

RAPPORT

# TPI TIME TIMER



Michel Gaëtan

DIVTEC - ELT

31/05/2024

## Table des matières

1.	Introduction.....	4
1.1	Intérêts.....	4
1.2	Schéma de principe.....	5
1.3	Fonctionnement.....	6
1.4	Caractéristiques techniques.....	7
1.4.1.	Connectique et environnement extérieur.....	7
1.4.2.	Éléments imposés.....	7
1.4.3.	Fonctionnalités.....	7
2.	Planification.....	8
3.	Développement de la solution .....	9
3.1	Solution imaginée .....	9
4.	Choix des composants.....	10
4.1	Microcontrôleur .....	10
4.2	Alimentation .....	11
4.2.1	Connecteur USB-C .....	11
4.2.2	Gestion de charge.....	11
4.2.3	Batterie .....	11
4.2.4	Convertisseurs DC/DC.....	12
4.3	Interface UART et microSD.....	13
4.3.1	Interface UART .....	13
4.3.2	Interface microSD .....	13
4.4	Capteur d'orientation.....	13
4.5	Encodeur rotatif .....	13
4.6	Real Time Clock .....	14
4.7	Affichage .....	14
4.8	Gestion audio.....	14
4.8.1	I2S converter.....	14
4.8.2	Amplificateur audio .....	14
4.8.3	Haut-parleur .....	15
4.9	Power switch.....	15
5.	Développement circuits.....	16
5.1	Alimentation .....	16
5.1.1	Circuit de sélection d'alimentation .....	16
5.1.2	Circuit de charge de la batterie.....	17
5.1.3	Convertisseurs DC/DC.....	18



5.2	Interface UART .....	20
5.3	Interface SPI .....	20
5.4	Sélection du Chip Select de la carte micro SD.....	21
5.5	Circuit d'auto-reset de l'ESP .....	21
5.6	Connections de l'ESP32.....	22
5.7	Circuit de la RTC .....	23
5.8	Circuit interface I2S .....	24
5.9	Circuit de l'amplificateur audio .....	24
5.10	Circuit du capteur d'orientation.....	25
5.11	Circuit de l'écran OLED .....	26
5.12	Carte SD.....	27
5.13	Circuit de mesure de la batterie.....	27
5.14	Reset du circuit.....	28
5.15	Batterie et autonomie.....	28
6.	Schéma complet .....	29
7.	Liste de composants .....	31
8.	Conception des PCB.....	33
9.	Programmation.....	34
9.1	Fonctionnement général.....	34
9.2	Programmation de l'écran.....	35
9.3	Programmation de l'encodeur rotatif .....	35
9.4	Programmation de la RTC .....	36
9.5	Programmation de la lecture/écriture SD.....	38
9.6	Programmation du minuteur .....	39
9.7	Récupération du pourcentage de la batterie .....	39
10.	Fichiers de la carte SD .....	40
11.	Test final selon cahier des charges .....	41
11.1	Éléments internes imposés .....	41
11.2	Fonctionnalités imposées .....	41
12.	Problèmes rencontrés .....	42
13.	Améliorations .....	45
13.1	Améliorations hardware .....	45
13.2	Améliorations software.....	45
13.3	Autres améliorations.....	45
14.	Auto-évaluation du travail.....	45
15.	Conclusion .....	46



16.	Remerciements.....	46
17.	Annexes .....	46

# 1. Introduction

## 1.1 Intérêts

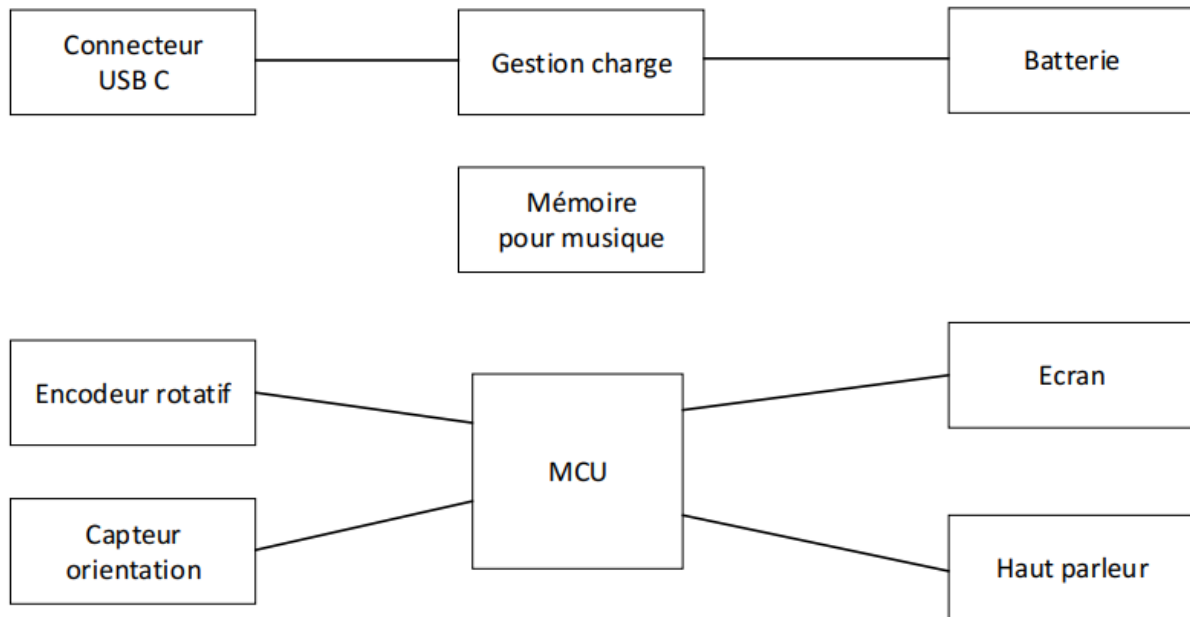
Le time timer est un outil visuel qui permet de matérialiser le temps qui passe en indiquant le temps restant à une activité. Il est très utilisé autant avec les enfants que pour les adultes.

Un exemple d'utilisation dans l'école serait de le mettre à la vue des étudiants pour montrer le temps restant pour faire un exercice ou une évaluation.

Cette version de time timer est électronique et affiche le temps sur un écran. Il est équipé d'un encodeur rotatif pour les réglages, d'un haut-parleur pour jouer une musique ou un bip lorsque que le décompte est terminé.



## 1.2 Schéma de principe



## 1.3 Fonctionnement

L'appareil s'enclenche à l'aide d'un switch. Il est alimenté par une batterie qui se charge à l'aide d'un port USB-C.

Le time timer est équipé d'un encodeur rotatif avec bouton. Il permet de faire certains réglages et de régler facilement le temps désiré. La durée maximale du décompte est de 4 heures.

Lorsque le time timer est positionné dans le sens « OFF » (exemple ci-dessous gauche) le décompte est stoppé. En retournant l'appareil dans le sens « ON » (image de droite), le décompte est actif. L'affichage retourne son contenu en fonction de la position de l'appareil.



Lorsque que la durée restante est plus grande que 1 heure, les heures et minutes sont affichées. Puis quand le temps passe en-dessous de l'heure, l'afficheur affiche uniquement les minutes et les secondes.

L'afficheur doit permettre de savoir lorsque le décompte est actif. Il doit également indiquer à l'utilisateur lorsqu'il reste moins que 1 heure d'autonomie de batterie.

Une musique ainsi que des avertissements sonores (5min, 2min, 1min) sont stockés sur l'appareil (format MP3 ou autre) soit à l'aide de l'USB, soit sur une carte mémoire. La musique peut être sélectionnée comme sonnerie à la place d'un bip standard.

La musique ou le bip sera joué lorsque le décompte se termine. Il s'arrête lorsque l'on presse sur le bouton poussoir ou après 20 secondes. Le volume est réglé dans les paramètres.

Fonction du bouton rotatif :

- Rotation du bouton sans clic préalable : Réglage du temps de la minuterie.

Prendre en compte la vitesse de rotation pour un réglage rapide de la durée.

- Double clic : accès au réglage du volume de la sonnerie, choix entre la musique et un bip pour la fin du décompte, activation/ désactivation des avertissements (5 minutes, 2 minutes et 1 minute).

## 1.4 Caractéristiques techniques

### 1.4.1. Connectique et environnement extérieur

- Affichage à définir
- Encodeur rotatif avec bouton poussoir à définir
- Le système sera placé dans un boîtier avec l'écran et l'encodeur rotatif sur la partie frontale.
- Connecteur USB-C

### 1.4.2. Eléments imposés

- Recharge de la batterie par le connecteur USB-C
- Taille d'écran minimale : 7cm x 2cm
- Le PCB avec toute l'électronique doit être créé par l'apprenti. L'utilisation de module est proscrite sauf exception (à voir avec le supérieur et/ou les experts).

### 1.4.3. Fonctionnalités

- Interrupteur pour allumer l'appareil
- Réglage de la minuterie de minimum 10 secondes à 4 heures
- Réglage du volume.
- Activation / désactivation des avertissements sonore du temps restant.
- Start/ stop de la minuterie en fonction de la position du time timer.
- Témoin visuel pour savoir si le décompte est démarré ou stoppé.
- Affichage du temps restant sur l'écran au format (hh :mm puis mm : ss quand le temps est plus petit que 1 heure)
- Musique ou bip pour annoncer la fin de la minuterie
- L'appareil doit pouvoir fonctionner 8h sur batterie



## 2. Planification

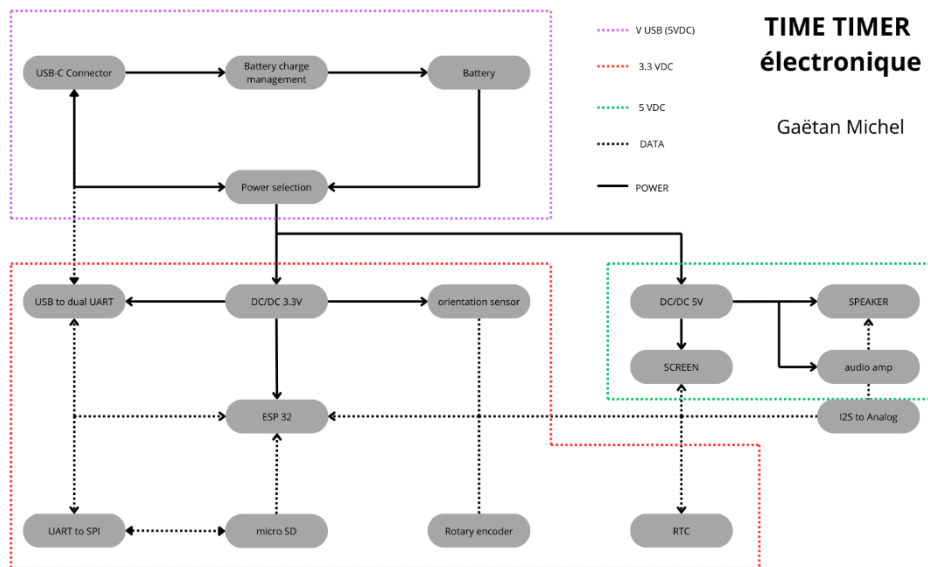
Pour avoir une idée claire et visuelle de la planification de mon travail, j'ai réalisé un planning regroupant toutes les tâches à effectuer en quatre-vingt heures de travail.

		Absents, Maladie, ...			Semaine 1			Semaine 2			Semaine 3			Semaine 4			Semaine 5			Semaine 6			Semaine 7	
Lundi			Férié	Date	15.04.2024	17.04.2024	18.04.2024	22.04.2024	24.04.2024	25.04.2024	29.04.2024	01.05.2024	02.05.2024	06.05.2024	08.05.2024	09.05.2024	13.05.2024	15.05.2024	16.05.2024	20.05.2024	22.05.2024	23.05.2024	27.05.2024	29.05.2024
Mercredi			planning prévu	Temps par jour	5h55	7h25	4h25	5h55	7h25	4h25	5h55	Férié	4h25	5h55	7h25	Férié	5h55	7h25	4h25	Férié	7h25	4h25	5h55	7h25
Jeudi			Planning réel	Temps écoulé	5h55	13h20	17h45	23h40	31h05	35h30	41h25	41h25	45h50	51h45	59h10	59h10	65h05	72h30	76h55	76h55	84h20	88h55	94h50	102h15
TPI	S'informer	Lecture du CDC	initiale																					
			Réel																					
	Planifier	Recherche de solutions / composants	initiale																					
			Réel																					
	Décider	création du Planning	initiale																					
			Réel																					
		Choix de composant	initiale																					
			Réel																					
	Réaliser	Définir forme/solution	initiale																					
			Réel																					
		Commande de composant	initiale																					
			Réel																					
		Schéma Principal	initiale																					
			Réel																					
		Montage sur plaque d'essai	initiale																					
			Réel																					
		Programmation	initiale																					
			Réel																					
		Réalisation de PCB	initiale																					
			Réel																					
	Évaluation	Montage de composant	initiale																					
			Réel																					
		Création / Impression du boîtier	initiale																					
			Réel																					
	Documentation	Assemblage des éléments	initiale																					
			Réel																					
		Mise en service	initiale																					
			Réel																					
Documentation	Protocole de test	initiale																						
		Réel																						
	Dépannage	initiale																						
		Réel																						
Documentation	Auto-évaluation	initiale																						
		Réel																						
	Rapport	initiale																						
		Réel																						
Documentation	Journaux de travail	initiale																						
		Réel																						
	Mode d'emploi	initiale																						
		Réel																						

## 3. Développement de la solution

### 3.1 Solution imaginée

Voici la solution que j'ai imaginé afin de répondre fidèlement au cahier des charges.



Ce système permet à l'utilisateur d'atteindre le contenu de la carte microSD sans la sortir de l'appareil. La carte SD ainsi que le microcontrôleur sont joignables avec l'USB-C.

Un circuit de sélection d'alimentation est intégré. Lorsque le câble USB est branché, l'appareil est alimenté via ce dernier. Dans le cas contraire, l'appareil fonctionne alors sur batterie.

## 4. Choix des composants

### 4.1 Microcontrôleur

Le choix du microcontrôleur est très important dans ce type de projet, plusieurs critères m'ont aidé à le définir comme la taille, le prix, le nombre de GPIO disponibles ou encore les interfaces disponibles.

J'ai besoin d'un microcontrôleur qui possède une interface SPI pour communiquer avec l'écran et la carte microSD, d'une interface I2C pour communiquer avec la RTC. Il faut aussi que je puisse le programmer avec l'USB C, il faut donc qu'il possède une interface UART.

