Projet: Programmation Orientée Objet
Le Damier

Jeunet Maxime
Bouyer Hugo
Ordi Axel
4AE SE

Introduction

Notre projet est un jeu de Dames sans pion.
La problématique était de localiser les “pions” sur un plateau sans que ceux-ci soient physiquement présents. Pour y répondre nous avons décidé de représenter les pions par des couleurs sur une matrice de LED.
Pour pouvoir accepter les entrées de coordonnées des joueurs, une matrice de boutons superposée à la matrice de LED est utilisée. Cela nous permet de récupérer les coordonnées en 2D des instructions des joueurs.
Sur un damier 10x10 cases, le jeu de dame se jouant en diagonale, seules les cases de couleur blanche ou noir sont nécessaires.
On a ainsi créé un damier avec 50 cases (case blanche) avec un bouton et des LED et 50 cases qui sont inactives (case noir).

Le Damier

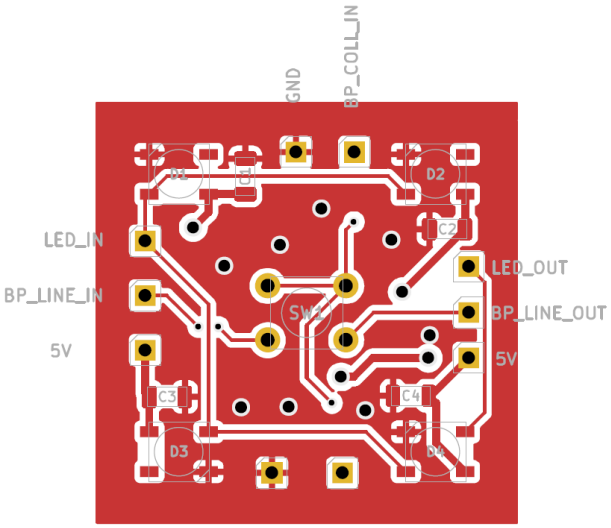
Les PCB

Pour réaliser le damier nous avons décidé de créer 2 types de PCB:

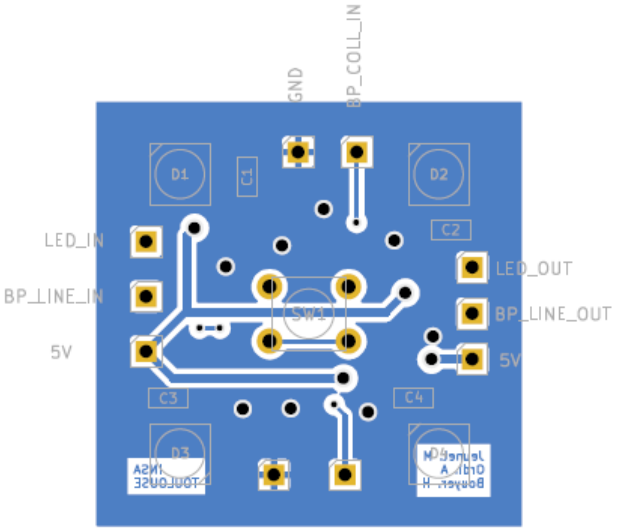
- PCB Slave

Ce PCB contient 4 LED en parallèle ainsi qu'un bouton.

