INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN

Projet de Physique P6

STPI/P6/2016 – #17

REALISATION D'UN PROGRAMME DE PILOTAGE D'UN ANCIEN AFFICHEUR SNCF

Figure 1 : Image du panneau d'affichage.

Etudiants :

Simon BRUNET

Ali BOUJADARIA

Pierre LARRENIE

Maxime CARPENTIER

Marianne RUBAT-CIAGNUS

Enseignant-responsable du projet :

Clément KELLER

Date de remise du rapport : ***13/06/2016***

Référence du projet : ***STPI/P6/2016 – #17***

Intitulé du projet : ***REALISATION D'UN PROGRAMME DE PILOTAGE D'UN ANCIEN AFFICHEUR SNCF***

Type de projet : ***Expérimental, conception, simulation.***

Objectifs du projet *(10 lignes maxi)* :

Le but du projet est de comprendre et d'analyser le fonctionnement d'un ancien afficheur SNCF dans l'optique de pouvoir le faire fonctionner avec du matériel récent.

En effet, à partir des modules qui composent l'afficheur, nous devons former des lettres et donc des mots. Ces mots seront saisis sur un ordinateur et devront s'afficher sur les modules.

Mots-clefs du projet *(4 maxi)* : ***Arduino, afficheur, SNCF.***

TABLE DES MATIERES

NOTATIONS, ACRONYMES

- **Arduino, Arduino Méga :** microcontrôleur programmable (le model « Méga » est le plus puissant).
- **CLK, Clock :** signal horloge permettant la synchronisation de la carte Arduino avec le MAX7219.
- **Démultiplexeur, MAX, MAX7219, Driver LED** : composant électronique servant à piloter une matrice LED.
- **DIN, DOUT :** signal portant les informations des lignes et colonnes.
- **Fritzing :** Logiciel d'édition de schéma électronique.
- **LedMatrix :** Librairie de fonction (langage C) servant à piloter le MAX.
- **LOAD :** signal permettant de sélectionner un MAX.
- **Shield Web :** Extension Arduino permettant de générer un serveur web pour cette dernière.
- **ULN2803, transistors en série** : composant électronique servant à relier, de manière logique, deux circuits alimentés par différentes sources électriques.
- **UND2982, amplificateurs opérationnels en série** : composant électronique servant à amplifier, à une tension donnée, un signal entrant.

1. INTRODUCTION

Dans le cadre des projets de P6 de deuxième année de STPI, un ancien afficheur SNCF nous a été fourni par M. Keller. Les seuls éléments que nous avions au départ étaient le panneau constitué de 12 modules d'affichages et les câbles reliés à d'anciennes cartes mères. Sans le matériel utilisé à l'époque pour faire fonctionner l'afficheur, celui-ci était inutilisable par un proche de M. Keller, artiste, dont le but était d'utiliser l'afficheur en tant qu'œuvre artistique.

L'objectif principal de ce projet est donc de concevoir un système permettant dans un premier temps de faire fonctionner l'afficheur, en formant différentes lettres ou chiffres sur chacun des modules et de pouvoir les piloter à travers une interface informatique. Cette interface permettrait ainsi d'afficher une phrase de manière presque instantanée.

Ce projet, nécessitant des notions d'électronique et d'informatique, implique par ailleurs une grande rigueur, une bonne gestion du temps et des recherches. Tous ces éléments participent activement à notre cursus pédagogique et nous ont permis de nous former sur tous ces aspects qui sont nécessaires à un ingénieur.

Pour mener à bien ce projet, nous devions tout d'abord analyser le système mis à notre disposition et déterminer les éléments à garder et ceux à mettre de côté. Par, exemple, nous avons décidé de ne pas garder tout le système de câblage, trop ancien, qui ne s'adaptait pas aux nouveaux composants que nous avions trouvés. Car, en effet, ce projet demandait aussi une phase de recherche de composants pour notre montage. Nous avons ensuite réalisé un montage initial à partir de nos composants principaux et un code que nous avons modifié complexifié et amélioré au fur et à mesure des tests que nous avons réalisés. Enfin nous présenterons le résultat final, les solutions et modèles envisageables pour faire fonctionner l'afficheur.

2. METHODOLOGIE / ORGANISATION DU TRAVAIL

L'étude de ce projet s'est réalisée en plusieurs étapes.
Dans un premier temps, nous avons cherché à analyser le système pour comprendre son fonctionnement : composants utilisés, mécaniques constatées...
Ensuite, une longue phase de documentation s'est mise en place. A l'aide d'Internet, il a fallu chercher des solutions envisageables pour la réalisation du projet. L'utilisation de cartes Arduino s'étant vite imposée, il a fallu se renseigner sur le fonctionnement de ces cartes et de leurs limites (coûts, puissance fournie par la carte, composants utilisables en complément…).
Une phase de sélection et de conception d'une solution a été nécessaire : inventaire des composants à utiliser, réalisation d'un schéma électronique et commande des pièces.

La répartition des tâches a été un processus complexe. En effet, bien que le travail était plus ou moins réparti en petits groupes différents, selon les séances, l'aspect technique du projet ne permettait pas d'être réellement efficace séparément. Les travaux sur le code et le montage sont liés : sans le code, il est impossible de faire fonctionner le montage, et inversement. Ainsi, si l'un des deux ne fonctionne pas, c'est tout le processus de division des tâches qui est affecté.

Néanmoins, les séances de projet (à raison d'une heure et demie par semaine) nous permettent de faire un point en début de séance sur l'avancée du projet, afin de fixer des objectifs pour la séance en cours. Ainsi, les différents paliers nécessaires à l'avancée du projet sont clairement établis, et chaque membre du groupe peut se focaliser sur un objectif en particulier. Le but étant que tous les membres du groupe soient impliqués sur toutes les parties du projet (montage, code…) et d'éviter ainsi une division du travail qui entraînerait une méconnaissance sur le travail des autres.

3. TRAVAIL REALISE ET RESULTATS

3.1. Travail préliminaire

3.1.1. Analyse du système

Le mode d'emploi de l'afficheur n'étant malheureusement pas fourni avec celui-ci, nous nous sommes attelés à essayer de comprendre le fonctionnement de l'afficheur. Celui-ci est en effet composé de 12 modules : chaque module commande une lettre, ou un chiffre indépendamment ; la combinaison de ces modules permet d'afficher un mot, une phrase ou une série de chiffres. Ces modules, disposés côte à côte, sont reliés à ce qui s'apparente à deux cartes mères. Différents composants sont utilisés par ces cartes mères (des transistors, et une sorte de RAM -mémoire vive-), qui permettent probablement de piloter les différents modules.

