Rapport de projet

Charles-Eloi SMITH

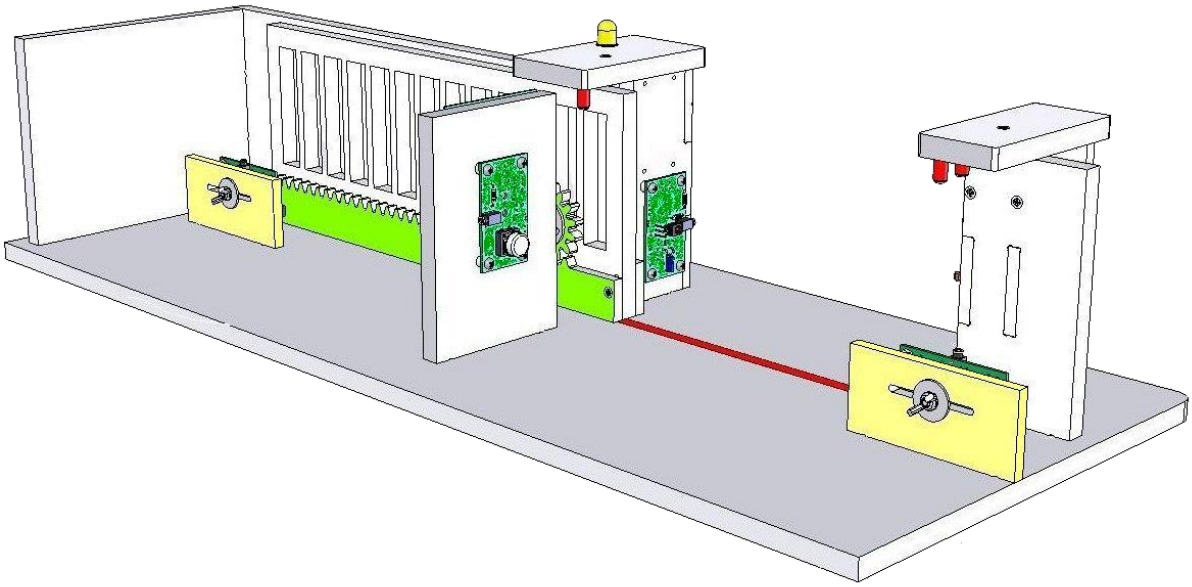
Hikaru GOTO

Wassila IMAMBACCUS

Yaofang WANG

1ère année ENSEA

« Conception d'un portail automatique »



Encadré par Mr Lounis KESSAL

2016-2017

ENSEA
Ecole Nationale
Supérieure
de l'Electronique
et de ses Applications

Remerciements

Nous tenons à remercier notre encadrant pour ce projet, Monsieur Kessal qui nous a bien aidé. De même, nous tenons à remercier les techniciens des labo du bâtiment C, Monsieur Beguin et Monsieur Nganga, ainsi que ceux du bâtiment D, pour leur disponibilité et leur aide, sur le plan technique et matériel.

Rapport de projet

ENSEA
Ecole Nationale
Supérieure
de l'Electronique
et de ses Applications

Sommaire

ENSEA
Ecole Nationale
Supérieure
de l'Electronique
et de ses Applications

Introduction

Le projet d'électronique en 1ère année à l'ENSEA est plus important qu'il le semble. Libre le choix de choisir entre un sujet personnel et un sujet proposé par l'encadrant, mais le résultat reste la même : nous (les étudiants) sommes confronté à un atelier où l'on est chef de nous-même de a à z. C'est donc une occasion parfaite pour se rapprocher de la vie professionnel d'un ingénieur, qui est constamment confronté à ce genre de situation, ce qui rend l'atelier très instructif et intéressant.
C'est pour cela que nous, deux binômes de la 1G1TD1TP2, avons choisi de faire du mieux de notre possible pour tirer le plus de cette expérience. Pour cela, nous voulions un sujet abordable, tout en restant intéressant et utile dans notre vie. Ainsi nous nous sommes décidé d'étudier le fonctionnement et la conception d'un portail automatique. Ce sujet, qui était un des sujets que proposait notre encadrant, a su tout de suite attiré notre attention ne reste que par son degré de liberté inégal aux restes, mais aussi par sa fonction courante mais importante dans la vie de tous les jours.
En effet, les portails automatiques sont très répandus, et tous les bâtiments en ont, même pour les plus petits. Mais pour beaucoup, son fonctionnement reste méconnu. Ce qui était notre cas, ce qui nous a poussé à traiter ce sujet.
Notre sujet porte donc sur la conception d'un système d'automatisation d'ouverture et de fermeture d'un portail. Un système très industriel, mais qui cache beaucoup de chose, que nous allons découvrir et tenter de comprendre à travers ce projet.

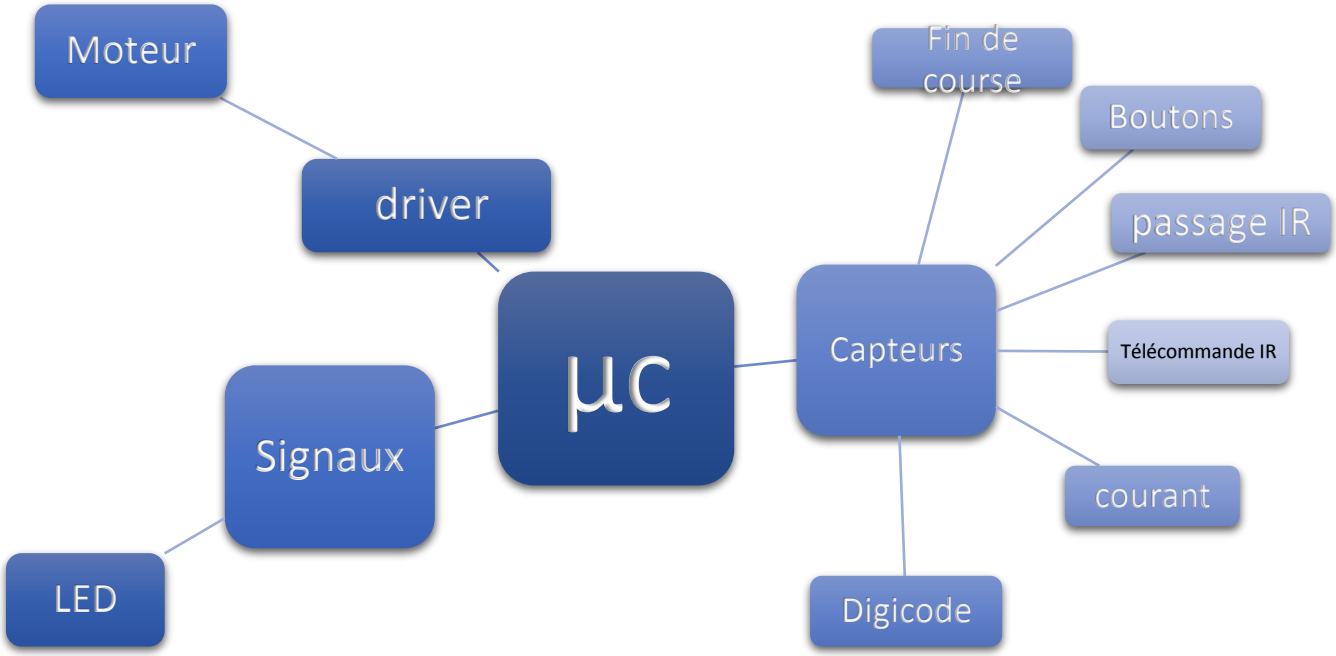
ENSEA
Ecole Nationale Supérieure de l'Electronique et de ses Applications

1. Etude du projet

1.1. Cahier des charges

Notre sujet consiste à commander de manière automatique l'ouverture et la fermeture d'un portail.

Voici un graphe comportant les composants du système final :



Fonctionnement :

- Rapidité : temps de réponse à >0.5s
- Sécurité :
  - Clignotants indiquant chaque fonctionnement
  - Capteur IR / courant moteur
  - Fermeture automatique après un délai
- Confort :
  - Bouton poussoir
  - Digicode
  - Télécommande

ENSEA
Ecole Nationale
Supérieure
de l'Electronique
et de ses Applications

1.2. Inventaire du dispositif

Notre projet n'est pas un projet partant du point zéro : c'est un projet que nous avons succédé de la part d'anciens élèves de l'ENSEA. Ainsi, nous avons pu récupérer leur portail à eux, pour y rajouter nos propres éléments. Ainsi, il s'avère utile de faire un petit inventaire des éléments préexistants que nous avons utilisés :

-la maquette du portail comportant le portail et son mur support

-les boutons poussoirs

-les capteurs de fin de courses

-le moteur

1.3. Les objectifs

Dès le commencement du projet, nous nous sommes fixés un but très précis :
Faire fonctionner ce portail de plusieurs manières et de lui fournir le maximum de sécurité possible.

Ainsi, nous nous sommes accordés ces objectifs :

Premier objectif, commander le portail à l'aide des boutons poussoirs et de capteurs de fin de courses pour assurer le bon fonctionnement du système.

Second objectif, rajouter d'autres moyens de contrôle du portail, par exemple par une télécommande Infrarouge ou par un digicode.

Troisième objectif, de rajouter diverses mesures de sécurité pour perfectionner le fonctionnement du portail.

ENSEA
Ecole Nationale Supérieure de l'Electronique et de ses Applications

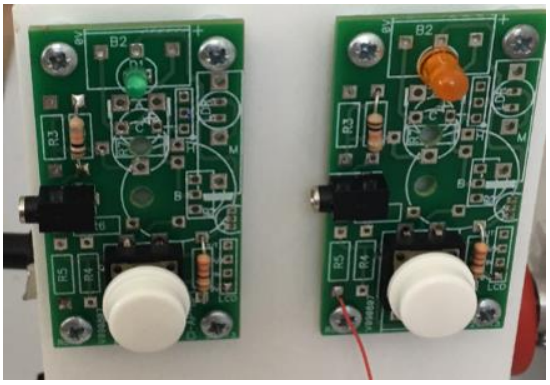
2. Déroulement du projet

2.1. Les différentes étapes

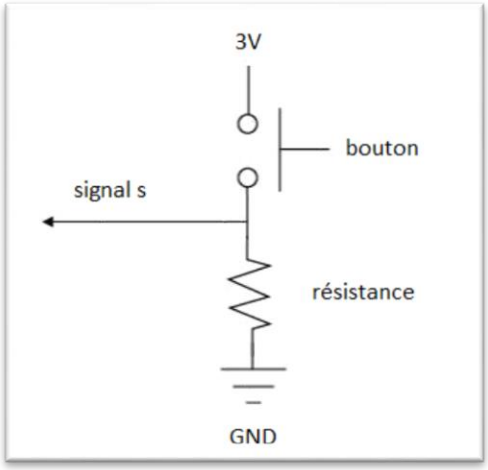
2.1.1. Prise en main du dispositif préexistant

Le dispositif fournit comportait plusieurs PCB, ce qui nécessitait une phase de « décryptage ». Dans cette phase, nous avons tenté de comprendre au mieux les différents circuits qui nous été fournis, ce qui n'était pas une si simple tâche.

