Phase de réalisation

Ecole supérieure
Électronique

Laboratoire R110 PROJ

Buzzer Wire Game

Réalisé par :

Santiago Valiante

A l'attention de :

Mr. Bovey et Mr. Castoldi

Dates:

Début du laboratoire : 14 Décembre 2022
Fin du laboratoire : 26 Janvier 2023

Table des matières

Cahier des charges

Voir le cahier des charges détaillé dans le répertoire :

K:\ES\PROJETS\SLO\2224_BuzzerWireGame\doc

Nom du fichier : 2224_BuzzerWireGame-CDC-v1.doc.

Pré-étude

Résumé du projet

Principe

WireGame est un jeu qui consiste à guider une boucle métallique le long d'une longueur de fil en serpentin sans toucher la boucle au fil. La boucle et le fil sont connectés à une source d'alimentation de telle manière que, s'ils se touchent, ils forment un circuit électrique fermé.

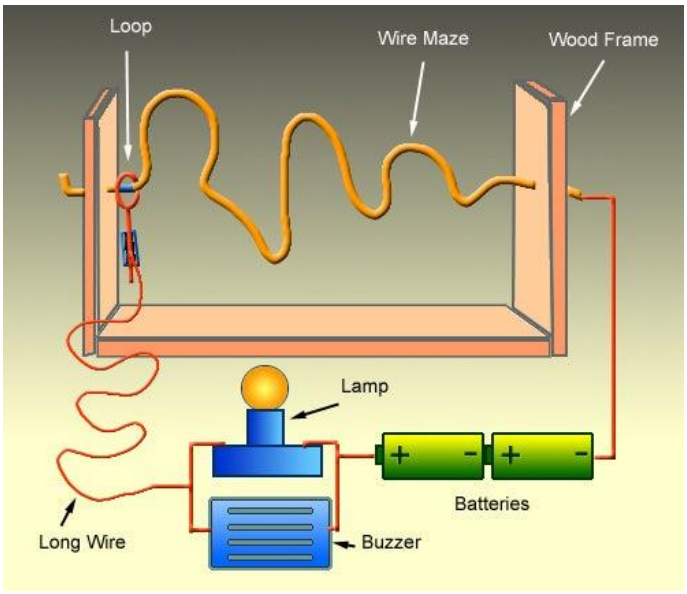


Figure 1

But

Le but de ce projet est de développer un WireGame autonome et optimisé pour qu'une personne mal entendant puisse jouer. Il sera possible de gérer le système à travers une application C# depuis un PC connecter en USB ou manuellement avec l'encodeur.

Schéma bloc global

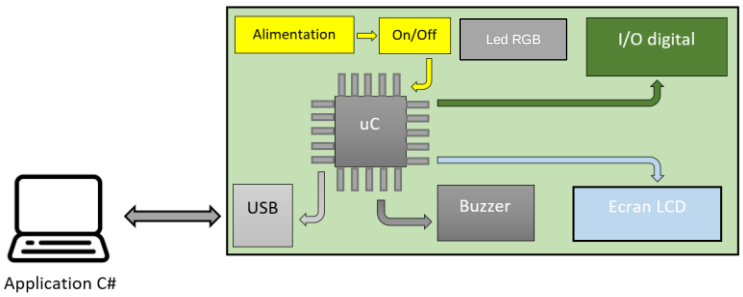
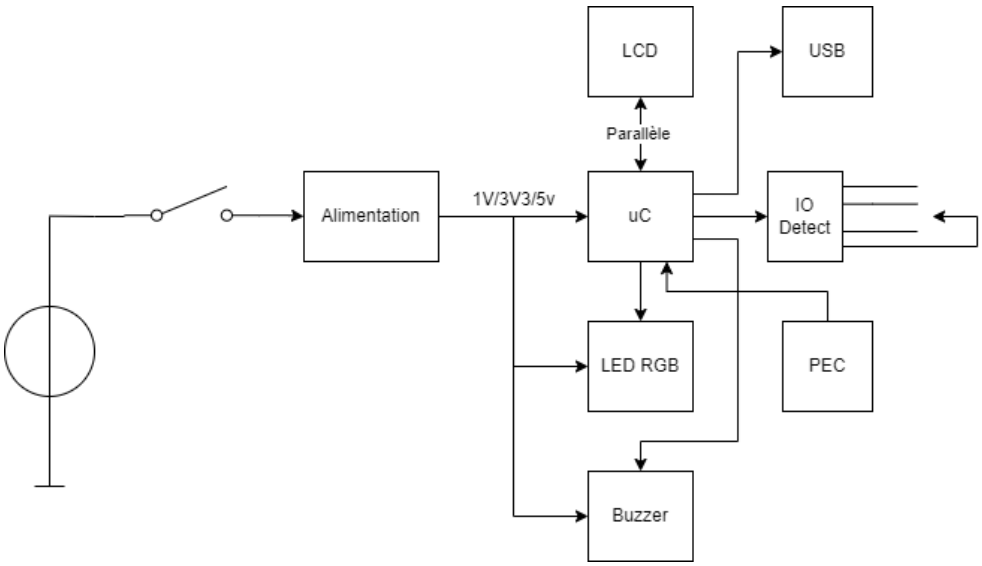


Figure 2

Composition du projet

Schéma bloc Hardware



Hardware :

Alimentation

Le projet devra être alimenté avec une pile, ensuite il y aura un bouton ON/OFF pour gérer l'alimentation.

uC

Utilisation d'un pic 32 (Imposé par le client) pour la gestion de la communication USB, du LCD, des I/O et ect.

LCD

L'affichage LCD sera utilisé afin d'indiquer le nombre de fois que l'on a touché le fil ainsi que la durée du parcours. De plus, afficher le nom du joueur et d'interfacer le menu du jeu.

Port USB

Utilisation d'un port USB afin de communiquer avec un PC, pour que celui-ci prenne le lead sur la gestion du menu et du mode de jeu. De plus faire la gestion de statistiques (meilleurs scores avec nom du joueur).

I/O detect

Les I/O seront utilisées afin de gérer le départ, l'arrivée et le toucher de la boucle.

Start, touch et end :

Pour savoir quand le jeu débute, se finit ou que l'on touche le fil de cuivre deux idées me sont venues en tête.

- **1ère idée :**
  Grâce au microcontrôleur, il faudra détecter le changement d'état de 3 entrées. Une entrée pour le Start. Cela va servir à détecter la position du joueur (qu'il soit bien au départ du jeu) et détecter quand il commence. Une deuxième pour l'arrivée afin de détecter la fin du jeu. Et pour finir, une entrée pour le fil du parcours afin de détecter si on a touché le fil. Des résistances pull-up seront aux entrées et la manette sera reliée au GND afin de mettre ces entrées à 0.

- **2ème idée :**
  Avoir différentes valeurs résistives sur les trois parties afin de les différencier.

Buzzer

Utiliser un buzzer comme aspect sonore lors d'un toucher.

Leds RGB

Les leds RGB serviront comme aspect visuel lors d'un toucher. Le nombre de led sera déterminé avec des tests pour l'éclairage d'un plexi glace opaque. Il faut en compter environ deux.

Encodeur

Utilisation d'un encodeur pour pouvoir naviguer dans le menu.

Boîtier

Pour le boîtier, nous avons l'idée de faire une boîte en plexi glace avec un socle en bois. Il faudra faire les plans du boîtier sur Solidworks.

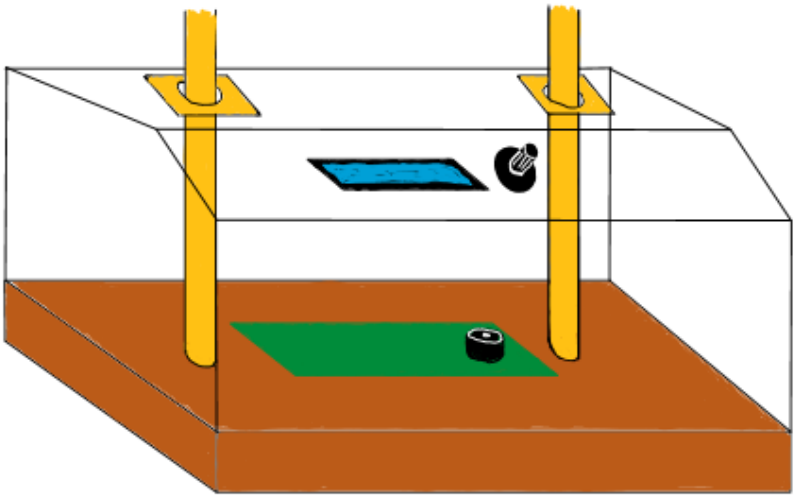


Figure 3

Représentation et explications des interactions externes

Le joueur pourra interagir directement avec le système. Il y aura un bouton ON/OFF pour activer ou désactiver l'alimentation. A l'aide d'un affichage LCD et d'un encodeur, il pourra naviguer dans le menu. Mais, il pourra également interagir avec le système via un PC connecté via un port USB pour naviguer également dans le menu et en plus voir les statistiques des joueurs. Pour lancer le jeu, il suffira de décoller la boucle de la plaque de cuivre et de même pour y mettre fin mais à l'arrivée.

Voir les figures 2, 3, 4 et 5 pour un meilleur aperçu du menu et des interactions manuelles.

Choix de composant avec justification

Alimentation

La pile ou l'accumulateur sera choisie en fonction de la puissance que consommera le circuit qui est encore à déterminer avec les LEDs RGB (priorité). Je pourrais prendre par exemples 2 piles en séries de 3v de 2Ah pour avoir une alimentation assez élevée et fournir assez de courant pendant 2 à 3h. Il faudra également des régulateurs step down pour alimenter en 5v, en 3v3 et 1v mes divers périphériques.

Switch On Off

J'ai choisi un switch à bascule rond (RR11122FWC).

Microcontrôleur µC

Dans le cadre de mon projet, on m'a imposé d'utiliser le PIC32. Ce µC est déjà utilisé dans les différents projets du client et nous avons déjà de bonnes connaissances de bases du µC.