Figure 2 : Intérieur du panneau d'affichage

Nous sommes rapidement arrivés à la conclusion que l'étude du fonctionnement d'un seul module permettrait d'élaborer un modèle pour faire fonctionner un groupe de modules. Chaque module est composé de 35 pastilles à deux côtés : un noir, et un jaune. Différentes broches sont connectées à ces pastilles : une partie des broches commande la rotation des pastilles d'une colonne, tandis que l'autre partie commande celle d'une ligne. En alimentant une broche de la partie ligne et une broche de la partie colonne, on commande ainsi la rotation d'une pastille en particulier. Le module fonctionne donc comme une matrice 7x5. En inversant le branchement, on peut au contraire revenir à l'état de base, c'est à dire celui où les pastilles affichent un côté noir. Pour se faire, il faut mettre le pôle plus sur la broche reset et ensuite pointer sur les différentes colonnes à faire passer au noir.

Les broches sont en fait reliées à des bobines : une perturbation du champ électrique entraîne donc une rotation d'une pastille en particulier. De plus, nous avons déterminé à l'aide d'un multimètre que la rotation d'une pastille ne se faisait que sous une tension minimum de 10V et une intensité de 300mA. L'objectif a donc été dans un premier temps de déterminer une solution permettant de délivrer une tension suffisante pour chaque combinaison de broches afin de provoquer la rotation d'une pastille sélectionnée par l'utilisateur. Ensuite, nous devions trouver un moyen de contrôler chaque pastille indépendamment les unes des autres pour pouvoir afficher la lettre désirée.

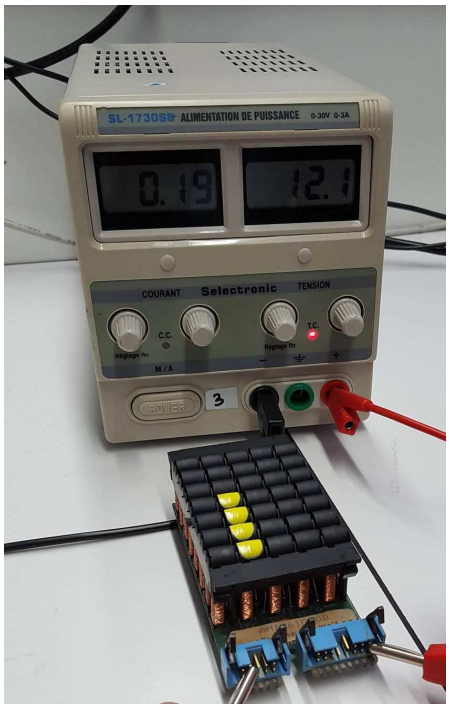

Figure 3 : Test de fonctionnement des modules

3.1.2. Recherche d'une solution : les composants

Figure 4 : Exemple d'affichage possible

Nous avons tout d'abord essayé de retrouver l'afficheur à travers diverses recherches sur Internet. Après consultation du numéro de série et des références indiquées sur le boîtier, il s'est rapidement avéré que l'afficheur étudié était un modèle assez original. En effet, le système date de 1994, et a été retrouvé dans une ancienne gare espagnole. De plus, l'utilisation de ces afficheurs a été limitée, contrairement à d'autres types d'afficheurs (à palettes ou LED par exemple). La recherche d'un mode d'emploi ou de documentation a donc été inefficace.

L'étude de projets analogues a néanmoins permis de convenir d'une solution avec pour base une carte Arduino. Ces microcontrôleurs peuvent être utilisés pour produire des signaux électriques et peuvent être entièrement programmables. De plus, les cartes Arduino sont accessibles à prix abordables, et peuvent, en fonction du modèle, disposer d'une large gamme de connecteurs. Divers projets sur Internet traitent d'un système qui se rapproche de

celui étudié : les matrices LED. Ces matrices LED utilisent en effet une carte Arduino pour fonctionner, et reposent sur le même système que celui étudié : chaque LED peut s'éteindre et s'allumer indépendamment des autres en fonction du souhait de l'utilisateur.

En outre, des bibliothèques de fonctions peuvent être utilisées par Arduino pour gérer l'affichage d'une matrice LED à travers un programme informatique, ce qui correspond aux objectifs de notre projet.

Néanmoins, la plus grande des cartes Arduino (la carte “Méga”) ne dispose pas d’assez de sorties pour contrôler 12 modules de 35 broches. Il nous fallait donc trouver une solution pour multiplier le nombre de sorties contrôlables à partir d’une carte Arduino comprenant 54 sorties. Pour notre panneau, il faut pouvoir contrôler près de 150 sorties indépendamment les unes des autres. Ainsi pendant une séance entière, nous avons cherché un composant capable de gérer ce nombre important de sorties.

Nous avons finalement choisi d’utiliser un démultiplexeur MAX7219, dont nous détaillerons le fonctionnement par la suite. Nous avons également dû trouver une solution pour fournir les 10V (12 de préférence) requis par notre module car la carte Arduino ne peut délivrer qu'une tension de 5V, insuffisante pour faire tourner une pastille de notre module. Toutefois, le démultiplexeur MAX7219 ne peut supporter qu'une tension de 5V. Il fallait donc pour ne pas endommager le composant amplifier le signal en sortie du démultiplexeur. Pour cela nous avons donc commandé l'UDN2982 et les ULN2803 qui ne correspondent ni plus ni moins qu’à des amplificateurs et transistors indépendants les uns des autres mais regroupés en un seul composant. En connectant les sorties du MAX aux entrées de l'UDN on pouvait théoriquement amplifier le signal sans perdre d'information.

