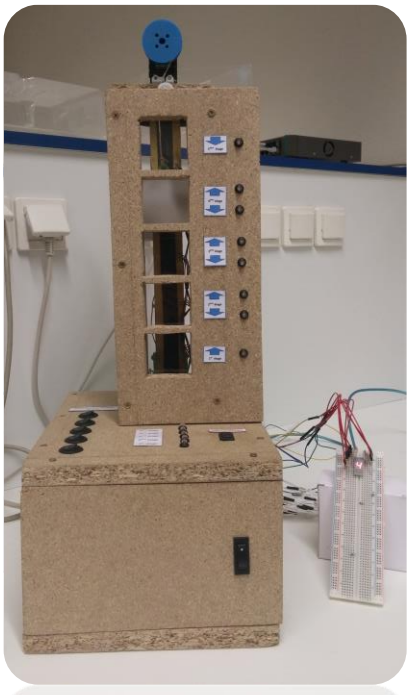
Maquette mécatronique durcie d'ascenseur 5 étages

Rapport de projet de fin d'études



Louis CHAUCHARD
Romain IMBERT
Informatique, Microélectronique et Automatique
2015-2016

M. Blaise CONRARD
M. Alexandre BOÉ
M. Xavier REDON
M. Thomas VANTROYS

Sommaire

Table des figures

Introduction

Dans le cadre de notre projet de fin d'études de cinquième année en Informatique, Micro- électronique, Automatique, nous avons choisi de nous pencher sur le sujet de « La maquette mécatronique durcie d'ascenseur 5 étages ». Ce sujet initialement proposé par M. Conrard consistait à faire l'étude, mais aussi l'élaboration de deux maquettes d'ascenseur 5 étages utilisables lors de travaux pratiques.

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.

Afin d'exposer au mieux l'élaboration de ce projet, nous ferons, dans un premier temps, un rappel du cahier des charges. Puis nous détaillerons ensuite, les différentes réalisations nécessaires. Enfin, nous terminerons par une analyse des difficultés que nous avons rencontrées ainsi que les perspectives et les améliorations possibles de ce projet.

Présentation du projet

1. Contexte

Actuellement, une seule maquette d'ascenseur est proposée pendant les séances de travaux pratiques d'automatisme. D'autres maquettes identiques permettront, tout d'abord d'augmenter le nombre de binômes travaillant sur le système qu'est l'ascenseur en automatisme, mais elles offriront aussi un exemple supplémentaire lors de portes ouvertes ou de présentation de la filière Informatique, Microélectronique et Automatique et plus précisément des cours enseignés en automatisme.

2. Objectif du projet

L'objectif initial de ce projet était de réaliser deux maquettes d'ascenseurs cinq étages supplémentaires destinées à être utilisées en Travaux Pratiques et elles devaient être commandables par automate programmable.

Dans le but de réaliser un système fiable et robuste, un soin devait être apporté à la sûreté de fonctionnement.

En effet, les différents composants utilisés devaient, soit posséder une fiabilité élevée, soit être simples et rapide à changer.

3. Cahier des charges

Toujours dans l'objectif d'obtenir un système fiable et robuste, un mode permettant la détection de panne ou de défaut devait être réalisé. Ce mode déclenché par l'appui d'un bouton-poussoir "test" permet de faire fonctionner les différents voyants afin de déceler rapidement un quelconque dysfonctionnement.

Afin de pouvoir être utilisées lors des travaux pratiques d'automatisme, ces maquettes doivent pouvoir être connectées à un automate fonctionnant avec des tensions industrielles (0-24V).

Dans le but de garantir la protection de l'utilisateur, la partie mobile de la maquette (c'est-à-dire la cabine) ne doit pas être accessible.

La maquette possédera deux modes de fonctionnement en plus d'un mode "défaut". Ces deux modes de fonctionnement seront soit l'ascenseur relié à l'automate pour réaliser un TP d'automatismes, soit un mode "démonstration" où l'ascenseur ne sera géré que par le biais de l'Arduino et ne nécessitera donc pas de programmation préalable de l'automate pour fonctionner.

Fonctionnement de l'ascenseur relié à l'automate (pour les Travaux Pratiques) :

Ce mode de fonctionnement permettra de réaliser un scénario particulier en créant un programme qui sera transféré ensuite dans l'automate qui lui-même actionnera les différents composants de la maquette.

Fonctionnement de l'ascenseur en mode "démonstration" qui sera utilisé pour les journées portes ouvertes de Polytech' Lille :

Ce mode de fonctionnement permettra d'actionner les différents composants de la maquette directement par le biais de l'Arduino dans le but de démontrer le fonctionnement de celle-ci.

Un scénario possible peut être le suivant:

L'utilisateur appelle l'ascenseur à un l'étage x. Il choisit entre le bouton poussoir bas ou haut. Si l'utilisateur est à l'étage 1, il n'aura que le choix d'appuyer sur le bouton poussoir d'appel haut. Si l'utilisateur est à l'étage 5, il n'aura que le choix d'appuyer sur le bouton poussoir d'appel bas. C'est deux obligations permettent d'éviter un dysfonctionnement de la maquette. Dès que l'ascenseur est arrivé à cet étage x, l'utilisateur entre dans la cabine et il appuiera sur le bouton poussoir concernant l'étage désiré. L'ascenseur monte ensuite à l'étage désiré. L'utilisateur sort alors de la cabine. Après une durée t, cet utilisateur appelle à nouveau la cabine avec le bouton poussoir appel palier.

Fonctionnement de l'ascenseur en mode "détection de défauts" :

À tout moment de l'utilisation de la maquette, il doit être possible via un bouton "test" de vérifier l'état de tous les voyants lumineux de la maquette afin de vérifier leur bon fonctionnement.

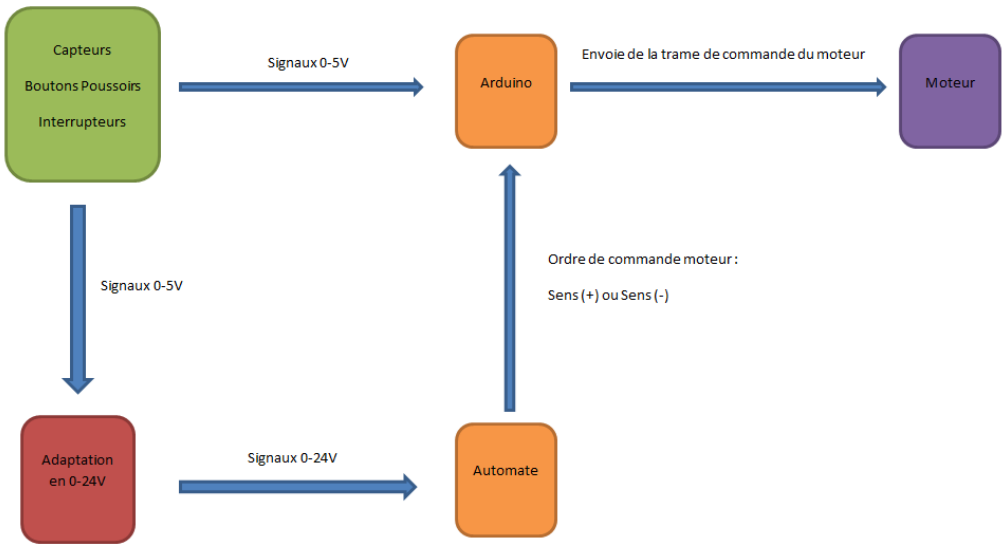


Figure 1 : Fonctionnement détaillé de la maquette

Réalisations techniques

Suite aux étapes d'étude et de commande des composants détaillées lors de notre rapport intermédiaire, la conception de la maquette finale nous a permis de réaliser différentes tâches dans divers domaines telles que la conception du bâti, la conception de la partie mobile ou encore la réalisation et le câblage de différentes cartes électroniques.

1. Réalisation du bâti

Dans le but d'assurer une solidité de la maquette au vu de son utilisation en travaux pratiques, nous avons fait le choix d'utiliser de l'aggloméré d'une épaisseur de 20mm découpé en plusieurs panneaux. Ces panneaux permettant ensuite de concevoir assez rapidement le bâti imaginé et modélisé précédemment via le logiciel SolidWorks pendant la phase d'étude.

Cette modélisation nous a permis d'avoir un aperçu général et de définir les dimensions de la maquette finale.

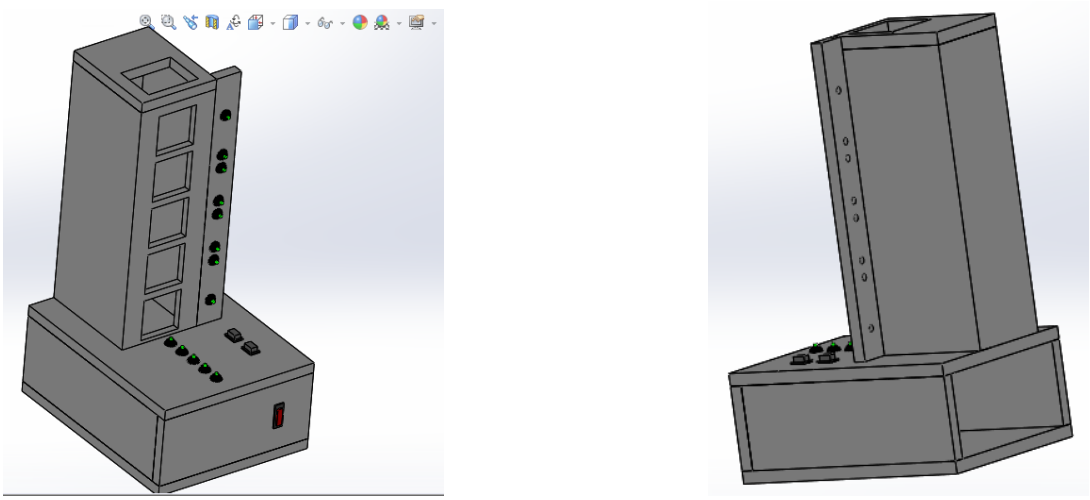


Figure 2 : Modélisation de la maquette sous SolidWorks (face avant et arrière)

La découpe de 5 carrés sur la face avant permet de modéliser les 5 étages qui composent la maquette.

À la droite de ces différents étages se trouvent les boutons d'appel extérieurs. Pour chaque étage (sauf pour le premier et dernier étage), deux boutons permettant de monter, mais aussi de descendre sont présents afin de rendre possible la gestion de manœuvre collective.

Un boîtier, situé sous l'ascenseur servant de base pour les boutons de la cabine, mais aussi les interrupteurs simulant l'ouverture ou la fermeture des portes de chaque étage,

permet d'accueillir les nombreuses cartes électroniques de la maquette et leurs câblages. (Carte des boutons cabine, des boutons étages, des capteurs ainsi que la carte d'alimentation de la maquette).

