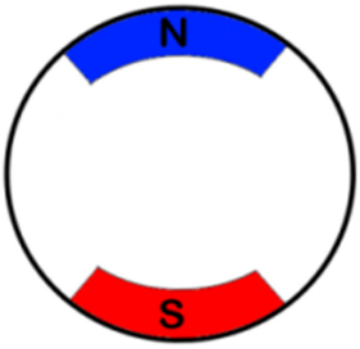


Figure II.28. Stator d'un moteur a courant continu.[15]

Le rotor :

Le rotor est la partie en rotation de moteur, celui qui tourne. Il est constitué du bobinage induit. Il faut alimenter cette bobine pour la transformer en

électroaimant qui entrera en interaction avec le stator.si on n'alimente pas le rotor, il ne serait l'objet d'aucune force et ne tourne pas.

Un système de frottement spécial permet d'alimenter le rotor : des balais ou charbons montés sur des ressorts frottent dur les contacts en rotation : le collecteur. [10]

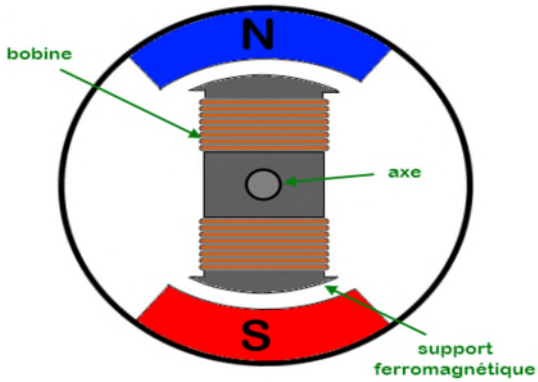


Figure II.29 : Rotor d'un moteur à courant continu. [15]

Le collecteur :

Le collecteur est un ensemble de plages métalliques qui font contact avec les charbons.il appartient au rotor.le frottement des charbons font qu'ils s'usent : ils sont montés sur ressort pour garantir un contact meme lorsqu'ils raccourcissent a cause de l'usure [10]

Exemple :

Faire tourner un moteur courant continu dans un sens, puis dans un sens inverse sous Arduino.

Figure II.30 : Moteur courant continue 3-6v. [5]

Adaptateur :

Le moteur nécessite plus de puissance d'alimentation que la carte MEGA2560 R3 est en mesure de fournir. C'est pour cela que nous avons besoin d'une alimentation séparé. [5]

Figure II.31 : Adaptateur pour moteur à courant continu. [5]

Pont hacheur (L293) :

C'est un composant très courant et utile pour commander des moteurs dans son principe de base, le pont H est un assemblage de 4 transistors (2 PNP et 2 NPN) monté de telle façon que le courant puisse passer soit dans un sens, soit dans l'autre au travers de la charge (un moteur courant continu par exemple). [5]

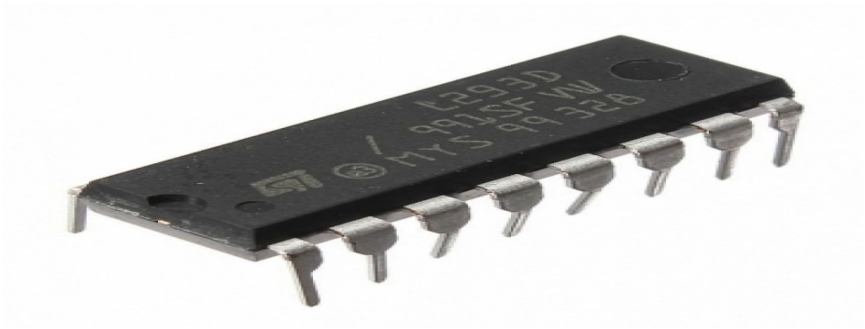


Figure II.32 : Adaptateur: Pont-hacheur L293D. [5]

Matériel utilisés :

- Une carte Arduino MEGA2560 R3
- Une planche prototype
- un circuit intégré L293D
- un moteur 3-6v
- cinq câbles male-male

- Alimentation
- Adaptateur 9v 1A7

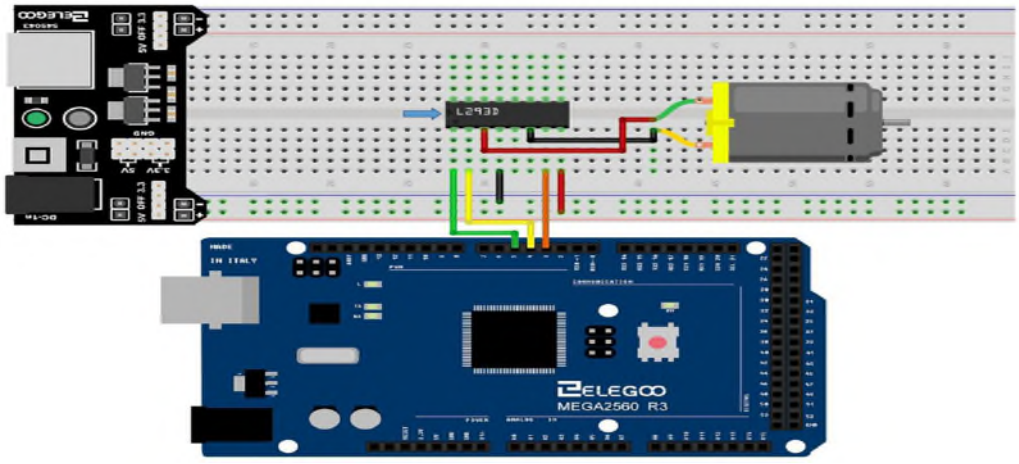


Figure II.33 : Vue prototype d'un MCC.

Programmation :

Au niveau de la partie déclarative : on déclare les variables utilisé dans notre programme.

Au niveau de la fonction initialisation setup() :on configure les variables comme entrées sorties et on initialise le moniteur série.

Au niveau de la boucle principale, la fonction loop ( ) :on fait tourner le moteur dans un sens pendant 5 seconde puis dans un autre sens pendant 5 seconde et on affiche le sens de rotation du moteur sur le moniteur série.

DC_Motor §

```
#define active 5
#define A 3
#define B 4

int i;
void setup() {
  pinMode(active,OUTPUT);
  pinMode(A,OUTPUT);
  pinMode(B,OUTPUT);
  Serial.begin(9600);

void loop() {
    Serial.println("un sens, apres l'inverse");
    digitalWrite(active,HIGH);
    for (i=0;i<5;i++) {
    digitalWrite(A,HIGH); //un sens
    digitalWrite(B,LOW);
    delay(500);
    digitalWrite(A,LOW);  //l'inverse
    digitalWrite(B,HIGH);
    delay(500);
  }


}
```

Figure II.34 : Code pour Moteur à courant continu

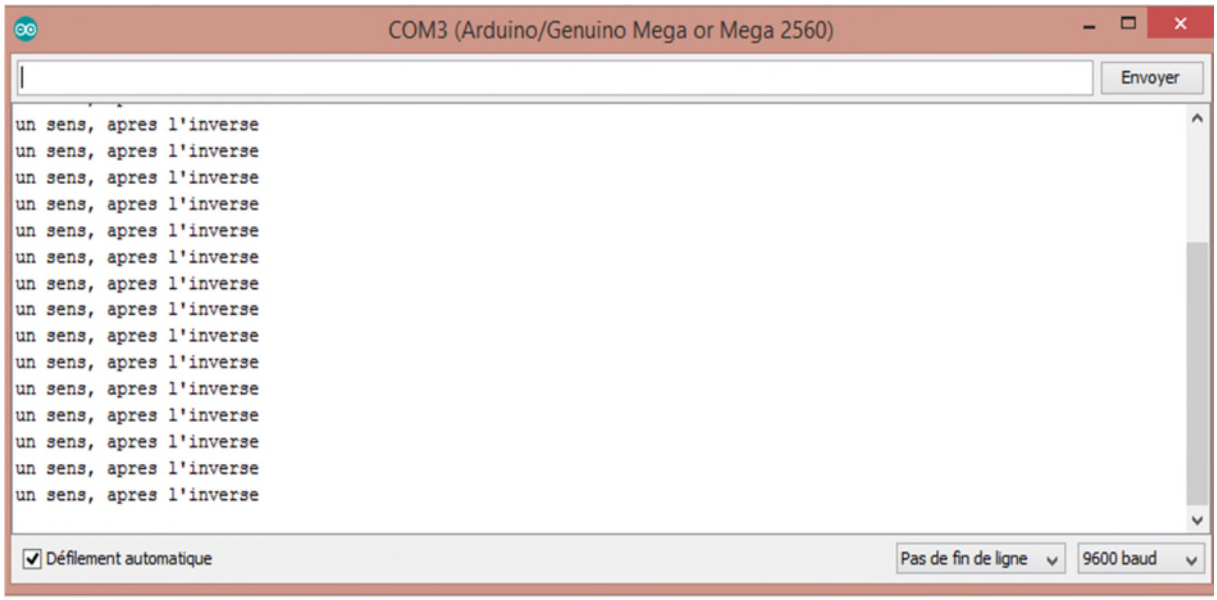


Figure II.35 : Résultat obtenu de la compilation.

II.7. Conclusion

Dans ce chapitre nous avons expliqué en premiers lieux les deux parties essentielles d'Arduino :

Partie matériel : en citant ses caractéristiques, la composition particulière d'Arduino Méga 2560.

Partie logiciel : ou on a décrit le logiciel de programmation Arduino.