Dans mon cas, le microcontrôleur va gérer **4 différentes GPIO**, cela permettra de commander une ou des Leds RGB, un écran LCD, un encodeur et un buzzer. Il y aura besoin d'une **communication UART** afin de communiquer avec un PC via un convertisseur USB to UART.

Choix : PIC32MX130F064B

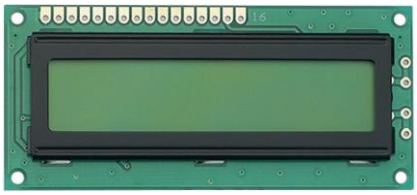
Device	Pins	Program Memory (KB)(1)	Data Memory (KB)	Remappable Peripherals					Analog Comparators	USB On-The-Go (OTG)	I²C	PMP	DMA Channels (Programmable/Dedicated)	CTMU	10-bit 1 Msps ADC (Channels)	RTCC	I/O Pins	JTAG	Packages
				Remappable Pins	Timers(2)/Capture/Compare	UART	SPI/I²S	External Interrupts(3)											
PIC32MX130F064B	28	64+3	16	20	5/5/5	2	2	5	3	N	2	Y	4/0	Y	10	Y	21	Y	SOIC, SSOP, SPDIP, QFN

Connecteur USB + USB to Uart

J'ai choisi de prendre un connecteur USB B coudé ainsi qu'utilisé un USB-to-UART Bridge Controller (CY7C64225-28PVXC) afin de convertir mon signal USB en UART pour communiquer depuis le PC avec mon µC. Le chip peut communiquer en Full Speed 12Mb/s.

Affichage LCD

Pour l'écran LCD, je l'ai choisi afin de gérer au mieux l'interface du menu. Pour ça, j'ai décidé de prendre un écran LCD 4 lignes avec 20 caractères. Backlight intégrer. J'ai choisi cet LCD car c'est un équivalent du LCD qu'on retrouve sur le Kit de l'ES. Il sera donc facile à l'interfacer. Communication parallèle.



Encodeur

J'ai choisi de prendre un encodeur (Pec12) qui a été lui aussi déjà utilisé lors de différents projets du client. Afin de naviguer dans le menu du jeu.

Bande LED RGB

Pour un aspect physique, nous avons décidé de mettre une bande led. Elle servira comme indication lors d'un touché et éclairer un plexiglass. J'ai trouvé une bande de led de 1m, fonctionnant en 5v. Elle consomme 9W au max si les 30 leds sont allumées. J'utiliserais seulement la couleur rouge comme indication lors d'un toucher et je ne pense pas l'allumé plus de 5 sec afin de ne pas vider la pile.

Buzzer

J'ai choisi un buzzer (CVS-2308) qui a été déjà utilisé dans un projet du client. Les décibels sont assez élevés (88db) à titre d'indication lors d'un toucher.

.

Evaluation des coûts

Nom	Quantité	Type	Fabriquant	N° Fabriquant	Fournisseur	Prix u	Total
Microcontrôleur	1	PIC32MX130F064B	Microchip	PIC32MX130F064B-I/SS	Mouser	fr. 3,55	fr. 3,55
Pile 3v 2 Ah	2	Lithium	Varta	CR AA	Distrelec	fr. 11,70	fr. 23,40
Connecteur USB	1	RND 205-01048	RND connect	RND 205-01048	Distrelec	fr. 0,47	fr. 0,47
USB-to-UART Bridge Controller	1	Bridge Controller	Infineon Technologies	CY7C64225-28PVXC	Mouser	fr. 4,28	fr. 4,28
Affichage LCD	1	DEM 20485 SYH	Display Elektronik GmbH	DEM 20485 SYH	Distrelec	fr. 22,83	fr. 22,83
Encodeur	1	PEC12	Bourns	PEC12R-4220F-S0024	Mouser	fr. 1,49	fr. 1,49
Bande LED RGB	1	LES RGB 9W 1m 5v	Seeed Studio	104020108	Distrelec	fr. 7,72	fr. 7,72
Buzzer	1	CVS-2308	CUI Device	CVS-2308	Mouser	fr. 3,00	fr. 3,00
PCB	1	-	Eurocircuit	-	Eurocircuit	fr. 60,00	fr. 60,00
Flis de cuivres 5mm	1	2m	-	-	Distrelec	fr. 10,00	fr. 10,00
Planche en bois	1	25X15 cm				fr. 5,00	fr. 5,00
Plexis	1					fr. 15,00	fr. 15,00
						Total	fr. 157,74

Planning

No semaine projet	1	2	3	4	5	6			7	8	9	10	11		12	13	14	15	16	17	18			19	20	21	22	23	24	25	26	27	28
No semaine	46	47	48	49	50	51	52	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
Date	16.11.22	23.11.22	30.11.22	07.12.22	14.12.22	21.12.22	28.12.22	04.01.23	11.01.23	18.01.23	25.01.23	01.02.23	08.02.23	15.02.23	22.02.23	01.03.23	08.03.23	15.03.23	22.03.23	29.03.23	05.04.23	12.04.23	19.04.23	26.04.23	03.05.23	10.05.23	17.05.23	24.05.23	31.05.23	07.06.23	14.06.23	21.06.23	28.06.23
Tâches							V	V						V								V	V									
Pré-étude				R	P																											
Design + Schéma											R	P																				
PCB																			R													
Montage																																
Softwares																																
Tests et mise au point																																
Rédaction du rapport																																
Préparation présentation + démo																																
Présentations finales																														R	P	
Finalisation/corrections/documentation																																

Commentaires

Rapport de pré-étude

Rapport de design

Fichiers de fabrication

Rapport final y compris annexes, fichiers, résumé, affiche

V : Vacances
R : Remise de documentation/dossier/fichiers. Peut être remis plus tôt.
P : Présentation.
Remise d'un rapport chaque semaine précédant une présentation.

Conclusion et perspective

Perspective

Je pense que les parties la plus compliquées du projet va être la partie software. En ce qui concerne la partie hardware du projet, je me fais moins de soucis.

Conclusion

Etant ma première pré-étude, cela a été compliqué pour moi de commencer et de se poser les bonnes questions.

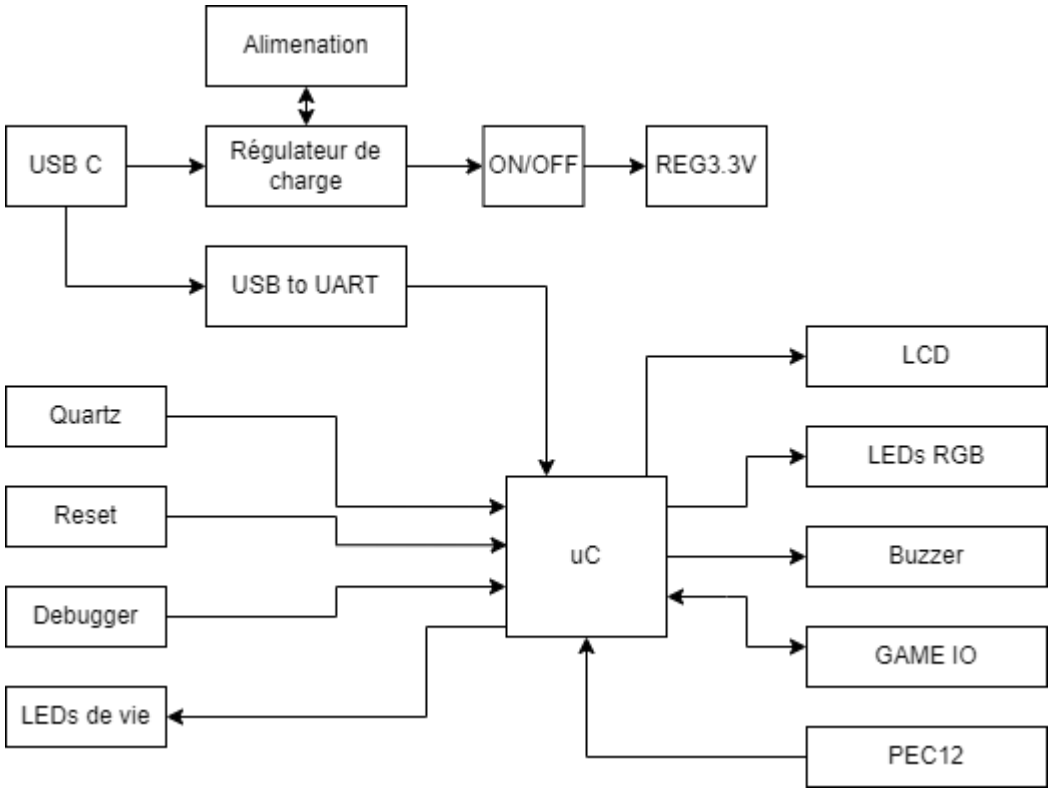
Il me faudra en premier lieu, calculer la puissance dissipée de mon système afin de déterminer mes piles pour l'alimentation de mon circuit.

Il me faudra également clarifier avec le client, le nombre de temps maximum que les LEDs RGB soient allumées lors d'un touché avec la boucle pour ne pas vider la pile si on la laisse pendant 1h en contact au fil de cuivre.

Ensuite, je vais devoir commencer par le design du schéma. Une fois cette étape réalisée, je vais être amené à faire le design du PCB, par la suite ceci me permettra de définir la taille finale du boitier. Ensuite, je vais être amené à gérer la communication entre mes différents modules par programmation.

Design

Schéma bloc



Alimentation

USB C

Afin de pouvoir communiquer avec un PC et recharger notre batterie Li-On. J'utilise un port USB C. Selon la note d'application (Figure 3), si nous sommes un device et que nous devons recharger notre système. Nous devons mettre deux résistances de 5k1 reliées à la masse aux sorties CC1 et CC2.