3.2. Conception

3.2.1. Schéma du montage et code informatique

Les composants utilisés pour le fonctionnement du système sont :

Nom du composant / outils de test	Rôle	Quantités
Carte Arduino MEGA	Gérer la transmission de l’information et alimentation	1
Générateur	Alimenter l’UDN et les MAX	1 à 2
UDN2982	Amplifier le courant	1 (pour 1 module) 12 (pour tous les modules)
MAX7219CNG (DIP24)	Sélectionner la transmission du courant (pour les broches)	1 (pour 1 module) 12 (pour tous les modules)

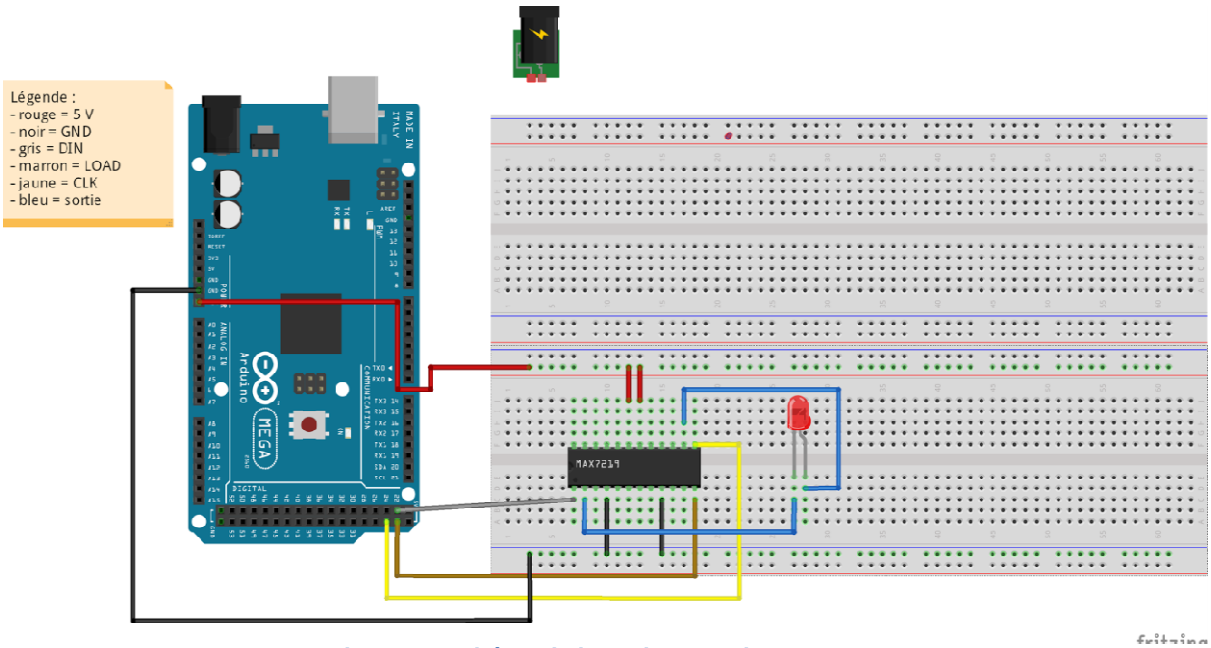

Figure 5 : Schéma de branchement du MAX7219

Après avoir trouvé et commandé le démultiplexeur, nous avons conçu un schéma à l'aide du logiciel Fritzing. Nous avons tout d'abord décidé de réaliser le montage permettant de contrôler un module. Avant de connecter le démultiplexeur, nous avions étudié avec attention son fonctionnement, à l'aide de la documentation fournie avec le composant.

Le MAX7219 reçoit trois informations principales de la carte Arduino: le CLK (un signal horloge permettant la synchronisation de la carte Arduino avec les MAX7219), le LOAD (qui permet de sélectionner le MAX souhaité) et le DIN (permettant d'envoyer les données au composant). Ainsi, lorsque la carte envoie le voltage (via le LOAD) correspondant au démultiplexeur n°1 par exemple, ce dernier va regarder sur sa broche DIN le signal à appliquer sur ses broches contrôlant la matrice LED. Ainsi, cela permet de décider quel port du démultiplexeur va se fermer au niveau de la colonne et de la ligne (la masse), permettant ainsi de contrôler une pastille précise de notre module.

Nous avons donc tout d'abord réalisé un premier montage (Voir Figure 6). Nous avons connecté le VIN de la carte Arduino au branchement d'alimentation du démultiplexeur afin de lui fournir une tension de 5V. Cette tension n'est délivrée qu'à la partie supérieure du MAX7219, partie qui correspond au contrôle des colonnes. La partie des lignes est quant à elle connectée au "GND" de la carte Arduino.

Dans le cas où nous voudrions brancher plusieurs modules à la fois, et donc plusieurs MAX il nous faut comprendre en détail l'utilité des trois inputs spéciaux du démultiplexeur :

- Le CLK (entrée 13 du MAX7219) permet de contrôler le MAX7219 en fonction du temps (CLK pour clock) et de se synchroniser avec la carte Arduino. Il est connecté à la sortie 25 de cette dernière.
- Le DIN (entrée 1 du MAX) est utilisé pour coder au MAX à quelle colonne et quelle ligne il doit permettre le passage du courant, et est relié à la sortie 23 de la carte.
- Le LOAD (entrée 12 du MAX), permet de déterminer quel MAX7219 il faut utiliser (dans le cas où l'on en aurait plusieurs), et est connecté à la sortie 22 de la carte.

Ainsi, si plusieurs MAX7219 sont branchés en série, et que l'on souhaite sélectionner le démultiplexeur n°2, le démultiplexeur n°1 recevra sur la broche LOAD une tension qui n'est pas la sienne. De ce fait, il transmettra l'information reçue sur la broche DIN vers la broche DOUT. Cette dernière est reliée à la broche DIN du démultiplexeur n°2.

Toutes les sorties du démultiplexeur codant pour les colonnes sont sur la ligne du haut et celles codant pour les lignes du module sont sur la ligne en bas du MAX (reliée à la masse).

Nous avons représenté sur l'image ci-dessous un pixel du module par une LED et l'on peut voir la carte Arduino connectée au démultiplexeur, lui même connecté par des câbles bleus, à la LED.

