**Compte rendu**

**Présentation**

Ce projet est réalisé dans le cadre de notre cours d’Arduino en PEIP2 de l'école Polytech Nice-Sophia. Nous allons développer un système permettant d'apprendre à lire le Braille en autonomie. Nous utiliserons pour cela une carte Arduino.

**Séance du 15 Décembre :**

Lors de cette séance je devais en apprendre plus sur la manière dont je pouvais créer un code Arduino capable d’associer chaque lettre de l’alphabet a un code en braille. Pour ce faire je me suis dans un premier temps renseigné à l’aide de projet déjà préexistant sur le net. C’est alors que je suis tombé sur le projet de Cesare qui est très similaire au notre.

(Projet braille similaire (ASCII)

<https://create.arduino.cc/projecthub/CesareBrizio/ascii-braille-real-time-translation-via-arduino-dd97a9> )

Je me suis donc attelé à la compréhension de son code. Son code est assez simple à comprendre puisque très bien commenter.

On a aussi pu avoir accès au github du groupe d’il y a trois an. Nous avons donc pris la décision de reprendre leur code puisque celui-ci était censé fonctionner.

(Projet braille d’il y a 3 ans <https://github.com/BlindTouch/Projet> )

La compréhension de celui-ci a été plus difficile mais petit à petit on a pu comprendre quelles étaient les rôles de tels ou tels variables, fonctions.

J’ai ensuite essayé d’adapter en partie leur programme en un nouveau capable de lire un mot donné dans le terminal et de convertir chaque lettre en code binaire (sur six bit ?) et de nous renvoyer celui-ci. Je n’ai pas eu le temps de finir correctement ce programme et il y a encore de choses à revoir dans le programme des anciens étudiants, je continuerais cela chez moi.

**Séance du 5 Janvier :**

J’ai expliqué à mon camarade les parties du code que j’ai comprises (et ses nouvelles modifications) et je l’ai questionné sur celles non comprises. Cela ne m’a pas beaucoup avancé, il y a toujours des fonctions que je ne comprends pas, mais je pense avoir trouvé une solution puisqu’en fin de séance j’ai compris un détail (conversion pour stocker en hexadécimal) qui peut me permettre de mieux comprendre ses zones d’ombres.

Jean a identifié quelle broche correspondait au plus et au moins, nous avons ensuite reçu quelques instructions quant aux risques de mal brancher le générateur 12V (le circuit est alimenté en 12V). Il faut mettre le plus sur Vin pour ne pas faire fondre la carte Arduino ou endommager mon ordinateur.

Pour vérifier les branchements nous avons d’abord tester l’initialisation du programme qui fait bouger les solénoïdes, l’Arduino était alimenté par ma batterie externe pour ne pas casser mon ordinateur en cas d’erreur de branchement.

Après vérification, rien n’a pris feu et l’initialisation fonctionnait, on l’a donc branché à mon ordinateur pour tester le programme. Nous nous sommes malheureusement rendu compte que le programme ne fonctionne pas correctement puisqu’au-delà d’un mot de plus de deux lettres, les lettres sont mal traduites, il y a donc un problème de conversion en binaire (au moment du stockage puis de la redistribution des informations), je pense savoir d’où vient le problème.

Nous avons aussi réalisé quelques soudures au cours de cette séance notamment pour réparer le câblage du plus du circuit.

Après plusieurs minutes de test le système (les solénoïdes) chauffe un peu, mais nous sommes rassurés car on pensait qu’il chaufferait beaucoup plus.

Une image contenant équipement électronique, adaptateur

Description générée automatiquementUne image contenant texte, intérieur

Description générée automatiquement(images des solénoïdes en mouvement (en position haute/pas tous) et des branchements)

**Séance du 12 Janvier :**

J’ai pu me rendre compte qu’un certain nombre de problèmes n’avait pas été soulever/noter au cours de la dernière séance par exemple le fait que l’initialisation ne marche pas toujours. Je ne sais pas si cela vient du fait que les solénoïdes chauffent et donc reste en haut ou si cela vient d’un dysfonctionnement ou même d’un bug lier au code.

Cela nous a amené au second problème, étant donné que certains solénoïdes restent bloquer en haut, il chauffe, on n’avait pas remarqué que ça chauffait autant la semaine dernière étant donné qu’on le coupait après chaque test. Aujourd’hui on l’a laissé activer plus longtemps et ça a nettement plus chauffer (certains solénoïdes se sont même décollés).

Le dernier problème vient du code, j’ai passé l’entièreté de la séance à essayer de trouver l’erreur de traduction. J’ai avancé et pu comprendre d’où provenait l’erreur (conversion), mais je me suis confronté à un nouveau problème, désormais le code me renvoie les bonnes informations (c’est-à-dire la bonne combinaison de chiffres) mais le système avec les solénoïdes ne renvoie pas le bon visuel. Je pense que cela provient du fait qu’une certaine partie d’entre eux étant à l’état haut (bien qu’initialiser à l’état bas) l’information n’est pas stockée et on ne peut traduire un mot complet. La traduction ne fonctionnant que sur deux lettres (avec malgré tous des bugs de temps en temps).

