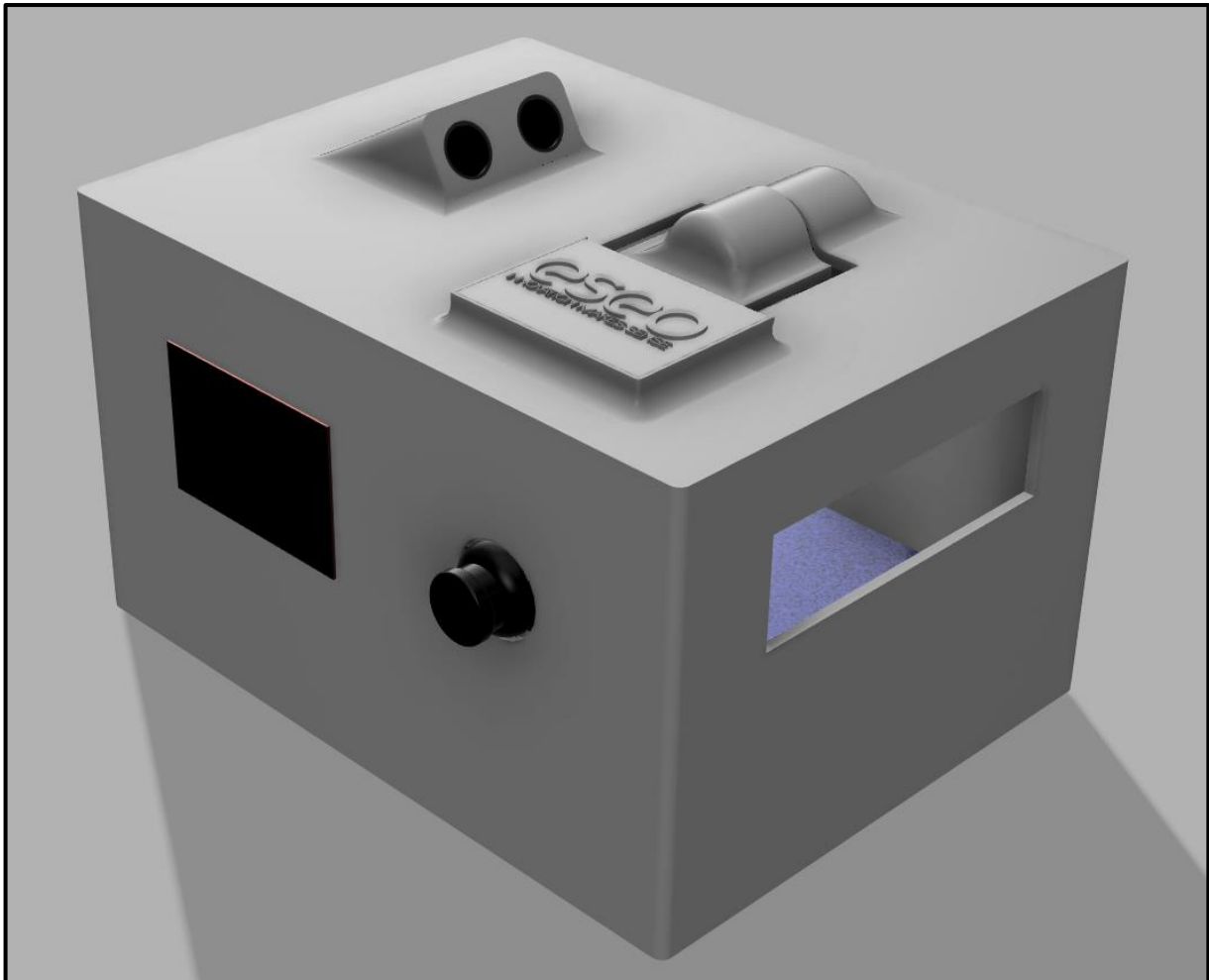


# Rapport Intermédiaire

## 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.

Le projet débute en novembre et se termine fin janvier. Ce rapport contient l'avancement qui a été fait sur les 8 premières séances, le cahier des charges, les composants utilisés et les ports associés.

## Sommaire :

1. Cahier des charges
2. Composants utilisés
3. Tableau d'affectation des ports
4. Carnet de bord
5. Point Modélisation
6. Point Software
7. Etat d'avancement et analyse du projet
8. Compléments envisagés

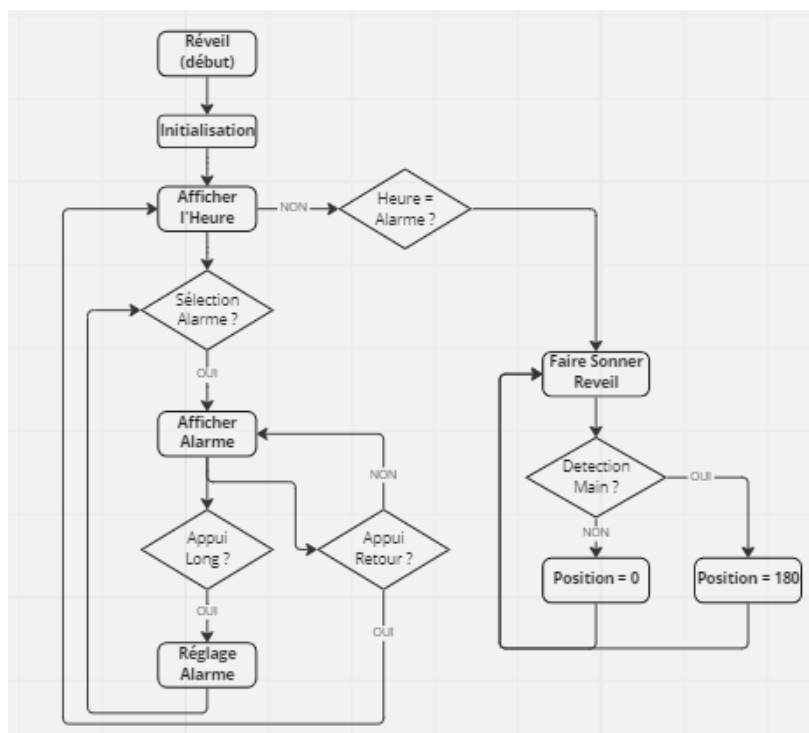
# 1. Cahier des charges :

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 mais aussi la vidéo lancée en même temps. 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.

