POLYTECH
NICE-SOPHIA

La Poly Boy

Une console portable basée sur Arduino Due.

Rapport de Projet Arduino

Étudiants :
Joao BRILHANTE
Olivier DOUSSAUD

Enseignants :
Pascal MASSON
Nassim ABDERRAHMANE

Mars 2019

Université Nice Sophia Antipolis

Remerciements

Nous tenons à remercier tout particulièrement Monsieur Marek Buriak, l'auteur de la bibliothèque d'affichage *ILI9341_due*, sans quoi le projet n'aurait pu avancer aussi vite. De plus, un grand merci à toute l'équipe de *VitaSDK*, un kit de développement pour la *PS Vita*, une console portable de *Sony*. Enfin, nous souhaitons remercier Messieurs Pascal Masson et Nassim Abderrahmane pour ce cours d'électronique sur Arduino qui nous a permis de mettre en pratique nos connaissances.

Sommaire

1 Introduction

1.1 Présentation

Ce projet intitulé *Poly Boy*, en référence à la célèbre console portable de *Nintendo*, est une console portable basée sur *Arduino Due*. En effet, tous deux passionnés de jeux vidéo et de développement, nous avons décidé d'offrir la meilleure expérience de jeu possible sur ce support. La console est notamment équipée d'un lecteur de carte *micro SD* permettant au joueur de télécharger tous les jeux développés pour cet environnement. Ainsi, le joueur a la possibilité de redécouvrir ses jeux préférés sur une console portable ergonomique et économique. Enfin, une communication réseau permet à ce dernier de profiter d'une aventure en multi-joueurs quelle que soit la distance.

1.2 Objectifs

Nous tenions à offrir la meilleure expérience de jeu possible et à nous démarquer de nos prédécesseurs. Pour cela, nous avons dressé une liste des objectifs primordiaux à atteindre :

- Une console portable basée sur Arduino Due.
- Un affichage rapide et fluide pour une bonne immersion.
- Une communication *Wi-Fi* pour accéder aux meilleurs scores et au multi-joueurs.
- Des boutons à membrane plus confortables et réactifs que des boutons poussoirs.
- Un boitier solide et élégant permettant une bonne prise en main de la console.
- Un lecteur de *carte SD* pour le stockage des jeux et des ressources nécessaires.
- Un port *micro USB* permettant la programmation de la console à tout instant.
- Un gyroscope pour les jeux nécessitant une capture des rotations et du mouvement.
- Un son mono pour accentuer l'immersion et la nostalgie.
- Une interface de lancement des jeux et applications depuis la *carte SD*.

2 Matériel utilisé

2.1 Écran TFT

Notre première priorité a été de trouver un écran ayant une fréquence de rafraîchissement assez grande. En effet, dans le cadre des jeux vidéo, la fréquence de rafraîchissement est primordiale pour un affichage correct et des transitions fluides.

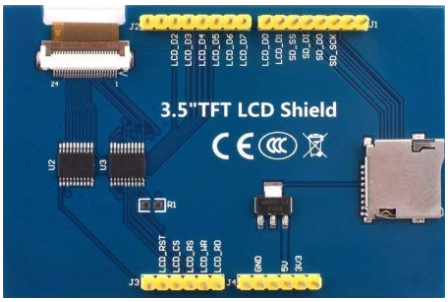


Figure 1: Ecran TFT à 8 bits en parallèle

Nous avons tout d'abord opté pour un écran *TFT* (*Thin-Film Transistor*) ayant 5 pins de contrôle, 8 pins de communication et un contrôleur *ILI9486*. En effet, nous pensions, à prime abord, que le nombre de pins de communication impacterait grandement la fréquence de rafraîchissement de l'écran. Cependant, après quelques tests, nous avons constaté que l'écran pouvait prendre jusqu'à deux minutes pour rafraîchir individuellement chaque pixel avec une couleur aléatoire.

Figure 2: Ecran TFT à liaison SPI

Nous avons finalement opté pour un écran *TFT* à liaison *SPI* (*Serial Peripheral Interface*) qui n'utilise que 3 pins de contrôle, 3 pins de communication et un contrôleur *ILI9341*. L'avantage de la liaison *SPI* est son débit de données bien plus important notamment en mode *DMA* (*Direct Memory Access*).

