Année 2020

Revue de projet Rushball

VINCENT Fanny, BUQUET Johan,
AUDE Camille, LIARD Guillaume,
SEURRE Matthieu
Professeurs référents : ANTOINE Philippe, HORTOLLAND Christian
Année 2020

ycée Benoit L'Isle Sur La Sorgue

BTS Systèmes Numériques IR EC

Sommaire

Préambule

Le projet Rushball est un projet dont 10 personnes s'occupent. Nous sommes donc, en réalité, une équipe de 10 séparée en 2 équipes de 5 personnes. Nous sommes donc amenés à tous communiquer.

1. Introduction

1.1 Contexte de réalisation

Constitution de l'équipe de projet :	Étudiant 1 EC **IR**	Étudiant 2 EC **IR**	Étudiant 3 EC **IR**	Étudiant 4 **EC** IR	Étudiant 5 **EC** IR
Projet développé :		Au lycée ou en centre de formation **Mixte**			En entreprise
Type de client ou donneur d'ordre (commanditaire) :		Entreprise ou organisme commanditaire : **Oui** Nom : Pierre ROCHE..... Adresse : ..... Contact : rushball@hotmail.fr Origine du projet : ➢ Idée : Lycée **Entreprise** ➢ Cahier des charges : **Lycée** Entreprise ➢ Suivi du projet : **Lycée** **Entreprise**			
Si le projet est développé en partenariat avec une entreprise :		Nom de l'entreprise : Pierre ROCHE Adresse de l'entreprise : Adresse site : Tél. : 06 60 53 24 30 Courriel : rushball@hotmail.fr			

1.2 Présentation du projet

Le Rushball est un jeu de balle qui se pratique seul ou en groupe jusqu'à quatre joueurs. Chaque joueur est équipé d'une raquette dans chaque main (fixée par strippe) et doit, à son tour, frapper la balle de manière à toucher une cible parmi d'autres (3 lignes de 6 cibles maximum) fixées au mur.
Le joueur peut jongler autant de fois que nécessaire, avec toutes les parties de son corps.
Chaque cible (50cm x 50cm) éclairée d'une même couleur est associée à un nombre de points.
Selon la règle du jeu initiale, 9 cibles sont allumées sur les 18.
Lorsqu'une cible est touchée, elle s'éteint et une nouvelle s'allume de la même couleur.
Les résultats en temps réel sont affichés à la vue des joueurs et des spectateurs.

En cas de faute identifiée par un arbitre (joueur), un pupitre de commande permet de corriger les scores avant de reprendre le jeu.

Les règles sont adaptables en fonction des joueurs et des objectifs à atteindre.
Le projet consiste à développer un système numérique permettant de gérer l'ensemble du jeu (paramétrage informatique du jeu, gestion de la grille de cibles, comptage des points, affichage, correction en cas de faute).

Ci-après le cahier des charges.

1.3 Cahier des charges – Expression du besoin

Pierre ROCHE, inventeur du Rushball, souhaite gérer le déroulement d'une partie de ce jeu de façon efficace. À ce jour, il n'existe pas de système numérique aidant à la gestion du jeu. Pour cette raison, Pierre ROCHE propose le développement d'une grille de cibles tactiles à LED multicolores facilement transportable.

Chaque cible aura une surface tactile permettant de réagir au "toucher de balle", sans interférer avec les autres cibles. Le système permettra de commander chaque cible pour l'allumer d'une certaine couleur ou pour l'éteindre.

Afin de paramétrer une partie de Rushball, il est demandé de disposer d'une application smartphone Android et/ou un logiciel PC, connecté au dispositif de gestion des cibles. Une autre solution consiste à créer un site WEB "local" afin de s'y connecter par un PC ou téléphone.
Afin de gérer les fautes durant une partie, un pupitre de commande raccordé au système de gestion des cibles, doit contenir un bouton poussoir "coup de poing" ainsi que des commandes permettant de corriger les scores et de mettre à jour le joueur qui doit reprendre la partie.

En option, il serait intéressant de garder en mémoire les résultats d'une partie, de les exporter sous forme de fichier CSV.

La partie logicielle doit se révéler suffisamment paramétrable pour accepter toutes les formes et règles du Rushball.

2. Précisions après les échanges

- Le système doit être portable et donc alimenté par batterie si la tension secteur n'est pas possible.
- Nombre de joueur entre 1 et 4.
- Les cibles ont une dimension de 50cm x 50cm et seront transportables par panneau de trois cibles verticales qui doivent s'emboiter les unes aux autres, tout en respectant la continuité électrique de chaque cible entre les panneaux.
- Si la balle frappe à cheval sur plusieurs cibles, le choix d'une seule sera prise en compte aléatoirement.
- Prévoir un décompte de point.
- Afficher la durée du jeu.
- En option, prévoir une durée de jeu.

Schéma de la grille de cibles

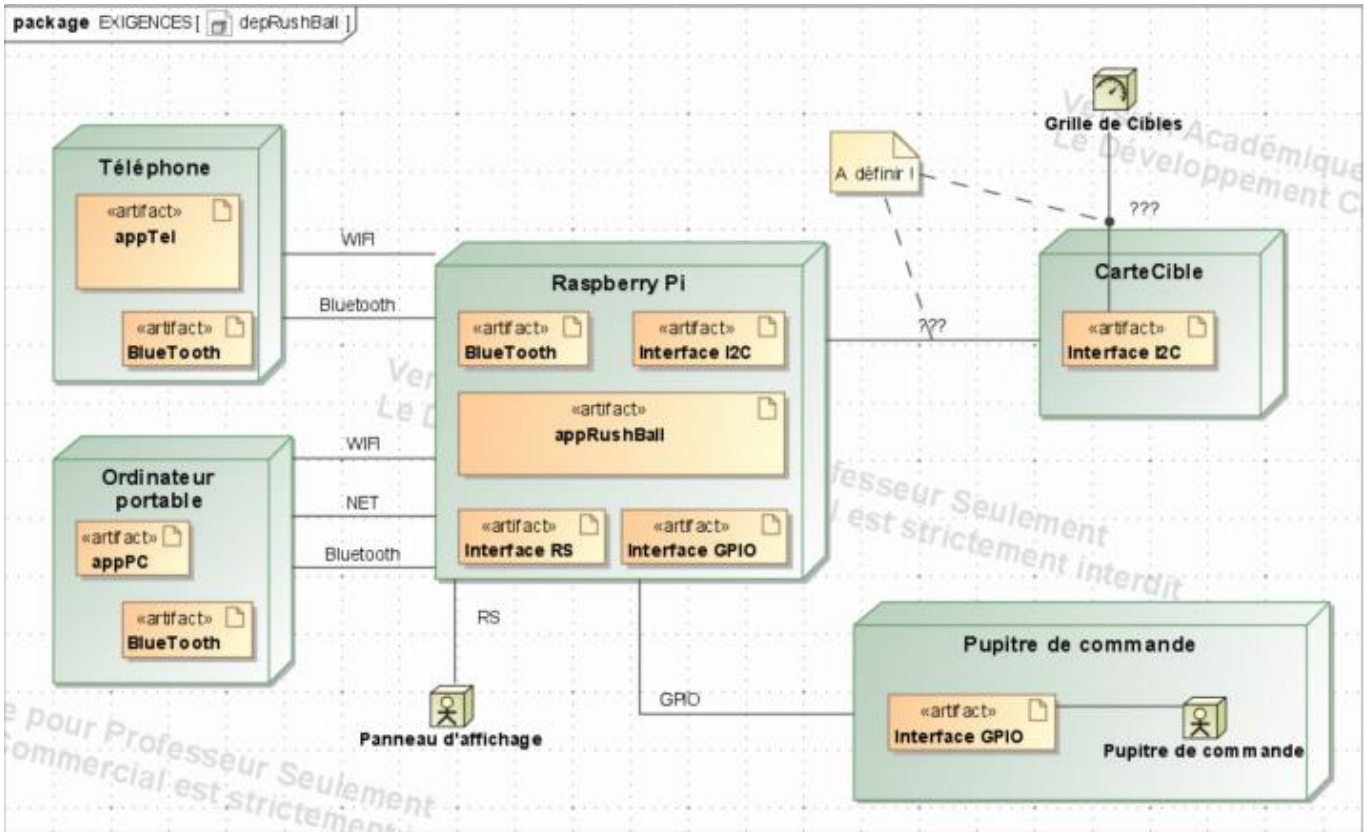
Panneau

Dans cet exemple, 9 cibles sont éclairées. Chaque couleur correspond à un nombre de point paramétrable. Lorsque la cible est touchée par la balle, elle s'éteint. Une autre cible de la même couleur s'allume. Option paramétrable : la cible violette est le Joker, elle s'allume que de temps en temps aléatoirement pendant un temps suffisant pour que les 4 joueurs puissent la viser au moins une fois. Elle n'apparait que rarement.

Tout doit être paramétrable au départ. Il existe une configuration par défaut correspondant à la règle standard.

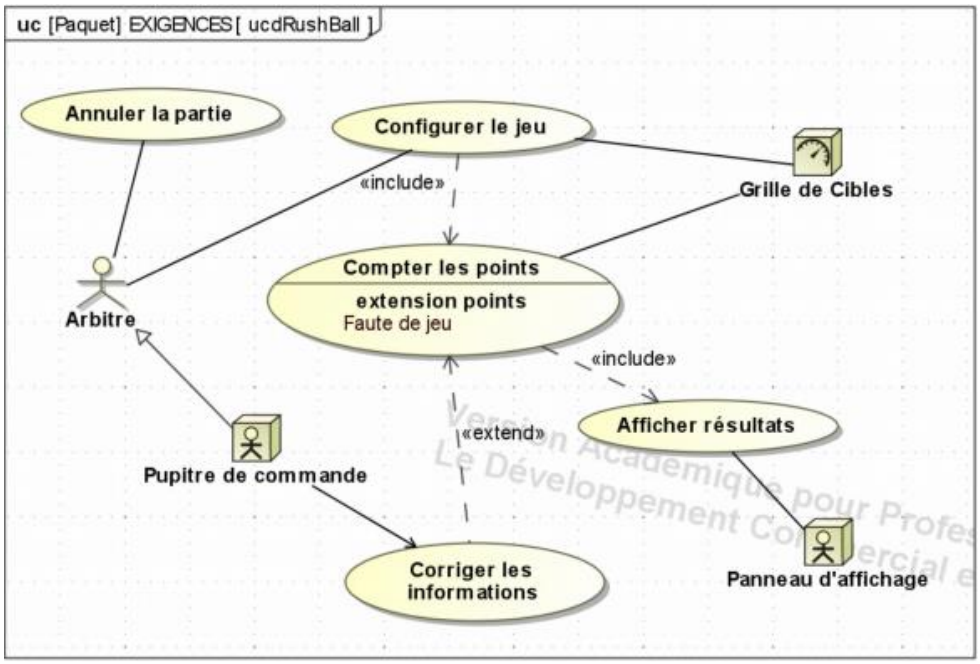
3. Spécifications

Un diagramme de déploiement de l'ensemble figure ci-dessous.



3.1 Diagrammes UML / SYSML

3.1.1 Diagrammes des cas d'utilisation



3.1.2 Architectures Matérielle & Logicielle

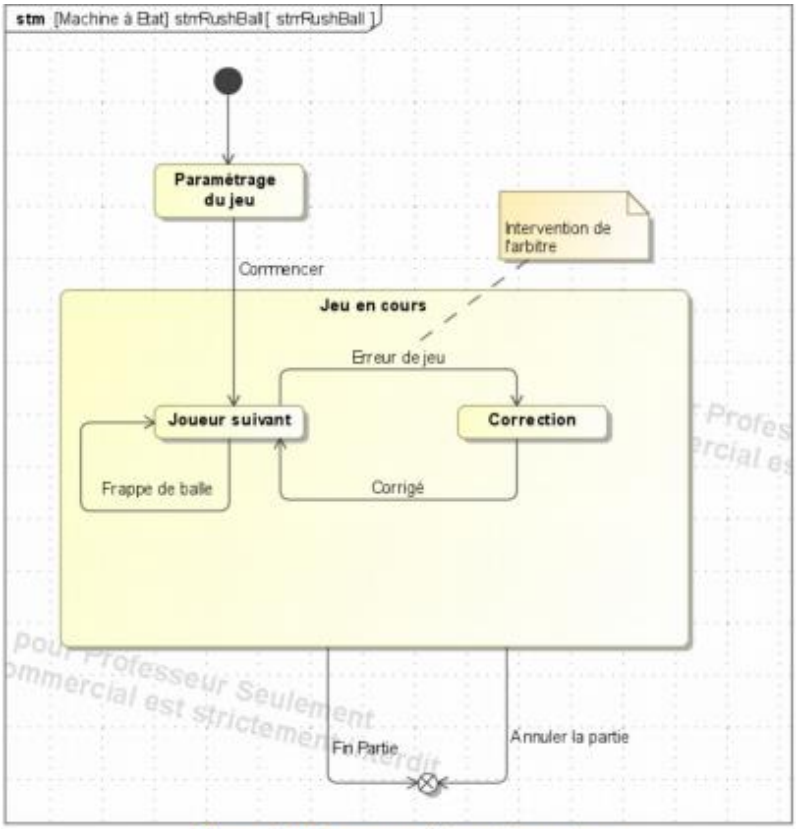


Figure 1 : Diagramme d'états du système

3.1.3 Scénarios des cas d'utilisation

3.1.3.1 Cas « Configurer le jeu »

Connecter le téléphone ou l'ordinateur à la carte Raspberry.

Configurer le nom et nombre de joueur (max 4).

Configurer le nombre de cibles (multiple de 3 et max 18).

Configurer le nombre de couleurs utilisées et la pondération associées.

Configurer le nombre de cibles allumées/éteintes (sélection aléatoire).

Configurer le mode du jeu (éteindre toutes les cellules ou durée de jeu).

Configurer la pénalité pour chaque faute.

Configurer la cible Joker.

3.1.3.2 Cas « Compter les points »

Précondition : Initialisation du jeu

Postcondition : Activer les cibles selon la règle choisie

1 – Eclairer le score du joueur.

Précondition : Durant le jeu

Postcondition : A chaque contact de la balle avec une cible :

Jeu 1 (voir UC Configurer le jeu) :

1 – Si la cible est celle éclairée, éteindre la cible en provoquant un effet visuel, compter le point pour le joueur.

2– Éclairer la cible suivante (aléatoirement).

Jeu 2 (voir UC Configurer le jeu) :

1 – Identifier le joueur,
2 – Si la cible atteinte est éclairée, éteindre la cible en provoquant un effet visuel, compter le point pour le joueur.
3 – Eclairer le score du joueur suivant.

3.1.3.3 Cas « Afficher résultats »

Affiche en permanence les résultats des joueurs.

Afficher un symbole indiquant à qui est le tour de jouer.

3.1.3.4 Cas « Annuler la partie »

Précondition : Partie engagée

Postcondition :

- Les scores sont annulés.

- Retour au cas d'utilisation « Configurer le jeu » (garder la configuration en cours)

3.1.3.5 Cas « Corriger les informations »

Précondition : Appui sur le bouton poussoir et partie engagée.

Postcondition : Diminuer le point du joueur concerné.

Fautes déterminées par l'arbitre :

La balle est en dehors du tableau de cibles.

La balle rebondie en dehors du terrain.