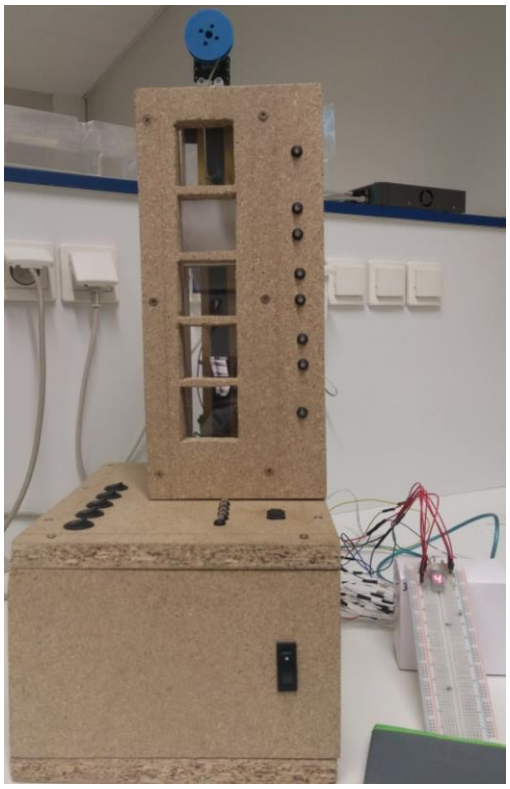


Figure 3 : Illustration de la maquette complète

2. Partie mécanique

D'autre part, nous avons réalisé deux plaques latérales qui sont vissées à la face avant et qui permettent de fixer les tasseaux pour la bonne translation de la cabine.

La cabine est modélisée à l'aide d'un morceau de mélaminé. Son revêtement plastifié permet de minimiser les frottements.

L'utilisation du logiciel SolidWorks et de l'imprimante 3D du Fabricarium nous a permis de réaliser différentes pièces utiles à la translation de la cabine.

Tout d'abord, le support de fixation de la cabine. Celui-ci permet de lier la cabine au fil qui assure le mouvement.

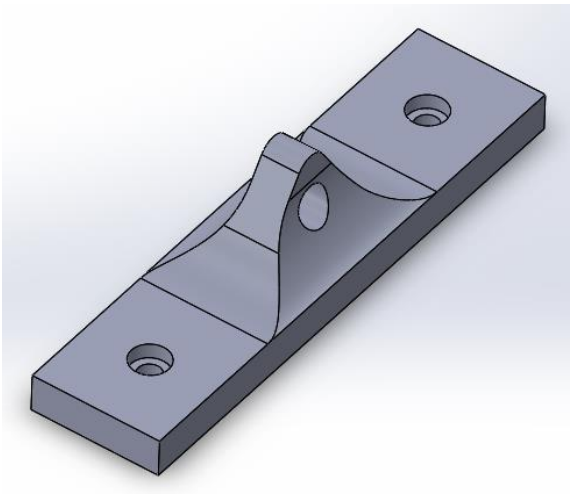


Figure 4 : Modélisation et impression du support cabine

La poulie qui est fixée au moteur a été également réalisée à l'aide de l'imprimante 3D. Ce qui permet de l'adapter parfaitement au moteur AX12+ utilisé.

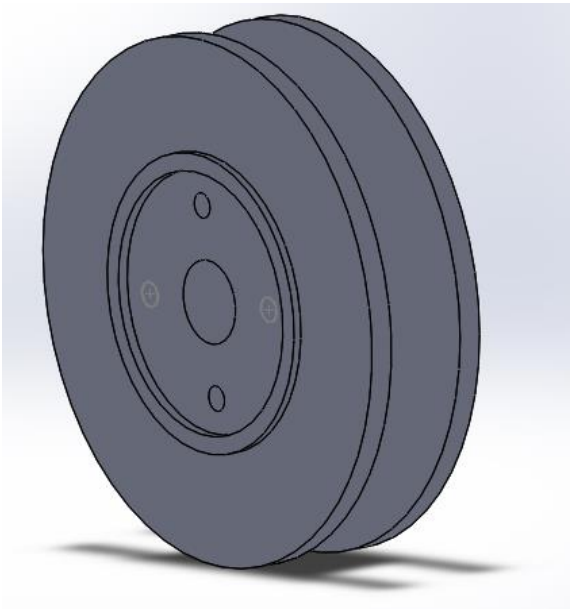


Figure 5 : Modélisation et impression de la poulie adaptée au moteur AX12+

Enfin nous avons aussi imprimé des équerres de fixation.

3. Partie électronique

Concernant la partie électronique, plusieurs réalisations ont été nécessaires.

En effet, la principale réalisation était bien évidemment la gestion des deux modes de la maquette, c'est à dire la gestion du mode démonstration par le biais de l'Arduino et le mode Automate. Cependant, nous avons également réalisé un circuit de protection afin d'éviter tout dysfonctionnement pouvant détériorer fortement la maquette ainsi qu'une carte d'alimentation.

Gestion partie Arduino et partie Automate

Dans cette partie, nous allons développer tout d'abord l'étude de la gestion partie automate et partie Arduino.

Comme nous l'avons remarqué ultérieurement, une "difficulté" du projet est la gestion de deux tensions de fonctionnement. En effet, la partie automate doit fonctionner en 24V tandis que la partie gérée par l'Arduino a une tension de fonctionnement de 5V.

Il faut donc trouver un moyen de manipuler ces deux tensions en évitant surtout que l'Arduino reçoive des tensions supérieures à 5V.

Une première solution envisagée était l'utilisation d'un relais Reed, qui permettait d'éviter des liens entre 24V et 5V. Cependant, étant donné le nombre d'entrées-sorties nécessaires (boutons, voyants et capteurs) et le prix (un peu plus de deux euros pour un seul composant), cette solution était bien trop coûteuse et c'est pourquoi nous ne l'avons pas retenue.

Une seconde solution possible et envisageable en termes de coût est l'utilisation d'optocoupleurs.

Nous avons utilisé cette seconde solution qui est la plus judicieuse du fait du nombre de composants que nous devons utiliser pour la gestion des deux tensions.

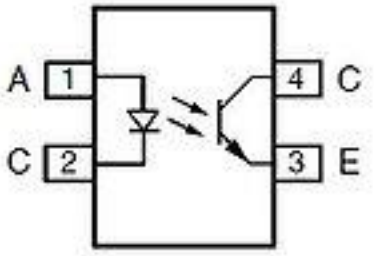


Figure 6 : Schéma d'un optocoupleur

Ces composants permettent l'échange d'un signal électrique tout en réalisant une isolation galvanique, c'est-à-dire qu'il n'existe aucune liaison par conducteur électrique.

Les seuls échanges se font par “liaison” optique. Ce qui permet donc de limiter les tensions sur l'Arduino.

Nous avons alors réalisé les divers montages nécessaires avec à chaque fois l’utilisation de ce dispositif afin de limiter les tensions sur l’Arduino pour chaque entrée-sortie composant la maquette.

Pour la détection des étages lors de la montée ou de la descente de la cabine d'ascenseur, nous utilisons des capteurs inductifs à effets Hall dans le but d’éviter les contacts et donc une usure des capteurs. Ce qui entraîne une réduction de la maintenance de ceux-ci.

Dès la réception de ces capteurs numériques à effet Hall, nous avons réalisé des tests afin de valider leur fonctionnement. Lorsque l’on approche un aimant près du capteur, une variation de tension est effectuée. Une tension de 0V est affichée lorsque l’on approche un aimant et une tension d’environ 5V (tension d’alimentation) si l’aimant est non présent devant le capteur. Ce capteur est alimenté en 5V, la sortie (Output sur le schéma) de celui-ci doit donc être câblée à une résistance dite de “pull up”. Cette résistance est nécessaire pour fixer les deux niveaux de tensions disponibles selon les deux scénarios. Nous avons gardé cette logique inversée pour les entrées Arduino. Par contre, pour les entrées Automate nous avons réalisé une inversion logique en sortie de l’optocoupleur. Cette modification permet d’avoir un niveau logique haut (tension de 5V) lorsque l’aimant est devant le capteur et un niveau logique bas (tension de 0V) si l’aimant n’est pas présent.

Nous avons choisi de faire cette inversion seulement pour la partie automate pour ainsi rentre plus intuitif l’utilisation de la maquette lors des Travaux Pratiques d’automatisme.

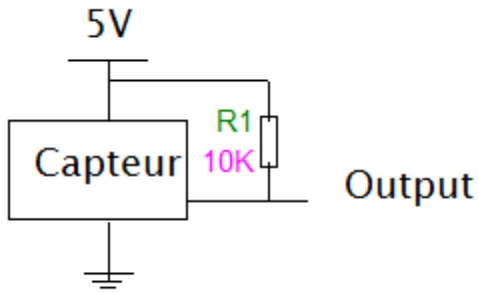


Figure 7 : Schéma du capteur à effet Hall

Le premier montage ci-dessous utilise un optocoupleur pour fournir une information à l’automate en 0-24V. La tension envoyée à l’Arduino est directement la tension fournie par le capteur à effet Hall.

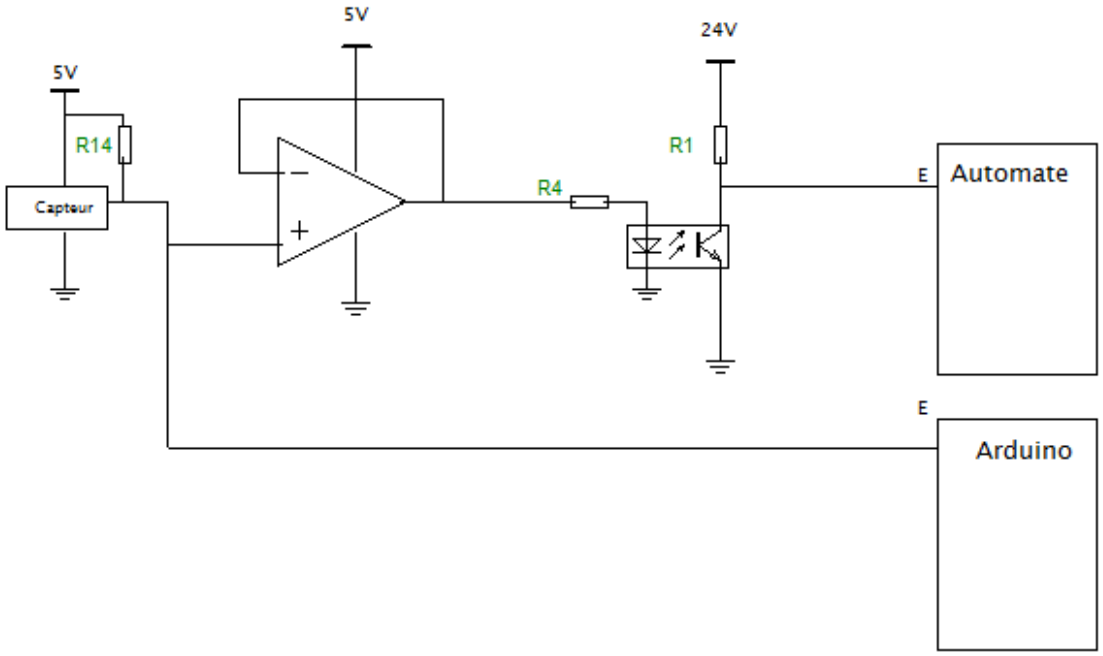


Figure 8 : Adaptation des niveaux de tension pour le capteur à effet Hall

Loi des mailles 1:

Vd= tension aux bornes de la diode

Vcapteur: tension délivrée par le capteur à effet Hall

i4: courant traversant la résistance R4

Vcapteur = R4*i4 + Vd

On fixe i4= 20mA, on peut donc calculer la valeur de la résistance R4.

