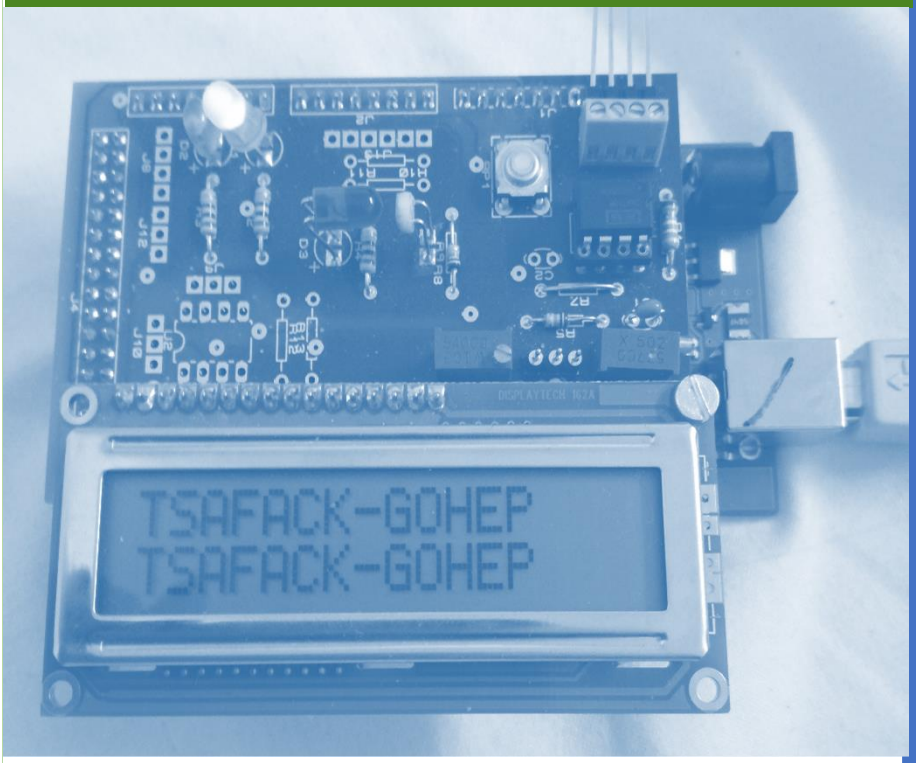
ENSTA Bretagne

2016-2017

UV 2.5 : Mini projet

Réalisation d'une communication LI-FI



❖*Jordane TSAFACK*
❖*Maurelle GOHEP*

Encadrant : B. Zerr

Description de la méthode d'effacement de l'écran LCD avec l'appui sur le Bouton poussoir

L'appui sur le bouton poussoir fait varier l'état de l'entrée numérique (bp1 (IO18)de LOW à HIGH) ce qui déclenche une interruption. Nous associons à cette interruption une fonction qui sera excutée via l'instruction attachInterrupt.

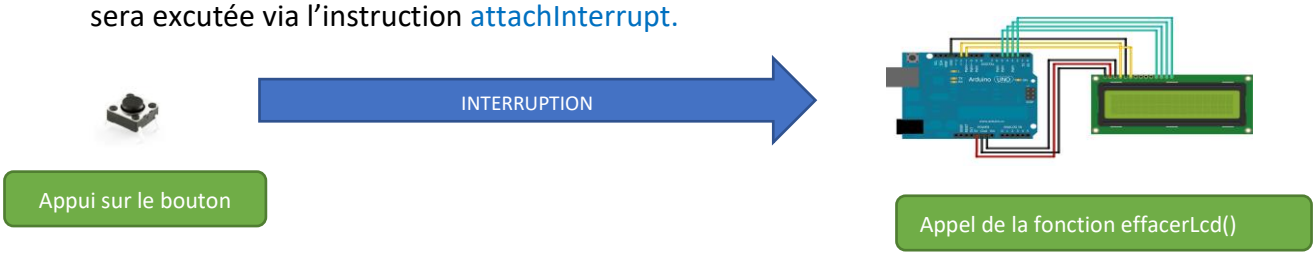


Figure1: effacement de l'écran LCD

Code :

```
Void setup () {
attachInterrupt(digitalPinToInterrupt(bp1), effacerLcd, RISING) ;//
}
void effacerLcd () {
  lcd. clear() ;
 colonneLigne1=colonneLigne0=0 ;/* on remet la position du curseur du curseur de la première et deuxième ligne à 0 */
 }
```

Modulation PCM (Pulse Code Modulation)

La modulation PCM (Pulse Code Modulation) est une modulation numérique a` deux codes, un pour le 0 et un pour le 1. Cette modulation ne peut coder le caractère en seul coup et il faudra coder, un après l'autre, les 8 bits du caractère. Pour le caractère **'A'**, le code binaire est **01000001**. Si on choisit arbitrairement de commencer par le bit de poids faible, il faudra coder un 1, puis un 0, puis un 0 ... et finir par un 0. Les impulsions sont toutes de durées identiques. Comme la montre la figure 2, il suffit de définir deux durées **T0(dans notre cas 160ms)** et **T1(80 ms)**. Si le bit est à 1, la durée entre impulsions est **T1** et si le bit est à 0 la durée est **T0**. La durée **Tsep**, supérieure à T**0** et T1, permet d'indiquer la fin d'un caractère.