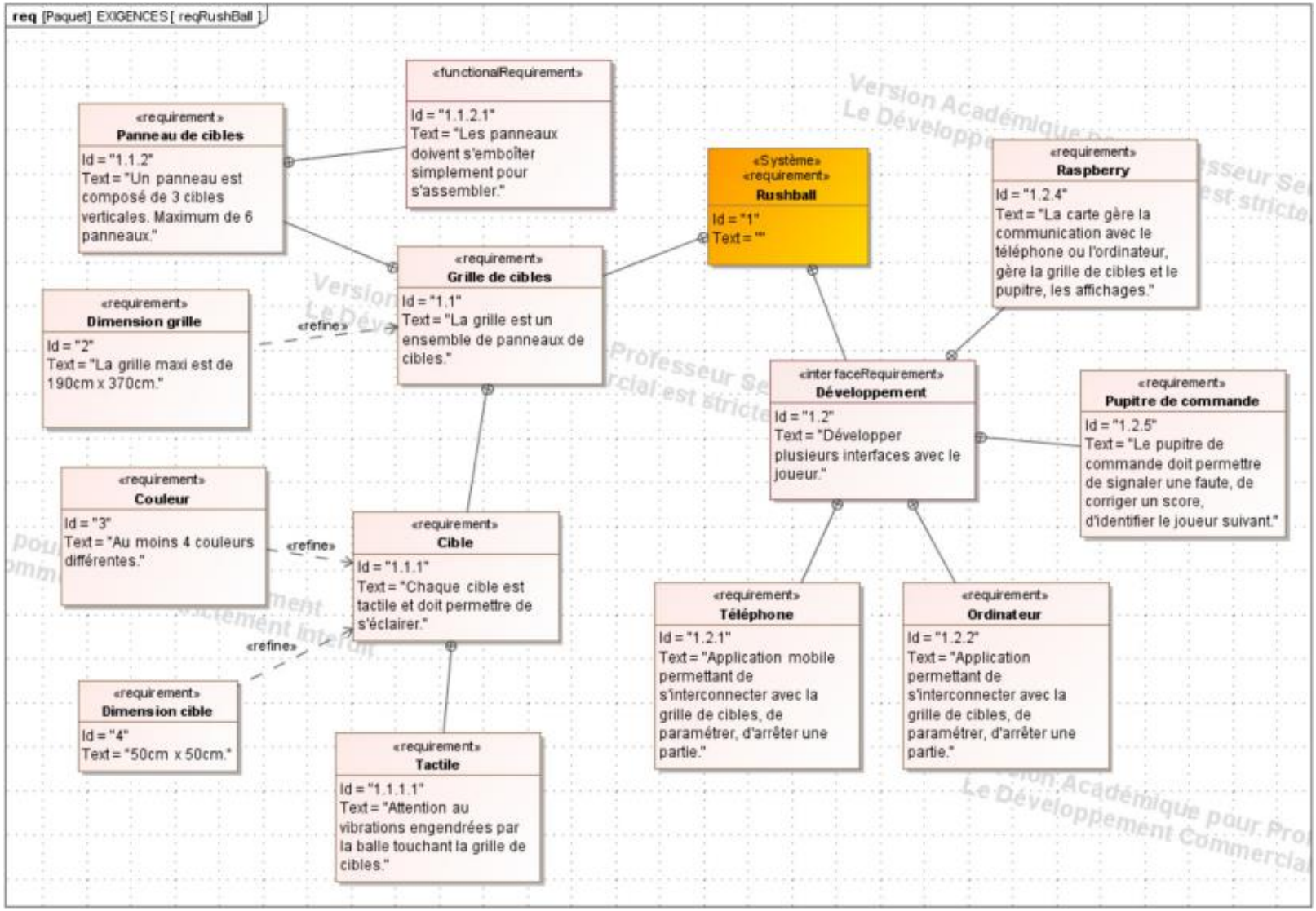
Le mauvais joueur tape la balle.

Mauvais jongle.

Nombre de rebonds > 1

NOTE : Nécessité de mémoriser l'attribution des points durant la partie.

3.1.4 Exigences



3.2 Contraintes de réalisation

Contraintes financières (budget alloué) :

Budget estimé de 500€ selon avancement du projet à charge partielle du demandeur.

Contraintes de développement (matériel et/ou logiciel imposé / technologies utilisées) :

La spécification, conception et codage seront modélisés.

Contraintes qualité (conformité, délais, …) :

Maintenable, maniable (ergonomie).

Contraintes de fiabilité, sécurité :

Les accès logiciels seront sécurisés.

3.3 Ressources mises à disposition des étudiants (logiciels / matériels / documents)

Matériels :

- Composants pour la réalisation ;
- Matériel de laboratoire (alimentation, oscilloscope, analyseur logique)
- Cartes de développement pour microcontrôleur Microchip : Curiosity HPC ;
- Outils de programmation pour microcontrôleur Attiny, ou carte Arduino (à définir).

Logiciels :

- Système d'exploitation ;
- Logiciel de modélisation SysML/UML : MagicDraw v7.02 ;
- Logiciels de conception électronique : KiCAD 5 ;
- Logiciel de conception électronique Fritzing (uniquement pour illustrer le prototypage rapide) ;
- Environnement de développement pour microcontrôleur Microchip : MPLAB X IDE ;
- Le logiciel de saisie de schéma et de simulation (Proteus ISIS) pourra éventuellement être utilisé pour illustrer des simulations.

Documents :

- Site de la section BTS SN mettant à disposition les différentes documentations.

4. Fiche de réunion

Compte-rendu de réunion.

Date	*Participants*	*Activité*
•30/01/20	• Pierre Roche • Philippe Antoine • Chirstian Hortolland • Mathieu Seurre • Léo Scholl • Gaétan Vogel • Liard Guillaume • Johan Buquet • Maxime figoureux • Floraian Amato • Enzo Ribeiro • Camille Aude • Fanny Vincent	• Présentation générale du projet et de son avancement: - Chaque membre du projet a présenté oralement à M.Roche les tâches qui leurs sont assignées. - Par la suite chacun a montré leur avancement (Montages électroniques, programmes). • Teste du jeu en extérieur: - M.Roche nous a montré comment ce déroule une partie de RushBall. - Nous avons ensuite essayé le RushBall. • Spécifications générales du projet: - Mise en accord sur la taille des panneaux à 50cm sur 50cm. - On a défini le nombre de couleurs à 7 (vert, bleu, rouge, jaune, magenta, cyan, blanc). - On a défini les règles des différents modes de jeu, voir document « Règles du jeu » rédigé par Mathieu Seurre. - On est passé de 6 panneaux à 8 panneaux pour une question de taille. - On a décidé de l'ajout du paramétrage des fautes. - On s'est mis d'accord sur une production de 2 panneaux (de 3 cibles chacun) pour la fin du projet.

REVUE DE PROJET N°1 EC21

BUQUET Johan

1. Introduction

Le projet Rushball est un jeu de balle assisté par ordinateur, praticable autant en intérieur qu'en extérieur, seul ou a plusieurs jusqu'à quatre joueurs en simultané. Chaque joueur est équipé d'une raquette dans chaque main et doit, à son tour, frapper la balle de manière à toucher une cible parmi les 21 (8 colonnes de 3 cibles) fixée au mur. Le joueur peut jongler autant de fois que nécessaire avec toutes les parties de son corps. Chaque cible (50x50cm) d'une même couleur possède un nombre de point défini au préalable. Lorsqu'une cible est touchée, elle s'éteint et une nouvelle s'allume de la même couleur. Les résultats en seront affichés en temps réel sur un afficheur externe à la vue des joueurs et spectateurs. En cas de faute, un pupitre de commande permet de corriger les scores si besoin avant de reprendre le jeu.

Le projet consiste à développer un système numérique permettant de gérer l'ensemble du jeu. (Paramétrage informatique du jeu, gestion de la grille de cible, comptage des points, affichage, correction en cas de faute.)

Nous sommes 10 étudiants sur le projet (6 EC et 4 IR), il y a deux binômes d'étudiant qui travaille sur le même objectif final mais avec des solutions différentes. (Arduino / Microchip)

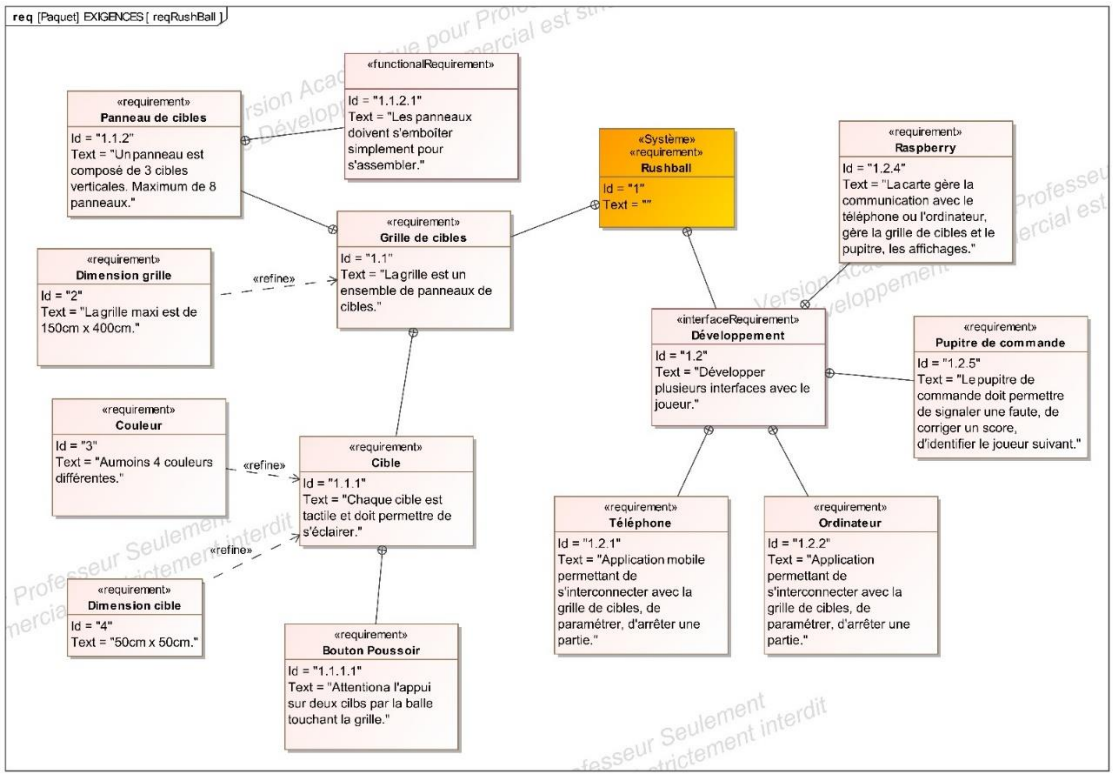
2. Rappel des objectifs

Ce qu'il m'est demander de faire :

Étudiant 3 **EC 21**	*Liste des tâches assurées par l'étudiant* • Participer au choix d'un modèle de microcontrôleur adapté au projet. • Programmer le microcontrôleur en esclave sur le bus I2C • Participer à l'élaboration d'un protocole de communication sur le bus I2C avec l'étudiant IR3. • Participer à la conception de la partie mécanique d'un panneau de 3 cibles • Programmer le microcontrôleur pour détecter des impacts de balles sur les 3 cibles d'un panneau • Le schéma structurel d'une solution possible, non testée, sera fourni à l'étudiant. • Conjointement avec l'étudiant EC2 produire un schéma structurel complet . • Effectuer la saisie du schéma et le routage de la solution proposée complète (routage individuel). Produire les fichiers Gerber afin que la fabrication du PCB soit sous-traitée. • Câbler la carte et effectuer les essais. • *Documenter la mise en service de la carte finalisée.*	**Installation :** MPLAB X IDE sous Windows. Prise en main d'une carte de développement à microcontrôleur Curiosity HPC. PIC18F45K50 . **Mise en œuvre :**Tester/valider/modifier une structure utilisant un microcontrôleur PIC18F45K50 fonctionnant en tant que circuit esclave sur le bus I2C, et permettant de détecter les impacts sur 3 cibles. Sauf si une meilleure solution est trouvée, la détection se fera grâce à des interrupteurs de fin de course. Le microcontrôleur devra communiquer avec un Raspberry Pi au travers d'un level-Shifter. Si nécessaire, en accord avec l'étudiant EC2, un autre modèle de PIC pourra être retenu. **Réalisation :** Suite aux essais préalables, proposer un schéma structurel complet de ce projet conjointement avec l'étudiant EC2, dont les essais sont complémentaires. L'alimentation devra être compatible avec le travail effectué par l'étudiant EC3 de l'autre projet. Après validation de la solution, concevoir un circuit imprimé devant être fabriqué industriellement. **Documentation :** • Schéma de câblage rapide (Fritzing) pour documenter la phase d'essais. • Documents de fabrication des cartes (KiCad). Ces documents devront avoir un niveau de qualité permettant une fabrication industrielle du circuit imprimé. • Schéma structurel avec contours IBD. • Liste complète des composants avec leurs sources d'approvisionnement. • Programme en C/C++ de communication sur le bus I2C, et de détection des impacts, accompagné des commentaires et diagrammes nécessaires à sa compréhension. • Fiche de mise en service. • Fiche de dépannage.

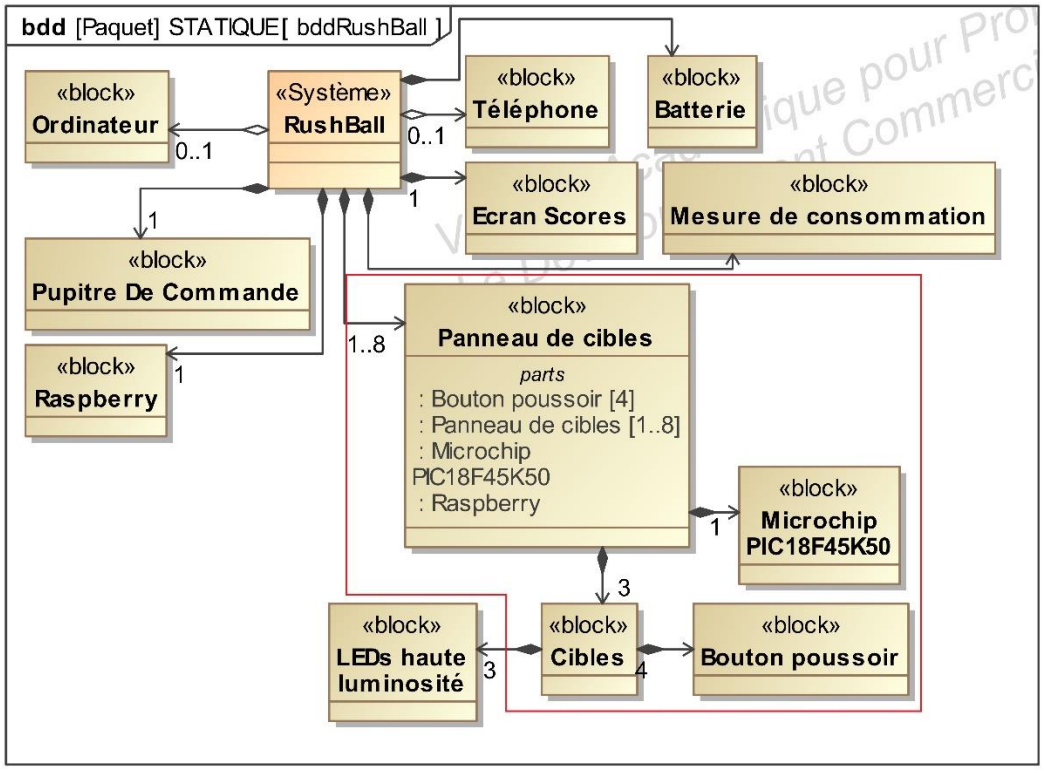
3. Diagrammes / Mise à disposition

3.1 Diagramme des exigences



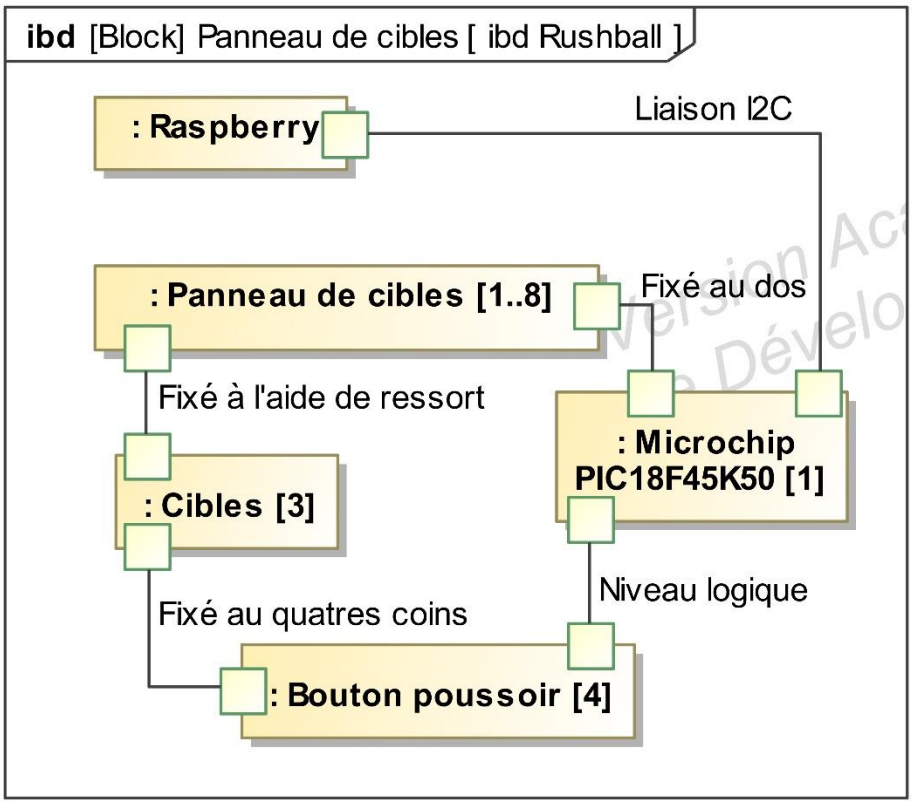
Pour ma partie du projet, il est question de mettre en œuvre les panneaux comportant 3 cibles, à l'aide de bouton poussoir positionner aux quatre coins d'une cible.

