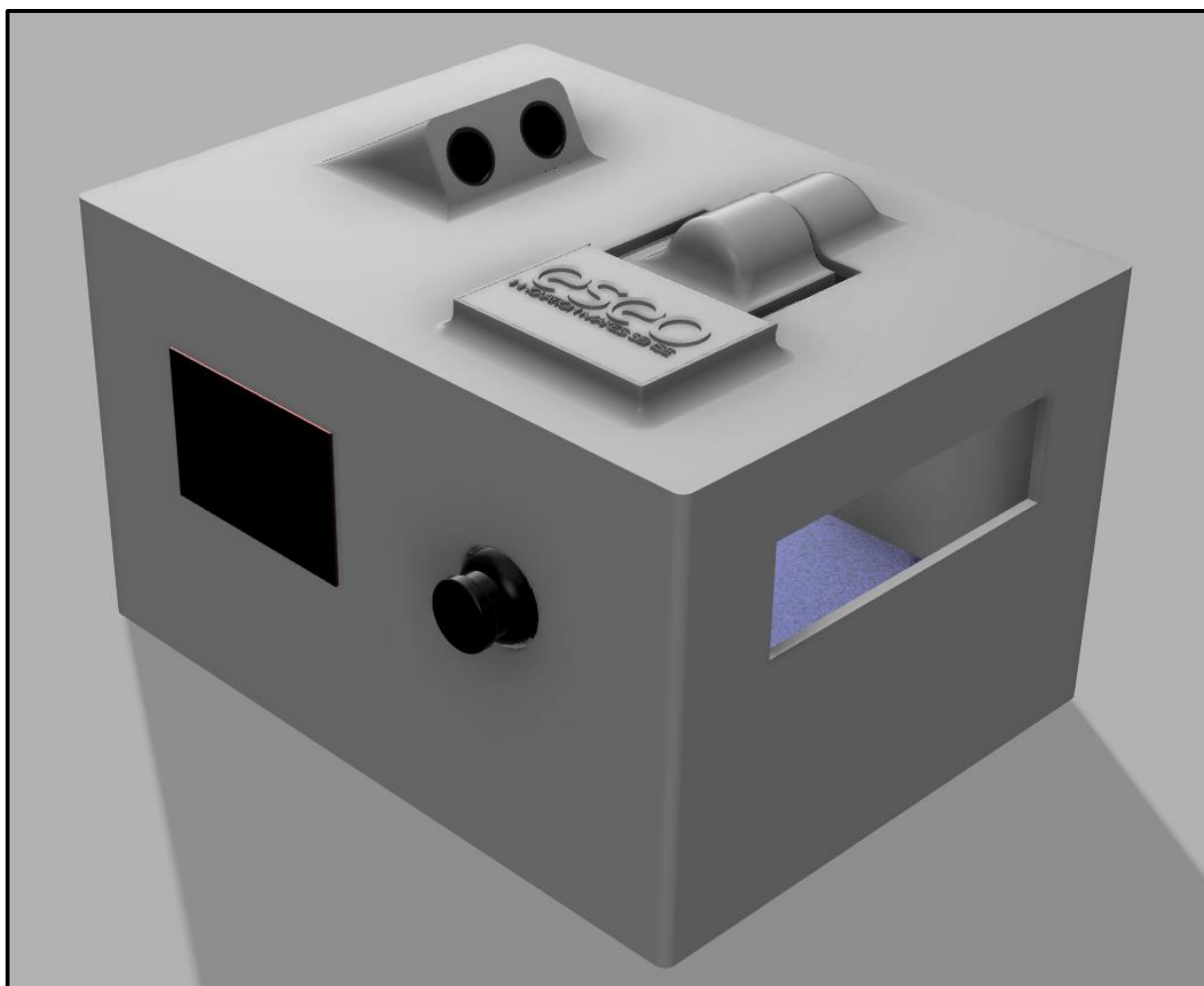


Rapport Final Projet DEEP



MISSION UNSLEEPABLE

Introduction du Projet :

L'objectif de notre projet DEEP est de réaliser un réveil qui réveillera à coup sûr l'utilisateur. Pour se faire, nous allons utiliser les connaissances apprises lors des différentes missions réalisées auparavant, aussi bien Hardware que Software.

Ce rapport contient l'avancement qui a été fait sur la totalité des séances, le cahier des charges, les composants utilisés, les ports associés, et les compléments.

Sommaire :

1. Compléments choisis
2. Cahier des charges
3. Manuel d'utilisation
4. Composants utilisés
5. Tableau d'affectation des ports
6. Description d'un algorithme du programme
7. Structure du programme
8. Tests
9. Carnet de bord
10. Etat d'avancement et analyse du projet
11. Conclusion
12. Design CAO
13. Design PCB
14. Doxygen

1. Compléments choisis :

Complément (voir l'annexe correspondante à la fin du rapport)	pts	
A- utilisation d'un analyseur logique pour déchiffrer des trames	2	<input type="checkbox"/>
B- design de PCB		
	avec bluepill	3 <input checked="" type="checkbox"/>
Composants CMS et microcontrôleur nu	4	<input type="checkbox"/>
C- design CAO d'un boîtier	2	<input checked="" type="checkbox"/>
D- documentation doxygen du code source	1	<input checked="" type="checkbox"/>
E- mesure de conso et d'énergie selon scénarios	2	<input type="checkbox"/>
F- enregistrement de paramètres en flash	1	<input type="checkbox"/>
G- gestion de version du code source / Git ou SVN	1	<input type="checkbox"/>
T- jeu de tests pour valider une fonctionnalité software ou hardware	1	<input type="checkbox"/>
Nombre de points ciblés au maximum :		6 / 6

2. Cahier des charges :

Nous avons choisi de réaliser un réveil avec affichage LCD. Il y aura un menu parcourable grâce à un joystick dans lequel on pourra régler l'heure, l'alarme, et les différents modes.

L'utilisateur aura la possibilité de choisir le son de l'alarme. Pour éteindre l'alarme il y aura un bouton placé sur une plateforme à bascule qui pivotera grâce à un servomoteur. Sur la face supérieure sera placé un capteur à ultrason, qui détectera la présence de la main de l'utilisateur. A partir d'une certaine distance entre le capteur et la main, le servomoteur fera basculer la plateforme de 180° rendant l'appui sur le bouton impossible. Lorsque la main sera reculée, la plateforme reprendra sa position initiale. Pour vraiment arrêter l'alarme, l'utilisateur devra garder sa main au-dessus du capteur et appuyer sur le bouton avec sa 2e main en dessous.

Les contraintes sont principalement physiques : Il faut réaliser une structure de boîtier à la fois solide et légère, de façon à avoir un réveil qui ne casse pas s'il tombe de la table de nuit (ou si il est jeté par l'utilisateur fou de rage), mais qui soit facile à déplacer. De plus le boîtier doit posséder des logements permettant d'encastrement les composants et de les fixer pour qu'ils ne se déplacent pas à l'intérieur, tout en laissant la possibilité de les extraire s'ils doivent être changé. Pour finir, la conception mécanique doit prendre en compte la gestion des câbles lors de la rotation de la plateforme, sans gêner le mouvement et qu'il soit réalisable un grand nombre de fois.






3. Manuel d'utilisation :

Le produit est comparable à un réveil classique. Voici les différentes fonctionnalités : Affichage du jour, de la date, de la sonnerie, et de l'heure de l'alarme. Ces deux dernières sont aussi personnalisables grâce au menu d'options.

La valeur ajoutée de ce produit est de vous réveiller à coup sûr car pour éteindre l'alarme à votre réveil le matin il faudra appuyer sur un bouton fixé sur une trappe, trappe qui pivotera quand votre main approchera. Pour désactiver l'alarme, il faudra garder votre main au-dessus du réveil et appuyer sur le bouton en dessous avec votre 2^e main.