- Les boutons poussoirs



Ici, nous pouvons voir les circuits pour les boutons poussoirs :
à gauche au centre, se trouve la prise jack, qui va être la connexion entre le bouton et le boîtier de commande.
En haut, se trouve une LED qui s'affiche lorsque le bouton est appuyé, celui-ci qui se trouve en bas du circuit.



La particularité de ce bouton est qu'il fonctionne avec une résistance pull down.
Le principe est simple : tant que le bouton n'est pas appuyé, la tension que l'on récupère au niveau de la borne s, étant reliée à la résistance « pull down », est nulle. Dès le moment que l'on appuie sur le bouton, une tension proche de 3V peut être captée à la borne de sortie S.
Un petit oubli, nous avons une LED qui se trouve entre la borne de sortie S et la résistance sur le schéma.

ENSEA
Ecole Nationale Supérieure de l'Electronique et de ses Applications

- Les capteurs de fin de courses (nous les appellerons butée)

Le principe de fonctionnement des butées reste identique à celui des boutons. La seule différence étant que les butées stoppent le portail, au contraire des boutons qui le démarre.

- Le moteur

Le moteur fourni est un moteur à courant continu dont la tension nominale est de 1,5 à 3 V. La consommation en courant du moteur à vide est d'environ 220 mA. Cependant nous avons bien pris conscience que le moteur doit en réalité entrainer le portail. Celui-ci impose une charge au moteur ce qui a pour effet d'augmenter le courant consommé à environ 600 mA, en diminuant son impédance.
De fait, plus la charge résistive sur le moteur est importante, plus l'impédance diminue et plus le courant augmente ; c'est ce qu'on utilisera pour réaliser le capteur de couple du moteur.

ENSEA
Ecole Nationale Supérieure de l'Electronique et de ses Applications

2.1.2. Créations de modules manquantes

Lors de la prise en main du dispositif, nous avons remarqué l'absence du module de commande ainsi que du module permettant le contrôle du moteur.

2.1.2.1. Driver du moteur

2.1.2.1.1. Nécessité d'un driver pour le moteur

Le moteur consomme donc au minimum 220 mA et en pratique plutôt 600 mA, ce qui est bien trop par rapport au courant maximal que peut fournir une sortie du microcontrôleur (STM32f4) soit 25 mA et seulement jusqu'à 240 mA pour les pins d'alimentations :

Table 12. Current characteristics

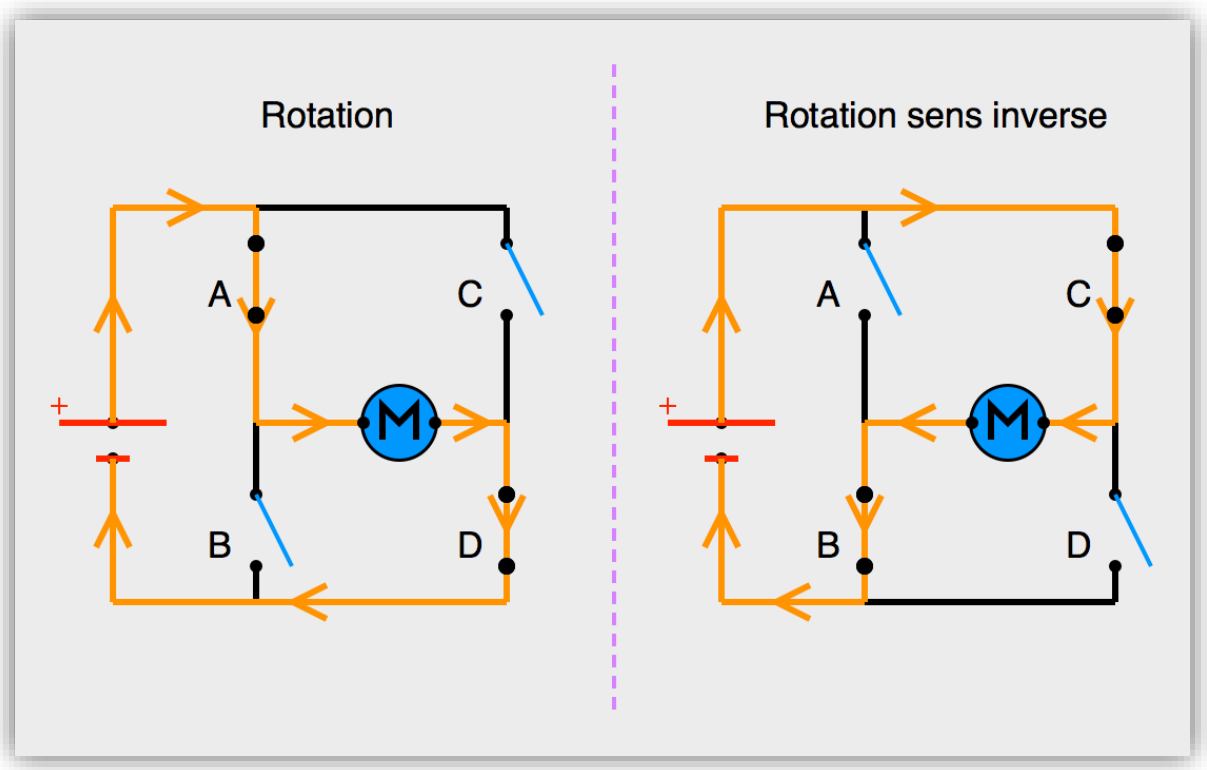
Symbol	Ratings	Max.	Unit
I_{VDD}	Total current into V_{DD} power lines (source)(1)	240	mA
I_{VSS}	Total current out of V_{SS} ground lines (sink)(1)	240	
I_{IO}	Output current sunk by any I/O and control pin	25	
	Output current source by any I/Os and control pin	25	
$I_{INJ(PIN)}$ (2)	Injected current on five-volt tolerant I/O(3)	–5/+0	
	Injected current on any other pin(4)	±5	
$\Sigma I_{INJ(PIN)}$(4)	Total injected current (sum of all I/O and control pins)(5)	±25	

D'où la nécessité d'un circuit de driver du moteur, commandé par une ou plusieurs sorties du microprocesseur et avec une alimentation à part entière.
Un autre détail important à ne pas oublier était la commande du moteur dans les deux sens de manière à pouvoir ouvrir et fermer le portail.
Nous avons d'abord cherché un circuit 'tout fait' pour répondre à ce besoin et nous avons trouvé de nombreuses références de circuits intégrés (exemple : NXP® MPC17510) ou de cartes électroniques (exemple : 1.5A Mini Dual Channel DC Motor Driver Module PWM Speed control beyond L298N X5). Nous nous sommes cependant penché sur une possible réalisation par nous même d'un tel circuit pour mieux comprendre comment il fonctionnerait et essayer de dépenser le moins possible pour le projet.

ENSEA
Ecole Nationale Supérieure de l'Electronique et de ses Applications

2.1.2.1.2. Concept du pont en H

Nous avons découvert que la façon la plus simple et la plus efficace de réaliser un tel circuit était d'utiliser le principe de fonctionnement du pont en H :

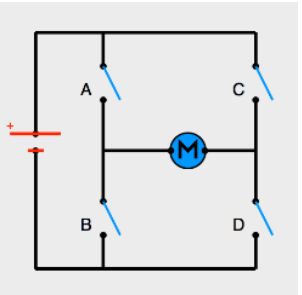


Le pont en H utilise l'association de quatre interrupteurs commandés disposés en forme de H d'où le nom. Ces interrupteurs sont commandés par pairs (A/D et B/C) et ces deux pairs d'interrupteurs ne doivent en aucun cas être fermés en même temps sinon on aura un court-circuit sur l'alimentation.

-Si A et D sont fermés alors B et C doivent être ouverts, le moteur tourne dans un sens entrainant la barrière dans un sens.

-Si B et C sont fermés alors A et D doivent être ouverts, le moteur tourne dans l'autre sens entrainant la barrière dans l'autre sens.

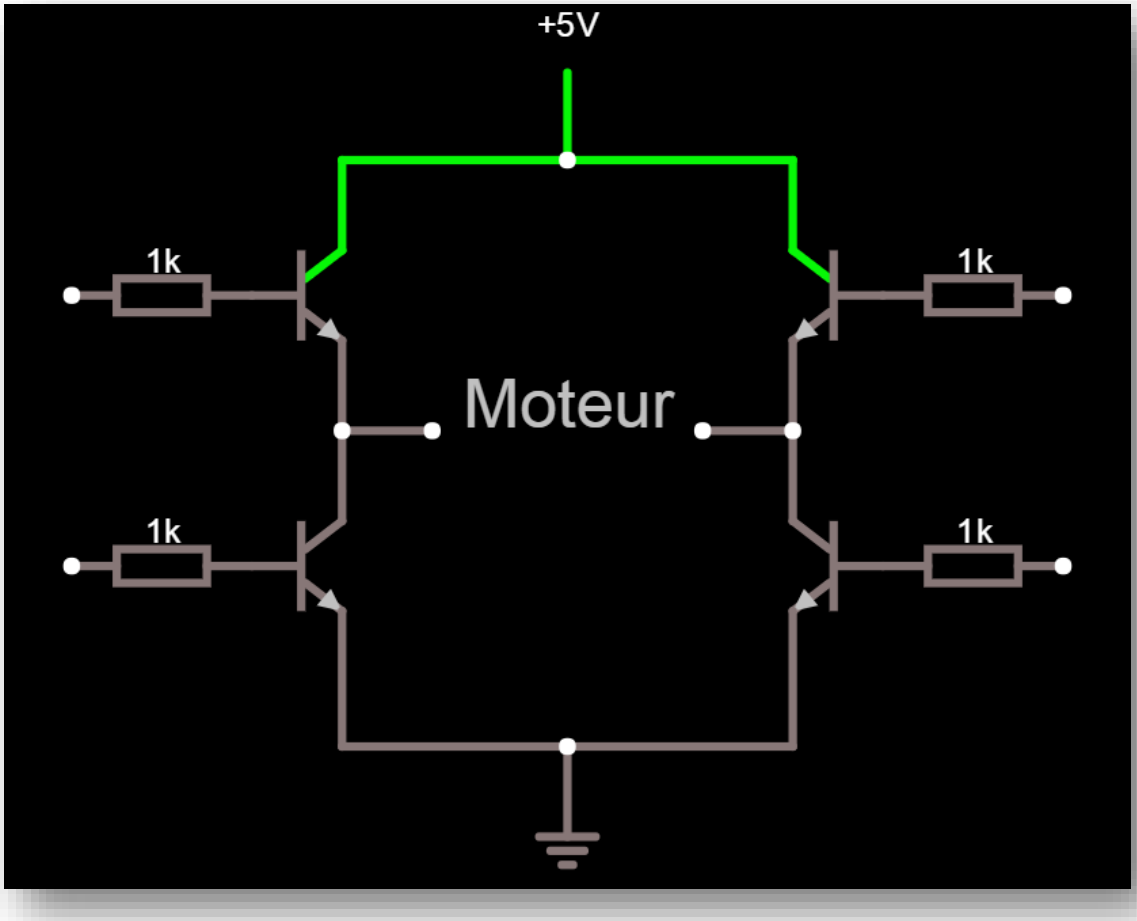
-Si A et D, et B et C sont ouverts, le moteur n'est plus alimenté il ne tourne plus et la barrière est arrêtée.