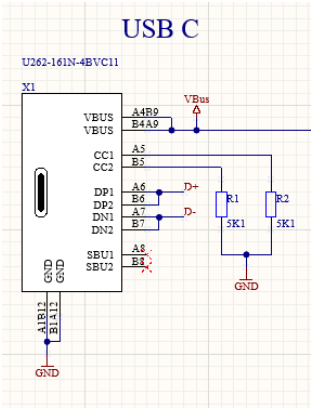


Figure 4 : Schéma USB C

3.2 UFP Rd Pull-Down Resistors.

An upstream facing port must connect a valid Rp pull-down resistor to GND (or optionally, a voltage clamp) to both CC1 and CC2 pins. A 5.1kΩ ± 10% is the only acceptable resistor if USB Type-C charging of 1.5A@5V or 3.0A@5V is to be used. The details are shown in the table below.

TABLE 7: VALID UFP RD PULL-DOWN RESISTOR VALUES

Rd Implementation	Nominal Value	Detect Power Capability?	Current Source to 1.7V - 5.5V
± 20% voltage clamp	1.1V	No	1.32V
± 20% resistor to GND	5.1kΩ	No	2.18V
± 10% resistor to GND	5.1kΩ	Yes	2.04V

Figure 5 : Annotation note USB C

Accumulateur Liion

J'ai choisi de prendre un accumulateur à la place de piles, car cela est plus économique. Dans le cahier des charges une autonomie de 2h minimum a été demandé. Après une estimation du courant, dans le pire des cas (les leds RGB sont allumées pour une animation), nous avons donc une consommation d'environ 500 mA.

$$Capacité = Autonomie * Consomation = 2 * 500m = 1000\ mAh$$

J'ai choisi de prendre le même accumulateur que mes camarades afin de faciliter la logistique. L'accumulateur a donc une capacité de 3400 mAh. J'aurais donc une autonomie de 6.8 h. Le prix de départ est plus cher qu'une pile, mais si nous visons sur la durée un accumulateur revient moins chers.

$$Autonomie = \frac{Capacité}{Consomation} = \frac{3400m}{500m} = 6.8\ h$$

Régulateur de charge et ON/OFF

Pour charger l'accumulateur, j'utilise un régulateur de charge **MCP73871**. Le modèle a été repris d'un design déjà utilisé à l'ETML-ES. J'utilise un switch afin de gérer l'alimentation de mon circuit.

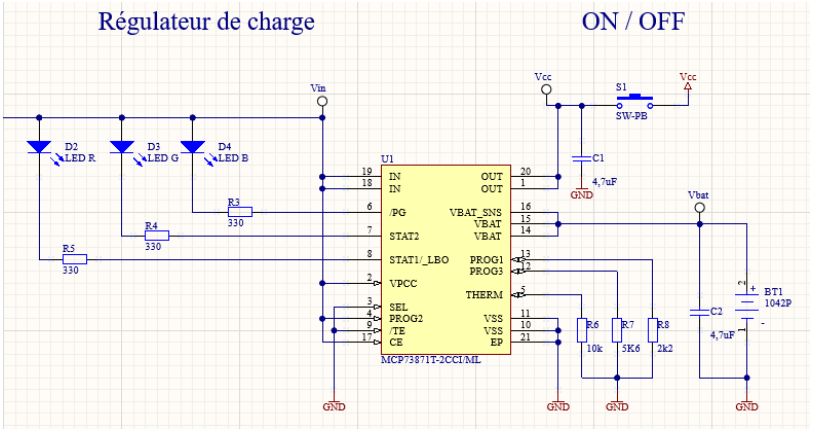


Figure 6 : Régulateur de charge et ON/OFF

Comme nous sommes en mode USB, la pin 3 (SEL) doit être mis à la masse.

3	SEL	I	Input type selection (low for USB port, high for AC-DC adapter)

Puis, la pin 9 (TE) sera mise à l'état bas pour activer la sécurité.

9	$\overline{TE}$	I	Timer Enable; Enables Safety Timer when active-low

Pour finir, la pin 17 (CE) à l'état haut pour activer la charge.

17	CE	I	Device Charge Enable; Enabled when CE = high

Pour les états de sortie, nous avons 3 LED's indicatives. Elles ne sont pas obligatoires, mais j'ai décidé de les mettre à titre d'indication pour connaître l'état de l'accu. Elles sont de couleurs vertes, bleues et rouges. Voir **Table 5-1**.

Etat des Leds

TABLE 5-1: STATUS OUTPUTS

CHARGE CYCLE STATE	STAT1	STAT2	$\overline{PG}$
Shutdown (V_{DD} = V_{BAT})	High-Z	High-Z	High-Z
Shutdown (V_{DD} = IN)	High-Z	High-Z	L
Shutdown (CE = L)	High-Z	High-Z	L
Preconditioning	L	High-Z	L
Constant Current	L	High-Z	L
Constant Voltage	L	High-Z	L
Charge Complete - Standby	High-Z	L	L
Temperature Fault	L	L	L
Timer Fault	L	L	L
Low Battery Output	L	High-Z	High-Z
No Battery Present	High-Z	High-Z	L
No Input Power Present	High-Z	High-Z	High-Z

Dimensionnement

Pour le dimensionnement des résistances et des condensateurs, il faut d'abord connaitre le courant maximum avec lequel on peut charger notre accumulateur (3400 mA). Les valeurs des condensateurs ont été prisent de l'exemple d'application (Figure 5). De même pour la résistance sur la sortie THERM.

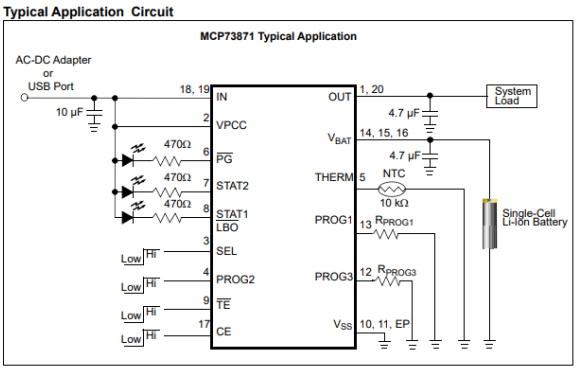


*Figure 7 : Exemple **MCP73871***

Courant de régulation :

Pour trouver la résistance Rprog1, il faut savoir le courant de régulation. Dans notre configuration, nous aurons un courant de 450 mA.

USB Fast Charge Current	I_{REG}	80	90	100	mA	PROG2 = Low, SEL = Low, (Note 2) T_A = -5°C to +55°C
		400	450	500	mA	PROG2 = High, SEL = Low, (Note 2) T_A = -5°C to +55°C

Figure 8: Configuration courant de charge

EQUATION 4-1:

$$I_{REG} = \frac{1000V}{R_{PROG1}}$$

Where:

R_{PROG} = kilo-ohms (kΩ)

I_{REG} = milliampere (mA)

$$Rprog1 = \frac{1000}{450m} = 2k2$$

Figure 9 : Formule Rprog1

Courant de charge de terminaison :

Pour trouver la résistance Rprog3, il faut savoir le courant de charge fixé depuis les informations de la batterie (3400 mA).

EQUATION 4-2:

$$I_{TERMINATION} = \frac{1000V}{R_{PROG3}}$$

Where:

R_{PROG} = kilo-ohms (kΩ)

I_{REG} = milliampere (mA)

Figure 10 : Formule Rprog3

$$Iter = 0{,}05C = 0{,}05 * 3400m = 170\ mA$$

$$Rprog3 = \frac{1000V}{170\ mA} = 5k8 => E24 = 5k6$$

Régulateur 3,3V

Comme on est alimenté par un accumulateur la tension de sortie de celui-ci peut varier au fur et à mesure du temps. Un étage de remises à niveaux est nécessaire pour garantir une tension de 3,3V sur le micro et les divers modules qui fonctionnent à cette tension. Le régulateur accepte une tension d'entrée entre 2.5V et 5.5V. J'ai repris un design de l'ETML-ES qui utilise le modèle MAX1793, il est notamment disponible dans les stocks de l'ES.

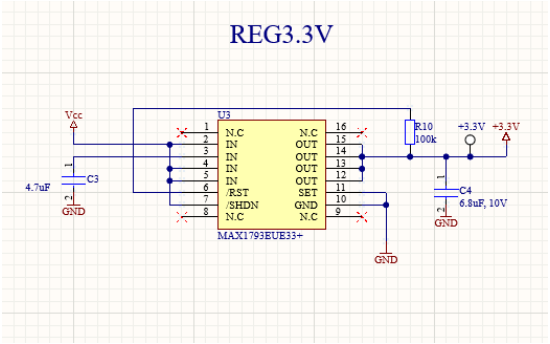


Figure 12: Régulateur 3.3V

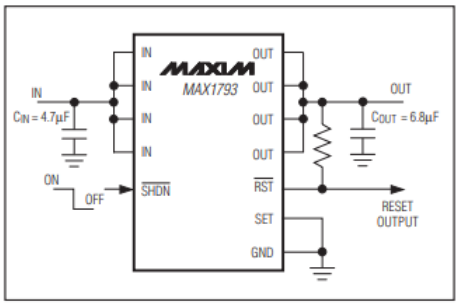


Figure 11 : Circuit typique Régulateur 3.3V

USB to UART

Afin de pouvoir se connecter au PC, on passera via un port USB C, il suffira de connecter les pines D- et D+ sur le connecteur. Il faudra lire les informations USB qui arriveront sur la carte. Pour cela un convertisseur USB vers UART sera utilisé.

Cela simplifie la prise en charge par le microcontrôleur de traiter une communication UART, plutôt qu'une communication USB lourde à traiter.

J'ai donc opté pour le composant « CY7C64225 » qui a déjà été utilisé lors d'un projet de l'ETML-ES. Pour le montage, un exemple du cas d'application alimenté en 3.3V a été suivi.