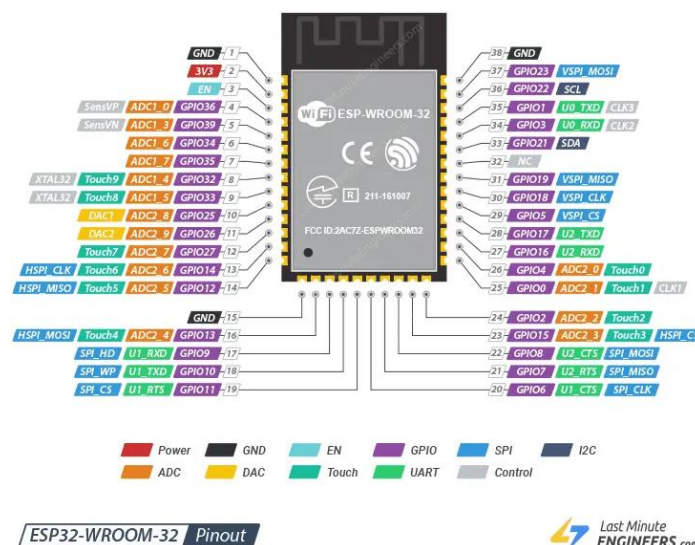
Au niveau des GPIO, j'ai besoin d'un minimum de 21 pins disponibles.

- 3 pour l'encodeur rotatif
- 1 pour le capteur d'orientation
- 1 pour la batterie
- 2 pour l'UART (RX et TX)
- 2 pour l'I2C (SDA et SCL)
- 4 pour la microSD (MISO, MOSI, CLK et CS)
- 4 pour l'écran (DIN, CLK, CS et DC)
- 3 pour l'I2S (BCLK, LRCLK, DATA)
- 1 pour l'activation ou non de l'alimentation du haut-parleur

Pour ce travail, j'ai le choix entre trois plateformes :

- Arduino
- ESP
- Raspberry Pi pico

Pour une question de taille et comme je ne trouve pas nécessaire d'utiliser une carte entière, j'ai opté pour l'ESP32-WROOM-32E-N16. Il se programme par UART, possède 15 GPIO utilisable sans contrainte, 4 en entrée uniquement, 2 pour la programmation UART et 5 qui doivent être HIGH ou LOW pendant le démarrage. Il possède également toutes les interfaces dont j'ai besoin.




## 4.2 Alimentation

### 4.2.1 Connecteur USB-C

Pour pouvoir alimenter et recharger mon circuit par USB-C, j'ai commandé une Breakout Board de chez Adafruit car le pinout des réceptacles USB-C seul est trop serré pour pouvoir être usiné à l'atelier.

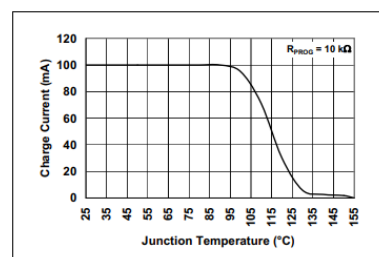
Cette carte possède déjà deux résistances de 5.1kΩ, elles indiquent au port amont de fournir une tension de 5V et un courant max de 1.5A.


Fournisseur	N° fournisseur	Libellé	Prix unitaire	
DigiKey	1528-2873-ND	ADAFRUIT USB C BREAKOUT BOARD	2.68 CHF	

### 4.2.2 Gestion de charge

Une bonne partie de du circuit est reprise du schéma de l'ESP32 Thing Plus, la gestion de charge en fait partie car il possède un chargeur de batterie LiPo 1s et c'est le type de batterie utilisé sur ce projet. Le MCP73833 sera donc le circuit employé.


J'ai utilisé le boîtier possédant un pad de dissipation thermique car sans celui-ci, le boîtier a tendance à vite chauffer lors de la charge. Cela réduit donc considérablement son efficacité.



Fournisseur	N° fournisseur	Libellé	Prix unitaire	
Mouser	579-MCP73833T-AMI/MF	Gestion de batterie/pile	1.06 CHF	

### 4.2.3 Batterie

La batterie utilisée est une LiPo 1s de 2700mAh

Fournisseur	N° fournisseur	Libellé	Prix unitaire	
Conrad	1214021 - UQ	Renata ICP606168PRT	45.95 CHF	

#### 4.2.4 Convertisseurs DC/DC


Pour l'alimentation j'ai besoin d'une alimentation 3.3V pour la majorité de mon circuit et d'une alimentation 5V pour l'audio.

Comme la tension d'alimentation principale varie entre 3V et 5V, en fonction de l'USB ou de la batterie, il faut un circuit capable de :

- Augmenter sa tension d'entrée
- Abaisser sa tension d'entrée
- Avoir une tension d'entrée variable.

Ce critère m'a donc orienté vers un circuit buck/boost. J'ai effectué quelques recherches et le LTC3440 est revenu plusieurs fois, et dans des applications semblables à la mienne.

J'ai d'abord utilisé ce circuit avec un autre boîtier (DFN-10) car j'en avais à disposition mais lorsque j'ai dû les remplacer, j'ai commandé un boîtier plus facile à braser afin d'éviter tout problème de connections et comme ils ne chauffent pas, le pad de dissipation de chaleur n'était pas nécessaire.


Fournisseur	N° fournisseur	Libellé	Prix unitaire	
Mouser	584-LTC3440EMS#PBF	Buck-Boost DC/DC Conv	7.19 CHF	

## 4.3 Interface UART et microSD

### 4.3.1 Interface UART

Pour pouvoir communiquer avec le microcontrôleur, l'USB doit être converti en Serial. Pour faire cela, le circuit utilisé sur l'ESP Thing Plus a également été repris, il s'agit du CP210x.


Comme je voulais intégrer l'option de pouvoir communiquer avec la carte SD via le câble USB-C, deux ports UART sont nécessaires. Pour parfaire cela, le CP2105 est utilisé.

Fournisseur	N° fournisseur	Libellé	Prix unitaire	
Mouser	634-CP2105-F01-GMR	USB to Dual UART Bridge	4.41 CHF	

### 4.3.2 Interface microSD


Pour la carte microSD, un simple réceptacle est utilisé.

Afin de pouvoir communiquer avec l'ordinateur, un convertisseur UART to SPI est indispensable.

Fournisseur	N° fournisseur	Libellé	Prix unitaire	
Mouser	771-SC16IS0IPW128	USB to Dual UART Bridge	3.18 CHF	


## 4.4 Capteur d'orientation

Pour détecter le sens dans lequel le minuteur est orienté, plusieurs solutions sont possibles. Comme seule l'information debout/retourné est utile, un accéléromètre ou encore un gyroscope serait démesuré. J'ai donc utilisé un interrupteur à bascule, une bille en déplacement libre dans le boîtier passe, ou pas, devant un phototransistor. Cette solution est également bien plus simple d'utilisation et simplifiera le software.

Fournisseur	N° fournisseur	Libellé	Prix unitaire	
Mouser	769-AHF22	Interrupteurs à bascule	5.75 CHF	


## 4.5 Encodeur rotatif

La commande du minuteur est effectuée avec un encodeur rotatif. J'ai utilisé le KY-040 de chez JOY-IT car je l'ai déjà utilisé et il m'en restait un en stock. Le circuit imprimé possède déjà des pull-up/down sur le CLK, le DT et le Switch.

Fournisseur	N° fournisseur	Libellé	Prix unitaire	
Conrad	1695709 - UQ	Encodeur rotatif	2.55 CHF	


## 4.6 Real Time Clock

Afin d'être précis pour la minuterie, une RTC est requise. Cela va me permettre également d'implémenter l'heure et la date sur le minuteur. La RTC utilisée ici est la MCP79410, je l'ai déjà utilisée et elle est disponible à l'atelier.

Fournisseur	N° fournisseur	Libellé	Prix unitaire	
Mouser	579-MCP79410-I/SN	Horloge en temps réel I2C	0.92 CHF	

## 4.7 Affichage

L'affichage est constitué d'un écran OLED de 3.12 pouces. J'ai choisi celui-ci car il correspond à la taille minimale demandée sans être pour autant trop haut et peut être contrôlé en SPI.

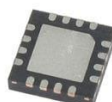
Fournisseur	N° fournisseur	Libellé	Prix unitaire	
Mouser	763-NHD31225664UCW2	3.12 Graphic OLED White	37.21 CHF	

## 4.8 Gestion audio

### 4.8.1 I2S converter


Pour pouvoir obtenir un signal analogique de bonne qualité, j'ai utilisé un circuit DAC utilisant l'interface I<sup>2</sup>S (Inter-IC Sound).

Ce circuit possède un amplificateur intégré mais il ne permet qu'une amplification maximum de 12dB. J'ai donc réalisé un amplificateur externe.

Fournisseur	N° fournisseur	Libellé	Prix unitaire	
Mouser	700-MAX98357AETE+T	Amplificateurs audio	2.69 CHF	

### 4.8.2 Amplificateur audio


L'amplificateur audio externe est réalisé avec le circuit LM386. J'ai déjà eu l'occasion de l'utiliser lors de la réalisation d'un amplificateur/égalisateur audio et il est disponible à l'atelier.

Fournisseur	N° fournisseur	Libellé	Prix unitaire	
Mouser	926-LM386N-3/NOPB	Amplificateurs audio	0.95 CHF	




### 4.8.3 Haut-parleur

Un haut-parleur de  $8\Omega/3W$  est utilisé, il possède un diamètre 28.5mm donc sa taille convient pour ce projet. J'ai utilisé celui-ci car je l'avais en stock.

Fournisseur	N° fournisseur	Libellé	Prix unitaire	
Distrelec	13020757	miniature-speaker-ohm	3.51 CHF	

### 4.9 Power switch

Pour l'alimentation du circuit, j'ai utilisé un bouton poussoir ON/OFF.

Fournisseur	N° fournisseur	Libellé	Prix unitaire	
Mouser	1684.1101	Interrupteur ON/OFF	5.24	



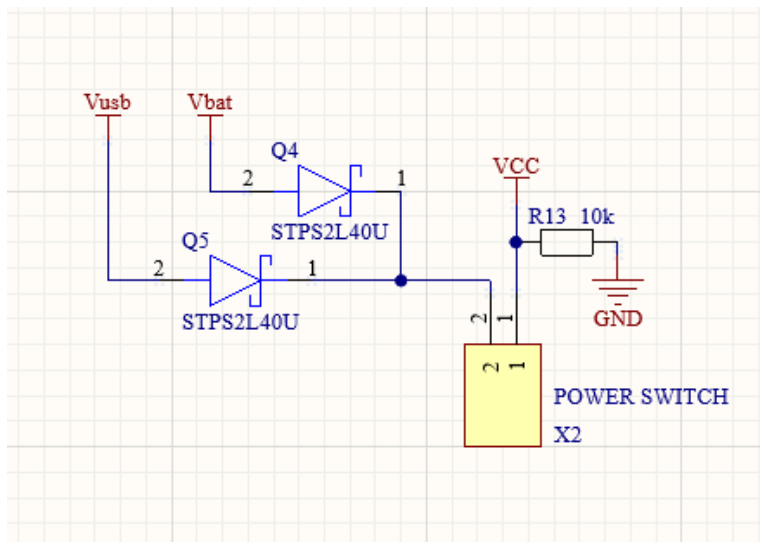
## 5. Développement circuits

### 5.1 Alimentation

#### 5.1.1 Circuit de sélection d'alimentation

Comme le système peut être alimenté de façons différentes, j'ai créé un petit circuit permettant de sélectionner la source. J'ai également ajouté un switch ON/OFF afin d'allumer ou éteindre l'appareil.

Schéma du circuit d'alimentation :



Ces deux diodes servent à sélectionner la source d'alimentation ; Lorsque l'USB est branché, la tension à la borne 2 du switch sera supérieure à Vbat, donc cette dernière ne se déchargera pas et la diode D4 bloque l'entrée d'un courant dans la batterie. Dans le cas où l'USB est déconnecté, la batterie alimentera le système.

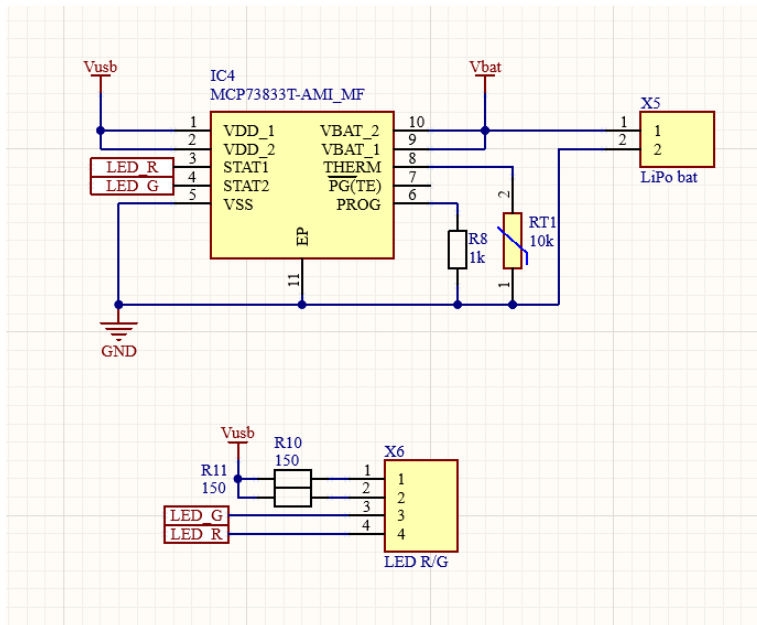
Des diodes Schottky sont utilisées pour avoir une plus légère chute de tension.

La résistance pull-down R13 permet de fixer le potentiel VCC à 0V lorsque le switch est en position ouvert.

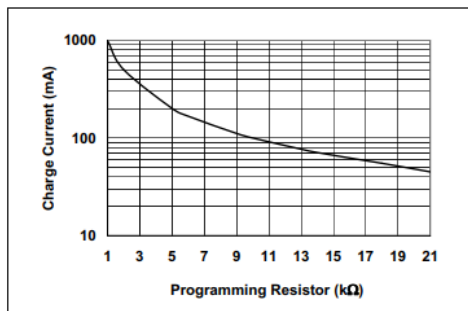
## 5.1.2 Circuit de charge de la batterie

La maintenance de la batterie s'effectue à l'aide du MCP73833.

Schéma du circuit de charge :



La résistance R8 sert à définir le courant de charge maximal que l'on souhaite et la thermistance RT1 permet le contrôle du courant de charge en fonction de la température de la batterie, ce qui évitera la surchauffe. Comme ma batterie possède une grande capacité, j'ai choisi une résistance qui me permet de la recharger au plus vite.