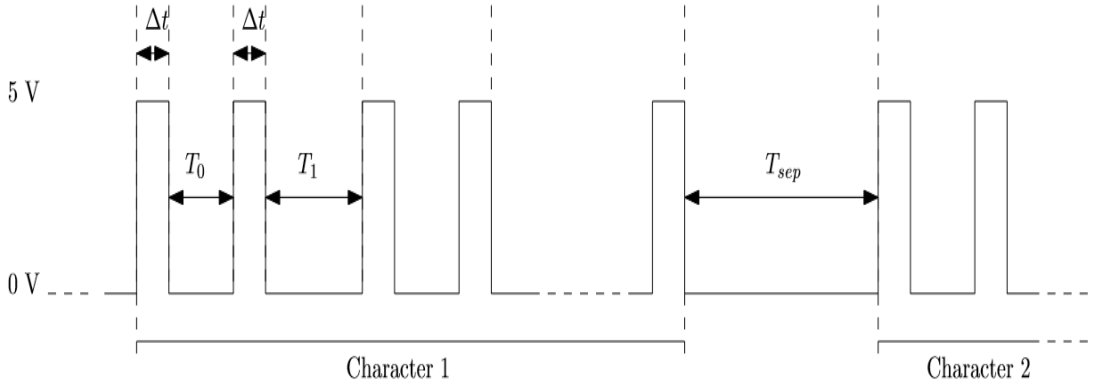


Figure 2 : Modulation PCM (Pulse Code Modulation)

Comparaison de la modulation PCM par rapport à la modulation PPM :

Méthode	PCM	PPM
Fiabilité	Très fiable (possibilité de détecter et corriger les erreurs avec **crc**)	Peu fiable (impossible de détecter des erreurs)
Bande passante	Faible	Large
Débit	Faible	Bon
Vitesse	Lent	Rapide
Implémentation	Difficile	Simple
Application	Numérique	Analogique

Modulation lumineuse

Le code ascii du caractère est obtenu en « transtypant » la valeur du caractère dans un entier (int code=char). Ainsi, chaque bit du code est ensuite récupéré en faisant des et logique (&) avec un masque (1<<i) où i est la position du bit. Chaque bit est transformé en impulsion suivant le principe suivant :

- **1**

On initialise la communication en allumant la led rouge pendant une durée **thigh**=15ms

Puis on l'éteint pendant une durée t1= 80ms

Enfin on allume la led rouge pendant une durée **tHigh=15ms** pour marquer la fin de la transmission

- **0**

On initialise la communication en allumant la led rouge pendant une durée **thigh**=20ms

Puis on l'éteint pendant une durée t1= 160ms

Enfin on allume la led rouge pendant une durée **tHigh=15ms** pour marquer la fin de la transmission

Démodulation

La lecture se fait en interruption avec **TimerOne** qui déclenche à une période de **10 ms,** l'exécution de la fonction **demodulation** qui lit la valeur de l'entrée analogique : **photoR** (A0) la valeur obtenue(lum) est comparée à la valeur **offset (off)** déterminée dans la fonction **setup ()** lors de l'initialisation de la carte ; si **(lum-loff) <-100** on détecte que la led rouge est allumée. Un test sur la valeur du **timer** suivant qu'elle soit dans une plage :

(timer>=-5+temp1-2*tempsHIgh&& timer<temp1+tempsHIgh*2+5) correspond à un 1

(timer>=-5+temp0-2*tempsHIgh-10&& timer<temp0+tempsHIgh*2+5) correspond à un 0 on déduit qu'il agit d'un 0 ou 1.

Au cas où il s'agit d'un 1, une variable permettant de reconstruire le code ASCII est mise à jour (**codeAscii+=1<<compteurDebit**) une seconde variable compteurDebit qui compte le nombre de bits reçus est aussi mise à jour. Si on atteint 8 bits (compteurDebit==8) on peut décoder le code (**char c=codeAscci**) et remettre le compteur à 0 et la variable **codeAsciirecu** à 0.

Tests préliminaires

Nous avons commencé par déterminer les valeurs offset (quand la led rouge est éteinte) de la photorésistance et la valeur de saturation quand la led est allumée. Puis nous avons essayé de détecter si la led rouge est allumée ou pas. Nous avons d'abord commencé par envoyer des 0, et 1 puis des entiers quelconques. Nous avons écrit les fonctions pour l'envoi d'un caractère et sa réception, puis nous avons essayé d'ajuster les valeurs du **tempsHigh** (durée de la led rouge allumée) et t0, et t1 afin de pourvoir distinguer la réception des 0 et celle des 1 avec précision. Puis nous avons essayé d'optimiser le temps de communication.

Détection de l'inactivité (permettant l'allumage de la LED verte)

La réception se faisant instantanément, il suffit que la led verte soit éteinte lorsqu'une donnée est disponible sur le port série ; une fois la modulation achevée, la led verte est rallumée.

```
while (Serial.available()) {
 digitalWrite (ledVerte, LOW);
……//modulation
digitalWrite (ledVerte, HIGH);
}
```

Conclusion

Nous avons pu mettre sur pied notre communication Li-fi en modulant grâce à la méthode du PCM l'information sous-forme d'impulsion lumineuse. Toutefois, la led rouge et le récepteur doivent être couverts pour éliminer les lumières parasites. Notamment celle produite par la lumière verte. De même, nous restons critiques sur le débit maximal de transmission qui contrairement à nos attentes est très faible, cela serait dû aux capacités limitées des composants (temps de réponse de la photorésistance, luminosité de la led). Nous aurions aimé tester la communication entre deux cartes afin de donner au projet une dimension plus réaliste.