En deuxième lieu nous avons donné une description de la plateforme fritzing sur laquelle on a fait des visualisations virtuelles de nos câblages.

A la clôture de ce chapitre nous avons donné quelques concepts théoriques et nous avons expliqué quelques composants .Le bon fonctionnement de les matériaux cités ici est essentiel pour le fonctionnement global de notre système.

Dans le chapitre suivant nous allons présenter, en détail, la commande réalisée.

Chapitre III : Etude et conception de la commande

III.1. Introduction :

Ce chapitre présente une étude conceptuelle et étude de commande qui est le cœur de notre projet, en premier lieu nous allons faire une présentation globale de notre ascenseur, ensuite on va décrire la maquette où nous allons traiter le cahier de charge ainsi qu'un organigramme de son fonctionnement.

D'un point de vue d'automaticien, l'étude d'un système automatique se décompose en deux parties : une partie opérative et une partie de coopération dépend de la bonne interaction synchrone ou asynchrone entre les divers partie du système (capteurs, actionneurs, commande, etc.).

Dans un ascenseur, l'ensemble électromécanique (cabine, moteur, porte,…) constitue la partie opérative, la logique (programme) constitue la partie commande, les boutons d'appels et les fins de course pour la détection de présence ainsi que les LEDs témoins constituent les entrées/sorties du système de commande.

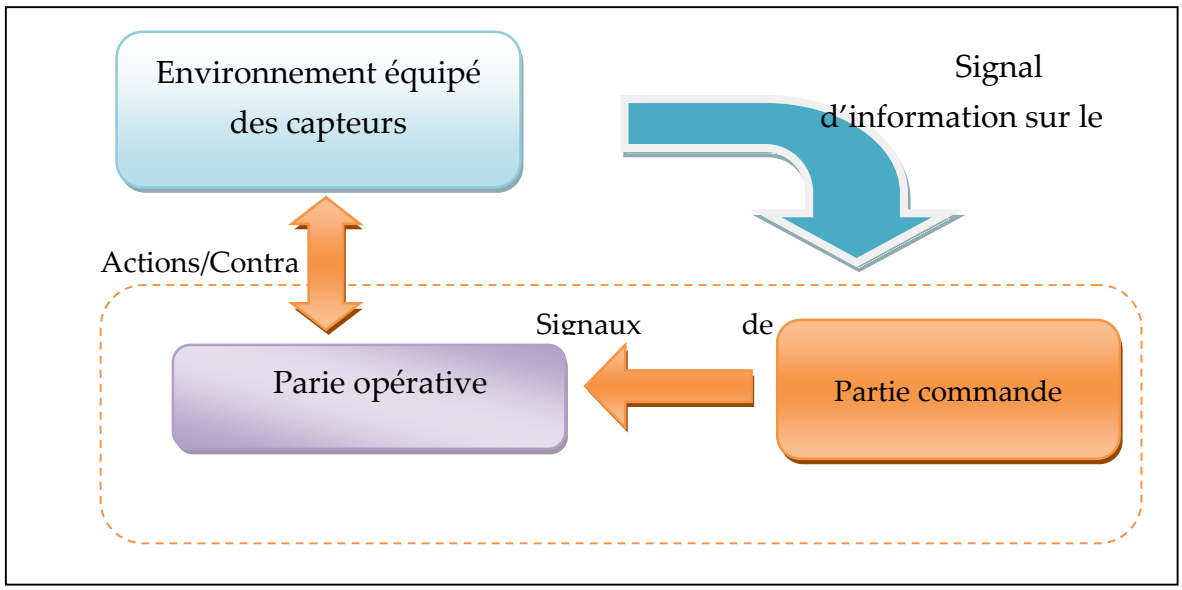


Figure III.1 : La structure de fonctionnement de l'ascenseur

III.2. Présentation de l'ascenseur commandé

C'est une maquette utilisée dans le laboratoire, elle comporte une cabine entrainée par un moteur électrique à courant continu à l'aide d'une courroie d'entrée en plastique. La monté et la descente de la cabine se font par le changement du sens de rotation du moteur.

Figure III.2 : La maquette de l'ascenseur

III.2.1. Spécification de l'ascenseur

- Dimension
- Hauteur 1.35m.
- Section 235mm*235mm
- 4 niveaux

- Structure en plexiglas fumé avec armature métallique.
- Détection de chaque étage et de l'ouverture des portes
- Signalisation par une LED a chaque niveau.
- Lampe d'allumage cabine.
- Poussoirs d'appel à chaque étage. [19]

III.2.2. Les entrées sorties

Les entrées/sorties de la maquette sont :

III.2.2.1. Les signaux de sortie

Sens de rotation : ce sont les deux signaux de rotation du moteur (sens1, sens2) pour commander la monté ou la descente de la cabine.

Allumage cabine : c'est un voyant qui permet de commander l'allumage de la cabine à l'intérieur de la cage.

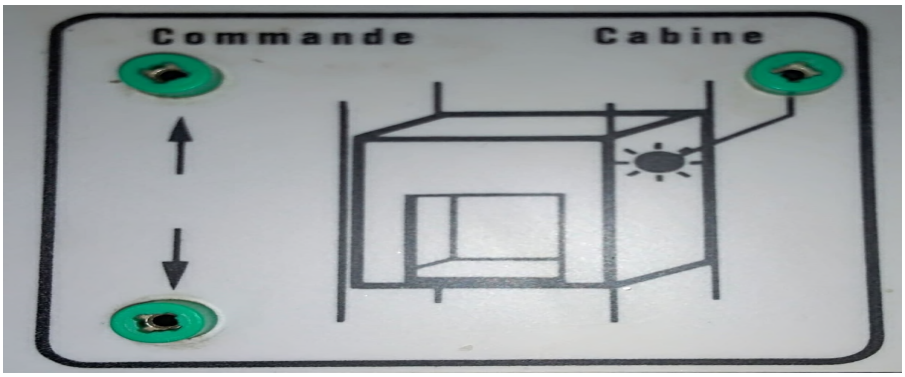


Figure III.3 : Les deux sens de la cabine et son allumage.

Les témoins : ce sont4 voyants situés aux quatre étages de la cage (LED1, LED2, LED3 LED4)

Figure III.4 : Les quatre témoins de l'ascenseur.

III.2.2.2. Les signaux d'entrée

Boutons d'appels : 4 poussoirs sont situées aux 4 étages de l'ascenseur pour appeler cette dernière (appel1, appel2, appel3, appel4)

Boutons d'étage : 4 poussoirs sont situées a l'intérieur de la cabine pour se rendre à l'étage désiré (étage 1, étage 2, étage 3, étage 4).

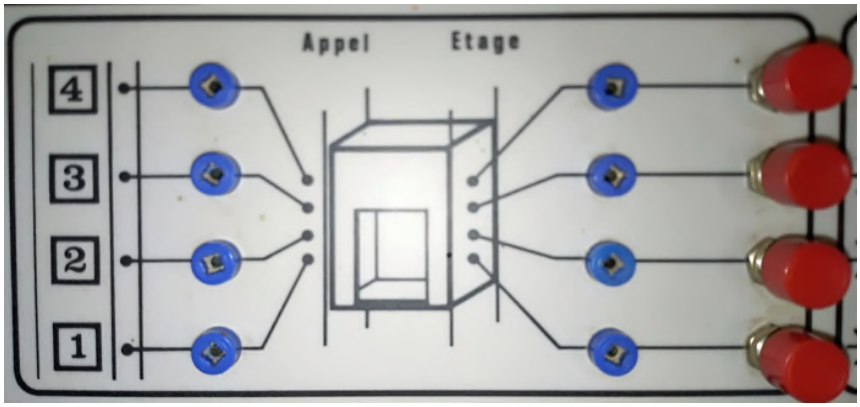


Figure III.5 : Les boutons d'appels et d'étages.

Arrêt d'urgence : un poussoir qui permet de bloquer l'ascenseur en cas d'anomalie.

Figure III.6 : Arrêt d'urgence.

Test porte : l'ascenseur est équipé de 4 capteurs qui indiquent l'ouverture de l'un des portes situées à chaque étage.

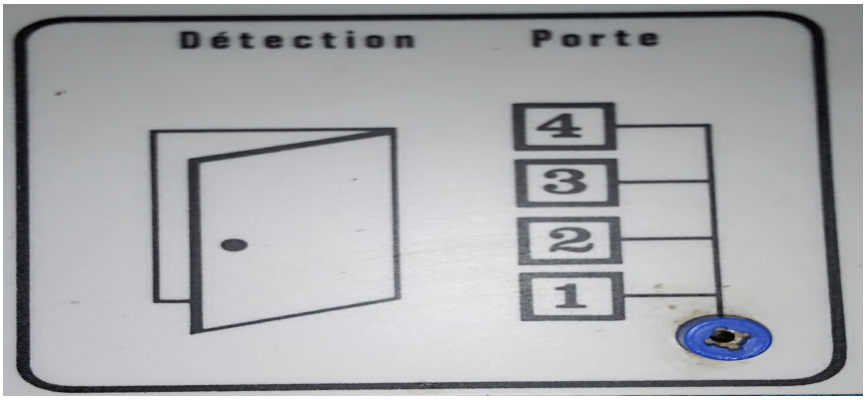


Figure III.7 : Teste de fermeture/ouverture des portes.

Présence à l'étage : ce sont des voyants qui indiquent la position de la cabine.