$R4=\frac{Vcapteur - Vd}{i4} = \frac{5-1.2}{0.020}$ = 190Ω (180Ω valeur normalisée)

Lois des mailles 2:

Vcc= 24V

Vce0: tension entre collecteur et émetteur du transistor NPN

i1=courant traversant la résistance R1

Vcc = Vce0+ R1*i1

On fixe le courant i1 à 1mA

$R1= \frac{Vcc - Vce0}{i1}$ = 23.9 k Ω (valeur normalisée : 22kΩ)

Le schéma électrique ci-dessous utilise la même solution technologique que le schéma précédent. Nous utilisons deux optocoupleurs. Le premier sert à envoyer l'information si l'utilisateur a appuyé sur le bouton poussoir et le second permet d'allumer le voyant de ce bouton poussoir grâce à l'information fournie par l'automate. Ces deux composants servent toujours pour l'adaptation de tension entre les deux dispositifs.

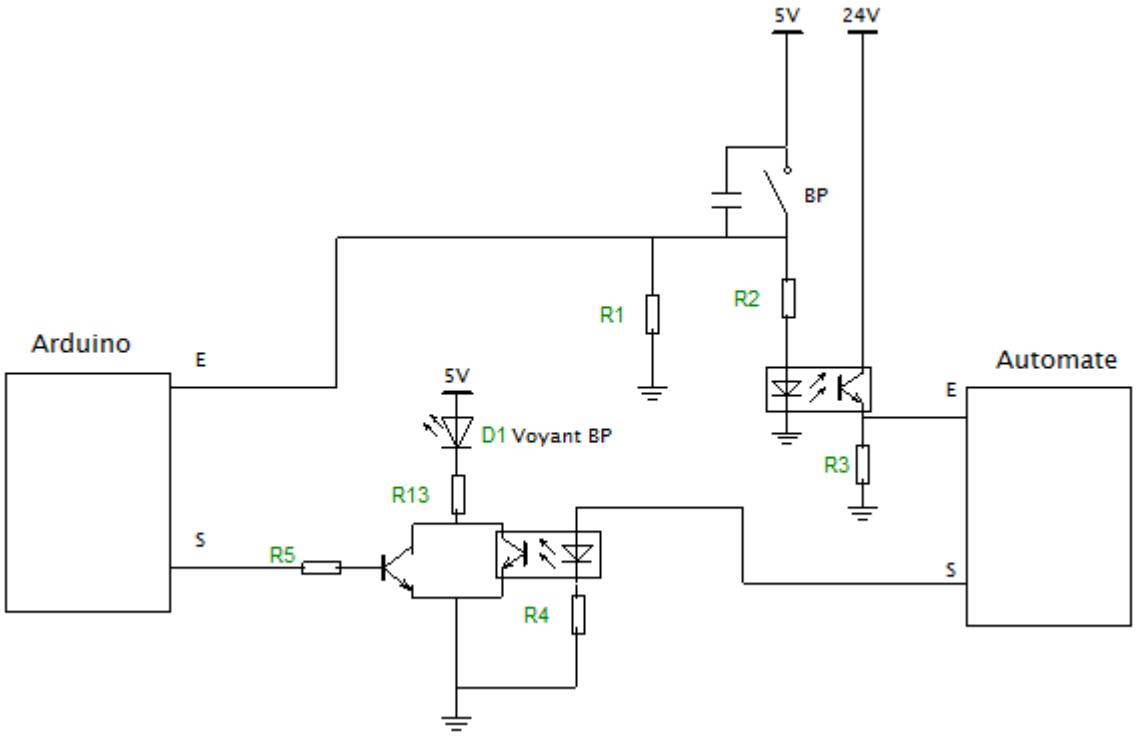


Figure 9 : Schéma pour la commande des boutons poussoirs et leur LED

Dans le but de supprimer des problèmes de rebonds lors de la génération de fronts montants avec les divers boutons poussoirs, il est nécessaire d'ajouter un condensateur à leurs bornes.

- **Calcul des résistances pour la partie bouton-poussoir:**

<u>Lois des mailles 1:</u>

Vcc= 5V

Vd= tension aux bornes de la diode

i2: courant traversant la résistance R2

Vcc = R2*i2 + Vd

On fixe i2= 20mA, on peut donc calculer la valeur de la résistance R2.

$R2 = \frac{Vcc-Vd}{i2} = \frac{5-1.2}{0.020} = 190\Omega$ (valeur normalisée : 180Ω)

L'ajout d'un condensateur de 1uF aux bornes du bouton poussoir a été ajouté au schéma initial, car les tests ont montré qu'il y avait des rebonds.

Loi des mailles 2:

Vcc=24V

Vcc= Vce0+R3*i3

On fixe le courant i3 à 1mA

R3 = $\frac{Vcc - Vce0}{i3}$ = 23.9 kΩ (valeur normalisée: 22kΩ)

- **Calcul des résistances pour la partie voyant lumineux:**

Loi des mailles 1:

Vautomate=0- 24V

Vd= tension aux bornes de la diode

i4: courant traversant la résistance R4

Vautomate = R4*i4 + Vd

On fixe R4= 1.5k22k Ω pour avoir un courant i4 inférieur à légèrement inférieur à 20mA.

i4 =$\frac{Vautomate-Vd}{R4}$ = $\frac{24-1.2}{1500}$ = 15.2mA

Cette valeur de résistance choisie permet de minimiser la puissance dissipée dans cette résistance.

Puissance dans R4 = R4*i4^2= 1500*(0.0152)^2= 0.346 W

Il est donc nécessaire d'utiliser une résistance de 0.5 W.

Loi des mailles 2:

Vcc=5V

Vled: tension aux bornes du voyant lumineux

Vce0:tension aux bornes du transistor NPN et du phototransistor

Vcc= Vled+Vce0+R13*i13

On fixe le courant i13 à 20mA

R13 = $\frac{Vcc-Vled-Vce0}{i13}$ = $\frac{5-2.1-0.1}{0.020}$ = 140Ω (valeur normalisée : 150Ω)

Loi des mailles 3:

Varduino=0-5V

Varduino: tensions fournies par la sortie de l'Arduino

Vled:tension aux bornes du voyant lumineux

Vbe0:tension base émetteur du transistor NPN

Varduino= Vbe+R5*i5

On doit avoir i5 > $\frac{i13}{\beta}$ pour que le transistor soit saturé

R5 = $\frac{\text{Varduino} - \text{Vbe}}{i5}$ = $\frac{5-0.70}{0.020}$ = 215Ω (valeur normalisée : 220Ω)

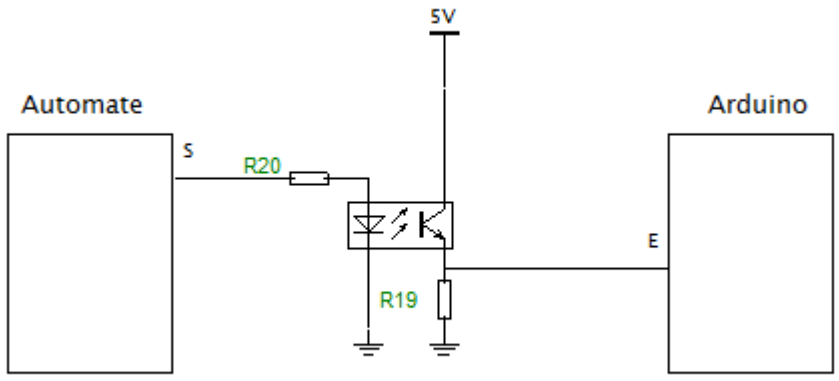


Figure 10 : Schéma pour la commande du moteur

Ce schéma permet d'adapter le niveau de tension de l'information de commande moteur vers les entrées Arduino. Ce circuit est présent deux fois afin de gérer les deux sens de rotation du moteur.

- **Calcul des résistances pour la partie commande du moteur:**

Lois des mailles 1:

Vautomate=0-24V

Vd= tension aux bornes de la diode

i20: courant traversant la résistance R20

Vcc = R20*i20 + Vd

On fixe R20= 1.5 k Ω afin d'avoir un courant inférieur à 20mA pour limiter la puissance dissipée dans la résistance.

$i20 = \frac{Vautomate - Vd}{R20} = \frac{24-1.2}{1500} = 15.2$ mA

Puissance dissipée dans la résistance: PR10= R10 i20² = 1500*0.0152² = 0.346 W*

Nous avons donc choisi une résistance de 0.5W.

Loi des mailles 2:

Vcc=5V

Vcc= Vce0+R19*i19

On fixe le courant i19 à 1mA

$R19 = \frac{Vcc - Vce0}{i19}$ = 4.8 kΩ (valeur normalisée: 5.6kΩ)

Réalisation des PCB sous Altium Designer[1]

Une fois ces montages testés et validés nous nous sommes intéressés à la conception des cartes électroniques de la maquette.

Nous avons fait le choix de faire plusieurs petites cartes pour les boutons et voyants plutôt qu'une seule et même carte afin de faciliter les dépannages éventuels. En effet, il est plus simple de dépanner une carte électronique contenant seulement 2 boutons que dépanner une seule carte contenant la totalité des boutons de la maquette.

C'est pourquoi chaque carte possède les montages de deux boutons et de deux voyants représentants un seul et même étage. Concernant l'étage 1 et l'étage 5 (qui ne comporte qu'un seul bouton voyant) nous les avons disposés sur la même carte.

La maquette possédant 13 boutons poussoirs et voyants (boutons à chaque étage ainsi que les boutons cabine), nous avons un autre PCB similaire, mais gérant seulement un bouton.

Nous avons réalisé un troisième PCB qui permet de gérer l'adaptation des niveaux de tension pour le capteur effet Hall. Cette carte permet de gérer aussi la commande du moteur et la gestion de l'ouverture et fermeture des portes de chaque étage modélisées par des interrupteurs deux positions.

Nous avons également réalisé un PCB pour l'alimentation générale de la maquette. Cette carte permet d'alimenter le moteur et l'Arduino avec une tension de 7.5V, les différents composants présents sur les cartes sont alimentés avec une tension de 5V.

Nous avons ajouté un fusible de 2000 mA afin de protéger l'alimentation universelle de toute éventuelle surcharge en courant. De plus, nous avons utilisé une diode de redressement pour aussi protéger cette source de tension contre les tensions négatives générées par le moteur.

[1] Vous trouverez l'ensemble des schématiques et PCB des différentes cartes en ANNEXE 1

Nous avons utilisé un régulateur de référence LM 7805 pour obtenir la tension de 5V. Les condensateurs 330nF en amont et 100nF en aval afin d'éloigner tout risque d'une auto-oscillation du régulateur.

Enfin, une fois les cartes imprimées et soudées, nous sommes passés au câblage de celles-ci.[2]

Réalisation du circuit de protection

Pour éviter la détérioration de la maquette, deux capteurs de fin de course bas et haut de technologie mécanique sont installés. Nous utiliserons des capteurs mécaniques, car la technologie diffère de celle utilisée pour chaque étage.