ENSEA
Ecole Nationale Supérieure de l'Electronique et de ses Applications

2.1.2.1.3. Conception du circuit électronique correspondant

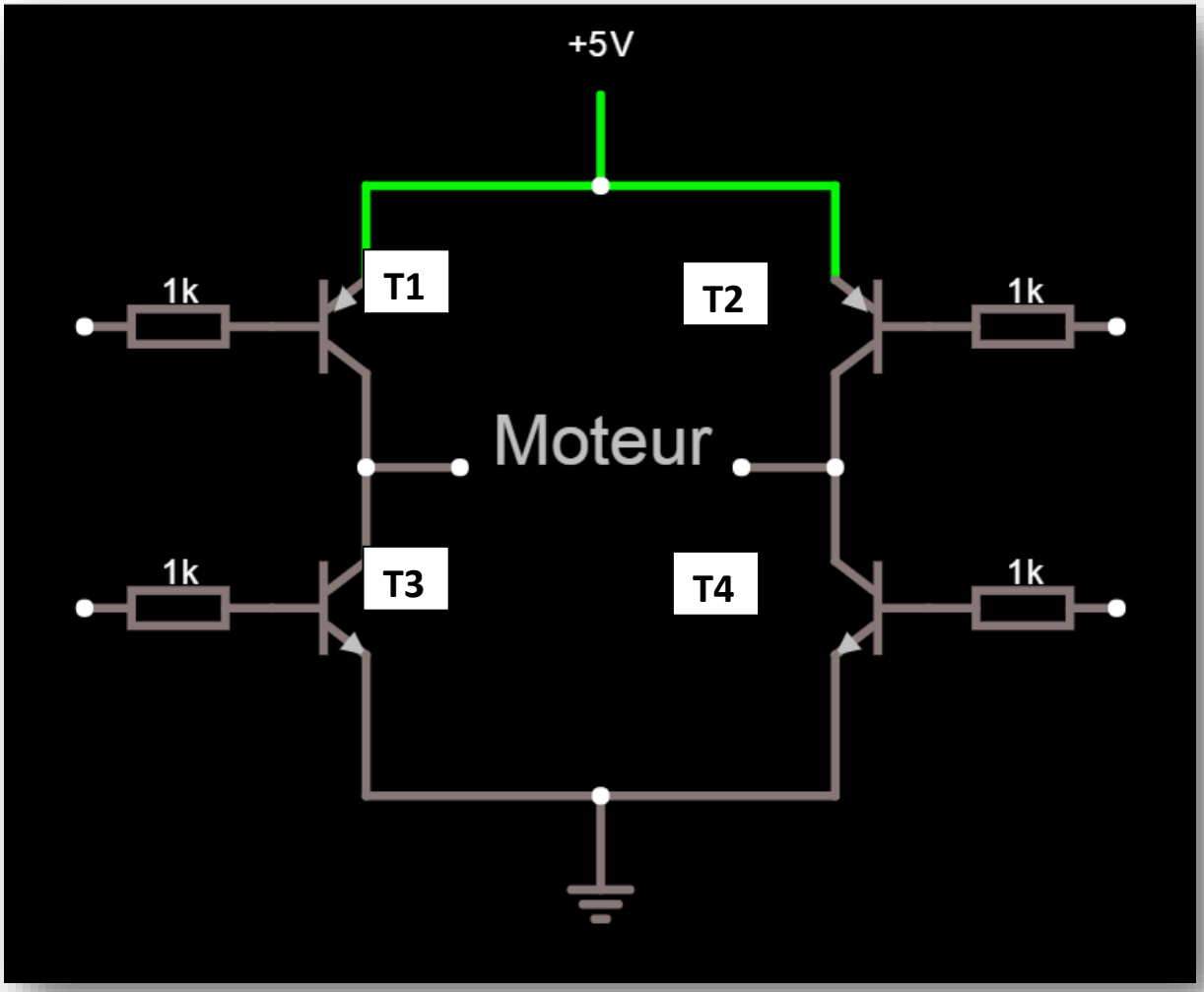
On peut trouver de nombreuses variantes pour la réalisation d'un pont en H sur un circuit électronique, nous avons donc décidé de partir du modèle le plus simple pour le tester et étudier les améliorations à apporter.
Nous avons d'abord opté pour des transistors bipolaires afin de s'en servir comme interrupteurs commandée par les sorties du microprocesseur :



Nous avons d'abord voulu utiliser quatre transistors NPN mais nous avons réalisé que la polarisation à leur imposer pour qu'ils soient suffisamment passant était difficile à définir. En effet c'est le courant de base qui détermine les courants émetteurs et collecteur et donc le courant traversant le moteur ; or le courant de base dépend de la tension base-émetteur (v_{be}) imposée et si la tension base-masse (v_b) est entièrement défini par ce qu'impose la sortie connectée du microprocesseur, la tension émetteur-masse (v_e) des deux transistor du haut (dont le collecteur est relié à l'alimentation V^+=5 V) est inconnue, elle change même en fonction de la polarisation des autres transistors.

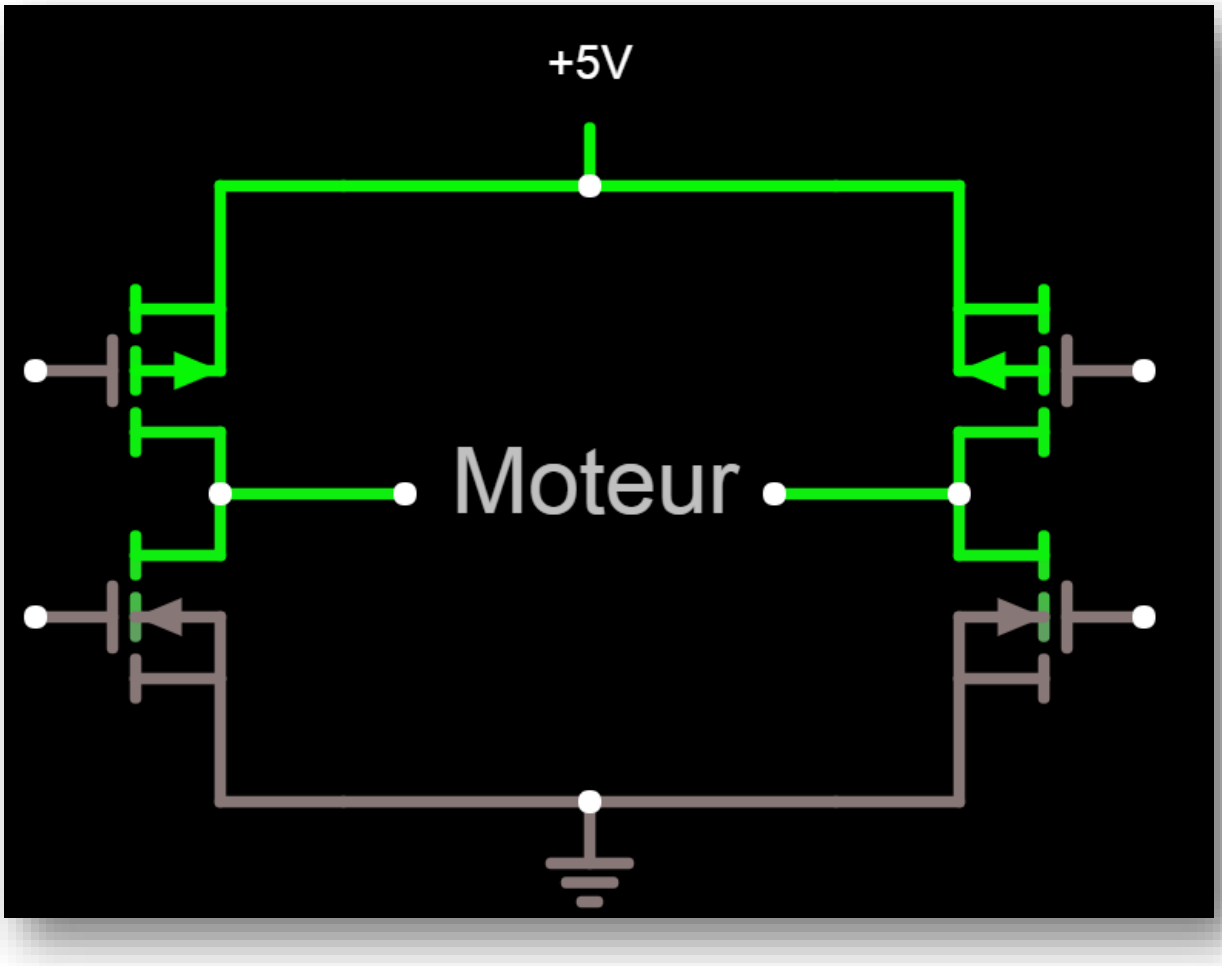
ENSEA
Ecole Nationale Supérieure de l'Electronique et de ses Applications

On peut remédier à cette difficulté en utilisant de transistors PNP aux lieu de NPN pour les transistor qui posait problème :

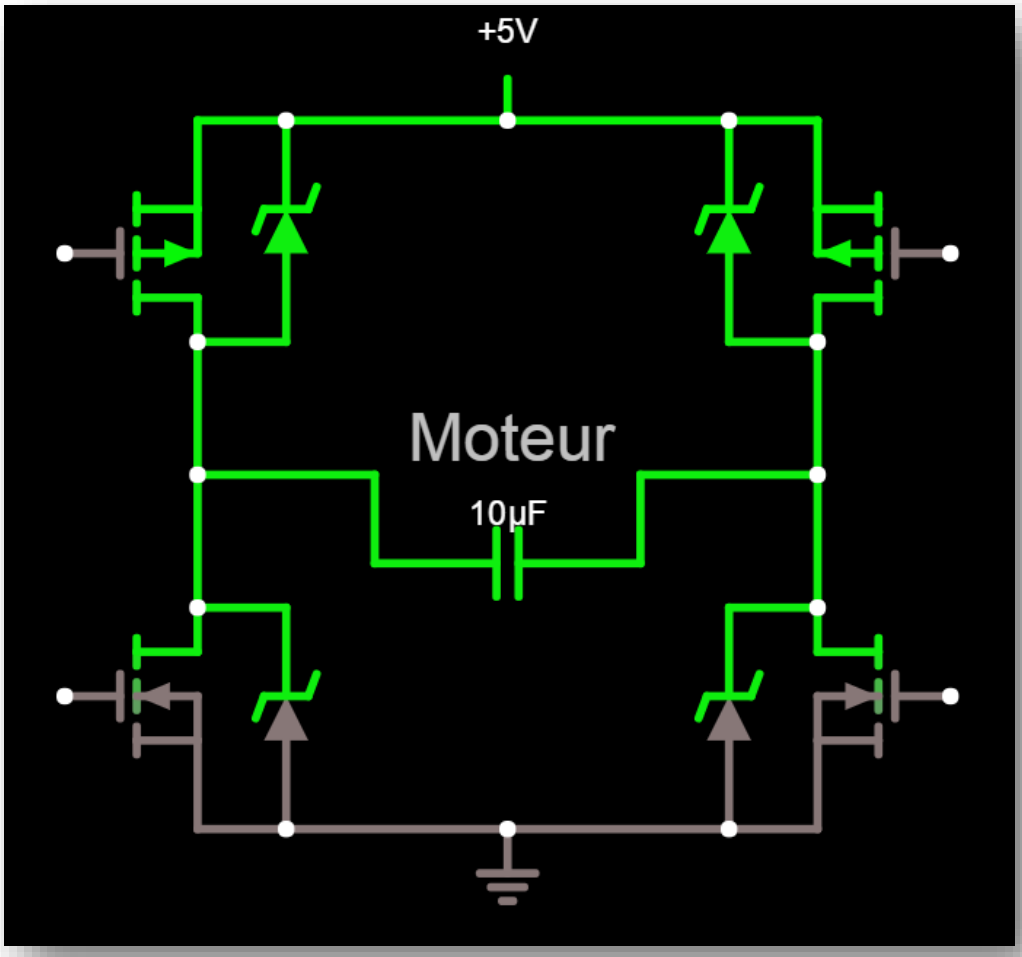


Ainsi les tensions à imposer pour polariser les transistors peuvent provenir directement des sorties du microprocesseur avec pour T1 et T2 : $v_{eb1} = 5 - v_{em1}$ et $v_{eb2} = 5 - v_{em2}$.
Cependant les transistors PNP supportent mal les forts courants.