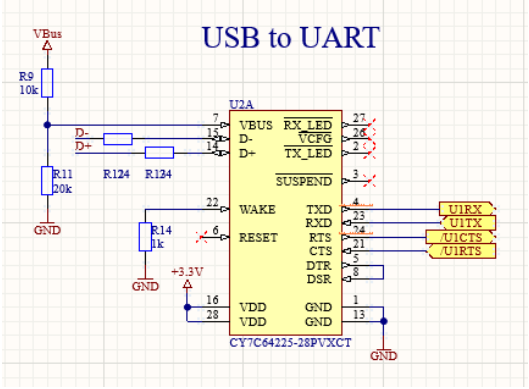


Figure 13: USB to UART

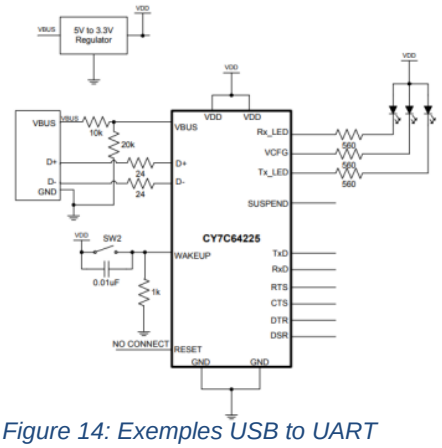


Figure 14: Exemples USB to UART

uC

Pour le choix du microcontrôleur, la famille PIC32 de Microchip a été conseillée dans le cahier des charges. Le 28 pin étant trop petit, j'ai donc choisi de prendre le PIC32MX130F256D à 44 pins, disponible dans le stock de l'ES.

Quartz

Le projet comprenant une communication UART. J'ai besoin d'un oscillateur pour avoir une vitesse en baud rate la plus précise possible.

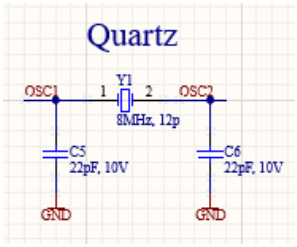


Figure 15: Quartz

Le calcul des condensateurs a été fait grâce au CL = 18pF et le C0 = 7pF du quartz utilisé.

$$C5 = C6 = 2 * (CL - C0) = 2 * (18 - 7) = 22pF$$

Reset

Afin de pouvoir reset le microcontrôleur lors de la phase de développement, un bouton poussoir a été placé. Le dimensionnement a été repris du kit micro utilisé au labo.

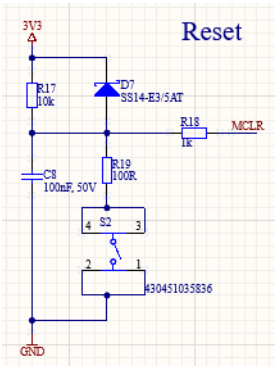


Figure 16: Reset

Port de programmation

Le débuggeur est là pour pouvoir programmer le microcontrôleur en chargeant le firmware à l'intérieur. Je pourrai également le connecter pour débugger en temps réel le firmware.

Il a été désigné pour accueillir un debuger « ICD3 ».

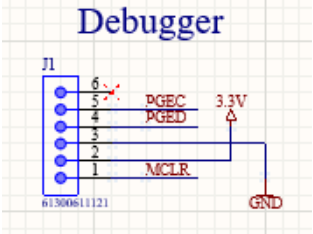


Figure 17 : Port de programmation

Périphériques

Game IO

Nous avons trois pull-up pour maintenir un état haut sur les entrées start, stop et wire. Lorsqu'on vient relier ces entrées avec la manette connectée au GND (pin 4 du header), nous aurons un état bas.

Il sera donc possible de détecter si nous avons touché le fil de cuivre ou si nous sommes au départ ou à l'arrivée du jeu.

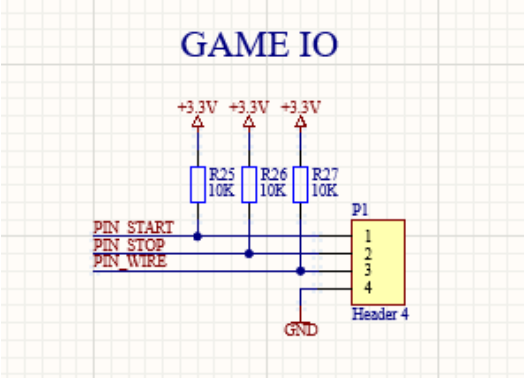


Figure 18: Game IO

Encodeur Rotatif PEC12

Afin de pouvoir naviguer dans le menu et valider des actions, l'utilisation d'un encodeur a été demandé. Je me suis inspiré de celui du KIT microcontrôleur de l'ES. Les signaux A et B servent à déterminer le sens dans la gestion du menu. Et le signal PB servira comme validation du choix dans le menu.

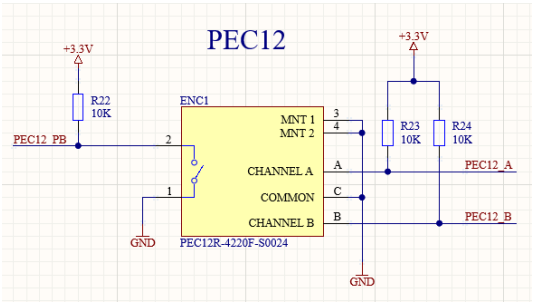


Figure 19 : PEC 12

LCD

Afin d'afficher le menu, l'utilisation d'un LCD a été demandé. Je me suis inspiré de celui du KIT microcontrôleur de l'ES. Pour le back light du LCD, j'ai dû redimensionner les composants afin de fournir 20 mA.

Backlight Supply Current	I_{LED}	V_{LED} = 3.0V	10	20	30	mA

Figure 20: Back light

$$R = \frac{U - Vled}{I} = \frac{3.3 - 3}{20m} = 15 => 15\Omega$$

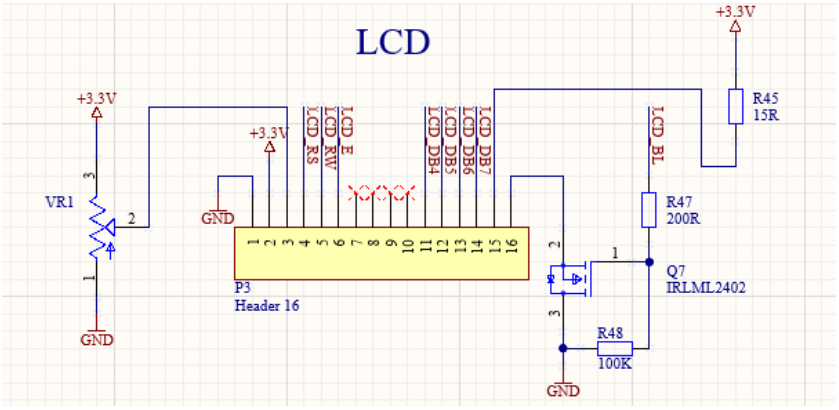


Figure 21: LCD

Buzzer

Afin d'avoir une indication sonore, j'utilise un buzzer avec une tension de fonctionnement de 2.5V à 4.5V et d'une résistance de 16Ω. Le courant est limité à 20 mA. Le modèle est le suivant SMTB0827TWR.

$$I = \frac{U}{R} = \frac{3{,}3}{150 + 16} = 19.8\,mA$$

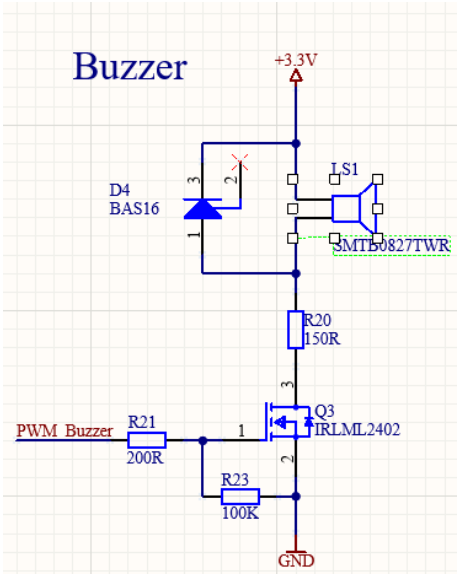


Figure 22:

LED RGB

Afin d'avoir une animation visuelle, des leds RGB sont utilisées. Il me faut des leds assez puissantes pour pouvoir diffuser de la lumière dans un plexiglass opaque. Pour une question de logistique, je vais prendre les mêmes leds qu'un camarade.

Une led consomme 250 mA, pour une luminosité à 75%. Il faudra certainement adapter la luminosité pour la diffusion de la lumière dans le plexiglass. Ces leds sont commandées via une sortie du µC qui fait commuter le MOSFET.