3.2 Diagramme de bloc



Voici encadré en rouge la partie qui m'est associé, elle est en parallèle avec Fanny VINCENT qui s'occupe de l'éclairage sur la cible. Nos deux parties sont doublé, Maxime FIGOUREUX et Florian AMATO on les même objectifs de nous mais sous un environnement de développement différent.

3.3 Diagramme de bloc interne



Les 3 cibles seront fixées sur le panneau avec un système de fixation (en cours de réflexion) et un ressort pour l'effet de pression. Aux quatre coins sont placés des bouton poussoir pour enregistrer l'impact de la balle. Un microcontrôleur de type PIC18F45K50 de chez Microchip est là pour gérer ces impacts, il fonctionne en interruption pour économiser des ressources et quand une cible est touchée, l'information remonte à la Raspberry via une communication I2C, qui va scruter à cet instant précis chaque microcontrôleur et modifié la couleur de la cible touchée.

3.4 Matériel / Logiciel mis à disposition

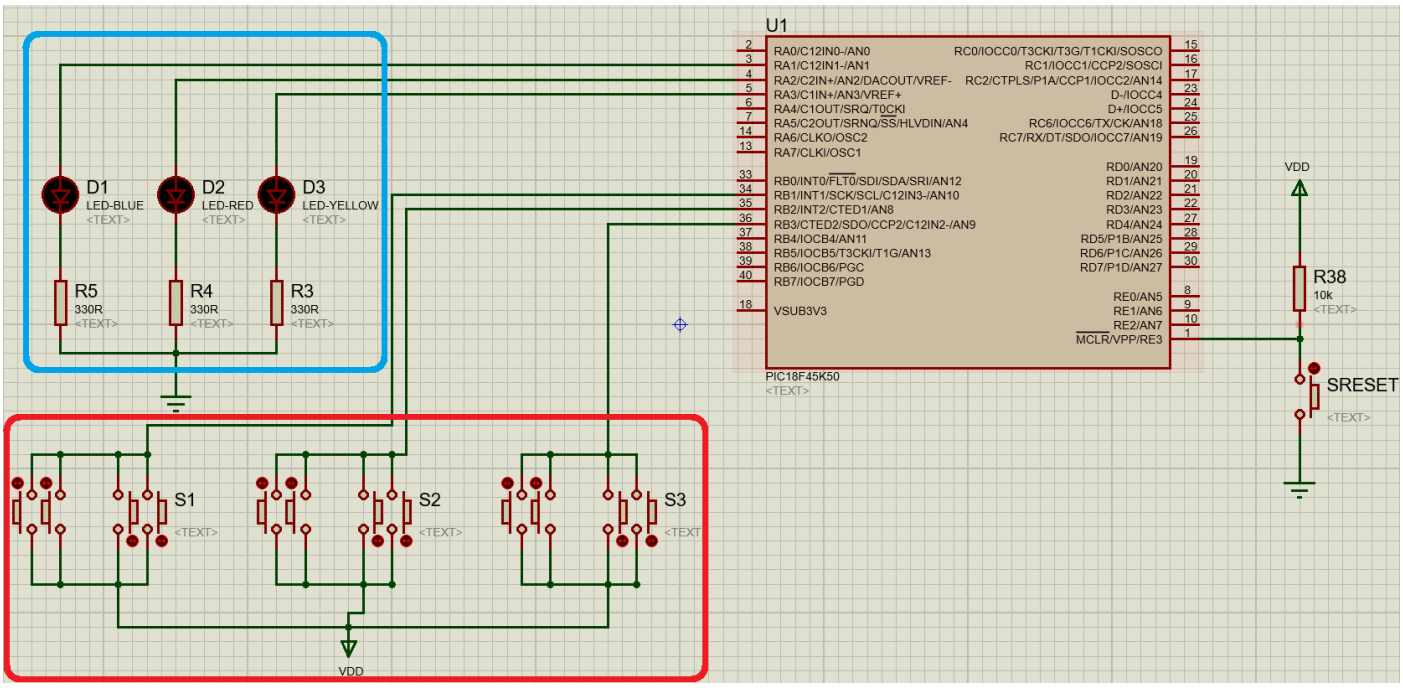
-Carte Curiosity HPC avec PIC18F45K50 / MPLAB X IDE v5.20

-Raspberry pi 3

-Bouton poussoir → 

4. Travail accompli

Au tout début du projet, j'ai réalisé un fichier Proteus pour avoir une visualisation grossière de ma partie.



En rouge sont les 3 cibles (S1, S2, S3) chacune composé de 4 boutons poussoirs.

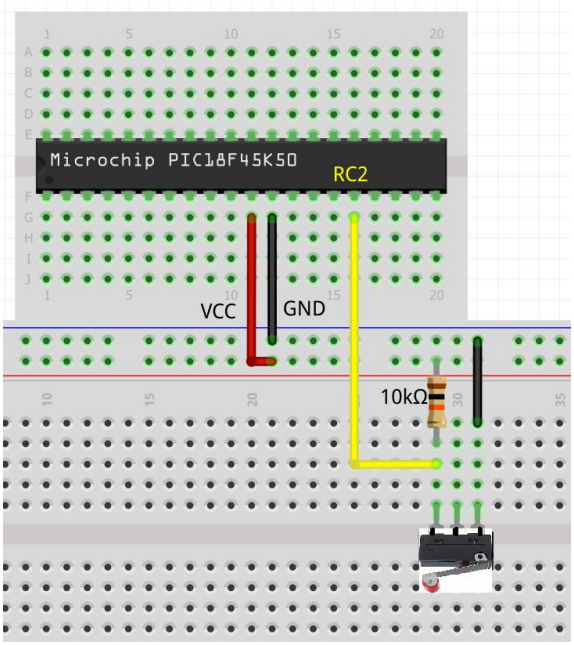
En bleu sont les 3 LEDs associés aux cibles

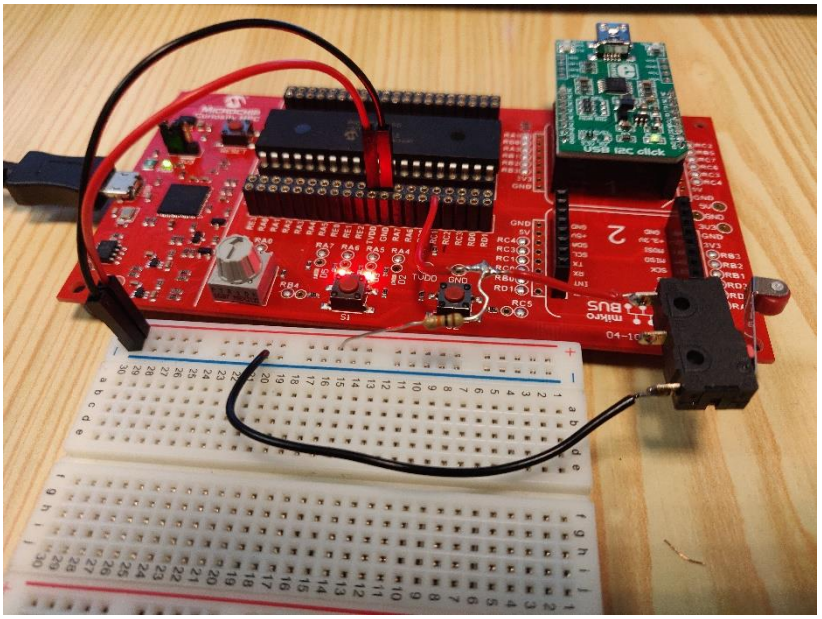
Ce schéma m'a servi par la suite à tester directement mon programme du microcontrôleur sans rien câblé, uniquement en simulation.

```
#include <xc.h>
#include "mcc_generated_files/mcc.h"

void main(void) {
    SYSTEM_Initialize();
    while(1)
    {

        //Panneau 1
    if (S1_GetValue())
        {
        D1_SetLow();
      }
    else
        {
        D1_SetHigh();

        }

        //Panneau 2
   if (S2_GetValue())
        {
        D2_SetLow();
        }
    else
        {
        D2_SetHigh();
        }
```

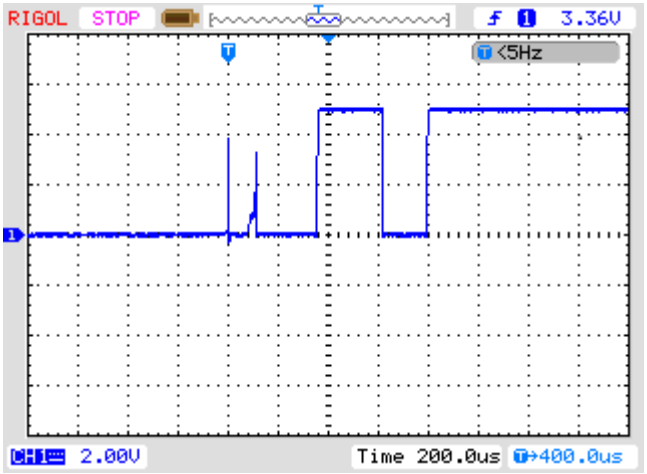
← Exemple de code du PIC18F45K50 très bref, si le bouton poussoir renvoie une valeur différente de son état de base, alors la led s'éteint, et inversement.

J'ai créé un Fritzing avec les composants à ma disposition pour mettre de l'ordre dans ma partie et avoir un modèle de câblage. Puis je l'ai câblé sur breadboard.



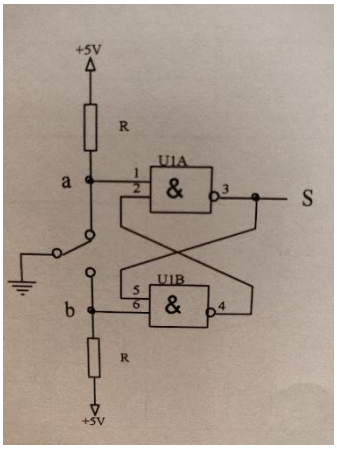


Avec cette solution, lors d'un appui ou relâchement du bouton, un rebond se produisais et perturbait le fonctionnement de la led, le niveau logique envoyé au microcontrôleur n'était pas net et ce dernier traduisais ces rebonds en clignotant rapidement la led voir à carrément changer l'état logique de la led et ainsi inverser le programme initial.



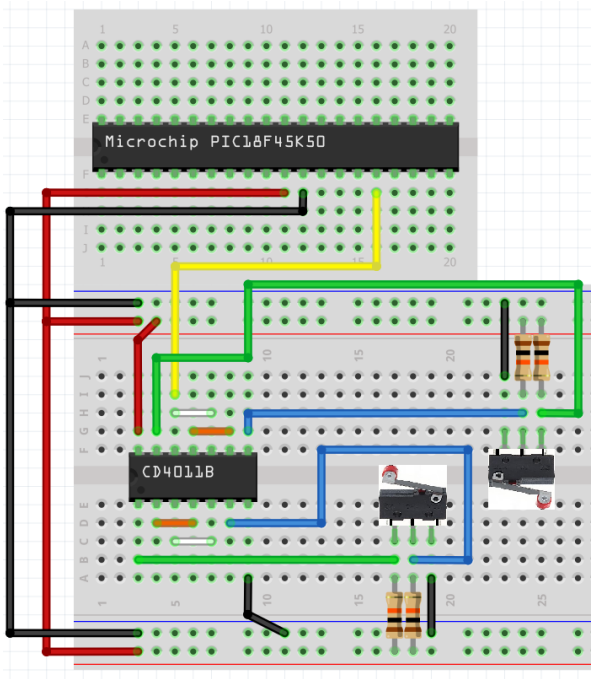
← Chronogramme lors de l'appui du bouton avec rebond

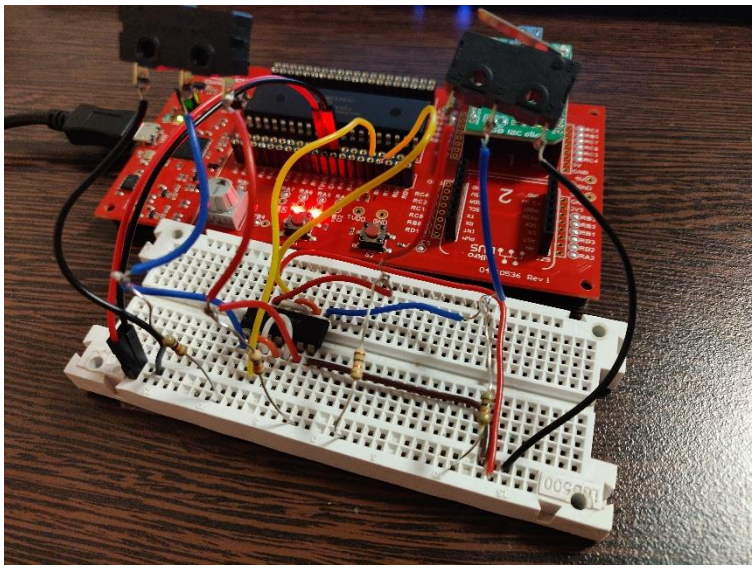
Pour pallier à ce problème, nous avons plusieurs options mais nous avons retenu l'utilisation d'un Circuit anti-rebonds à bascule RS NAND.

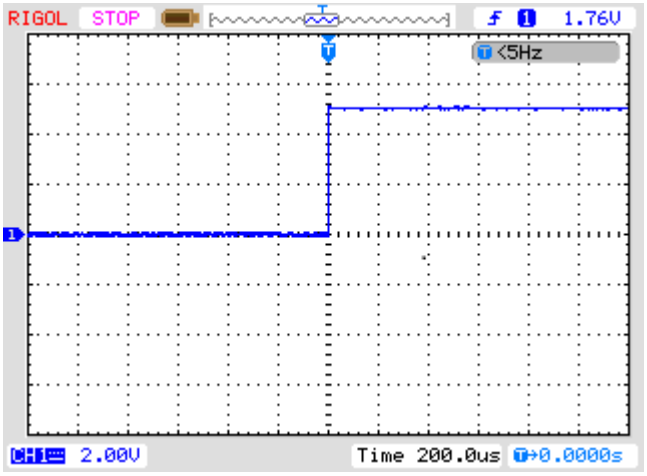


← Cette structure est réalisée par ce circuit →

J'ai donc câblé cette solution sur Fritzing puis sur breadboard, et relevé un chronogramme pour vérifier la disparition des rebonds.







← Chronogramme avec la structure anti-rebonds

5. Bilan

Dans l'avenir, il va falloir que je me penche sur la communication en I2C entre la Raspberry et les 8 microcontrôleurs, optimisé et achever les programmes de ces derniers. Commencé à réfléchir à la conception de la cible et les fixations adapté pour pouvoir réaliser des tests en condition de jeu réel.