Ces deux dispositifs sont câblés directement sur l'alimentation du moteur afin d'obtenir une sécurité supplémentaire en cas de problème venant de l'Arduino et de l'automate programmable.

Un bouton d'arrêt d'urgence sera à disposition de l'utilisateur pour couper l'alimentation du moteur en cas de dysfonctionnement.

Le bouton poussoir situé en parallèle des capteurs mécaniques de fin de course haut et bas permet de replacer la cabine dans sa plage de fonctionnement normal. Cette action sera réalisée à l'aide de l'Arduino.

Ce circuit de protection n'est pas géré par l'Arduino ni par l'automate programmable afin que cette protection soit fonctionnelle en cas de défaillance de l'un des deux dispositifs de commande.

Le dispositif de commande Arduino ne peut pas exécuter d'action sur la commande de cette protection.

Les entrées Arduino permettent seulement de savoir si la cabine est en surcourse haute ou basse.

[2] Vous trouverez en ANNEXE 2 la disposition des différentes cartes électroniques dans le coffret et en ANNEXE 3 les informations nécessaires au câblage

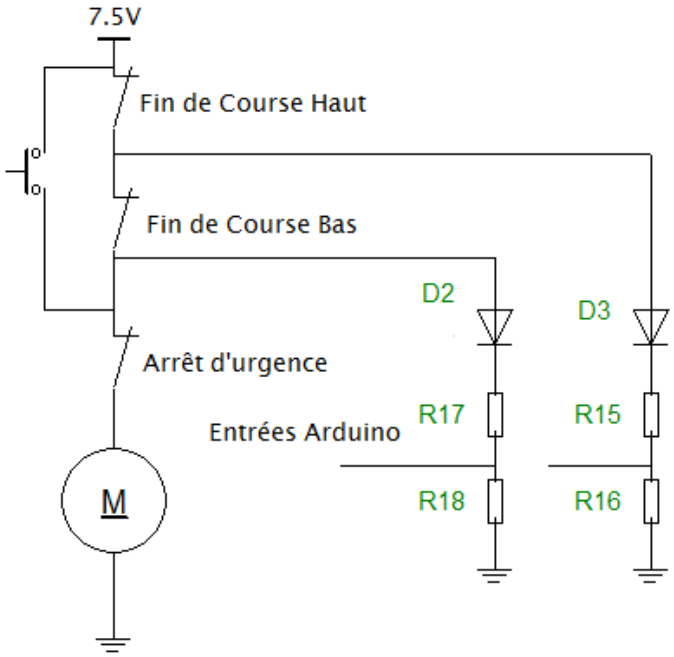


Figure 11 : Schéma du circuit de protection

Nous avons réalisé un pont diviseur de tension afin de visualiser une tension de 5V lors d'un fonctionnement normal (fin de course haut et bas normalement fermé). Ce niveau de tension sera utilisé par l'Arduino pour visualiser d'éventuels défauts. Les deux diodes permettent d'éviter le changement de sens du courant.

Concernant le dimensionnement des résistances, nous avons choisi des résistances de valeur relativement élevées pour limiter le courant dans ces deux branches.

Calcul des résistances:

On veut que Varduino1 et Varduino2 soient égales à 5V lorsque les deux capteurs de fin de course ne sont pas déclenchés (normalement fermés).

Vcc= 7.5V

Vcc=Vd2+(R15+R16)*i15 (1)

et Varduino1= *R16R15+R16(Vcc-Vd2)*

Vcc=Vd3+(R17+R18)*i17 (2)

et Varduino2= *R18R17+R18(Vcc-Vd3)*

On obtient donc la relation suivante:

0.38 R16 = R15

0.38 R18 = R17

On fixe alors:

R16=R18= 100 kΩ (valeur normalisée)

R15=R17= 39 kΩ (valeur normalisée)

4. Programmation Arduino

Après lecture de la datasheet du moteur AX12+, il apparaît que le moteur est alimenté entre 7 et 10 V. Il est contrôlé via un bus série et un système d'adresse. Il possède 2 connecteurs de 3 PINs:

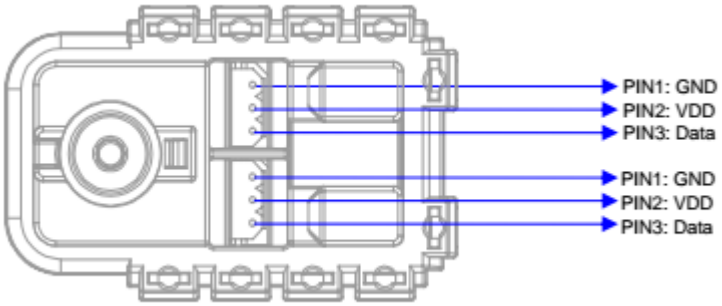


Figure 12 : Illustration du moteur AX12+

Le second connecteur est nécessaire pour l'utilisation de plusieurs moteurs en parallèle.

Au vu de notre application, ce second connecteur n'est pas utilisé.

Cet actionneur sera donc commandé par l'Arduino. Il sera alors nécessaire d'utiliser les pin 0(RX0) et pin 1 (TX0) par exemple. Le connecteur Data transmettra les données série TX et RX alternativement.

Pour notre application, l'utilisation de la pin 0 (RX0) n'est pas réellement nécessaire, car nous ne souhaitons pas obtenir d'informations de retour.

Le câblage de l'Arduino est le suivant, mais les entrées "Data Control" et "RX" ne seront pas utilisées. La commande du moteur sera effectuée par l'entrée TX (verte). L'alimentation se fait par un dispositif extérieur.

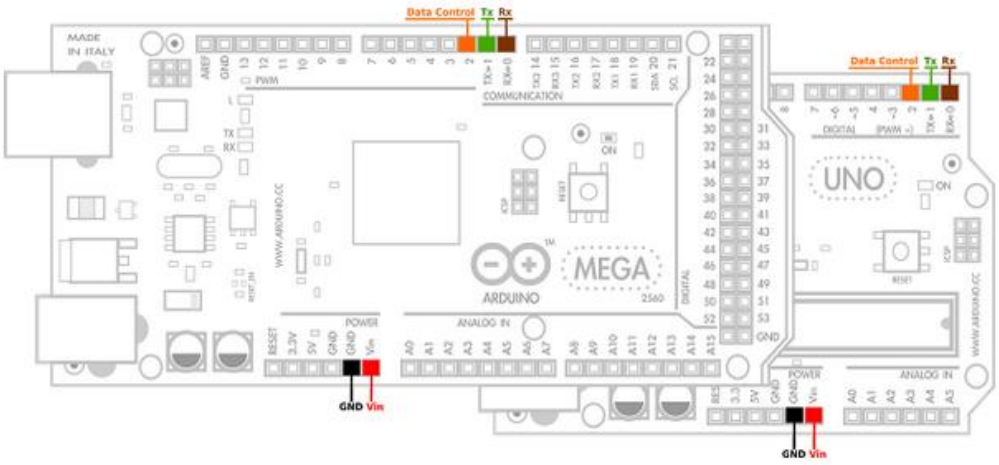


Figure 13 : Câblage du moteur AX12 avec Arduino

Pour la commande de ce servomoteur, nous avons utilisé la librairie "Dynamixel.h".

Cette librairie met à disposition des fonctions suivantes :

Dynamixel.setEndless(ID,Status): permet d'activer le fonctionnement du moteur en "mode continu".

Dynamixel.turn(ID,Side,Speed) : permet de faire tourner le moteur de manière continue en précisant le sens et la vitesse.

Dynamixel.move(ID,Position) : permet de réaliser une rotation du moteur en indiquant la position en paramètre.

Le moteur utilisé est repéré par son identifiant (ID), car il est possible de commander plusieurs moteurs simultanément.

Nous allons seulement envoyer des commandes de l'Arduino vers le moteur donc nous utiliserons seulement le port TX0 de l'Arduino qui sera connecté au port DATA du servomoteur.

Pour notre application, il est nécessaire d'utiliser le servomoteur en fonctionnement continu. Il tourne en continu grâce à la fonction "setEndless()".

Réalisation du programme de démonstration[3]

Au vu du nombre d'entrées sorties de l'Arduino, nous n'avons pas connecté tous les composants de la maquette. Cependant le fonctionnement serait le même avec le reste des composants. Seuls les boutons de l'étage 1, 4 est 5 sont connectés. Pour pouvoir tester leur bon fonctionnement, tous les voyants sont connectés à l'Arduino. Le bouton test est câblé directement sur l'Arduino, son appui permet d'allumer tous les voyants présents sur la maquette (les voyants des appels cabine et les voyants des appels extérieurs).

Le scénario choisi initialement est assez basique, mais permet de valider le fonctionnement des divers composants de la maquette :

L'utilisateur peut appeler l'ascenseur à l'étage 1, l'étage 4 ou l'étage 5. La cabine se déplace selon sa position initiale. Tant que la cabine n'est pas arrivée à destination, les voyants étages et cabine sont allumés. Dès qu'elle est arrivée à l'étage demandé, les voyants s'éteignent et l'utilisateur peut de nouveau choisir l'étage souhaité en sélectionnant un des boutons poussoirs de la cabine.

À tout moment si un des interrupteurs simulant l'ouverture et la fermeture des portes de chaque étage est resté en position « porte ouverte », le déplacement de la cabine est impossible.

[3] Vous trouverez la totalité du code du programme en ANNEXE 4

Ressentis vis-à-vis du projet

1. Difficultés rencontrées et solutions choisies

Lors des premiers tests qui ont suivi la réception de notre commande, nous avons été confrontés à un premier problème concernant les capteurs à effet Hall.

En effet, ceux commandés donnent une réponse “numérique” c’est-à-dire que s’il y a présence d’un aimant ou d’un matériel modifiant le champ magnétique, la sortie va être de type tout ou rien (la tension de sortie va alors correspondre à la tension d’alimentation). La finalité est donc ce que l’on souhaitait. C’est concernant le mode de fonctionnement et donc le mode de détection ou non d’un objet que n’avons pas obtenu les résultats souhaités.

Les capteurs à effet Hall commandés sont de type bipolaire, il est alors possible de les déclencher selon n’importe quel type de pôles que ce soit un pôle Nord ou Sud. Cependant, le capteur, une fois déclenché, garde l’état et nécessite la présence du pôle opposé pour revenir à l’état initial.

Cet inconvénient pouvait cependant être rectifié au niveau de la programmation de l’Arduino. Malheureusement, concernant la partie automate ce n’était pas réalisable étant donné que l’automate attend à ses entrées, une valeur 1 (état haut).

Finalement, nous avons pu régler assez facilement ce problème, en commandant des échantillons gratuits sur le site Texas Instruments.

Les nouveaux capteurs sont cette fois-ci, unipolaires et donc ne réagissent qu’à un seul pôle. Le capteur ne se déclenche qu’à l’approche d’un aimant et il suffit simplement d’éloigner l’aimant pour revenir à l’état initial.