4. Composants utilisés :

Nom/Référence du Composant	Utilisation/Fonctionnement
Ecran LCD 3 pouces ILI9341 	Ecran utilisé pour naviguer dans le menu principal. Option tactile non utilisée car le pilotage se fera avec le joystick. Option lecture de carte SD utilisée pour lire les vidéos. Résolution : 320x240. Alimentation en 3.3V.
Servo Moteur MG996R 	Moteur utilisé pour entrainer la plateforme dans une rotation de 180°. Le choix du duty du signal PWM permet de choisir sa rotation : 0.1 → 90° 0.2 → 180° Alimentation en 5V.
Capteur Ultrason HCSR04 	Capteur utilisé pour détecter la présence de la main à partir d'une certaine distance réglable avec le soft. Alimentation en 5V.
Bouton poussoir 	Bouton placé sur la trappe permettant d'éteindre l'alarme quand elle sonne.
Joystick 	Joystick placé à côté de l'écran permettant de naviguer dans le menu de paramétrages. (Remplacé par des boutons dans la version finale, voir partie 10)

<p>Hautparleur <i>TMQ60500</i></p> 	<p>Hautparleur permettant d'avoir une alarme musicale personnalisée.</p>
<p>Amplificateur Audio <i>PAM8403</i></p> 	<p>Amplificateur permettant de régler le volume sonore de l'hautparleur.</p>
<p>Lecteur mp3 <i>HW311</i></p> 	<p>Lecteur mp3 utilisé pour la lecture des musiques téléchargées sur la carte SD.</p>
<p>BluePill</p> 	<p>Placée sur la DEEP Purple Complete Board. Permet la gestion des périphériques.</p>
<p>Nucleo STM32</p> 	<p>Permet à la BluePill d'utiliser la partie débogueur de la Nucleo STM32</p>

5. Tableau d'affectation des ports :

Composants	Pins (Côté composant)	Pins (Côté Bluepill)
Ecran LCD	VCC	3,3V
	GND	GND
	CS	PB11
	RESET	PB10
	D/C	PB1
	MOSI / T_DIN	PA7
	SCK / T_CLK	PA5
	LED	3,3V
	MISO / T_DOUT	PA6
	T_IRQ	PA4
	T_CS	PA0
Servo Moteur	VCC	5V
	GND	GND
	PWM	PA8
Capteur Ultrason	VCC	5V
	ECHO	PB5
	TRIGGER	PB4
	GND	GND
Lecteur MP3	VCC	5V
	TX	PB7
	RX	PB6
	GND	GND
Amplificateur	VCC	5V
	GND	GND
Bouton Poussoir	VCC	3,3V
	GND	GND
	DATA Bouton Gauche	PA1
	DATA Bouton Centre	PA2
	DATA Bouton Droite	PA3

6. Description d'un algorithme

Pour la partie Software, nous avons commencé à programmer l'écran LCD. Pour ce faire, nous avons importé la librairie Ili9341. Dans le fichier config.h, nous avons défini l'utilisation de l'écran ainsi que la taille d'écriture. Après initialisation et rotation à l'horizontale (fonction *ILI9341_Rotate*), il nous reste juste à afficher des caractères ASCII au position x et y voulues avec la fonction *ILI9341_Puts*.

```
ILI9341_Puts(100,100, timeString, &Font_11x18, ILI9341_COLOR_BROWN,ILI9341_COLOR_WHITE);  
ILI9341_Puts(100,200, dateString, &Font_11x18, ILI9341_COLOR_BROWN,ILI9341_COLOR_WHITE);
```

Puis, pour réaliser notre mission principale, l'affichage de l'heure et la date, nous avons besoin d'utiliser le capteur RTC. Pour cela, nous avons besoin de créer 2 structures de type *RTC_TimeTypeDef* et *RTC_DateTypeDef* de la librairie RTC. Pour chaque structure, nous définissons des attributs entiers non signés 8 bits (Hours, Minute, Seconds pour le temps et Date, Month, Year pour la date).

```
RTC_TimeTypeDef currentTime;      RTC_DateTypeDef currentDate;  
  
currentTime.Hours = 10;  
currentTime.Minutes = 40;  
currentTime.Seconds = 0;  
  
currentDate.Year = 22;  
currentDate.Month = 11;  
currentDate.Date = 24;
```

Grâce à l'horloge interne, le module est capable d'incrémenter le temps. Nous avons à utiliser les fonctions *RTC_get_time(¤tTime)* et *RTC_get_date(¤tDate)* pour récupérer nos informations avec les pointeurs. Cependant, nous devons transformer les attributs en chaîne de caractères pour pouvoir les afficher sur l'écran. Pour cela, on initialise 2 variables, tableau de char, *timeString* et *dateString*. Il nous reste à utiliser la méthode *sprintf* comme ci-dessous.

```
sprintf(timeString, "%d:%d", currentTime.Hours, currentTime.Minutes);  
sprintf(dateString, "%d/%d/%d", currentDate.Date, currentDate.Month, currentDate.Year + 2000);
```

Enfin, pour utiliser le réveil, nous avons besoin de jouer une musique. Pour cela, il nous faut configurer le lecteur MP3. Nous avons à préalable enregistrer une chanson dans la carte SD du lecteur. Le module MP3 communique en série avec la carte Bluepill. Pour transmettre des commandes, nous allons donc utiliser la communication UART. Pour cela, on initialise une liaison de 9600 Baud sur les pins RX et TX (PB6, PB7).

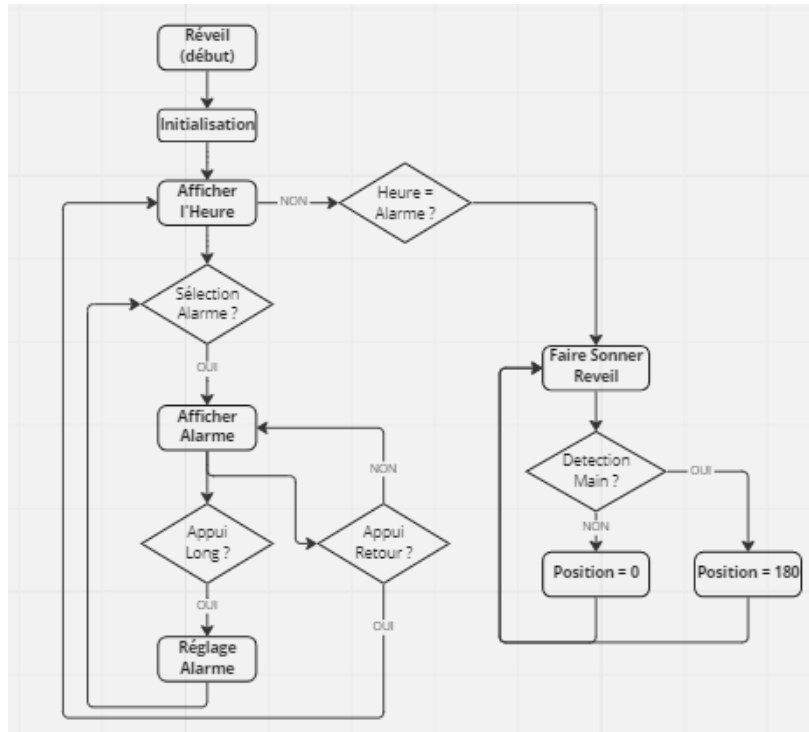
Les data à transmettre sont des listes de nombres hexadécimaux. Au minimum, nous avons 2 ordres à transmettre pour jouer une musique (l'accès à la carte SD et la lecture de la musique 1). On retrouve alors 2 tableaux de caractères hexadécimaux.

```
char storage[8] = {0x7E, 0xFF, 0x06, 0x09, 0x00, 0x00, 0x02, 0xEF};  
char playCommand[8] = {0x7E, 0xFF, 0x06, 0x0F, 0x00, 0x01, 0x01, 0xEF};
```

Il nous reste à envoyer les ordres sur le lecteur avec la fonction `UART_puts`.

```
UART_puts(UART1_ID, storage, 8);
```

D'un point de vue Software, le système complet sera représenté par une machine à états. Nous avons donc créé les différents états (voir schéma ci-dessous) dans une structure ENUM et une variable booléenne *entrance* pour l'entrée dans un état.



7. Structure du programme

Liste des fichiers :

FICHIER STATE.C

Nom de la fonction	Paramètres	Retour	Description
State_machine			Gère la machine à état du système

FICHIER VARIABLE.C

Nom de la fonction	Paramètres	Retour	Description
getAlarm	Entier 8bits	Alarme	Obtenir les données de l'alarme
setAlarmHours			Modifier la variable Hours
getListMusics		Tableau de Musique	Obtenir les données de ListMusics

FICHIER BUTTON.C

Nom de la fonction	Paramètres	Retour	Description
button_left_press_event		Booléen ret	Gestion de l'appui simple du bouton
button_center_press_event		Booléen ret	Gestion de l'appui simple du bouton
button_right_press_event		Booléen ret	Gestion de l'appui simple du bouton

Nom de la fonction	Paramètres	Retour	Description
screen_hours	Alarme		Affichage sur l'écran des éléments uniques
setHourString	Alarme		Affichage de l'heure
changeAlarmHourVariable			Gérer la modification de l'heure

Les fichiers SetMinute, SetMusic et AlarmeRing suivent le même principe.