REVUE DE PROJET N°1 EC22

VINCENT Fanny

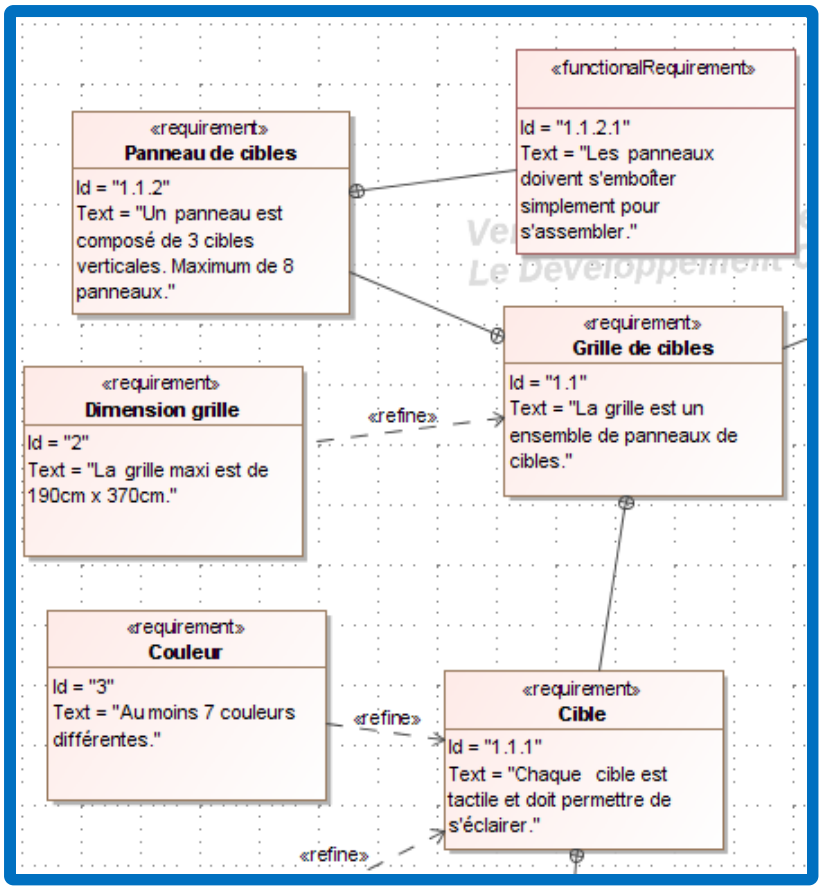
1. Introduction

1.1 Cahier des charges de l'étudiant EC22

Ce qui m'est demandé de faire dans ma partie :

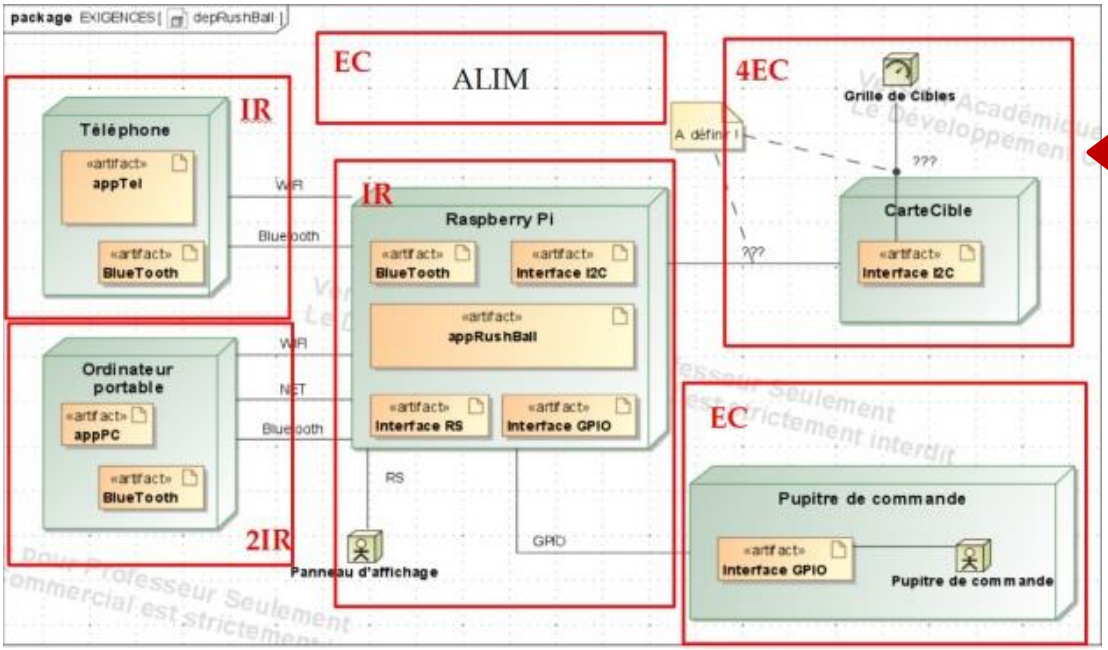
Étudiant 4 **EC 22**	*Liste des tâches assurées par l'étudiant* • Participer au choix d'un modèle de microcontrôleur adapté au projet. • Programmer le microcontrôleur en esclave sur le bus I2C • Participer à l'élaboration d'un protocole de communication sur le bus I2C avec l'étudiant IR3. • Programmer le microcontrôleur pour commander l'éclairage individuel de 3 cibles en utilisant des LEDs RGB haute luminosité de 3W en PWM. • Le schéma structurel d'une solution possible, non testée, sera fourni à l'étudiant. • Conjointement avec l'étudiant EC1 produire un schéma structurel complet . • Conjointement avec l'étudiant EC3 de l'autre projet définir les critères pour dimensionner une alimentation pouvant fonctionner sur secteur et de manière autonome. • Effectuer la saisie du schéma et le routage de la solution proposée complète (routage individuel). Produire les fichiers Gerber afin que la fabrication du PCB soit sous-traitée. • Câbler la carte et effectuer les essais. • *Documenter la mise en service de la carte finalisée.*	**Installation :** MPLAB X IDE sous Windows. Prise en main d'une carte de développement à microcontrôleur PIC18F45K50 (Curiosity HPC). **Mise en œuvre :**Tester/valider/modifier une structure utilisant un Attiny fonctionnant en tant que circuit esclave sur le bus I2C, et permettant de générer 3 sources lumineuses colorées utilisant des LEDs WS2812. Le microcontrôleur devra communiquer avec un Raspberry Pi au travers d'un level-Shifter. Si nécessaire, en accord avec l'étudiant EC1, un autre modèle de PIC pourra être retenu. **Réalisation :** Suite aux essais préalables, proposer un schéma structurel complet de ce projet conjointement avec l'étudiant EC2, dont les essais sont complémentaires. L'alimentation devra être compatible avec le travail effectué par l'étudiant EC3 de l'autre projet. Après validation de la solution, concevoir un circuit imprimé devant être fabriqué industriellement. **Documentation :** • Schéma de câblage rapide (Fritzing) pour documenter la phase d'essais. • Documents de fabrication des cartes (KiCad). Ces documents devront avoir un niveau dequalité permettant une fabrication industrielle du circuit imprimé. • Schéma structurel avec contours IBD. • Liste complète des composants avec leurs sources d'approvisionnement. • Programme en C/C++ de communication sur le bus I2C, et de commande de l'éclairage, accompagné des commentaires et diagrammes nécessaires à sa compréhension. • Fiche de mise en service. • Fiche de dépannage.

1.2 Diagramme des exigences EC22



Pour ma partie, je suis chargée de tester une des solutions de LED choisit qui permettra d'éclairer les cibles des panneaux. Pour cela, je mets en œuvre des LED RGB 3W avec le logiciel MPLABXIDE, avec une carte Curiosity HPC Microchip en esclave et un Raspberry Pi en maitre qui communiquent en I2C.

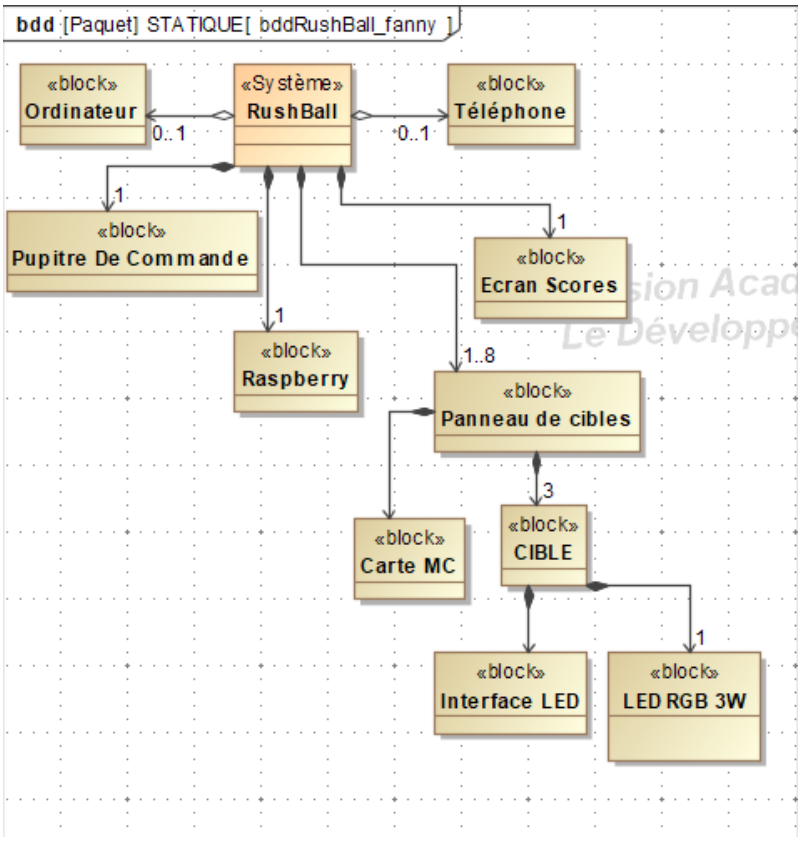
J'ai analysé le diagramme des exigences fournit où j'ai modifié certains paramètres qui n'étaient plus correct à la suite des deux réunions effectuées.



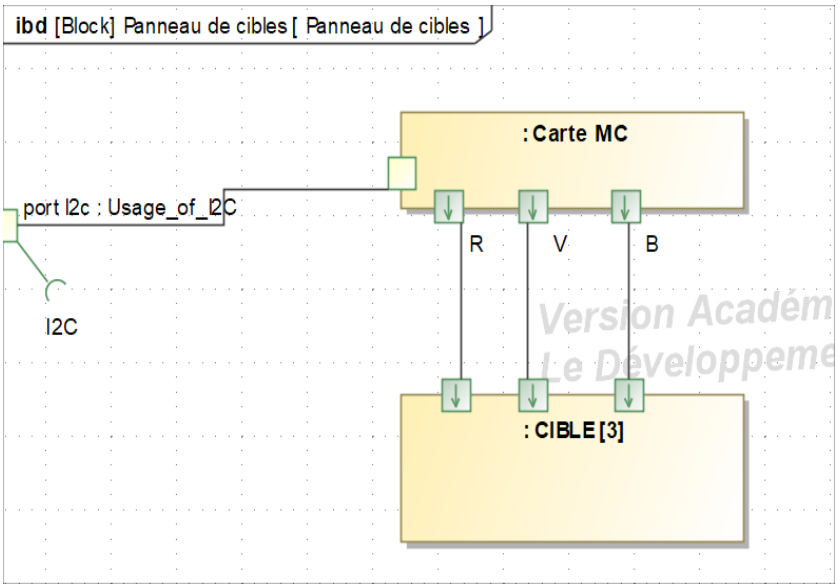
Puis, afin de mieux comprendre et voir ou je me situe globalement dans mon projet, j'ai pris soin d'étudier le diagramme des exigences fournit afin de bien comprendre ma partie.

1.3 Diagramme de bloc

Après avoir étudier le diagramme de bloc fournit, j'ai ajouté des détails concernant ma partie pour mieux visualiser ce qui met demander.



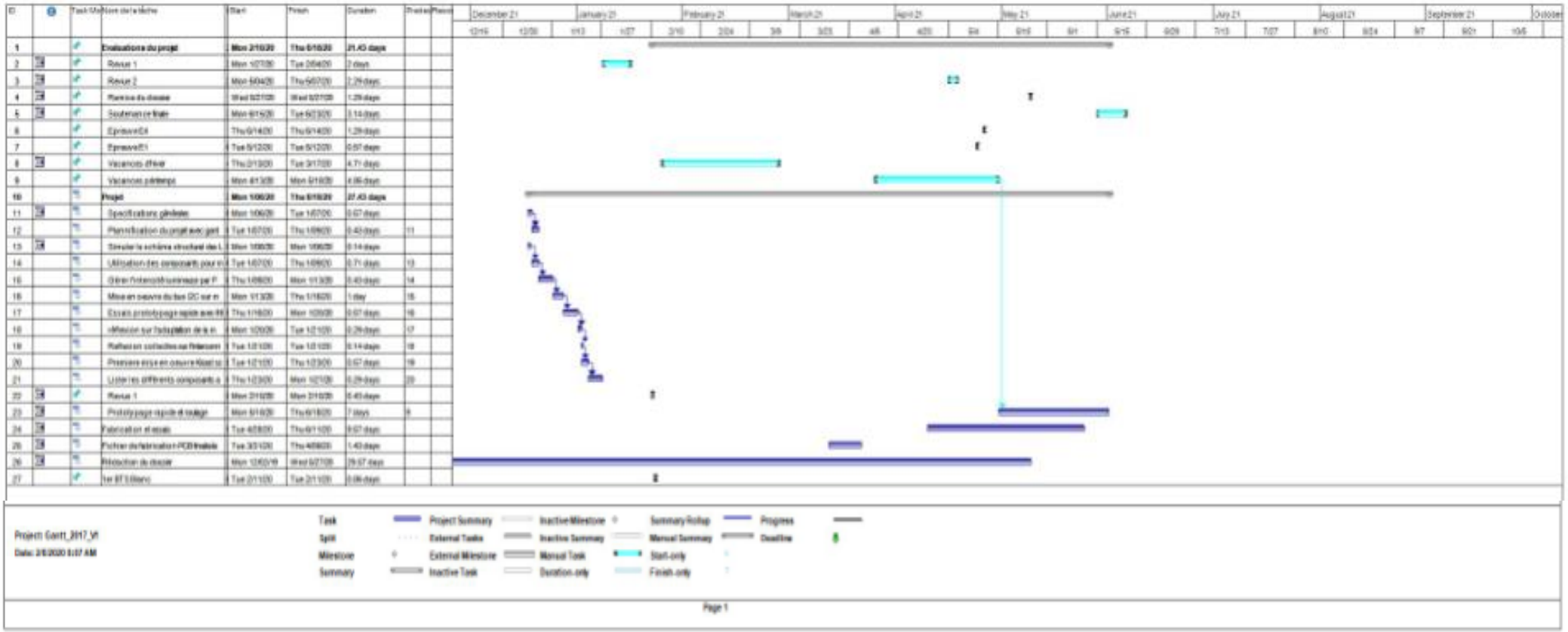
1.4 Diagramme de bloc interne



J'ai réalisé un diagramme de bloc interne afin Voici le diagramme de bloc modifié afin de voir plus en détail la partie qui me concerne.

Le jeu est donc constitué de 8 panneaux eux-mêmes composés de 3 cibles chacun. Ces panneaux ont aussi une carte Curiosity HPC Microchip par panneaux. Cette carte communique en I2C avec la Raspberry Pi, cette dernière étant en maitre et la carte Microchip en esclave.