Nous avons finalement remplacé les transistors bipolaires par des transistors MOSFET de puissance qui supportent un courant beaucoup plus fort et que l'on peut utiliser en régime de commutation.



Ces transistors MOSFET n'aime pas beaucoup les tensions inverses imposées entre le drain et la source or le moteur étant inductif et dans le but de faire un circuit robuste il est nécessaire d'avoir des diodes de roue libre qui limitent ces tensions (même si avec ce petit moteur les diodes ne devraient pas avoir beaucoup d'importance et le circuit driver est surdimensionné). Ces diodes de roue libre sont intégrées au transistor MOSFET choisis. Il y a également un condensateur en parallèle avec le moteur pour amortir les gros pics de tensions.



ENSEA
Ecole Nationale
Supérieure
de l'Electronique
et de ses Applications

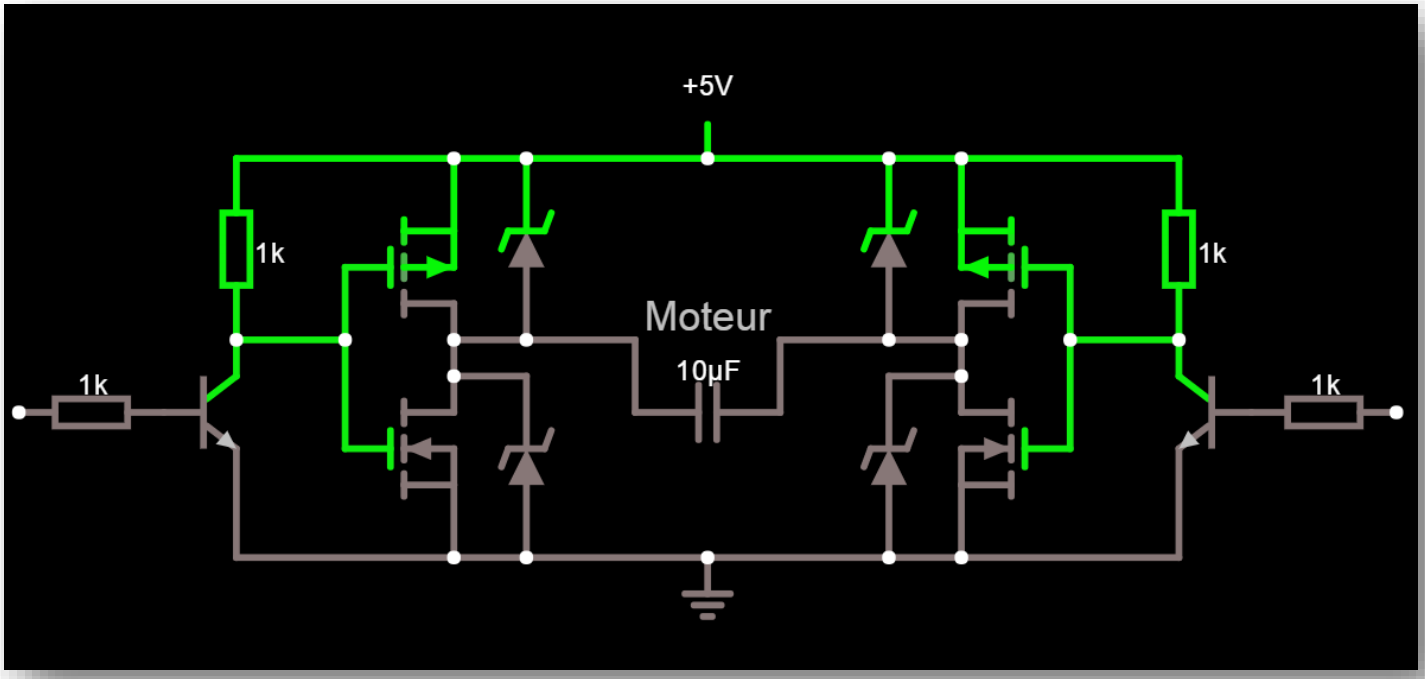
Les transistors MOSFET sont plus faciles à commander car ils se polarisent par une tension entre grille et source, le courant de grille étant nul (contrairement aux transistors bipolaires qui se polarisent par un courant) or il est plus facile d'imposer une tension qu'un courant. Nous avons donc remplacé les transistors NPN par des MOSFETs à canal N et les transistors PNP par des MOSFETs à canal P.
Néanmoins nous ne pouvions toujours pas commander les MOSFETs directement par le microprocesseur, en effet celui-ci renvoie pour une sortie numérique : 0 V pour un niveau bas et 3 V pour un niveau haut, or les tensions v_{GS} à imposer pour un régime commuté satisfaisant sont :

-pour les deux MOSFETs à canal N, 0 V (transistors bloqués) et 4 V minimum (transistors passant), sachant que la source de ces transistors est reliée à la masse (en pratique on utilisera 0 V/5 V, tension d'alimentation).

-pour les deux MOSFETs à canal P, les même tensions opposées pour les mêmes régimes de fonctionnement, mais sachant que la source est reliée à l'alimentation positive à environ +5 V, les tensions correspondantes à imposer entre la grille et la masse sont 5 V et 1 V (le microprocesseur ne peut qu'imposer des tensions de sortie par rapport à la masse et de même on utilisera 5 V/0 V).

On a donc dû créer un buffer de tension pour élever le niveau logique haut de la carte de 3 V à 5V. On réalise cela en pratique à l'aide de résistances pull-up et de transistors bipolaires qui, en mode passant ramène les tensions de grilles à une tension proche de 0 V.

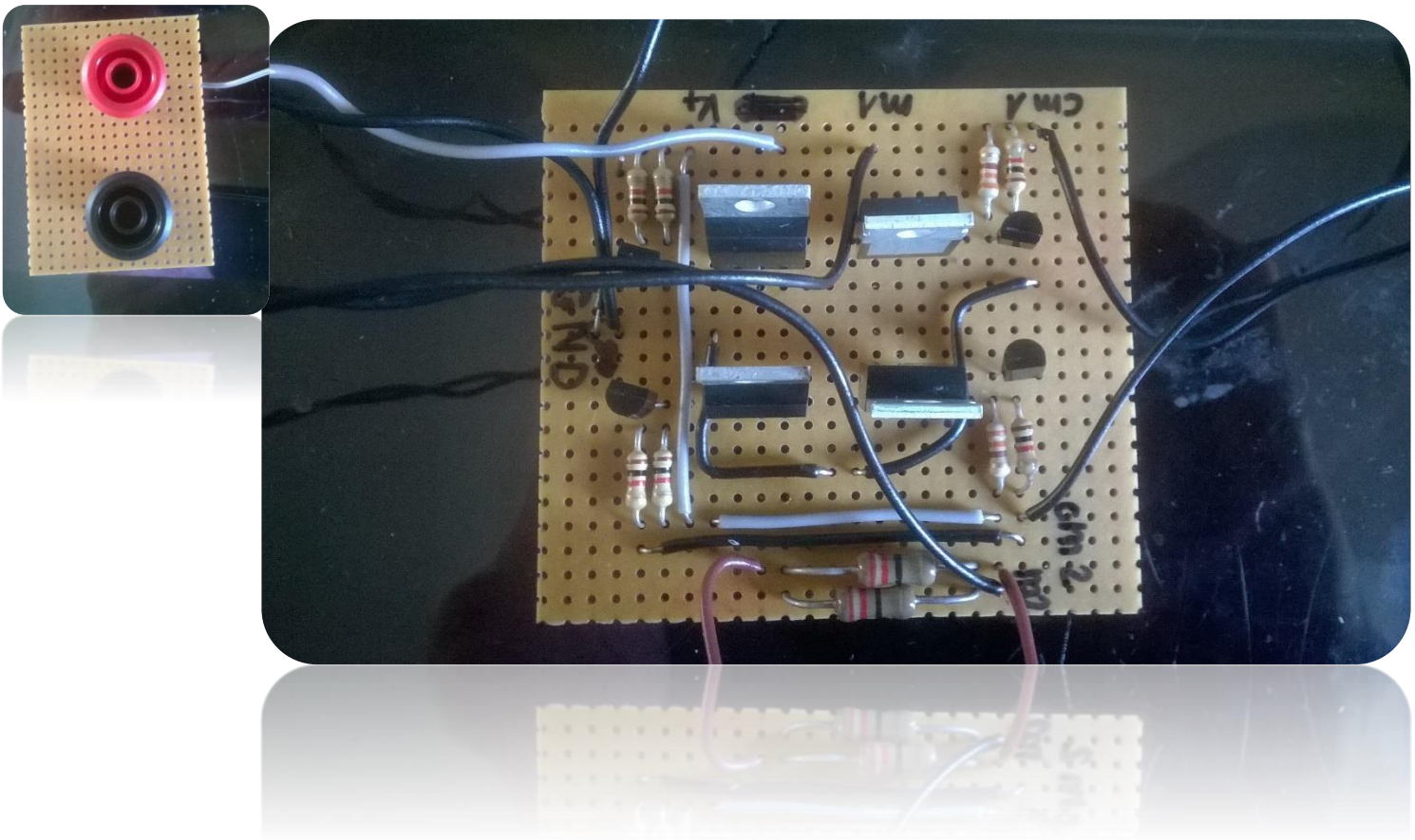
ENSEA
Ecole Nationale
Supérieure
de l'Electronique
et de ses Applications