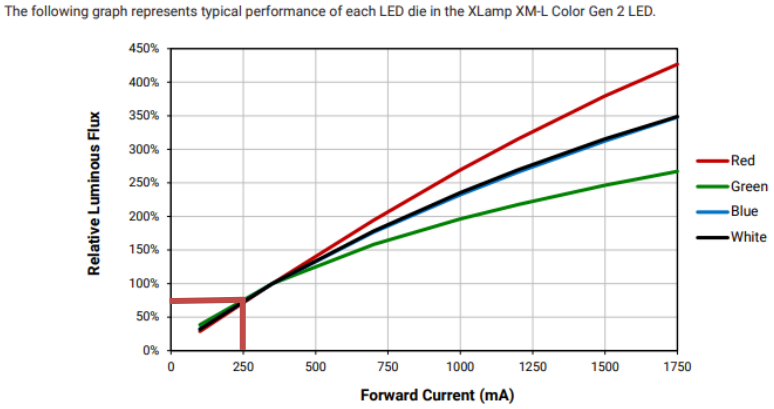


Figure 23: Courant – Luminosité

Forward voltage (@ 350 mA, 25 °C) - red	V		2.1	2.7
Forward voltage (@ 350 mA, 25 °C) - green	V		2.6	3.0
Forward voltage (@ 350 mA, 25 °C) - blue, white	V		2.9	3.2

Figure 24: Tension LED

Calculs pour 200 mA :

Pour la led bleue :

$$R = \frac{3.3 - UD}{I} = \frac{3.3 - 2.9}{200m} = 2 => 2\Omega$$

Pour la led verte :

$$R = \frac{3.3 - UD}{I} = \frac{3.3 - 2.6}{200m} = 3.5 => 3.3\Omega$$

Pour la led rouge :

$$R = \frac{3.3 - UD}{I} = \frac{3.3 - 2.1}{200m} = 6 => 6\Omega$$

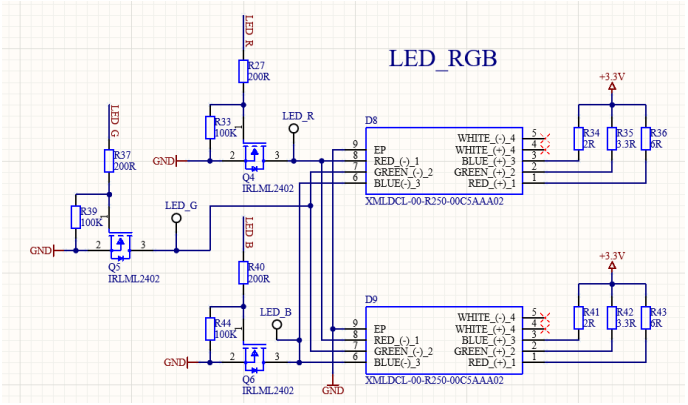


Figure 25: LED RGB

Flowchart

Firmware & Software

Application C#

Réaliser une interface de statistiques : meilleurs temps, nombre de fois toucher.
Réaliser différent mode de jeu : compte à rebours, nbr de touche max, ect.

Flowchart

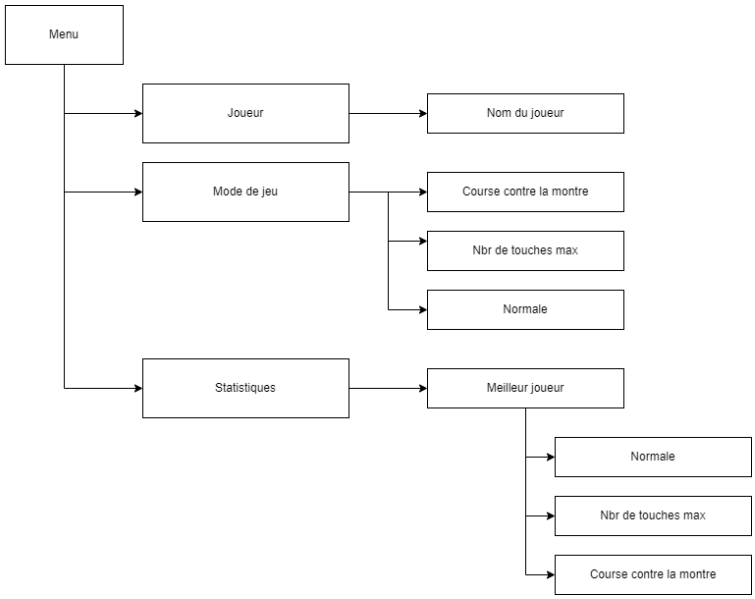
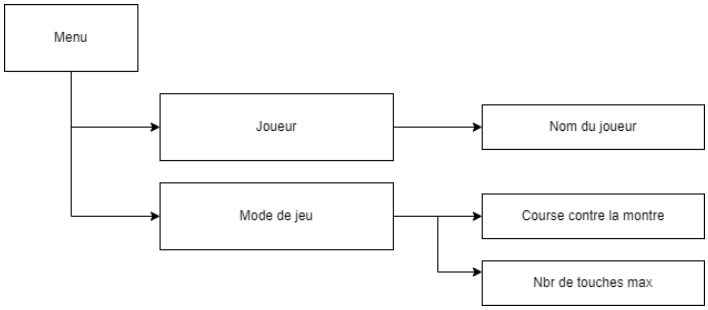


Figure 26

Firmware uC

Réaliser différent mode de jeu : compte à rebours, nbr de touche max, ect.
Gestion joueur.

Flowchart



Explications des interactions externes

Le joueur pourra interagir directement avec le système. Il y aura un bouton ON/OFF pour activer ou désactiver l'alimentation. A l'aide d'un affichage LCD et d'un encodeur, il pourra naviguer dans le menu. Mais, il pourra également interagir avec le système via un PC connecté via un port USB pour naviguer également dans le menu et en plus voir les statistiques des joueurs. Pour lancer le jeu, il suffira de décoller la boucle de la plaque de cuivre START et de même pour y mettre fin mais à l'arrivée (STOP).

Routage et explication

PCB

Le circuit a été prévu pour être fixé dans un boitier dimensionné sur mesure. Sa taille est donc de de 125x82mm.

Des zones de restrictions ont été placées autour des trous de fixation du PCB afin de n'avoir aucun cuivre, signal ou composant sur la zone.

Il a été routé pour correspondre à la classe de fabrication 6C selon les critères d'Eurocircuit.

Les pistes ont été routés avec une largeur de 0.254mm. Il y a un plan de masse (GND) sur chaque couche.

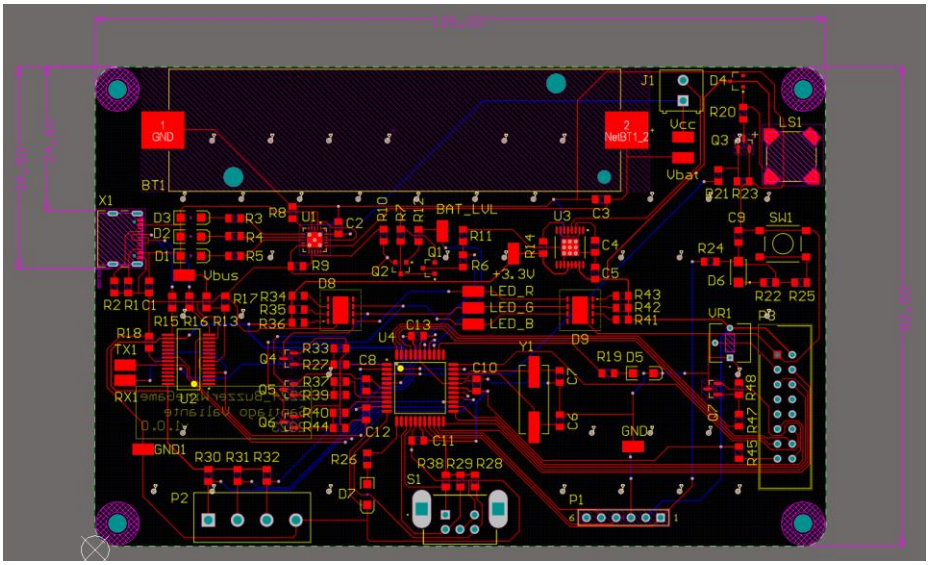


Figure 27: Circuit imprimé

Taille des pistes

Par défaut, Altium propose une taille de piste standard de 0,254 mm. Sachant que le courant max traversant dans les pistes est de 200 mA pour les leds, j'ai utilisé l'application Saturn PCB Design afin de m'assurer que la taille des pistes corresponde. Avec une taille de piste de 0.254 mm, la piste peut supporter jusqu'à 1.7 [A]. Je n'ai donc pas à me soucier des pistes.

Figure 28: Taille piste

PCB vue TOP

Sur cette vue du circuit, le socle de l'accumulateur est visible en haut du PCB. A sa droite, nous trouvons le connecteur J1 pour le switch ON/OFF et le buzzer (LS1).

Sur la gauche du PCB, nous retrouvons le connecteur USB-C ainsi que les différents éléments pour la conversion de la communication USB to UART.

Sur la partie centrale, nous avons le µC, les LEDs RGB, le régulateur de charge et le régulateur 3.3V.

En bas, nous retrouvons le connecteur pour la manette et les trois éléments qui compose le start, le stop et le fil de conduction du jeu. Egalement, nous retrouvons le PEC12 pour la gestion du menu et le connecteur P1 pour programmer.

Pour finir, sur la droite, nous avons le connecteur de l'écran LCD ainsi que le bouton reset.

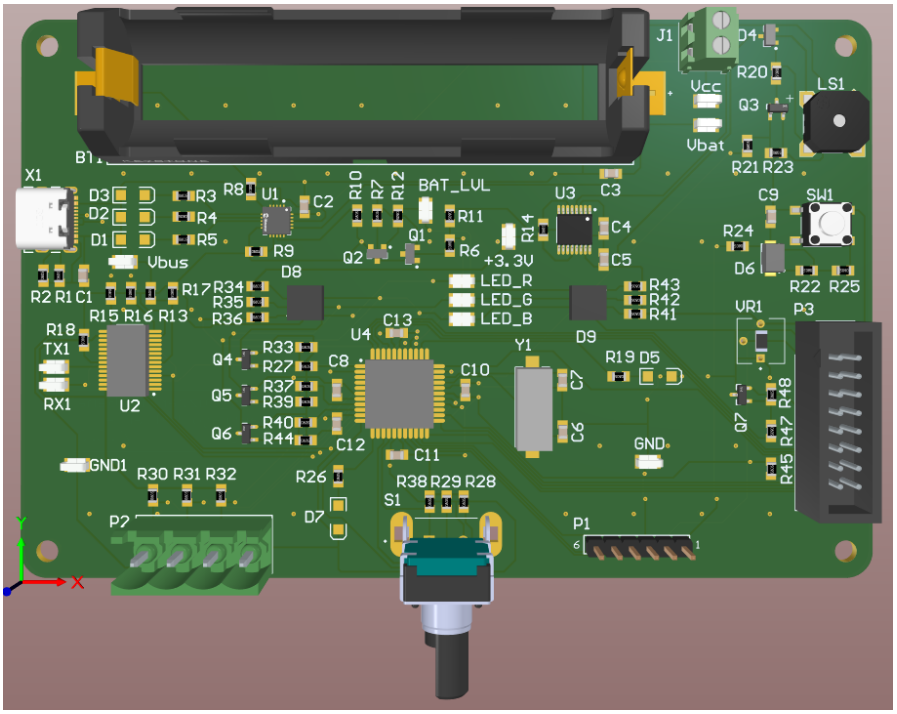


Figure 29: Vue top du PCB

PCB vue BOT

Sur le bottom, nous retrouvons seulement la notation du projet (Numéro, créateur, année et version).