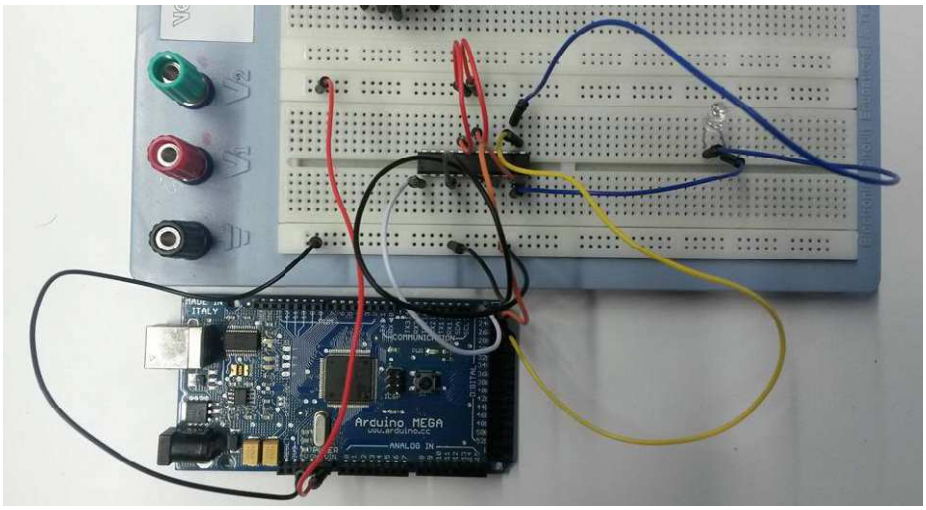

Figure 6 : Montage permettant l'allumage d'une LED

Néanmoins, il s'est avéré que nous avons dû rajouter un amplificateur UDN2982A.

En effet, en sortie du montage, nous nous retrouvions avec une tension oscillant entre 4.4 et 5V. Cependant lors des tests que nous avons réalisés pour comprendre le fonctionnement des modules, nous avons trouvé que la tension minimale pour faire tourner une pastille était d'environ 10V. Pour que notre montage puisse fonctionner avec les modules en notre possession, il fallait donc amplifier le signal en sortie du MAX7219.

Nous avons dans un premier temps pensé à faire une porte logique avec deux transistors. Cela nous aurait donc permis d'obtenir la puissance nécessaire au fonctionnement du module.

Mais vu le nombre de transistors nécessaires au fonctionnement de cette solution (deux par broche environ), nous nous sommes repliés sur des amplificateurs opérationnels.

L'UDN2982 a un fonctionnement simple. On lui apporte une tension en entrée et elle ressort augmentée en sortie. Cet élément est composé d'amplificateurs opérationnels branchés en parallèle.

L'utilisation de l'UDN2982 et du MAX7219 nous permettait donc, en théorie, de résoudre les problèmes liés au montage électrique. Il reste maintenant à étudier la phase de programmation qui est, elle aussi, aussi complexe.

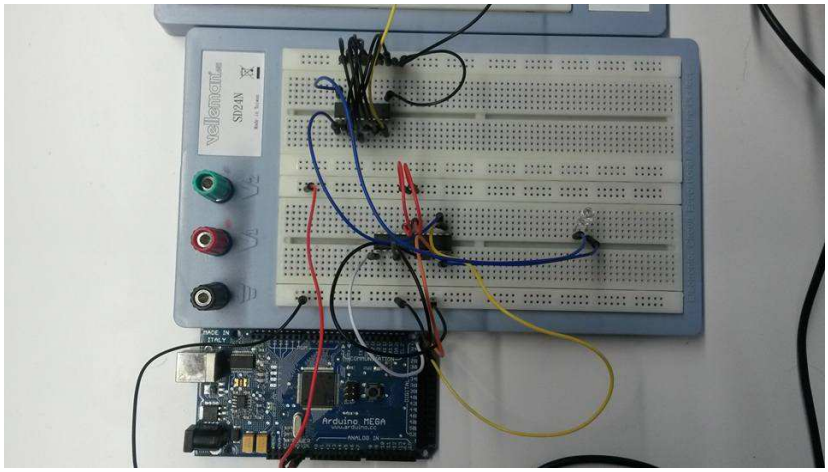

Figure 7 : Montage permettant d'allumer une LED avec amplificateur.

Les contrôleurs Arduino sont programmables à l'aide de différents langages de programmation, comme le Java ou le C. Nous avons choisi ici d'utiliser le langage C, plus accessible. L'utilisation de la bibliothèque de fonctions "LedMatrix" permet de contrôler le démultiplexeur, pour choisir quelles sorties de ce composant sont à alimenter. Avant de commander une carte Arduino, nous avons utilisé une carte gracieusement fournie par les laboratoires de l'INSA de Rouen. Pour réaliser le code, nous nous sommes inspirés d'implémentations préexistantes sur les cartes Arduino. (Voir Annexe 4 -partie 6.4-)

Une fois le montage réalisé et le code écrit nous avons fait nos premières phases de tests.

3.2.2. Problèmes et phases de tests

Sur les premières phases de tests, notre souci principal était que nous ne savions pas d'où provenaient le ou les problèmes. Ainsi, comme dans tout circuit électronique, une mauvaise connexion des composants pouvaient bloquer tout le système.

Étaient-ce les composants, que nous avions trouvés sur Internet, qui théoriquement devaient convenir mais qui en pratique n'étaient peut-être pas adaptés à notre projet ? Ou bien était-ce le code qui contenait une erreur de programmation ? Peut-être étaient-ce les deux? Nous devions ainsi tâtonner pour trouver le ou les problèmes de notre montage.

Différents outils d'analyse ont été utilisés pour vérifier le bon fonctionnement du montage. Ainsi, nous avons placé en sortie du démultiplexeur un oscilloscope afin de visualiser les signaux que l'on obtenait en sortie de notre montage.

Au début nous avons adapté notre montage en essayant d'obtenir un signal cohérent sur l'oscilloscope. Mais voyant que cela n'était pas très efficace, nous avons tout démonté et nous avons reconstitué notre montage petit à petit en testant à chaque étape si chaque composant renvoyait bien ce que l'ont désirait.

Tout d'abord nous nous sommes assurés que la carte Arduino nous fournissait bien 5V, et si elle renvoyait bien une tension associée au démultiplexeur pour le fermer, et deux autres tensions pour fermer une ligne et une colonne. Nous avons également remarqué que notre code n'envoyait pas la tension à la bonne colonne et la bonne ligne. Pour s'affranchir de ce problème, nous avons temporairement programmé la fermeture de tous les ports afin de visualiser en continu notre signal.

Par la suite, nous avons effectué le test en codant pour une ligne et une colonne précise. Nous avons finalement réussi à observer un signal en créneau. Nous avions donc la certitude d'avoir, à priori, connecté le MAX7219 correctement.