**FIGURE 2-4:** Charge Current ( $I_{OUT}$ ) vs. Programming Resistor ( $R_{PROG}$ ).

### 3.7 Thermistor Input (THERM)

An internal 50  $\mu$ A current source provides the bias for most common 10 k $\Omega$  negative-temperature coefficient thermistors (NTC). The MCP73833/4 compares the voltage at the THERM pin to factory set thresholds of 1.20V and 0.25V, typically.

J'ai également connecté une LED rouge/vert au pins STAT1 et STAT2 afin d'avoir un visuel sur le statut de la batterie.

Pour dimensionner les résistances à mettre en série avec la LED, je me suis référé à la datasheet pour obtenir la tension  $V_f$  de la LED. Cette dernière a une valeur de 2.1V.

$$R = \frac{V_{usb}-V_F}{I_F} = \frac{5-2.1}{0.02} = 145 = \underline{\underline{150\Omega}}$$

$$P = R * I^2 = 150 * 0.02^2 = 60mW$$



Electrical/Optical Characteristics					Luminous Intensity			Dominant Wavelength			Forward Voltage			Viewing Angle (°deg.)	Thermal Resistance (°C/W)	Reverse Current Intensity (μA)
Dialight P/N		Emitted Color	Material	Lens Color	If = 20 ma										Rth(j-s) If=20 ma	(Ir)VR=5V
20-Piece Tape	7" Reel, 4000 pcs				Min	Typ	Max	Min	Typ	Max	Min	Typ	Max			
598-8610-202F	598-8610-207F	● Red	AlInGaP		40	65	90	630	635	640	1.8	2.1	2.4	140	450	10
		● Green			30	40	50	565	570	575	1.8	2.1	2.4			
598-8621-202F	598-8621-207F	● Red-Orange	InGaN	Non-Diffused	90	120	150	620	625	630	1.8	2.1	2.4			
		● Green			260	580	900	520	525	530	3.0	3.2	3.4			
598-8640-202F	598-8640-207F	● Yellow	AlInGaP		80	155	230	585	590	595	1.8	2.1	2.4			
		● Green			30	40	50	565	570	575	1.8	2.1	2.4			
598-8660-202F	598-8660-207F	● Yellow			90	145	200	585	590	595	1.8	2.1	2.4			
		● Red			40	65	90	630	635	640	1.8	2.1	2.4			

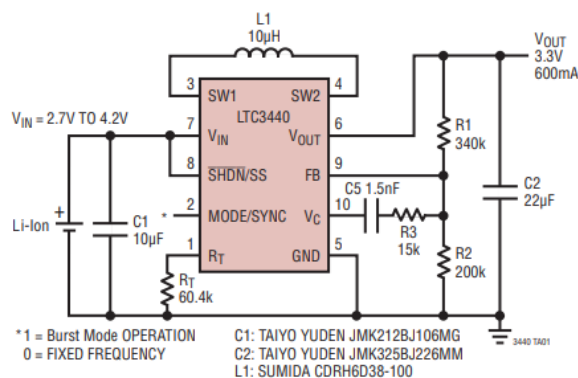
### 5.1.3 Convertisseurs DC/DC

Le reste du circuit fonctionne avec une tension d'alimentation de 3.3V ou 5V. il faut donc dimensionner deux circuits pour le LTC3440 afin d'obtenir deux sources différentes.

Sur la datasheet, on trouve directement un circuit générant une tension de 3.3V pour 600mA, ce qui convient parfaitement à notre utilisation.

## TYPICAL APPLICATION

### Li-Ion to 3.3V at 600mA Buck-Boost Converter



Les valeurs de R1 et R2 présentes sur ce schéma ne sont pas dans la série E12, j'ai alors redimensionné ces résistances pour 3.3V et 5V.

$$V_{OUT} = 1.22V \cdot \left(1 + \frac{R1}{R2}\right)$$

$R_T$  a été arrondi à  $56k\Omega$ .

**Calculs pour dimensionner les résistances :**Pour  $U_{out} = 3.3V$ 

$$\frac{R1}{R2} = \frac{3.3}{1.22} - 1 = 1.705$$

Pour  $U_{out} = 5V$ 

$$\frac{R1}{R2} = \frac{5}{1.22} - 1 = 3.098$$

Pour trouver des résistances dont le ratio correspond à ces valeurs, j'ai utilisé le site <https://jansson.us/resistors.html>.

Select which type of ratio to solve for:

☒ Resistor ratio  R1/R2

☐ Voltage divider  $V_H$ :   $V_L$ :   $V_H > V_L$

Optional: ☐ Inverse [What is this?](#)

Calculate...

Single:  ÷  =  0.47 %

Series:  ÷ ( + ) =  0.02 %

Parallel:  ÷ ( // ) =  0.01 %

[How does it work?](#)

Select which type of ratio to solve for:

☒ Resistor ratio  R1/R2

☐ Voltage divider  $V_H$ :   $V_L$ :   $V_H > V_L$

Optional: ☐ Inverse [What is this?](#)

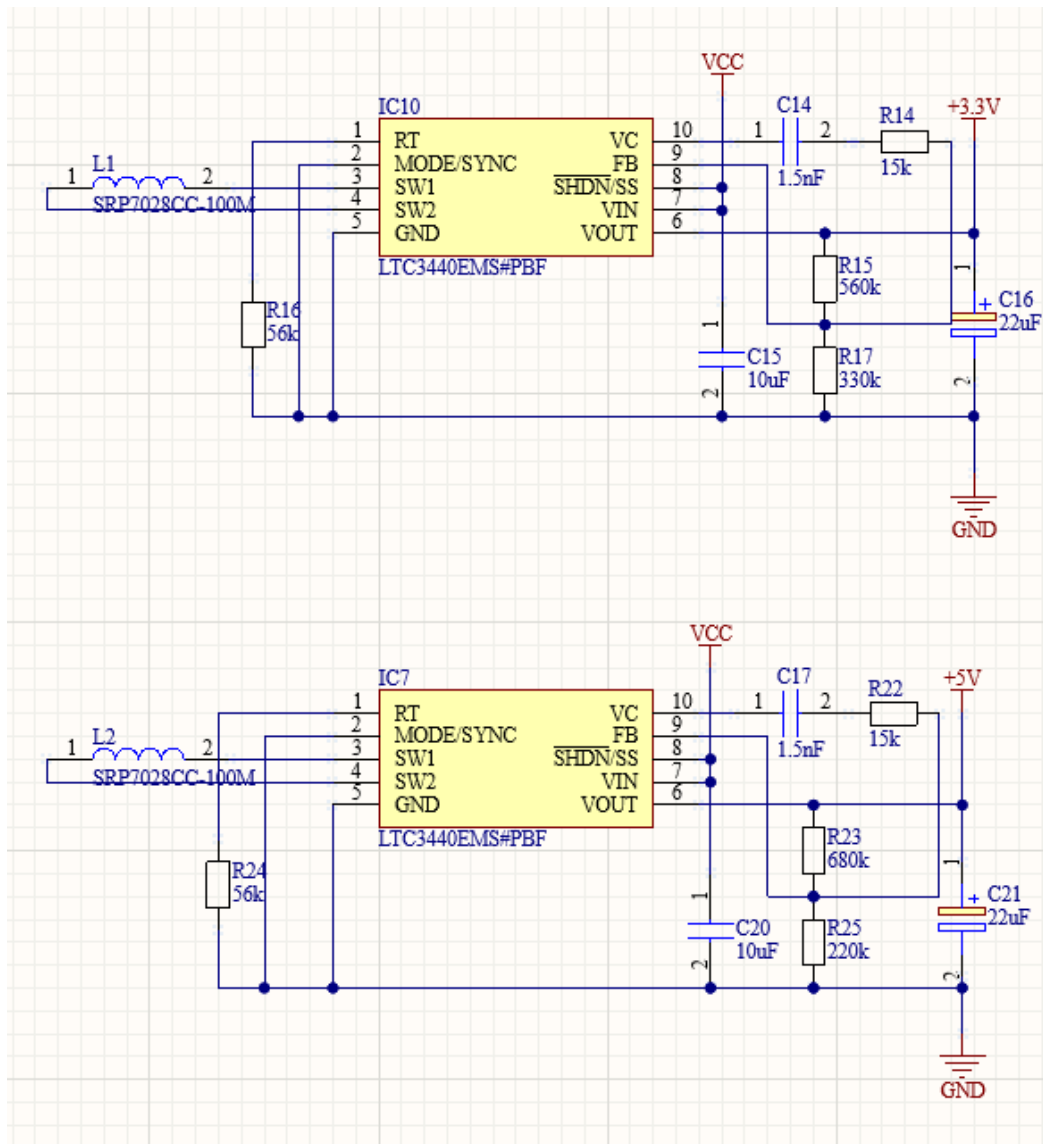
Calculate...

Single:  ÷  =  0.23 %

Series:  ÷ ( + ) =  0.02 %

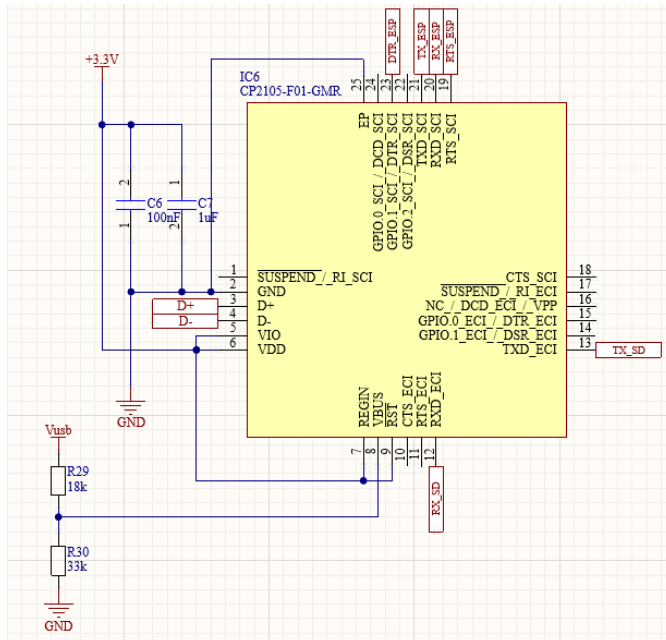
Parallel:  ÷ ( // ) =  0.01 %

La valeur de cette résistance a été multipliée par dix pour réduire la fuite de courant.

**Schéma final des convertisseur DC/DC :**

## 5.2 Interface UART

Pour dimensionner ce circuit, j'ai repris les mêmes composants que sur la datasheet, en redimensionnant ceux qui n'étaient pas compris dans la série E12.



$$UR_{30} = \frac{5 * 33k}{33k + 18k} = 3.24V$$

## 5.3 Interface SPI

Pour ce circuit également, j'ai simplement repris les exemples de la datasheet.

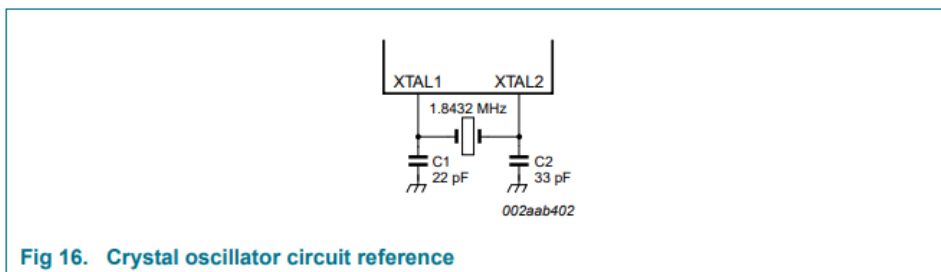
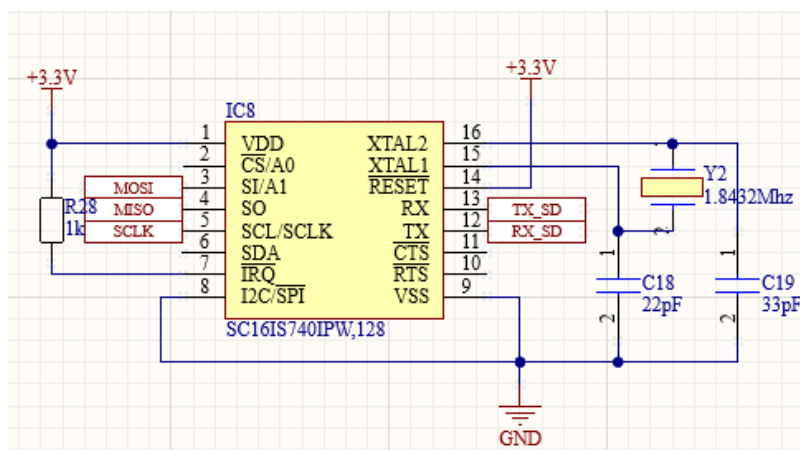


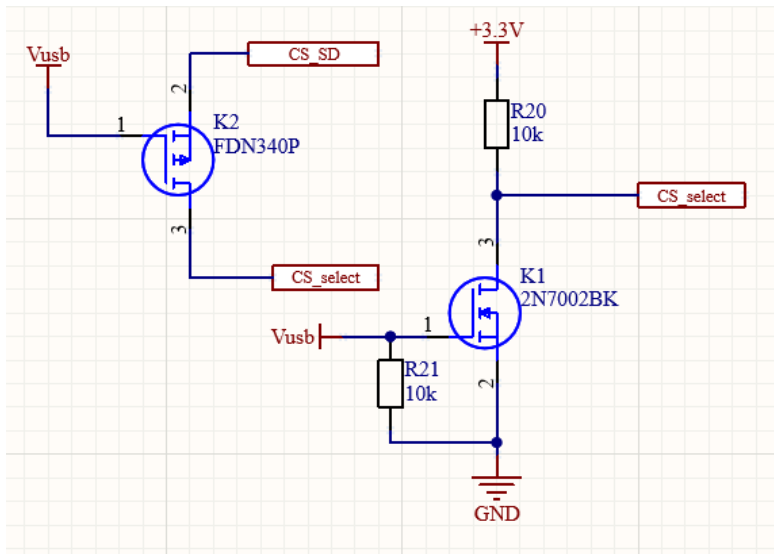
Fig 16. Crystal oscillator circuit reference

Schéma interface SPI :



## 5.4 Sélection du Chip Select de la carte micro SD

Afin de pouvoir accéder aux fichiers de la carte SD avec deux maîtres, j'ai créé un circuit permettant de sélectionner quel maître aura effet sur le CS de la carte.



Lorsque l'USB est connecté, le CS\_select sera mis à 0 et CS\_SD, n'aura aucun effet, ce qui permet le transfert de données.