Après l’impression du PCB, nous avons constaté un problème en sortie du capteur. Ce dispositif permet bien de transmettre une information pour obtenir la tension en sortie de l’optocoupleur de 24V. La tension en sortie du capteur était de 1.1V ce qui était bien trop faible comparé à la tension initiale de 5V.

Pour corriger ce problème, nous avons dû ajouter un montage pour isoler la sortie du capteur et l’optocoupleur. Le montage utilisé est à base d’un amplificateur opérationnel monté en suiveur. On obtient ainsi à sa sortie une tension de 5V (tension d’alimentation) lors de l’absence de l’aimant devant celui-ci. Cette information sera utilisée par l’Arduino. Le montage avec l’optocoupleur a aussi pour fonction d’inverser les niveaux de tension fournis par le capteur et donc d’obtenir 24V lors de la présence de l’aimant devant le capteur effet Hall.

Enfin du point de vue de la réalisation du projet, une autre difficulté fut la gestion du temps. En effet, si nous faisons un bilan de fin de projet, en comparant le planning réel au planning prévisionnel des semaines restantes depuis la soutenance intermédiaire (figure 14), nous pouvons constater que notre avancement n'a pas été aussi linéaire que celui initialement prévu.

Au final, il aurait été nécessaire de détailler plus précisément les différentes étapes de réalisation, par exemple l'étape de câblage de la maquette qui nous a demandé plusieurs jours au vu du travail à effectuer.

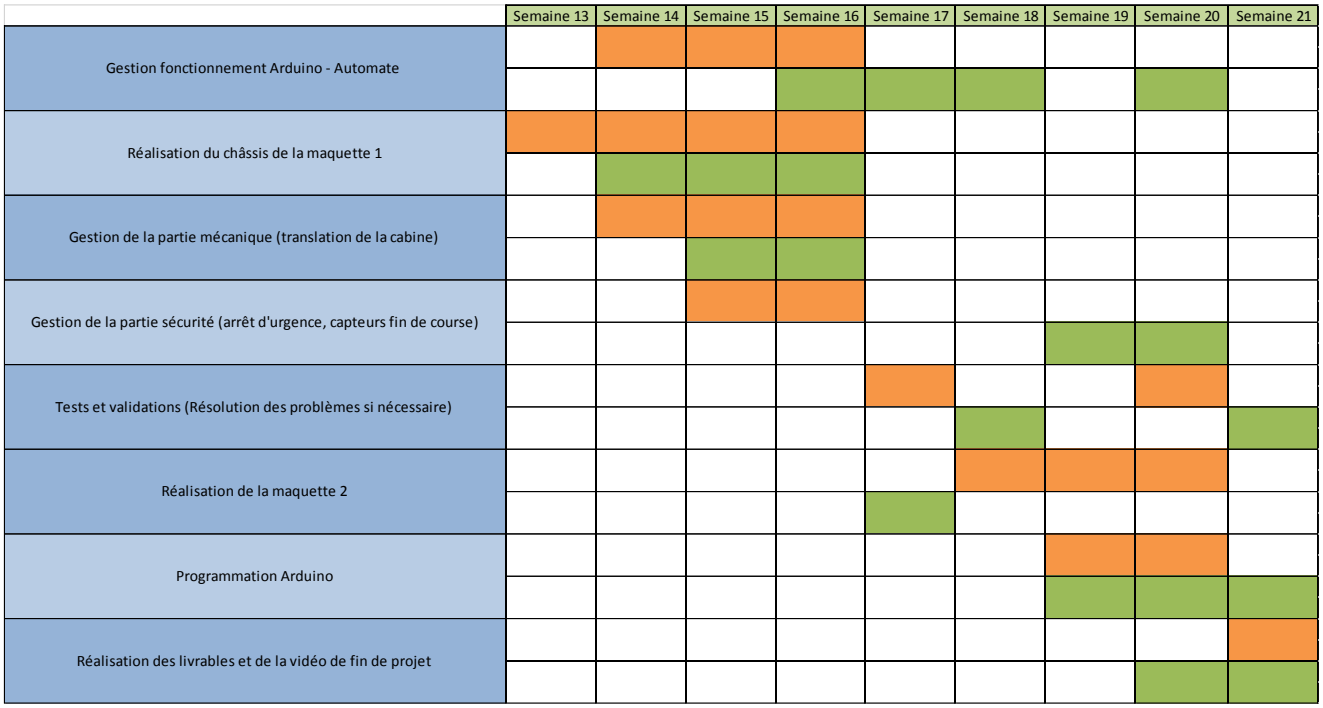


Figure 14 : Planning prévisionnel et réel

2. Améliorations et perspectives possibles

Plusieurs améliorations peuvent être réalisées afin d'obtenir une maquette plus aboutie et respectant encore plus le cahier des charges initial.

Tout d'abord, le choix de planches de mélaminé plus fines ou de MDF permettrait d'utiliser la découpe laser du Fabricarium de l'école pour tout d'abord obtenir une découpe précise et ainsi obtenir une maquette finale plus esthétique.

Une autre solution aurait été de réaliser un coffret avec les circuits déportés. Cette alternative permettrait un câblage plus aisé et un dépannage plus rapide de la maquette.

Pour faciliter le repérage des câbles et surtout éviter un nombre beaucoup trop important de fils, il serait intéressant d'utiliser des câbles en nappe et donc de modifier les PCB des cartes électroniques.

Concernant la conception des cartes électroniques, prévoir des trous de fixation serait un point non négligeable.

De plus, il serait intéressant de modéliser les portes physiquement au lieu d'avoir des interrupteurs pour une utilisation plus réaliste.

Enfin, le choix d'un bouton d'arrêt d'urgence en technologie normalement fermée serait pertinent afin de rendre son utilisation plus intuitive.

Conclusion

En conclusion, nous avons répondu à la problématique qui nous a été posée en réalisant une maquette avec un fonctionnement plutôt satisfaisant.

Suite à quelques problèmes rencontrés lors du déroulement du projet, nous n'avons malheureusement pas pu réaliser une seconde maquette dans sa totalité. En effet, seul le bâti est pour le moment réalisé.

D'un point de vue "humain", ce projet nous a permis de travailler en binôme, avec les difficultés qui en découlent (être en accord avec l'autre, recherche d'efficacité...).

Il nous a également confrontés aux difficultés de la gestion de projets et ses aléas (gestion d'un planning, délais de livraison des composants, ainsi que la gestion du temps qui nous est attribué).

Ce projet nous a particulièrement intéressés, car nous avons pu travailler sur toutes les étapes de réalisation d'un système automatisé: la définition d'un cahier des charges, l'étude, la réalisation de la maquette et sa mise en service.

Enfin, ce projet a été très enrichissant, car il nous a donné la possibilité de travailler dans différents domaines tels que : l'électronique, la conception et la programmation Arduino.

Bibliographie

- Mapping Arduino
https://www.arduino.cc/en/Hacking/PinMapping2560

- Librairie Dynamixel
http://savageelectronics.blogspot.fr/2011/01/arduino-y-dynamixel-ax-12.html

- Description de la librairie Dynamixel
http://austinlpalmer.com/Projects/Documentr/#/home

- Datasheet Servomoteur AX12+
http://www.trossenrobotics.com/images/productdownloads/AX-12%28English%29.pdf

- Datasheet Capteur effet Hall
http://www.ti.com/lit/ds/symlink/drv5023.pdf

- Datasheet Optocoupleur
http://docs-europe.electrocomponents.com/webdocs/13e4/0900766b813e4118.pdf

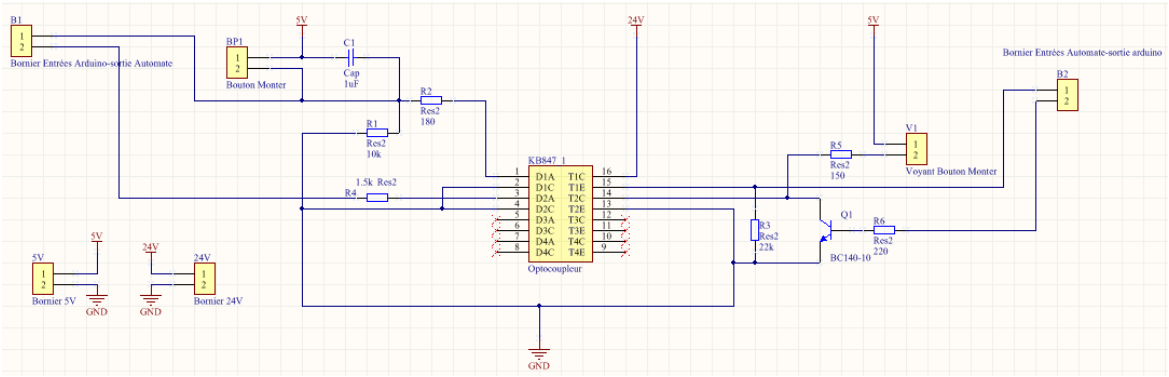
- Datasheet Amplificateur Opérationnel
http://www.st.com/web/en/resource/technical/document/datasheet/CD00000489.pdf

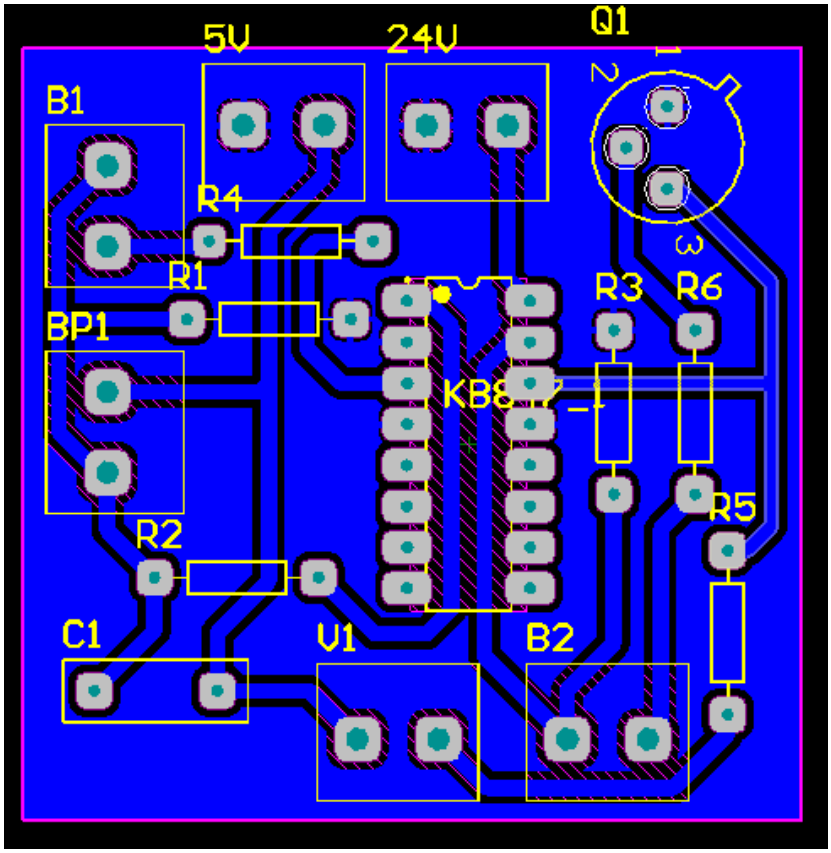
- Datasheet Régulateur
http://www.ti.com/lit/ds/symlink/lm7805c.pdf