ENSEA
Ecole Nationale
Supérieure
de l'Electronique
et de ses Applications

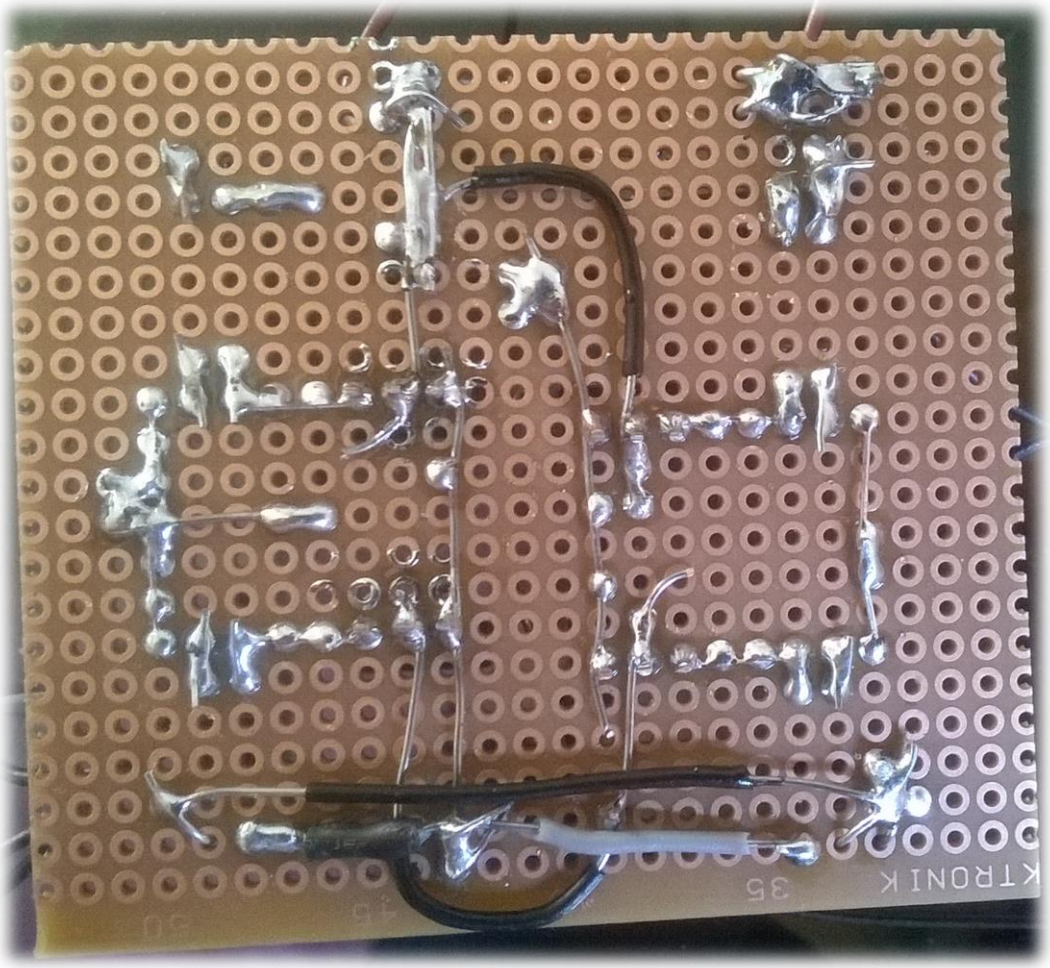
2.1.2.1.4. Réalisation du circuit

Nous avons manqué de temps pour réaliser un pcb pour ce circuit donc nous avons dû le souder sur une carte de protoypage :



ENSEA
Ecole Nationale
Supérieure
de l'Electronique
et de ses Applications

La difficulté tenait de ce qu'aucune piste n'étant imprimée il a fallu les remplacer par du fil dénudé optimiser la disposition pour avoir le moins de croisement possible et ne pas se tromper dans le remontage et les connexions du circuit sur la plaque de prototype.



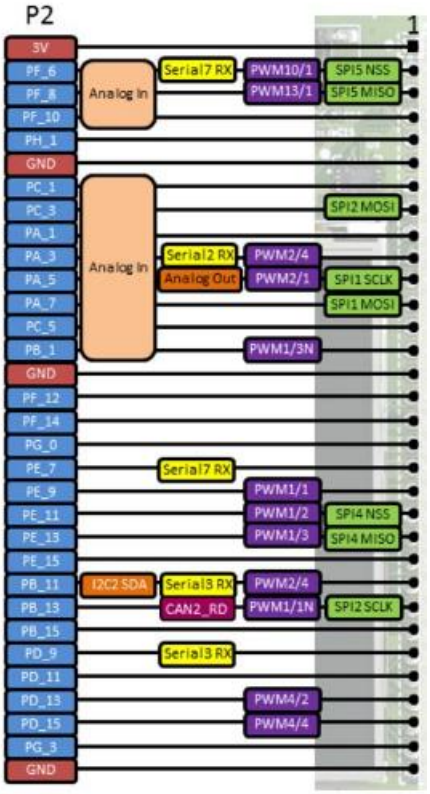
ENSEA
Ecole Nationale
Supérieure
de l'Electronique
et de ses Applications

Ensuite, pour la commande du portail, nous avons choisi d'utiliser un microcontrôleur, la stm32f429ZI. Ce choix a été difficile, notamment pour la programmation, car elle nous a imposé le langage de programmation, entre le langage C et le langage VHDL.
En effet, les deux avaient, selon nous, leurs avantages et leurs inconvénients :
Le langage C, permet une programmation facile pour chaque comportement, notamment par sa structure ordonnée comportant une fonction main et ces fonctions auxiliaires.
Au contraire, le langage VHDL, permet une étude temporelle, facilitant la programmation des comportements incorporant une notion du temps.
Finalement, le critère définitif qui nous a fait basculer au C, donc au microcontrôleur, était la connaissance plus certifiée en langage C.

Enfin, il nous été nécessaire de créer un PCB, pour pouvoir contrôler toutes nos fonctionnalités.

Le but est de faire un circuit permettant le lien entre les différents éléments et circuits que l'on utilise sur le portail, et la STM 32, car c'est la STM qui permet le contrôle de ces éléments. Pour cela nous utilisons Eagle, qui permet de tracer un circuit imprimé. Le circuit imprimé fait le lien entre les pins de la STM et des pins header qui sont connecté à chaque circuit composant le portail. La STM a 4x32 pins. Nous voulions utiliser 2 colonnes de broches de la STM afin que les liaisons entre la STM et les pins header soient assez éloignés. Cependant, les pin header pour les 2 colonnes de la stm32 auraient été trop proches sur le circuit imprimé avec les paramètres et configurations imposés pour que l'impression soit réalisable par le technicien. Nous avons donc choisi de prendre la colonne gauche P2 de gauche car il y a plusieurs broches correspondant à la masse, ce qui minimise le croisement des liaisons et donc le nombre de soudure sur la deuxième couche et aussi car il y a des pins analogiques nécessaire au portail.

ENSEA
Ecole Nationale
Supérieure
de l'Electronique
et de ses Applications

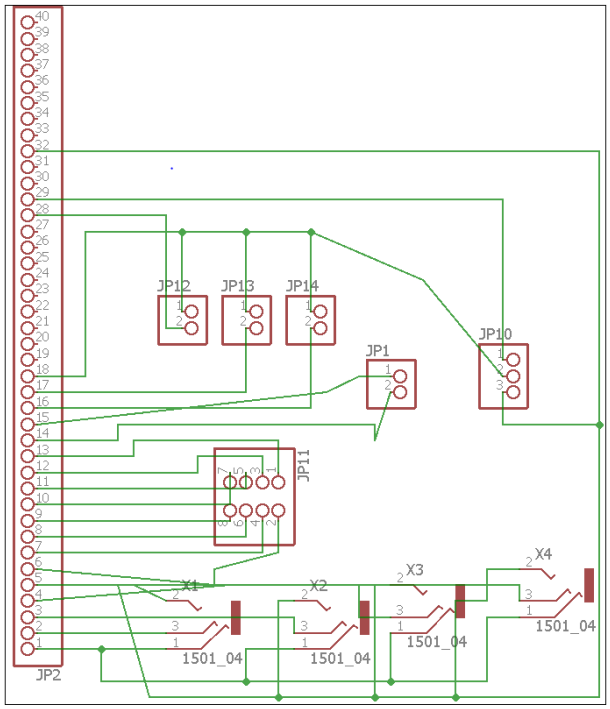


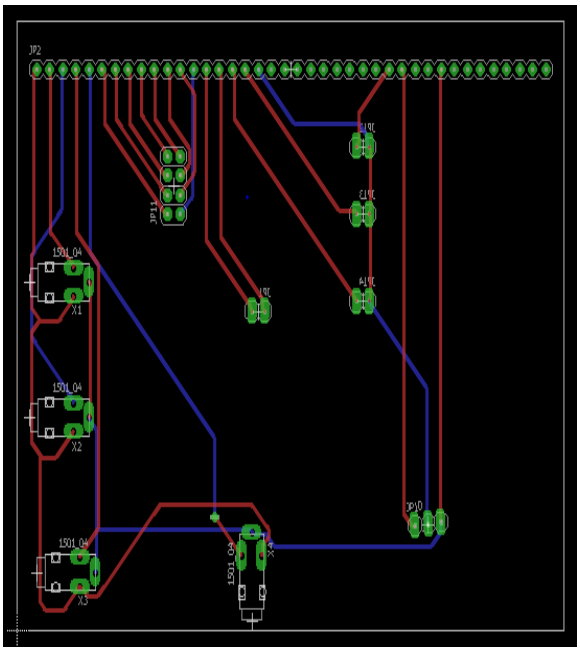
Les pin header :

Pour chaque leds nous avons choisi des pins header de taille 2x1 pour les relier à la masse et à une sortie de signal du microcontrôleur.
Pour le digicode nous avons pris un pin header de taille 4x2 pour relier les 8 pins du digicode à des pin d'entrée de la stm32. La configuration 4x2 permet de différencier plus facilement les pins correspondant aux colonnes et aux lignes du digicode afin que le PCB soit plus clair et aussi pour faciliter la soudure.

Pour les boutons, le récepteur infrarouge et les capteurs de butée, nous avions d'abord choisit des pins header de taille 3x1 car il faut un pin pour relier l'élément à la masse, un pin pour le relier à une entrée de la STM et un pour la tension d'entrée. Mais nous avons ensuite opté pour des prises jack car certains éléments du portail était déjà doté de prise jack, mais aussi car utiliser ce genre de fil permet de désencombrer le montage global.

Nous avons tenté plusieurs combinaisons de liaisons et de placement des pin header afin de minimiser les couches, mais nous avons au final un PCB à 2 couches. Il a fallu changer plusieurs fois le circuit et refaire le routage car il fallait tenir compte des paramètres de réalisation afin que l'impression soit possible pour le technicien.





ENSEA
Ecole Nationale Supérieure de l'Electronique et de ses Applications

2.1.3. La programmation en C

Le choix était fait, il était ensuite le temps de programmer. Le portail étant un système asservi, il était nécessaire de trouver un moyen de pouvoir laisser le programme tourner de manière continue. Nous avons opté pour la programmation de type machine à états.
Le principe est simple, le programme effectue une boucle de manière permanente, en parcourant différents états avec chacun des conditions d'entrées différentes. Ainsi, le système peut évoluer de manière spontanée en fonction des instructions que nous imposons.
En C, il existe des fonctions permettant de créer une machine à états, tels que while(1), permettant de faire une boucle infinie, ou les fonctions switch(etat) case(), permettant la création des états différents.
Ainsi, en combinant les 3 fonctions, nous avons pu créer une machine à états.