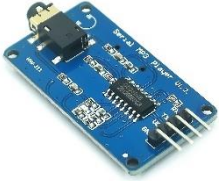


Les contraintes sont principalement physiques : Il faut réaliser une structure de boîtier à la fois solide et légère, de façon à avoir un réveil qui ne casse pas s'il tombe de la table de nuit (ou qui est jeté par l'utilisateur fou de rage), mais qui soit facile à déplacer. De plus le boîtier doit posséder des logements permettant d'encastrer 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.

D'un point de vue Software, le système sera représenté par une machine à états.



## 2. Composants utilisés :

Nom/Référence du Composant	Utilisation/Fonctionnement
<b>Ecran LCD 3 pouces</b> <b>ILI9341</b> 	<p>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.</p>
<b>Servo Moteur</b> <b>MG996R</b> 	<p>Moteur utilisé pour entrainer la plateforme dans une rotation de 180°. Le choix du duty du signal PWM permet de choisir sa rotation :</p> <p>0.1 → 90° 0.2 → 180° Alimentation en 5V.</p>
<b>Capteur Ultrason</b> <b>HCSR04</b> 	<p>Capteur utilisé pour détecter la présence de la main à partir d'une certaine distance réglable avec le soft. Alimentation en 5V.</p>
<b>Bouton poussoir</b> 	<p>Bouton placé sur la trappe permettant d'éteindre l'alarme quand elle sonne.</p>
<b>Joystick</b> 	<p>Joystick placé à coté de l'écran permettant de naviguer dans le menu de paramètres.</p>
<b>Hautparleur</b> <b>TMQ60500</b> 	<p>Hautparleur permettant d'avoir une alarme musicale personnalisée.</p>
<b>Amplificateur Audio</b> <b>PAM8403</b> 	<p>Amplificateur permettant de régler le volume sonore de l'hautparleur.</p>

<p><b>Lecteur mp3 HW311</b></p> 	<p>Lecteur mp3 utilisé pour la lecture des musiques téléchargées sur la carte SD.</p>
<p><b>BluePill</b></p> 	<p>Placée sur la DEEP Purple Complete Board. Permet la gestion des périphériques.</p>
<p><b>Nucleo STM32</b></p> 	<p>Permet à la BluePill d'utiliser la partie débogueur de la Nucleo STM32</p>

### 3. Affectation des ports :

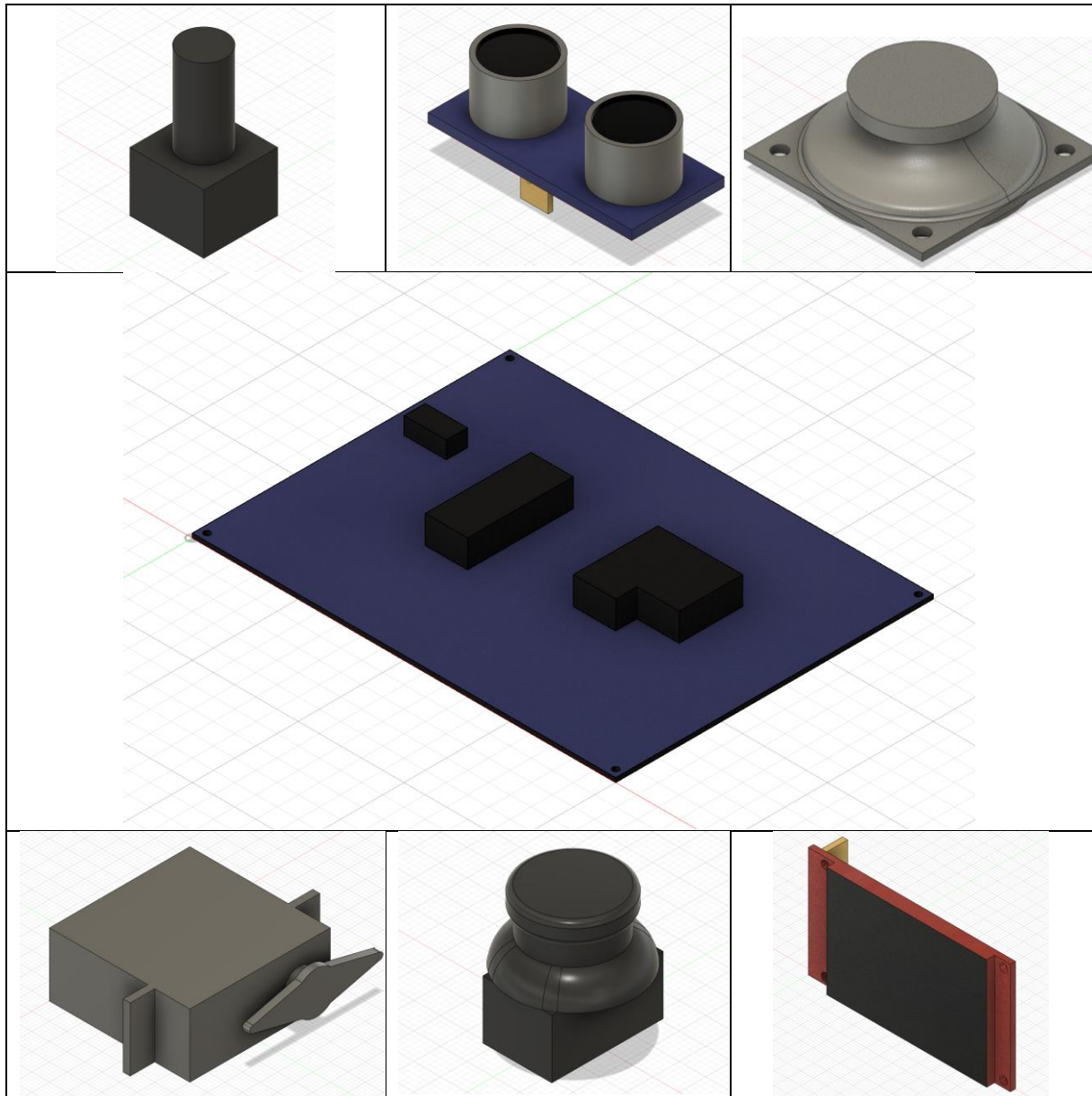
Composants	Pins (Côté composant)	Pins (Côté Bluepill)
Ecran LCD	VCC	3,3V
	GND	GND
	CS	PB11
	RESET	PB10
	D/C	PB1
	MOSI / T_DIN	PA7
	SCK / T_CLK	PA5
	LED	3,3V
	MISO / T_DOUT	PA6
	T_IRQ	PA4
	T_CS	PA0
Servo Moteur	VCC	5V
	GND	GND
	PWM	PA8
Capteur Ultrason	VCC	5V
	RX	PA2
	TX	PA3
	GND	GND
Lecteur MP3	VCC	5V
	TX	PB7
	RX	PB6
	GND	GND
Amplificateur	VCC	5V
	GND	GND
Joystick	VCC	3,3V
	GND	GND
	JOYX	PB5
	JOYY	PB4
	JOYBTN	PB3
Bouton Poussoir	VCC	3,3V
	GND	GND
	DATA	PB10