2.2 Carte SD

Une de nos priorités a également été de trouver une solution au stockage des ressources des différents jeux et applications de la console. En effet, les données de jeux peuvent être assez gourmandes en mémoire.

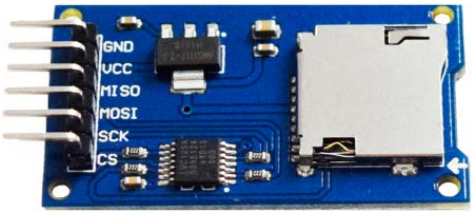


Figure 3: Module de carte *micro SD*

Nous décidons d'utiliser pour cela un module de carte *micro SD* à liaison *SPI*. En effet, les mémoires *SRAM* et *FLASH* offertes par l'*Arduino Due* ne sont pas suffisantes pour stocker toutes les ressources. De plus, cela nous permettra de stocker les ressources de tous les jeux sans avoir à reflasher l'*Arduino Due*.

Figure 4: Carte micro *SDHC* de 16 Go

Figure 5: Carte micro *SDHC* de 4 Go

Dans un premier temps, nous essayons une carte *micro SDHC* de 16 Go et nous constatons que sa fréquence de fonctionnement est très basse. En effet, il semblerait que la fréquence de fonctionnement d'une carte *micro SD* soit d'autant plus grande que sa capacité est faible. Nous décidons alors d'utiliser une carte *micro SDHC* de 4 Go et de classe 10 pour laquelle nous trouvons une fréquence de fonctionnement plus raisonnable.

2.3 Boutons

Dans l'optique d'un meilleur confort de jeu, nous avons décidé d'utiliser des boutons à membrane au lieu de boutons poussoirs. Nous devons alors chercher les boutons, les membranes en caoutchouc et le circuit imprimé permettant leur fonctionnement. Malheureusement, les boutons à membrane sont souvent vendus sans le circuit imprimé, pour permettre la réparation de vieilles manettes.

Figure 6: Manette de *SNES*

Figure 7: Composants de la manette

Nous choisissons alors une option plus simple qui consiste à réutiliser les composants d'une manette filaire de *SNES*. Nous démontons la manette et récupérons les boutons, les membranes et le circuit imprimé.

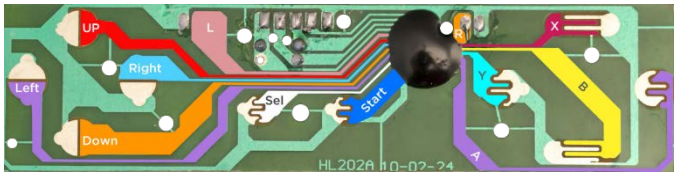


Figure 8: Circuit imprimé de la manette

Nous découpons ensuite le circuit imprimé en cinq morceaux :

- Un morceau pour la croix directionnelle.
- Un morceau pour les boutons *Start* et *Select*.
- Un morceau pour les boutons *A, B, X* et *Y*.
- Deux morceaux pour les gâchettes gauche et droite.

En effet, cela permettra de disposer les boutons plus aisément ultérieurement. Enfin, nous soudons fils sur les contacts des circuits imprimés afin de détecter la pression de chaque bouton à l'aide d'une résistance de tirage interne à l'*Arduino Due*.

2.4 ESP8266

L'*ESP8266* est un microcontrôleur développé par le fabricant chinois *Espressif*. Cette puce possède en outre l'avantage d'avoir un prix plus abordable que son équivalent Arduino (moins de 2 euros) et permet de communiquer en *Wi-Fi*. Cette-dernière ne s'est démocratisée que récemment après de la traduction de la documentation du chinois à l'anglais.

Figure 9: ESP8266 MiniDK avec écran OLED

Notons que le même fabricant a également développé de nouvelle puces, telles que l'*ESP32*, qui en plus de la communication *Wi-Fi* permettent une communication *Bluetooth*. Pour notre projet, nous choisissons l'*ESP8266* puisque nous n'aurons besoin que de la communication *Wi-Fi*.

2.5 Boitier

Nous choisissons de faire imprimer en 3D le boitier de notre console. Pour cela, nous devions nous occuper au plus tôt de la modélisation 3D du boitier. Nous utilisons alors le logiciel *SolidWorks* pour modéliser l'intégralité des composants avec une grande précision pour un résultat au plus proche de la réalité.