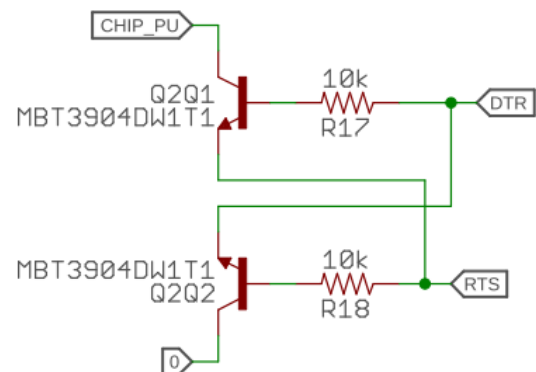
Lorsque l'USB est déconnecté, CS\_select sera contrôlé par CS SD.

## 5.5 Circuit d'auto-reset de l'ESP

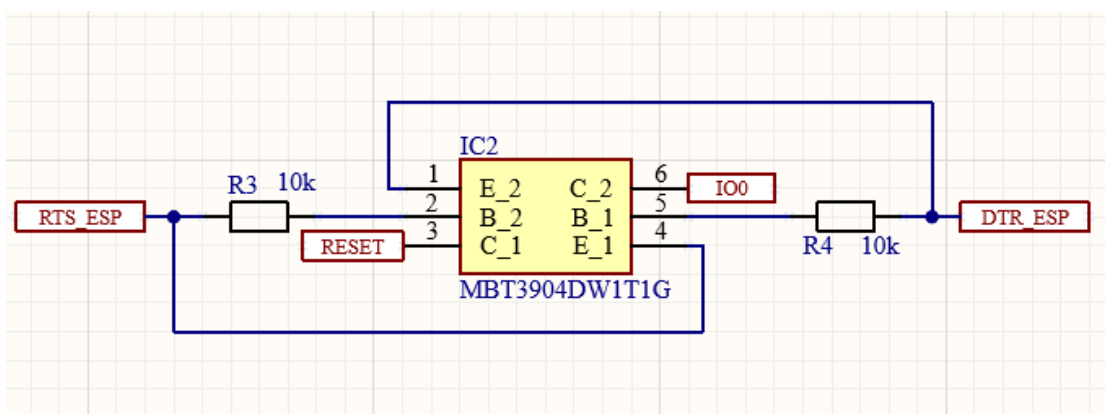
J'ai repris le circuit trouvé dans le schéma de l'ESP-32-thingPlus pour l'auto-reset.

Boot Mode Configuration			
Pin	Default	Boot	Download
GPIO0	1	1	0
U0TXD	1	1	x
GPIO2	0	x	0
GPIO4	0	x	x
MTDO	1	x	x
GPIO5	1	1	x

If U0TXD, GPIO2, GPIO5 are floating, GPIO0 determines boot mode



### Schéma final de l'auto-reset :



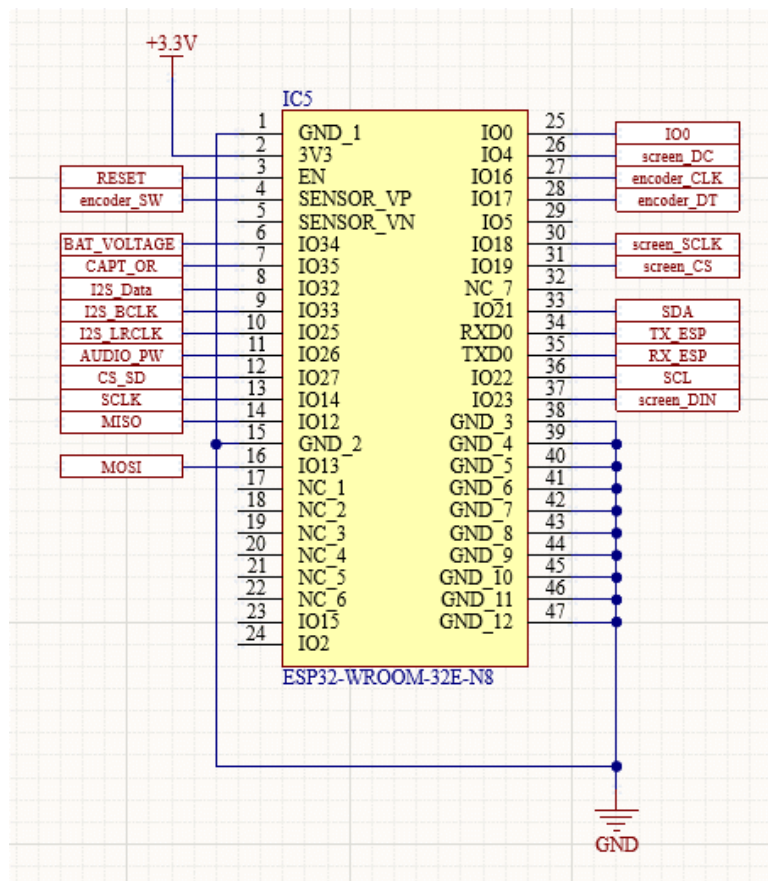
## 5.6 Connections de l'ESP32

Pour assigner les pins de l'ESP32, je me suis renseigné sur les fonctionnalités de chaque pin et leurs contraintes. Le site <https://lastminuteengineers.com/esp32-wroom-32-pinout-reference/> explique parfaitement tout cela.

Pin	Pin Label	GPIO	Safe to use?	Reason
4	SENSOR_VP	GPIO36	⚠	Input only GPIO, cannot be configured as output
5	SENSOR_VN	GPIO39	⚠	Input only GPIO, cannot be configured as output
6	IO34	GPIO34	⚠	Input only GPIO, cannot be configured as output
7	IO35	GPIO35	⚠	Input only GPIO, cannot be configured as output
8	IO32	GPIO32	✅	
9	IO33	GPIO33	✅	
10	IO25	GPIO25	✅	
11	IO26	GPIO26	✅	
12	IO27	GPIO27	✅	
13	IO14	GPIO14	✅	
14	IO12	GPIO12	⚠	must be LOW during boot
16	IO13	GPIO13	✅	
17	SHD/SD2	GPIO9	❌	Connected to Flash memory
18	SWP/SD3	GPIO10	❌	Connected to Flash memory

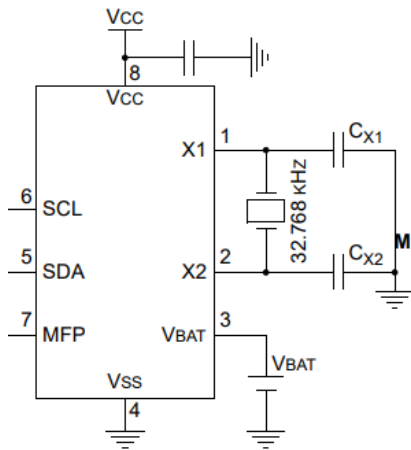
19	SCS/CMD	GPIO11	❌	Connected to Flash memory
20	SCK/CLK	GPIO6	❌	Connected to Flash memory
21	SDO/SD0	GPIO7	❌	Connected to Flash memory
22	SDI/SD1	GPIO8	❌	Connected to Flash memory
23	IO15	GPIO15	⚠	must be HIGH during boot, prevents startup log if pulled LOW
24	IO2	GPIO2	⚠	must be LOW during boot and also connected to the on-board LED
25	IO0	GPIO0	⚠	must be HIGH during boot and LOW for programming
26	IO4	GPIO4	✅	
27	IO16	GPIO16	✅	
28	IO17	GPIO17	✅	
29	IO5	GPIO5	⚠	must be HIGH during boot
30	IO18	GPIO18	✅	
31	IO19	GPIO19	✅	
33	IO21	GPIO21	✅	
34	RXD0	GPIO3	❌	Rx pin, used for flashing and debugging
35	TXD0	GPIO1	❌	Tx pin, used for flashing and debugging
36	IO22	GPIO22	✅	
37	IO23	GPIO23	✅	

**Assigination finale :**



## 5.7 Circuit de la RTC

Le MCP79410 est une RTC utilisant l'interface I2C pour communiquer avec le microcontrôleur. Le circuit utilisé est repris de la datasheet. La formule ci-dessous permet le dimensionnement des condensateurs.



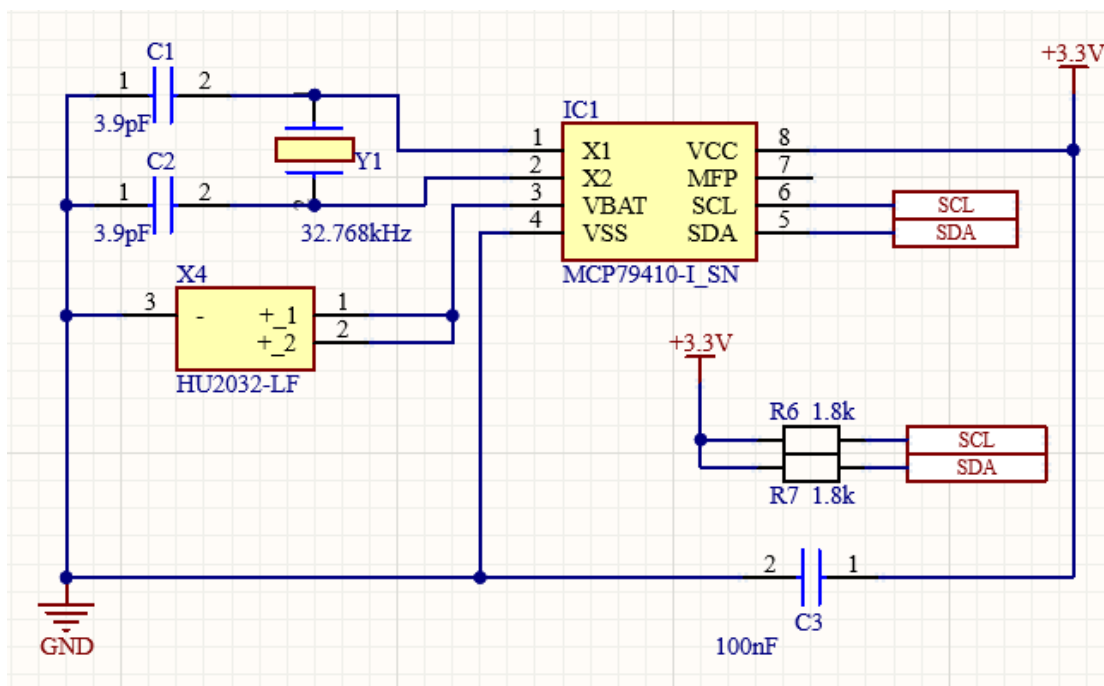
$$C_L = \frac{C_{X1} \times C_{X2}}{C_{X1} + C_{X2}} + C_{STRAY}$$

Sur mon projet, les condensateurs utilisés ont une valeur erronée. J'ai repris les valeurs que j'avais utilisées sur un autre projet sans les vérifier avec cette formule. Cette erreur peut potentiellement créer un très léger décalage sur le temps.

Pour calculer la valeur des condensateurs,  $C_{STRAY}$  sera négligée car elle est inconnue et infime. La capacité du quartz utilisé est de 7pF, la valeur des condensateurs devrait donc être de  $2 * C_L = 2 * 7 = 14pF \rightarrow \underline{15pF}$ .

Comme dit précédemment, la communication s'effectue en I2C, il ne faut donc pas oublier d'ajouter deux résistances pull-up. Pour assurer le bon fonctionnement de la communication dans n'importe quelles conditions, ces résistances ne doivent pas être trop élevées.

**Schéma final de la RTC :**





## 5.8 Circuit interface I2S

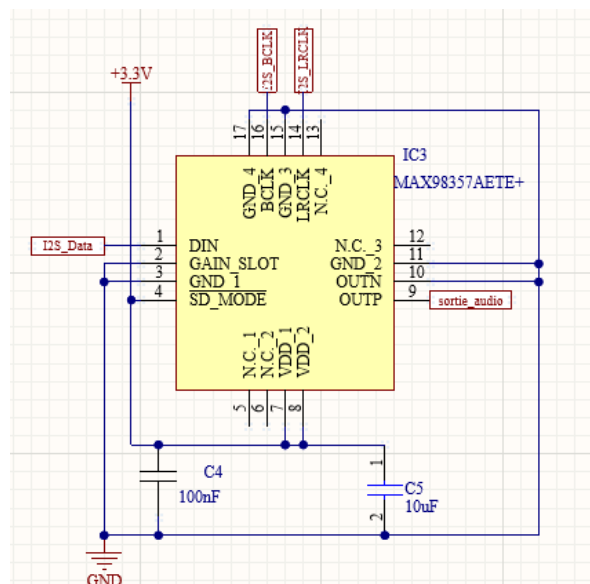
La transformation du signal I2S en un signal analogique se fait à l'aide du MAX98357. Le schéma utilisé est tiré de la datasheet.

Ce circuit possède un amplificateur interne dont le gain est réglable avec le pin GAIN\_SLOT. Pour faciliter le circuit, j'ai mis ce pin à 0V, ce qui donne un gain de 12dB.

Table 8. Gain Selection

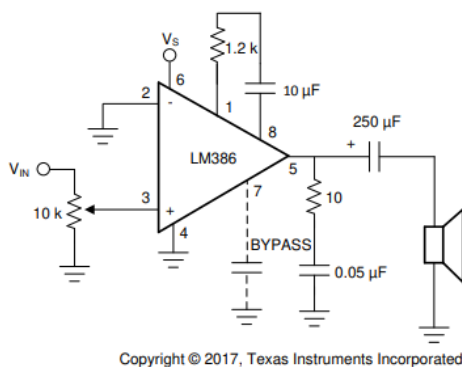
GAIN_SLOT	I <sup>2</sup> S/LJ GAIN (dB)
Connect to GND through 100kΩ ±5% resistor	15
Connect to GND	12
Unconnected	9
Connect to V <sub>DD</sub>	6
Connect to V <sub>DD</sub> through 100kΩ ±5% resistor	3

Schéma final de l'interface I2S :



## 5.9 Circuit de l'amplificateur audio

Le circuit de l'amplificateur est également tiré de la datasheet. Il possède une amplification de 50dB et un potentiomètre sur l'entrée afin de régler le volume.