FICHER RTC.C

Nom de la fonction	Paramètres	Retour	Description
initModuleRTC	Year, Month, Date, Hour, Minute, Second		Initialiser le module RTC

FICHER HCSR04.C

Nom de la fonction	Paramètres	Retour	Description
hcsr04_state_machine			Machine à état du capteur à ultrason

FICHER MP3PLAYER.C

Nom de la fonction	Paramètres	Retour	Description
initPlayer			Initialiser le lecteur mp3
playMusic	Entier		Jouer une musique
stopMusic			Stopper la musique

Nom de la fonction	Paramètres	Retour	Description
SERVO_init			Initialiser le servomoteur
SERVO_set_position	Entier		Fixer le servomoteur à une position précise
getCurrentPosition		Entier	Récupérer la position courante

8. Tests :

Nom du Test	Description	Résultat
Test de l'alimentation	Mesure au voltmètre des entrées. On attend 5V.	Validé
Test d'envoi UART	Mesure à l'oscilloscope. On doit observer un changement de tension lorsqu'une commande est envoyée.	Validé
Test Lecteur mp3	Mesure à l'oscilloscope. On doit observer un changement de tension lorsqu'une musique est envoyée.	Validé
Test Servomoteur	Mesure à l'oscilloscope. On doit observer le signal PWM avec le bon Duty.	Validé
Test Ecran	Vérification visuelle de l'écriture sur l'écran	Validé
Test Bouton poussoir	Mesure à l'oscilloscope. On doit observer un changement de tension lorsque l'on appuie sur le bouton.	Validé

9. Carnet de Bord :

Date	Tâches, réalisateurs, difficultés rencontrées.	A faire la prochaine fois
08/11/22	Victor Benoit & Maxence Bigot : Réflexion sur le projet	Choix des composants
10/11/22	Victor Benoit & Maxence Bigot : Choix des composants	Définir les broches utilisés
15/11/22	Victor Benoit & Maxence Bigot : Définition des broches utilisés, Réalisation de la fonction Alimentation	Soudure des composants de la carte
17/11/22	Victor Benoit & Maxence Bigot : Soudure des composants	Développement soft, Soudure des composants restants, Conception Boitier
22/11/22	Victor Benoit : Soudure des composants restants, Conception du boitier Maxence Bigot : Développement soft - Ecran LCD - capteur RTC	Développement Soft Lecteur de carte SD, Conception Boitier
24/11/22	Victor Benoit : Conception Boitier Maxence Bigot : Développement soft - Ecran LCD - Lecteur carte SD	Développement Soft du Servomoteur, Soudure et Câblage
29/11/22	Victor Benoit : Développement soft - Servomoteur - HCSR04 Maxence Bigot : Soudure + Câblage Hautparleur, Lecteur mp3	Rédaction du rapport intermédiaire, Développement Soft UART
01/12/22	Victor Benoit : Rédaction du rapport intermédiaire Maxence Bigot : Développement Soft UART & Hautparleur	Développement Soft Hautparleur & Conception maquette démo
06/12/22	Victor Benoit : Découverte d'Altium et routage de la PCB Maxence Bigot : Développement Soft UART & Hautparleur	Continuer la conception de la PCB, Créer les variables Globales Musiques et Alarme
13/12/22	Victor Benoit : Tracé des nets de la PCB Maxence Bigot : Création des variables globales Musiques et Alarme	Continuer la conception de la PCB, Développement des différents affichages de l'écran

15/12/22	Victor Benoit : Tracé des nets, DRC, Estimation du coût de production Maxence Bigot : Développement des différents affichages de l'écran	Rédiger le rapport du complément PCB, Gestion des boutons poussoirs
03/01/23	Victor Benoit : Rédaction du rapport de complément PCB, Rédaction du rapport final Maxence Bigot : Gestion des boutons poussoirs	Continuer la rédaction du rapport final, Vérification générale du projet
19/01/23	Victor Benoit : Rédaction du rapport final Maxence Bigot : Vérification générale du projet	

10. Etat d'avancement et analyse du projet réalisé :

La partie Hardware est terminée, tous les composants utilisés ont été soudés sur la DEEP Purple Complete Board. Nous avons aussi préparé beaucoup de broches femelles nous permettant de placer et enlever les composants comme bon nous semble afin de mieux manipuler la plaque PCB. La partie CAO est terminée, le boîtier est fonctionnel et prêt pour une potentielle impression. Le design de la PCB est terminé mais une version avec les composants qui assurent la fonction du son pourrait être envisagée.

La partie Software est pratiquement terminée. Les principaux objectifs ont été réalisés comme l'affichage de l'heure et de la date, l'affichage de l'alarme et son réglage. Nous pouvons également ajouter des musiques dans la carte SD et manuellement dans le code pour qu'elle puisse être utilisée comme sonnerie. Le servomoteur est fonctionnel. Le seul problème est le capteur à ultrason HCSR04 qui ne fonctionne pas.

Avec plus de temps et une meilleure approche dans le projet, nous pourrions ajouter d'autres fonctionnalités comme l'annulation de l'alarme ou le réglage du capteur à ultrasons. Nous avons décidé d'utiliser des boutons à la place du joystick car c'est plus simple à utiliser.

Malgré tout, il nous reste des composants à paramétrer. Nous avons perdu du temps à cause des erreurs mémoires qui provoquent l'erreur « asm volatile », les boutons poussoirs qui étaient mal soudés au début. Avant d'utiliser les getters et setters pour l'alarme, on n'arrivait pas à modifier ses attributs.

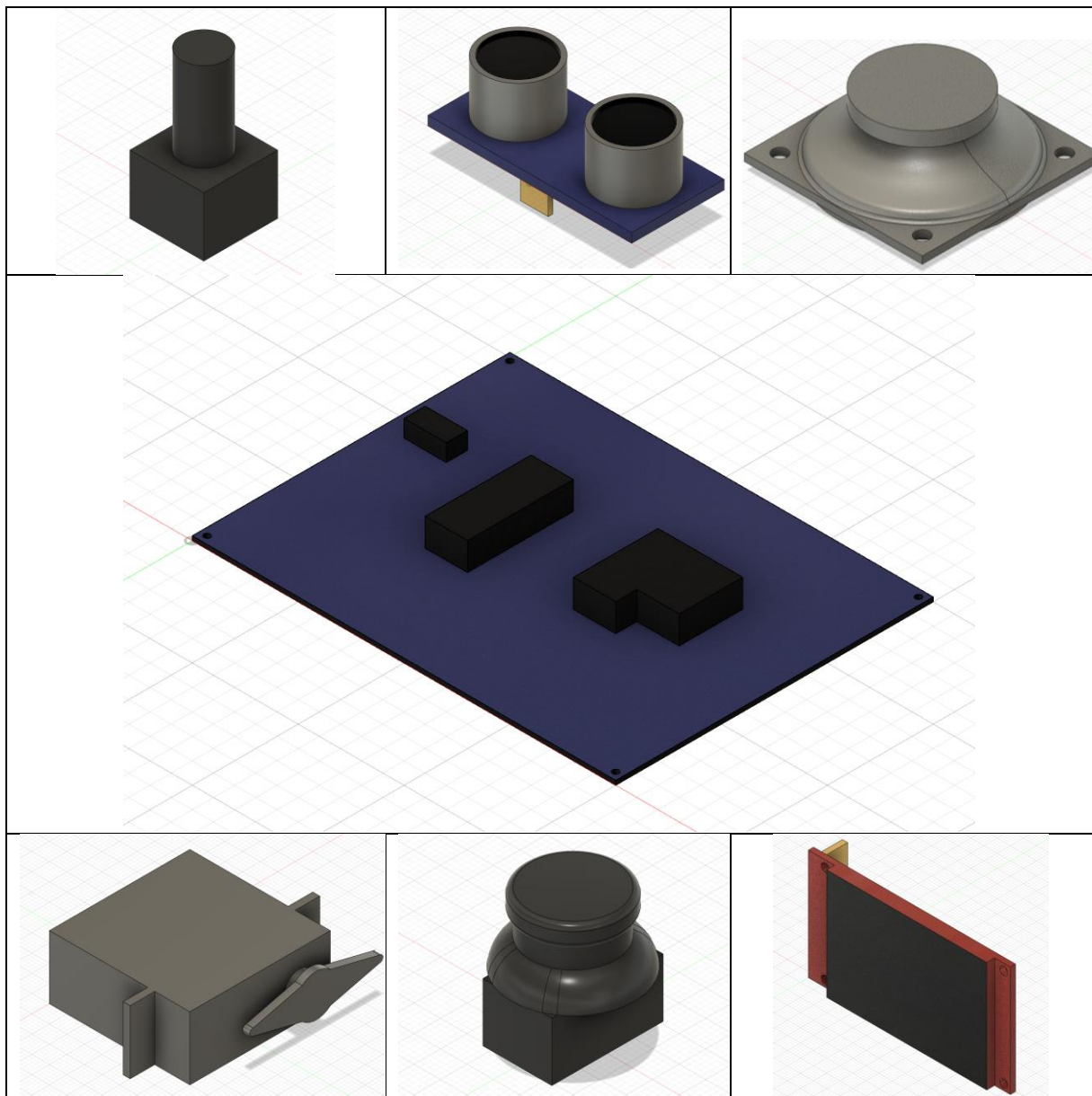
Le document du lecteur MP3 indiquait que l'on pouvait mettre une valeur hexadécimale à 0x01 pour obtenir des informations retour de l'actionneur. Cependant, cela envoyait une mauvaise commande et nous n'arrivions pas à jouer une musique avec cette valeur.