Figure 10: Modélisation 3D du boitier

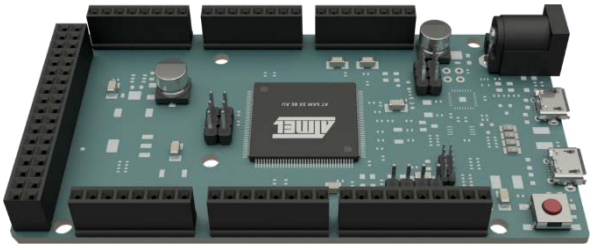


Figure 11: Modélisation 3D de l'*Arduino Due*

De plus, il est nécessaire de réfléchir à la meilleure disposition des composants pour la miniaturisation et l'ergonomie de la console. Nous n'avons pas beaucoup de temps pour réaliser des prototypes et d'espace pour les composants. Il s'agit alors de laisser une certaine marge à l'impression 3D pour mieux corriger et emboîter les deux parties du boitier a posteriori.

Figure 12: Boitier de la console

Une fois ce long processus terminé, nous contactons le *SoFab*, un atelier partagé de fabrication situé sur le campus *SophiaTech*, qui accepte notre projet d'impression 3D. L'impression 3D reste plutôt correcte cependant elle manque cruellement de précision. Nous nous occupons alors des finitions du boitier.

3 Fonctionnement et schémas

3.1 Affichage

Après quelques recherches, nous trouvons une bibliothèque *ILI9341_due* très complète conçue pour notre contrôleur d'écran et pour *Arduino Due*. Cette bibliothèque correspond parfaitement à notre projet et implémente d'ores et déjà des fonctions de communication *SPI* et d'affichage indispensables.

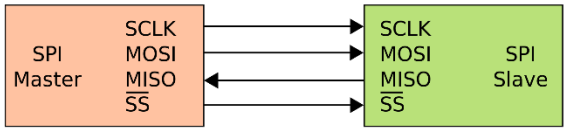


Figure 13: Schéma d'une liaison *SPI*

Nous parvenons très rapidement à afficher des textes, des formes et des images et à les animer. Cependant, nous avons toujours affaire à quelques artéfacts d'affichage qui sont dus à la fréquence de rafraîchissement de l'écran que nous devons prendre en compte.

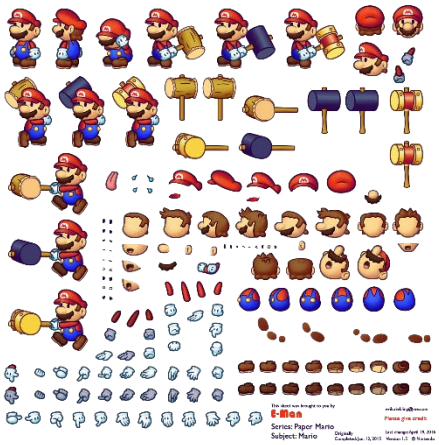


Figure 14: Une feuille de sprites

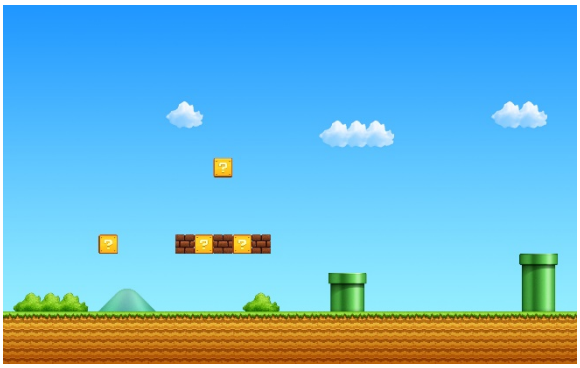


Figure 15: Une image de fond

Nous imaginons ensuite une bibliothèque permettant l'implémentation d'autres fonctions utiles et une initialisation plus simple de l'écran. En effet, lors du développement des jeux, nous aurons besoin de pouvoir afficher des images avec ou sans transparence. De plus, nous ajoutons la gestion de feuille de sprites, des tableaux de toutes les images nécessaires à l'affichage d'un objet ou personnage.