Copyright © 2017, Texas Instruments Incorporated

Figure 9-5. LM386 with Gain = 50

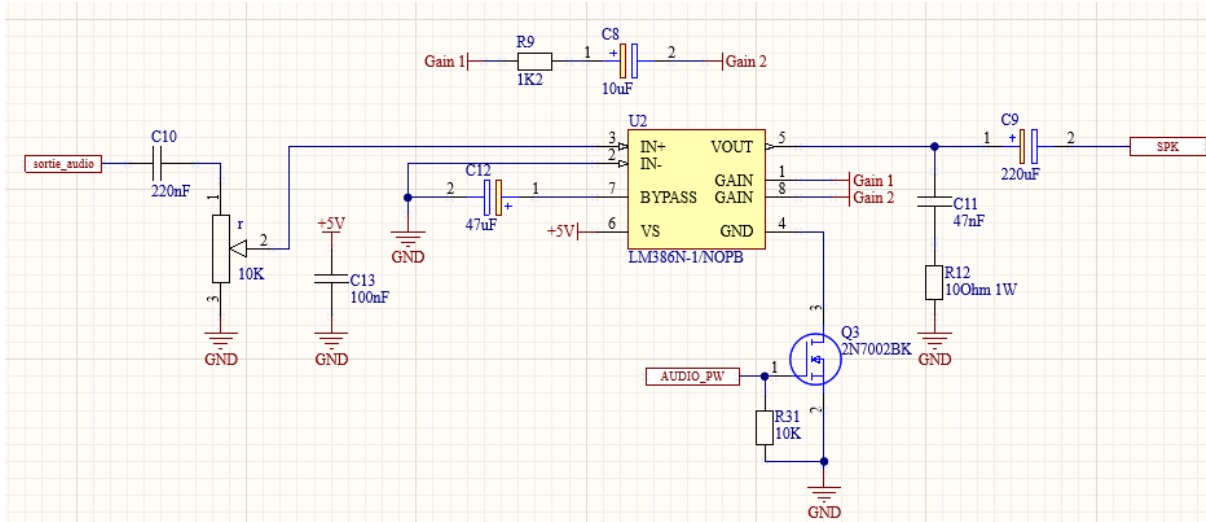
J'ai ajouté à ce schéma un MOSFET N au GND afin de pouvoir contrôler son alimentation. Ce système me permet d'éviter le bruit de fond constant du haut-parleur.

Calcul de la puissance pour la résistance de 10Ω :

$$P = \frac{U^2}{R} = \frac{5^2}{10} = 2.5W$$

Comme le signal de sortie n'est pas à 5V constant, une résistance de 1W suffit.

### Schéma final de l'amplificateur audio :



## 5.10 Circuit du capteur d'orientation

Le composant utilisé est un simple phototransistor, il faut donc dimensionner la résistance à mettre en série avec la LED et celle pour limiter le courant de collecteur.

### ■ Absolute maximum rating

Ta = 25°C

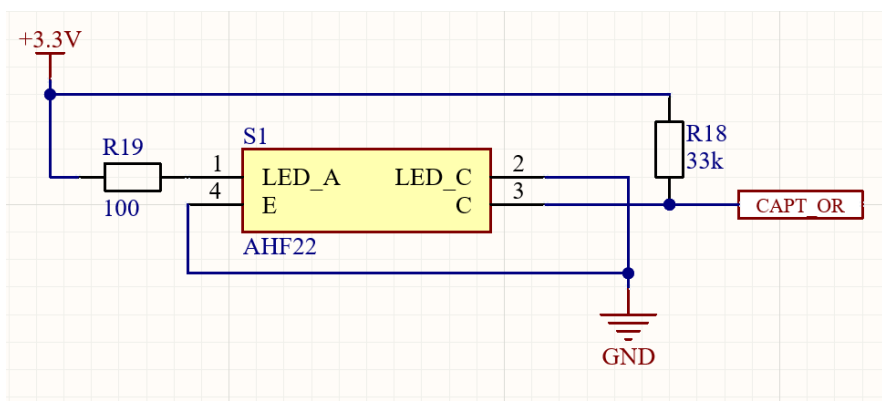
Item	Symbol	Rating	Unit
Input side Infrared light emitting diode (between terminal No.1 and No.2)	Forward Current	50	mA
	Reverse Voltage	5	V
	Power Dissipation	75	mW
	Collector-Emitter Voltage	30	V
Output side Phototransistor (between terminal No.3 and No.4)	Emitter-Collector Voltage	3	V
	Collector Current	20	mA
	Collector Power Dissipation	50	mW

D'après la datasheet, la tension  $V_F$  de la LED est égale à 1.2V. j'ai dimensionné le circuit pour un courant  $I_F$  de 20mA et un courant de collecteur de 0.1mA.

$$R_{LED} = \frac{V_{CC} - V_F}{I_{LED}} = \frac{3.3 - 1.2}{0.02} = 105 = \underline{100\Omega}$$

$$R_C = \frac{V_{CC} - V_{CE_{SAT}}}{I_C} = \frac{3.3 - 0}{0.0001} = 33'000 = \underline{33k\Omega}$$

### Schéma final du capteur d'orientation :



## 5.11 Circuit de l'écran OLED

Pour alimenter l'écran, je voulais d'abord alimenter le panneau OLED en 5V et la logique en 3.3V ; des jumpers sur le PCB de l'afficheur permettent une telle utilisation. Cela m'aurait permis de considérablement réduire la consommation de mon circuit.

**Jumper Option #1 - Independent Supply Voltage for Boost Converter (BC\_VDD)**

R14	R15	R18	R1	Description
Open	Close	Open	Open	Boost converter + OLED panel are powered from BC_VDD (pin #3). OLED Logic Circuit is powered from VDD (pin #2). This allows for increased efficiency through the boost converter, by allowing a supply voltage up to +12V at its input, BC_VDD (pin #3).

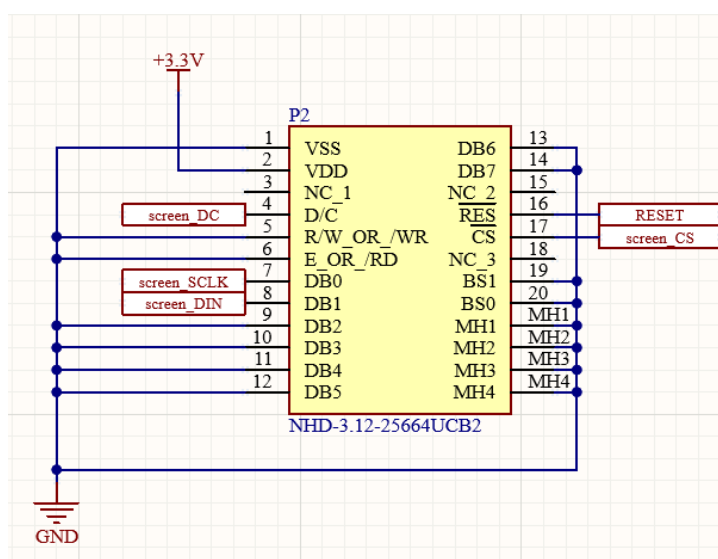
**Default Jumper Setting**

Default Jumper Setting						
Supply Voltage for Module	VDD	-	2.8	3.3	3.5	V
Supply Current for Module	IDD	VDD=3.3V, 100% ON	-	310	340	mA
Jumper Option #1						
Supply Voltage for Module	VDD	-	2.8	3.3	3.5	V
Supply Current for Module	IDD	VDD=3.3V	-	170	200	μA
Supply Voltage for Boost Converter	BC_VDD	-	2.8	-	12	V
Supply Current for Boost Converter	BC_IDD	BC_VDD=5.0V, 100% ON	-	150	170	mA
		BC_VDD=12.0V, 100% ON	-	55	70	mA
Jumper Option #2						
Supply Voltage for Module	VDD	-	2.8	3.3	3.5	V
Supply Current for Module	IDD	VDD=3.3V	-	170	200	μA
Supply Voltage for OLED Panel	VCC	-	11.5	12	12.5	V
Supply Current for OLED Panel	ICC	VCC=12V, 100% ON	-	45	55	mA
Jumper Option #3						
Supply Voltage for Logic	G_VDD	-	2.4	2.5	2.6	V
Supply Current for Module	G_IDD	VDD=3.3V	-	100	120	μA

On peut voir sur la datasheet que l'écran aurait passé d'une consommation de 310mA (100% ON) à 150mA (100% ON).

Malheureusement, la version de l'écran vendue chez Mouser ne correspond pas à celui sur l'image et la datasheet du site. J'ai donc reçu une ancienne version de l'écran, ne pouvant fonctionner que de la façon « Default Jumper Setting ».

**Schéma final de l'écran OLED :**



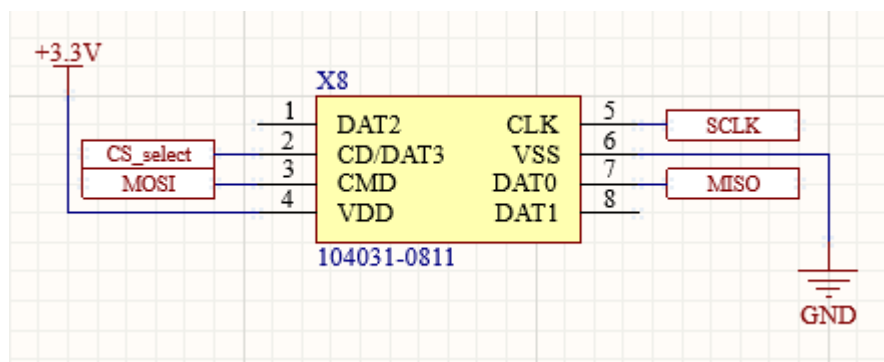


Les bus SPI utilisés ne sont pas les mêmes que ceux utilisés pour la carte SD car, lors des tests effectués sur l'écran, je n'avais que trouvé le constructeur de la librairie utilisé pour initialiser le SPI de l'écran avec un « Software SPI ». Les tests effectués ont montré que l'utilisation des deux périphériques ne pouvaient se faire simultanément.

J'ai par la suite trouvé le constructeur permettant l'utilisation d'un SPI Hardware, mais comme le routage de mon PCB était déjà fait, je n'ai pas voulu perdre du temps à refaire ce dernier. Le bus utilisé est donc le deuxième hardware SPI disponible sur l'ESP (CLK : GPIO18, MISO : GPIO19, MOSI : GPIO23).

## 5.12 Carte SD

Pour communiquer avec la carte SD, je connecte directement les pattes de la carte au SPI de mon ESP. Le circuit de Chip Select a déjà été décrit auparavant. (CLK : GPIO14, MISO : GPIO12, MOSI : GPIO13)



## 5.13 Circuit de mesure de la batterie

Pour que la batterie ne se décharge pas lorsque l'appareil est éteint, il faut créer un circuit qui ne va alimenter le diviseur de tension permettant de calculer la tension restante dans la batterie que lorsque l'appareil est allumé.

### Dimensionnement du diviseur de tension :

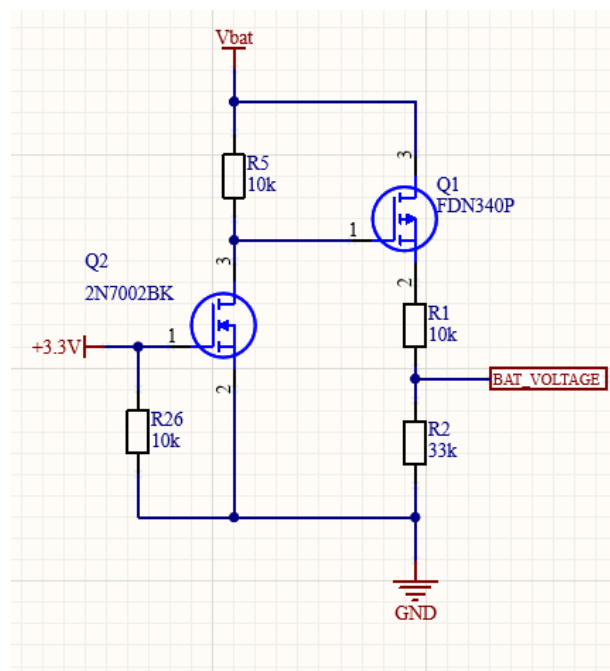
$$V_{BAT\_MAX} = 4.3V$$

Plage de mesure : 0V – 3.3V

☒ Voltage divider  $V_H$ : 4.3  $V_L$ : 3.3  $V_H > V_L$   
 Optional: ☐ Inverse What is this?  
 Search for equivalent resistor ratio: 0.303030303030303  
 Calculate...  
 Single: 10 K $\Omega$   $\div$  33 K $\Omega$  = 0.303030303 0.00 %  
 Series: 10 K $\Omega$   $\div$  (15 K $\Omega$  + 18 K $\Omega$ ) = 0.303030303 0.00 %  
 Parallel: 18 K $\Omega$   $\div$  (68 K $\Omega$  // 470 K $\Omega$ ) = 0.303003754 0.01 %  
How does it work?

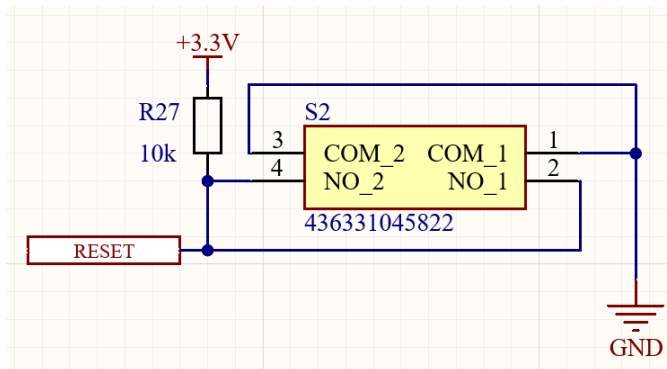
$$R_1 = 10k\Omega$$

$$R_2 = 33k\Omega$$



## 5.14 Reset du circuit

Un bouton de reset a également été prévu. Ce dernier est connecté à l'ESP et à l'écran OLED.



## 5.15 Batterie et autonomie

Pour dimensionner la batterie, j'ai d'abord estimé la consommation moyenne de mon circuit.

- ESP32-WROOM : comme je n'utilise ni le WI-FI ni le Bluetooth, ~30mA
- Amplificateur audio : ~5mA
- Ecran OLED : max. 310mA
- Autres : max 5mA

Comme je ne savais pas encore la consommation de l'écran en moyenne, car elle est définie par le nombre de pixels allumés, j'ai pris la consommation maximum afin d'avoir de la marge.

L'autonomie minimum de l'appareil était fixée à 8 heures.