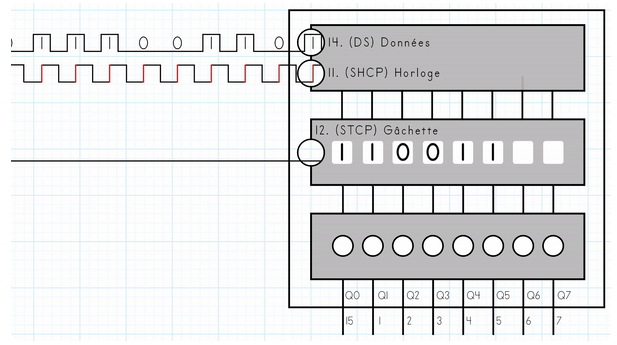
(Après différents tests, je me suis rendu compte que la semaine dernière la traduction était faussé une fois sur deux, elle ne fonctionnait qu’après la première initialisation pour un mot de deux lettres).

Le point clé de résolution de ses problèmes réside donc dans la compréhension des modules ULN2803 et 74HC595.

**Séance du 19 Janvier :**

Nous avons dans un premier temps passé l’oral de 5 minutes, celui-ci consistait en la présentation du projet, des objectifs, du matériels et de l’état d’avancement de celui-ci.

Je me suis ensuite occupé de la compréhension du 74HC595 et de ULN2803.

Je suis tombé sur ce site qui explique très bien le fonctionnement du 74HC595 (<https://siliciumcorp.com/comment-fonctionne-le-8-bit-shift-register-74hc595>).

On a le port DS, qui reçoit la donnée bit par bit.

Le port SHCP qui sert d’horloge, passe de l’état haut a l’état bas, à chaque changement il transmet la donnée au STCP.

(Le remplissage se fait de la gauche vers la droite, sans inversion du code)

Et quand le STCP passe de l’état bas a l’état haut il transmet d’un coup l’entièreté du message (sur 8 bits).

Exemple de code utilisé :

//On s'assure que la gâchette est à l'état LOW

digitalWrite(PIN\_STCP, LOW);

//On lui soumet la valeur à transmettre au registre

shiftOut(PIN\_DS\_DATA, PIN\_SHCP\_CLOCK, LSBFIRST, 1);

//On inverse l'état de la gâchette pour mettre à jour le registre digitalWrite(PIN\_STCP, HIGH);

Au niveau de la documentation sur le ULN2803 cela a été plus compliqué j’ai trouvé moins d’exemples d’utilisation concrète. ULN2803 est un circuit intégré qui dans notre cas permet de contrôler et de commander les solénoïdes.

La chaine d’information se déroulera de la manière suivante : notre Arduino transmet des instructions au 74HC595, celui-ci envoie des signaux logiques pour activer ou désactiver les sorties du ULN2803. L’ULN2803 peut alors se charger de fournir la tension nécessaire pour activer les solénoïdes.

Je n’ai malgré tout pas pu résoudre l’actuel bug (problème d’initialisation) malgré de nombreux tests aux niveaux du code. J’ai essayé de comprendre s’il y avait un pattern au niveau du bug, pour quelles valeurs celui-ci se modifiait, mais aucune logique n’était présente.

Je n’ai donc pas beaucoup bougé depuis la dernière séance même si je pense que le problème vient possiblement : de tensions insuffisantes ou de mauvaises connexions. J’ai trouvé sur internet qu’une des raisons pour laquelle un solénoïde reste en position haute peut-être dû (mis a par le code) à l’envoi d’une mauvaise tension. Voici m’a prochaine mission vérifié chaque tension de chaque solénoïdes bloqué en position haute. (C’est-à-dire plus de 60%).

**Séance du 30 Janvier :**

Lors de cette séance nous avons principalement travaillé ensemble, dans un premier temps nous avons testé une nouvelle liste de test à faire pour essayer de localiser/résoudre le bug. Malheureusement une soudure celle qui alimente le circuit s’est défaite. Le professeur nous a donc proposé de changer cette soudure et de la remplacer par un adaptateur, permettant ainsi de ne plus tirer sur les soudures à chaque branchement ou débranchement du circuit.

Les broches de l’adaptateur ne rentraient pas dans les trous plus et moins prédéfinis (ses broches étaient trop espacées.) nous avons donc dû « tordre » ou du moins retravailler ses broches, puis les soudés. Nous sommes par la suite retournés plusieurs fois à la zone de soudage, car il fallait soit retravailler la soudure des files, soit mieux souder l’adaptateur (lors du premier essaie le courant ne passait pas dans la broche plus).

Nous nous sommes ensuite hâtés au test de continuité, et à la mesure du voltage dans le circuit. Celui-ci étant assez complexe et grand, il a fallu que nous soyons deux pour exécuter cette tâche correctement. Cela a pris pas mal de temps. Il a fallu tester 60 solénoïdes, et le reste du circuit. Aucun problème a signalé de ce côté-là, ce qui est à la fois rassurant et problématique, étant donné qu’on pensait que l’erreur venait des branchements. Mais d’un autre côté, désormais, on sait que l’erreur ne vient pas de là.

C’est en toute fin de séance que nous somme parvenu a correctement initialisé les solénoïdes. Suite à l’utilisation d’un nouveau code, et a des tests au niveau du branchement de la cloc et de la pin des data.

Nous ne sommes pas encore sûr que cette initialisation marche, car les mêmes programmes dans le setup et dans le loop, ne donne pas le même résultat. Une chose est sûre, il faut réécrire le code nous-même et de manière différente si l’on veut de meilleur résultats. On a donc pas mal d’heures de code devant nous.