## 4. 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	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 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

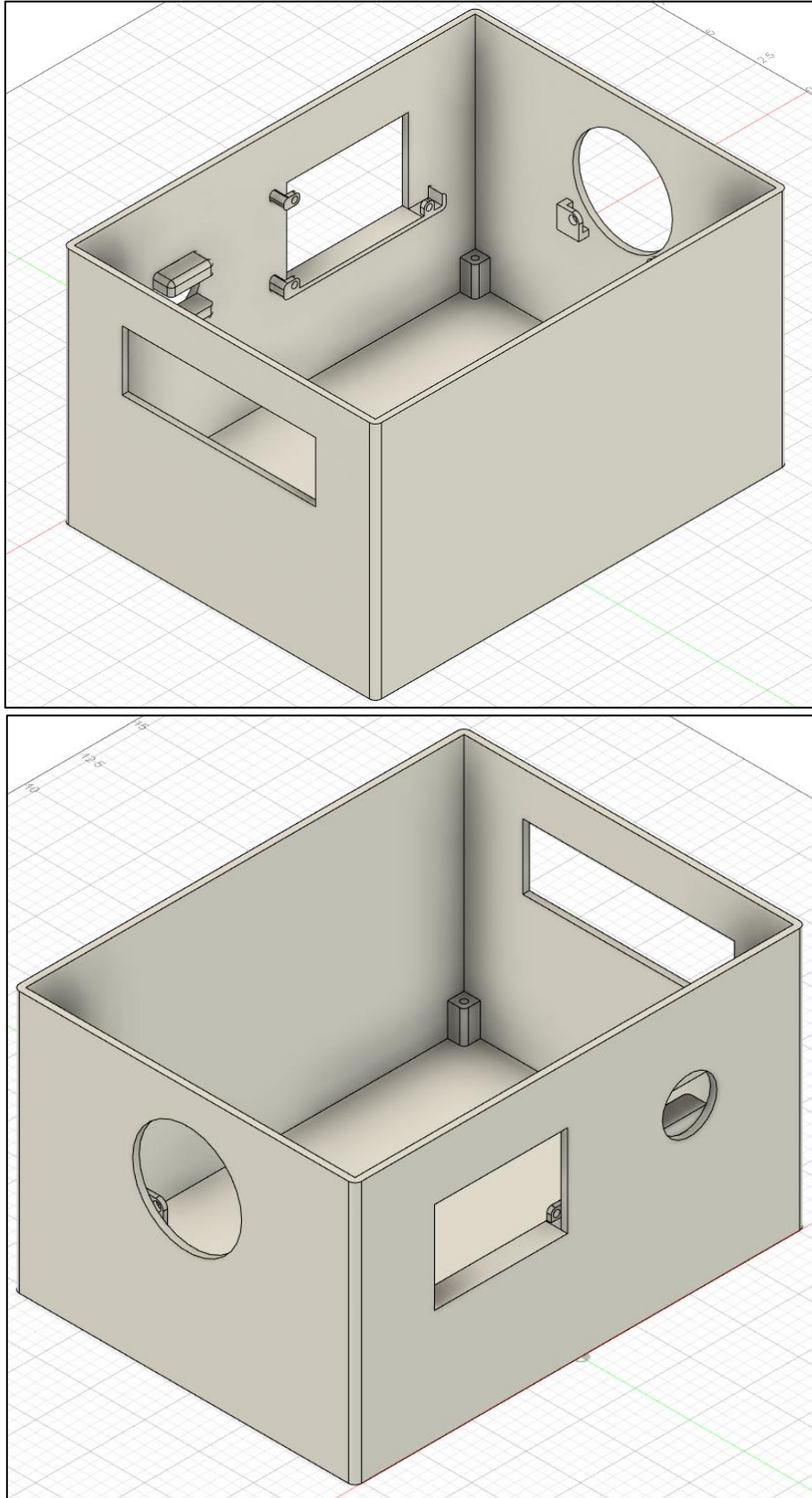
## 5. Point Modélisation :