11. Conclusion :

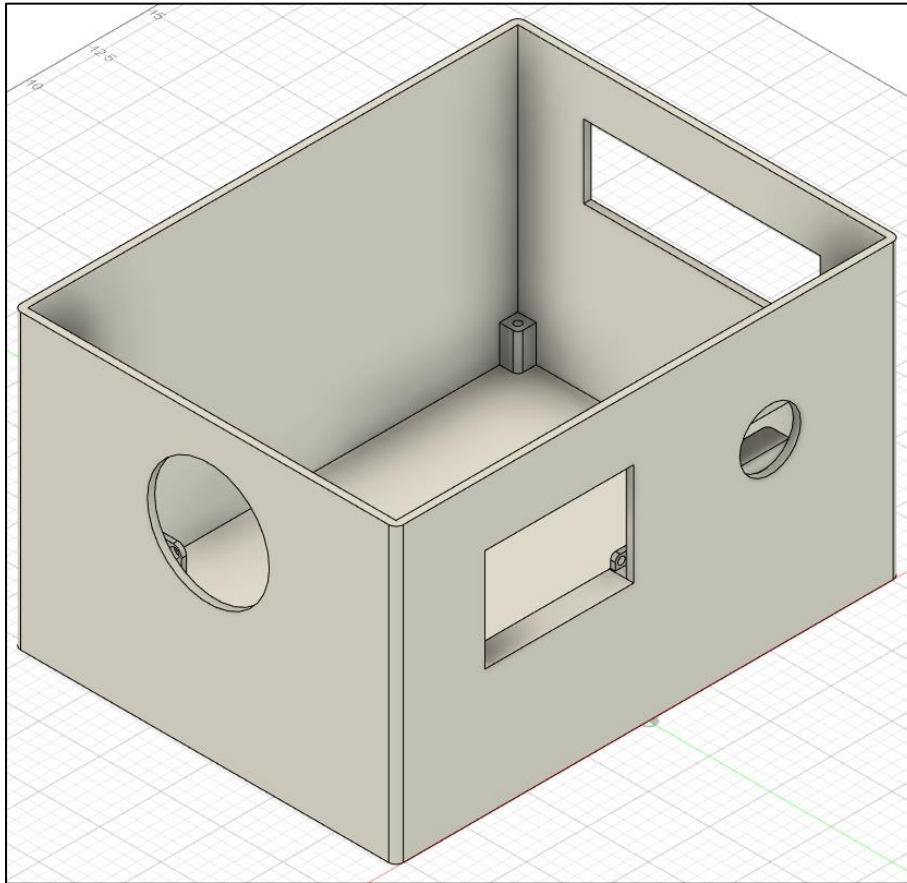
Nous sommes fiers de la réalisation de ce projet réveil. Même si la partie mécanique n'est pas encore fonctionnelle, le système de base du réveil marche très bien. Nous avons aussi ajouté certaines options avec succès, le choix de musique personnalisée notamment. Nous avons perdu du temps sur l'implémentation des capteurs et le débogage. Pour finaliser totalement le projet, il ne nous manquerait plus qu'à assurer le fonctionnement simultané de tous les périphériques.

12. Design CAO :

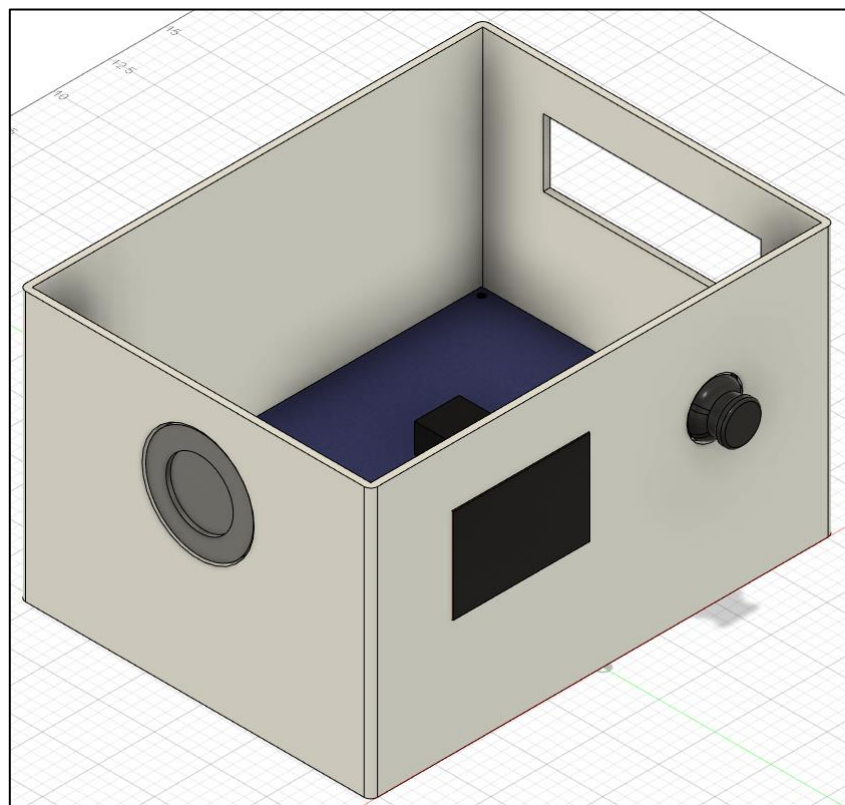
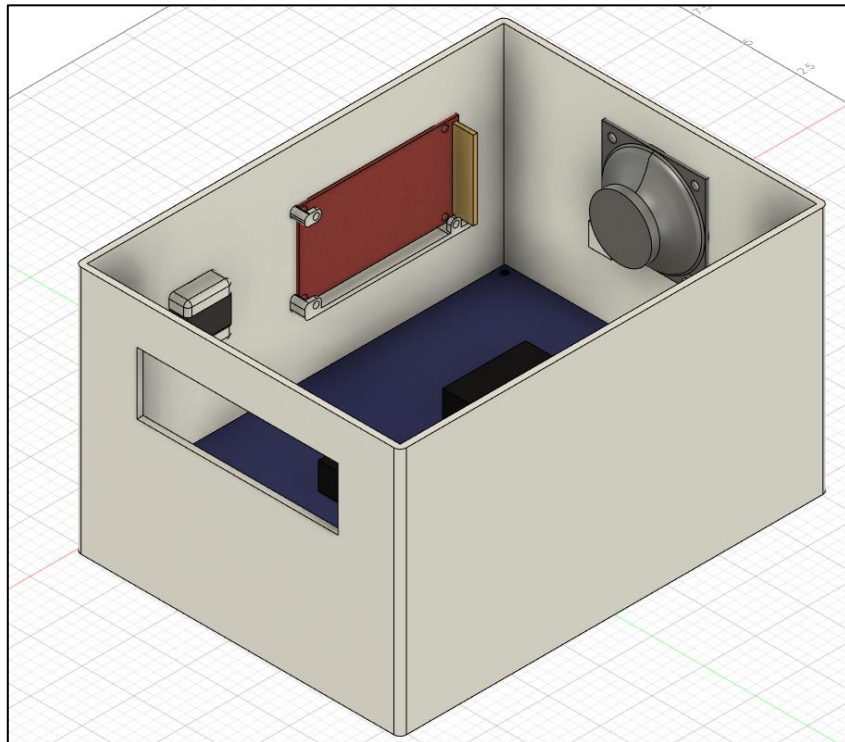
Les composants ont tout d'abord été mesurés et modélisés dans Fusion 360 pour s'assurer que les contraintes étaient respectées :



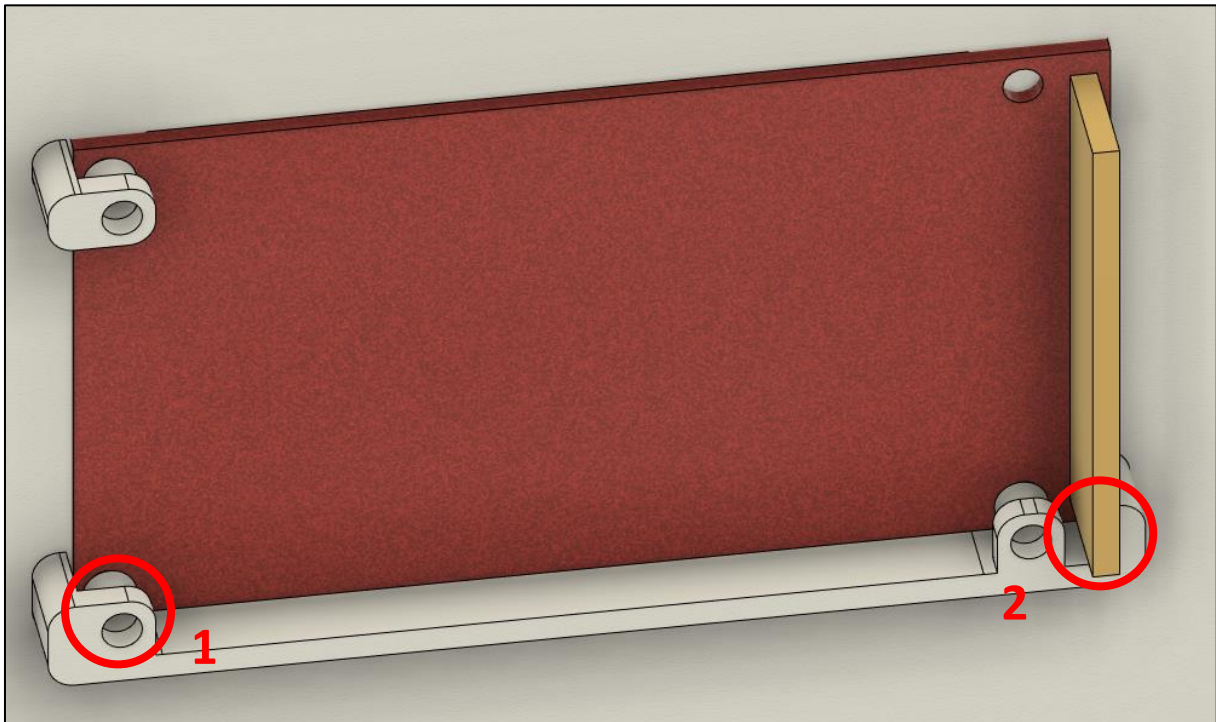
La 2^e étape était de réaliser le boîtier avec les encastréments pour les composants.