**Calcul de la capacité de batterie :**

$$C = I * T = 350 * 8 = 2800mAh$$

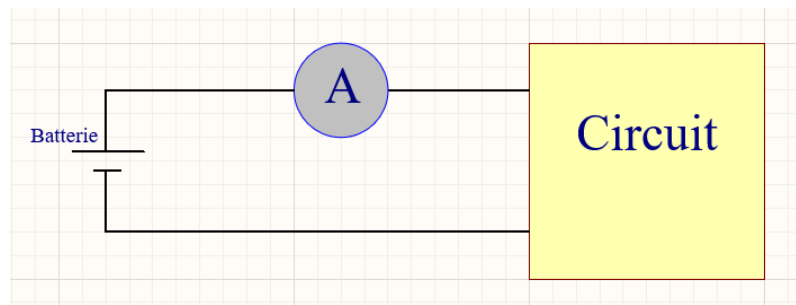
La batterie trouvée proche de cette valeur est de 2700mAh (même si cette dernière est notée à 2800mAh sur le site). Cette légère différence n'est pas dramatique car, comme expliqué ci-dessus, l'écran consommera moins de 310mA.

**Autonomie réelle :**

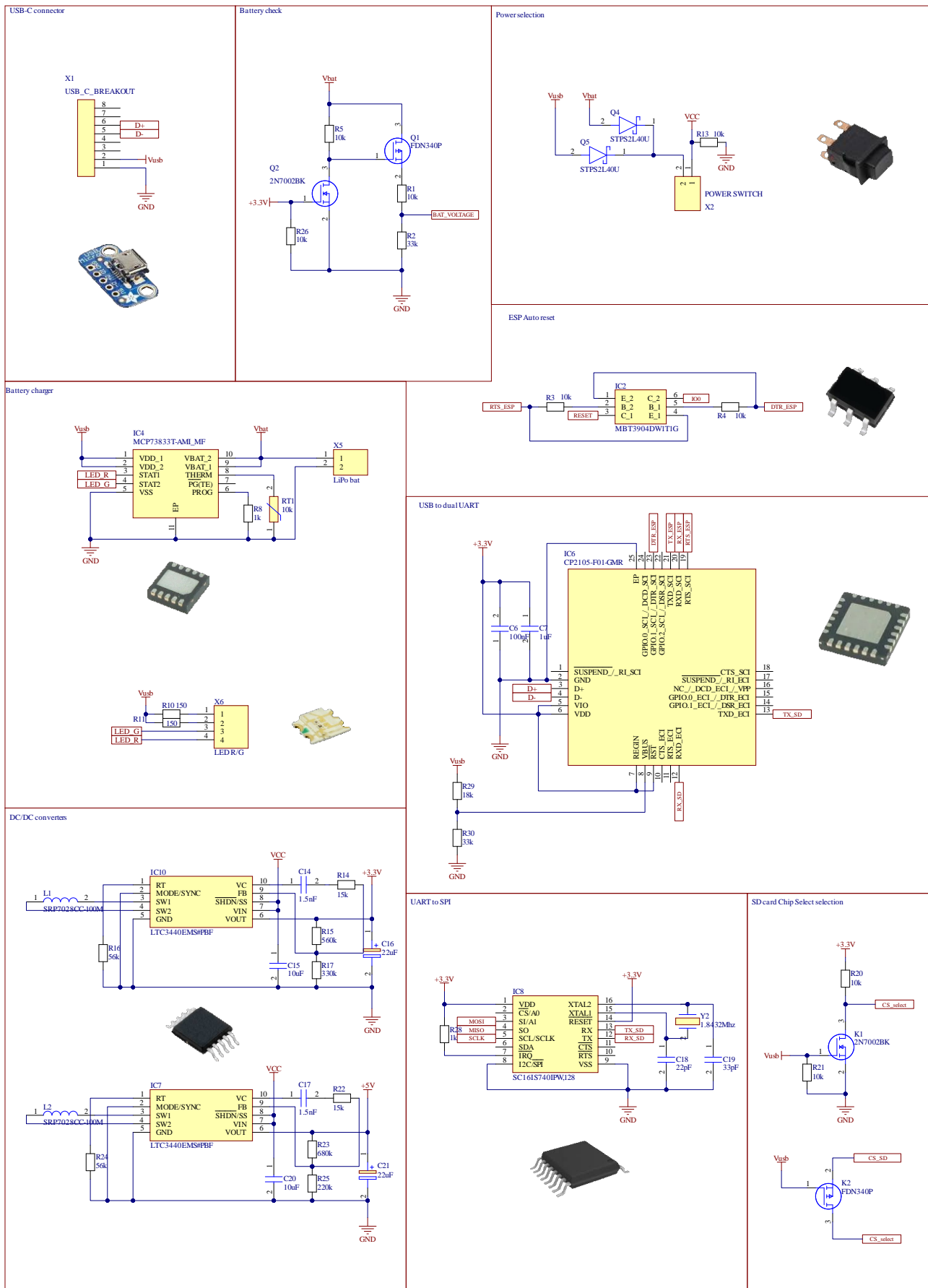
Une fois le circuit terminé, j'ai mesuré sa consommation de courant moyenne.

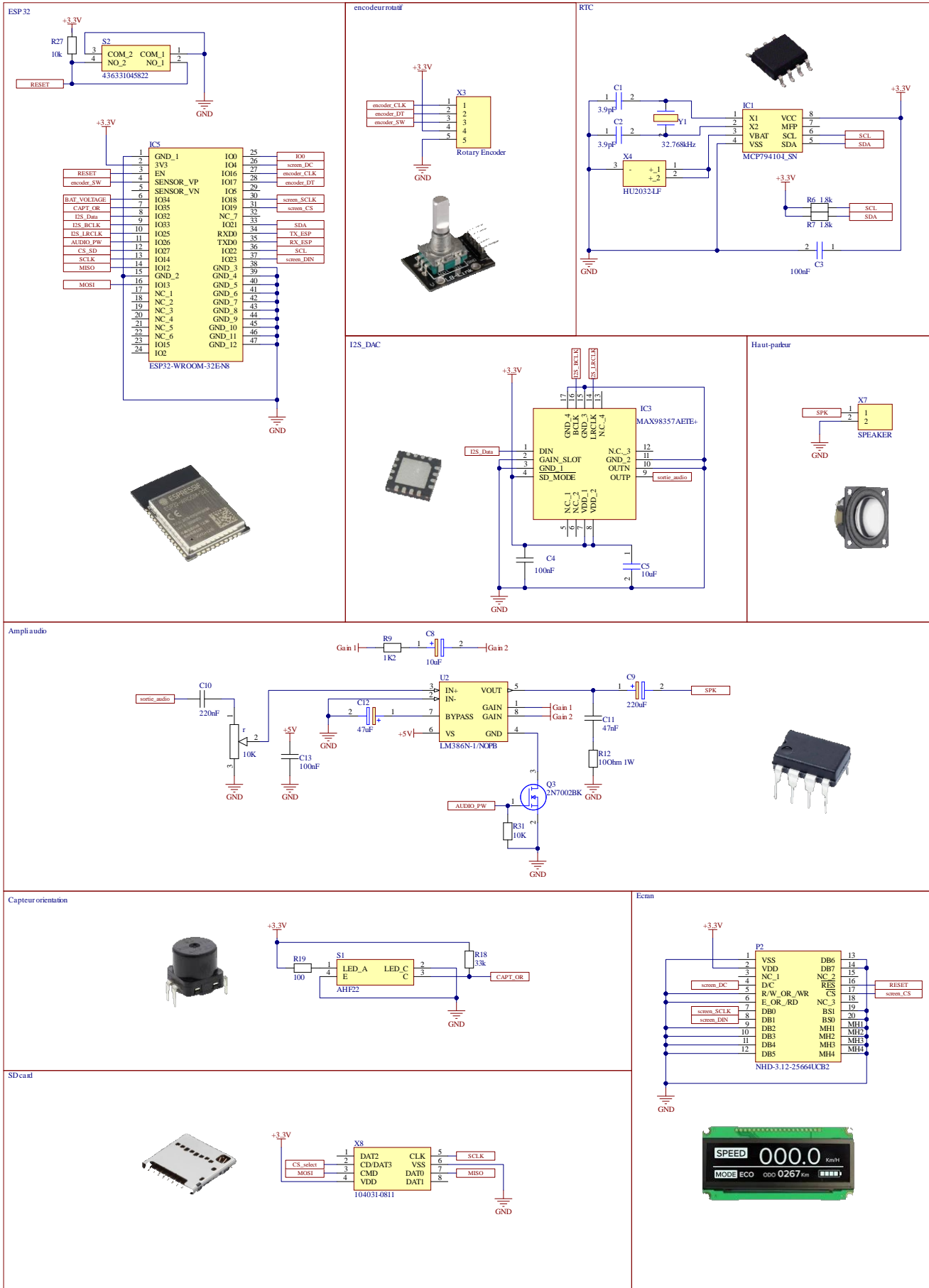
Consommation moyenne = 175mA

Autonomie = 2700/175 = ~15H



## 6. Schéma complet







## 7 Liste de composants

Référence schéma	Fournisseur	N° fournisseur	Libellé	Quantité	Prix unitaire	Total
C1, C2	Atelier ELT	-	Condensateur 3,6pF 0805	2	0,20 CHF	0,40 CHF
C3, C4, C6, C13	Atelier ELT	-	Condensateur 100nF 1206	4	0,20 CHF	0,80 CHF
C5, C15, C20	Atelier ELT	-	Condensateur 10uF 1206	3	0,20 CHF	0,60 CHF
C7	Atelier ELT	-	Condensateur 1uF 1206	1	0,20 CHF	0,20 CHF
C8	Mouser	710-865230542002	Condensateur 10uF polarisé	1	0,19 CHF	0,19 CHF
C9	Mouser	80-A785MN227M1VLAV23	Condensateur 220uF polarisé	1	1,38 CHF	1,38 CHF
C10	Atelier ELT	-	Condensateur 220nF 1206	1	0,20 CHF	0,20 CHF
C11	Atelier ELT	-	Condensateur 47nF 1206	1	0,20 CHF	0,20 CHF
C12	Mouser	667-EEH-ZA1K470V	Condensateur 47uF polarisé	1	2,07 CHF	2,07 CHF
C14, C17	Atelier ELT	-	Condensateur 1,5nF 1206	2	0,20 CHF	0,40 CHF
C16, C21	Mouser	647-PCM1J220MCL1GS	Condensateur 22uF polarisé	2	1,80 CHF	3,60 CHF
C18	Mouser	581-1206GA330JAT1A	Condensateur 33pF 1206	1	0,65 CHF	0,65 CHF
C19	Mouser	80-C1206C220JGGAUTO	Condensateur 22pF 1206	1	0,91 CHF	0,91 CHF
IC1	Mouser	579-MCP79410-I/SN	RTC MCP79410-I/SN	1	0,91 CHF	0,91 CHF
IC2	Mouser	863-MBT3904DW1T1G	Transistors bipolaires - BJT 200mA 60V Dual NPN	1	0,18 CHF	0,18 CHF
IC3	Mouser	700-MAX98357AETE+T	MAX98357	1	2,62 CHF	2,62 CHF
IC4	Mouser	579-MCP73833T-AMI/MF	Charge Management Controllers	1	1,03 CHF	1,03 CHF
IC5	Mouser	356-ESP32WRM32E128PH	ESP32-WROOM-32E-N16	1	2,83 CHF	2,83 CHF
IC6	Mouser	634-CP2105-F01-GMR	CP2105-F01-GMR	1	4,37 CHF	4,37 CHF
IC7, IC10	Mouser	584-LTC3444EMS#PBF	LTC3444EMS#PBF	2	7,19 CHF	14,38 CHF
IC8	Mouser	771-SC16IS01PW128	SC16IS740IPW,128	1	3,19 CHF	3,19 CHF
K1, Q2, Q3	Mouser	771-2N7002BK215	MOSFET N	3	0,24 CHF	0,71 CHF
K2, Q1	Mouser	512-FDN340P	MOSFET P	2	0,40 CHF	0,80 CHF
L1, L2	Mouser	710-744772100	Bobines radiale 10uH	2	0,65 CHF	1,31 CHF
P2	Mouser	763-NHD31225664UCW2	Ecran oled NHD-3.12-25664UCB2	1	37,21 CHF	37,21 CHF
R31	Distrelec	302-86-069	Potentiomètre 10kΩ SMD	1	1,85 CHF	1,85 CHF
Q4, Q5	Mouser	511-STPS2L40U	Diode Schottky 2.0A / 40V	2	0,35 CHF	0,71 CHF
R1, R3, R4, R5, R13, R20, R21, R26, R27, R31	Atelier ELT	-	Resistance 10kΩ	9	0,15 CHF	1,35 CHF