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

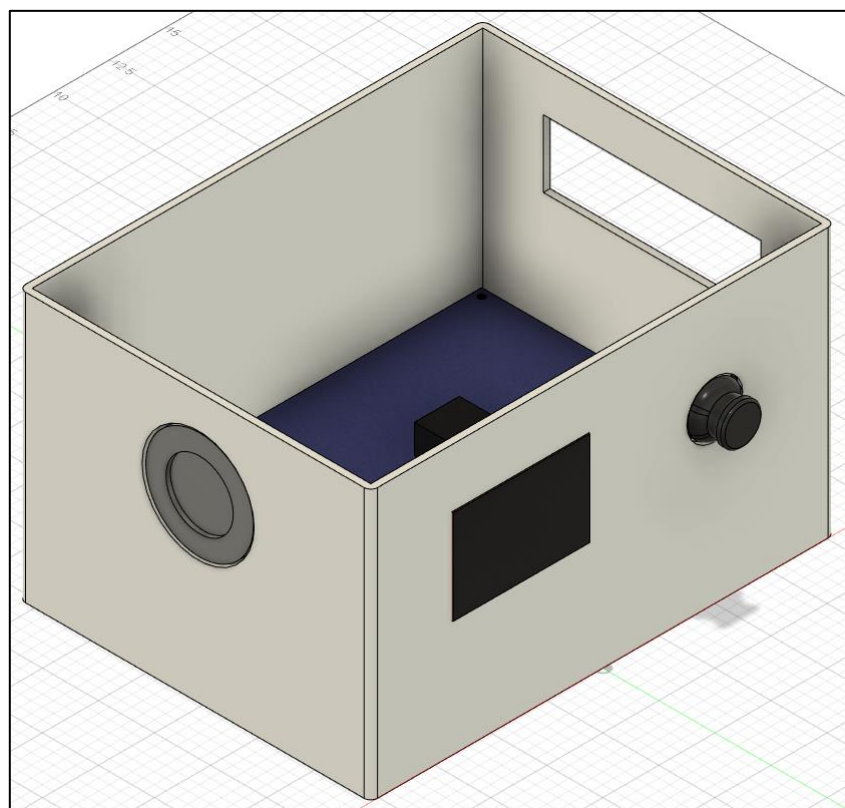
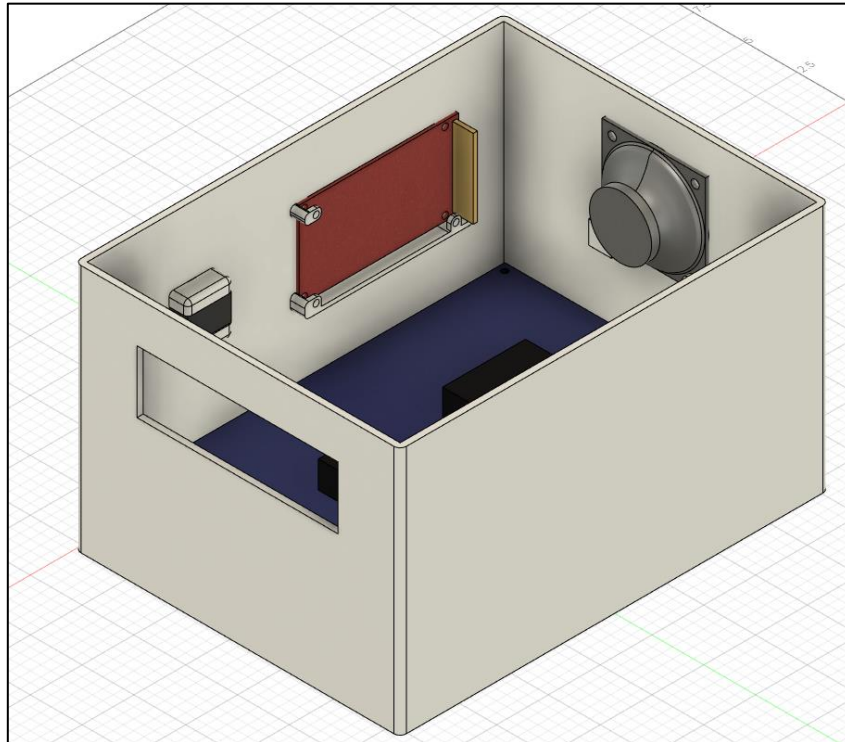




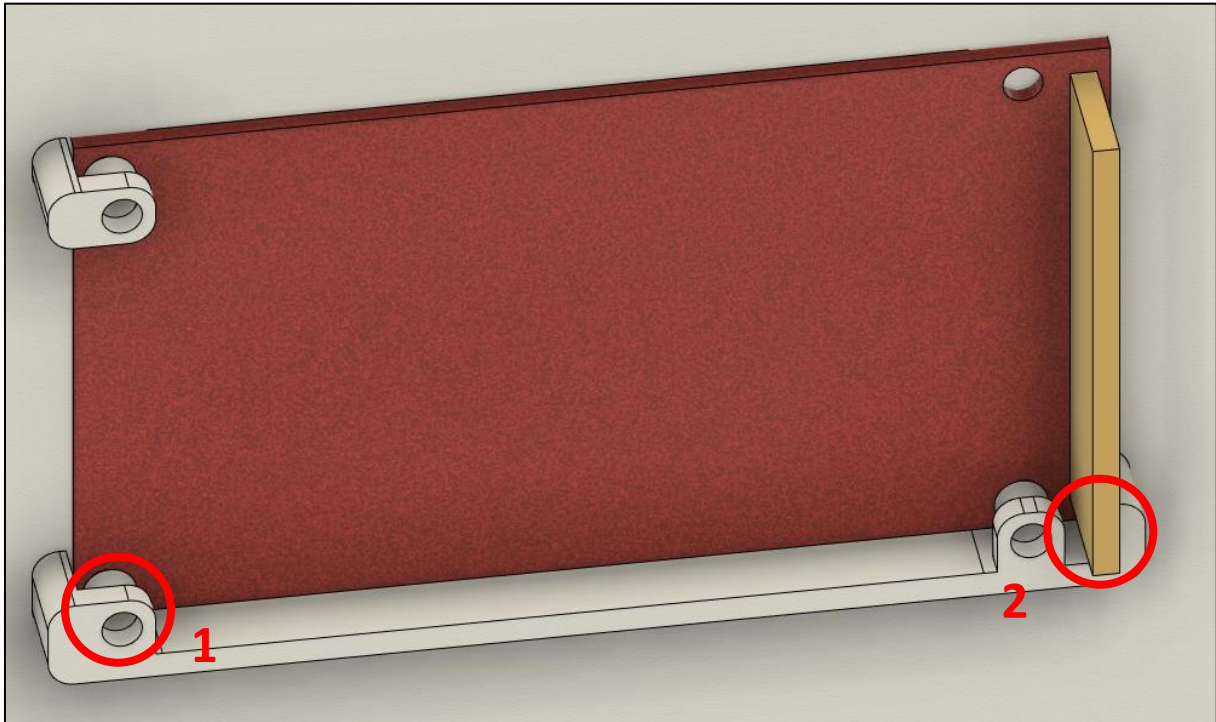
La 2<sup>e</sup> étape était de réaliser le boîtier avec les encastréments pour les composants.