Contre toute attente, mise à part la création de la machine à états, la programmation en C nous a pas posé de problèmes majeurs.
Cependant, nous avons tout de même rencontré de nombreux problèmes, notamment au niveau du passage code en C => microcontrôleur.
En effet, les pins et leurs utilisations en C n'était pas une tâche facile. C'est d'ailleurs pour cela que dans ce projet, nous avons utilisé deux logiciels de programmation :
Au début, nous avons utilisé mbed, puis ensuite keilvision.

ARM mbed enabled

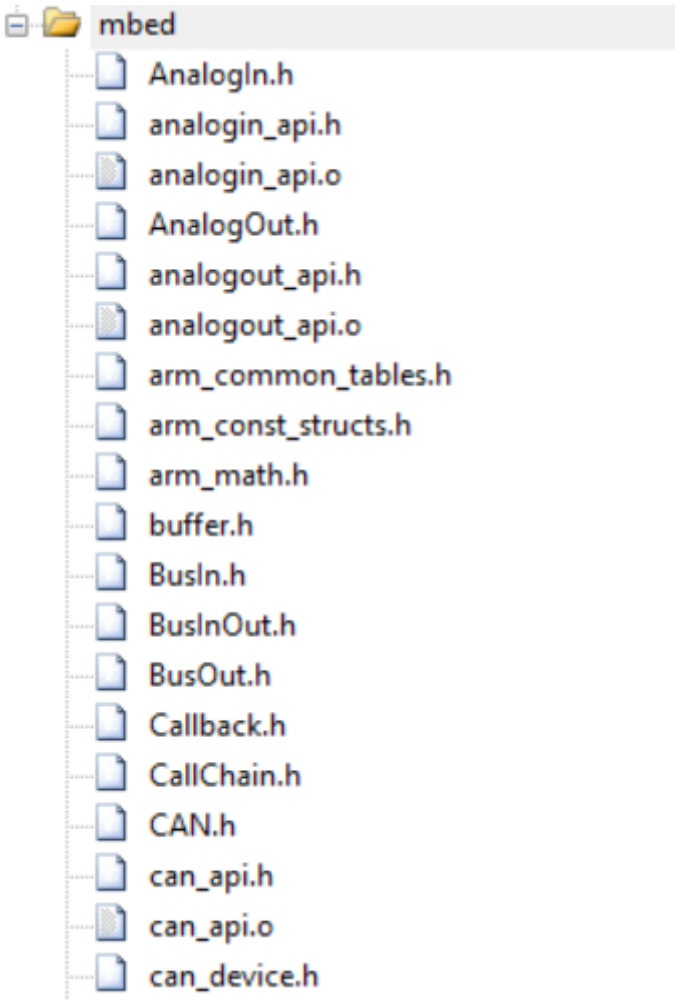
KEIL™ Tools by ARM

ENSEA
Ecole Nationale
Supérieure
de l'Electronique
et de ses Applications

Pourquoi avoir utilisé deux programmes différents ?
Tout simplement pour une meilleure gestion du temps. Au commencement du projet, nous ne connaissions pas assez le langage dédié à keilvision pour les configurations des ports, ce qui nous bloquait à la progression du codage. Ainsi, nous avons préféré utiliser mbed, qui proposait des fichiers d'initialisations des ports prédéfinis.

Voici une partie du fichier :

ENSEA
Ecole Nationale Supérieure de l'Electronique et de ses Applications



Ces fichiers nous a permis de gagner considérablement du temps pour pouvoir continuer notre programmation.
Cependant, l'inconvénient majeur de mbed est qu'il ne possède pas de débugger. Or, à un certain stade de la programmation, il nous été nécessaire de trouver un moyen de simuler notre programme, car le programme ne contenait aucune erreur sans qu'il ne fonctionne.

Ainsi, nous sommes passé à Keilvision, qui lui possédait un mode debug.

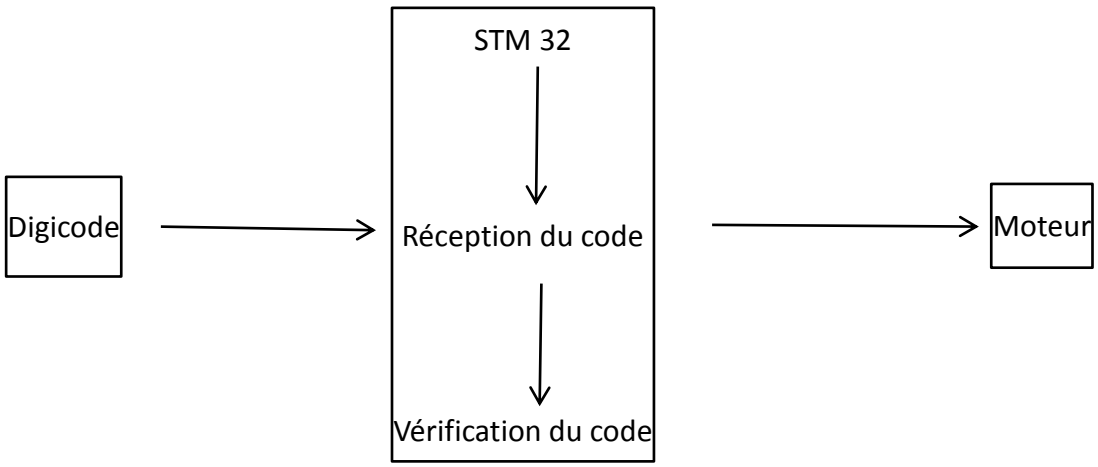
Lors du transfert, nous avons décidé de garder les fichiers d'initialisations de mbed, pour pouvoir garder les mêmes configurations pour les ports.

ENSEA
Ecole Nationale Supérieure de l'Electronique et de ses Applications

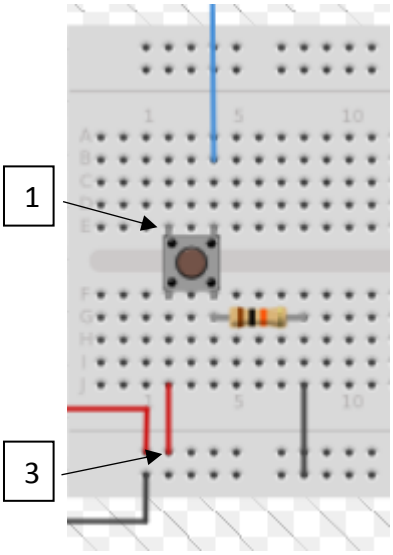
2.1.4. Rajout de fonctionnalités

Après avoir programmé le fonctionnement basique du système et l'avoir testé, nous avons commencé à chercher des moyens pour améliorer le fonctionnement du portail.
La première chose qui nous est passé par la tête était le digicode. De nos jours, le fonctionnement le plus courant d'un portail automatique est la commande par un digicode, permettant l'ouverture du portail avec identification de l'utilisateur, limitant les utilisations non voulues du portail par des tiers.

Le principe du digicode est qu'un utilisateur entre un code personnel ce qui active l'ouverture de la porte. Notre code doit donc pouvoir vérifier quel numéro est appuyé, conserver le code entré par l'utilisateur, vérifier si ce code est correct et envoyer une commande au moteur pour ouvrir le portail.



Etant simplement une idée, nous nous sommes d'abord imaginer le digicode le plus simple, avec dix boutons poussoirs qui ont chacun leur chiffre de 0 à 9 pour pouvoir mettre en place les fonctions nécessaires pour un digicode. Les boutons poussoirs sont tous reliés aux pins de tension de la stm32 (3V) ainsi qu'à une résistance et enfin à la masse. Les signaux des boutons poussoirs sont ensuite récupérés grâce à des fils connectés à des pin DigitalIn du microcontrôleur. Le principe est que lorsqu'on appuie sur un des boutons, le courant peut passer dans le circuit et on récupère le signal sur la stm32. L'utilisation d'une résistance pull down permet d'assurer que lorsque le bouton n'est pas pressé, la tension de sortie est nulle.

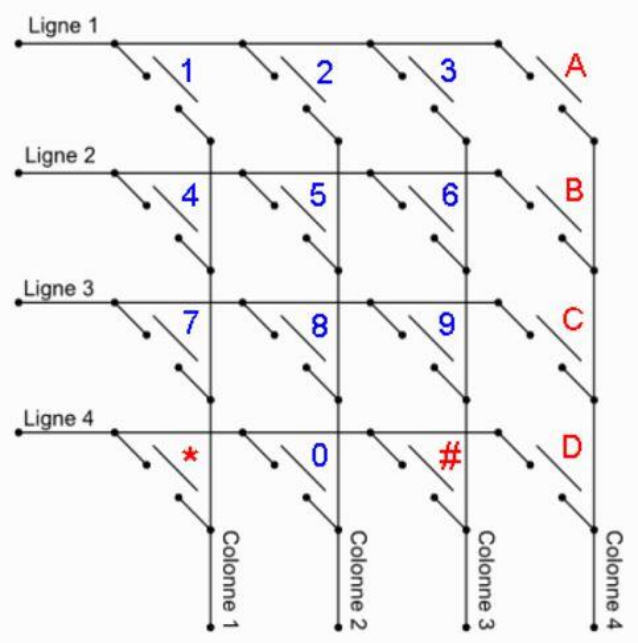


Fil bleu : le signal récupéré par la stm32.
Fil rouge : le signal entrant, ici de 3V.
Fil noir : la terre.
La résistance : résistance de type pull down.

Bouton poussoir à 4 broches : les broches du haut est relié à celui d'en bas. Donc ici, la 1ère broche est reliée à la 3ème broche, et la 2ème à la 4ème. Lorsque le bouton est appuyé, une connexion se forme entre la broche 2 et la broche 3, permettant ainsi au courant de passer.
Le principe reste donc le même que le bouton poussoir du portail, un fonctionnement en pull down.

La partie programmation de ce digicode ne nous a pas posé de gros problèmes, à part le fait que le type DigitalIn de mbed était une structure et non pas un int, ainsi nous empêchant de pouvoir récupérer directement la valeur du bouton poussoir. (0 s'il n'est pas appuyé, et 1 s'il est appuyé) Le problème a vite été résolu en passant en mode Debug sur KeilVision, où l'on a trouvé le vrai type de DigitalIn. Ainsi, pour récupérer la valeur du bouton au sein de cette structure, il nous a suffi d'utiliser la commande .read().

Après avoir vérifié le bon fonctionnement de notre digicode simplifié, nous sommes passé au vrai digicode que nous allions utiliser pour notre portail.
Il a fallu changer certaines fonctions de notre programme précédent afin de l'adapter au fonctionnement du digicode qui est totalement différent. Le digicode est composé de 16 boutons donc de 4 lignes et 4 colonnes.