Après avoir déterminé que notre code et notre montage fonctionnaient ensemble et renvoyaient bien ce que l'on désirait, nous avons réalisé un test sur notre module. Il s'est rapidement avéré que les modules ne parvenaient pas à interpréter le signal électrique reçu pour engager la rotation des pastilles.

En effet, les modules ont besoin de recevoir au moins 10V (12V de préférence, pour être sûr que toutes les pastilles tournent) pour fonctionner. Cependant, notre démultiplexeur ne peut pas envoyer plus de 6V d'après la notice, comme la plupart des composants que nous avions à notre disposition.

Néanmoins, une fois la bonne tension obtenue grâce à l'UDN précédemment cité, les pastilles sont restées immobiles. Nous avons donc conclus que cela n'était pas seulement qu'une affaire de tension mais également un problème d'intensité. En effet la rotation de la pastille est due au champ magnétique induit par la bobine correspondante du module. Or nous savons que le champ magnétique est fonction de plusieurs paramètres tels que le nombre de spires, le rayon ou l'intensité par exemple. Seulement, le seul paramètre qui nous est permis de modifier est l'intensité pour alimenter le module. En effet après d'autres tests, nous avons remarqué que la pastille tournait seulement si le module était alimenté avec un courant de 0.3-0.5 A à 10V et 0.4-0.6A à 12V.

3.3. Résultats et solution finale

3.3.1. Résultats attendus

Il est évident que le résultat que nous attendions était de pouvoir faire fonctionner plusieurs modules simultanément à l'aide d'une seule carte Arduino. Pour cela nous voulions tout d'abord réussir à mettre en marche un seul module. Enfin, nous voulions généraliser

notre montage en branchant les modules au montage via les démultiplexeurs. Il aurait alors suffit simplement, à l'aide du code théorique, d'écrire la lettre, pour un module, ou bien le mot ou la phrase, pour une combinaison de modules, et cela aurait été affiché.

De plus, le site d'Arduino propose un code JavaScript incorporable à une page web permettant de télé-verser du code sur une carte Arduino. Ainsi, un simple champ texte d'une page web aurait pu servir d'interface pour écrire un mot sur les modules.

3.3.2. Résultats obtenus

A notre dernière séance d'essais nous avons fini par obtenir entre 12 et 25V en sortie mais l'intensité était quasiment nulle. Lors des premières mesures, nous avions réussi à faire tourner une pastille avec 0.3 A, or le composant d'amplification était censé nous fournir un courant de 0.5A au maximum. Cela était donc théoriquement suffisant pour faire tourner une pastille. Cependant rien ne tournait. En effet il s'est avéré que l'amplificateur ne délivrait quasiment aucun courant et au delà de ce problème il amplifiait une tension nulle. Ainsi, le composant qui devait amplifier le signal ne jouait pas son rôle car il alimentait toutes les colonnes du module en permanence. En théorie quand ce dernier reçoit en entrée une tension de 5V, il l'amplifie de manière à obtenir, en sortie, une tension proche de celle de son alimentation.

En somme, nous avons réussi à faire un montage nous permettant en théorie de commander différentes lignes et colonnes du module.

3.3.3. Résultat théorique

En supposant que tous les composants fonctionnent correctement, il nous est possible de réaliser un code théorique capable d'afficher des lettres.

En effet, si le MAX continue à jouer son rôle et que l'UDN délivre la puissance nécessaire à la rotation d'une pastille, notre montage est censé fonctionner.

Nous proposons en annexe 1 (partie 6.1) le code théorique en rapport avec notre montage. Il ne faut cependant reprendre ce code qu'avec recul car il a été développé avec la librairie "LedMatrix" qui est la librairie spécifique au MAX7219.

3.3.4. Autres pistes

Nous avons détecté plusieurs problèmes qui sont restés sans solutions.

Le démultiplexeur ne fonctionne pas comme prévu. En effet, selon les informations de la notice, le MAX doit laisser les lignes et colonnes non alimentées si elles ne sont pas utilisées dans le programme. Or, même avec un code vide (c'est-à-dire en utilisant aucune ligne ni colonne) ces dernières restent toutes alimentées. Plus étrange encore, en ne laissant que les trois broches d'informations (CLK, DIN et LOAD), sans l'alimentation du MAX, ce dernier continue à fonctionner.

Nous avons donc réfléchi à un montage qui pourrait fonctionner sans ce composant.

Il s'agirait de commander un module avec uniquement les ports de l'Arduino. Cette solution avait été rejetée au début de notre projet pour son manque de flexibilité avec l'augmentation du nombre de modules. Malgré tout, nous voulions essayer de commander un module à partir de l'Arduino pour afficher quelque chose.

En étudiant cette solution plus en détail, il s'est avéré que nous ne pourrions pas afficher de lettre mais simplement un « rideau de pluie ».

En effet, comme sur le schéma ci-dessous, nous utilisons l'ULN pour pallier le manque de puissance de l'Arduino. Ainsi toutes les colonnes d'un module sont reliées au pôle + du générateur et les lignes sont branchées sur l'ULN qui est relié à la carte Arduino. Etant donné que L'UDN à notre disposition ne fonctionne pas, nous sommes obligés de perdre l'information sur les colonnes. Ainsi, lorsque l'on va commander une ligne avec la carte, toutes les pastilles de la ligne vont se retourner en même temps.

En faisant une porte logique XOR avec l'alimentation, on peut réaliser le reset sur toute la matrice en même temps.

Cette solution pourrait donc faire tourner des pastilles en faisant des sortes de "rideaux de pluie"