Vue avec les composants :



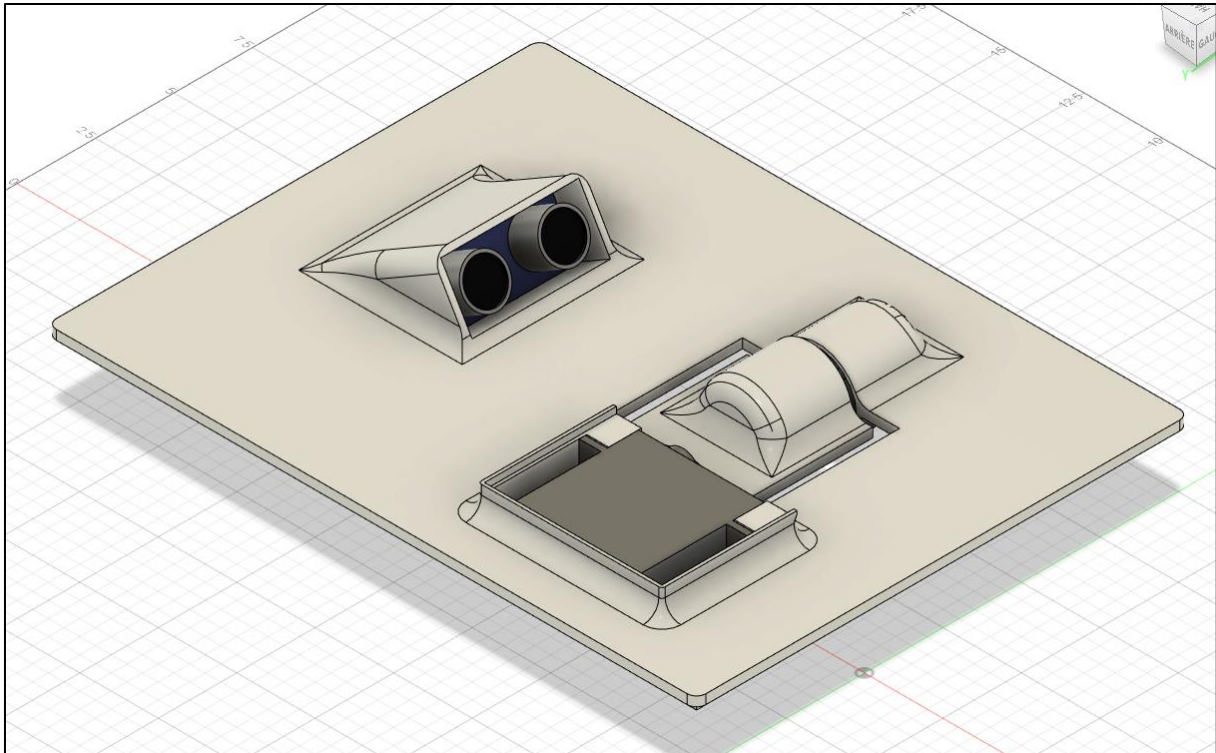
**Exemple de système de fixation :**



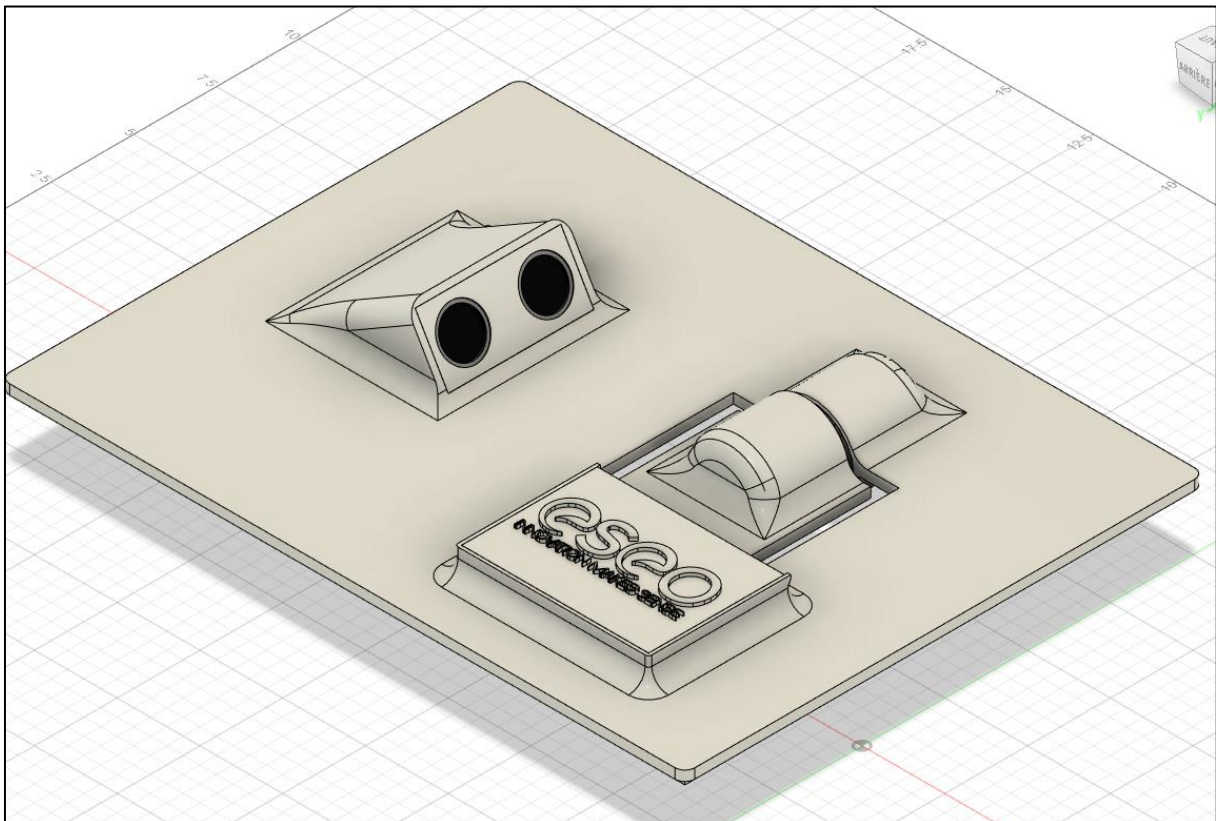
- 1) Espace de 0.5mm (comblé ensuite par un boulon de 0.3mm et l'épaisseur de la carte de 0.2mm). Permet de glisser le composant de haut en bas, puis de le pousser dans la paroi (l'épaisseur de la partie LCD et de la paroi sont de 0.3mm)
- 2) Espace prévu pour laisser l'accès aux broches de l'écran LCD.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