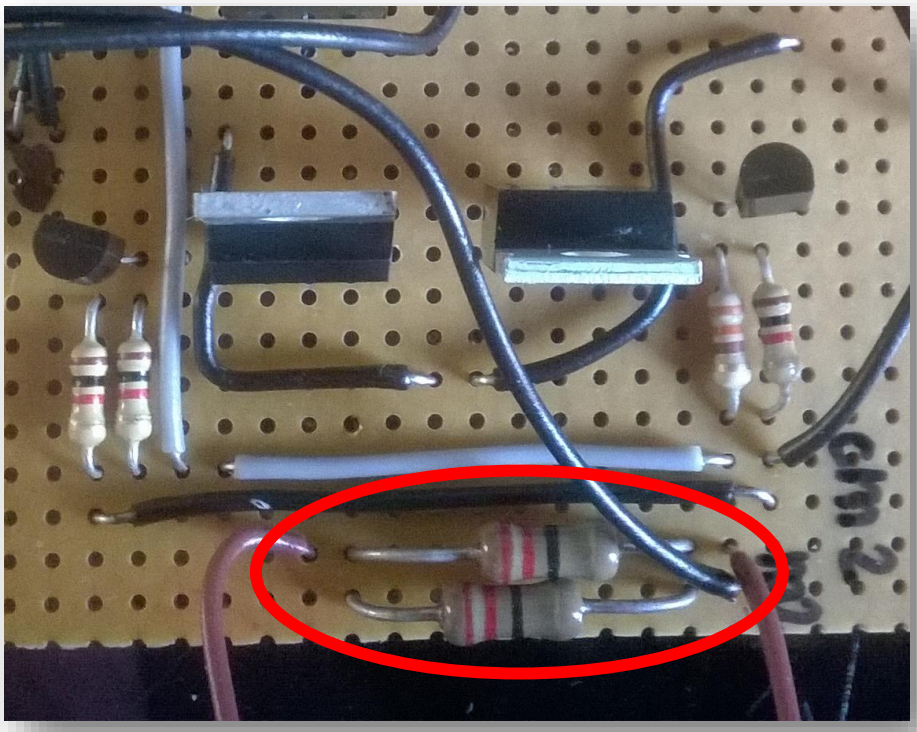
L'état d'un bouton se récupère par l'intersection de la ligne et de la colonne sur lesquelles il se trouve.
Les informations des lignes et des colonnes se récupèrent sur les 8 pins se trouvant au dos du digicode. Pour comprendre quel était l'ordre des pins, il a fallu tester les pins grâce à un multimètre en mode diode : lorsqu'on appuie sur un bouton et qu'il est relié à la bonne ligne et la bonne colonne, on reçoit bien un signal sur le multimètre. Ainsi, les lignes et les colonnes sont bien dans l'ordre du schéma.

Pour ce qui est de sa programmation, nous avons dû prendre en compte le fait que le digicode en lui-même ne pouvait pas suffire pour effectuer une connexion. En effet, il n'y a pas de moyen pour mettre une source de tension commune, car cela créerait des courts circuits. Il nous a donc fallu définir les lignes comme entrées pour le digicode, et les activer une par une. Seulement après, nous pouvions vérifier si une colonne est activée : si un bouton est appuyé, cela crée une connexion entre la ligne et la colonne correspondante, et nous pouvons observer une tension à la colonne associée.
Cette mise en entrée des lignes nous a obligé d'écrire un programme qui tournerait de manière permanente, qui nous a poussé à faire une boucle infinie avec un while(1).

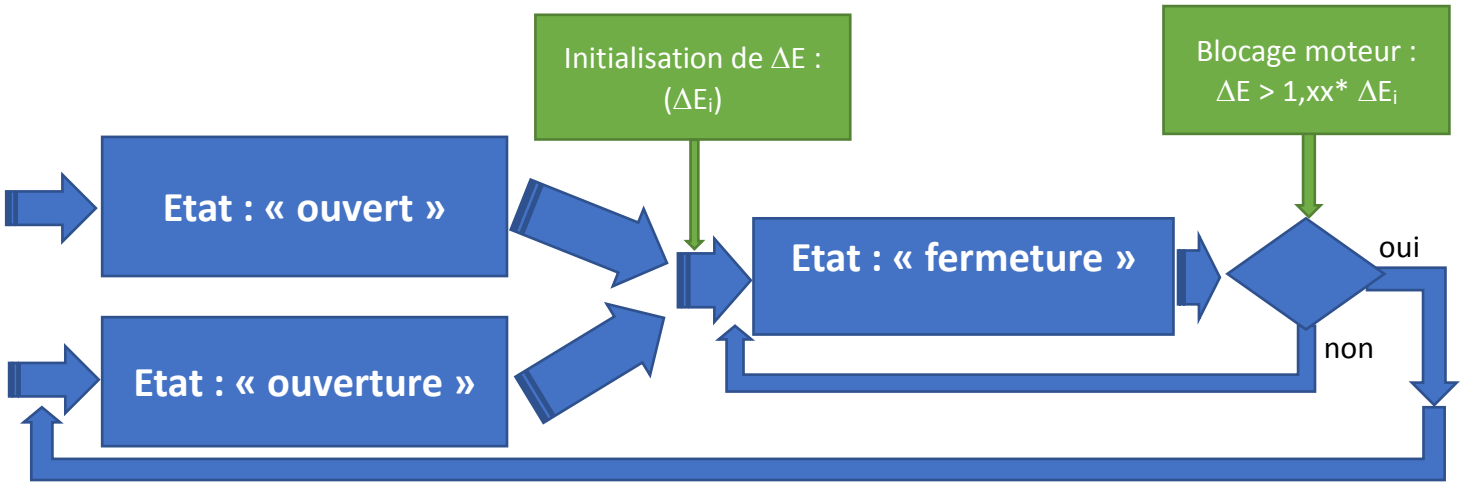
D'autre part, pour le numéro du bouton appuyé, comme on peut le remarquer sur le schéma, lorsque l'on incrémente la colonne de 1, la valeur augmente de 1. Et si l'on incrémente la ligne de 1, la valeur augmente avec un pas de 3.

2.1.4.2 Le capteur de courant du moteur

Dans le but d'améliorer la sécurité dans l'utilisation du système nous avons voulu ajouter un capteur de couple du moteur. Nous utilisons le fait que le courant du moteur augmente quand on lui applique une résistance.
Là encore, on trouve beaucoup de capteurs de courant différents : des transimpédances à base d'ampli op, des capteurs à effet hall (plutôt pour les gros courants), des sondes de courant sous forme de CI, nous avons opté pour la simplicité en utilisant une résistance shunt en série avec le moteur. On peut relever la tension à ses bornes qui est proportionnelle au courant qui la traverse et qui est le même que le courant qui traverse le moteur. Le problème est que cette résistance crée une chute de tension au niveau du moteur et amène des pertes d'énergie par effet joule. Nous l'avons donc prise la plus petite possible en pratique on a utilisé deux résistance de 22 ohms en parallèle, et qui puisse supporter la puissance qu'elles



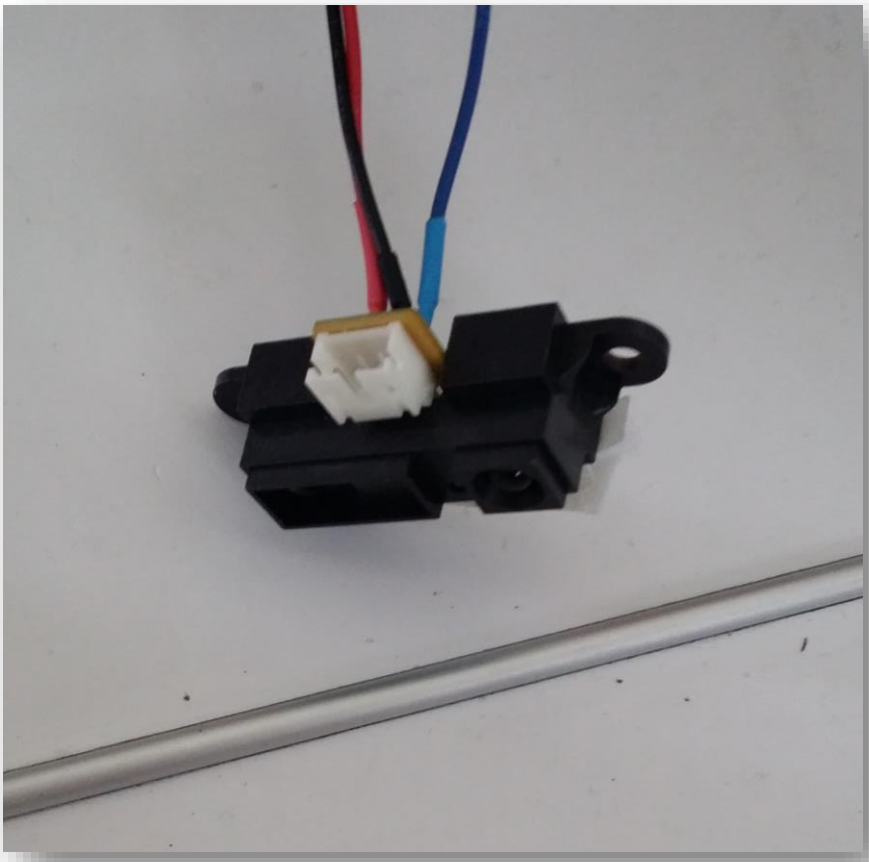
dissipent sous forme de chaleur soit, ici, $P_j = R*i^2 = 11*0,6^2 = 4$ W en tout donc 2 W par résistance ce qui n'est pas à négliger.
L'idée était donc de couper le moteur quand la tension au bornes de la résistance de mesure et donc le courant moteur dépassait une certaine valeur car cela signifierait que le moteur subit une résistance plus importante et donc que quelque chose coince au niveau de la barrière. Ce système de protection est surtout important lors de la fermeture du portail.
On branche donc chaque côté de la résistance de mesure à des entrées analogique du microprocesseur, ainsi, dans le programme en machine à état du microprocesseur, on ajoute une nouvelle sortie conditionnelle de l'état « portail en fermeture » à l'état « portail en ouverture » ; la condition étant : 'lorsque la différence de tension entre les deux entrées analogique (ΔE) dépasse une certaine valeur strictement supérieure à la valeur de tension différentielle initialisée au début de l'ouverture (ΔE_i), alors l'état devient : « fermeture du portail »'.