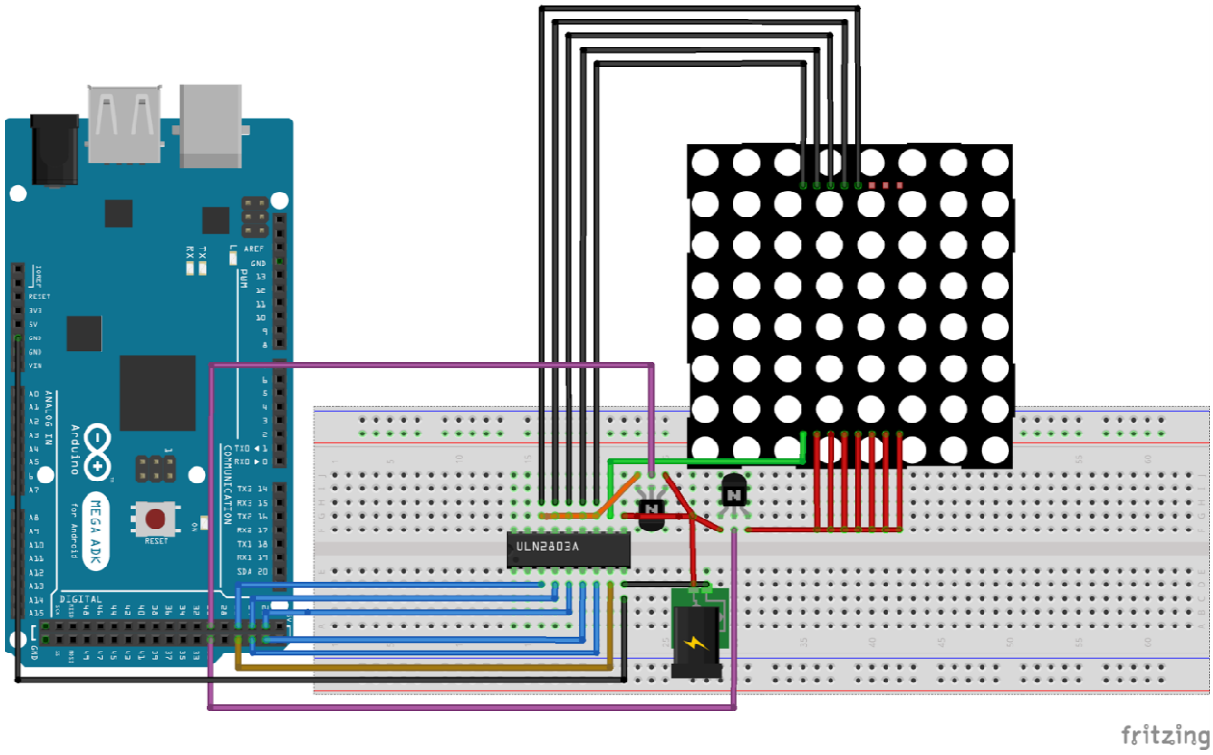

Figure 8 : Schéma d'une solution alternative : "Rideaux de pluie"

Dans l'optique où nous arriverions à faire fonctionner le MAX7219, il y aurait une solution qui serait de construire son propre UDN2982. En effet, comme le fait l'ULN, nous pourrions avec deux transistors faire une porte logique qui, lorsqu'elle recevrait une information du MAX, s'ouvrirait et fermerait le circuit.

Nous avons des doutes quant à la mise en place d'un tel système à cause du grand nombre de transistors nécessaires à la réalisation d'un seul module.

Enfin, si nous possédions un budget important, il serait envisageable d'acheter des composants avec un rôle similaire et fonctionnant directement avec des tensions de l'ordre de 12V.

Figure 9 : Exemple de Driver LED 12V (TWLC by The Warm Light Company Ltd.)

4. CONCLUSION ET PERSPECTIVE

Ce projet nous a permis de mettre en œuvre deux compétences importantes pour le métier d'ingénieur : la recherche en autonomie et l'auto -apprentissage.

Dès les premières séances, nous avons commencé nos recherches en étudiant le fonctionnement du panneau d'affichage. Nous avons pu réaliser un modèle de solution en définissant les caractéristiques des composants à trouver.

Nous en sommes rapidement venus à la conclusion qu'il nous fallait reprendre les composants principaux (modules) et reconstruire un montage permettant de les contrôler. Ainsi, nous avons appris à chercher des composants adéquats à notre problème, à comprendre leur fonctionnement et leur schéma de branchement et à réaliser un montage cohérent avec notre modèle.

Nous avons également représenté nos montages à l'aide du logiciel Litzing afin d'avoir un support sur lequel se référer lors de nos tests.

Par ailleurs ce projet demande de bonnes compétences en électronique afin de réaliser un montage fonctionnel avec des composants que nous avions trouvés nous-mêmes. Lors du cursus INSA, nous n'avons pas pu bénéficier d'assez de cours sur ce sujet pour avancer rapidement sur le projet. Néanmoins, nous avons su nous adapter pour répondre au mieux aux attentes de ce travail.

Enfin, suite à l'analyse de notre résultat final, nous avons élaboré une autre piste encore non exploitée qui permettrait par la suite de tester un autre montage et trouver une solution fonctionnelle.

Ainsi ce projet nous a permis d'apprendre et de développer des compétences d'organisation, de recherche et de modélisation, ainsi que d'élargir nos connaissances techniques en électronique et en informatique.

5. BIBLIOGRAPHIE ET TABLE DES ILLUSTRATIONS

5.1. Bibliographie

[1] Allegro Inc., “Datasheet of UDN2981 & UDN2982”.

[2] Maxim ltd., “Datasheet of MAX7219 DS”.

[3] Eskimon et Olyte, “Arduino pour bien commencer en électronique et en programmation”, *OpenClassRoom*, v2.5, 2012.

[4] lien internet : http://tronixstuff.com/2013/10/11/tutorial-arduino-max7219-led-display-driver-ic (valide à la date du 12/06/2016) : Site proposant un exemple d’utilisation du MAX7219.