1.5 Diagramme de Gantt



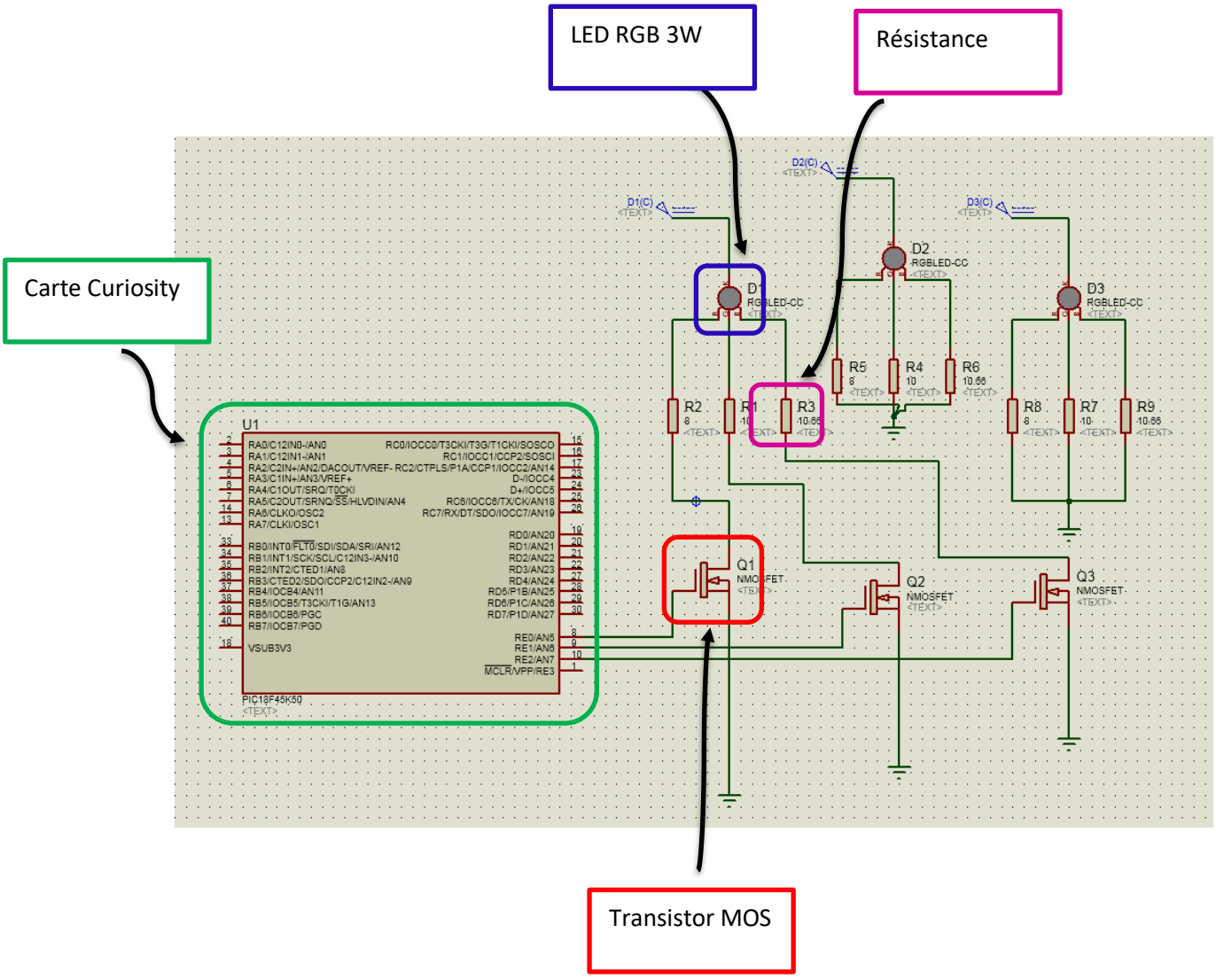
1.6 Matériel mis à ma disposition

- Curiosity HPC + Câble USB Type A, Micro USB Type B
- Carte Raspberry Pi 3
- LED RGB 3W × 3
- ULN2003 pour commencer les essais

2. Travail réalisé pour la revue

2.1 Essais Virtuel

2.1.1 Schéma Proteus



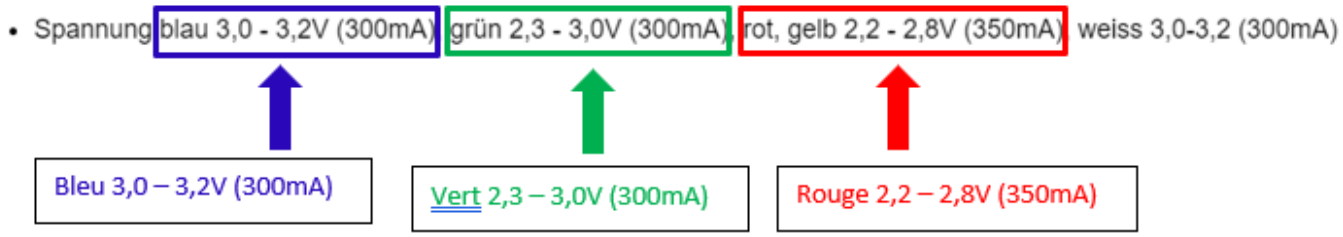
J'ai réalisé un schéma sur Proteus afin d'avoir une première approche du rendu possible de la LED RGB 3W. Pour ce faire, j'ai utilisé une carte Curiosity, un transistor MOS, des résistances, et une LED RGB 3W.

2.1.2 Comment choisir une résistance adaptée ?

Le choix des résistances est différent suivant la couleur voulue.

D'après la documentation (https://www.ebay.fr/itm/3-Watt-RGB-RGBW-RGBWY-LED-Chip-COB-Mehrfarben-LED-rot-gr%C3%BCn-blau-weiss-gelb/173433843370?ssPageName=STRK%3AMEBIDX%3AIT&var=472608536457&_trksid=p2060353.m2749.l2649),

pour une LED RGB 3W voici les paramètres à prendre en compte pour le calcul des résistances des différentes couleurs :



Pour calculer la résistance qu'il me faut pour chaque couleur j'utilise la formule U = RI soit R = U/I.

Je prends comme valeur de tension le maximum afin de prendre une marge pour être sûr d'avoir une résistance convenable même si la tension est le minimum.

Je prends soin de convertir les valeurs en mA en A.

350mA = 0.35A ;300mA = 0.3A

Pour la couleur rouge : 2.2/0.35 = 8 Ohm

Pour la couleur verte : 3/0.3 = 10.66 Ohm

Pour la couleur bleue : 3.2/0.3 = 10 Ohm

Voilà les valeurs de résistance dont j'ai besoin pour mes 3 différentes couleurs.

2.1.3 Programme

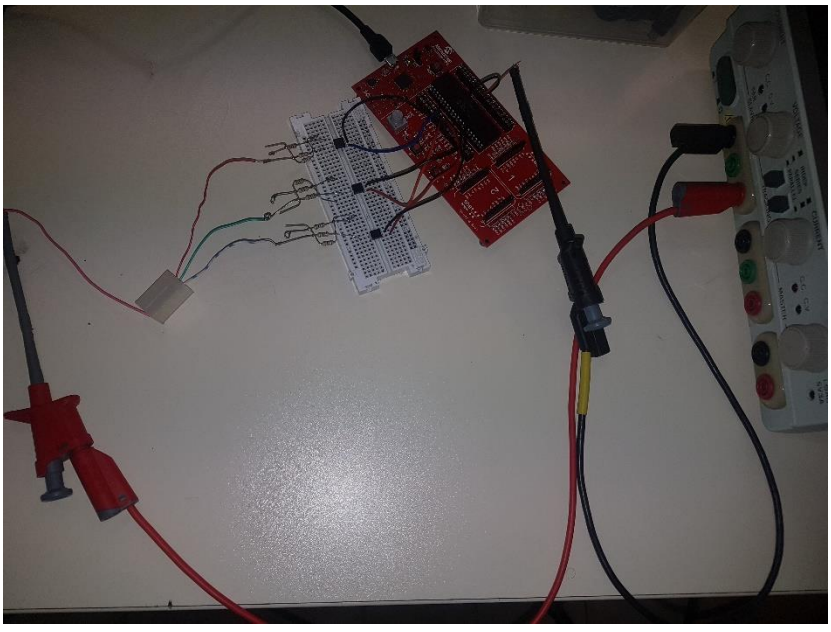
```
#include <xc.h>
#include "mcc_generated_files/mcc.h"
#define RE0 //définition des broches que j'utilise
#define RE1
#define RE2

void main(void) {
    SYSTEM_Initialize();
    while(1)
    {
        Bleu_D1_SetLow(); // la lumière bleu est éteinte
        Rouge_D1_SetLow(); // la lumière rouge est éteinte
        Vert_D1_SetHigh(); // la lumière verte est allumée

    __delay_ms(2000);    // on attend 2s

        Bleu_D1_SetLow();
        Rouge_D1_SetHigh();
        Vert_D1_SetLow();

    __delay_ms(2000);
```

Voici le programme que j'ai réalisé sur le logiciel MPLABX afin de mettre en œuvre la LED sur Proteus.

Après avoir définit les broches que j'utilise, et déterminer les noms de chaque couleur, j'exprime que je souhaite une ou plusieurs couleurs allumées avoir la commande SetHigh, ou éteinte avec SetLow.

2.2 Essais réel

2.2.1 Câblage



Voilà ce que donne le câblage, une fois réalisé.

J'utilise une alimentation de 5V qui correspond à 100% de la luminosité.

La tension sera éventuellement plus élevée (9V ou bien 12V). C'est une question à laquelle on répondra plus tard.

2.2.2 Programme

```
#include <xc.h>
#include "mcc_generated_files/mcc.h"
#define RE0 //définition des broches que j'utilise
#define RE1
#define RE2

void main(void) {
    SYSTEM_Initialize();
    while(1)
    {
        Bleu_D1_SetLow(); // la lumière bleu est éteinte
        Rouge_D1_SetLow(); // la lumière rouge est éteinte
        Vert_D1_SetHigh(); // la lumière verte est allumée

    __delay_ms(2000);   // on attend 2s

        Bleu_D1_SetLow();
        Rouge_D1_SetHigh();
        Vert_D1_SetLow();

    __delay_ms(2000);

        Bleu_D1_SetHigh();
        Rouge_D1_SetLow();
        Vert_D1_SetLow();

    __delay_ms(2000);

        Bleu_D1_SetHigh();
        Rouge_D1_SetHigh();
        Vert_D1_SetLow();

    __delay_ms(2000);

        Bleu_D1_SetLow();
        Rouge_D1_SetHigh();
        Vert_D1_SetHigh();

    __delay_ms(2000);

        Bleu_D1_SetHigh();
        Rouge_D1_SetLow();
        Vert_D1_SetHigh();

    __delay_ms(2000);

        Bleu_D1_SetHigh();
        Rouge_D1_SetHigh();
        Vert_D1_SetHigh();

    __delay_ms(2000);
```

Puis ci-contre voici le programme afin d'obtenir les différentes couleurs possibles les unes à la suite des autres afin de s'assurer que la LED fonctionne correctement.

Ici la LED s'éclairera dans l'ordre suivant avec un délai de 2 secondes entre chaque couleur :

- Vert
- Rouge
- Bleu
- Magenta
- Jaune
- Cyan
- Blanc

Il faudra voir par la suite avec les règles du jeu pour déterminer l'ordre des couleurs sur chaque cible de chaque panneau.

3. Conclusion

3.1 Ce qui a été accompli

Le dossier a bien été compris. Une première mise en œuvre d'une LED RGB 3W a été réalisée, en virtuel puis en réel avec une carte Curiosity HPC. Une première association des couleurs a permis de prendre connaissance des couleurs à notre disposition qui comprend donc le rouge, le vert et le bleu qui peuvent elles-mêmes être associées pour donner le cyan, le jaune, le magenta et le blanc. Ce qui donne en tout 7 couleurs possibles.

3.2 Ce qui reste à faire

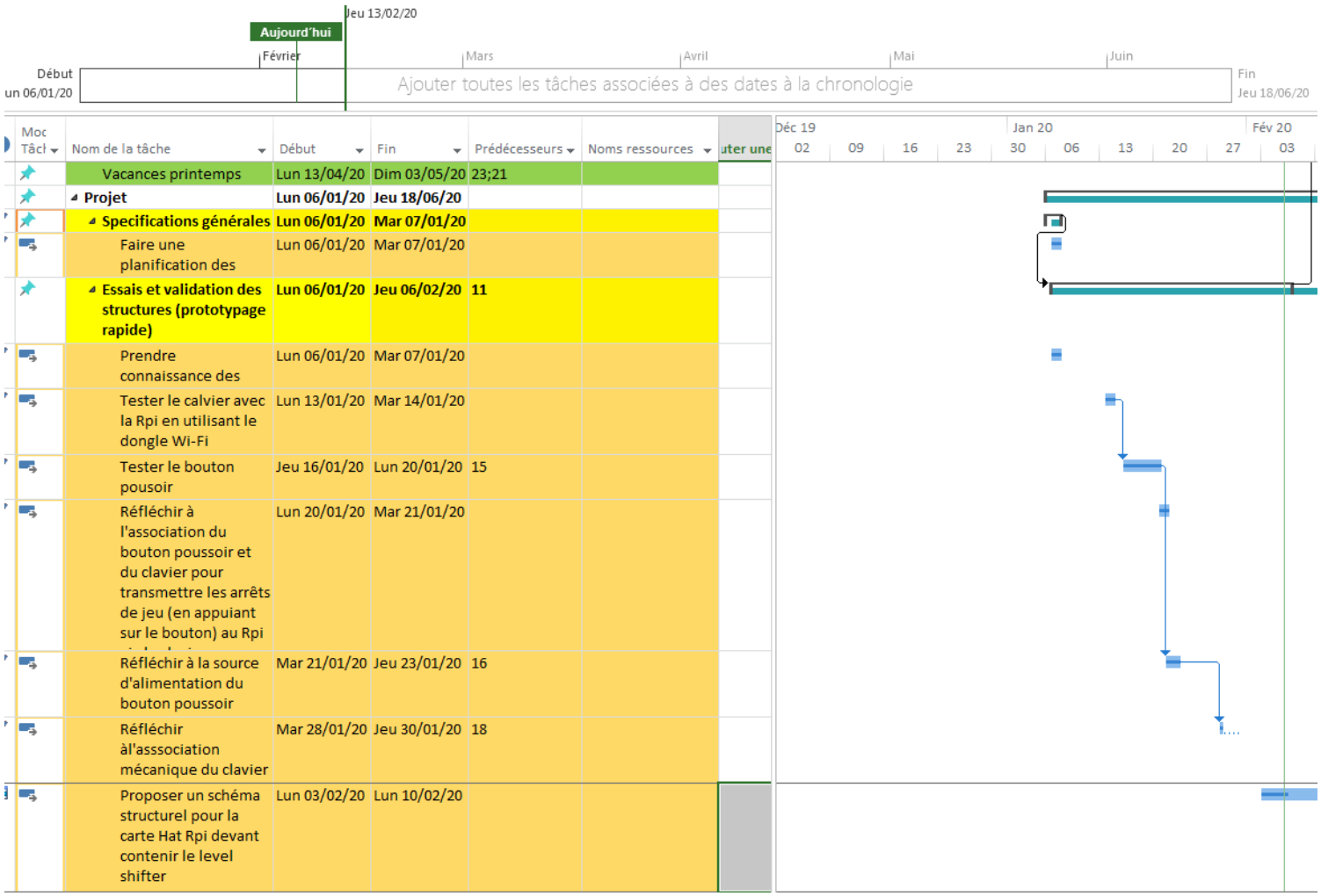
- Mise en œuvre du bus I2C sur microcontrôleur Microchip en mode maître et en mode esclave.
- Gestion de l'intensité lumineuse par PWM programmable.
- Si nécessaires (en fonction des tensions d'alimentation) un level shifter sera câblé entre Rpi et Curiosity HPC.
- Suite aux essais vérifiés si le PIC18F45K50 est adapté pour le projet, ou doit être remis en question pour un autre modèle.
- Réfléchir collectivement à l'interconnexion des différentes cartes.
- Proposer une première version de schéma structurel (KiCAD).

Liste des composants.

