FOYANG Joël

KHA Alexandre

TD7

Rapport du mini-projet Arduino

1. INTRODUCTION

L'objectif du projet consiste à illustrer l'utilisation d'une communication Li-Fi (Light Fidelity), qui est un type de communication sans fil reposant sur l'émission et la réception de signaux du spectre visible.

Plus précisément, il s'agit de convertir une chaîne de caractères saisie dans le monieur série d'une carte Arduino en signal lumineux, puis de réceptionner ce signal via un photorécepteur et en faire la conversion inverse pour écrire la chaîne obtenue sur un écran LCD.

Si la conversion de la chaîne en signal lumineux ainsi que la conversion inverse sont fonctionnelles, la chaîne de caractères inscrite sur l'écran LCD correspond exactement à celle saisie dans le moniteur série.

Pour mieux constater le fonctionnement ou non du programme, on choisit d'écrire sur la ligne du haut de l'écran LCD la chaîne transmise, puis sur la ligne du bas la chaîne réceptionnée.

2. LA MODULATION PPM

La première étape qui consiste à transformer la chaîne de caractères saisie en signal lumineux est la modulation. La deuxième qui consiste à transformer ce signal lumineux en chaîne est la démodulation.

Plusieurs types de modulation existent et celle que nous choisissons pour mener à bien notre projet est la PPM (Pulse Position Modulation).

Celle-ci consiste à considérer le code ASCII de chaque caractère de la chaîne saisie.

A partir du moment où un caractère est lu, on fait allumer la led rouge pendant une durée Δt que l'on choisit soi-même, puis on l'éteint pendant une durée proportionnelle à son code ASCII, pour se rallumer ensuite lorsqu'un autre caractère est lu (cela peut être le retour chariot).

Par exemple le caractère « a » de code ASCII 97 est lu à t_0. Jusqu'à $t_0 + \Delta t$ la led rouge est allumée. Puis de ($t_0 + \Delta t$)à $(t_0 + \Delta t + 97\alpha)$ (avec α un coefficient que l'on fixe soi-même, de façon à ce que l'on puisse différencier les deux précédents temps) la led rouge est éteinte. Ensuite un autre caractère est lu et ceci marque le temps t_1, qui coïncide avec $t_0 + \Delta t + 97\alpha$, puisque le code Arduino tourne en boucle.

Pour démoduler le signal lumineux ainsi produit, il s'agit alors de retrouver le code ASCII contenu par le signal pour retrouver le caractère correspondant. Dans le cas précédent, on voit que ce caractère est obtenable en calculant $\frac{t_1-t_0-\Delta t}{\alpha}$ et en en prenant le char, ce qui est généralisable à tout caractère saisi.

3. INTERRUPTION POUR LA DEMODULATION

La fonction d'interruption centrale du projet, nommée **photo_mesure** , est liée à la réception d'un signal lumineux, et fait passer la variable booléenne *signal_recu* à *true*.

Préalablement on aura associé respectivement aux bornes AIN+ et AIN- du comparateur interne, la tension de référence INTERNAL_REFERENCE (de 1.1V) et l'entrée A0 qui est liée à la photorésistance. L'évènement qui déclenche l'interruption est l'augmentation de la tension à travers la photorésistance, donc le passage de la tension de référence en dessous de la tension captée en A0. Ceci qui correspond à un évènement de type FALLING.

Le code de la fonction **photo_mesure** a été réduit en taille afin d'optimiser sa durée d'exécution. En changeant le variable booléenne *signal_recu* en *true*, elle induit la lecture de la condition

if (signal_recu) dans le loop. Celle-ci permet l'affichage du caractère lu sur la ligne du haut puis celle du caractère reçu juste en dessous, le positionnement en colonne étant indiqué par *pos_char*.

A la fin de la boucle, on modifie *signal_recu* en *false*, pour permettre de détecter les prochaines interruptions.

4. TESTS PRELIMINAIRES

Tests :

Test 1 : Nous faisons varier le *delta_t* puis le coefficient de proportionnalité devant l'ASCII des caractères
Test 2 : Fonction *photo_mesure* « alourdie » avec *t1 = millis(),* puis « légère », avec *t1 = millis()* dans le loop

Résultats :

Test 1 : Ajouter juste l'ASCII d'un caractère à une durée (unité en ms) n'est pas suffisant, on remarque que la modulation est d'autant plus précise que les temps mis en jeu sont longs. Au final on retient un *delta_t = 400* et un coefficient de proportionnalité de 4 pour l'ASCII (la durée d'extinction de la led rouge est donc 4*ASCII), ce qui donne un certain temps d'exécution, sans être excessif.
Test 2 : La fonction photo_mesure avec un *t1 = millis()* a un temps de d'exécution non négligeable ce qui fausse ensuite les différences de durée mesurées. On préfère alors coder dans la condition *if (led_rouge_alu)* le *t1 = millis()*.

5. EFFACEMENT DE L'ECRAN

C'est la fonction d'interruption **Clear**, associée au bouton poussoir, qui permet d'effacer les caractères inscrits sur l'écran LCD et de saisir ensuite de nouveaux caractères.

Elle appelle *lcd.clear()* pour effacer l'écran, puis réinitialise la colonne sur laquelle se trouve le caractère lu à 0 avec *char_pos = 0* ainsi que le nombre de caractères lus à 0 avec *nb_char_read = 0.*

L'évènement qui produit l'interruption est l'appui sur le bouton poussoir, ce qui est caractérisé par RISING.

6. DETECTION DE L'INACTIVITE ET ALLUMAGE DE LA LED VERTE

Le critère déterminant l'inactivité est l'absence de caractère saisi dans le port série, ce qui correspond encore à *Serial.read() == 0*. Ainsi, dans une condition *if (Serial.read() == 0),* on code l'allumage de la led verte.

On réinitialise également le nombre de caractères lus *nb_car_read* à 0, afin que si on le souhaite, on puisse saisir une autre chaîne après un retour chariot.

7. CONCLUSION

La modulation PPM semble relativement aisée à coder puisqu'il s'agit de considérer le code ASCII de caractères, de leur affecter un certain coefficient pour les retranscrire en durées d'extinction de led, puis faire une opération inverse à partir des signaux reçus pour retrouver leur code ASCII.

On note également que le programme fonctionne sur le simulateur ainsi que sur la carte réelle.

Cependant on remarquera que cette méthode nécessite une certaine précision dans la mesure de temps. C'est pour cela que l'on a choisi de « dilater » les durées en choisissant un facteur 4 associé à l'ASCII des caractères, permettant d'avoir des durées longues donc mesurables. Mais alors la modulation et la démodulation prennent un certain temps. Si on considère qu'en moyenne un caractère possède un ASCII de 60, en saisissant 16 caractères et en se donnant un *delta_t* de 400ms, cela donne une durée d'environ 10s pour l'affichage complet. On préférera peut-être d'autres types de modulation pour communiquer à grande vitesse.