5.2. Table des illustrations

6. ANNEXES

6.1. Programme théorique final.

```
#include <LedControl.h> // Code inspiré et modifié de tronixstuff.com

const int numDevices = 5;      // number of MAX7219s used

LedControl lc=LedControl(22,25,23,numDevices);

prog_uchar scrollText[] PROGMEM ={
    "HELLO\0"};

void setup(){
    for (int x=0; x<numDevices; x++){
        lc.shutdown(x,false);       //The MAX72XX is in power-saving mode on startup
        lc.setIntensity(x,8);       // Set the brightness to default value
        lc.clearDisplay(x);         // and clear the display
    }
}

void loop(){
    dispMessage(scrollText);
    dispFont();
}

////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////

prog_uchar font5x7 [] PROGMEM = {      //Numeric Font Matrix 7*5 Char
    B00000000,B00000000,B00000000,B00000000,B00000000,B00000000,B00000000,6,//Space (Char 0x20)
    B10000000,B10000000,B10000000,B10000000,B00000000,B00000000,B10000000,2,//!
    B01000000,B11000000,B01000000,B01000000,B01000000,B01000000,B11100000,4,//1
    B01110000,B10001000,B00001000,B00010000,B00100000,B01000000,B11111000,6,//2
    B11111000,B00010000,B00100000,B00010000,B00001000,B10001000,B01110000,6,//3
    B00010000,B00110000,B01010000,B10010000,B11111000,B00010000,B00010000,6,//4
    B11111000,B10000000,B11110000,B00001000,B00001000,B10001000,B01110000,6,//5
    B00110000,B01000000,B10000000,B11110000,B10001000,B10001000,B01110000,6,//6
    B11111000,B10001000,B00001000,B00010000,B00100000,B00100000,B00100000,6,//7
    B01110000,B10001000,B10001000,B01110000,B10001000,B10001000,B01110000,6,//8
    B01110000,B10001000,B10001000,B01111000,B00001000,B00010000,B01100000,6,//9
    B01110000,B10001000,B00001000,B00010000,B00100000,B00000000,B00100000,6,//?
    B01110000,B10001000,B10001000,B10001000,B11111000,B10001000,B10001000,6,//A
    B11110000,B10001000,B10001000,B11110000,B10001000,B10001000,B11110000,6,//B
    B01110000,B10001000,B10000000,B10000000,B10000000,B10001000,B01110000,6,//C
    B11100000,B10010000,B10001000,B10001000,B10001000,B10010000,B11100000,6,//D
    B11111000,B10000000,B10000000,B11110000,B10000000,B10000000,B11111000,6,//E
    B11111000,B10000000,B10000000,B11110000,B10000000,B10000000,B10000000,6,//F
    B01110000,B10001000,B10000000,B10111000,B10001000,B10001000,B01111000,6,//G
    B10001000,B10001000,B10001000,B11111000,B10001000,B10001000,B10001000,6,//H
    B11100000,B01000000,B01000000,B01000000,B01000000,B01000000,B11100000,4,//I
    B00111000,B00010000,B00010000,B00010000,B00010000,B10010000,B01100000,6,//J
    B10001000,B10010000,B10100000,B11000000,B10100000,B10010000,B10001000,6,//K
    B10000000,B10000000,B10000000,B10000000,B10000000,B10000000,B11111000,6,//L
    B10001000,B11011000,B10101000,B10101000,B10001000,B10001000,B10001000,6,//M
    B10001000,B10001000,B11001000,B10101000,B10011000,B10001000,B10001000,6,//N
    B01110000,B10001000,B10001000,B10001000,B10001000,B10001000,B01110000,6,//O
    B11110000,B10001000,B10001000,B11110000,B10000000,B10000000,B10000000,6,//P
    B01110000,B10001000,B10001000,B10001000,B10101000,B10010000,B01101000,6,//Q
    B11110000,B10001000,B10001000,B11110000,B10100000,B10010000,B10001000,6,//R
    B01111000,B10000000,B10000000,B01110000,B00001000,B00001000,B11110000,6,//S
    B11111000,B00100000,B00100000,B00100000,B00100000,B00100000,B00100000,6,//T
    B10001000,B10001000,B10001000,B10001000,B10001000,B10001000,B01110000,6,//U
    B10001000,B10001000,B10001000,B10001000,B10001000,B01010000,B00100000,6,//V
    B10001000,B10001000,B10001000,B10101000,B10101000,B10101000,B01010000,6,//W
```

```
    B10001000,B10001000,B01010000,B00100000,B01010000,B10001000,B10001000,6,//X
    B10001000,B10001000,B10001000,B01010000,B00100000,B00100000,B00100000,6,//Y
    B11111000,B00001000,B00010000,B00100000,B01000000,B10000000,B11111000,6,//Z
};

void dispFont() {
    for (int counter=0x20;counter<0x80;counter++){
        loadBufferLong(counter);
        delay(500);
    }
}

// Display Message
void dispMessage(prog_uchar * messageString) {
    int counter = 0;
    int myChar=0;
    do {
        // read back a char
        myChar =  pgm_read_byte_near(messageString + counter);
        if (myChar != 0){
            loadBufferLong(myChar);
        }
        counter++;
    }
    while (myChar != 0);
}
// Load character into  buffer
void loadBufferLong(int ascii){
    if (ascii >= 0x20 && ascii <=0x7f){
        for (int a=0;a<7;a++){                      // Loop 7 times for a 5x7 font
            unsigned long c = pgm_read_byte_near(font5x7 + ((ascii - 0x20) * 8) + a);     // Index
into character table to get row data
            unsigned long x = bufferLong [a*2];     // Load current buffer
            x = x | c;                              // OR the new character onto end of current
            bufferLong [a*2] = x;                   // Store in buffer
        }
        byte count = pgm_read_byte_near(font5x7 +((ascii - 0x20) * 8) + 7);     // Index into
character table for kerning data
        for (byte x=0; x<count;x++){
            printBufferLong();
        }
    }
}

// Display Buffer on LED matrix
void printBufferLong(){
  for (int a=0;a<7;a++){                    // Loop 7 times for a 5x7 font
    unsigned long x = bufferLong [a*2+1];   // Get high buffer entry
    byte y = x;                             // Mask off first character
    lc.setRow(3,a,y);                       // Send row to relevent MAX7219 chip
    x = bufferLong [a*2];                   // Get low buffer entry
    y = (x>>24);                            // Mask off second character
    lc.setRow(2,a,y);                       // Send row to relevent MAX7219 chip
    y = (x>>16);                            // Mask off third character
    lc.setRow(1,a,y);                       // Send row to relevent MAX7219 chip
    y = (x>>8);                             // Mask off forth character
    lc.setRow(0,a,y);                       // Send row to relevent MAX7219 chip
  }
}
```