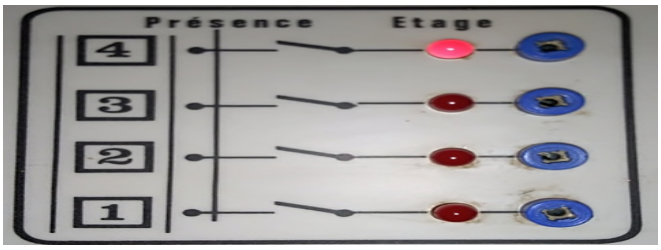


Figure III.8 : Témoins de présence à un étage

Limite haut et bat : ce sont des capteurs de positions (limite haute et limite basse) qui indiquent le dépassement de la cabine de sa limite.

III.3. Commande de l'ascenseur

III.3.1. Cahier de charge de l'ascenseur

Le fonctionnement de l'ascenseur doit être décris par des règles bien précises. Les pannes et incendies qui peuvent menacer l'utilisateur doivent être pris en considération dans la conception de n'importe quel ascenseur.

- A la mise sous tension, la position de repos de l'ascenseur est le réez de chaussé, Le capteur de l'état bas est actif et tous les autres capteurs sont désactivés. Dès qu'il y a une demande, il doit se rendre à l'étage demandé a condition que les portes soient fermées.
- Toutes les requêtes des utilisateurs de l'ascenseur sont signalées par des voyants lumineux.
- Le trafic de l'ascenseur proposé est optimisé par un algorithme de gestion de priorité :

En premier lieu l'ascenseur est en train de monter, il doit s'arrêter à tous les étages où il y a une demande extérieure de monter.

En deuxième lieu l'ascenseur est en train de descendre, il doit s'arrêter à tous les étages où il y a une demande extérieure de descendre.

- A chaque fois, qu'il y a un changement de position le numéro de l'étage et l'état de la cabine (monté, descendre, arrêt) s'affichent sur l'écran LCD.
- L'ascenseur termine toujours sa direction avant de changer.

- Lorsque l'ascenseur arrive à un étage demandé, les portes s'ouvrent, si les portes ont été ouvertes, après un délai de 5secondes elles doivent se refermer.
- L'ascenseur ne doit pas se mettre en marche si les portes sont ouvertes. Lors d'une demande de l'intérieur, l'ascenseur doit se mettre en marche après que les portes se soient fermées (après un délai).
- Si le bouton d'urgence est actionné une alarme se déclenche immédiatement.
- Si le bouton d'arrêt d'urgence est actionné le système s'arrête.

III.3.2. Conception de la maquette

Cette maquette permet de simuler le fonctionnement réel d'un ascenseur à 4 niveaux (R+3). Qui se compose de :

Un moteur courant continu qui actionne le sens de mouvement de la cabine (monter / descendre).

4 capteurs de position qui indique la position de la cabine une fois qu'il contact entre la cabine et l'un de ces capteurs.

Un clavier constitué de boutons poussoirs de chaque niveau :

6 boutons de l'extérieur (aux différents étages) pour envoyer un appel de monter ou de descendre et 4 à l'intérieur de la cabine qui représentent les 4 niveaux (étages).

10 voyants lumineux signalent à l'utilisateur la prise en charge de la demande, jusqu'à l'arrivé à l'étage demandé (exécution de la demande).

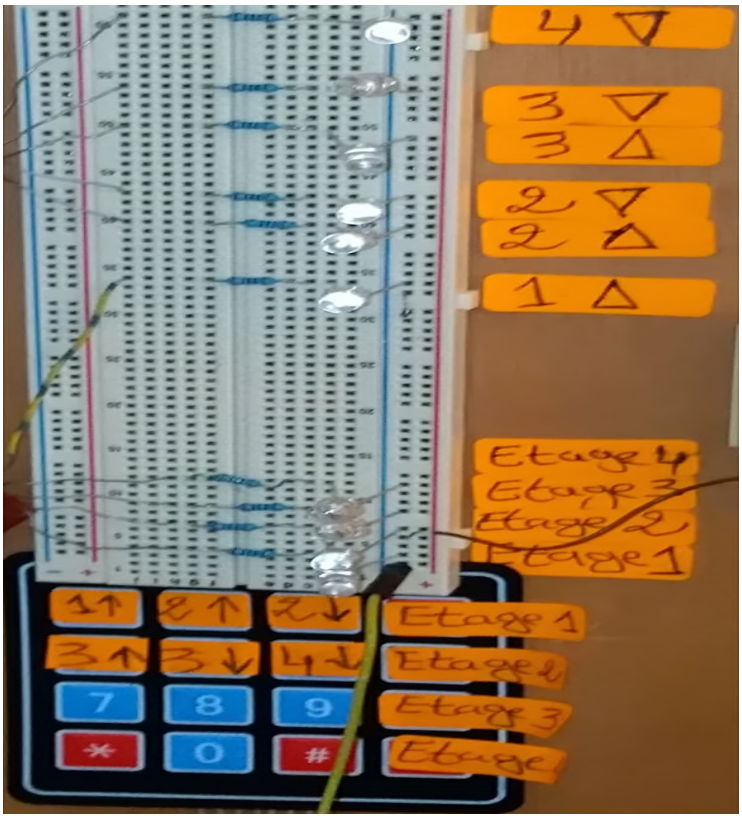


Figure III.9 : Maquette des buttons d'appels.

Un afficheur LCD a l'intérieur de la cabine indique la position et la direction du mouvement de la cabine (monter/ descendre).

Figure III.10 : Afficheur LCD.

Un bouton d'alerte : en cas d'anomalie l'utilisateur appuie sur le bouton une LED s'allumera et une alarme se déclenche (buzzer).

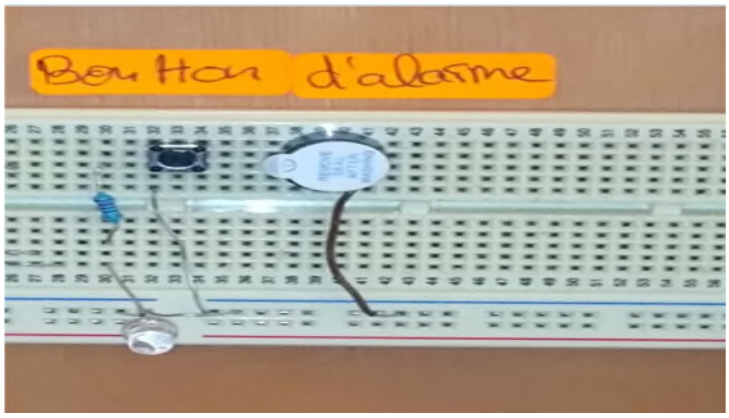


Figure III.11 : Bouton d'alarme.

Détecteur de porte : qui détecte l'ouverture et la fermeture de la porte.

Interrupteur d'arrêt d'urgence : Si le bouton d'arrêt d'urgence est actionné le système s'arrête.

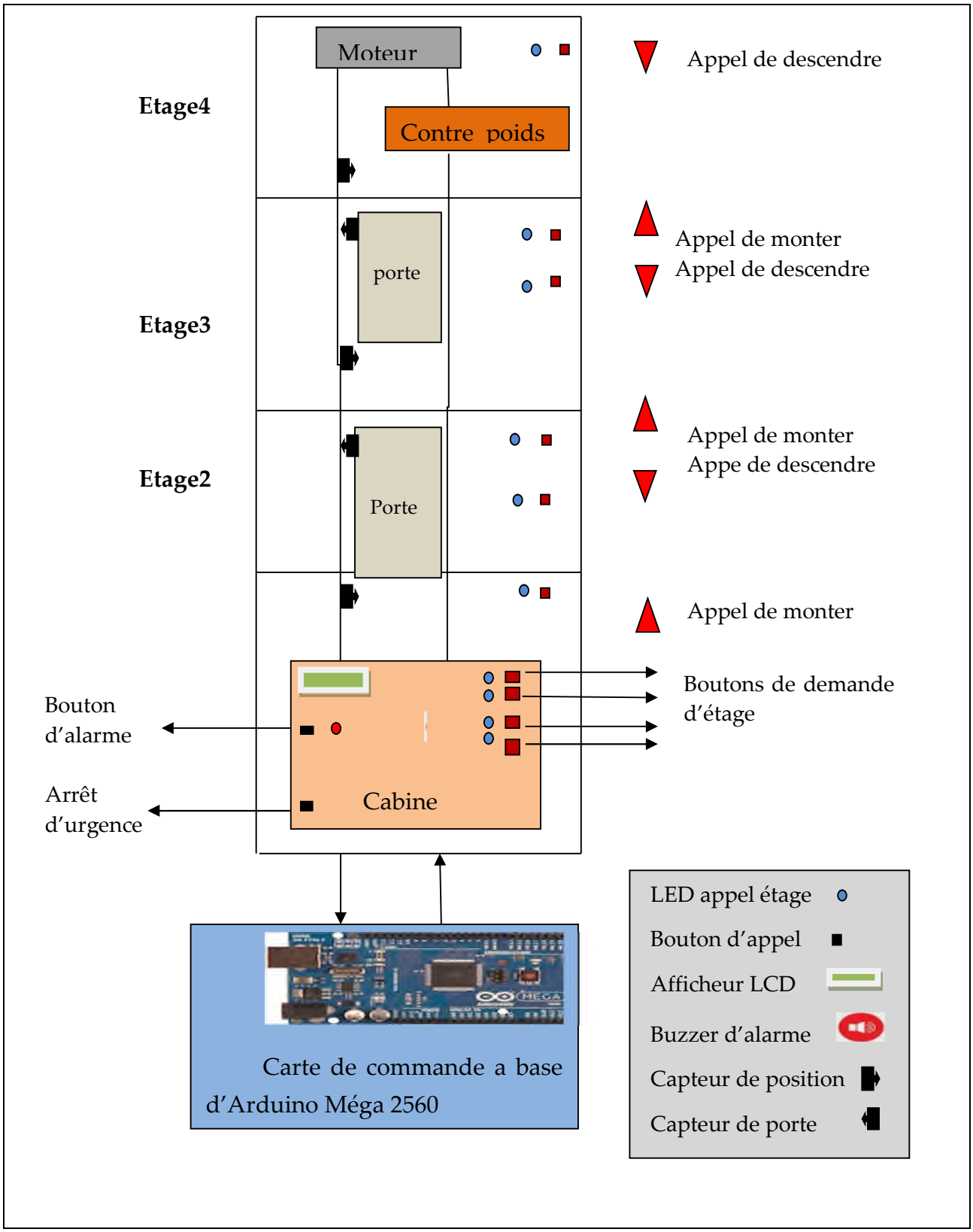


Figure III.12 : Schéma fonctionnel de l'ascenseur.