3.2 Fichiers

Afin gérer l'accès à la mémoire de la carte *SD*, nous devions choisir entre deux bibliothèques. La première bibliothèque *SD* est celle fournie par l'*Arduino IDE*. La deuxième bibliothèque *SdFat* est celle que nous choisissons puisqu'il s'agit de la plus maintenue à jour par une communauté active. De plus, cette-dernière permettrait un accès aux ressources bien plus rapide.

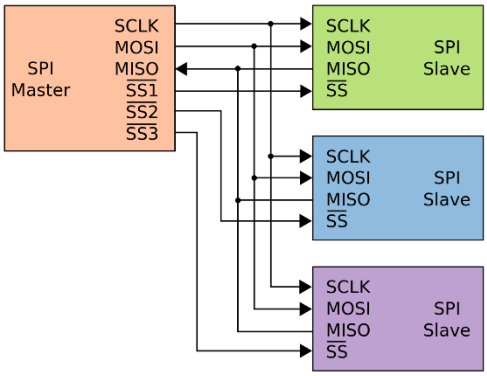
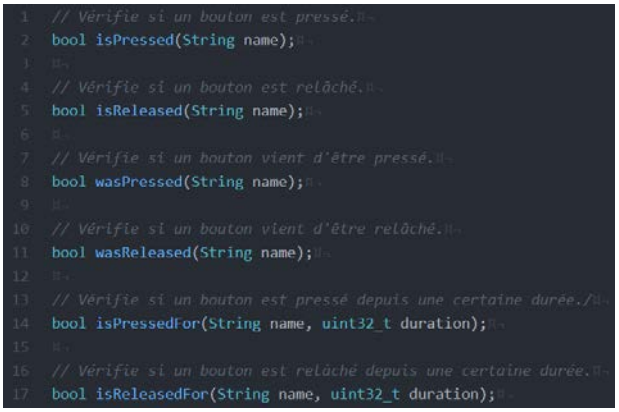


Figure 16: Schéma d'une liaison *SPI* à plusieurs esclaves

Il est ensuite très simple d'accéder à la mémoire et de manipuler les fichiers en lecture et en écriture. Nous imaginons encore une fois une bibliothèque permettant une initialisation plus simple de la carte *SD* et de désactiver la communication avec l'écran *TFT*, également en liaison *SPI*, lors de l'accès à la mémoire afin d'éviter les conflits.

3.3 Boutons

Une fois les circuits des boutons soudés et testés, il est nécessaire d'accéder facilement à l'état de pression des boutons. Pour cela, nous imaginons une bibliothèque permettant un accès simple et explicite à l'état et aux évènements des boutons.



```
// Vérifie si un bouton est pressé.
bool isPressed(String name);

// Vérifie si un bouton est relâché.
bool isReleased(String name);

// Vérifie si un bouton vient d'être pressé.
bool wasPressed(String name);

// Vérifie si un bouton vient d'être relâché.
bool wasReleased(String name);

// Vérifie si un bouton est pressé depuis une certaine durée./
bool isPressedFor(String name, uint32_t duration);

// Vérifie si un bouton est relâché depuis une certaine durée.
bool isReleasedFor(String name, uint32_t duration);
```

Figure 17: Fonctions disponibles.

```
PolyGamepad gamepad;

void setup() {
    Serial.begin(9600);
    gamepad.begin();
}

void loop() {
    gamepad.update();
    Serial.println(gamepad.isPressed("A"));
    Serial.println(gamepad.isPressedFor("X", 1000));
    Serial.println(gamepad.wasReleased("UP"));
}
```

Figure 18: Exemple d'utilisation.

Il est ainsi possible de savoir si un bouton est pressé, s'il est pressé depuis une certaine durée ou bien s'il vient d'être pressé ou relâché. Cette bibliothèque représente donc un pilier du développement des jeux pour la console.

3.4 Réseau

Pour notre projet, nous utilisons un Arduino Due et un *ESP8266*. L'Arduino s'occupe ainsi de la gestion du jeu et *l'ESP8266* gère la communication *Wi-Fi*. Nous avons également développé deux serveurs pour la communication avec *l'ESP*. Nous utilisons *l'ESP* uniquement pour transférer les requêtes au serveur et les réponses à l'Arduino par la communication série.