6.2. Programme avec interface internet

6.2.1. Le programme

Ce programme est une solution pour contrôler le message à afficher depuis une page Web. Il faudrait installer un shield web ou sélectionner la carte Arduino Mega version Ethernet.

```
// Set up webserver functionality (from Kevin Haw)
void WebServerSetup() {
      Ethernet.begin(mac, ip);
      server.begin();}
// Web server loop (Inspire et modifié du code de Kevin Haw)
////////////////////////////////////////////////////////////////////////
void WebServerLoop() {
  EthernetClient client = server.available();
  boolean bPendingHttpResponse = false; // True when HTTP request and need to output the webpage
  boolean bPreventSecondParse = false;  //Stops a second successive run of parseHttpHeader()
  if (client) {
    // Loop as long as there's a connection
    while (client.connected()) {
      // Do we have pending data (an HTTP request) and is this the first parse?
      if (client.available() && !bPreventSecondParse) {
        // Indicate we need to respond to the HTTP request as soon as we're done processing it
        bPendingHttpResponse = true;
        bPreventSecondParse = true;
        ParseHttpHeader(client);}
      else {
        if(bPendingHttpResponse) {
          bPendingHttpResponse = false;
          bPreventSecondParse = false;
          // send a standard http response header and HTML header
          HtmlHeader(client);
          client.println("<H2>Texte a afficher sur les modules</H2>");
          client.println("<b>Texte affiche:</b> " + line1 + "");
          client.println("<b>Auteur:</b> " + line2 + "");
          client.println("<H2>Nouveau message a afficher</H2>");
          client.println("<form action=\"/?\" method=get>");
          client.println("<b>Message: </b><input type=\"text\" name=\"L1\" maxlength=\"50\" size=\"50\
"/>");
          client.println("<b>From: </b><input type=\"text\" name=\"L2\" maxlength=\"10\" size=\"10\"
/>");
          client.println("<input type=\"submit\" value=\"Submit\" /></form>
");
          // send HTML footer
          HtmlFooter(client);
          // give the web browser time to receive the data
          delay(1);
          client.stop();}}} // End while(connected)  }}
```

6.2.2. *Exemple d'affichage*

6.3. Quelques photos

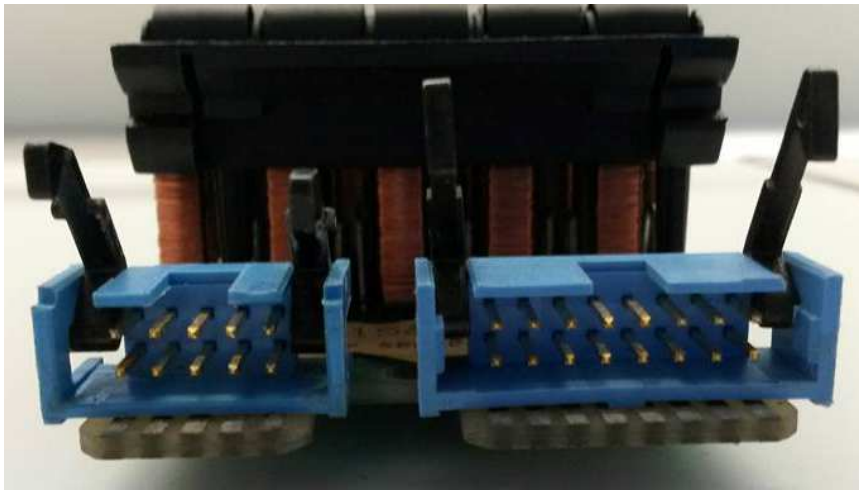

Figure 10 : Broches de connexion d'un module

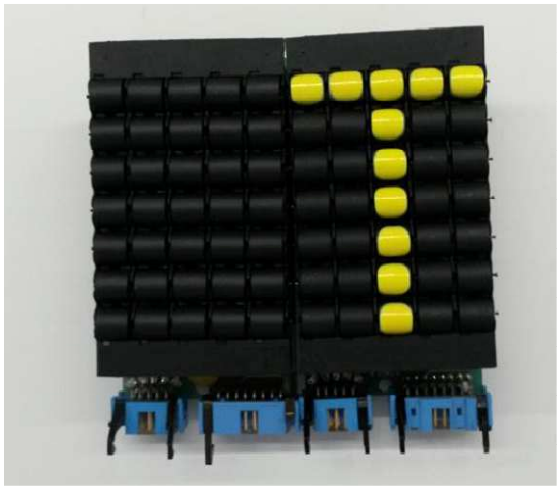

Figure 11 : Deux modules

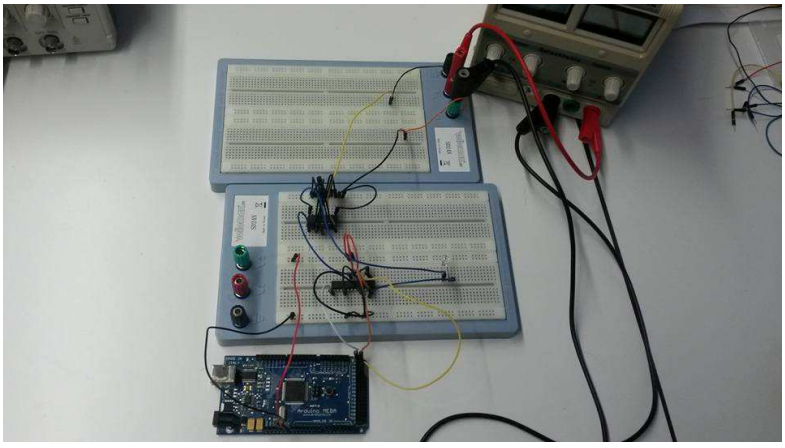

Figure 12 : Test du montage final sur une LED

6.4. Programme test

```
#include "LedControl.h" // need the library
LedControl lc=LedControl(22,25,23,1); //

// pin 22 is connected to the MAX7219 DIN pin 1
// pin 25 is connected to the MAX7219 CLK pin 13
// pin 23 is connected to the MAX7219 LOAD pin 12
// 1 as we are only using 1 MAX7219

void setup()
{
  lc.shutdown(0,false);// turn off power saving, enables display
  lc.setIntensity(0,12);// sets brightness (0~15 possible values)
  lc.clearDisplay(0);// clear screen
  lc.shutdown(1,false);// turn off power saving, enables display

  //lc.clearDisplay(1);// clear screen
  //lc.setLed(0,5,4,true);// test unitaire –commenter loop() si utilise
}

void loop()
{

    for (int row=0; row<7; row++)
    {
    for (int col=0; col<5; col++)
      {
        lc.setLed(0,col,4,true);
    lc.setLed(0,col,row,true); // turns on LED at col, row
        delay(500);
        lc.setLed(0,col,row,false); // turns off LED at col, row
        delay(500);
      }
    }


  for (int row=0; row<7; row++)
  {
    for (int col=0; col<5; col++)
    {
     lc.setLed(0,col,row,false); // turns off LED at col, row
     delay(1000);
      lc.setLed(1,col,row,true); // turns on LED at col, row

    }
  }
}
```