III.3.3. Organigramme de fonctionnement

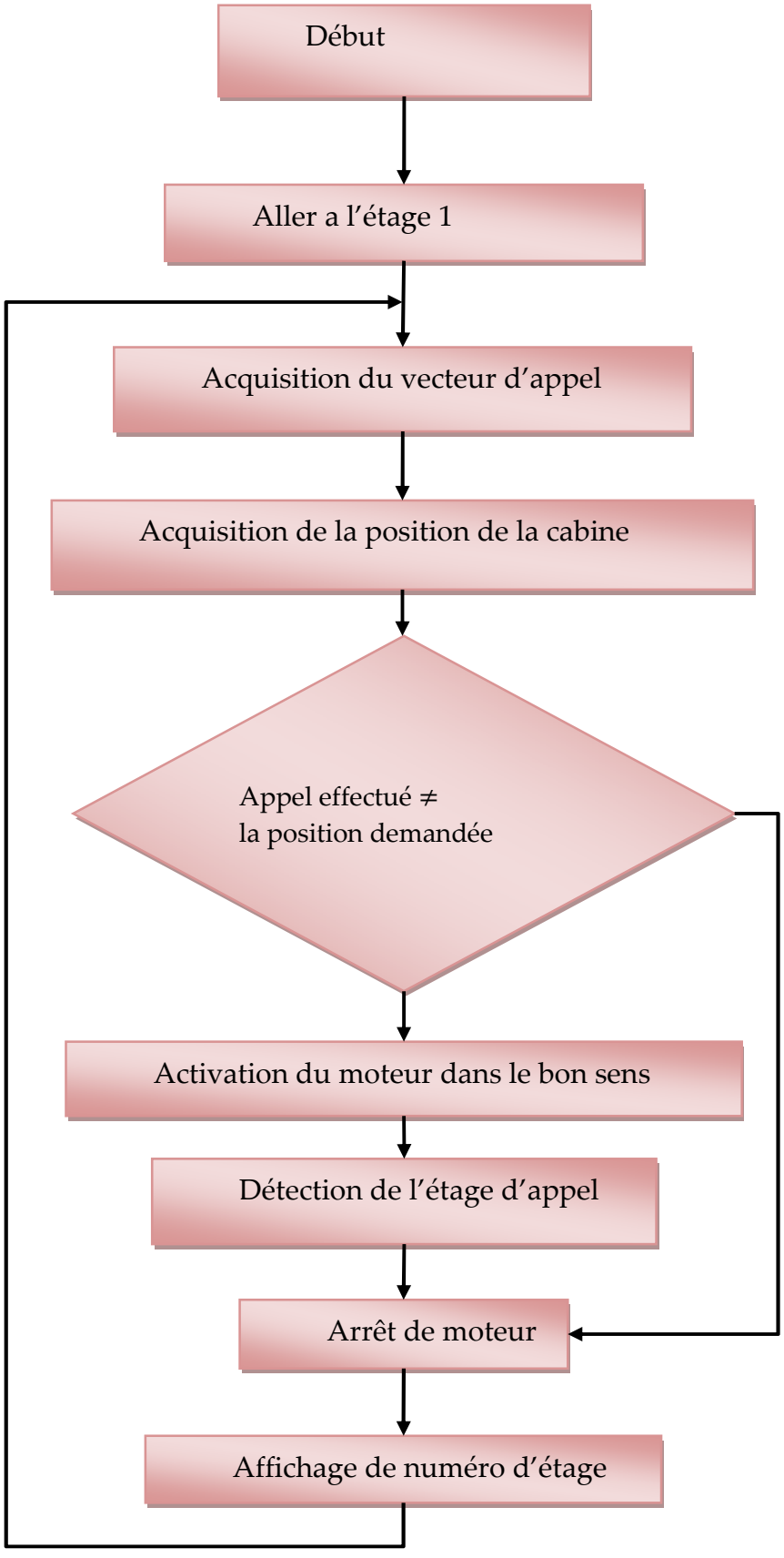


Figure III.13 : Organigramme de fonctionnement.

III.3.4. Structure du programme

Le programme développé est composé de plusieurs parties qui font appel à une multitude de fonctions (voir Annexe 2), nous expliquons dans ce qui suit quelques exemples de déroulement du programme.

Au niveau de la partie déclarative :

- On inclut les bibliothèques nécessaires pour l'utilisation du clavier et l'afficheur LCD.

```
#include <Keypad.h>
#include <LiquidCrystal.h>
```

- On déclare toutes les variables utilisées (moteur, clavier, LEDs, capteurs de positions, buzzer, afficheur, porte, bouton d'arrêt d'urgence).

Au niveau de la partie initialisation fonction setup() :

- On configure toutes variables citées précédemment comme entrées ou sorties.
- On initialise le moniteur série et l'afficheur LCD.
- A fin d'envoyer la cabine a l'étage 1 juste âpres la mise sous tension en utilise le sous programme ci-dessous :

```
 if   (demandvect[0]==LOW && demandvect[1]==LOW && demandvect[2]==LOW
       && demandvect[3]==LOW && demandvect[4]==LOW && demandvect[5]==LOW
       && demandvect[6]==LOW && demandvect[7]==LOW && demandvect[8]==LOW && demandvect[9]==LOW)
{while(pos!=1 && digitalRead(porte)==1 )
   {   descendre();
       pos=lirepos();
       etagecabine(pos);
   }
   arreter();
}
```

- Dans le sous programme précédent on a utilisé une fonction qui permet de récupérer la position de la cabine a partir des capteurs de fin de course.

```
int lirepos()
{
  for(int i =0; i<4;++i)
  {
   if(digitalRead(captetage[i])==HIGH)    pos = i+1;
  }
     Serial.print("La cabine est à l'étage :  ");
     Serial.println(pos);
     etagecabine(pos);
     return pos;

}
```

Au niveau de la partie principale, fonction loop () :

Au repos, avant l'actionnement du moteur, on récupère tout d'abord la position de la cabine avec la fonction cité précédemment, ensuite on récupère le vecteur d'appels (appel de monter, appel de descendre, aller à l'étage désiré) effectué par l'utilisateur.

```
char demand = customKeypad.getKey();
    switch (demand)
{
case '1' : // appel de monté de l'etage 1
{   demandvect[0]=1;
    Serial.println(demandvect[0]);
    digitalWrite(ledboutton1up,HIGH);
    break;
}
  case '2' ://appel de monter de l'etage2
{   demandvect[1]=1;
    Serial.println(demandvect[1]);
    digitalWrite(ledboutton2up,HIGH);
    break;
}
 case '4' : //appel de monté  de l'etage 3
{   demandvect[3]=1;
    Serial.println(demandvect[3]);
    digitalWrite(ledboutton3up,HIGH);
    break;
}
case '3' : //appel de descendre de l'etage 2
{   demandvect[2]=1;
    Serial.println(demandvect[2]);
    digitalWrite(ledboutton2down,HIGH);
    break;
}
```

```
case '5' : //appel de descendre de l'etage 3
{   demandvect[4]=1;
    Serial.println(demandvect[4]);
    digitalWrite(ledboutton3down,HIGH);
    break;
}
case '6' : //appel de descendre de l'etage 4
{   demandvect[5]=1;
    Serial.println(demandvect[5]);
    digitalWrite(ledboutton4down,HIGH);
    break;
}
case'A': // aller vers l'etage 1
{
  demandvect[6]=1;
  Serial.println(demandvect[6]);
  digitalWrite(ledboutton1,HIGH);
   break;
}
case'B':// aller vers l'etage 2
{
    demandvect[7]=1;
    Serial.println(demandvect[7]);
    digitalWrite(ledboutton2,HIGH);
     break;
}

case'C'://aller vers l'etage 3
{
  demandvect[8]=1;
  Serial.println(demandvect[8]);
  digitalWrite(ledboutton3,HIGH);

   break;
}
case'D'://aller vers l'etage 4
{
  demandvect[9]=1;
  Serial.println(demandvect[9]);
  digitalWrite(ledboutton4,HIGH);

       break;

}
}
```

Notre programme principal se décompose en trois parties essentielles :

- Cabine au repos : une fois qu'un appel est effectué par un utilisateur le moteur s'actionne dans un sens (monter, descendre) selon la position de l'utilisateur à condition que les portes soient fermées.
- Une fois que la cabine est en marche :

Si elle est traine de monter :

- ✓ Il ya un seul appel de monter, elle le prend en charge et elle s'arrête a la position désirée.
- ✓ Il ya un seul appel de descendre elle le prend en charge et elle s'arrête a la position désirée.
- ✓ Il ya plusieurs appels, elle prend en charge tout d'abord tous les appels de monter au dessus ensuite elle change de direction pour répondre aux autres appels.