REVUE DE PROJET N°1 EC23

AUDE Camille

1. Gantt :



Moc Tâcl	Nom de la tâche	Début	Fin	Prédécesseurs	Noms ressources	uter une
	Vacances printemps	Lun 13/04/20	Dim 03/05/20	23;21		
	Projet	**Lun 06/01/20**	**Jeu 18/06/20**			
	Specifications générales	**Lun 06/01/20**	**Mar 07/01/20**			
	Faire une planification des	Lun 06/01/20	Mar 07/01/20			
	Essais et validation des structures (prototypage rapide)	**Lun 06/01/20**	**Jeu 06/02/20**	**11**		
	Prendre connaissance des	Lun 06/01/20	Mar 07/01/20			
	Tester le calvier avec la Rpi en utilisant le dongle Wi-Fi	Lun 13/01/20	Mar 14/01/20			
	Tester le bouton pousoir	Jeu 16/01/20	Lun 20/01/20	15		
	Réfléchir à l'association du bouton poussoir et du clavier pour transmettre les arrêts de jeu (en appuiant sur le bouton) au Rpi	Lun 20/01/20	Mar 21/01/20			
	Réfléchir à la source d'alimentation du bouton poussoir	Mar 21/01/20	Jeu 23/01/20	16		
	Réfléchir àl'asssociation mécanique du clavier	Mar 28/01/20	Jeu 30/01/20	18		
	Proposer un schéma structurel pour la carte Hat Rpi devant contenir le level shifter	Lun 03/02/20	Lun 10/02/20			

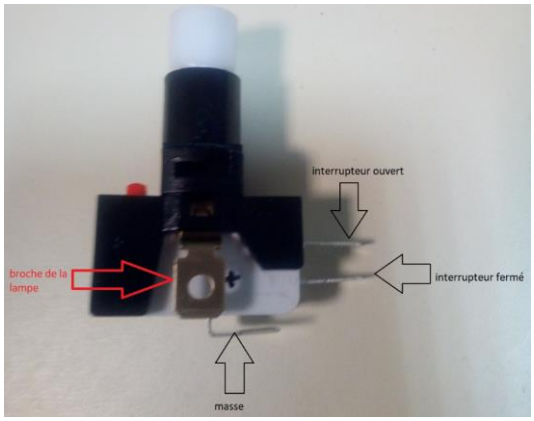
2. Cahier des charges :

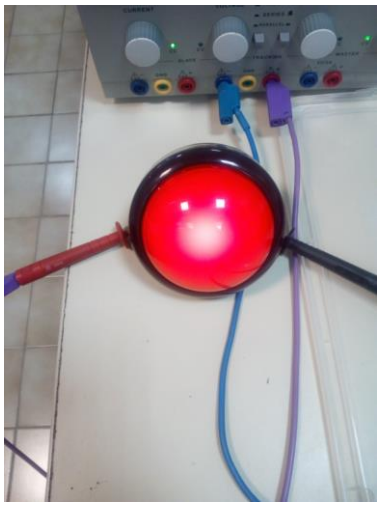
Réalisation d'un pupitre de commande permettent interrompe le jeu quand il y a une faute (exemple : quand la balle n'est plus dans la zone de jeu) grâce au bouton poussoir ou à une touche du clavier, ici la touche [*], et pour on utilisera la touche [/] pour redémarrer la partie.

3. Essai et test :

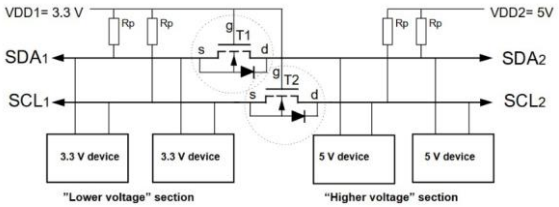
J'ai tout d'abord testé le clavier à 18 touches avec la Raspberry pi 3 pour vérifier que toute les touches fonctionnent et que la communication entre le clavier et le plug wifi se déroule sans encombre.
Pour effectuer ce teste, j'ai des piles de type AAA de 1,5V dans le clavier puis j'ai connecté le plug wifi à un port USB de la Raspberry pi puis j'ai ouvert un éditeur de texte puis visualiser les caractères afficher, en appuyant dessus une par une. Sauf les touches [/], [+], [-], [*], [Num lock] et [Back space], les autres ont deux options comme par exemple la touche [=] et aussi la touche [entrée].

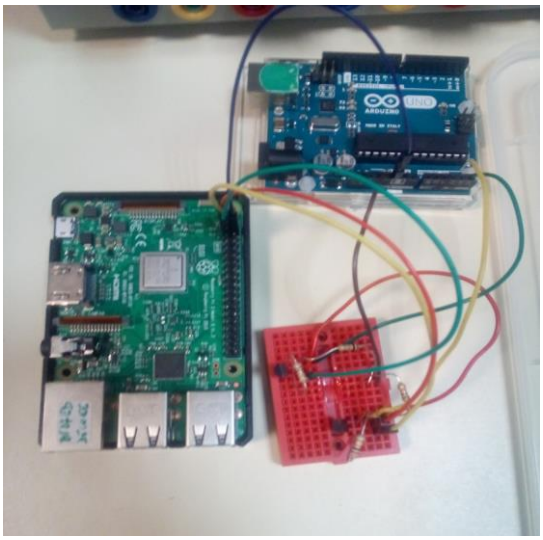
Puis j'ai testé le bouton, qui est lumineux, grâce à un multimètre et un générateur basse tension.
Avec le multimètre j'ai vérifié quelle broche fonctionne quand on appui sur le bouton et quelle broche fonctionne quand le bouton est relâché.
Avec le générateur basse tension j'ai illuminé le bouton en branchant aux bonnes broches car il y a un sens.

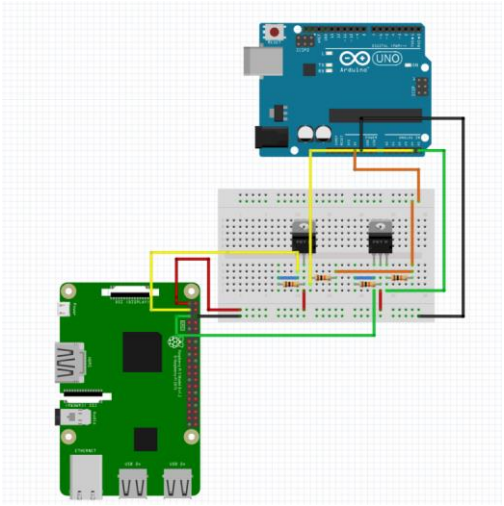




J'ai effectué le branchage d'un level-shifter, ce principe permet de régulariser le niveau de tension entre deux appareils.
J'ai effectué le branchement d'un level-shifter pour communiquer entre une carde Arduino Uno et une Raspberry pi 3 peut voir ci-dessous les images.







Pour la communication, on utilise une communication maître-esclave ci-dessous on le programme pour l'Arduino

```
**********************************************************************************
#include <Wire.h>

void setup() {

Wire.begin(); // join i2c bus (address optional for master)

Serial.begin(9600); // start serial for output

}

void loop() {

Wire.requestFrom(8, 6); // request 6 bytes from slave device #8

while (Wire.available()) { // slave may send less than requested

char c = Wire.read(); // receive a byte as character

Serial.print(c); // print the character

}

delay(500);

}
**********************************************************************************
```

Pour utiliser le programme sous Raspberry pi on utilise ses commandes :

```
**********************************************************************************************************************
pi@raspberrypi:~ $ cd Documents
pi@raspberrypi:~/Documents $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
pi@raspberrypi:~/Documents $ g++ progrpi_arduino1.cpp -l bcm2835 -o progrpi_arduino1
pi@raspberrypi:~/Documents $ sudo ./progrpi_arduino1
**********************************************************************************************************************
```

Et le programme est celui-ci

```
**********************************************************************************************************************
#include <iostream>
#include <bcm2835.h>
using namespace std;

#define clk_div BCM2835_I2C_CLOCK_DIVIDER_2500

// g++ DS1621_V1.cpp -l bcm2835 -o DS1621_V1

void init_I2c_bcm2835()
{
        if (!bcm2835_init())
                {
                printf("bcm2835_init failed. Are you running as root??\n");
                }
        if (!bcm2835_i2c_begin())
                {
        printf("bcm2835_i2c_begin failed. Are you running as root??\n");
                }
        bcm2835_i2c_setClockDivider(clk_div);  // Division de l'horloge Rpi pour obtenir une vitesse de 100KHz
}
int main(void)
{
        //double mesure;
        unsigned int slave_address=0x08; // Adresse du DS1621
        char i2cOut[3];     // Tableau des données I2C à sortir
        char i2cIn[8];                  // Tableau des données I2C entrées
        init_I2c_bcm2835();
        bcm2835_i2c_setSlaveAddress(slave_address);       // Affectation de l'adresse de l'esclave
//      i2cOut[0] = 0xAC;
        //i2cOut[1] = 0x00;
        //bcm2835_i2c_write(i2cOut, 2);     // Commenter
        //i2cOut[0] = 0xEE;
        //bcm2835_i2c_write(i2cOut, 1);     // Commenter
        delay(1000);                                            // 1 seconde avant première lecture
        while(1)
                {
//              i2cOut[0] = 0xAA;                               // Commenter
        //      bcm2835_i2c_write(i2cOut,1);        // Commenter
                bcm2835_i2c_read(i2cIn,6);
                //cout<<"i2cIn[0] = "<< (int)i2cIn[0]<< "      i2cIn[1] = "<< (int)i2cIn[1]<<endl;
//              if (i2cIn[1]!=0) mesure = (double)i2cIn[0] + 0.5;          // Commenter

        //      else mesure = (double)i2cIn[0];
                cout<<"valeur reçue :
 "<<i2cIn[0]<<i2cIn[1]<<i2cIn[2]<<i2cIn[3]<<i2cIn[4]<<i2cIn[5]<<i2cIn[6]<<i2cIn[7]<<endl;
                delay(1000);                            // 1 seconde entre chaque conversion
                }
        bcm2835_close();
        return 0;
}
**********************************************************************************************************************
```

L'Arduino envoie un « hello » à la Raspberry.

REVUE DE PROJET N°1 IR21

LIARD Guillaume

1. Présentation générale de mes tâches

1.1 Présentation

Pour la réalisation du projet Rushball, on m'a attribué la réalisation des tâches concernant le développement WEB.

J'ai pour travail de développer un site WEB permettant la configuration d'une partie de jeu (nombre de joueurs, nom des joueurs, choix du mode de jeu, choix des différents paramètres de la partie, …) et de visualiser en temps réel le déroulement de la partie. Je suis également en charge de la réalisation d'un serveur WEB sur la Raspberry PI permettant d'héberger ce site WEB en local, ainsi que de la réalisation de la base de données permettant stocker en mémoire certaines données (identifiant et mot de passe du ou des propriétaires, scores des précédentes parties, …) mais aussi de transmettre et récupérer toute données nécessaires lors d'une partie.

Pour débuter dans ce projet, j'ai décidé de commencer par réaliser une première maquette du site WEB. J'ai fait ce choix afin de me permettre d'avoir un premier support illustré me permettant de visualiser dans un premier temps l'aspect graphique vers lequel je vais m'orienter pour la suite du développement de ce site, mais aussi de bien me rendre compte de tous les paramétrages d'une partie et toutes autres données que je vais devoir intégrer à la base de données.

1.2 Planification

	Evaluations du projet	**Lun 10/02/20**	**Lun 13/07/20**	
	Evaluations du projet	Lun 10/02/20	Lun 10/02/20	
	Evaluations du projet	Jeu 18/06/20	Jeu 18/06/20	
	Revue 1	Lun 17/02/20	Mar 25/02/20	16
	Revue 2	Lun 18/05/20	Mar 26/05/20	11
	Remise du dossier	Jeu 28/05/20	Mar 02/06/20	20
	Soutenance finale	Lun 29/06/20	Lun 13/07/20	18
	Epreuve E4	Jeu 14/05/20	Mar 19/05/20	
	Epreuve E1	Mar 12/05/20	Mar 12/05/20	
	Vacances d'hiver	Mar 25/02/20	Mar 17/03/20	4
	Vacances printemps	Mar 21/04/20	Jeu 14/05/20	19;17
	Projet	**Lun 06/01/20**	**Jeu 25/06/20**	
	Projet	Lun 06/01/20	Lun 06/01/20	
	Projet	Jeu 18/06/20	Jeu 18/06/20	
	Specifications générales	Lun 06/01/20	Mar 07/01/20	
	Essais et validation des structures (prototypage rapide)	Mar 07/01/20	Jeu 13/02/20	15
	Prototypage rapide et routage	Jeu 19/03/20	Mar 21/04/20	10
	Fabrication et essais	Lun 18/05/20	Jeu 25/06/20	11
	Fichier de fabrication PCB finalisés	Lun 06/04/20	Jeu 09/04/20	
	Rédaction du dossier	Jeu 09/01/20	Mer 27/05/20	
	1er BTS Blanc	Mar 11/02/20	Jeu 13/02/20	
	2nd BTS Blanc	Mar 07/04/20	Lun 13/04/20	
		Lun 13/01/20	Jeu 16/01/20	
	Premier prototype du site web	Lun 06/01/20	Lun 27/01/20	
	Création de la base de données	Lun 27/01/20	Jeu 27/02/20	
	Mise en place du serveur sur la Raspberry PI	Jeu 27/02/20	Jeu 05/03/20	
	finalisation du site web	Jeu 05/03/20	Jeu 14/05/20	

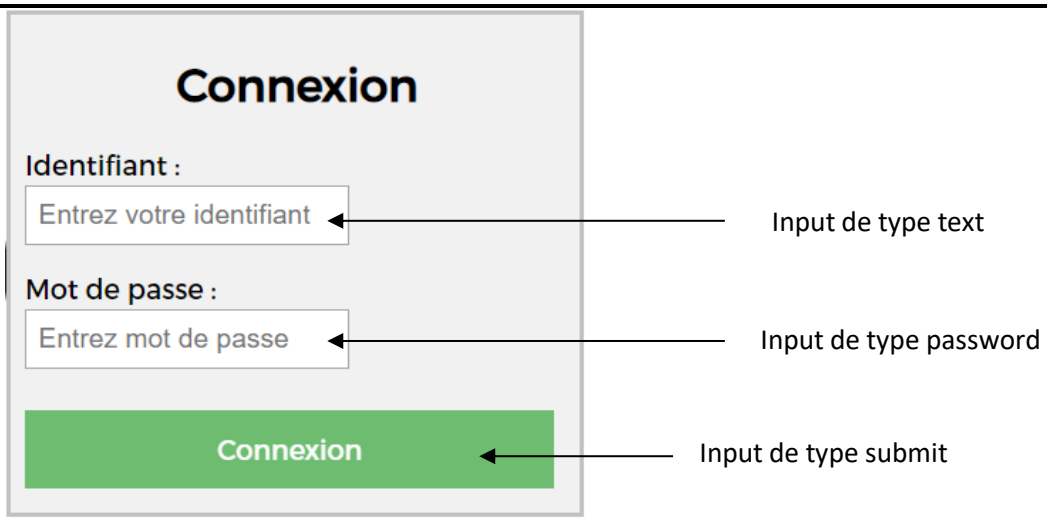
2. Présentation de mon avancement dans le projet

J'ai donc commencé par développer une maquette du site WEB. Pour cela j'ai utilisé trois technologies, Le HTML me permettant de réaliser la structure principale de la page, le CSS pour styliser le code HTML, et le JavaScript pour rendre le site dynamique et donc permettre à l'utilisateur d'interagir avec.

2.1 L'index, page d'accueil du site

Vu qu'il ne s'agit pour l'instant que d'une maquette, je ne me suis que peu attardé sur l'aspect visuel de cette page. Pour l'instant elle ne contient que le logo du Rushball situé en haut à gauche ainsi qu'un lien « Jouer » permettant de continuer sur la suite du site.

Un Pop-Up qui permettra par la suite de s'identifier sur le site, apparait au centre de la page lors de son chargement. Pour la réalisation de ce formulaire de connexion j'ai utilisé la balise html <form> permettant d'entrer plusieurs données dans des balises <input> variable en fonction des informations que l'on souhaite récupérer (ici ils sont de type text pour l'identifiant et de type password pour le mot de passe) et de les exploiter par la suite à l'aide d'un input de type submit.



Code du formulaire :

```
<div class="PopUpForm">
      <form id="PopUpConnexion">
          <h1 id="h1Connexion">Connexion</h1>

          <label class="labelConnexion" for="identifiant">Identifiant :</label>
          <input class="inputConnexion" type="text" placeholder="Entrez votre identifiant" name="identifiant" required >

          <label class="labelConnexion" for="psw">Mot de passe :</label>
          <input class="inputConnexion"type="password" placeholder="Entrez mot de passe" name="mdp" required >

          <button type="submit" class="btn" id="btnConnexion">Connexion</button>
      </form>
  </div>
```

Un script permet de faire disparaitre ce Pop-Up lors de l'appuie sur le bouton connexion une fois que les informations ont été saisies.

```
btnConnexion.addEventListener( "click", function() {

    PopUpConnexion.style.display ="none";
});
```

Ce script, à la détection de l'événement « click » sur la balise à la quelle est assigné l'id btnConnexion, va déclencher une fonction qui va changer l'attribut CSS display de la balise assignée à l'id PopUpConnexion en none.