Figure 30: Vue bot du PCB

Mise en service

Lors de la mise en service de mon projet, le régulateur 3.3V était fonctionnel. Cependant, mon µC ne se programmait pas. Des modification HW ont dû être effectuée.

Après avoir réalisé celles-ci, le µC était programmable. Ensuite, il me faut tester les différents périphériques (LCD, PEC12, Buzzer, Lecture des entrées digitale, LEDs et UART).

Modification HW

Les modifications hardware ont été effectuées. Le document a été mis à jour et a été placé dans **K:\ES\PROJETS\SLO\2224_BuzzerWireGame\hard\B**. La version du PCB qui a été produit est dans le dossier A. Le document des modifications effectuées se trouve en annexe.

1ère Modification

La 1ère erreur a été de prendre la mauvaise schématique du µC. J'ai pris celle du PIC32MX250F256D au lieu du PIC32MX130F256D. Ceci est dû à une faute d'inattention, car ces deux µC ont 44 pins. Heureusement seulement 3 pins ont dû être changée.

Avant Modification		Après Modification	
PGED3	Pin 19	PGED3	Pin 12
PGEC3	Pin 20	PGEC3	Pin 13
LED_Vie	Pin 12	LED_Vie	Pin 10

2ème Modification

La 2ème erreur a été de nommer la source de mon alimentation 3.3V de deux manières différentes. Heureusement, seul l'alimentation du bouton reset était erronée. J'ai donc dû relier le bouton à une source de 3.3V.

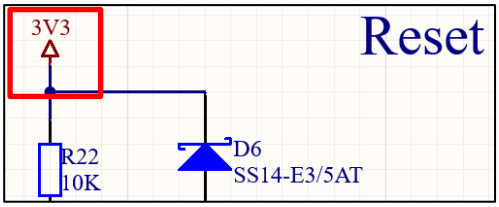


Figure 31: Avant Modif

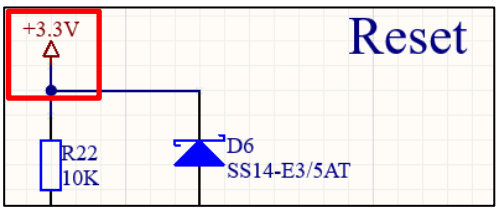


Figure 32: Après Modif

FIRMWARE

Concept

Le firmware est codé en langage C. Il a été réalisé à l'aide de MPLAB X IDE avec le framework Harmony v2.06 de Microchip. Compilé avec le compilateur XC32 v2.15 sans optimisation de compilation.

Le programme consiste à naviguer dans un menu afin de sélectionner un mode de jeu et de lancer une partie. Le programme devra également gérer un écran Lcd, un buzzer, des LEDs RGB, des I/O, un encodeur incrémental (PEC12) et une communication UART.

Synthèse des éléments réalisé lors de l'élaboration du programme

Afin de mener à bien la réalisation de ce programme, les éléments suivants ont dû être réalisés :

- Gestion du menu (MenuJeu.c).
- Gestion du buzzer (Buzzer.c).
- Gestion du jeu (Jeux.c).
- App.c

Les éléments qui ont été repris mais ont eu besoins d'être ajustés pour ce projet :

- Gestion du LCD (Mc32DriverLcd.c).

Les éléments repris tels quels :

- Gestion de la communication UART (GesFifoTh32.c, Mc32gest_RS232.c).
- Gestion du PEC12 (Mc32Debounce.c, GesPec12.c).
- Délais (Mc32Delays.c).

Configuration Harmony

Configuration du clock

Configuration du clock à 40 MHz.

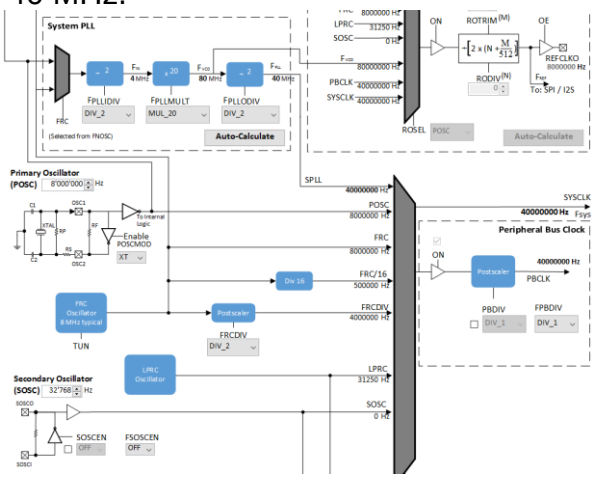


Figure 33 : Configuration à 40MHz

Timer 1

L'interruption du timer 1 cycle toutes les ms et active l'application toutes les 10 ms.

Ce calcul est utilisé a été utilisé pour toute les configurations :

$$nbTicks = \frac{fsys}{fTimer * PRESC} - 1$$

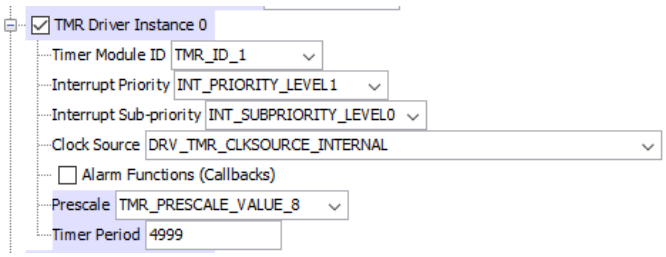


Figure 34: Timer 1

Timer 2

L'interruption du timer 2 sers à toggle l'état de la sortie du buzzer. Timer Period sera changer dans la librairie Buzzer.c afin de choisir la fréquence de la note voulue.

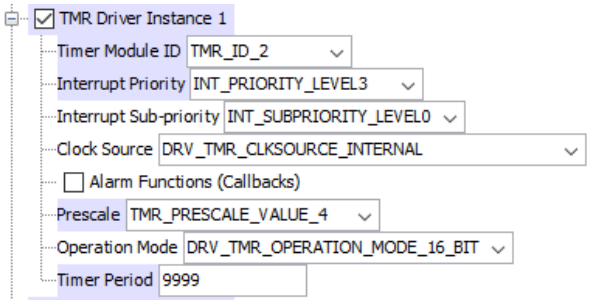


Figure 35: Timer 2

Timer 3

L'interruption du timer 3 sers à compter toutes les secondes pour la gestion du temps du jeu.

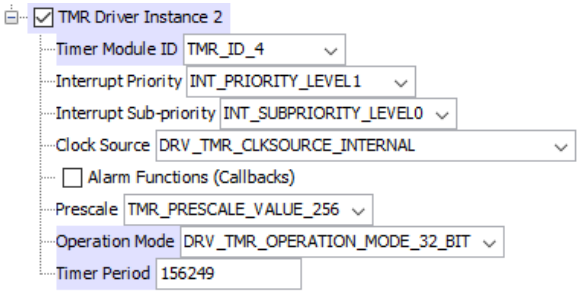
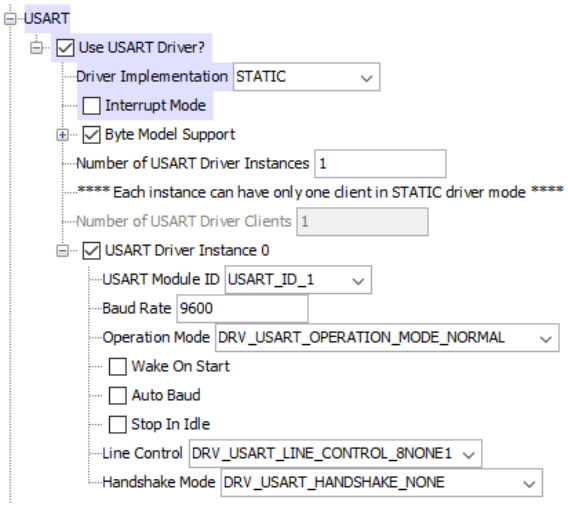


Figure 36: Timer3

UART

Communication UART afin de communiquer avec le PC. Je n'utilise pas d'interruption car j'envoie moins de 8 caractères.



Programme

App

Au début de l'app passe dans l'état d'initialisation, attends 3sec et poursuit son chemin dans le SERVICE TASK. On regarde si nous recevons une communication via la communication UART puis la gestion du menu.

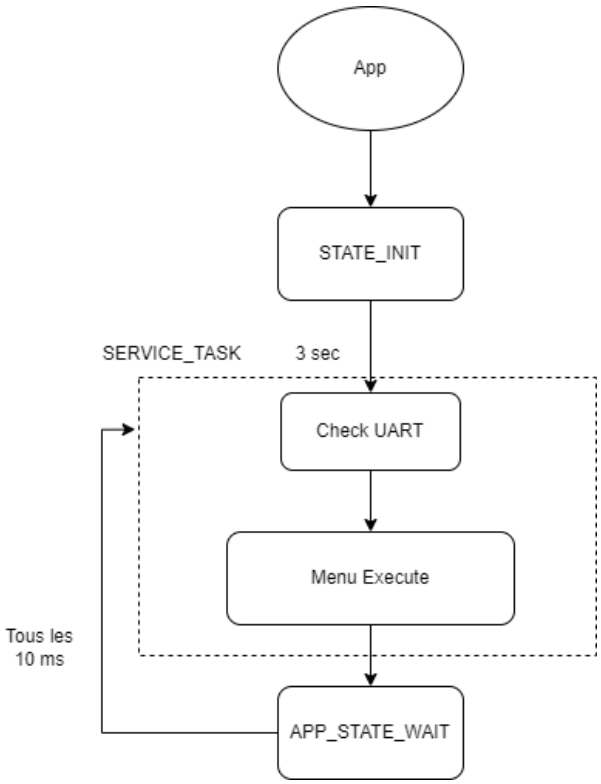


Figure 37: App.c

Menu

Dans le menu, on se sert du pec12 pour naviguer. Selon sa position, nous pouvons choisir le mode de jeu ou si nous voulons lancer une partie. L'affichage se met à jour en même temps.