Vue avec les composants :

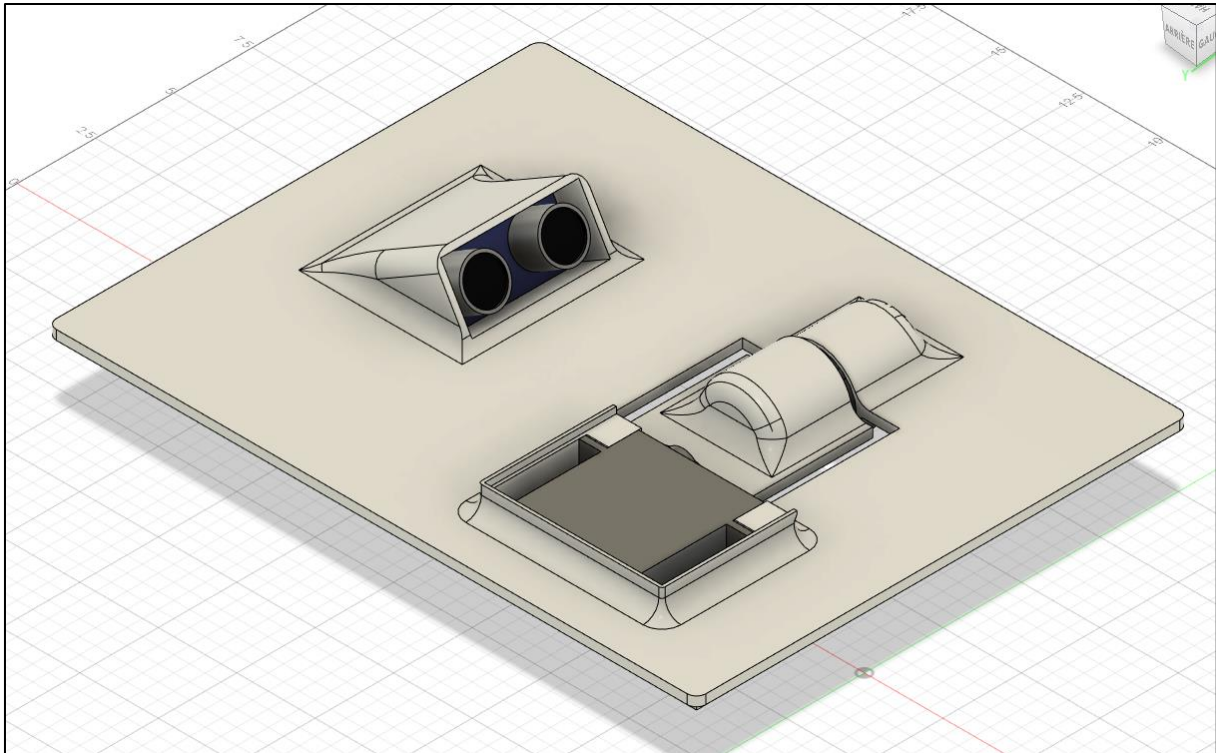


Exemple de système de fixation :

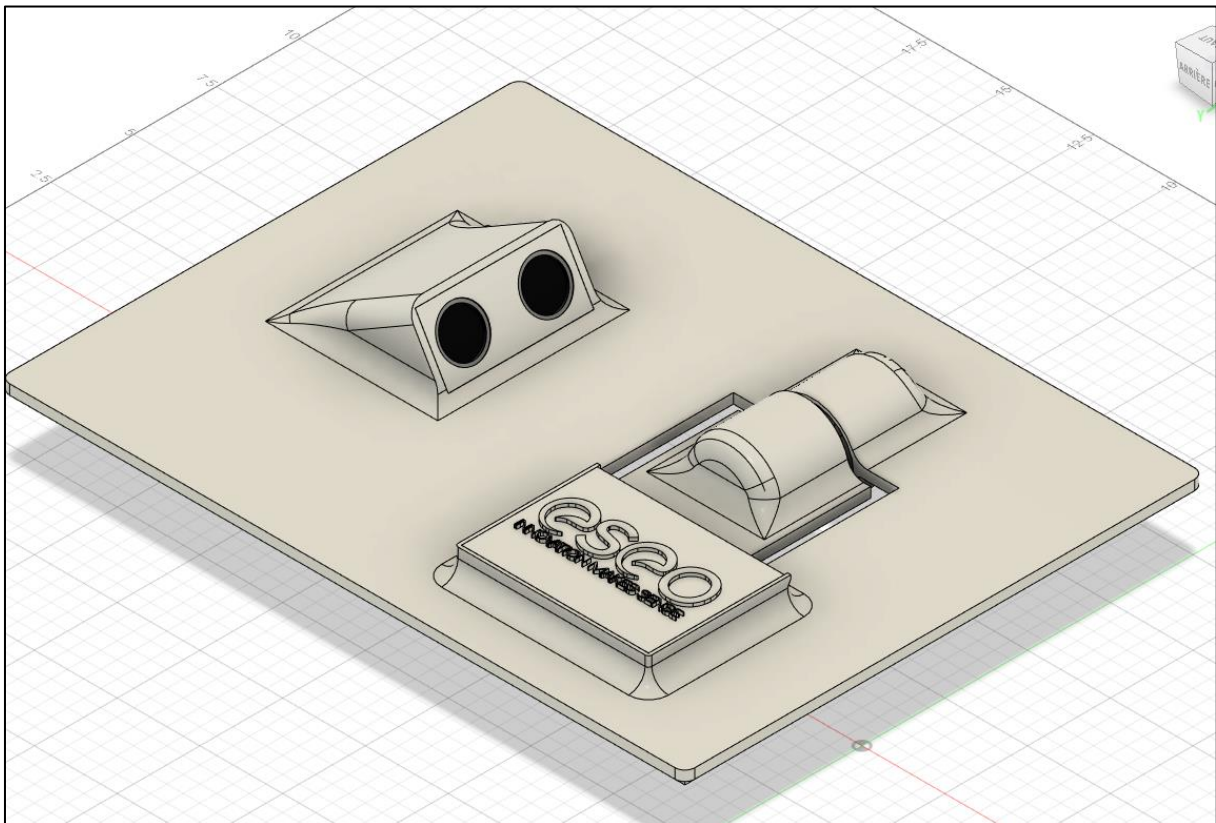


- 1) Espace de 0.5mm (comblé ensuite par un boulon de 0.3mm et l'épaisseur de la carte de 0.2mm). Permet de glisser le composant de haut en bas, puis de le pousser dans la paroi (l'épaisseur de la partie LCD et de la paroi sont de 0.3mm)
- 2) Espace prévu pour laisser l'accès aux broches de l'écran LCD. Rebord permettant le maintien horizontal de la carte.

La 3^e étape était de réaliser le couvercle de la boîte, qui va comprendre un emplacement pour maintenir le capteur ultrasons et la partie mécanique avec la trappe et servomoteur.