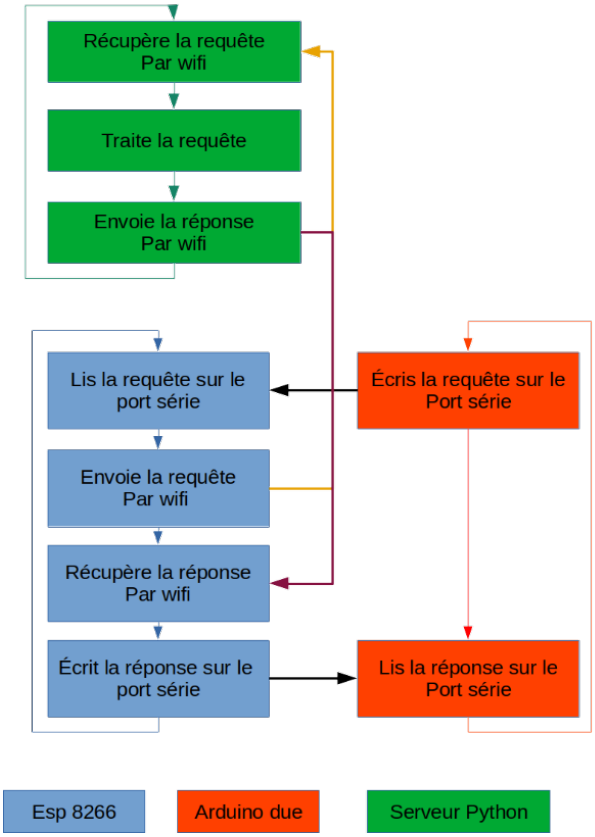


Figure 19: Schéma de fonctionnement de la communication réseau.

3.5 Serveur

Pour la partie connectée de notre console, nous avons développé plusieurs serveurs de communication afin de rajouter des fonctionnalités.

- Un serveur contenant pour chaque jeu, les scores et les noms des joueur pour sauvegarder un tableau des meilleurs joueurs. L'intérêt de le mettre sur un serveur est que les données ne sont pas sauvegardées sur la console. De plus, nous pourrions ajouter un site web qui afficherais le tableau de score de chaque jeu.

- Un serveur pour faire tourner des jeux en multi-joueurs. Ce serveur est lié à un « Game Core », un objet qui contient la logique de jeu et l'état du jeu. Ce serveur vérifie que les personnes qui communiquent avec lui sont bien des joueurs de la partie en cours et répond aux requêtés envoyées par la console (voir l'état du jeu, déplacement à tel position, etc...).

Pour gérer ces serveurs nous avons créé un programme sur lequel on rentre des commandes pour démarrer/éteindre le serveur, voir quel serveur est allumé et afficher ou non ce que font les serveurs. Il s'agit de la partie « Manager ».

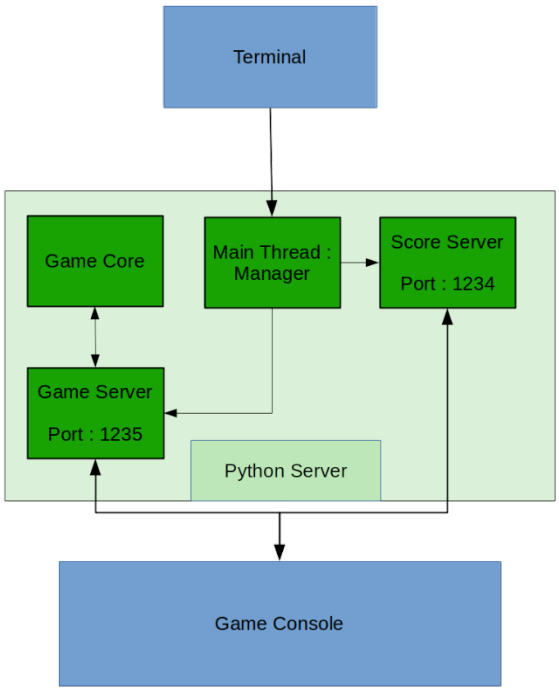


Figure 20: Schéma de fonctionnement des serveurs.