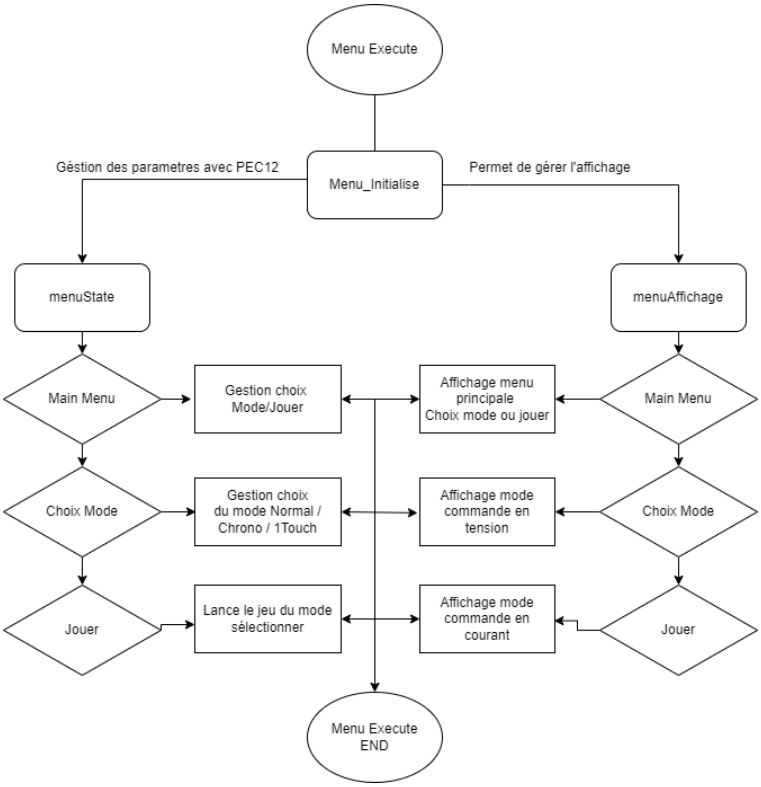


Figure 38: menuJeu.c

Jeu – Mode Normal

Dans le mode de jeu normal, nous regardons si nous sommes déjà rentrés dans la fonction. Si ce n'est pas le cas on remet tous à zéro. Si oui nous attendons que le joueur se remette sur le point de départ. Ensuite un compte à rembours s'active avec le lancement du timer. Une fois le joueur arrivé, le timer s'arrête et la fonction retourne la valeur de 1 pour détecter la fin du jeu. En parallèle, nous mettons à jour l'affichage du temps et du nombre de toucher que le joueur a fait.

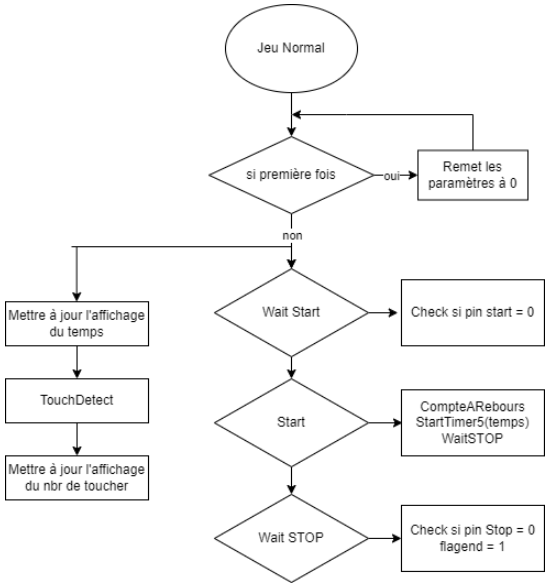


Figure 39: Jeu.c

Test et mesure

Communication UART

Pour tester la communication UART, j'envoie une trame de commande tel que **!M=1#** via Putty. En retour, le µC renvoie « **Reception Ok** ».

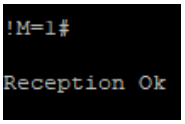


Figure 40: Visualisation des messages sur Putty

Protocole de mesure

Mesures prise sur les points de tests RX1 et TX1 avec le CH1 de l'oscilloscope.

Mesure Rx

Nous pouvons constater que nous recevons bien le message **!M=1#**.

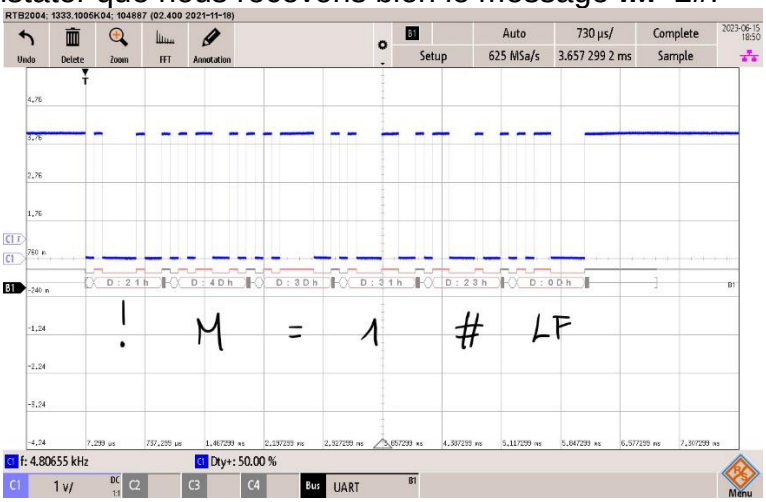


Figure 41: mesure sur Rx

Mesure Tx

Nous pouvons constater que nous recevons bien le message **!M=1#**.

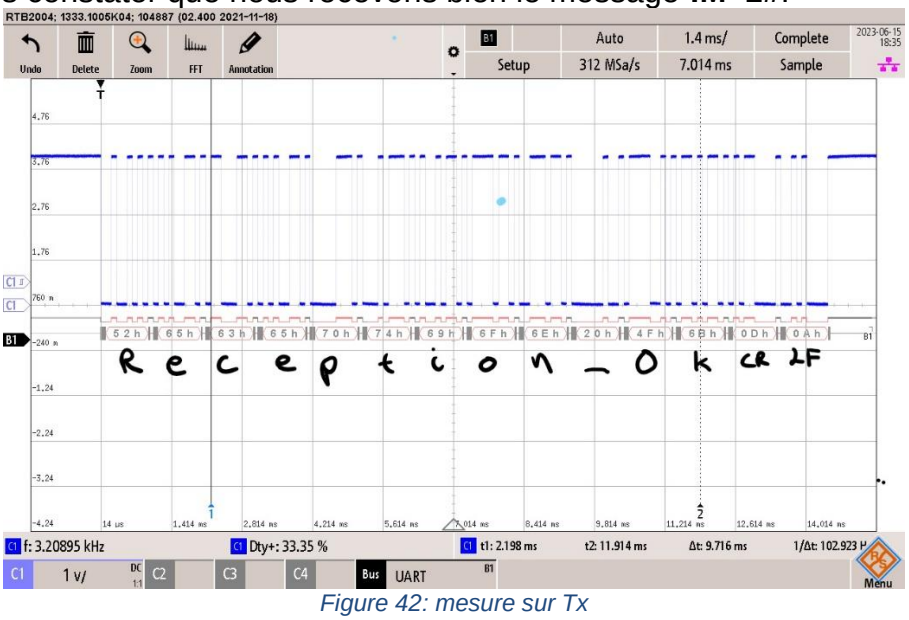


Figure 42: mesure sur Tx

Pec12

Vérification des signaux A, B et PB du PEC12.

Protocole de mesure

Mesures prise sur la pine A et la pin B du Pec12 avec le CH1 et le CH2. Ainsi que la pin PB avec le CH1.

Mesure des signaux A et B

Nous pouvons constater que les signaux A et B sont fonctionnels pour la gestion du menu.

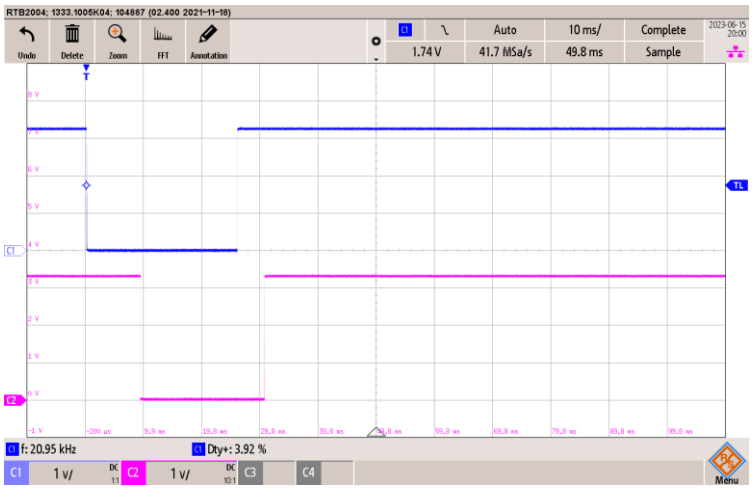


Figure 43: Signaux A et B Incrémentation

Mesure du signal PB

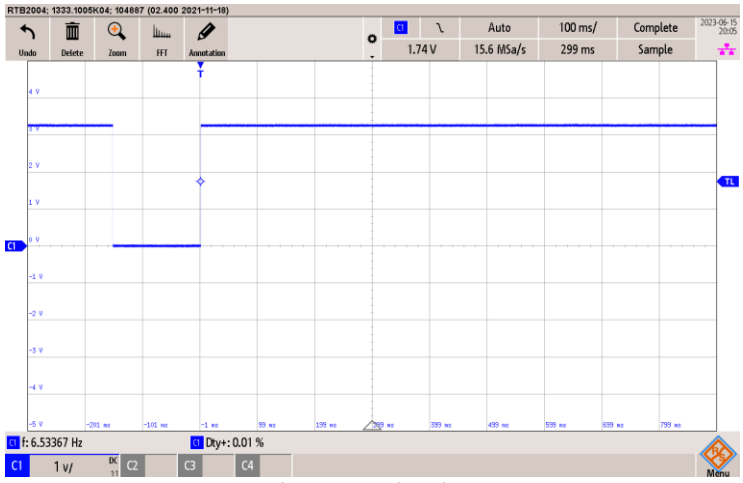


Figure 44: Signal PB

Visuel

Visuellement, nous pouvons valider le LCD. A l'état d'initialisation, le nom du projet ainsi que le prénom du créateur s'affichent.