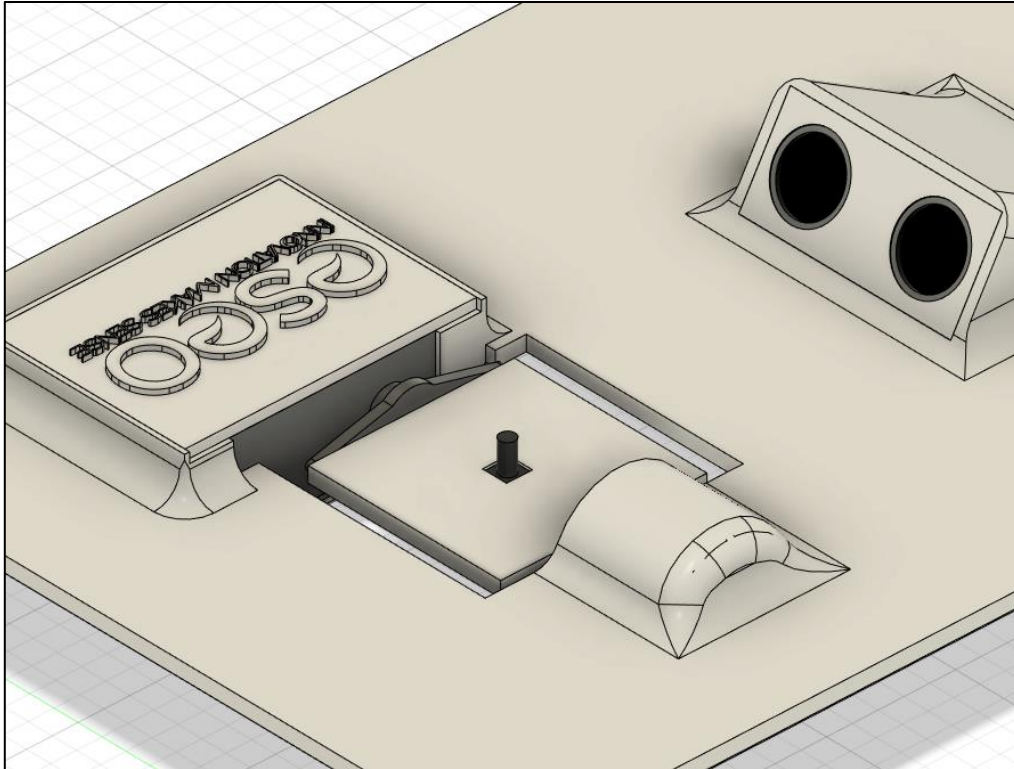


Avec les plaques de protection des composants :

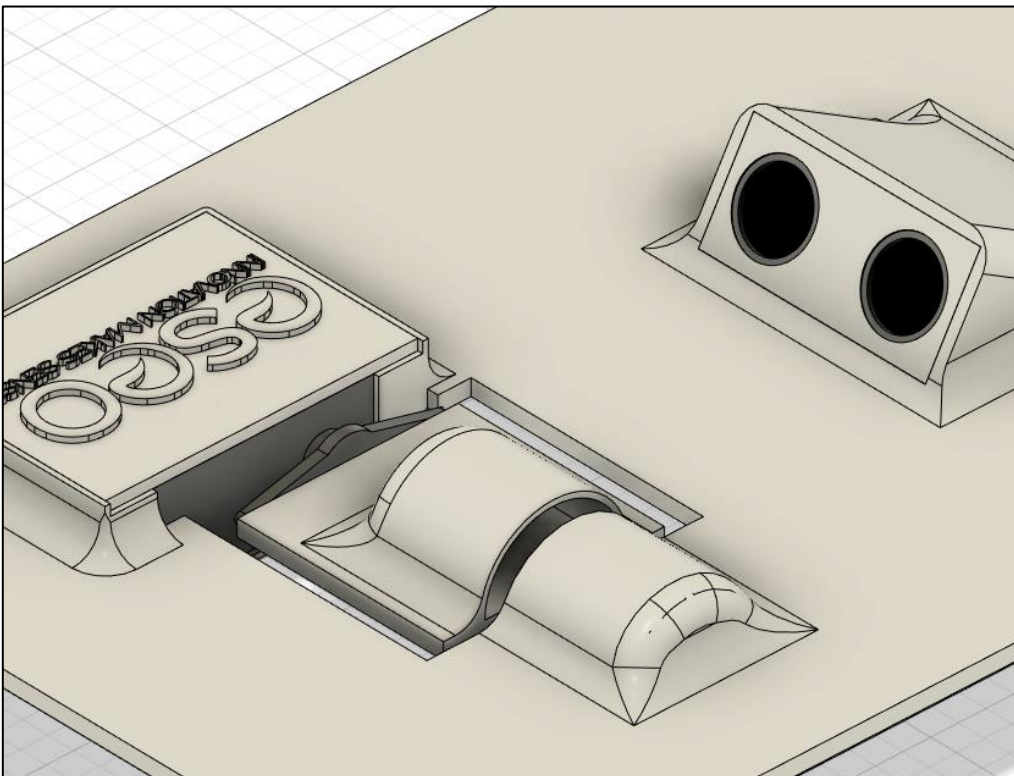


Fonctionnement de la trappe :

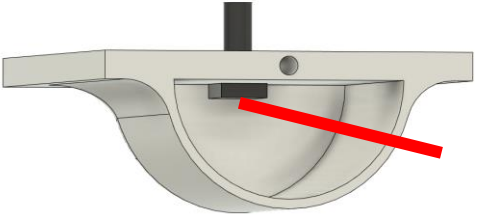

Cas 1 : La main n'est pas dans la zone de détection, la trappe est donc placée coté Bouton.

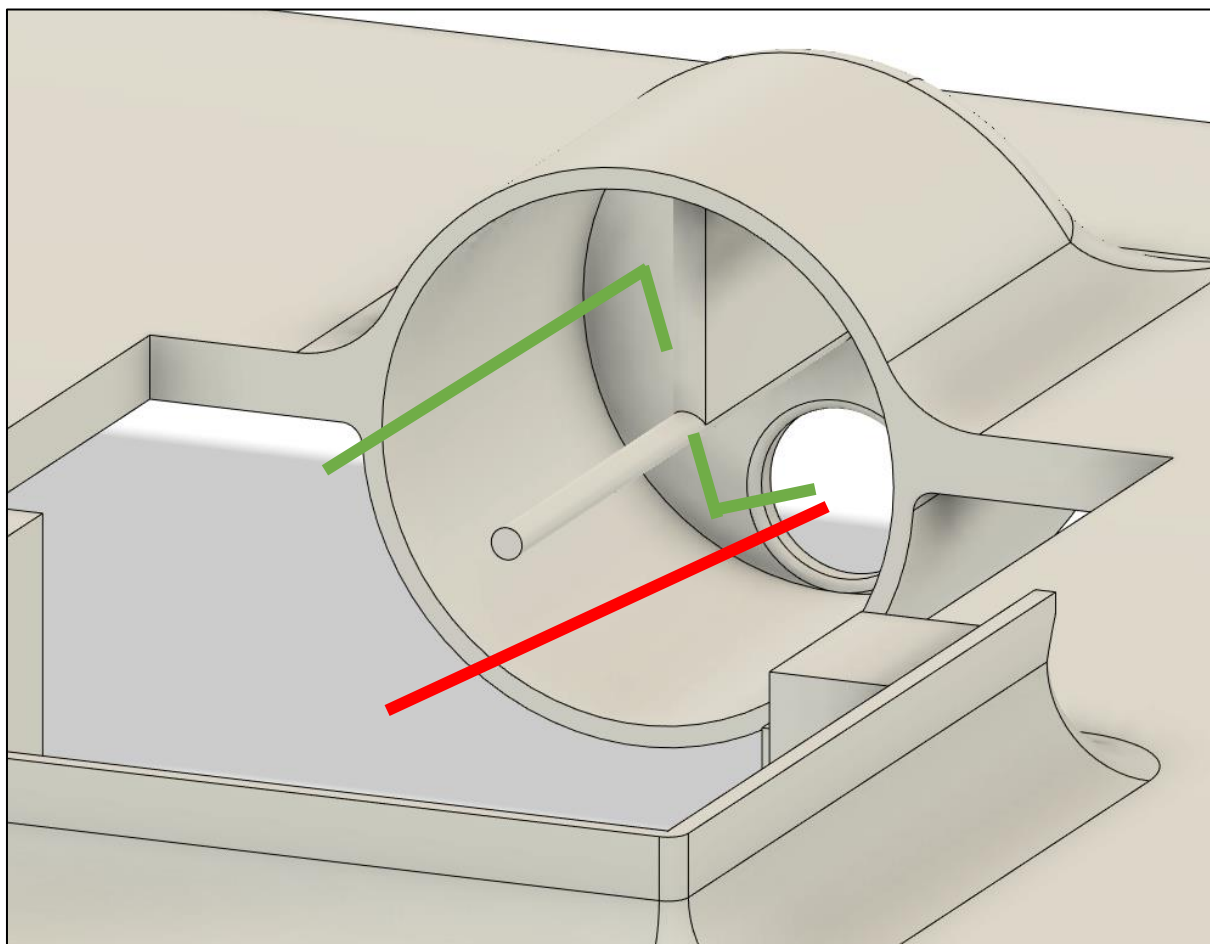


Cas 2 : La main est dans la zone de détection, la trappe effectue une rotation de 180°.