**La 3<sup>e</sup> 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.



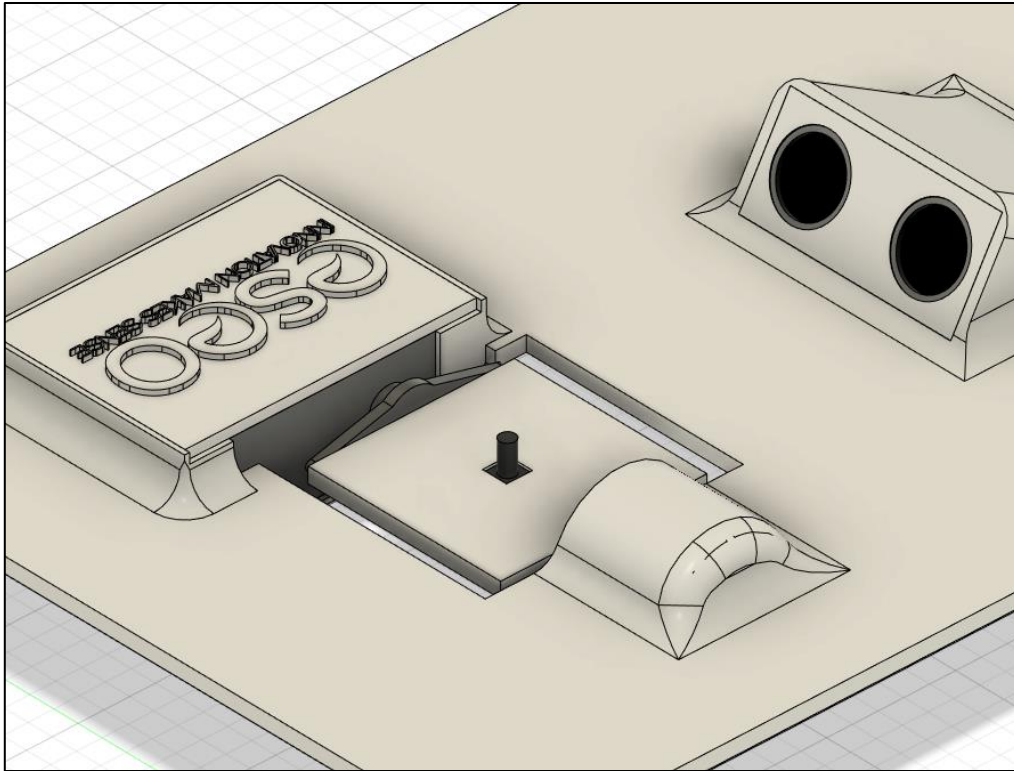


Avec les plaques de protection des composants :