Exemple :

```
if (digitalRead(mup)==1 && digitalRead(mdown)==0  )
{
  if (pos==2)
  {
    // demande ou appel de monter  a l'etage 2 uniquement
     if ((demandvect[1]==HIGH ||demandvect[7]==HIGH)  && (demandvect[3]==LOW
     && demandvect[4]==LOW &&demandvect[8]==LOW && demandvect[5]==LOW && demandvect[9]==LOW))
     {  arreter();
        ouverture_porte();

        //mise a 0 des appels et des LEDS
        demandvect[1]=LOW; digitalWrite(ledboutton2up,LOW );
        demandvect[7]=LOW; digitalWrite(ledboutton2,LOW );

        //arreter la cabine pendant 5s
         delay(5000);fermeture_porte() ;
     }
      //prise en charge:de l'appel de monté de l'etage 2 et tout les appels de monté qui sont  au dessus
     if (demandvect[1]==HIGH|| demandvect[7]==HIGH)
    {  arreter();
       ouverture_porte();
       //mise a 0 des appels et des LEDS
        demandvect[1]=LOW;digitalWrite(ledboutton2up,LOW );
        demandvect[7]=LOW;digitalWrite(ledboutton2,LOW );
```

```
    //arreter la cabine pendant 5s
    delay(5000);

    if(digitalRead(porte)==1)
    {fermeture_porte() ;
      monter();
    }
}
// demande ou appel de dscendre  a l'etage 2 uniqument
  if ((demandvect[2]==HIGH ||demandvect[7]==HIGH) && (demandvect[3]==LOW
  && demandvect[4]==LOW && demandvect[8]==LOW && demandvect[5]==LOW && demandvect[9]==LOW   ))
{   arreter();
    ouverture_porte();
    //mise a 0 des appels et leds
    demandvect[2]=LOW; digitalWrite(ledboutton2down,LOW );
    demandvect[7]=LOW; digitalWrite(ledboutton2,LOW );
    //arret la cabine pendant 10s
    delay(5000);
    fermeture_porte() ;
}

}
```

Si elle est en train de descendre :

- ✓ Il ya un seul appel de descendre, elle le prend en charge et elle s'arrête à la position désirée.
- ✓ Il ya un seul appel de monter, elle le prend en charge et elle s'arrête a la position désirée.
- ✓ Il ya plusieurs appels, elle prend tout d'abord touts les appels de descendre ensuite au dessous puis elle change de direction pour répondre aux autres appels.

Exemple :

```
if (digitalRead(mup)==0 && digitalRead(mdown)==1 )
{
  if (pos==2  )

    {
      //demande ou appel de descendre   a l'etage 2 uniqument
     if ((demandvect[2]==HIGH ||  demandvect[7]==HIGH) && (demandvect[0]==LOW && demandvect[6]==LOW))
    {
       arreter();
       ouverture_porte();
       //mise a 0 des appels et leds
       demandvect[2]=LOW;digitalWrite(ledboutton2down,LOW);
       demandvect[7]=LOW;digitalWrite(ledboutton2,LOW);

       //arreter le moteur pendant 5s
       delay(5000);

       fermeture_porte() ;


    }
```

```
//prise en charge:de l'appel de descendre  de l'etage 2 et tout les appels de descendre qui sont  au dessous
   if (demandvect[2]==HIGH || demandvect[7]==HIGH)
{
   arreter();
   ouverture_porte();
   demandvect[2]=LOW; digitalWrite(ledboutton2down,LOW);
   demandvect[7]=LOW;digitalWrite(ledboutton2,LOW);
   delay(5000);
   if(digitalRead(porte)==1)
   {
   fermeture_porte() ;
   descendre();
   }
}
//demande ou appel de monter a l'etage 2 uniqument
 if ((demandvect[1]==HIGH ||  demandvect[7]==HIGH) && (demandvect[0]==LOW && demandvect[6]==LOW))
 {
   arreter();
   ouverture_porte();
   //mise a 0 des appels et leds
   demandvect[1]=LOW;digitalWrite(ledboutton2up,LOW);
   demandvect[7]=LOW;digitalWrite(ledboutton2,LOW);

   //arreter le moteur pendant 5s
   delay(5000);

   fermeture_porte() ;
 }}
```

- On a cité précédemment dans le cahier de charge quelque mesure de sécurité, qu'on a utilisée pour notre ascenseur. Voici les fonctions qui les correspondent dans le programme :

```
void appel_durgence()
{
 if(digitalRead(boutton)==LOW)
 {
 digitalWrite(LED,HIGH);
 digitalWrite(buzzer,LOW);
 delay(1000);
 digitalWrite(LED,LOW);
 digitalWrite(buzzer,HIGH);
 delay(4000);
 digitalWrite(buzzer,LOW);
 }
}
```

```
//arret d'urgence
  if (digitalRead(button_darret_durgence)==HIGH)
   digitalWrite(mup,LOW);
   digitalWrite(mdown,LOW);
```

Toutes les fonctions et les sous programmes cités précédemment assurent le bon fonctionnement de l'ascenseur.

III.4. Conclusion

La maquette constitue un élément essentiel dans notre étude. A l'heure où nous rédigeons ce mémoire, les tests de la maquette étaient en cours. Ces tests concernent principalement la commande du moteur et la priorité des appels. Le reste des fonctions du programme principale ont étaient vérifiées sur la carte Arduino décrite au chapitre 2. Il est important de signaler que la structure mécanique de notre maquette nous a posées des difficultés de réalisation.

Conclusion générale

Ce projet nous a permis de développer nos connaissances et de mettre en application notre savoir-faire pour la concrétisation d'un projet de fin d'études. La recherche bibliographie qui a été riche en découverte sur la théorie et l'industrie des ascenseurs. Le développement des ascenseurs exige la collaboration de spécialistes de différents horizons. Les recherches actuellement et dans un proche future sont axées sur la supervision de batteries d'ascenseurs : il s'agit de transporter le maximum de personnes avec le minimum de risque tout en optimisant le temps des trajets.

En ce qui nous concerne, nous avons entamés notre travail par une carte de simulation basée sur Arduino. Ce choix est dicté par la disponibilité du matériel. Malheureusement avec la complexité de nos programmes (en taille et en nombres de fonctions) et en l'absence de logiciel de débogage, il est devenu très difficile pour nous d'ajouter d'autres fonctions à notre programme tel que le système d'interruption.

Nous avons mis au point des fonctions de gestion d'un ascenseur, une maquette a été réalisée elle servira pour la suite de ce projet.

En perspective, ce projet constitue un domaine d'études et de développement en master aussi bien en informatique industrielle, en automatique ou en instrumentation. Il est possible de poursuivre ce travail en exploitant un cahier de charge plus exigeant.

Bibliographie

[1] http://www.ascenseurs.fr.

[2] http://www.energieplus-lesite.be.

[3] Mezah S. « Cours introduction aux systèmes embarqué temps réel », Cours, Université Abderrahmane Mira, Béjaia, 2018.

[4] YVES MERGY.«prise en main de fritzing concevoir un circuit imprimé», cours, université de applied science potsdam,2007-2010.]

[5] «The most complete starter kit tutorie for Méga 2560»,france,2018

[6] https://zestedesavoir.com/tutoriels/686/arduino-premiers-pas-en-informatique-embarquee/743_gestion-des-entrees-sorties/3423_un-simple-bouton/ vu mai 2019.

[7] https://ardwinner.jimdo.com/arduino/viia-les-capteurs/1a-buzzer-alarme/ vu en avril 2019.

[8] https://openclassrooms.com/fr/courses/3290206-perfectionnez-vous-dans-la-programmation-arduino/3342221-programmez-un-ecran-lcd vu en mai 2019 vu en avril 2019.

[10] https://zestedesavoir.com/tutoriels/686/arduino-premiers-pas-en-informatique-embarquee/747_le-mouvement-grace-aux-moteurs/3437_le-moteur-a-courant-continu/ vu en mars 2019

[11] http://klervithuault.wixsite.com/tpe-ascenseurs/ascenseurs-lectriques vu en mars 2019.

[12] VASEUX Loïc & OUINE Corentin Lycée Le CorbusierDossier.TPE2007-2008

[13] http://schema-montage-electronique.blogspot.com/2013/05/preparation-dun-afficheur-lcd-pour-une.html vu en février 2019.

[14] Memoire fin d'étude master , « commande d'un ascenseur a base PIC›› Université sidi Mohammed ben Abdallah École Supérieure de Technologie, Département Maintenance Industrielle, mémoire de Master (2006).

[15] https://eskimon.fr/tuto-arduino-601-le-moteur-%C3%A0-courant-continu?fbclid=IwAR2SuvkmUGwBU7IaH96oH0Bftp8YDdiPEPj-f6BpJEmuMJICy7palaLCbSQ. Vu en mai 2019.

[16] Memoire fin d'étude master , «commande d'une maquette d'ascenseur » MOHAMED LAMINE DILMI, UNIVERSITE SETIF -ABBAS FERHAT, Faculté de Technologie Département d'Electrotechnique, (2014).

[17] https://www.lemoniteur.fr/article/la-mise-en-securite-des-ascenseurs-existants.346199. Vu en avril 2019.

[18] http://www.ikonet.com/fr/ledictionnairevisuel/arts-et,architecture /architecture/ascenseur/ascenseur. PHP. vu en mars 2019.