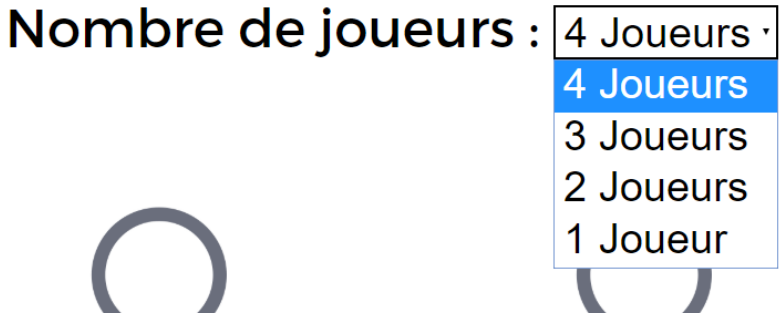
2.2 La page de sélection des joueurs

La seconde page, permet d'indiquer le nombre de joueurs ainsi que leur pseudo.

Il s'agit ici aussi d'une utilisation de la balise HTML <form>, contenant cette fois ci une balise <select>, quatre balises <input> de type text et une balise <input> de type submit.

```
<form action="selectionMode.html">
    <div class="nombreJoueur">
        Nombre de joueurs :
        <select name="nombreJoueur" class="choixJoueur" id="selectJoueur">
            <option value="4" class="nombres">4 Joueurs</option>
            <option value="3" class="nombres">3 Joueurs</option>
            <option value="2" class="nombres">2 Joueurs</option>
            <option value="1" class="nombres">1 Joueur</option>
        </select>
    </div>
    <div id="blockJoueurs">
        <div class="FormJoueur" id="Joueur1">
            <img class="perso" src="Images/perso.png"/>
            <input name="NomJoueur1" type="text" maxlength="13" placeholder="Nom du joueur 1" class="champDeSaisieJoueur"></input>
        </div>

        <div class="FormJoueur" id="Joueur2">
            <img class="perso" src="Images/perso.png"/>
            <input name="NomJoueur2" type="text" maxlength="13" placeholder="Nom du joueur 2" class="champDeSaisieJoueur"></input>
        </div>

        <div class="FormJoueur" id="Joueur3">
            <img class="perso" src="Images/perso.png"/>
            <input name="NomJoueur3" type="text" maxlength="13" placeholder="Nom du joueur 3" class="champDeSaisieJoueur"></input>
        </div>

        <div class="FormJoueur" id="Joueur4">
            <img class="perso" src="Images/perso.png"/>
            <input name="NomJoueur3" type="text" maxlength="13" placeholder="Nom du joueur 4" class="champDeSaisieJoueur"></input>
        </div>
    </div>
    <input type="submit" id="buttonValidationJoueur" value="Confirmer"></input>
</form>
```

La balise <select> permet à l'utilisateur de faire un choix entre plusieurs options à l'aide d'un menu déroulant. Ici il est possible de choisir le nombre de joueurs d'un à quatre.



J'ai aussi réalisé un script permettant d'afficher le nombre de joueurs en fonction de la valeur choisie dans le select.

```
window.addEventListener( "load", function() {

    let blocJoueur2 = document.getElementById("Joueur2");
    let blocJoueur3 = document.getElementById("Joueur3");
    let blocJoueur4 = document.getElementById("Joueur4");

    let selectJoueur = document.getElementById("selectJoueur");
    selectJoueur.addEventListener( "change", function(){
    console.log( "on change la value du select" );
    let selectedValue = parseInt( this.value );

      blocJoueur2.style.display = ( selectedValue >= 2 ? "inline-block" : "none" );
      blocJoueur3.style.display = ( selectedValue >= 3 ? "inline-block" : "none" );
      blocJoueur4.style.display = ( selectedValue >= 4 ? "inline-block" : "none" );
  });

});
```

Ce script va détecter avec l'événement « change » le changement de valeur de la balise <select>. En fonction de cette valeur on va effectuer des tests sur les différents blocs de joueurs (ces blocs contiennent l'image du personnage et le champ de saisie du pseudo), selon si cette valeur respect ou non la condition du teste on va assigner au bloc du joueur l'attribut CSS inline-block ou none, ce qui permettra de l'afficher ou de le rendre invisible.

2.3 La page de sélection de mode

La troisième page permet de sélectionner le mode de jeu. Trois modes de jeu sont disponibles, dont deux avec des variantes, se référer à la page « règles du jeu ». Cette page est composée de cinq liens menant en fonction du choix du mode et de sa variante vers les pages de paramétrage de la partie correspondante.

Les annotations 1 et 2 sont provisoires et seront par la suite remplacées par des images permettant d'illustrer la variante du mode.

Afin de permettre à l'utilisateur de prendre connaissance des règles de chaque mode, J'ai utilisé le sélecteur CSS hover, celui-ci permet de modifier certains attributs CCS quand l'on passe la sourie la balise définie. Dans le cas présent, quand l'on passe la sourie sur le bloc d'un mode (image du mode, nom du mode, et liens) il s'affiche en bas de page les règles de jeu du mode en question.

Les règles du mode "Contre la montre" c'est très simple, Il suffit de marqué un maximun de point dans un temps imparti. Jouable de un à quatre joueurs.
Il est possible de paramétrer la durée de la partie, ainsi que les nombres de cible et les points qu'elles apportent.

1-Lorsqu'une CIBLE est touchée, elle disparaît et réapparaît à un autre emplacement.
2-Toutes les CIBLES sont allumées et ne changent pas d'emplacement lorsqu'elles sont touchées.

Code CSS utilisé :

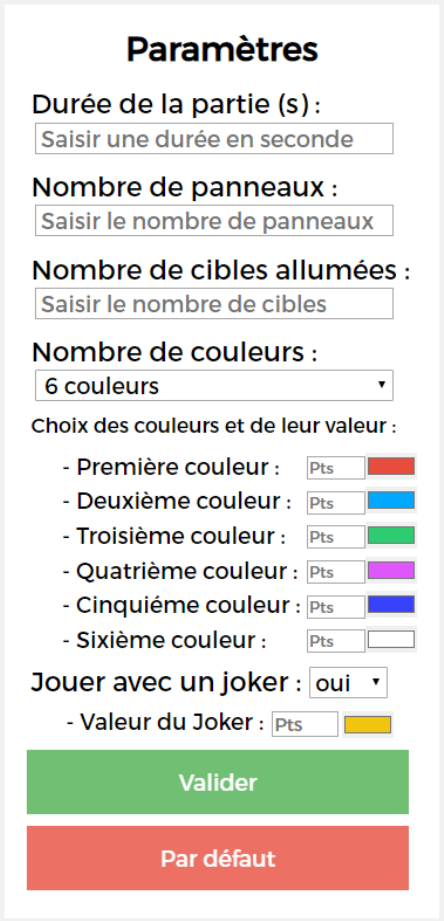
```css
.regles {
    font-size: 1.5vw;
    font-family: 'Montserrat', sans-serif;
    border: 5px solid #c9c9c9;
    margin-top: 6%;
    width: 90%;
    padding: 1%;
    display: none;
    position: absolute;
    float: left;
}

#blockTime:hover #reglesTime {
    display: inline-block;
    margin-left: -9%;
}

#blockScore:hover #reglesScore {
    display: inline-block;
    margin-left: -38%;
}

#blockElite:hover #reglesElite {
    display: inline-block;
    margin-left: -67%;
}
```

La classe « regles » possède de base un certain nombre d'attribut CSS, dont un display none permettant de ne pas être visible tant que la sourie n'est pas positionné sur le bloc du mode, et une position absolute qui permet de positionner les trois blocs de règles au même endroit.

Avec les sélecteurs hover on va modifier le display en inline-block afin de rendre le bloc visible, mais aussi la marge afin de le centrer

2.4 La page de paramétrage de la partie

Cette page permet deux choses, Paramétrer une partie avant son lancement et par la suite de suivre son déroulement en temps réel. Pour le moment seul la partie paramétrage a était mise en place.



Comme précédemment J'ai utilisé une balise <form> avec différents <input> et <select> pour réaliser cette interface de paramétrage. Les options proposées varient en fonction du mode de jeu sélectionné. Ici par exemple, on a sélectionné le mode « contre la montre », on nous propose donc de choisir la durée de la partie.

De la même manière que pour les précédents <select> j'ai réalisé un script permettant d'afficher ou non certaine balise en fonction de la valeur du select. Cette fonctionnalité est utilisé pour le nombre de couleurs sélectionnées ainsi que pour le choix d'une partie avec ou sans joker.

Code pour le nombre de couleurs :

```
let blocCouleur2 = document.getElementById("C2");
let blocCouleur3 = document.getElementById("C3");
let blocCouleur4 = document.getElementById("C4");
let blocCouleur5 = document.getElementById("C5");
let blocCouleur6 = document.getElementById("C6");

let selectColors = document.getElementById("selectColors");
selectColors.addEventListener( "change", function(){
console.log( "on change la value du select" );
let selectedValueColors = parseInt( this.value );

      blocCouleur2.style.display = ( selectedValueColors >= 2 ? "block" : "none" );
      blocCouleur3.style.display = ( selectedValueColors >= 3 ? "block" : "none" );
      blocCouleur4.style.display = ( selectedValueColors >= 4 ? "block" : "none" );
      blocCouleur5.style.display = ( selectedValueColors >= 5 ? "block" : "none" );
      blocCouleur6.style.display = ( selectedValueColors >= 6 ? "block" : "none" );
});
```

Code pour le joker :

```
let ParametreJoker = document.getElementById("ParamJoker");
let selectJoker = document.getElementById("selectJoker");

selectJoker.addEventListener("change", function(){
let selectValueJoker = parseInt( this.value );

     ParametreJoker.style.display = ( selectValueJoker == 1 ? "block" : "none");

});
```

C'est deux programmes fonctionnent de la même manière que le précédent script utilisé pour la sélection des joueurs.

3. Objectifs à réaliser pour la suite du projet

Pour la suite du projet je compte effectuer dans l'ordre suivant :

-Finaliser le code HTML.

-Transformer mes codes JavaScript en fonction pour faire une seule page JavaScript pour tous les pages HTML.

-Réaliser la base de données.

-Réaliser le serveur Apache sur la Raspberry PI.

-Gérer la transmission des données via la base de données.

REVUE DE PROJET N°1 IR22

SEURRE Mathieu

1. Objectifs

Réaliser un logiciel PC qui paramètre le début de la partie RushBall

2. Paramètres à envoyer :

-Nombre de joueur + pseudonyme du joueur (max 13 caractères)

-Choix du mode de jeu (cf. **Règles du Jeu**)

-Nombre de panneaux

-Nombre de cibles allumées

-Choix des couleurs + nombre de points des couleurs

-Choix du Joker + nombre de points du Joker

3. Réalisation supplémentaire :

-Affichage en temps réel de la partie de RushBall

4. Avancement du Projet :

Partie 1 : Communiquer en WIFI par protocole TCP/IP sur la Raspberry

Partie 2 : Réaliser un croquis de l'interface de l'application et faire le design final sur le GUI de Microsoft Visual Studio

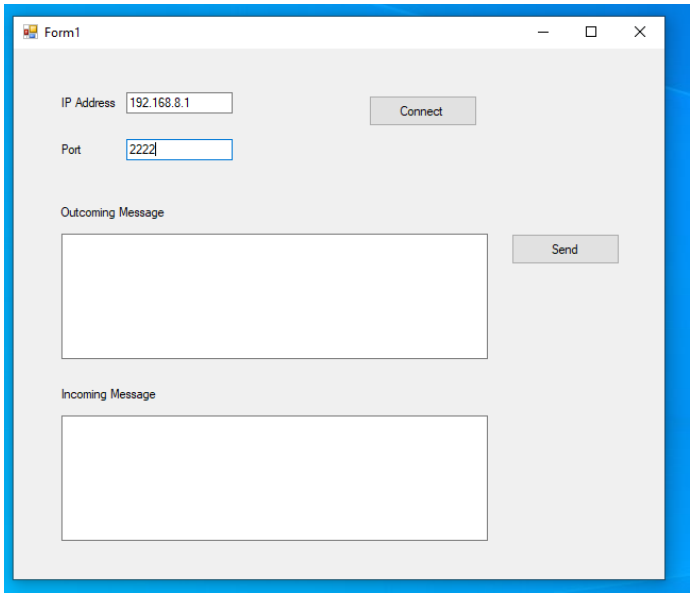
Partie 3 : Coder la cohérence de l'interface de l'application

Partie 4 : Ecriture et envoi de la trame sur la Raspberry

5. Application protocole TCP/IP Client :

L'objectif final de ma partie dans le projet est d'arriver à envoyer une trame sur la Raspberry de tous les paramètres choisis par le joueur dans le logiciel qui sera ensuite décomposé par mon collègue Léo Scholl.

J'ai donc dans un premier temps réalisé une application qui communique avec la Raspberry grâce à un protocole TCP/IP en wifi avant d'avoir commencé à créer l'interface du logiciel RushBall.



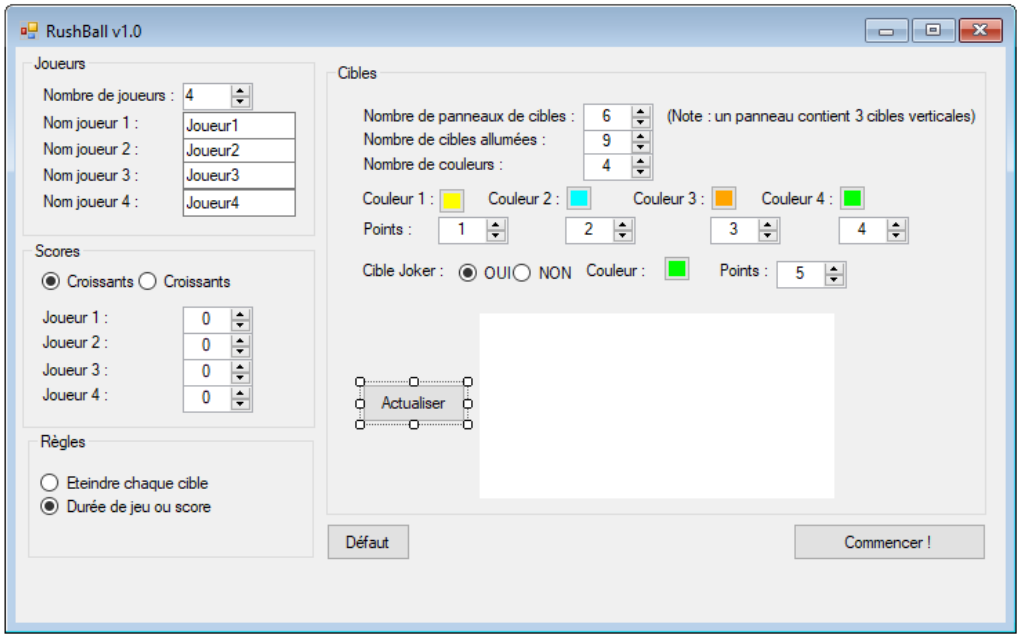
L'interface me permet d'illustrer le fonctionnement du logiciel. Je ne l'afficherai pas sur le produit final puisqu'il n'y a pas besoin de renseigner l'adresse IP ni le Port : l'adresse 192.168.8.1 et le port 2222 sont fixes.