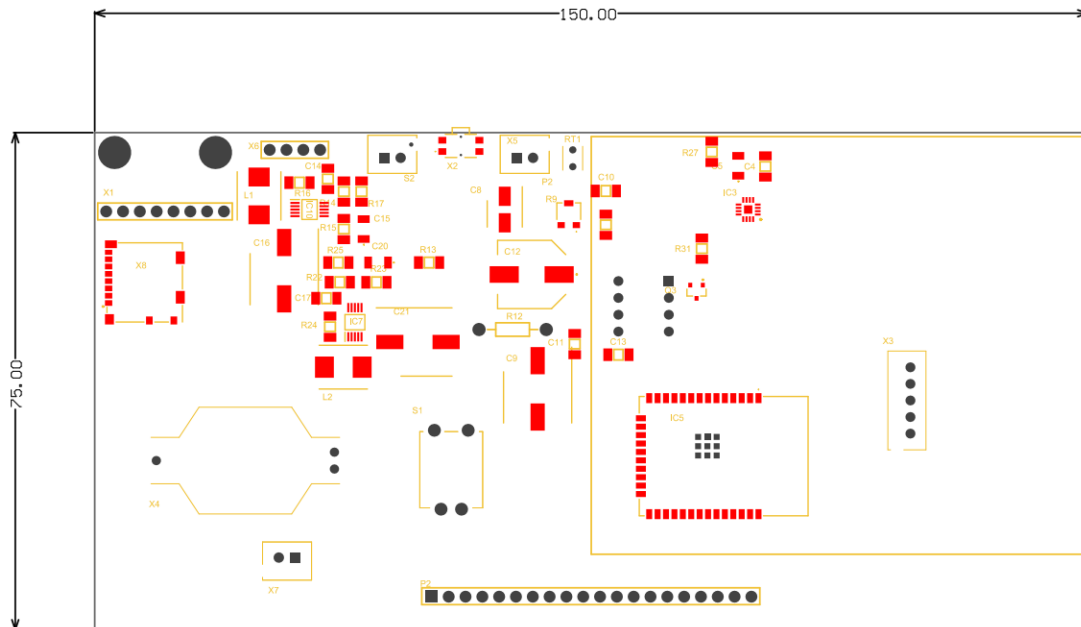
R2, R18	Atelier ELT	-	Resistance 33kΩ	2	0,15 CHF	0,30 CHF
R6, R7	Atelier ELT	-	Resistance 1.8kΩ	2	0,15 CHF	0,30 CHF
R8, R28	Atelier ELT	-	Resistance 1kΩ	2	0,15 CHF	0,30 CHF
R9	Atelier ELT	-	Resistance 1.2kΩ	1	0,15 CHF	0,15 CHF
R10, R11	Atelier ELT	-	Resistance 150Ω	2	0,15 CHF	0,30 CHF
R12	Atelier ELT	-	Resistance 10Ω1W	1	0,15 CHF	0,15 CHF
R14, R22	Atelier ELT	-	Resistance 15kΩ	2	0,15 CHF	0,30 CHF
R15	Atelier ELT	-	Resistance 560kΩ	1	0,15 CHF	0,15 CHF
R16, R24	Atelier ELT	-	Resistance 56kΩ	2	0,15 CHF	0,30 CHF
R17	Atelier ELT	-	Resistance 330kΩ	1	0,15 CHF	0,15 CHF
R19	Atelier ELT	-	Resistance 100Ω	1	0,15 CHF	0,15 CHF
R23	Atelier ELT	-	Resistance 680kΩ	1	0,15 CHF	0,15 CHF
R25	Atelier ELT	-	Resistance 220kΩ	1	0,15 CHF	0,15 CHF
R29	Atelier ELT	-	Resistance 18kΩ	1	0,15 CHF	0,15 CHF
R30	Atelier ELT	-	Resistance 33kΩ	1	0,15 CHF	0,15 CHF
RT1	Mouser	594-NTCLE203E3103GB0	Thermistance NTC 10kΩ	1	1,22 CHF	1,22 CHF
S1	Mouser	769-AHF22	Interrupteur à bascule AHF22	1	5,75 CHF	5,75 CHF
S2	Mouser	710-436331045822	Bouton poussoir SMD	1	0,53 CHF	0,53 CHF
IC9	Mouser	926-LM386N-1/NOPB	Ampli audio LM386	1	0,96 CHF	0,96 CHF
X1	DigiKey	1528-2873-ND	adafruit 4090 usb_c breakout board	1	2,68 CHF	2,68 CHF
X2, X5, X7	Atelier ELT	-	Connecteur JST XH 2pin	3	0,15 CHF	0,45 CHF
X3	Atelier ELT	-	Connecteur JST XH 5pin	1	0,17 CHF	0,17 CHF
X4	Mouser	614-HU2032-LF	Support pour pile 2032 (HU2032-LF)	1	2,02 CHF	2,02 CHF
X6	Atelier ELT	-	Header 4pin, 2.54mm 90°	1	0,20 CHF	0,20 CHF
X8	Mouser	538-104031-0811	Réceptacle micro SD	1	2,02 CHF	2,02 CHF
Y1	Mouser	732-FC135-32.76KAAG	Quartz 32.768kHz	1	0,54 CHF	0,54 CHF
Y2	Mouser	428-201112-ML01	Quartz 1.8432Mhz	1	14,97 CHF	14,97 CHF
-	Reichelt	DEBO ENCODER	Encodeur rotatif KY-040	1	2,02 CHF	2,02 CHF
-	Distrelec	13020757	Haut parleur 3W 32KC08	1	3,51 CHF	3,51 CHF
-	Mouser	979-1684.1101	Interrupteur ON/OFF 1684.1101	1	5,24 CHF	5,24 CHF
-	Conrad	1214021 - UQ	Batterie LiPo 2700mA 3.7V	1	45,95 CHF	45,95 CHF
-	Digitec	10434432	Carte microSD 8GB	1	7,80 CHF	7,80 CHF
P1	Mouser	645-598-8610-207F	LED standard - CMS Red/Green	1	0,47 CHF	0,47 CHF
R32, R33, R34	Atelier ELT	-	Resistance 0Ω	3	0,15 CHF	0,45 CHF
TOTAL						185,13 CHF

Les prix des composants récupéré à l'atelier ont été fixés environ au prix affiché chez Mouser

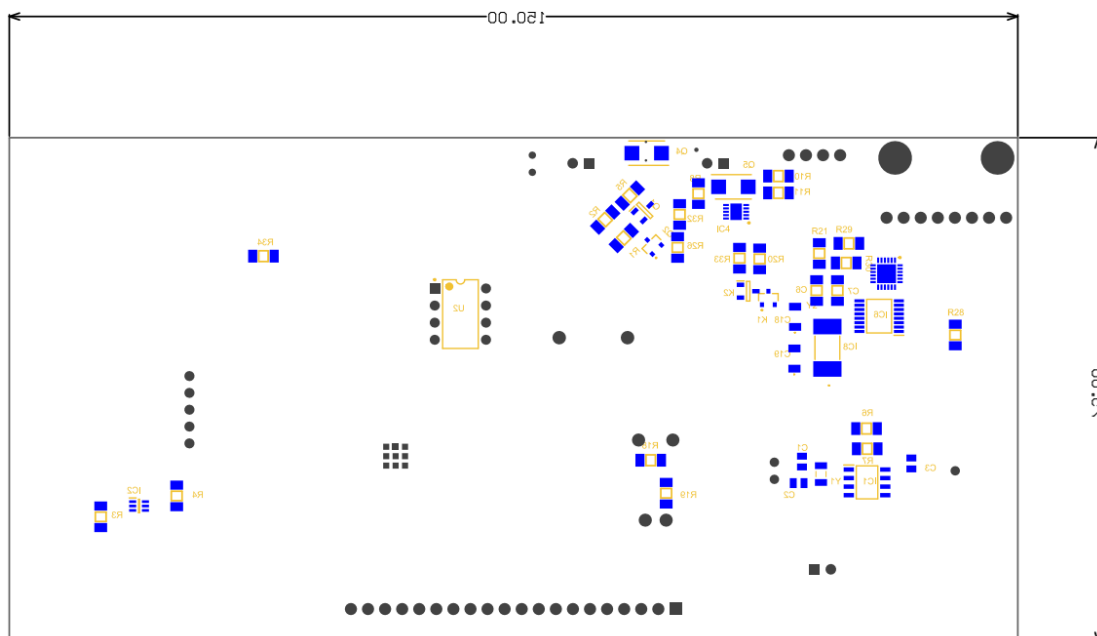
## 8. Conception des PCB

Mon circuit imprimé a été réalisé sur Altium Designer. Voici les plans :

**TOP FACE :**



**BOTTOM FACE :**



Pour les schémas et plans, U2 = IC9.

Je n'ai pas représenté les trous de fixation du car ils ont mal été conçus. Pour un futur PCB, seuls deux points de fixation suffisent ; celui en haut à gauche et un dans le coin en haut à droite (vue de dessus).

## 9. Programmation

Le programme a été réalisé sur l'IDE Arduino, il peut être téléchargé depuis le GitHub.

Dans ce rapport, je vais expliquer le fonctionnement général du programme ainsi que la façon dont j'ai réalisé certaines parties. Le détail de chaque partie se trouve en commentaire du programme.

### 9.1 Fonctionnement général

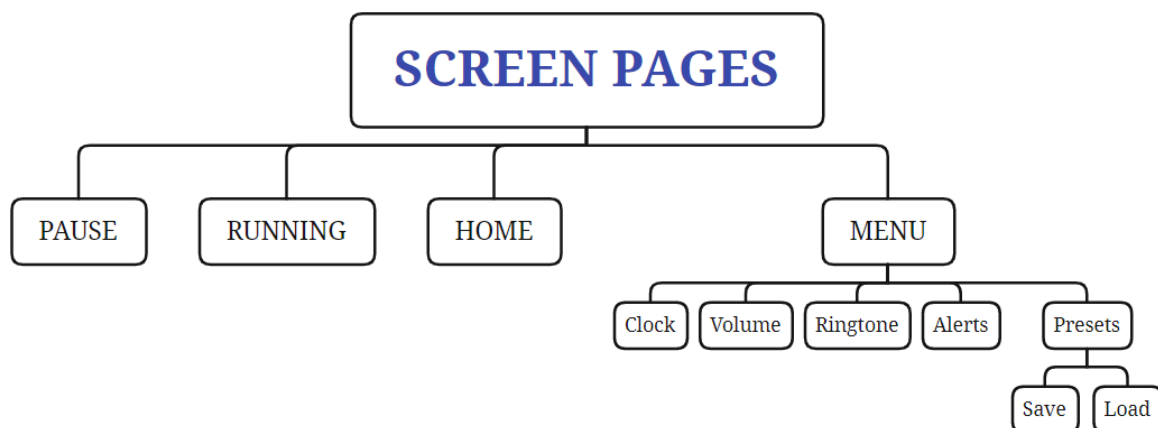
Le programme se base sur le principe d'une machine à état (état actuel déterminé par « screen\_page »).

Les actions effectuées sur l'encodeur rotatif ou sur la position de l'appareil permettent la transition entre états.

La carte SD sert de stockage audio tout comme d'EEPROM ; certaines valeurs y sont enregistrées et récupérées afin de conserver les réglages entre deux sessions.

La RTC sert de base de temps pour le minuteur ainsi que d'horloge (date et heure).

**Schéma de principe des états :**



L'affichage d'une horloge et l'enregistrements de presets n'étaient pas demandés, les autres pages étaient toutes décrites dans le cahier des charges.



## 9.2 Programmation de l'écran

J'utilise pour programmer l'écran la librairie [`<u8g2lib>`](#).

Chaque page d'état possède sa propre fonction, ces dernières se trouvent dans le fichier « ecran.cpp ». Pour afficher des images, le fichier « bitmap.h » contient les différents bitmaps.

Le constructeur de la librairie utilisée dépend de l'écran ainsi que du mode de communication.

```
U8G2_SSD1322_NHD_256X64_F_4W_HW_SPI u8g2(U8G2_R0, 19, 4);
```

On définit ici l'écran NHD256x64 avec une communication Hardware de 4 fils (CLK, DATA, CS et DC). Cette initialisation utilise le SPI par défaut du microcontrôleur utilisé, alors, pour que ce dernier corresponde avec celui connecté à l'écran, j'ai modifié le fichier « pins\_arduino.h » correspondant à ceux de l'esp32 :

**Extrait de « pins\_arduino.h » :**

(\\AppData\\Local\\Arduino15\\packages\\esp32\\hardware\\esp32\\2.0.17\\variants\\esp32\\pins\_arduino.h)

```
static const uint8_t SS = 5;|
static const uint8_t MOSI = 23;
static const uint8_t MISO = 19;
static const uint8_t SCK = 18;
```

## 9.3 Programmation de l'encodeur rotatif

Le programme de l'encodeur rotatif s'est fait à l'aide de la librairie [`<AiEsp32RotaryEncoder>`](#). Elle permet une gestion facile de l'accélération et de la plage de valeurs lues sur l'encodeur.

Les différentes fonctions que j'ai créées pour l'acquisition des valeurs / réglage de l'encodeur se trouvent dans le fichier « encoder.cpp »

## 9.4 Programmation de la RTC

Pour la RTC j'ai décidé de créer ma propre librairie, ce n'est pas très compliqué et comme ça les fonctions conviennent à mon application.

**Définition de la librairie :**

```
#define RTC_ADD 0x6F

class MyRTC {
public:
    MyRTC(uint8_t address = RTC_ADD);
    void init();
    void firstInit();
    void setTime(uint8_t hour, uint8_t minute, uint8_t second, uint8_t day,
uint8_t month, int year);
    uint8_t* getTime();
    void setCompTime();
    uint8_t getSec();
    static uint8_t old_sec;

private:
    int decToBcd(int val);
    int bcdToDec(int val);
    uint8_t _address;
    void getCompilationTime(uint8_t &hour, uint8_t &minute, uint8_t &second,
uint8_t &day, uint8_t &month, int &year);
};
```

Pour l'écriture et la lecture de la RTC, je me suis référé aux adresses décrites dans la datasheet. Toutes mes fonctions se base sur deux principales : getTime et setTime.

```
void MyRTC::setTime(uint8_t hour, uint8_t minute, uint8_t second, uint8_t day,
uint8_t month, int year)
{
    Wire.beginTransaction(_address);
    Wire.write(0); // Set register pointer to 0
    Wire.write(decToBcd(second) | 0x80); // Set bit to 1 to enable oscillator
    Wire.write(decToBcd(minute));
    Wire.write(decToBcd(hour) | 0x00); // Set 24H mode
    Wire.write(decToBcd(0 | 0x08)); // Day of the week (not used here)
    Wire.write(decToBcd(day));
    Wire.write(decToBcd(month));
    Wire.write(decToBcd(year));
    Wire.endTransmission();
}
```



```
uint8_t* MyRTC::getTime()
{
    uint8_t* timeArray = new uint8_t[6]; // Array to hold hour, minute, second,
    day, month, year
    Wire.beginTransmission(_address);
    Wire.write(0); // Set register pointer to 0
    Wire.endTransmission();

    Wire.requestFrom(_address, 7);
    timeArray[0] = bcdToDec(Wire.read() & 0x7F); // Mask ST bit for second
    timeArray[1] = bcdToDec(Wire.read()); // Minute
    timeArray[2] = bcdToDec(Wire.read() & 0x3F); // Mask mode bits for hour
    Wire.read(); // Day of the week (not used)
    timeArray[3] = bcdToDec(Wire.read()); // Day
    timeArray[4] = bcdToDec(Wire.read() & 0x1F); // Month
    timeArray[5] = bcdToDec(Wire.read()); // Year
    return timeArray;
}
```

## 9.5 Programmation de la lecture/écriture SD

Comme mes fichiers audios se trouvent sur la carte SD, j'ai créé une classe qui comprend la programmation de la carte SD et celle de l'audio.

Pour la carte SD, j'utilise les bibliothèques [<SD>](#) et [<FS>](#). Pour l'audio j'utilise la bibliothèque [<ESP32-audioI2S>](#)



La bibliothèque ESP32-audioI2S comprend son propre fichier `<Audio.h>`, ne pas confondre avec celui proposé par Arduino.

Voici la définition de ma classe :

```
class SDReader {
public:
    SDReader(int numFiles);

    void init();
    void listFilesInDirectory(fs::FS &fs, const char * dirname);
    String getFileName(int index) const;
    void play_audio(uint8_t fileNo);
    void stop_audio();
    void loop_audio();
    void set_volume(uint8_t volume);
    void writeFile(fs::FS &fs, const char * path, const char * message);
    void appendFile(fs::FS &fs, const char * path, const char * message);
    void readFile(fs::FS &fs, const char * path);
    void set_alerts(uint8_t index, bool val);
    bool read_alerts(uint8_t index);
    void play_alert(uint8_t fileNo, uint8_t pack);
    void play_bat_alert();
    void write_memory(const char* parameter, int value);
    int read_memory(const char* parameter);

private:
    Audio audio;
    String* fileNames;
    int numFiles;
    bool alerts_state[4];
    void updateFileContent(String& content, const char* parameter, int value);
    String getParameterValue(const String& content, const char* parameter);
};
```

Le programme permet la récupération des noms des fichiers audio, on peut donc modifier ces derniers à notre guise. La seule restriction est qu'il ne faut pas dépasser 12 caractères.