3.6 Circuit

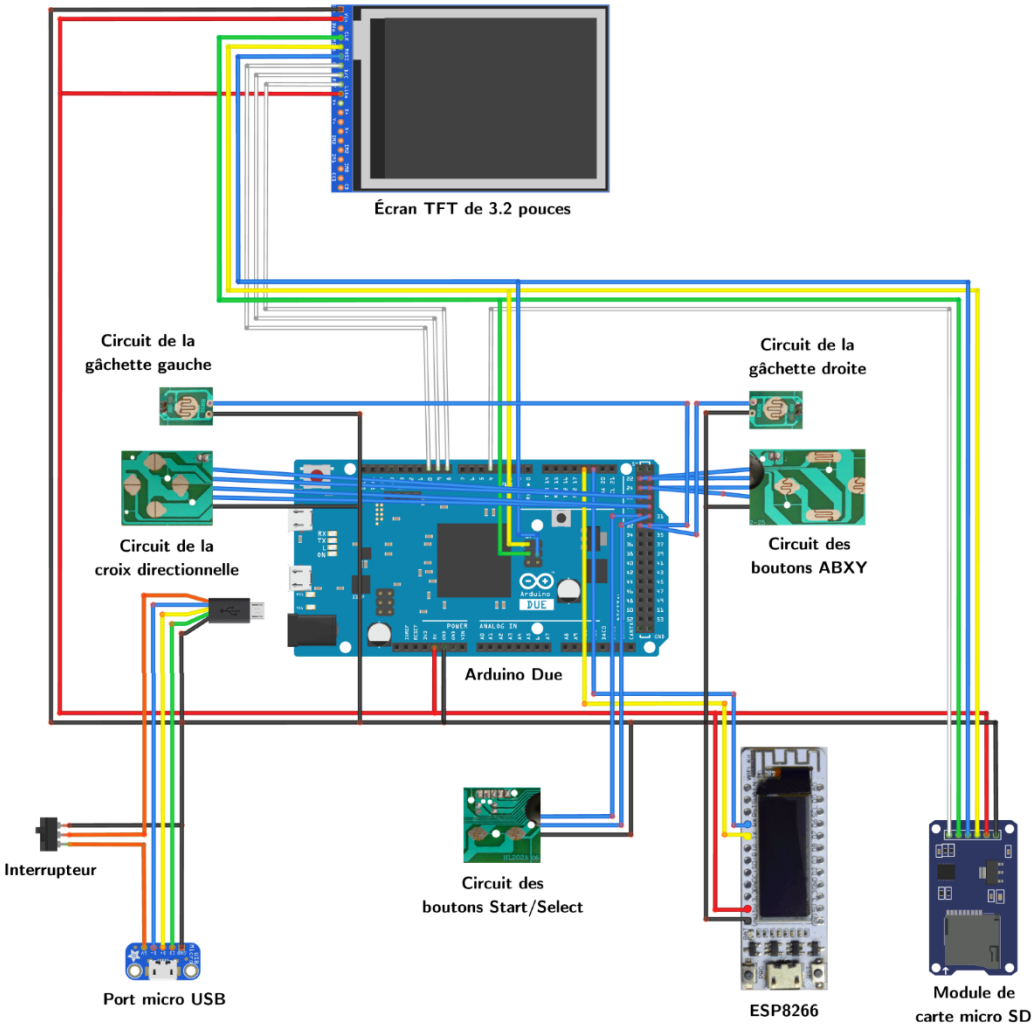


Figure 21: Circuit de la console.

1 Difficultés

3.1 Son

Notre première difficulté a été la gestion du son. En effet, nous n'avions pas beaucoup de temps et de connaissances sur le sujet. Nous avons notamment cassé le *DAC* (*Digital to Analog Converter*) de notre premier *Arduino Due* à cause d'une résistance manquante sur l'amplificateur audio.