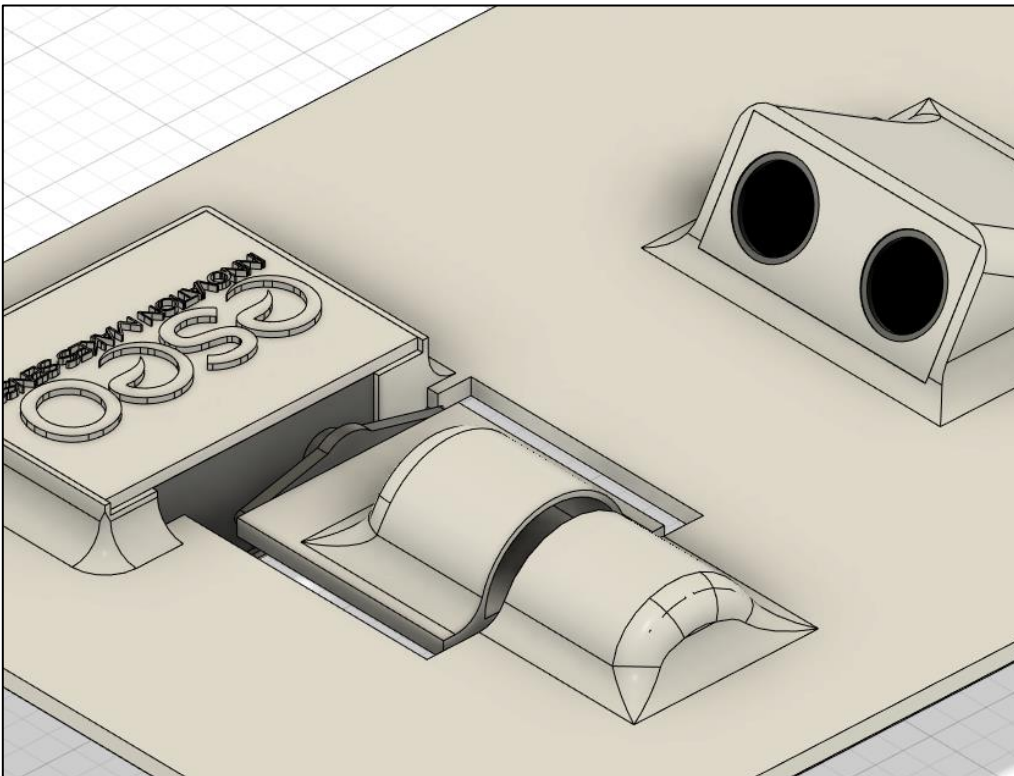


### Fonctionnement de la trappe :

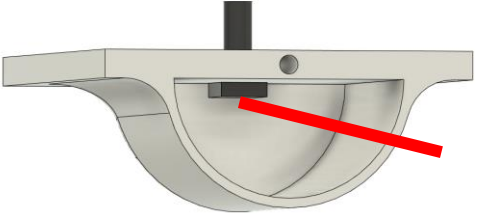

Cas 1 : La main n'est pas dans la zone de détection, la trappe est donc placée coté Bouton.

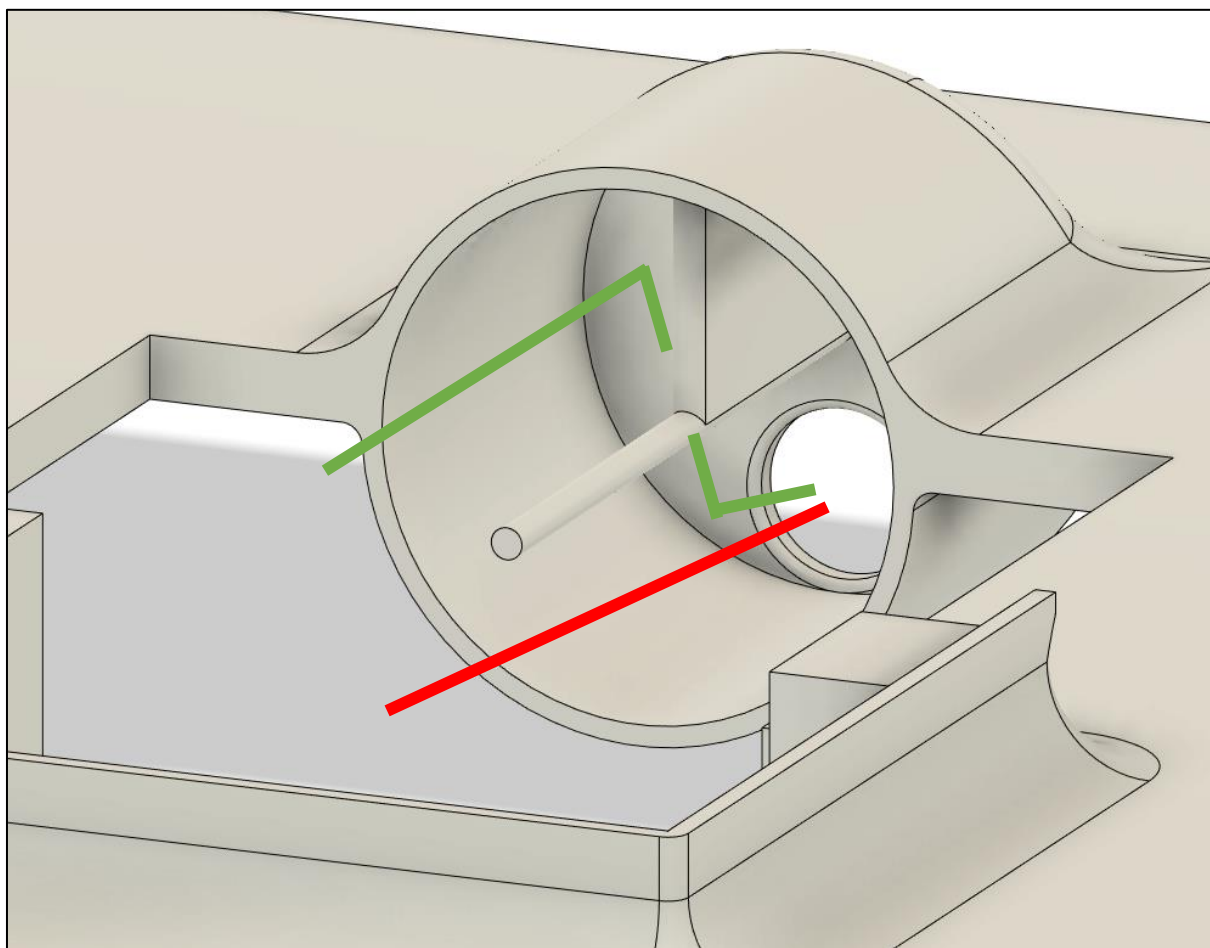


Cas 2 : La main est dans la zone de détection, la trappe effectue une rotation de 180°.