[19] Mémoire Master, « commande d'u ascenseur pédagogique à base d'un microcontrôleur ». Université Abderrahmane mira de Bejaia, 2007.

[20] Jimmy Keyrouz. Ascenseur sans local de machinerie. Mécanique des structures [physics.class-ph]. 2011. <Dumas-01081520>.

[21] https://www.energieplus-lesite.be. Vu en avril 2019.

[22] Extrait d'un livre ,The Vertical transportation handbook by Georges R.Strakosch.2016.

[23] Jimmy Keyrouz. Ascenseur sans local de machinerie. Mécanique des structures [physics.class-ph]. 2011.<dumas-01081520>.

[24] École Polytechnique d'Architecture & d'Urbanisme, Exposé sur les escaliers, les rampes et les ascenseurs.

[25] Richard Gedeon. Introduction d'une nouvelle technique à base de Laser dans le contrôle de positionsd'ascenseur. Electronique. 2012. <dumas-01314784>.

[26] Memoire fin d'étude master , « Intelligent Elevator control Based on Adaptive Learning andOptimization »université de Stellenbosch ,2014.

[27] « Sur la description et la programmation de manœuvres d'ascenseur »,Ghestem, Hubert,Siikonen, M-L., 2008, 'Elevator group control method', *European patent no. EP 1 549 581 B1*, 10 September2008.

[28] « GUIDE ASCENSEUR » par la Division Technique, Royaume du Maroc, Ministère de l'équipement et dutransport, Direction des équipements publics.

[29] Cour PPT « Introduction aux IHM », par Jean-Marc PUJOS, © CNAM – UE IHM – NSY110.

[30] Barney, G., 2003. Elevator Traffic Handbook: Theory and practice, Taylor & FrancisGroup.

[31] Extrait du Cahier Technique Schneider Electric n°207.

[32] Francis COTTET & Patrick RENARD « Programmation graphique des applications de contrôle commande».

[33] Académie de bordeaux « fiche de connaissance tice_43" ».

[34] «Controller for peak traffic detection in elevator systems »; University of Seville.

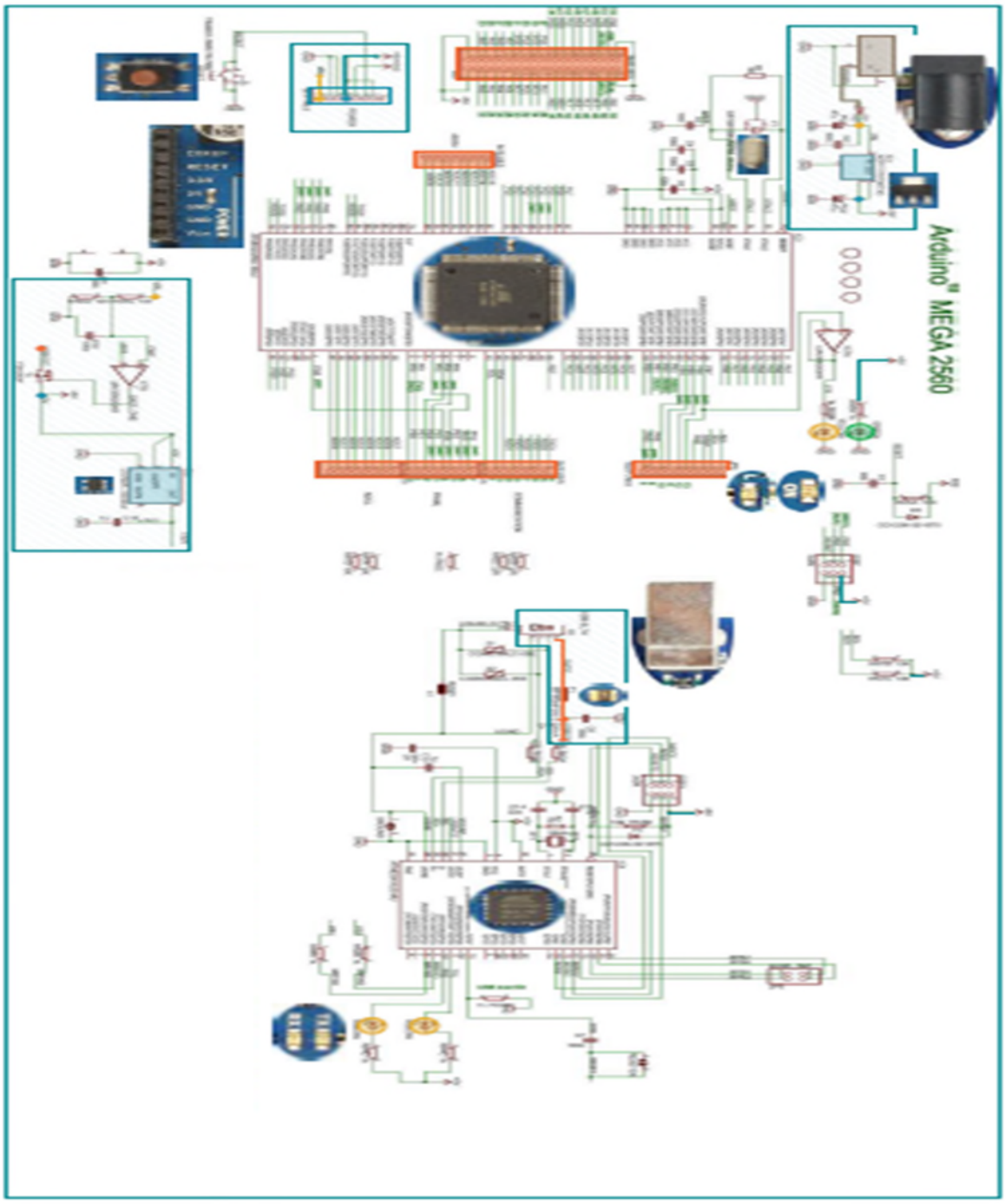
[35] Mémoire Master , « INTELLIGENT ELEVATOR CONTROL BASED ON ADAPTIVE LEARNING.AND OPTIMISATION ». University of Stellenbosch, 2014.

[36] Stanley, J., Williams, D., Simcik, P., Honma, H. & Mori, T., 2011, 'Elevator traffic control including.Destination grouping', *US patent no. 7 921 968 B2*, 12 April 2011. – 12.4.

[37] New developments in elevator,traffic analysis', Elevator Technology 5 proceedings of Elevcon '93, Vienna.