Couche TOP



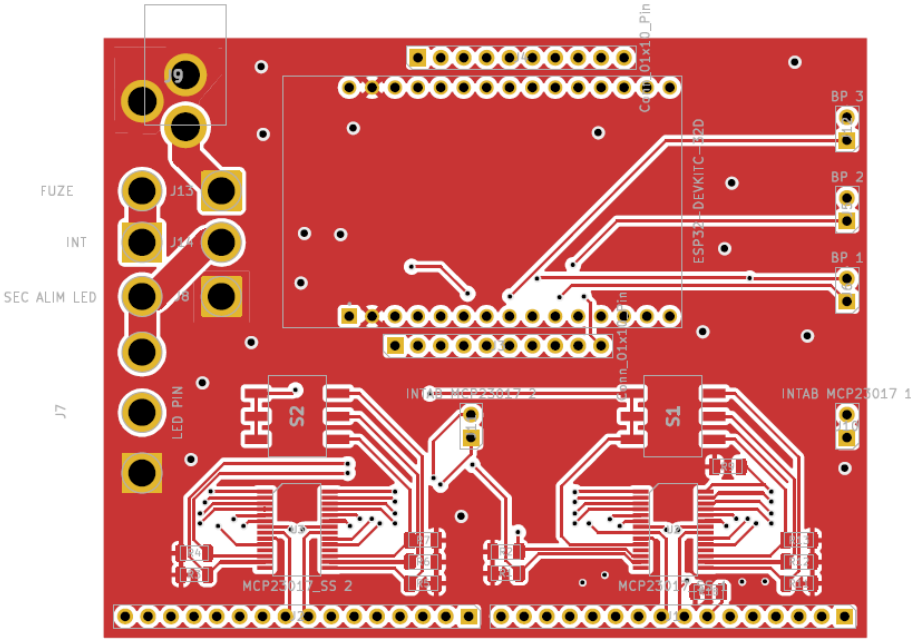
Couche BOTTOM



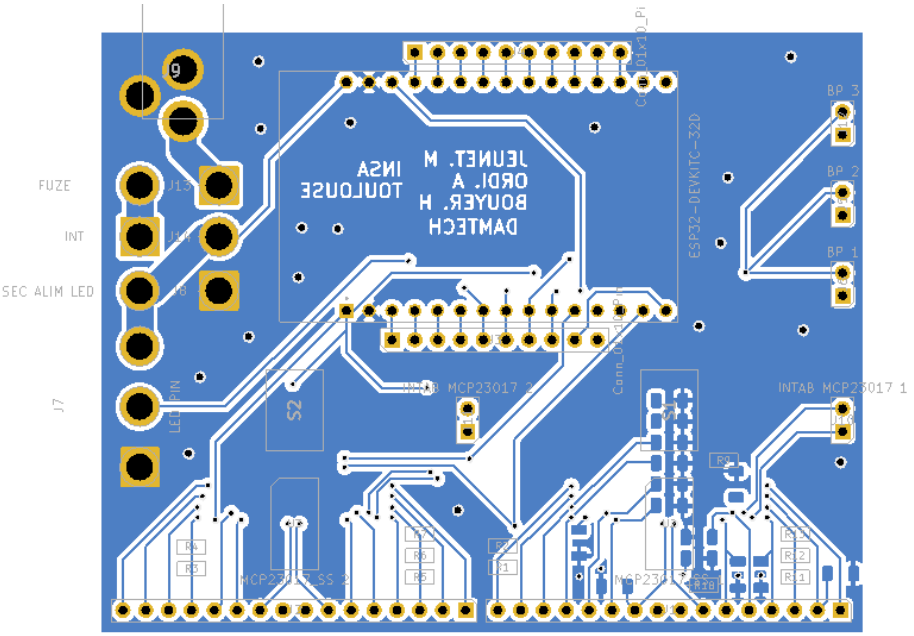
- PCB Maître

Ce PCB permet de traiter l'information et de recevoir l'alimentation.
Pour cela il contient l'ESP32 ainsi que deux multiplexeurs MCP23017.

Couche TOP



Couche BOTTOM



Le fonctionnement

- Le fonctionnement des LED:

Chaque PCB_Slave contient 4 LED en parallèle. C'est-à-dire que ces 4 LED reçoivent le même Data_IN, une seule renvoie le Data_OUT vers la case suivante. Cela veut dire que l'on a seulement 50 instructions pour contrôler les 200 LED.
La couleur des LED est gérée par l'intermédiaire de la librairie Adafruit_NeoPixel.