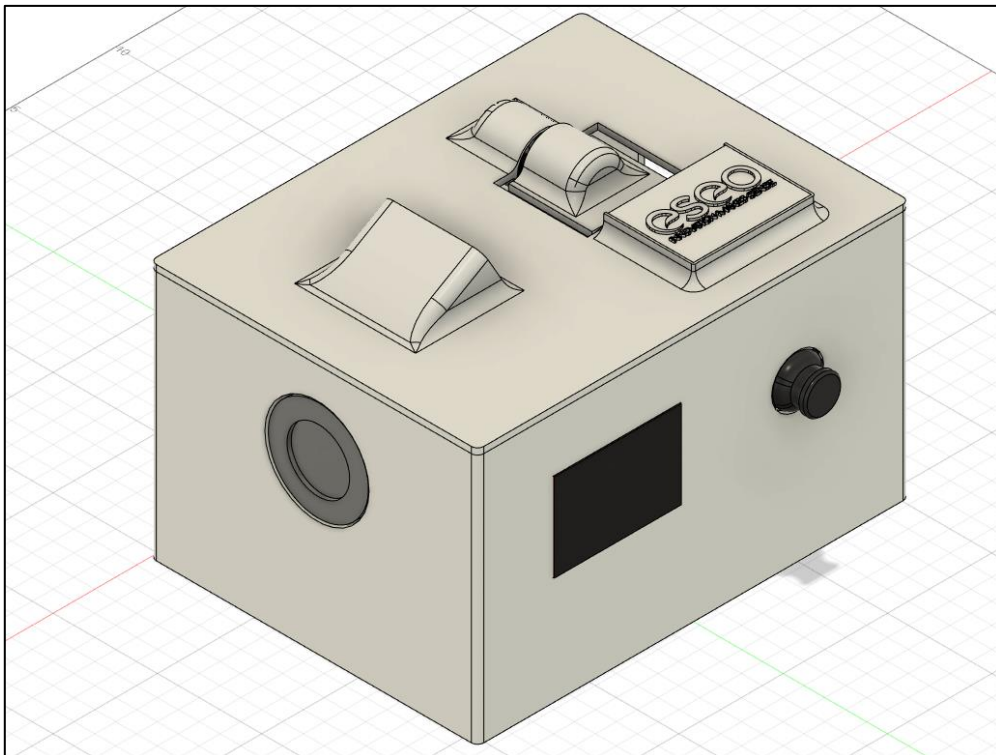
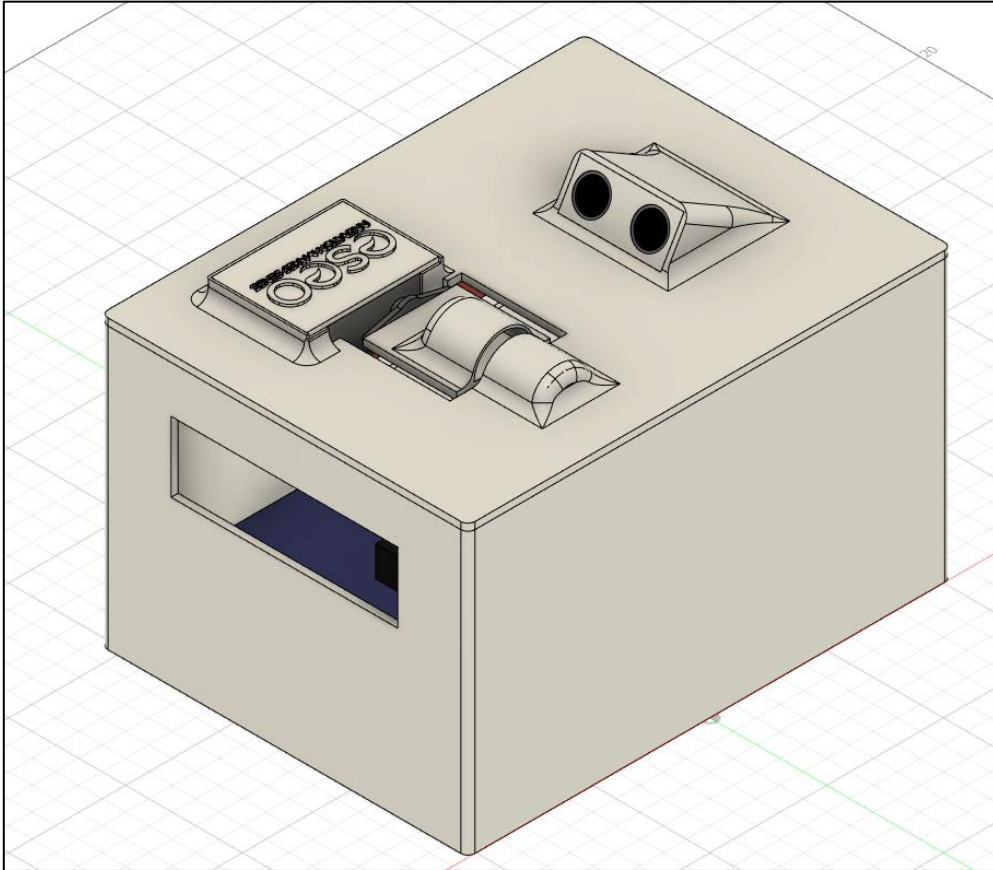
**Considération : Gestion des câbles lors de la rotation de la trappe**

Cas 1 : Trappe coté Bouton	Cas 2 : Trappe retournée
	





**Version finale une fois tous les composants assemblés :**



## 6. Point Software :

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

```
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);
```

avec la fonction *ILI9341\_Puts*.

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 32 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(&currentTime)* et *RTC\_get\_date(&currentDate)* pour récupérer nos informations. 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 9600Baud 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);
```

Pour la mise en place des différentes fonctions de notre réveil, nous avons commencé à implanter une machine à états. Nous avons défini les différents états dans une structure ENUM :

- Init : Initialisation des différents composants
- Affichage : Affichage sur l'écran de l'heure et de la date
- Song : Joue la musique du réveil

Nous créons ensuite les variables de l'état en cours et précédents, de la variable booléenne *entrance* à l'entrée d'une fonction. Le parcours de la machine à état se fait ensuite avec un switch des différents états mis en place.

## 7. Etat d'avancement et analyse du projet réalisé :

La partie Hardware est principalement 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 plus que finie car nous avons appris que l'impression des boîtiers ne se ferait pas après avoir réalisé la conception 3D du projet en entier.

Le gros du travail reste la partie Software car tous les composants n'ont pas été testés. Nous avons prévu de réaliser des fonctions de test pour chaque composant avant de nous lancer dans la programmation de la machine à états.

## 8. Compléments envisagés :

Voici la liste des compléments que nous envisageons de faire en plus du projet de base :

- Routage de PCB (avec bluepill = 3 pts    CMS microcontrôleur nu = 4pts)	
- Utilisation d'un analyseur logique pour déchiffrer des trames (2 pts)	
- Mesure de conso selon scénarios (2 pts)	X
- Enregistrement de paramètres en flash (1pt)	X
- Design CAO d'un boîtier (2 pts)	X
- Documentation doxygen du code source (1pt)	
- Jeu de tests pour valider une fonctionnalité software ou hardware (1pt)	X
- Gestion de version du code source (1pt)	X