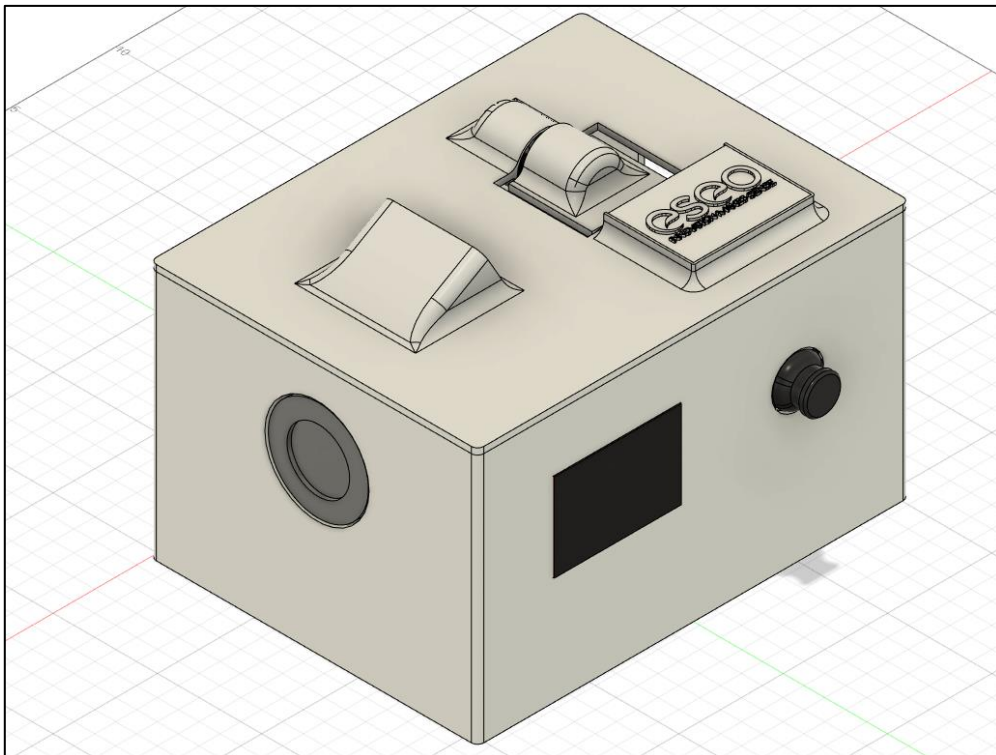
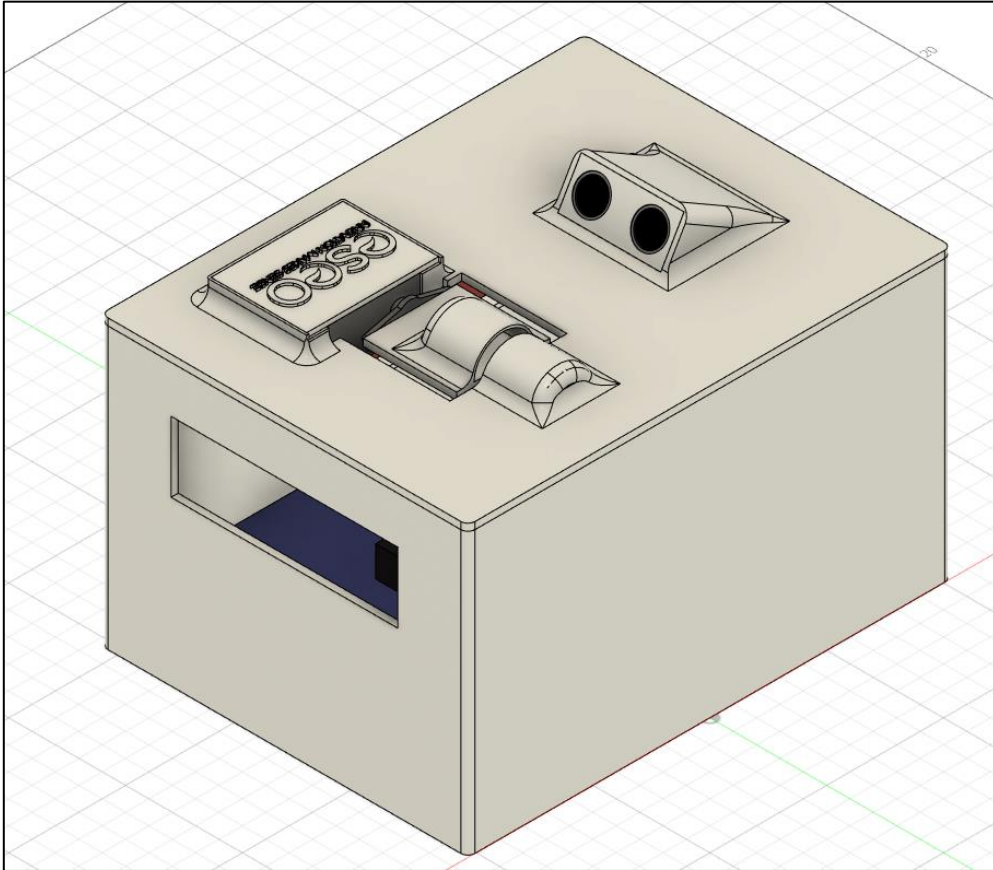


Considération : Gestion des câbles lors de la rotation de la trappe

Cas 1 : Trappe coté Bouton	Cas 2 : Trappe retournée
	



Version finale une fois tous les composants assemblés :

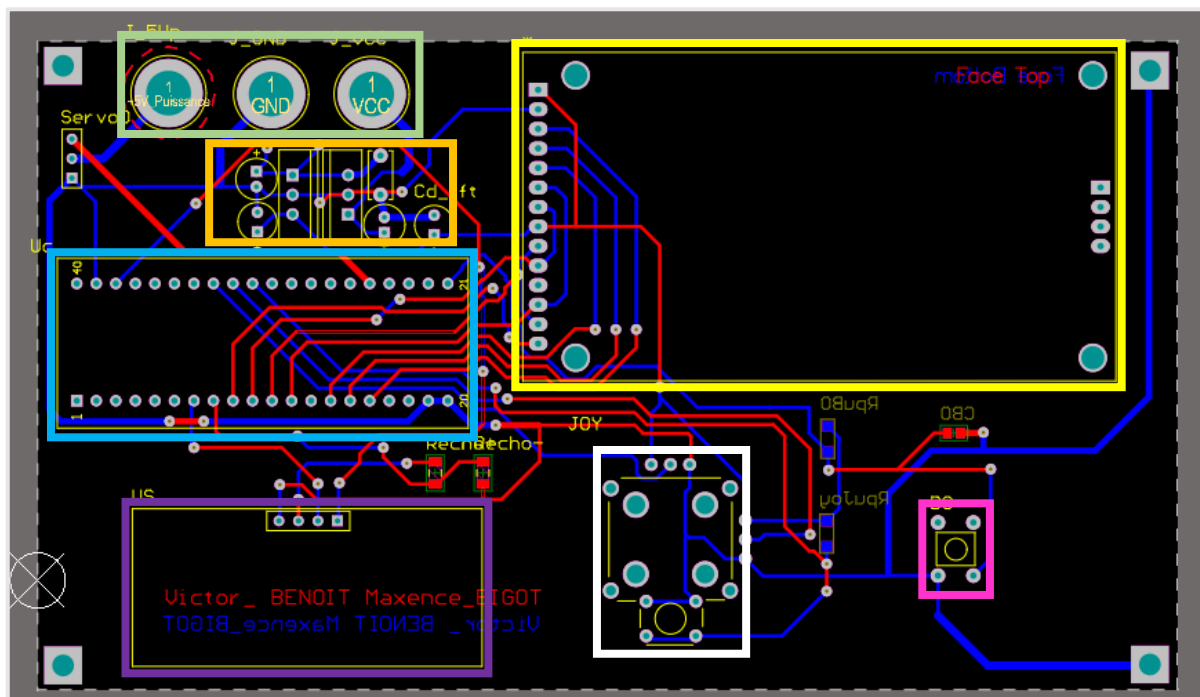


13. Design PCB :

Nous penserons à joindre à l'archive zip livrable : notre fichier SchDoc, notre fichier PcbDoc ainsi que le PDF généré par Altium contenant schéma, vue des couches de routage et vue 3D.	<input checked="" type="checkbox"/>
Les connecteurs sont accessibles	<input checked="" type="checkbox"/>
Un plan de masse est présent sur chaque couche de cuivre	<input checked="" type="checkbox"/>
Les GND sont correctement tous reliés par des pistes (même si le plan de masse les regroupe ensuite).	<input checked="" type="checkbox"/>
Le DRC (Design Rules Check) passe sans aucune erreur	<input checked="" type="checkbox"/>
Présence d'un connecteur UART (Rx + Tx + GND), parce que c'est toujours utile	<input type="checkbox"/>
Si le microcontrôleur est placé en CMS : présence d'un connecteur de programmation (SWDIO + SWDCLK + GND)	<input type="checkbox"/>

Pas d'arrivée en TOP non soudables sur des composants traversants	<input checked="" type="checkbox"/>
Noms des étudiants et site (Angers/Vélizy/Dijon) visibles sur le TOP.	<input checked="" type="checkbox"/>
Carte aux dimensions raisonnables (pas trop d'espace libre !)	<input checked="" type="checkbox"/>
Pistes ≥ 12 mils ; et au moins 25 mils lorsque c'est possible	<input checked="" type="checkbox"/>
Clearance ≥ 20 mils lorsque c'est possible. (Ponctuellement ≥ 10 mils)	<input checked="" type="checkbox"/>
Clearance des plans de masse réglée à 40 mils.	<input checked="" type="checkbox"/>

Un schéma correct	75/100
Qualité du placement (un bon placement garanti un bon routage)	80 /100
Qualité du routage	60 /100



Rôles des composants :

En vert, les bornes d'alimentation et de masse : Elles sont placées à l'extrémité de la carte pour faciliter l'emploi. Reliées à l'alimentation, elles permettent d'alimenter la carte.