- Le fonctionnement de la matrice de bouton:

La matrice de bouton sert à recueillir les coordonnées des déplacements des joueurs.
Chaque PCB slave contient un bouton relié en série sur la colonne et la ligne.
Pour obtenir l'information d'appui sur un bouton on se sert de deux multiplexeurs. Un multiplexeur (Multiplexeur_ligne) va régulièrement lire l'information sur les lignes. Le second (Multiplexeur_colonne) va écrire séquentiellement un "1" logique sur chaque colonne. On peut ainsi repérer un bouton poussé si lorsque le Multiplexeur_colonne met à 1 une colonne on peut lire une ligne mise à 1 sur le Multiplexeur_ligne.

Le Jeu de Dame

Pour effectuer le jeu de Dames nous avons décidé de représenter chaque composant du jeu par une classe. Le Diagramme de classe en annexe décrit la relation entre les différentes classes créées.
La logique derrière la formulation de ces classes étaient:

- Un jeu de Dames nécessite un plateau.
- Un plateau est composé de 100 cases.
- Ces cases peuvent avoir un pion noir ou blanc situés dessus.

Avec ces objets on peut ainsi formuler la logique du jeu de Dames.
Lorsqu'on appelle la méthode de jeu de Dames ***Commencer_Partie_Jeu_De_Dame()*** on crée un objet Plateau contenant 100 objets Case. Ces Cases sont ensuite initialisées avec les Pion Blanc et Noir en position de début de partie.
Le reste de la logique de jeu se fait dans la méthode ***Deroulement_Tour()***. Cette méthode met en place une machine à état qui permet le déroulement du jeu jusqu'à la victoire d'un des joueurs.
La machine à état de cette méthode est présentée en annexe.

Cette machine à état est composé de 8 états:

- Etat 1 → ***Etat_Attente_1 ()*** : On attend que le joueur appuie sur une case.
- Etat 2 → ***Verif_Couleur_Pion()*** : On vérifie que la case sélectionnée contient un pion de la couleur du joueur en cours de jeu.
- Etat 3 → ***Etat_Attente_2()*** : On attend que le joueur sélectionne une seconde case, ce seront les coordonnées d'arrivée si la case est vide. Si la case contient un pion, on retourne à l'état 1.
- Etat 4 → ***Verif_Deplacement_Possible()*** : Dans cet état on vérifie si le déplacement demandé par le joueur est possible. Il envoie ensuite vers deux états différents, soit un déplacement simple(État 5). Soit l'action de manger un pion(État 6).
- Etat 5 → ***Deplacer_Pion()*** : On déplace le pion en passant sa référence à la case de destination et en supprimant cette référence de la case initiale. On met aussi à jour les LED.
- Etat 6 → ***Manger_Pion()*** : Cet état permet de faire un saut de 2 cases en passant au dessus d'un pion de couleur opposé. Le pion mangé est supprimé ce qui décrémente une variable de classe (NbPionBlanc ou NbPionNoir), ceci nous permet de suivre la condition de fin de jeu.
- Etat 7 → ***Verif_Deplacement_Manger_Pion()*** : Si un autre pion peut être mangé après le 1er déplacement alors on est ramené à l'état 6.
- Etat 8 → ***Fin_du_Tour()*** : On met à jour le booléen tour qui permet de changer de joueur. On repart à l'état 1.

La condition actuelle de fin de jeu est la destruction de l'ensemble des pions d'une couleur.

Conclusion

Buts Atteints

Nous avons réussi à implémenter le détecteur de position de boutons appuyer avec des sécurités comme par exemple si on appuie simultanément sur 2 cases.

Nous avons réussi à commander les led en fonction de la couleur de la case correspondante.

Problèmes rencontrés

Suite à plusieurs jours de retard de la fabrication des PCB Slave, nous n'avons pas pu assemblé l'entièreté de notre damier et n'avons pas pu tester son bon fonctionnement.

Perspectives d'évolutions

Avec plus de temps, créer un outil de débogage qui nous permettrait d'être indépendant de la fabrication du damier.

L'ESP32 nous permettant d'être connecté au Wi-Fi et donc à Internet, on pourrait jouer en réseau et en physique simultanément (***chiche***).