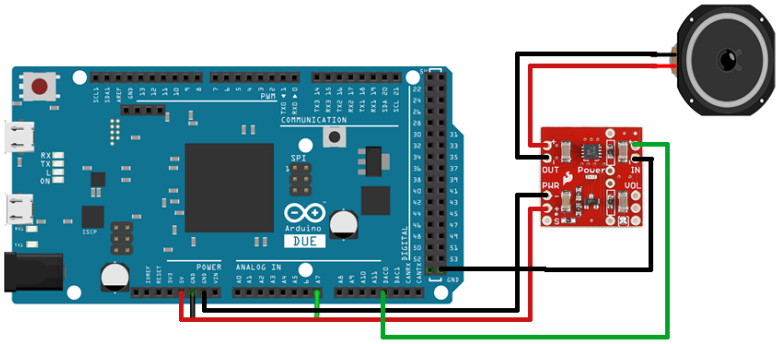


Figure 22: Circuit pour le son.

3.2 Mouvement

Enfin, notre deuxième difficulté a été la mise en place du gyroscope. En effet, nous avons eut un problème d'espace dans le boitier. De plus, les valeurs reçues du gyroscope étaient erronées. Cela était sans doute dû à un défaut de fabrication de notre gyroscope. Cependant, encore une fois nous n'avions plus le temps d'effectuer les vérifications nécessaires.

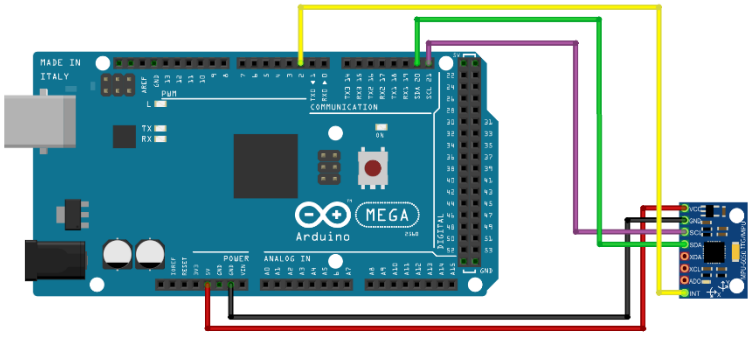


Figure 23: Circuit pour le gyroscope.

2 Conclusion

3.1 Planning

TITRE	PROGRESSION (EN %)	DATE PRÉVUE	DATE FINALE
ÉCRAN	100,00%	11/01/2019	26/02/2019
BOUTONS	100,00%	11/01/2019	18/01/2019
WIFI/BLUETOOTH	100,00%	11/01/2019	13/02/2019
CARTE SD	100,00%	18/02/2019	18/02/2019
SON	0,00%		
GYROSCOPE	0,00%		
MORPION	100,00%	18/01/2019	18/01/2019
JEU PS VITA	100,00%	04/03/2019	10/03/2019
LANCEUR	0,00%		
ALIMENTATION	0,00%		
DESIGN	100,00%	18/01/2019	13/03/2019
TOTAL	**63,64%**		

Figure 24: Planning et suivi de progression.

3.2 Perspectives

Notre projet est terminé cependant si nous avions eu plus de séances, nous aurions beaucoup d'idées pour l'améliorer. Avec l'état actuel de nos connaissances, nous avons dressé une liste des éléments qui amélioreraient grandement l'expérience de jeu :

- Ajouter une batterie pour pouvoir jouer sans chargeur.
- Ajouter un capteur de luminosité pour régler la luminosité de l'écran.
- Ajouter un gyroscope pour capturer les rotations et les mouvements.
- Implémenter le son pour accentuer l'immersion.
- Développer un lanceur de jeux et d'applications.
- Développer différents jeux et applications.

Références

[1] Marek Buriak, *ILI9341_due*, Bibliothèque d'affichage.

[2] Ilitek, *ILI9341*, Documentation du contrôleur *ILI9341*.

[3] Electronics Hub, *Basics of Serial Peripheral Interface*, Introduction à la liaison SPI.

[4] Bill Greiman, *SdFat*, Bibliothèque de gestion de la carte SD.

[5] Rinnegatamante, TheOfficialFloW et devnoname120, *VitaSDK*, Bibliothèque de développement pour la *PS Vita*.

[6] Communauté Arduino, *ESP8266Wifi*, Bibliothèque de communication *Wi-Fi* de l'*ESP8266*.

[7] Communauté Arduino, *ESP8266*, Documentation de l'ESP8266.

[8] Communauté Arduino, *Arduino Due*, Documentation de l'Arduino Due.

[9] Adafruit, *Preparing the Buttons, PiGRRL*, Tutoriel pour préparer le circuit des boutons.