En orange, la fonction alimentation : Permet d'alimenter les différents composants en 5V et 3,3V.

En bleu, la Bluepill : Elle permet la gestion des périphériques.

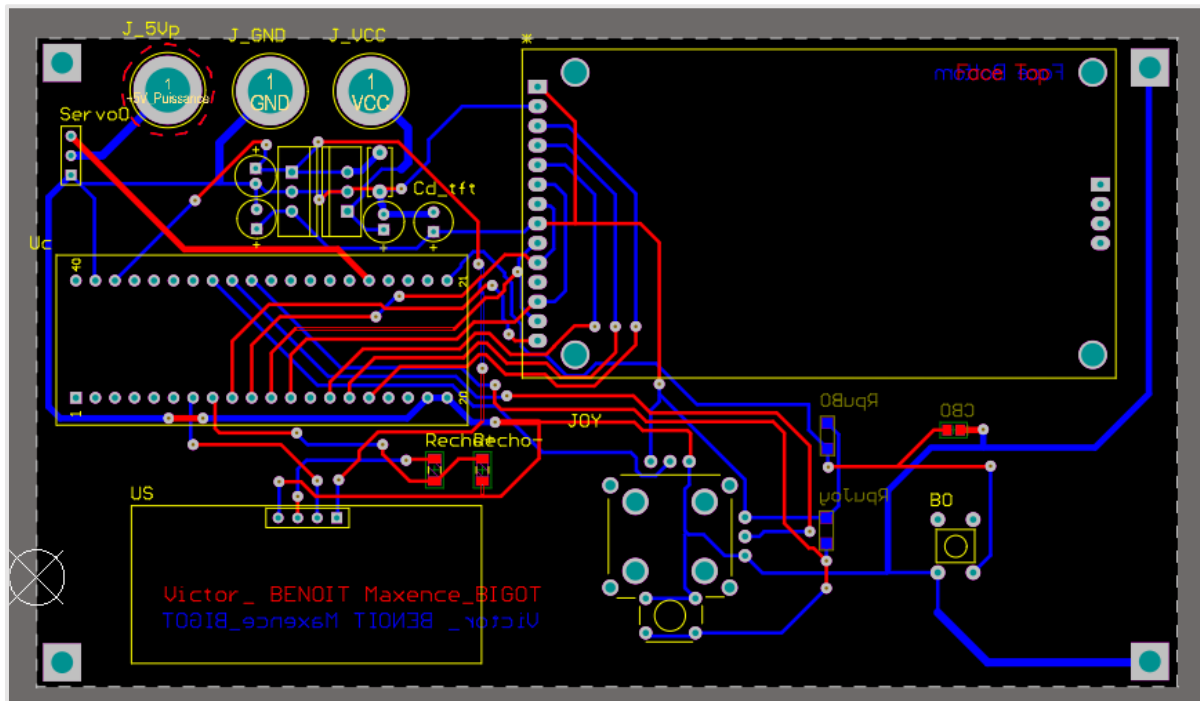
En violet, le capteur ultrason : Il communique par liaison UART et permet de détecter la distance à laquelle un objet est détecté, dans notre cas la main.

En jaune, l'écran LCD : Par défaut, il affiche l'heure. Si l'utilisateur veut changer des paramètres, il affiche le menu d'options.

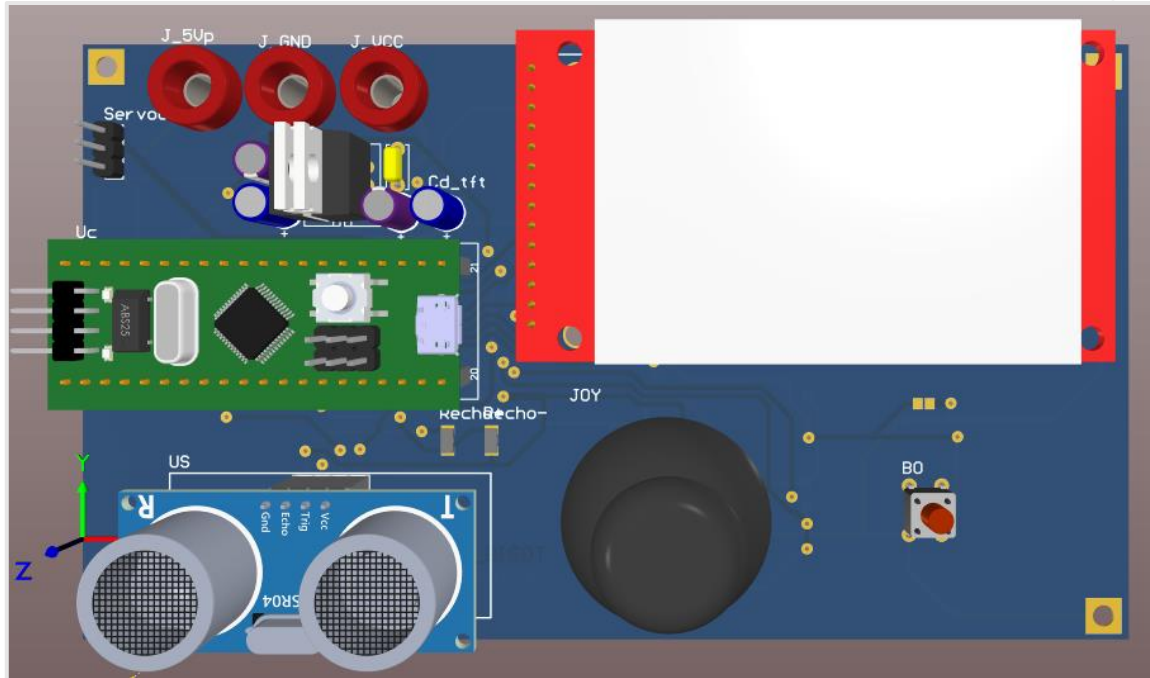
En blanc, le joystick : Il permet de naviguer dans le menu d'options. (Remplacé par deux boutons dans la version réelle car utilisation plus pratique.)

En rose, un bouton : Il permet de désactiver l'alarme quand celle-ci sonne.

Vue 2D du routage :



Vue 3D de la carte :



Difficultés rencontrées :

- Routage des nets au niveau de la Bluepill : Il y avait beaucoup de chevauchements donc j'ai dû utiliser beaucoup de via.
- Prise en main du logiciel : Compliquée au début, mais une fois pris en main, l'utilisation est agréable.
- Certains composants introuvables : Je n'ai pas trouvé le lecteur mp3, l'amplificateur, et l'hautparleur donc je me suis résolu à faire une version sans.

Estimations des couts de production :

Data Set: Customer - Basic PCB Analysis

Top View

Basic PCB analysis

- Generate PCB images
- Determine number of layers
- Determine board size

Full PCB analysis

A full automatic analysis of your data set will be performed after you add your job to a basket. During this more advanced analysis, the already determined parameters will be verified and extra parameters will be determined.

PCBConfigurator

Unit: **mm** | Inch

Select a Service

PCB proto | STANDARD prot | DEFINED REFERENCE prot | RF prot | RF prot | SMT prot | SMT prot

Number of layers: **2** | 4 | More Options

PCB size: **148.21** X **84.33** mm
Measured: 148.21 mm X 84.33 mm

Delivery format: **Single PCB** | Panel

eC-registration compatible PCB: **Yes** | No

Stencils

Top stencil: **Yes** | No

Bottom stencil: **Yes** | No

Proceed to Summary
to Review the Offer or Save to Basket

PCB proto service

PCB quantity: **1** (PCB(s))

Lead time: **Default (3 Working Days)**

Board surface / Order surface: **1.25 dm² / 1.25 dm²**

Est. shipment date: **23-01-2023**

	Net	Gross*
Single PCB	€ 60.55	€ 73.27
Total boards	€ 60.55	€ 73.27
Total**	€ 60.55	€ 73.27

* The gross prices include 21 % VAT.
** Login for available shipment dates, options and prices.

Business customer? Select here ☐

On a donc un prix total de 73.27€ (sans compter les frais de port).