ENSEA
Ecole Nationale
Supérieure
de l'Electronique
et de ses Applications

Le capteur de passage à infrarouge :

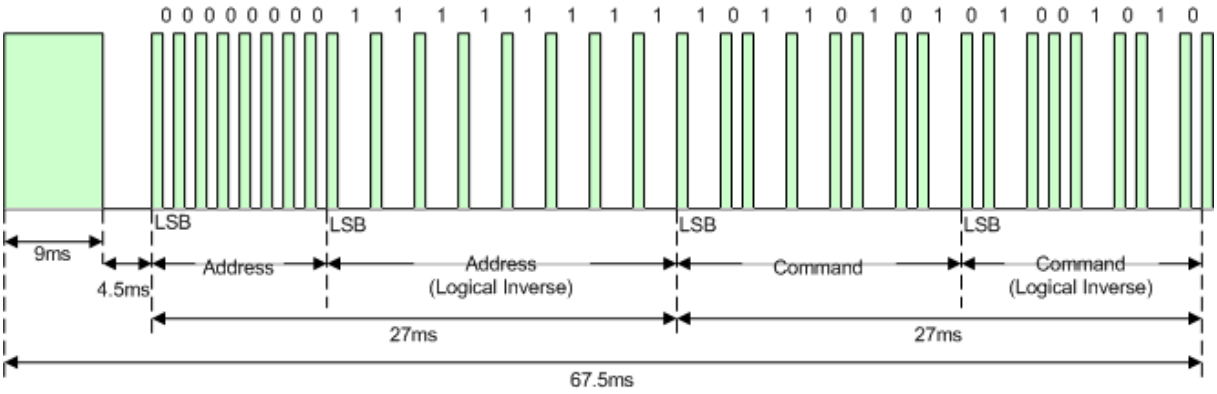
Un autre capteur de sécurité que nous voulions ajouter et qui est complémentaire au capteur de courant moteur est le capteur de passage à infrarouge qui détecte quand un objet ou une personne est en train de passer à travers l'entrée et qui est censé éviter que la barrière ne se ferme sur cet objet ou cette personne.
Nous avons sous-estimé ce capteur que nous ne pensions être qu'un simple rayon d'une LED infrarouge associé à un capteur photoélectrique que couperait un objet dans le passage, mais le fait est que pour éviter toute perturbation du milieux extérieur et en particulier des conditions d'ensoleillement, il faut que le rayon infrarouge soit modulé à une fréquence bien définie pour que le capteur puisse bien vérifié que la lumière infrarouge qu'il reçoit provient bien de l'émetteur associé.
En pratique nous avons donc finalement utilisé un capteur de distance à infrarouge qui renvoie une tension d'autant plus grande que la réflexions de la lumière infrarouge émise est grande soit que la distance de l'objet à partir duquel on fait la mesure est grande. On utilise exactement le même principe que pour le courant moteur pour intégrer les données du capteur dans le programme du microprocesseur.



ENSEA
Ecole Nationale Supérieure de l'Electronique et de ses Applications

La télécommande infrarouge :

Tout portail automatique se doit de pouvoir être ouvert à distance. Nous avons pensé à une télécommande infrarouge qui utilise le protocole NEC pour transmettre les données, avec des impulsions d'une durée précise et modulée comme pour le capteur de passage :



Associé au même genre de capteur que celui que l'on aurait utilisé pour le capteur de passage (et qui fonctionne en tout ou rien) il aurait simplement fallu adapter l'un des nombreux programmes en c que l'on peut trouver en ligne et qui utilisent des bibliothèques pour déchiffrer les données du capteur et l'intégrer au reste du programme du portail.

2.2. La répartition du travail

La gestion du temps était un facteur clé par la réussite de notre projet. Nous nous étions mis d'accord pour effectuer une tâche à la fois pour pouvoir assurer la finition de cette tâche. Au cours du projet, nous avons finalement opté pour effectuer deux tâches à la fois, pour augmenter le rendement. Nous avons donc séparé l'équipe en deux sous-groupes, un groupe qui se charge de la partie numérique-programmation, et l'autre de la partie analogique.

ENSEA
Ecole Nationale
Supérieure
de l'Electronique
et de ses Applications

2.3 les problèmes rencontrés et leurs solutions

- Dans tous les problèmes rencontrés, le problème au niveau de la gestion du temps nous a le plus pénalisé. Pour la première méthode qui consistait à effectuer une seule tâche à la fois, nous n'avons pas pu finir la toute première tâche, qui était la programmation du système global pour un fonctionnement basique, en prenant compte seulement des 2 boutons poussoirs. Cela nous a finalement pris plus de deux séances sans prendre le temps passé en dehors des heures de séances. Ainsi, elle n'était absolument pas efficace.

De même pour la deuxième méthode, même après s'être séparé, nous n'étions pas assez ordonnés et coordonnés pour pouvoir être vraiment efficace. Nous allions à l'efficacité d'une personne et demie lorsque tout se passait bien, mais nous étions plus aux alentours d'une efficacité d'1.2 personne par sous-groupe. La répartition du travail nous semblait, assez évident au début de chaque séance, mais dès que nous commencions à travailler, nous n'avions plus assez d'aisance et de confiance pour pouvoir agir de manière totalement indépendante de son partenaire.
Par exemple, le sous-groupe du numérique est resté sur le digicode pendant plus de 4 séances, pendant lesquelles nous sommes arrivé à avancer à pratiquement rien du tout. Tout ce temps mal géré nous a amené à laisser tomber la partie de la télécommande IR, qui était pourtant très intéressant et assez accessible du point de vue technique.
En effet, la télécommande IR consiste à envoyer une séquence de d'IR à une certaine fréquence, où l'information commande est cachée. Le capteur doté de la bibliothéque des codes, peut lire les instructions qu'envoie la télécommande. Comme nous étions en possession d'une télécommande, il nous suffisait de connaitre la bibliothéque de la télécommande pour pouvoir l'utiliser, la longueur d'onde étant normalisé à ~950nm.
D'autant plus que mbed possédait un code de décryptage de l'onde IR prédéfini. Il nous restait donc vraiment peu de choses pour pouvoir utiliser la télécommande.

ENSEA
Ecole Nationale
Supérieure
de l'Electronique
et de ses Applications

- Ensuite, voici les problèmes rencontrés avec le digicode :

Nous ne connaissons pas la manière de programmer sur la STM32. L'utilisation de mbed nous a permis d'avoir directement des fichiers d'initialisation de la STM32. Cependant, mbed ne possède pas de debug, donc lorsque nous avons eu des problèmes de programmation, nous ne pouvions pas trouver ce qui n'allait pas alors que le debug nous aurait aidé. Nous avons ensuite vu que mbed permettait l'exportation du programme vers keil µvision. Nous avons donc pu changer de logiciel de programmation et avancer dans notre code.

Lors de la programmation en C nous avons eu des problèmes pour utiliser les variables des boutons du digicode. En effet, grâce à mbed, les boutons sont simplement désignés par des variables, mais la valeur de ces variables ne correspond pas à l'état du bouton, car, comme expliqué plus haut, la variable Digital/AnalogIn est une structure. Nous avons donc utilisé les fonctions write() et read() afin d'écrire et de lire la valeur des boutons.

Nous avons aussi eu des difficultés avec l'utilisation d'une machine à état. L'idée d'utiliser une machine à état nous paraissait correspondre à notre envie d'enregistré le code composé numéro par numéro dans différents états et à la fin vérifier le code. Cette méthode marchait très bien en mode debug sur la stm32 mais lorsqu'il a fallu implémenter le code sur le microcontrôleur, le programme ne marchait plus. Nous avons alors compris que cette méthode n'était pas la meilleure car le programme ne restait pas dans le bon état sans l'appui d'un bouton, ou il passait trop vite d'un état à l'autre. Nous avons alors opté pour une boucle if dans le main qui s'incrémente pour enregistré les numéros appuyés et vérifie le code à la fin.

Le vrai digicode nous a aussi posé problème car il nous a fallu plusieurs tests avec un multimètre afin de vraiment comprendre quels pins correspondaient aux colonnes et lesquels correspondaient aux lignes. Nous avons aussi dû trouver une méthode pour récupérer les données du digicode car celui-ci n'est pas alimenté. Donc il ne fallait plus simplement récupérer les états des pins comme le faisait notre programme avec le digicode simple, mais aussi mettre au niveau haut certains pins (les lignes) et récupérer les colonnes.

- Puis, voici les problèmes rencontrés avec le PCB :

Le majeur problème rencontré a été de tracer un circuit imprimé qui correspondait à l'utilisation que l'on voulait faire de la stm32 mais aussi de respecter les paramètres qu'imposait le technicien pour l'impression du circuit. Nous avons obtenu 2 couches, mais à ce moment-là nous n'avions pas pensé que la soudure allait être compliqué car les soudures étaient à faire sur les deux couches. Ce qui l'a rendu très délicat, notamment pour le header.

Il a donc fallu faire très attention pour pas que les soudures se touchent et que les deux côtés soient bien soudés. C'est grâce au technicien que nous avons pu réaliser notre PCB.

Il nous a aussi manqué du temps pour réaliser le circuit du moteur en PCB. Nous avons donc choisi d'utiliser une plaque prototype pour le faire mais il était très compliqué de le souder sur ces plaques. N'ayant pas de liaison entre les composants, il était nécessaire de les faire par nous-même.

- Enfin, les problèmes pour les capteurs IR de présence et de courant moteur

Nous avons incorporé ces deux capteurs dans notre portail, pour avoir une mesure de sécurité optimale. Cependant, la conception des capteurs est restée très simpliste, à faute du manque de temps. Elle consiste en la comparaison entre la tension actuelle envoyée par les capteurs et la tension initial envoyé lors du démarrage du portail. Or, nous avions remarqué que les capteurs fluctuaient de manière assez marquée, ce qui nous empêchait d'avoir un fonctionnement correct du portail sous la surveillance des capteurs. En effet, avec une condition trop fine, le portail s'arrêtait de manière immédiate, et lorsque la condition était trop indulgente, ils ne servaient plus à rien.
Nous pensons faire, pour le capteur de courant, un système asservi pour le moteur, pour qu'il puisse tourner à une vitesse constante pendant le fonctionnement du portail. Ainsi, cela permettrait d'avoir un courant beaucoup plus stable et constant, ce qui pourrait rendre la comparaison plus réaliste et exacte.

3. Résultats

En prenant compte des objectifs donnés, nous avons accompli une grande partie de notre projet : nous avons réussi à faire fonctionner le portail, de deux manières différentes. Cependant, la non gestion du temps nous a fortement ralentie, ce qui nous a fait perdre une partie de la commande et les derniers tests sur la partie de la sécurité.
Le portail fonctionne correctement, mais sans aucune mesure de sécurité.

ENSEA
Ecole Nationale
Supérieure
de l'Electronique
et de ses Applications

Conclusion

Pour ce projet électronique, nous avions pour but de mettre en œuvre un portail automatique. La démarche était longue, pour commencer il était nécessaire de faire un point sur ce que nous avions à notre disposition et leur fonctionnalité, puis ensuite mettre en évidence les modules manquantes pour le fonctionnement du portail. La création de ceux-là n'étaient une simple tâche, où nous avons rencontrés de divers problèmes, qui nous avons dû contourner. Enfin, nous avons élaboré plusieurs méthodes d'améliorations du portail, et les avons mis en œuvre.

Ainsi, nous pouvons approuver le bon fonctionnement du système, avec plusieurs modules de commandes. Cependant, cela sans la prise en compte des modules de sécurité. Nous ne pouvons donc pas amplement nous satisfaire de nos résultats, et pensons continuer à poursuivre ce projet de manière personnel pour pouvoir aboutir à ses fins.

Finalement, ce projet de 1ère année nous a mis au défi des capacités importantes pour l'ingénieur, passant par la conviction et la volonté, ou par l'autodiscipline. Ce projet nous a fait remarquer la difficulté de mettre en œuvre un travail efficace et productif au sein d'une équipe, montrant les vrais enjeux que forment un travail collectif.
Finalement, nous pouvons affirmer que ce projet nous a apporté de beaucoup, notamment sur le domaine humain que met en jeu un projet, et avons pu toucher à un vrai projet professionnel que les ingénieurs font faces.
Globalement, nous sommes très satisfaits de cet atelier, et espérons effectuer de futurs projets pour fortifier les bases que nous aurons acquis ici.