Annexes

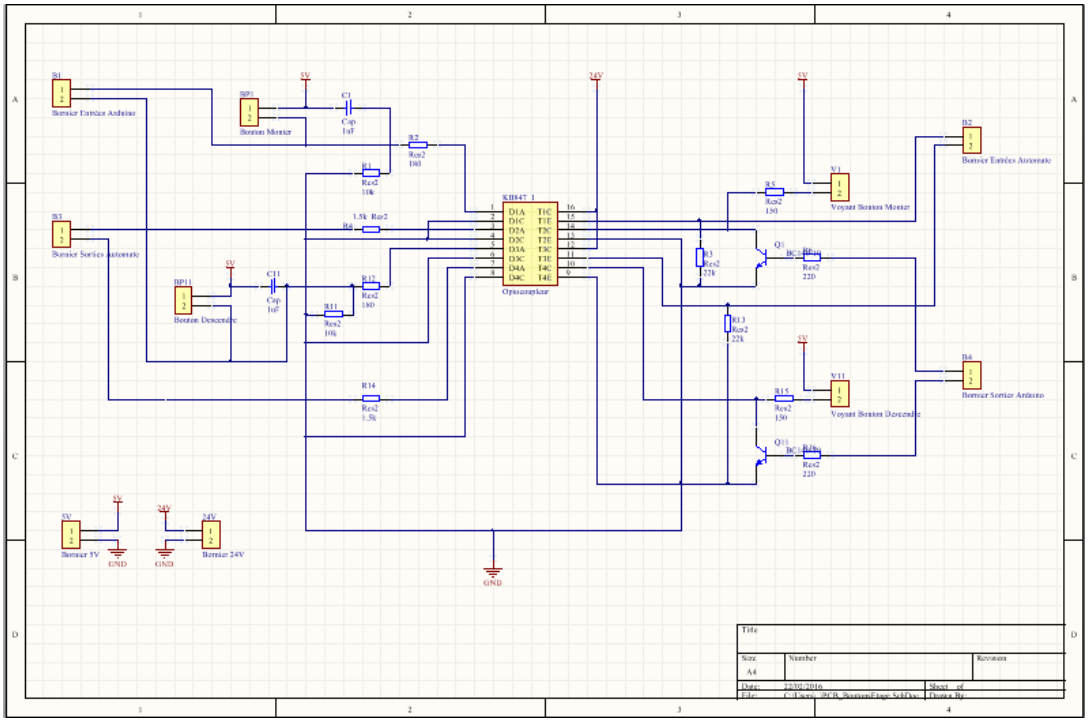
Annexe 1 : Schéma et PCB des cartes électroniques

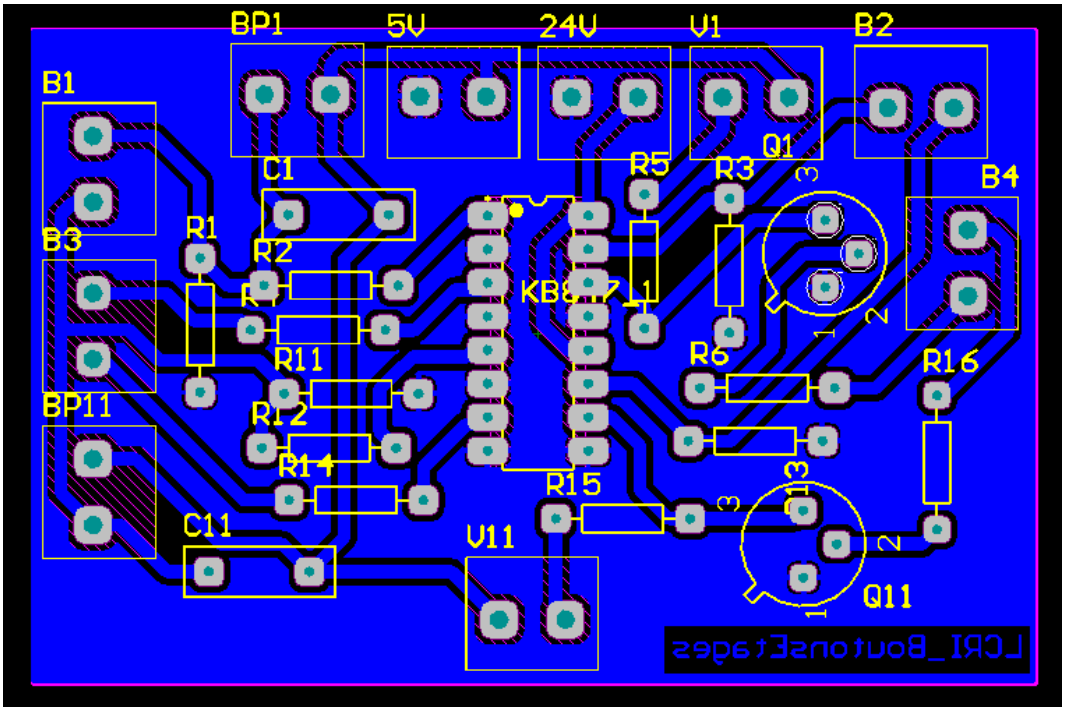
(Carte avec un seul bouton voyant)



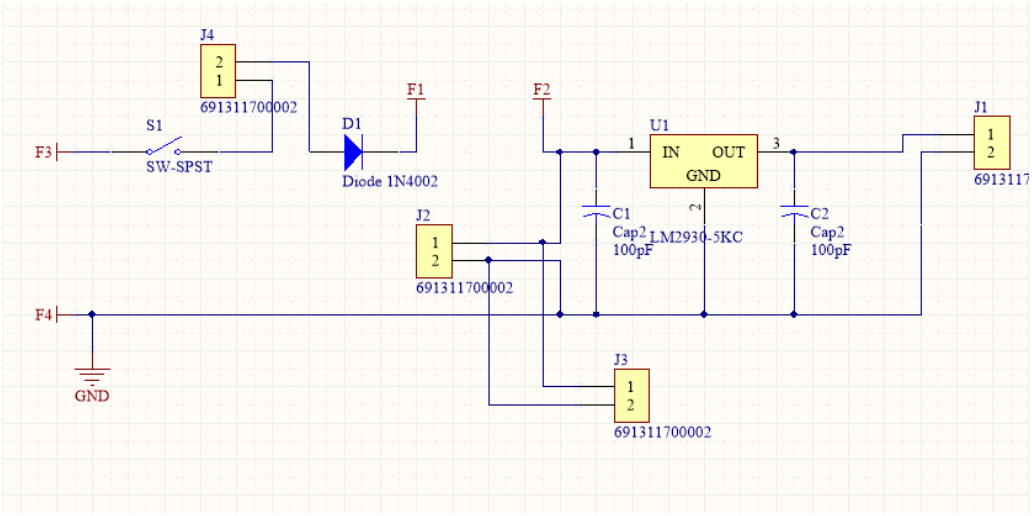


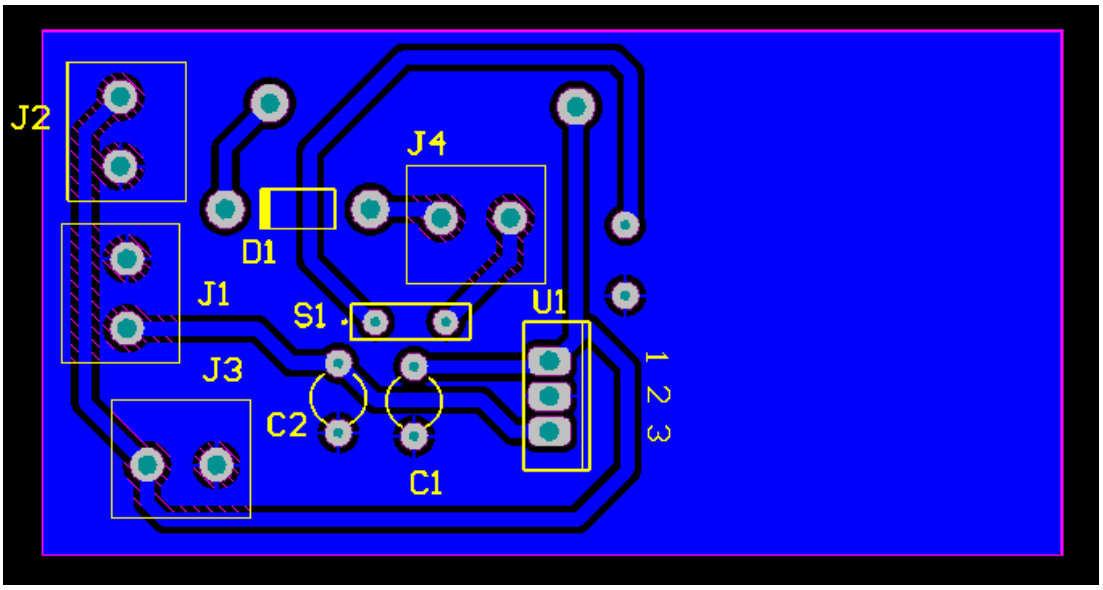
(Carte avec deux boutons voyants)



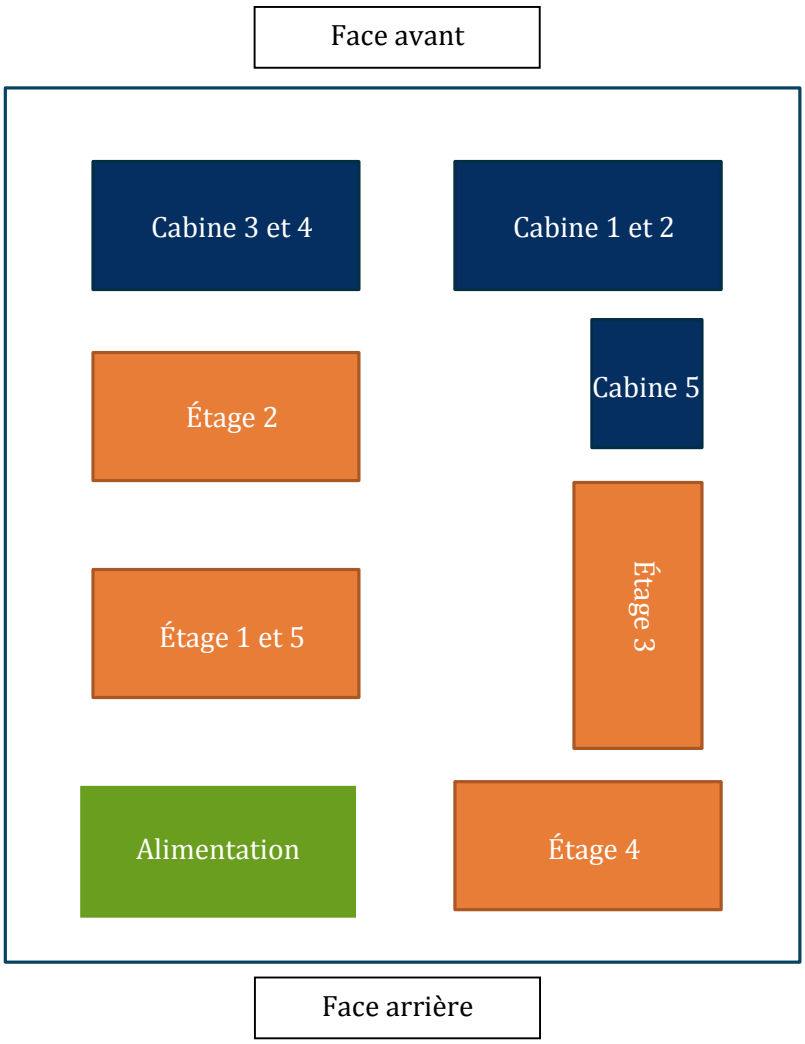


(Carte alimentation)