Annexes

Annexe 1 : Architecture d'une carte Arduino Méga 2560



Annexe 2 : Programme principal du fonctionnement de l'ascenseur

```
#include <Keypad.h>
#include <LiquidCrystal.h>
/*************************************     lcd pin     ******************************/
LiquidCrystal lcd(15,14,39, 38,37,36);
/*****************************************************************************************/


/*************************************     moteur pin     ******************************/
int mup=22;
int mdown=23;
/*****************************************************************************************/



/*************************************     porte pin     ******************************/
int porte =10;
/*****************************************************************************************/



/*************************************     Led pin     ******************************/
int ledboutton1=24;
int ledboutton2=25;
int ledboutton3=26;
int ledboutton4=27;

 int ledboutton1up=28;
 int ledboutton2up=29;
 int ledboutton2down=30;
 int ledboutton3up=31;
 int ledboutton3down=32;
 int ledboutton4down=33;


 /*************************************     Led pin     ******************************/

 /*************************************     capteur etage  pin     ******************************/

 int captetage1=50;
 int captetage2=51;
 int captetage3=52;
 int captetage4=53;
 int captetage[]={captetage1,captetage2,captetage3,captetage4};
 /*****************************************************************************************/
```

```
/*************************************     clavier      ****************************/
const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
char hexaKeys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'},
};
byte rowPins[ROWS] = {9, 8, 7, 6};
byte colPins[COLS] = {5, 4, 3, 2};
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

/*****************************************************************************************************/


/*************************************     pin appel_durgence ****************************************/
int LED=46;
int buzzer=47;
int boutton=48;
/*****************************************************************************************************/


/*************************************     pin    arret_durgence  ****************************************/

int button_darret_durgence=44;

/*****************************************************************************************************/

/*************************************     initialisation      ****************************************/
int demandvect[]={0,0,0,0,0,0,0,0,0,0};
int pos; // position de la cabine
int j=0;
int i=0;
int indiceetage = 0;

/*****************************************************************************************************/

void setup(){

  Serial.begin(9600);

  pinMode(mup,OUTPUT);
  pinMode(mdown,OUTPUT);

  pinMode(porte,INPUT);

  pinMode (ledboutton1,OUTPUT);
  pinMode (ledboutton2,OUTPUT);
  pinMode (ledboutton3,OUTPUT);
  pinMode (ledboutton4,OUTPUT);
```

```
pinMode (ledboutton1up,OUTPUT);
pinMode (ledboutton2up,OUTPUT);
pinMode (ledboutton2down,OUTPUT);
pinMode (ledboutton3up,OUTPUT);
pinMode (ledboutton3down,OUTPUT);
pinMode  (ledboutton4down,OUTPUT);

pinMode(LED,OUTPUT);
pinMode(buzzer,OUTPUT);
pinMode(boutton,INPUT_PULLUP);
pinMode(button_darret_durgence,INPUT);


lcd.begin(16,2);

/*************************************envoyer a l'etage 1       ********************************************/

   if    (demandvect[0]==LOW && demandvect[1]==LOW && demandvect[2]==LOW
          && demandvect[3]==LOW && demandvect[4]==LOW && demandvect[5]==LOW
          && demandvect[6]==LOW && demandvect[7]==LOW && demandvect[8]==LOW && demandvect[9]==LOW)
 {while(pos!=1 && digitalRead(porte)==1 )
    {   descendre();
        pos=lirepos();
        etagecabine(pos);
    }
    arreter();
 }
```

```
/*****************************************************************************************************************/

}
/********************** Récupérer la position de la cabine ********************************/
int lirepos()
{
  for(int i =0; i<4;++i)
  {
   if(digitalRead(captetage[i])==HIGH)   pos = i+1;
  }
     Serial.print("La cabine est à l'étage :  ");
     Serial.println(pos);
     etagecabine(pos);
     return pos;

}
/*****************************************************************************************************/

/************************************ affichage LCD ***************************************/
void etagecabine(int fNum)
  {
  lcd.setCursor(0,0);
    lcd.print("etage:" );
    lcd.setCursor(8,0);
   lcd.print(fNum);

  }
/*****************************************************************************************************/




/************************** arreter la cabine *********************************/
void arreter ()
   {
    Serial.println("arreter la cabine");
    digitalWrite(mup,LOW);
    digitalWrite(mdown,LOW);
     lcd.setCursor(0,1);
    lcd.print("     arret      ");

    }
/*****************************************************************************************************/
/************************************************** monter la cabine ************************/
void monter ()
   {

    Serial.println("Monter la cabine");
    digitalWrite(mup,HIGH);
    digitalWrite(mdown,LOW);
     lcd.setCursor(0,1);
    lcd.print("     Monter      ");
    }
/*****************************************************************************************************/
```

```
/************************* Descendre la cabine ***********************************/
void descendre ()
  {
    Serial.println("Descendre la cabine ");
    digitalWrite(mdown,HIGH);
    digitalWrite(mup,LOW);
    lcd.setCursor(0,1);
   lcd.print("   descendre   ");

  }
 /*********************************************************************************************/



/************************* fermiture des portes  ***************************************************/
void fermeture_porte()
   {
   lcd.setCursor(0,1);
   lcd.print("porte fermée");

   }


/*****************************************************************************************************/
```

```
/************************* ouverture des portes  ************************************************************/
void ouverture_porte()
   {
   lcd.setCursor(0,1);
   lcd.print("porte ouverte");

   }

/*************************************************************************************************************/


/************************************* appel d'urgence ******************************************************/
void appel_durgence()
{
 if(digitalRead(boutton)==LOW)
 {
 digitalWrite(LED,HIGH);
 digitalWrite(buzzer,LOW);
 delay(1000);
 digitalWrite(LED,LOW);
 digitalWrite(buzzer,HIGH);
 delay(4000);
 digitalWrite(buzzer,LOW);
 }
}
  /************************************************************************************************************/

void loop(){


/************************* charger le vecteur d'appels ******************************************************/

  char demand = customKeypad.getKey();
      switch (demand)
  {
  case '1' : // appel de monté de l'etage 1
  {   demandvect[0]=1;
      Serial.println(demandvect[0]);
      digitalWrite(ledboutton1up,HIGH);
      break;
  }
    case '2' ://appel de monter de l'etage2
  {   demandvect[1]=1;
      Serial.println(demandvect[1]);
      digitalWrite(ledboutton2up,HIGH);
      break;
  }
   case '4' : //appel de monté  de l'etage 3
  {   demandvect[3]=1;
      Serial.println(demandvect[3]);
      digitalWrite(ledboutton3up,HIGH);
      break;
  }
```

```
 case '3' : //appel de descendre de l'etage 2
 {   demandvect[2]=1;
     Serial.println(demandvect[2]);
     digitalWrite(ledboutton2down,HIGH);
     break;
 }
 case '5' : //appel de descendre de l'etage 3
 {   demandvect[4]=1;
     Serial.println(demandvect[4]);
     digitalWrite(ledboutton3down,HIGH);
     break;
 }
 case '6' : //appel de descendre de l'etage 4
 {   demandvect[5]=1;
     Serial.println(demandvect[5]);
     digitalWrite(ledboutton4down,HIGH);
     break;
 }
 case'A': // aller vers l'etage 1
 {
   demandvect[6]=1;
   Serial.println(demandvect[6]);
   digitalWrite(ledboutton1,HIGH);
    break;
 }

case'B':// aller vers l'etage 2
{
    demandvect[7]=1;
    Serial.println(demandvect[7]);
    digitalWrite(ledboutton2,HIGH);
     break;
}
case'C'://aller vers l'etage 3
{
  demandvect[8]=1;
  Serial.println(demandvect[8]);
  digitalWrite(ledboutton3,HIGH);

   break;
}
case'D'://aller vers l'etage 4
{
  demandvect[9]=1;
  Serial.println(demandvect[9]);
  digitalWrite(ledboutton4,HIGH);

       break;

 }
}
```

```
// Afficher le vecteur d'appels et demandes
for (i=0;i<10;i++) Serial.print(demandvect[i]);// affichage de vecteur d'appels
Serial.println("");
/*************************************************************************************************/

  pos=lirepos();// récuperer la position de la cabine

/*******************************au repos ***************************************************/

if (digitalRead(mup)==0 && digitalRead(mdown)== 0 && digitalRead(porte)==1 ) //le moteur et au repos && porte fermée
{   if (demandvect[0]==HIGH || demandvect[6]==HIGH  )    descendre() ; //appel ou demande de l'etage 1
    if (demandvect[1]==HIGH || demandvect[2]==HIGH || demandvect[7]==HIGH ) //appel ou demande de l'etage2
        {  if (pos>2)descendre() ;  else if (pos==1 ) monter(); }
    if (demandvect[3]==HIGH || demandvect[4]==HIGH || demandvect[8]==HIGH )//appel ou demande de letage3
        { if (pos==4)  descendre() ; else if (pos<3) monter();}
    if (demandvect[5]==HIGH || demandvect[9]==HIGH )  monter();//appel ou demande de letage4

}
/*************************************************************************************************/

 /*************************************entrain de monter ***********************************************/
  if (digitalRead(mup)==1 && digitalRead(mdown)==0  )
 {
   if (pos==2)
   {
     // demande ou appel de monter  a l'etage 2 uniquement
      if ((demandvect[1]==HIGH ||demandvect[7]==HIGH)  && (demandvect[3]==LOW
      && demandvect[4]==LOW &&demandvect[8]==LOW && demandvect[5]==LOW && demandvect[9]==LOW))
      {  arreter();
         ouverture_porte();

         //mise a 0 des appels et des LEDS
         demandvect[1]=LOW; digitalWrite(ledboutton2up,LOW );
         demandvect[7]=LOW; digitalWrite(ledboutton2,LOW );

         //arreter la cabine pendant 5s
          delay(5000);fermeture_porte() ;
      }
       //prise en charge:de l'appel de monté de l'etage 2 et tout les appels de monté qui sont  au dessus
      if (demandvect[1]==HIGH|| demandvect[7]==HIGH)
     {  arreter();
        ouverture_porte();
        //mise a 0 des appels et des LEDS
         demandvect[1]=LOW;digitalWrite(ledboutton2up,LOW );
         demandvect[7]=LOW;digitalWrite(ledboutton2,LOW );
```

```
    //arreter la cabine pendant 5s
    delay(5000);

    if(digitalRead(porte)==1)
    {fermeture_porte() ;
      monter();
    }
}
// demande ou appel de dscendre  a l'etage 2 uniqument
  if ((demandvect[2]==HIGH ||demandvect[7]==HIGH) && (demandvect[3]==LOW
  && demandvect[4]==LOW && demandvect[8]==LOW && demandvect[5]==LOW && demandvect[9]==LOW   ))
{   arreter();
    ouverture_porte();
    //mise a 0 des appels et leds
    demandvect[2]=LOW; digitalWrite(ledboutton2down,LOW );
    demandvect[7]=LOW; digitalWrite(ledboutton2,LOW );
    //arret la cabine pendant 10s
    delay(5000);
    fermeture_porte() ;
}

}


 else if (pos==3 )
{
  // demande ou appel de monté  a l'etage 3 uniqument
  if ( (demandvect[3]==HIGH   || demandvect[8]==HIGH) && (demandvect[5]==LOW && demandvect[9]==LOW))
{   arreter();
//mise a 0 des appels et des leds
    ouverture_porte();
    demandvect[3]=LOW;digitalWrite(ledboutton3up,LOW);
     demandvect[8]=LOW;digitalWrite(ledboutton3,LOW);

    // arreter le moteur pendant 5s
    delay(5000);
    fermeture_porte() ;
}

    //prise en charge:de l'appel de monté de l'etage 3 et tout les appels de monté qui sont  au dessus
    if (  (demandvect[3]==HIGH || demandvect[8]==HIGH ))
{   arreter();
    ouverture_porte();

    //mise a 0 des appels et LEDS
    demandvect[3]=LOW; digitalWrite(ledboutton3up,LOW);
     demandvect[8]=LOW; digitalWrite(ledboutton3,LOW);
     // arreter le moteur pendant 5s
    delay(5000);
```

```
    if(digitalRead(porte)==1)
    {
      fermeture_porte() ;
      monter();
    }

}
  //demande ou appel de dscendre  a l'etage 3 uniqument
    if ( (demandvect[4]==HIGH || demandvect[8]== HIGH) && (demandvect[5]==LOW && demandvect[9]==LOW))
{   arreter();
    ouverture_porte();

    //mise a 0 des appels et leds
    demandvect[4]=LOW; digitalWrite(ledboutton3down,LOW);
     demandvect[8]=LOW;digitalWrite(ledboutton3,LOW);
    //arreter pendant 5s
    delay(5000);
    fermeture_porte() ;

}
}



    //demande ou appel de descendre de l'etage 4
    else  if (pos==4  && (demandvect[5]==HIGH || demandvect[9]==HIGH))
    {
        arreter();
        ouverture_porte();
        //mise a 0 des appels et leds
        demandvect[5]=LOW;digitalWrite(ledboutton4down,LOW);
        demandvect[9]=LOW;digitalWrite(ledboutton4,LOW);

        //arreter l moteur pendant 5s
        delay(5000);
        fermeture_porte() ;

    }
  //prise en charge de tout les appels et arreter le moteur
 else  if   (demandvect[0]==LOW && demandvect[1]==LOW && demandvect[2]==LOW
    && demandvect[3]==LOW && demandvect[4]==LOW && demandvect[5]==LOW && demandvect[6]==LOW
     && demandvect[7]==LOW && demandvect[8]==LOW && demandvect[9]==LOW)
arreter ();

}
/*****************************************************************************************************/
```

```
/************************** entrain de descendre ************************************************/
if (digitalRead(mup)==0 && digitalRead(mdown)==1 )
{
  if (pos==2  )

    {
      //demande ou appel de descendre   a l'etage 2 uniqument
     if ((demandvect[2]==HIGH ||  demandvect[7]==HIGH) && (demandvect[0]==LOW && demandvect[6]==LOW))
    {
       arreter();
       ouverture_porte();
       //mise a 0 des appels et leds
       demandvect[2]=LOW;digitalWrite(ledboutton2down,LOW);
       demandvect[7]=LOW;digitalWrite(ledboutton2,LOW);

       //arreter le moteur pendant 5s
       delay(5000);

       fermeture_porte() ;

    }
    //prise en charge:de l'appel de descendre  de l'etage 2 et tout les appels de descendre qui sont  au dessous
       if (demandvect[2]==HIGH || demandvect[7]==HIGH)
    {
```

```
      arreter();
      ouverture_porte();
      demandvect[2]=LOW; digitalWrite(ledboutton2down,LOW);
      demandvect[7]=LOW;digitalWrite(ledboutton2,LOW);
      delay(5000);
      if(digitalRead(porte)==1)
      {
      fermeture_porte() ;
      descendre();
      }
 }
 //demande ou appel de monter a l'etage 2 uniqument
  if ((demandvect[1]==HIGH ||  demandvect[7]==HIGH) && (demandvect[0]==LOW && demandvect[6]==LOW))
  {
      arreter();
      ouverture_porte();
      //mise a 0 des appels et leds
      demandvect[1]=LOW;digitalWrite(ledboutton2up,LOW);
      demandvect[7]=LOW;digitalWrite(ledboutton2,LOW);

      //arreter le moteur pendant 5s
      delay(5000);

      fermeture_porte() ;
  }
  }
else  if (pos==3 )

{
 //demande ou appel de descendre   a l'etage 3 uniqument
if (( demandvect[4]==HIGH || demandvect[8]==HIGH)&& ( demandvect[0]==LOW
  && demandvect[6]==LOW && demandvect[1]==LOW && demandvect[2]==LOW && demandvect[7]==LOW ))
{

  arreter();
  ouverture_porte() ;

  //mise a 0 des appels et leds
  demandvect[4]=LOW;digitalWrite(ledboutton3down,LOW);
   demandvect[8]=LOW;digitalWrite(ledboutton3,LOW);

  //arreter le moteur pendant 5s
  delay(5000);

  fermeture_porte() ;

}
 //prise en charge:de l'appel de descendre  de l'etage 3 et tout les appels de descendre qui sont  au dessous
    if (demandvect[4]==HIGH || demandvect[8]==HIGH)
{
  arreter();
  ouverture_porte() ;
```

```
  //mise a 0 des appels et leds
  demandvect[4]=LOW; digitalWrite(ledboutton3down,LOW);
  demandvect[8]=LOW;digitalWrite(ledboutton3,LOW);

  //arreter le moteur pendant 5s
 delay(5000);

 if(digitalRead(porte)==1)
 {fermeture_porte() ;
  descendre();
 }
 }
 //demande ou appel de de monté l'etage 3 uniqument
  if (( demandvect[3]==HIGH || demandvect[8]==HIGH) && (demandvect[1]==LOW
     &&  demandvect[2]==LOW && demandvect[7]==LOW && demandvect[0]==LOW && demandvect[6]==LOW))
{
  arreter();
  ouverture_porte() ;

  //mise a 0 des appels et leds
  demandvect[3]=LOW;digitalWrite(ledboutton3up,LOW);
  demandvect[8]==LOW;digitalWrite(ledboutton3,LOW);

  //arreter le moteur pendant 5s
  delay(5000);
  fermeture_porte() ;
}
```

```
 }

  //demande et appel de monter de l'etage1
  else  if (pos==1  && (demandvect[0]==HIGH || demandvect[6]==HIGH))
  {
     arreter();
     ouverture_porte() ;

     //mise a 0 des appels et leds
     demandvect[0]=LOW; digitalWrite(ledboutton1up,LOW);
     demandvect[6]=LOW; digitalWrite(ledboutton1,LOW);

    // arreter le moteur pendant 5s
     delay(5000);

     fermeture_porte() ;


    }
    //prise en charge de tout les appels et arreter le moteur
 else if  (demandvect[0]==LOW && demandvect[1]==LOW && demandvect[2]==LOW
     && demandvect[3]==LOW && demandvect[4]==LOW && demandvect[5]==LOW && demandvect[6]==LOW
     && demandvect[7]==LOW && demandvect[8]==LOW && demandvect[9]==LOW)
arreter ();

}



/*****************************************************************************************************

//déclenchement d'alarme en cas d'urgence
appel_durgence();

 //arret d'urgence
   if (digitalRead(button_darret_durgence)==LOW)
    digitalWrite(mup,LOW);
    digitalWrite(mdown,LOW);

}
```