A la fin du paramétrage de l'utilisateur du logiciel de RushBall, la trame sera envoyée (cf **Protocole Rushball**) sur la Raspberry : ici dans *Outcoming Message.*

Durant la partie du RushBall, je vais devoir afficher en temps réel les noms, scores des joueurs ainsi que l'affichage des panneaux. La Raspberry devra donc m'envoyer également une trame de la partie en cours : ici reçu dans *Incoming Message.*

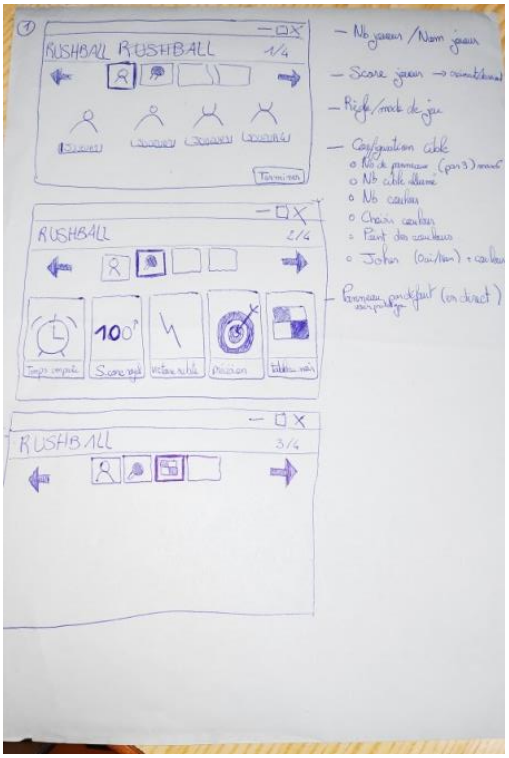
6. Application RushBall :

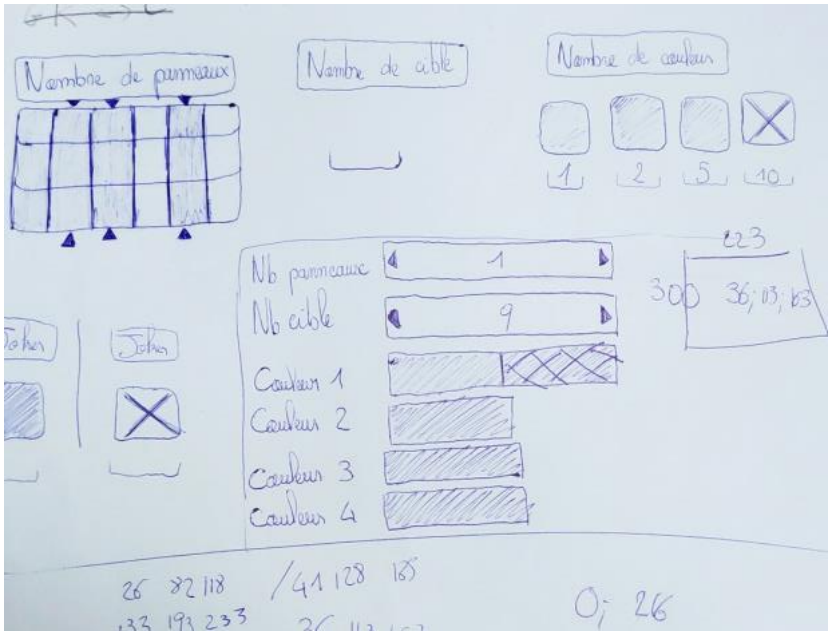
Mon professeur référent a construit un prototype du logiciel qui, potentiellement, pouvait ressembler à l'objectif final avec la cohérence de l'interface.



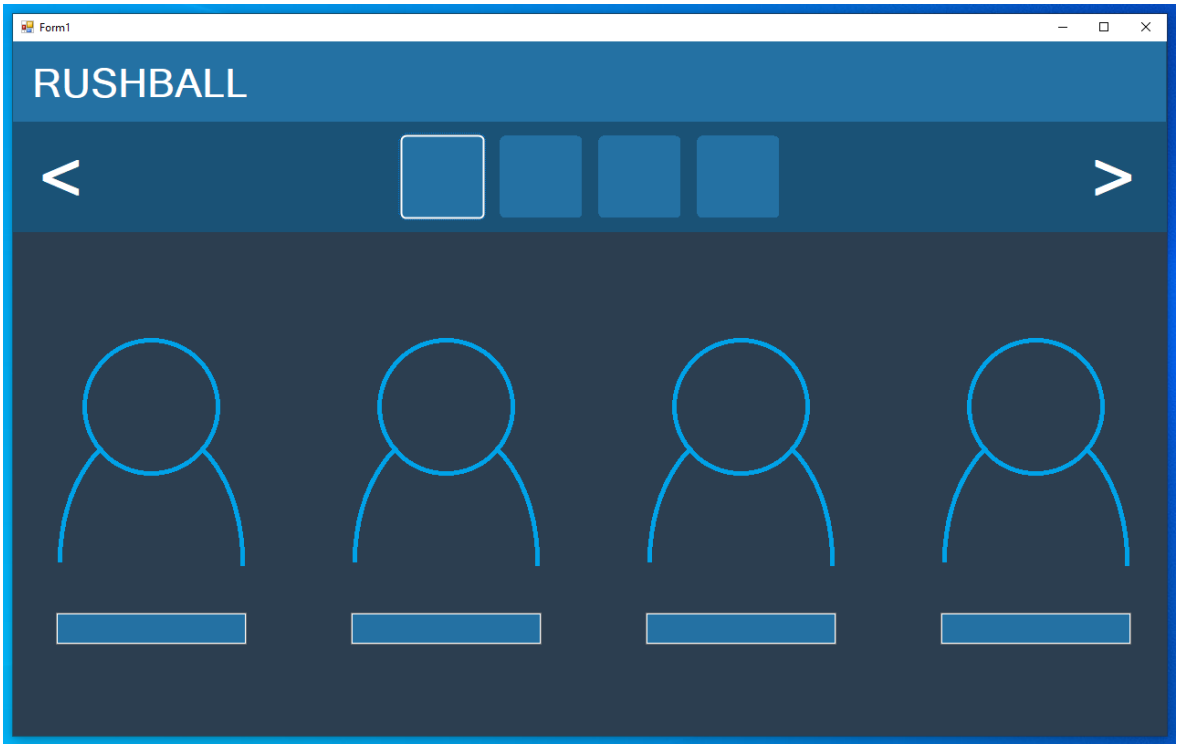
Cependant l'interface de Microsoft Visual Studio me déplaît beaucoup. De plus le RushBall est un sport qui se joue à tout âge et donc n'importe quel joueur doit être capable de paramétrer le jeu sans pour autant être perdu au milieu de tous ces boutons. J'ai donc voulu faire l'interface la plus intuitive possible de par sa simplicité et par son interface plus ludique au visuel.

Je me suis renseigné sur le flat design et me suis inspiré d'autres interfaces d'application. Avant de commencer, j'ai dessiné un croquis sur feuille





7. Menu Utilisateur :



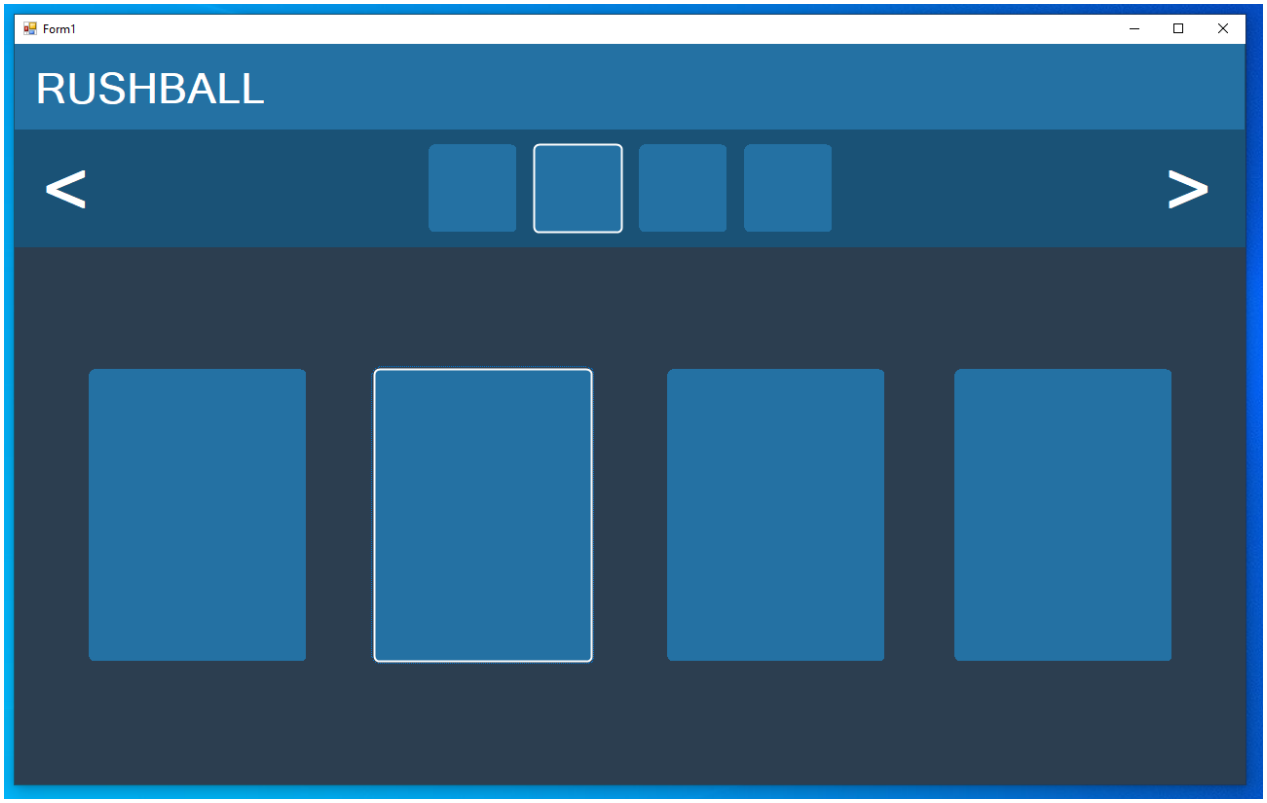
Le menu Utilisateur est la première interface qui apparaît lorsque le logiciel est lancé. L'utilisateur pourra choisir le nombre de joueur. Il inscrit le(s) pseudonyme(s) (max 13 caractères) dans le champ de saisie.

L'interface est très simplifiée de manière à ce qu'on comprenne facilement ce qu'il faut faire pour paramétrer le jeu. J'aiderai également l'utilisateur à avancer dans les menus lorsqu'il aura terminé sa saisie en affichant une aide par exemple ou par un bouton suivant mis en évidence.

On peut également cliquer sur les menus en haut au milieu. Je n'ai pas encore décidé des icônes à insérer.

8. Menu Mode de Jeu :

On choisira le mode de Jeu dans ce menu, il y aura également des images à l'intérieur des menus principaux et une aide qui explique les règles du jeu (cf **Règles du Jeu**).



Comme dans la documentation « Règles du Jeu » il y aura en dessous du mode de Jeu principal ses deux variantes. Le choix du mode de jeu modifie le menu Configuration :

9. Menu Configuration :

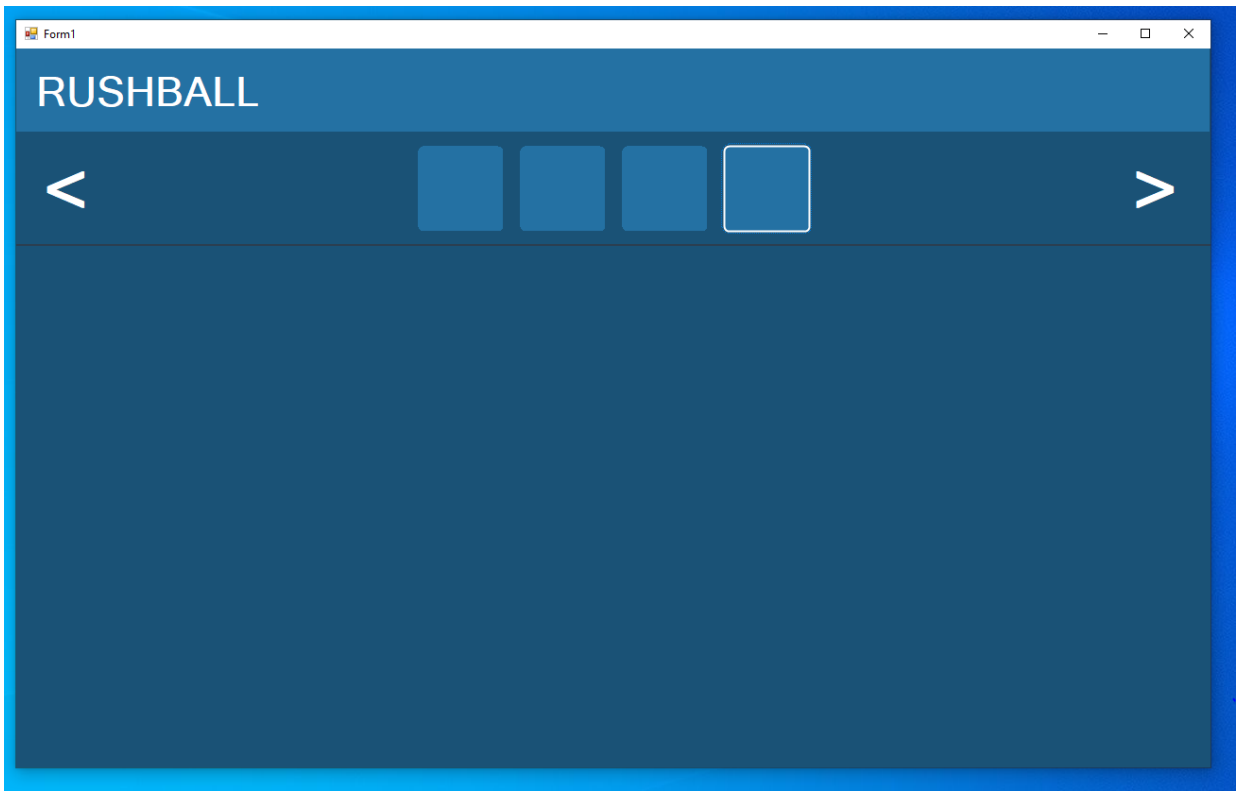
L'interface est encore loin d'être terminée. L'utilisateur pourra choisir le nombre de panneaux, puis le nombre de cibles.

J'ai été confronté à une problématique pour le choix des couleurs. Initialement l'utilisateur doit pouvoir choisir la couleur des cibles, cependant le colorDialog du GUI (palette de couleur de base de Windows) donne un choix beaucoup trop important.

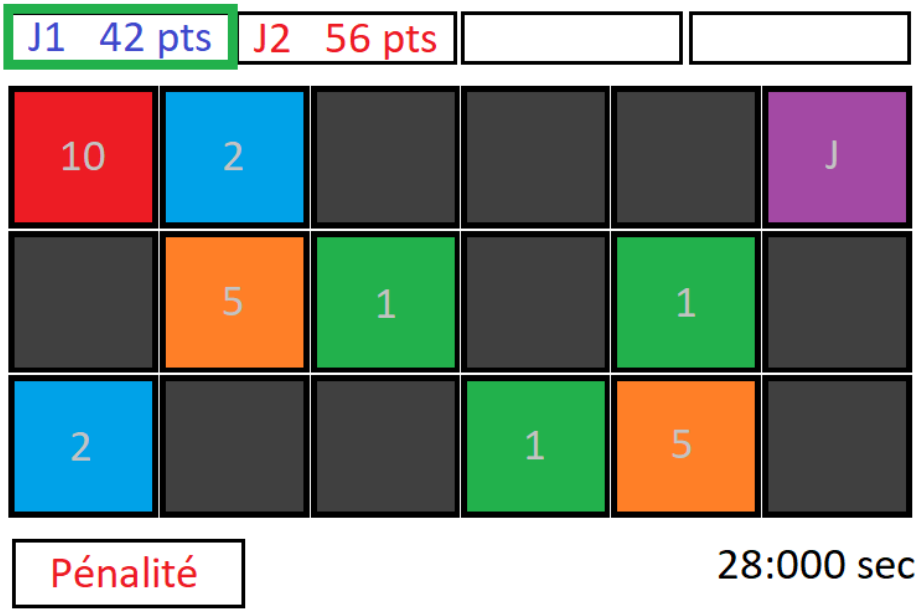
Le RushBall se joue avec 6 couleurs (plus le Joker). Pour faciliter l'écriture de la trame j'ai tout simplement mis toutes les couleurs disponibles puis l'utilisateur devra cliquer sur la couleur pour la désactiver comme dans l'exemple Menu Configuration.

10. Menu Temps réel :

Je n'ai pas commencé cette partie, elle sera consacrée à l'affichage en temps réel.



Pour cette partie, j'ai pour objectif de rendre une interface comme celle-ci :



Avec le nombre de points de la couleur sur la cible, les scores des joueurs et le Temps / Score en bas à droite selon le mode de jeu choisi.