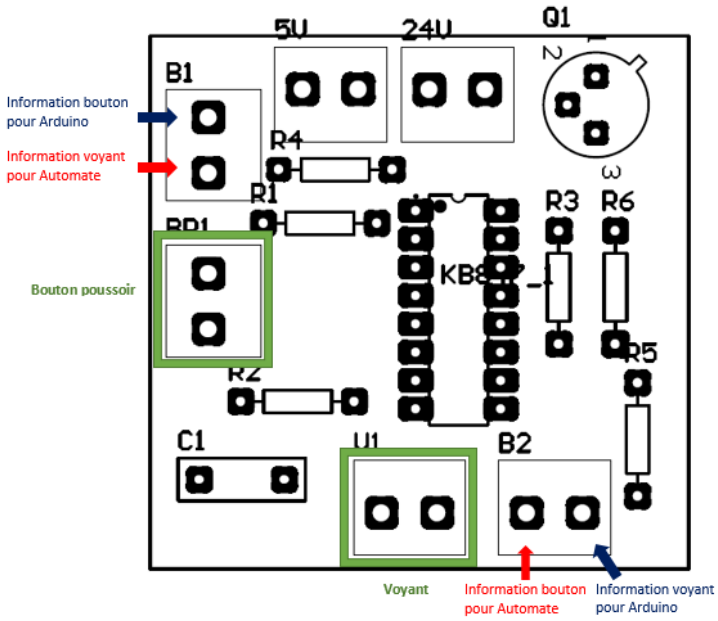
Annexe 2 : Disposition des cartes électroniques



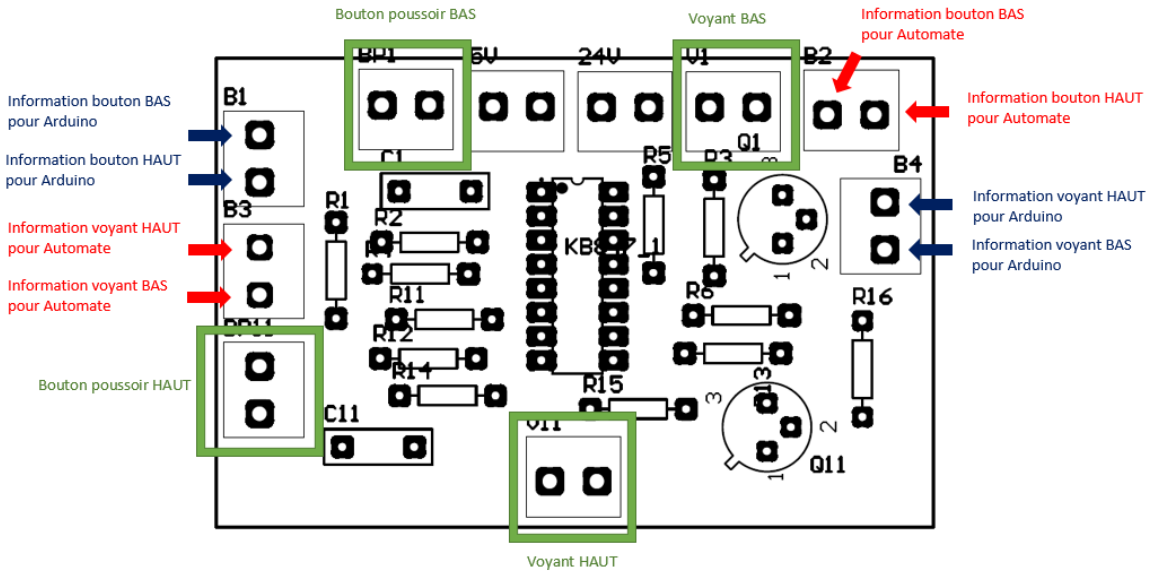
Annexe 3 : Informations pour câblage des différentes cartes

Positions des borniers pour câblage des cartes électroniques gérant les boutons étage et les boutons cabine

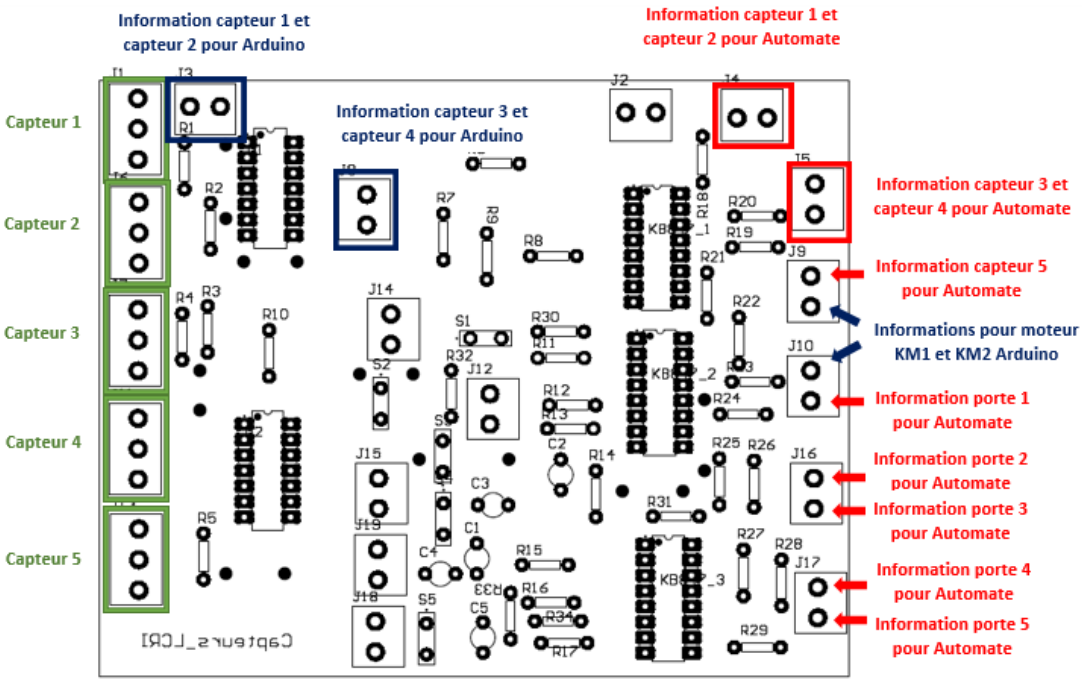
(Carte avec un seul bouton voyant)



(Carte avec deux boutons voyants)



Positions des borniers pour câblage de la carte électronique gérant les capteurs étages ainsi que les interrupteurs modélisant les portes



L'information capteur 5 pour Arduino est sur le bornier J14-1. L'information capteur 5 pour Automate est sur J9-1.

L'information KM1 et KM2 pour moteur pour Automate est sur le bornier J12.

Enfin les borniers J15, J18 et J19-1 sont pour les informations portes pour l'Arduino.