## 9.6 Programmation du minuteur

J'ai créé une classe pour le minuteur également. Celui-ci dépend du changement de la valeur des secondes de la RTC

**Définition de la classe MyTimer :**

```
class MyTimer
{
public:
    void set_timer(int seconds);
    int get_timer();
    void timer_state(bool val);
private:
    int seconds_left;
    bool state;
};
```

## 9.7 Récupération du pourcentage de la batterie

Afin de connaître le pourcentage de batterie restante, il faut d'abord lire la valeur reçue sur l'ADC, puis la reconvertir à la valeur « réelle » de la tension. Il faut ensuite convertir la tension de la batterie restante comparée à sa tension maximale en pourcentage.

**Récupération du pourcentage de la batterie :**

```
// Les valeurs maximales et minimales de tension pour une batterie LiPo 1S
const float MAX_VOLTAGE = 4.3; // Tension maximale de la batterie pleine
const float MIN_VOLTAGE = 3.0; // Tension minimale de la batterie vide
const float ADC_MAX = 4095.0; // Valeur maximale de l'ADC (résolution 12 bits)

int get_battery_state() {
    int adcValue = analogRead(BAT_PIN);
    float voltage = adcValue * (MAX_VOLTAGE - MIN_VOLTAGE) / ADC_MAX +
MIN_VOLTAGE;
    int batteryPercentage = map(int(round(voltage*100)), 10*MIN_VOLTAGE,
10*MAX_VOLTAGE, 0, 100);
    return batteryPercentage;
}
```

Pour le calcul du pourcentage, les variables sont multipliées par 10 car la fonction « map() » ne marche qu'avec des nombres entiers.





## 10. Fichiers de la carte SD

Comme dit précédemment, la carte SD sert à la fois de stockage audio et de stockage de valeurs. La carte SD contient du document et un fichier :

- Document « Sonneries » : contient les fichiers audios des différentes sonneries (MAX :5)
- Document « Alertes » : contient les différents fichiers audios des alertes (temps restant et batterie faible)
- Fichier « Data.txt » : Contient les valeurs à récupérer lors de l'initialisation

Les noms de ces trois fichiers ainsi que des paramètres du fichiers « Data.txt » ne doivent en aucun cas être modifiés.

Il faut également savoir que le temps de sonnerie correspond à la durée du fichier audio. Actuellement, les cinq fichiers durent 20 secondes car c'était le temps demandé dans le cahier des charges.

## 11. Test final selon cahier des charges

### 11.1 Éléments internes imposés

Ecran 7cm x 2cm minimum	✓
Encodeur rotatif	✓
Recharge batterie par USB_C	✓
Interrupteur pour allumer l'appareil	✓

### 11.2 Fonctionnalités imposées

Réglage de la minuterie 10secondes – 4heures	✓
Réglage du volume	✓
Activation ou non des avertissements sonores	✓
Start/Stop en fonction de l'orientation	✓
Témoin visuel pause	✓
Format d'affichage hh : mm puis mm : ss	✓
Musique ou Bip comme sonnerie	✓
Autonomie de minimum 8 heures	✓

## 12. Problèmes rencontrés

Lors de la réalisation de ce projet, plusieurs problèmes sont survenus. Pour commencer, après avoir monté mon circuit une première fois, je n'avais aucune tension en sortie des convertisseurs buck/boost. En inspectant minutieusement le circuit, j'ai remarqué qu'une petite boule de brasure court-circuitait le pin de feedback du convertisseur 3.3V avec le GND. Je l'ai alors enlevé et ai retesté le circuit, aucun résultat. J'ai alors revérifié le circuit et le schéma, tout était correct. J'en ai donc déduit que les convertisseurs ne marchaient pas.

Comme je n'en avais plus en stock pour les remplacer, j'ai profité de les commander dans un autre boîtier afin de faciliter le brasage et éviter tout problème. J'ai donc dû modifier le pcb car le footprint diffère légèrement entre les deux boîtiers. J'en ai profité pour modifier quelques autres détails sur mon pcb.

Modifications effectuées :

- Modification de la footprint des convertisseurs
- Modification des pins SPI du l'écran (expliqué au point 5.11)
- Modification de l'emplacement d'un connecteur JST, qui entraînait en collision avec le support de l'encodeur.

Une fois les nouveaux convertisseurs montés, le problème a disparu. La cause du dysfonctionnement des premiers circuits est encore incertaine mais d'après moi, le court-circuit entre le feedback et le GND a eu pour conséquence de demander une tension de sortie énorme (car les résistances sur le pin feedback contrôlent la tension de sortie voulue) et cette tension l'a alors cramé.

Ensuite, un autre problème est survenu ; la carte SD et l'ESP étaient inaccessibles depuis l'ordinateur, mais les ports COM générés par le CP2105 étaient affichés. En essayant de trouver d'où venait ce problème, une mauvaise manipulation du circuit a créé un court-circuit entre le 3.3V et le GND. Pour trouver d'où venait ce court-circuit, j'ai isolé chaque partie alimentée par le 3.3V et cela m'a permis de savoir que le court-circuit venait du CP2105, il avait cramé.

Causes probables :

- Court-circuit effectué avec ma bague lors de la manipulation du circuit.
- Surtension due à une décharge électrostatique lors de la déconnection du câble USB\_C (il m'est arrivé de le débrancher sans avoir pressé sur le switch d'alimentation au préalable).

J'ai alors changé ce composant et j'en suis revenu au stade précédent ; impossible de programmer l'ESP. Après avoir fait beaucoup de recherches, j'ai découvert que le CP2105 possédait deux modes ; un mode GPIO et un mode Modem. Il s'avère que le mode par défaut est le mode GPIO.

## 7. GPIO Mode and Modem Mode

Each interface on the CP2105 can be configured in either GPIO Mode or Modem Mode. This allows the SCI and ECI to have either modem control signals or GPIO signals available at various pins. Table 11 shows the functions that are available in each mode. By default, both interfaces are configured for GPIO Mode.

**Table 11. CP2105 Modem Mode and GPIO Mode**

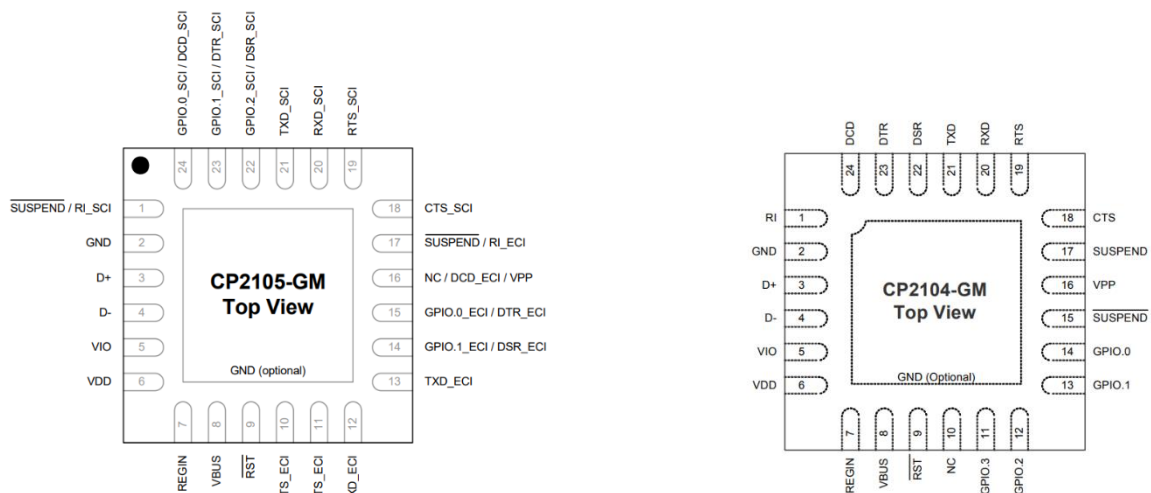
Interface	Pin #	Modem Mode	GPIO Mode
Standard Communications Interface	24	DCD_SCI	GPIO_0_SCI
	23	DTR_SCI	GPIO_1_SCI
	22	DSR_SCI	GPIO_2_SCI
	1	RI_SCI	$\overline{\text{SUSPEND\_SCI}}$
Enhanced Communications Interface	15	DTR_ECI	GPIO_0_ECI
	14	DSR_ECI	GPIO_1_ECI
	17	RI_ECI	$\overline{\text{SUSPEND\_ECI}}$

Only one mode can be selected for each interface. Also, the mode of the CP2105 can only be configured once and cannot be reset to the default configuration after being programmed. Refer to "AN223: Runtime GPIO Control for CP210x" for more information on how to configure the port pins of the CP2105.

Comme le dit la datasheet, le circuit peut être reconfiguré une seule fois. Une des solutions pour que mon circuit fonctionne était alors de créer un PCB sur lequel je pourrais connecter les pins nécessaires à cette configuration (PCB obligatoire car il aurait été trop difficile d'atteindre les pins sur le boîtier). Cette solution n'en est pas vraiment une car je n'avais pas le temps d'effectuer des recherches sur la façon exacte de le configurer.

J'ai alors décidé d'abandonner la fonction que je voulais ajouter : la communication avec la carte SD depuis l'ordinateur. Cela ne me coûterait pas de temps car je n'ai qu'à changer le CP2105 pour un CP2104 (même circuit mais avec un seul UART). D'autant plus que ces deux circuits ont le même pinout.

### Comparaison des pinouts :





Un dernier problème est alors apparu lors de la programmation de l'ESP, l'erreur suivante apparaissait : **MD5 of file does not match data in flash ! ( fatal error )**

J'ai effectué des recherches à propos de cette erreur et les forums ne parlaient jamais de la même chose ; certains parlaient de problème quant aux connections, certains parlaient d'ESP brûlé et d'autre de problème de configuration.

Au final, la page qui m'a réussi est la suivante :

<https://github.com/espressif/arduino-esp32/issues/1890>

J'ai lancé un terminal de commande et ai installé esptool avec la commande «pip install esptool »

J'ai ensuite écrit les commandes suivantes :

- esptool --port write\_flash\_status --non-volatile 0
- espfuse.py set\_flash\_voltage 3.3V

Après cela, j'ai pu programmer mon ESP et tout marchait à merveille.

Le dépannage de tous ces problèmes est la principale cause de l'état actuel du PCB ; des pistes ont été coupées puis rebrasées, plusieurs composants changés et deux bouts de pistes se sont arrachés, ce qui explique la présence de deux fils sur le PCB.

Comme ces problèmes m'ont pris pas mal de temps à résoudre, une rallonge de temps m'a été accordée.

## 13. Améliorations

### 13.1 Améliorations hardware

Le dimensionnement des condensateurs montés avec la RTC peut être revu comme expliqué plus haut. Comme expliqué au point 5.11, les pins utilisés pour l'écran peuvent être communs avec ceux utilisés pour la carte SD.

L'emplacement des trous de fixation doivent aussi être revu, pour qu'ils coïncident avec le boîtier et que ce soit facile à monter. L'utilisation d'un réceptacle de carte SD push/push pourrait aussi être une amélioration, cela éviterait le besoin de faire une trappe plutôt qu'un seul trou pour la carte.

La batterie pourrait également être remplacée par une plus petite, 1800 mAh suffiraient pour tenir au moins 8 heures.

Si la batterie est plus petite, la taille du boîtier pourrait également être réduite. Il faudrait juste, dans ce cas, revoir le PCB complet.

### 13.2 Améliorations software

Le programme pourrait probablement être optimisé à plusieurs endroits. L'ajout d'une possibilité de changer la date et l'heure était prévu, je n'ai pas eu le temps de le réaliser. C'est une fonctionnalité ajoutée, donc ce n'est pas grave, mais ça reste une possible amélioration.

### 13.3 Autres améliorations

La planification aurait également pu être améliorée, par exemple j'avais prévu de faire la programmation sur plaque d'essais avant le montage du circuit mais cela était impossible car mes composants sont, pour la plupart, SMD.

J'aurais également dû plus me renseigner sur les composants que j'utilise, cela m'aurait évité une surprise comme avec le CP2105.

## 14. Auto-évaluation du travail

L'appareil rendu est fonctionnel et répond à toutes les demandes du cahier des charges. Je trouve que j'ai appliqué correctement les connaissances que j'ai acquises durant mon apprentissage pour la réalisation de ce projet.

Je devrais en revanche plus prendre le temps de réfléchir lors du développement du circuit, cela m'aurait évité certaines erreurs. Malgré cela, j'ai réussi à trouver la solution aux problèmes rencontrés de manière autonome.

Sinon, je suis très content du résultat de mon programme. L'interface utilisateur est jolie et facile d'utilisation.

## 15. Conclusion

Pour conclure, je suis plutôt fier du résultat final de ce projet. J'ai beaucoup appris durant la réalisation car je ne connaissais pas très bien la communication avec la carte SD ou encore l'interface I2S pour l'audio. Je suis aussi content du résultat de mon boîtier, il est loin d'être parfait mais c'est la première fois que je design entièrement moi-même un boîtier.

Je reste un peu déçu par rapport à la fonctionnalité que je voulais ajouter, je trouve que pouvoir accéder à la carte SD directement sans la sortir du boîtier aurait été un bel avantage.

J'ai également appris à structurer un projet et à travailler avec des délais et des contraintes, ce qui est très important dans le monde professionnel.

## 16. Remerciements

Je remercie M. Choffat Frédéric pour m'avoir suivi tout au long de ce projet et d'avoir su répondre à mes questions.

Je remercie également M. Migliano Sandro et M. Wälti Gabriel pour m'avoir suivi en tant qu'experts et conseillé sur certains points.

Je remercie finalement mon père M. Michel Blaise pour la relecture de ce rapport.

## 17. Liens utiles

Lien vers mon GitHub :

[https://github.com/gaetan-m11/TPI\\_TIME\\_TIMER](https://github.com/gaetan-m11/TPI_TIME_TIMER)

Lien pour les informations concernant l'ESP32 :

<https://lastminuteengineers.com/esp32-pinout-reference/>

Lien vers le forum pour le problème rencontré :

<https://github.com/espressif/arduino-esp32/issues/1890>

Lien vers le site avec lequel j'ai dimensionné certains composants :

<https://jansson.us/resistors.html>

## 18. Annexes

- Journaux de travail