Résumé

Les ascenseurs sont des systèmes complexes. Les objectifs des recherches en cours est d'améliorer le confort et la sécurité, car aujourd'hui l'ascenseur est un élément essentiel dansla conception des immeubles à moyenne et grande hauteur.

Dans ce mémoire nous avons proposé une commande d'un ascenseur par Arduino. Une maquette a été réalisée aussi,elle servira pour la poursuite de ce projet. On a découvert à travers nos recherchesbibliographiques une grande activité de recherche sur les ascenseurs et notamment lesbatteries d'ascenseurs. Parmi ces travaux, le plus souvent la commande des ascenseurs estréalisée à base d'outils dédiés au milieu industriel tel que les microcontrôleurs industriel, API…etc. Ce domaine de développement exige le concours de différents spécialistes (automaticien, informaticien, mécanicien, etc.)

ملخص

المصاعد أنظمة معقدة وإن أهداف .الأبحاث في هذا المجال تنطوي تحت راية تحسين الراحة وأمان المستخدم , لأنالمصاعد اليوم تعد عنصر أساسي في تصميم المباني ذات الإرتفاعات المتوسطة ,والعالية . في هذه العمل سنحاولإقتراح منظومة تحكم للمصاعد عن طريق الأردوينو,اين قمنا بإنشاء نموذج مصغر أيضا لدعم السير الحسن لمشروعنا. لقد إكتشفنا خلال أبحاثنا المكتبتية الكثير من الأعمال خاصة ' المصاعد كثيرة المقصورات', ومن بين الأعمال الكثيرة في هذا المجال يلجأ فيها إلى مختلف API, µC ,منها يهتم بالتحكم بالمصاعد ,أين أستعملت مختلف الحاكمات الصناعية مثلالتخصصات الهندسية (تحكم ,آلية,إعلام آلي ,ميكانيك .....إلخ) .هذه الأنظمة في تطور مستمر.

هذه العمل هي تقدمةفي إطار الحصول على شهادة درجة الماستر, بجامعة باجي مختار عنابة , تخصص آلية و أنظمة. في هذه الوثيقة سنحاولإقتراح منظومة تحكم بمصعد كهربائي بإستعمالالأردوينو. العديد من الدراسات قد وضعة في إطار التحكم الصناعيايناالكثير منها قد إعتمدت تكنولوجيات موجهة للميدان مثل الحاكمات الآلية , الميكروكونترولر ...إلخ .سنحاولاقتراحمنظومة تحكم عن طريق الأردوينو , أين سنستكشف مختلف منظومات العمل الموجهة للتحكم بالمصاعد الكهربائيةثم القيام بمحاكاتها على نموذج مصغر لمصعد من أربعة طوابق من إقتراحنا.

Abstract

Elevators systems are complex. Research aim in this area to improving the comfort andsafety of the user, when elevators today is an essential element in the design of buildings withmedium and high elevations. In this work, we will try to propose a control system forelevators through the Arduino, where we have also created a miniature model to support thegood walk of our project. In the course of our bibliographic research, we have confronted anumber of works in this domain, especially 'groups elevators', many of which are concernedwith elevator control, where the various industrial controllers such as the API (µC) have usedvarious disciplines such as control, automation, Mechanics ... etc). These systems are inconstant evolution. This thesis is an introduction in the framework of obtaining the masterdegree at the University of Baji Mukhtar Annaba, specializing in machinery andsystems. In this document, we will try to propose a control system for an electric elevatorusing the Arduino. Many of the studies have been put into the framework of industrial control,many of which have adopted field-oriented technologies such as automatic controls,microcontroller, etc. We will try to propose a control system through the Arduino, where wewill explore various work systems designed to control the elevators, for a four-storey elevatorof our proposal.