Annexe 4 : Code du programme Arduino utilisé pour le mode démonstration

```
#include <DynamixelSerial.h>

//Déclarations des variables "externes" au programme (entrées-sorties Arduino)
//Déclaration des capteurs
int capt1 = 39;                 //Le capteur de l'étage 1 est sur la PIN 39
int capt2 = 38;                 //Le capteur de l'étage 2 est sur la PIN 38
int capt3 = 37;                 //Le capteur de l'étage 3 est sur la PIN 37
int capt4 = 36;                 //Le capteur de l'étage 4 est sur la PIN 36
int capt5 = 35;                 //Le capteur de l'étage 5 est sur la PIN 35

//Déclaration du bouton TEST des voyants
int btest = 44;                 //Le bouton test est sur la PIN 44

//Déclaration des boutons étages
//(Les boutons étages 1,3bas, 3haut, 4bas, 4haut et 5 sont câblés)
int be1 = 31;                   //Le bouton étage 1 est sur la PIN 31
int be4bas = 49;                //Le bouton étage 4bas est sur la PIN 49
int be4haut = 50;               //Le bouton étage 4haut est sur la PIN 50
int be5 = 33;                   //Le bouton étage 5 est sur la PIN 33

//Déclaration des boutons cabine
//(Les boutons étages 1,3bas, 3haut, 4bas, 4haut et 5 sont câblés)
int bc1 = 46;                   //Le bouton cabine 1 est sur la PIN 46
int bc4 = 29;                   //Le bouton cabine 4 est sur la PIN 29
int bc5 = 45;                   //Le bouton cabine 5 est sur la PIN 45

//Déclarations des interrupteurs simulants l'ouverture des portes
int p1 = 7;                     //L'interrupteur simulant la porte de l'étage 1 est sur la PIN
7
int p2 = 6;                     //L'interrupteur simulant la porte de l'étage 2 est sur la PIN
6
int p3 = 5;                     //L'interrupteur simulant la porte de l'étage 3 est sur la PIN
5
int p4 = 4;                     //L'interrupteur simulant la porte de l'étage 4 est sur la PIN
4
int p5 = 3;                     //L'interrupteur simulant la porte de l'étage 5 est sur la PIN
3
```

```
//Déclaration des voyants des boutons étage
int ve1 =  23;              //Le voyant du bouton étage 1 est sur la PIN 23
int ve2bas = 28;            //Le voyant du bouton étage 2bas est sur la PIN 28
int ve2haut = 26;           //Le voyant du bouton étage 2haut est sur la PIN 26
int ve3bas = 53;            //Le voyant du bouton étage 3bas est sur la PIN 53
int ve3haut = 48;           //Le voyant du bouton étage 3haut est sur la PIN 48
int ve4bas = 32;            //Le voyant du bouton étage 4bas est sur la PIN 32
int ve4haut = 30;           //Le voyant du bouton étage 4haut est sur la PIN 30
int ve5 = 24;               //Le voyant du bouton étage 5 est sur la PIN 24

//Déclaration des voyants des boutons cabine
int vc1 = 52;               //Le voyant du bouton cabine 1 est sur la PIN 52
int vc2 = 51;               //Le voyant du bouton cabine 2 est sur la PIN 51
int vc3 = 27;               //Le voyant du bouton cabine 3 est sur la PIN 27
int vc4 = 25;               //Le voyant du bouton cabine 4 est sur la PIN 25
int vc5 = 47;               //Le voyant du bouton cabine 5 est sur la PIN 47

//Déclaration des segments de l'afficheur
int a = 14;
int b = 15;
int c = 16;
int d = 17;
int e = 18;
int f = 19;
int g = 20;

//Déclarations des variables "internes" au programme
int etage_demande = 0;      //Variable représentant l'étage demandé
int position_cabine = 0;    //Valeur de la position (étage) où se situe la cabine
bool arretCabine = LOW;     //Variable représentant l'arrêt de la cabine
int sens = 0;               //Variable permettant de connaitre le sens du moteur

int etatMoteur = 0;

//Fonction liée au bouton TEST permettant de tester le bon fonctionnement de tous les voyants de la maquette
void Test_Voyant() {
  if (digitalRead(btest) == HIGH) {     //Si appui du bouton TEST...
    digitalWrite(ve1, HIGH);            //...on allume tous les voyants étage
    digitalWrite(ve2bas, HIGH);
    digitalWrite(ve2haut, HIGH);
    digitalWrite(ve3bas, HIGH);
```

```
    digitalWrite(ve3haut, HIGH);
    digitalWrite(ve4bas, HIGH);
    digitalWrite(ve4haut, HIGH);
    digitalWrite(ve5, HIGH);

    delay(1000);

    digitalWrite(ve1, LOW);                    //...on éteint tous les voyants étage de la maquette
    digitalWrite(ve2bas, LOW);
    digitalWrite(ve2haut, LOW);
    digitalWrite(ve3bas, LOW);
    digitalWrite(ve3haut, LOW);
    digitalWrite(ve4bas, LOW);
    digitalWrite(ve4haut, LOW);
    digitalWrite(ve5, LOW);

    digitalWrite(vc1, HIGH);          //...puis on allume tous les voyants cabine
    digitalWrite(vc2, HIGH);
    digitalWrite(vc3, HIGH);
    digitalWrite(vc4, HIGH);
    digitalWrite(vc5, HIGH);

    delay(1000);
    digitalWrite(vc1, LOW);
    digitalWrite(vc2, LOW);
    digitalWrite(vc3, LOW);
    digitalWrite(vc4, LOW);
    digitalWrite(vc5, LOW);
   }
  }

  //Fonction gérant l'appui des boutons
  void Appui_Bouton() {
   if ((digitalRead(be1) == HIGH) || (digitalRead(bc1) == HIGH)) {
                                                  //En fonction du bouton appuyé.....
    etage_demande = 1;                            //...on écrit la valeur de l'étage demandé
   }
   if ((digitalRead(be4bas) == HIGH) || (digitalRead(be4haut) == HIGH) || (digitalRead(bc4) == HIGH)) {
    etage_demande = 4;
```

```
  }
  if ((digitalRead(be5) == HIGH) || (digitalRead(bc5) == HIGH)) {
   etage_demande = 5;
  }
  if ((etage_demande == 1) && (position_cabine != 1)) {
   digitalWrite(ve1, HIGH);
   digitalWrite(vc1, HIGH);
  }
  if ((etage_demande == 4) && (position_cabine != 4)) {
   digitalWrite(ve4haut, HIGH);
   digitalWrite(ve4bas, HIGH);
   digitalWrite(vc4, HIGH);
  }
  if ((etage_demande == 5) && (position_cabine != 5)) {
   digitalWrite(ve5, HIGH);
   digitalWrite(vc5, HIGH);
  }
  if ((digitalRead(ve1) == HIGH) && (position_cabine == 1)) {
                                                    //Si la cabine est à l'étage souhaité...
   digitalWrite(ve1, LOW);                                  //...on éteint les voyants
correspondants
   digitalWrite(vc1, LOW);
  }

  if ((digitalRead(ve4bas) == HIGH) && (position_cabine == 4)) {
   digitalWrite(ve4bas, LOW);
   digitalWrite(ve4haut, LOW);
   digitalWrite(vc4, LOW);
  }

  if ((digitalRead(ve5) == HIGH) && (position_cabine == 5)) {
   digitalWrite(ve5, LOW);
   digitalWrite(vc5, LOW);
  }
 }

 //Affichage de l'étage sur l'afficheur 7 segments
 void AffichageEtage(int position_cabine) {
  switch (position_cabine) {
   case 1:                    //Code pour afficher 1
    digitalWrite(a, LOW);
    digitalWrite(b, HIGH);
```

```
  digitalWrite(c, HIGH);
  digitalWrite(d, LOW);
  digitalWrite(e, LOW);
  digitalWrite(f, LOW);
  digitalWrite(g, LOW);
  break;
 case 2:                  //Code pour afficher 2
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, LOW);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);
  digitalWrite(f, LOW);
  digitalWrite(g, HIGH);
  break;
 case 3:                  //Code pour afficher 3
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, LOW);
  digitalWrite(f, LOW);
  digitalWrite(g, HIGH);
  break;
 case 4:                  //Code pour afficher 4
  digitalWrite(a, LOW);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, LOW);
  digitalWrite(e, LOW);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
  break;
 case 5:                  //Code pour afficher 5
  digitalWrite(a, HIGH);
  digitalWrite(b, LOW);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, LOW);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
  break;
```

```
      default:
       break;
     }
    }

    //Fonction permettant de connaitre la position de la cabine
    void Gestion_Cabine() {
     int i = 0;
     for (i = 1; i <= 5; i++) {
      if (digitalRead(capt1) == LOW) {        //Si le capteur 1 est déclenché(logique inversée)...
       position_cabine = 1;                   //...la cabine se trouve à l'étage 1
       AffichageEtage(position_cabine);
      }
      if (digitalRead(capt2) == LOW) {    //Si le capteur 2 est déclenché (logique inversée)...
       position_cabine = 2;                   //...la cabine se trouve à l'étage 2
       AffichageEtage(position_cabine);
      }
      if (digitalRead(capt3) == LOW) { //Si le capteur 3 est déclenché (logique inversée)...
       position_cabine = 3;                   //...la cabine se trouve à l'étage 3
       AffichageEtage(position_cabine);
      }
      if (digitalRead(capt4) == LOW) {      //Si le capteur 4 est déclenché (logique inversée)...
       position_cabine = 4;                   //...la cabine se trouve à l'étage 4
       AffichageEtage(position_cabine);
      }
      if (digitalRead(capt5) == LOW) {      //Si le capteur 5 est déclenché (logique inversée)...
       position_cabine = 5;                   //...la cabine se trouve à l'étage 5
       AffichageEtage(position_cabine);
      }
     }
    }

    //Fonction permettant de gérer les arrêts de la cabine
    void gestion_arret() {
     arretCabine = LOW;
     if (((digitalRead(p1) == HIGH) || (digitalRead(p2) == HIGH) || (digitalRead(p3) == HIGH) || (digitalRead(p4) == HIGH) || (digitalRead(p5) == HIGH))) {
```

```
   arretCabine = HIGH;
  }
  if ((digitalRead(ve1) == HIGH) && (position_cabine == 1)) {
   arretCabine = HIGH;
  }
  if ((digitalRead(ve4bas) == HIGH) && (position_cabine == 4)) {
   arretCabine = HIGH;
  }
  if ((digitalRead(ve5) == HIGH) && (position_cabine == 5)) {
   arretCabine = HIGH;
  }
}

//Fonction permettant la rotation vers la droite en continu du moteur
void monter() {
  Dynamixel.turn(18, LEFT, 1000);
}

//Fonction permettant la rotation vers la gauche en continu du moteur
void descendre() {
  Dynamixel.turn(18, RIGTH, 800);
}

//Fonction permettant l'arrêt du moteur
void arret() {
  Dynamixel.turn(18, 0, 0);
}

//Fonction permettant de gérer le sens de déplacement de la cabine
void Gestion_Deplacement() {
  if ((sens == 1) && (etage_demande == position_cabine)) {
    sens = 0;
  }
  if ((sens == 2) && (etage_demande == position_cabine)) {
    sens = 0;
  }

  if ((sens == 0) && (etage_demande > position_cabine)) {
    sens = 1;
  }

  if ((sens == 0) && (etage_demande < position_cabine)) {
```

```
    sens = 2;
   }
  }

  //Fonction permettant de commander le moteur selon le sens souhaité
  void Gestion_Moteur() {
   if (arretCabine == HIGH) {
    sens = 0;
   }
   if (etatMoteur != sens) {
    if (sens == 1) {
     monter();
    }
    if (sens == 2) {
     descendre();
    }
    if (sens == 0) {
     arret();
    }
    etatMoteur = sens;
   }
  }

  //Fonction setup
  void setup() {

   //Déclarations des en entrées des pins reliées aux différents boutons, capteurs et interrupteurs
    pinMode(capt1, INPUT);          //Capteurs
    pinMode(capt2, INPUT);
    pinMode(capt3, INPUT);
    pinMode(capt4, INPUT);
    pinMode(capt5, INPUT);
    pinMode(captFCB, INPUT);
    pinMode(captFCH, INPUT);

    pinMode(be1, INPUT);            //Boutons étage
    pinMode(be4bas, INPUT);
    pinMode(be4haut, INPUT);
    pinMode(be5, INPUT);
    pinMode(bc1, INPUT);            //Boutons cabine
    pinMode(bc4, INPUT);
```

```
        pinMode(bc5, INPUT);
        pinMode(btest, INPUT);          //Bouton Test

        pinMode(p1, INPUT);             //Interrupteurs portes
        pinMode(p2, INPUT);
        pinMode(p3, INPUT);
        pinMode(p4, INPUT);
        pinMode(p5, INPUT);

        //Déclarationsdes en sorties des pins reliées aux différents voyants
        pinMode(ve1, OUTPUT);           //Voyants Boutons étage
        pinMode(ve2bas, OUTPUT);
        pinMode(ve2haut, OUTPUT);
        pinMode(ve3bas, OUTPUT);
        pinMode(ve3haut, OUTPUT);
        pinMode(ve4bas, OUTPUT);
        pinMode(ve4haut, OUTPUT);
        pinMode(ve5, OUTPUT);

        pinMode(vc1, OUTPUT);           //Voyants Boutons cabine
        pinMode(vc2, OUTPUT);
        pinMode(vc3, OUTPUT);
        pinMode(vc4, OUTPUT);
        pinMode(vc5, OUTPUT);

        pinMode(a, OUTPUT);             //7 segments de l'afficheur
        pinMode(b, OUTPUT);
        pinMode(c, OUTPUT);
        pinMode(d, OUTPUT);
        pinMode(e, OUTPUT);
        pinMode(f, OUTPUT);
        pinMode(g, OUTPUT);

        //Configuration des différents paramètres du moteur AX12+
        Dynamixel.begin(1000000);
        Dynamixel.setTempLimit(18, 80);          // Température maximale de 80 degrés
        Dynamixel.setVoltageLimit(18, 65, 160);  //Tension comprise entre 6.5v et 16v
        Dynamixel.setMaxTorque(18, 512);         // 50% de couple
        Dynamixel.setSRL(18, 2);                 // SRL à ALL
        Dynamixel.setEndless(18, ON);            // Activation du mode continu du
moteur
    }
```

```
void loop() {
  Test_Voyant();                  //Appel des différentes fonctions en "parallèle"
  Gestion_Cabine();
  Appui_Bouton();
  gestion_arret();
  Gestion_Deplacement();
  Gestion_Moteur();
}
```