Figure 45:Image Lcd

Les couleurs bleues et vertes des LEDs RGB fonctionnent, cependant, la couleur rouge ne fonctionne pas. Je n'ai pas eu le temps de corriger l'erreur.

Figure 46: Test led B

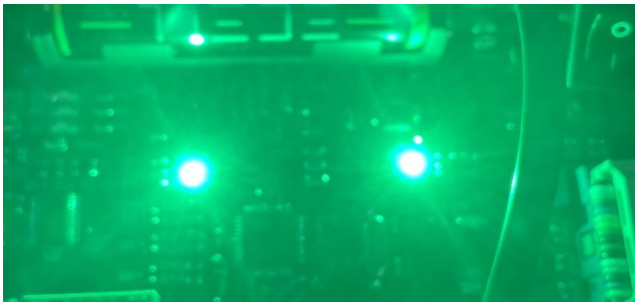


Figure 47: Test led G

Sonore

Grâce au son produit du buzzer, je peux déduire qu'il fonctionne. Cette fonction pourra être démontrée lors de la présentation.

Résultat de fonctionnement

Par rapport au cahier de charge :

- La partie HW est fonctionnelle (manque led rouge).
- Les périphériques sont fonctionnels.
- La communication UART est fonctionnelle mais n'a pas été implémentée, car il manque une pine de détection USB pour passer en mode remote. Il faudra l'implémenter pour une version 2.
- Le software n'a pas été réalisé.
- Le firmware n'a pas été finalisé.

Amélioration et tâches à effectuer

Partie HW :

- Ajouter la pine de détection USB.

Partie FW :

- Finaliser le jeu. (Les différents modes de jeu + amélioration menu).
- Implémenter la détection USB et finaliser
- Gestion Pseudo Joueur.

Partie SW :

- Faire l'application.

Conclusion

Ce projet a été réalisé en trois étape distinctes. La première étape était la pré-étude, mon erreur a été de ne pas avoir déterminé le courant consommé par circuit afin de déterminer la capacité de mon accumulateur.

La phase design c'est plutôt bien passé. Cependant, j'ai fait quelques erreurs sur mon schéma qui m'ont fait perdre pas mal de temps pour la phase de réalisation. Je n'ai donc pas pu eu le temps de dessiner le boîtier.

Le point positif est que tous mes périphériques sont fonctionnelles. Cependant, le firmware n'as pas pu être finalisé pour que le code soit fonctionnel afin de jouer. Je n'ai également pas pu commencer la partie software.

Il faudrait ajouter une pine pour la détection de la connexion USB pour effectuer le changement en mode local et remote.

Dans l'ensemble, le projet s'est bien déroulé, mais je n'ai pas pu répondre entièrement au cahier de charge.

Le projet n'étant pas fini, le mode d'emploi du système n'a pas été fait.

Lausanne ETML-ES, 15.06.23. Santiago Valiante

Annexes

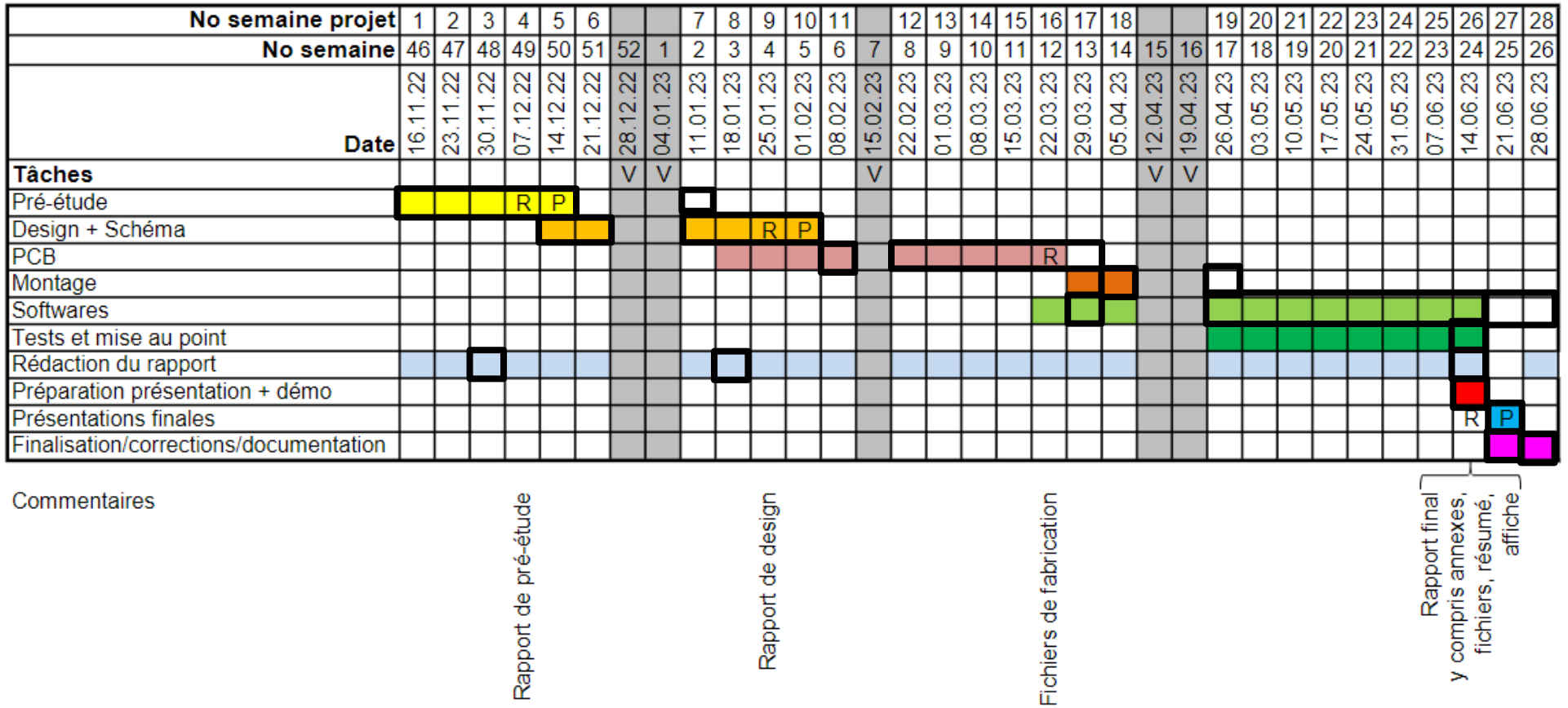
CDC

Dossier du CAO

Parties essentielles du code

Planning initial et tâches effectives

Tâche effectives en encadré en noir.



V : Vacances
R : Remise de documentation/dossier/fichiers. Peut être remis plus tôt.
P : Présentation.
Remise d'un rapport chaque semaine précédant une présentation.

Journal de travail

Journal de travail - PROJET		
Date	**Tâches**	**Problème rencontré**
16.11.2022	Choix du sujet	
17.11.2022 + 1/2j 18.11.2022	Rédiger cahier de charges	
23.11.2022 + 1/2j 24.11.2022	Rédiger cahier de charges, début pré-étude schéma bloc	
30.11.2022	Pré-étude : détaillé les blocs + Validation du CDC	
07.12.2022	Pré-étude : choix composants, estimation des prix, conclusion	
14.12.2022	Présentation pré-étude, début schématique	Déterminer l'accumulateur le plus vite possible
21.12.2022	Corrections pré-étude + Schéma bloc précis des différents blocs	
11.01.2023	Déterminer l'accumulateur, choix des composants	
18.01.2023	Dimensionnement des fifférents blocs + schéma	
25.01.2023	Finalisation du schéma, rédaction du rapport	
01.02.2023	Finalisation du schéma + Revue schéma + Préparation présentation projet	
08.02.2023	Présentation schéma projet, modifications	
15.02.2023	RELACHES	
22.02.2023	Correction du schéma + Recherche des footprints aux composants	
01.03.2023	Recherche des footprints aux composants + placement composant	
08.03.2023	Mise à jour BOM - vérification footprints, composants en stock etc	
15.03.2023	Finir le rootage	
22.03.2023	Finalisation du design du PCB et contrôle avec EuroCircuits	
29.03.2023	Commande du PCB et commande des composants + Commencer la programmation	
05.04.2023	Montage du PCB	
12.04.2023	Vacances	
19.04.2023	Vacances	
26.04.2023	Finalisation du montage plus test de la carte	
03.05.2023	Validation régualteur 3v3,	Problème microcontrôleur
10.05.2023	Checher problème + modification + validation microcontrôleur +Buzzer et Led RGB fonctionnells	problème venu de l'HW
17.05.2023	LCD + Gestion du menu + Jeu	LED Rouge non fonctionnel
24.05.2023	Gestion de menu + Jeu	
31.05.2023	Validation des périphériques + Gestion de menu + Jeu	Oubli de la communication UART
07.06.2023	Résoudre problème du menu + Communication UART	
14.06.2023	Rapport + Communication UART + Tests	
20.06.2023	Présentation	
28.06.2023		

Résumé du projet

Affiche du projet

Modifications HW