# Introduction

Dans le cadre du Summer School 2, nous devons réaliser un Pong. Le but est de le créer entièrement, c’est-à-dire que nous devons concevoir la schématique de l’électronique, faire le routage, concevoir le PCB, monter et souder les composants ainsi que de concevoir le code complet du jeu.

La plaque électronique doit être conçue comme un jeu portable. Le circuit sera alimenté par deux piles AAA. Un des enjeux de ce projet est de réussir à obtenir une durée de vie la plus longue possible, de ce fait, nous devons optimiser la consommation de courant et gérer le système de mise en veille automatique.

Nous allons donc d’abord vous expliquer en détail le cahier des charges, puis nous vous présenterons notre architecture matérielle ainsi que l’implémentation du routage. Enfin, nous vous détaillerons le fonctionnement logiciel de notre jeu à l’aide de diagramme UML.

# Cahier des charges

Les règles du jeu sont assez simples. Le joueur devra jouer contre l’ordinateur. Une option de départ permet de choisir le niveau de difficulté du jeu. En fonction de cette difficulté, la vitesse de la balle augmentera plus rapidement, ajouter autres difficultés ajoutées. La vitesse de réaction de l’ordinateur est constante, ainsi, après un certain temps de jeu, l’ordinateur sera plus lent que la balle et il sera donc possible de gagner.

Dans les contraintes matérielles, nous devons utiliser un écran tactile LCD, de type NHD. Le microcontrôleur est un PIC18LF25K22. Nous devons utiliser des supports de piles AAA pour la partie alimentation.

Concernant la conception, nous utiliserons le logiciel Altium Designer pour le développement hardware et MPLAB X pour l’environnement logiciel du microcontrôleur. Le langage de programmation utilisé est le C.

# Schéma bloc

Figure  : Schéma bloc

Comme on peut le voir ci-dessus, le schéma bloc de notre projet est relativement simple.

Le premier bloc concerne la partie alimentation, dont nos 2 piles ainsi qu’un MOSFET permettant de protéger le circuit contre une potentielle inversion de polarité. Ainsi, les deux piles AAA vont venir alimenter la plaque avec du +3V.

La partie PIC permettra de gérer la logique d’affichage de l’écran, ainsi qu’à contrôler les MOSFET pour venir éteindre l’écran LCD et le rétroéclairage, afin d’économiser de l’énergie.

# Conception

Dans cette partie, nous allons aborder la conception l’architecture matérielle, ainsi que la schématique et l’implémentation du PCB. Vous trouverez en annexe le schéma complet de notre circuit et le routage.

Notre architecture matérielle a été pensée pour que la prise en main de l’utilisateur soit la plus agréable possible. Les piles et supports de piles sont placées sur le bottom du PCB, car nous souhaitons imprimer en 3D un boitier qui viendra couvrir la partie arrière du circuit pour pouvoir protéger les composants et d’offrir une bonne prise en main.

La schématique contient un système d’anti-inversion de polarité ainsi que d’interrupteur On/Off permettant d’éteindre l’alimentation de l’écran LCD que nous allons détailler maintenant.

## Anti-inversion de polarité

Figure Sécurité d'inversion de polarité

Notre carte est équipée d’un système empêchant tout problème provenant d’une inversion de polarité lors de l’installation des piles.

Cette partie fonctionne grâce à un simple MOSFET channel N, ainsi, si les piles sont branchées à l’envers, ceci nous permet de garder l’alimentation à 0V au lieu de -3V, ce qui risquerait d’endommager nos composants. Il est important de choisir un MOSFET ayant une résistance Rdson la plus faible possible, pour ainsi limiter toute consommation excessive de courant pour obtenir une durée de vie des batteries la plus longue possible.

Nous avons mis des pointes de tests pour simplifier la mesure lors des tests et de la mise en service.

## ON/OFF Alim

Figure On/Off alimentation

Toujours dans l’optique d’économiser l’énergie, notre circuit possède un système de « veille ». C’est-à-dire que si l’utilisateur n’interagie plus avec l’écran LCD pendant un certain temps, le PIC va décider de couper l’éclairage de l’écran automatiquement.

Ce système est réalisé à l’aide d’un MOSFET channel N, qui fonctionne comme un interrupteur. Dès lors que l’on envoie un signal de 3V (label « BackLightCmd » sur le schéma), le MOSFET va commuter et ainsi alimenter les LEDs. Cela nous permet de gérer l’alimentation du rétroéclairage de l’écran, car c’est cette partie qui consomme le plus dans notre circuit, mais aussi de couper l’alimentation générale de tout l’écran LCD.

## Pin LCD

Figure 4 On/Off alim LCD

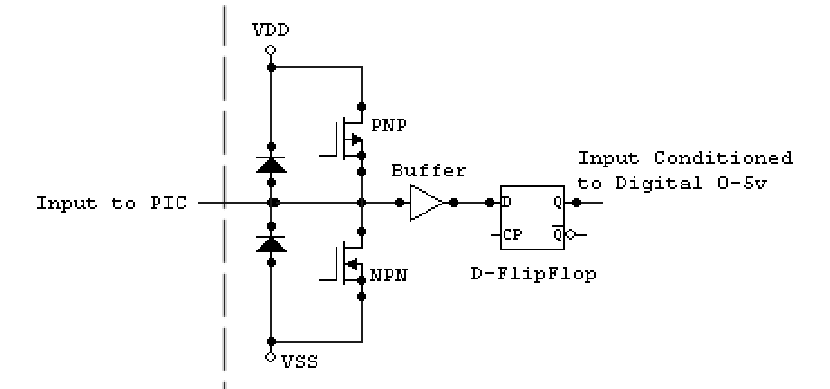
Sur la figure 4, on peut voir le montage avec MOSFET qui vient couper le GND de l’écran LCD pour pouvoir l’éteindre quand on le décide.

Figure Internal Port

Pour pouvoir couper l’alim du LCD, il est important que toutes les PINS reliés au PIC soient mis au VCC. Etant donné que l’on vient couper le GND, si d’autres PINS connectées directement au PIC sont au GND, cela peut causer une alimentation indirecte via ces PINS dû à la structure interne comme représenté sur la figure 5.

# Fonctionnalités du système

## Diagramme de cas d’utilisation

Insérer diagrame use case

Ci-dessus, voici le diagramme de cas d’utilisation de notre logiciel. Comme on peut le voir, il y a 3 parties distinctes.

En premier lieu, lorsqu’on vient allumer l’écran LCD, un menu principal s’affiche. Sur ce menu on peut choisir de jouer en solo, à deux, ou d’aller dans les paramètres. Dans le menu paramètre, on peut venir choisir la luminosité grâce à un slider ainsi que la difficulté du niveau que l’on souhaite jouer. Puis on peut revenir en arrière au menu principal.

Lorsque l’on rentre dans la partie jeu, on peut jouer contre l’ordinateur si on a choisi une partie en solo, ou alors contre un autre joueur si on a choisi la partie multijoueur. La seule action possible pour le joueur est de déplacer le paddle à gauche ou à droite. L’ordinateur va venir s’occuper du déplacement de la balle ainsi que le déplacement du second paddle en partie solo.

## Diagramme de séquence

Insérer diagramme de séquence

# Architecture logicielle

# Tests et mise en service

# Conclusion