

Rapport de projet

2228

Alarme de fenêtre ouverte

**Réalisé par :**

Miguel Santos

À l'attention de :

Serge Castoldi

Juan José Moreno

Début

16 novembre 2022

Fin

14 juin 2023

Table des matières

1.	Introduction	1
2.	Pré-étude	1
2.1.	Approche	1
2.2.	Détection de la fenêtre	1
2.3.	Bouton.....	1
2.4.	LED RGB	2
2.5.	Alimentation	2
2.6.	Microcontrôleur et module de communication.....	3
3.	Boitiers	4
4.	Évaluation des coûts	5
5.	Design	6
5.1.	Choix des composants : Émetteur	6
5.2.	Choix des composants : Récepteur	12
6.	Hardware.....	14
6.1.	PCB de l'émetteur	14
6.2.	PCB du récepteur.....	16
6.3.	Boîtier de l'émetteur	17
7.	Montage et mise en service	18
8.	Software	19
8.1.	Logique de programmation du NRF52840	19
8.2.	Software émetteur : NRF52840.....	20
8.3.	Software récepteur : NRF52840.....	21
8.4.	Software récepteur : PIC32MX130.....	21
8.5.	Tests de fonctionnement.....	23
9.	État final	26
10.	Conclusion.....	27
11.	Annexes	28
11.1.	Résumé du projet.....	28
11.2.	Affiche de projet	28
11.3.	Cahier des charges	28
11.4.	Fiche de modification	28
11.5.	Planning.....	28
11.6.	Journal de travail.....	28
11.7.	Émetteur : Schéma électrique.....	28

11.8.	Émetteur : BOM	28
11.9.	Émetteur : Schéma d'implémentation	28
11.10.	Émetteur : Schéma mécanique.....	28
11.11.	Récepteur : Schéma électrique.....	28
11.12.	Récepteur : BOM.....	28
11.13.	Récepteur : Schéma d'implémentation	28
11.14.	Récepteur : Schéma mécanique.....	28
11.15.	Boitier : Cotation « Top »	28
11.16.	Boitier : Cotation « Bottom »	28
11.17.	Software : NRF52840 Emetteur.....	28
11.18.	Software : NRF52840 Récepteur.....	28

1. Introduction

L'objectif de ce projet est la réalisation d'un dispositif permettant de détecter si une fenêtre est restée ouverte à certaines heures dans une salle de l'ETML-ES. Le cas échéant, une alerte sera envoyée par e-mail. Ce projet s'inspire des produits déjà présents sur le marché de la domotique.

Le cahier des charges est fournis à l'annexe 11.3.

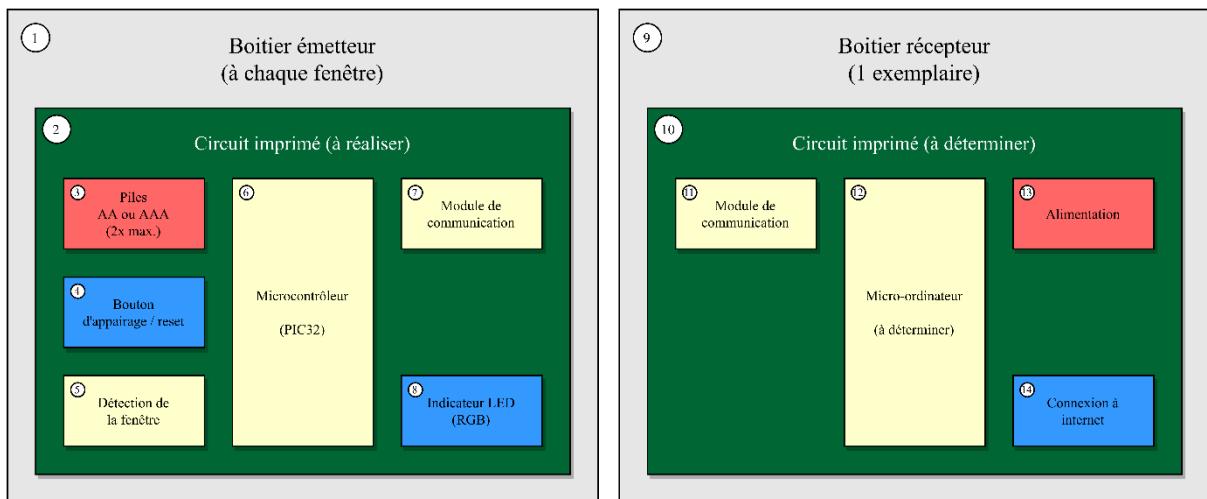
Le planning complet du projet est fournis à l'annexe 11.5.

2. Pré-étude

2.1. Approche

Le projet sera constitué de deux parties distinctes :

- Multiples émetteurs, disposés à chaque fenêtre souhaitée.
- Un unique récepteur, placé dans la même salle que les fenêtres.



2.2. Détection de la fenêtre

Pour réaliser la détection de la fenêtre, je me suis basé sur les produits équivalents disponibles sur le marché. La méthode consiste à placer un aimant permanent sur la fenêtre et à le détecter à l'aide d'un capteur à effet Hall.

L'avantage réside dans la fiabilité de la mesure. La probabilité d'avoir un autre aimant perturbant notre circuit est extrêmement faible. En comparaison, un capteur optique pourrait se voir perturbé par de l'eau, une feuille ou tout autre objet se plaçant devant le capteur.

La consommation de ce genre de circuit est faible et sont proposés à de faibles prix. Ce qui en fait un choix idéal pour cette application.

2.3. Bouton

Un bouton sera accessible depuis l'extérieur du boîtier pour permettre à l'utilisateur la réalisation de l'appairage ou de diverses configurations si nécessaires.

2.4. LED RGB

Une LED sera placé afin d'indiquer à l'utilisateur l'état de chaque émetteur lors de l'appairage ou si une erreur survient par exemple. Elle devra être éteinte lors d'un fonctionnement normal pour réduire la consommation de courant.

2.5. Alimentation

Le cahier des charges proposait l'utilisation de deux piles AAA. Néanmoins, je me suis rapidement rendu compte des avantages que peuvent offrir des batteries au lithium. Pour cette raison, j'ai décidé de proposé deux approches différentes pour réaliser l'alimentation du circuit.

	Avantages	Inconvénients
Piles AAA alcaline	Facilité d'entretien Facilité de développement	Coût sur le long terme Grande quantité de déchets
Batterie lithium	Meilleure densité d'énergie Meilleure durabilité	Développement plus complexe Nécessite des circuits de charge et de protection Recharge annuelle nécessaire

Figure 2 : Comparaison entre piles et batterie

Le design devra déterminer lequel de ces deux méthodes sera le plus adapté à implémenter dans ce projet.

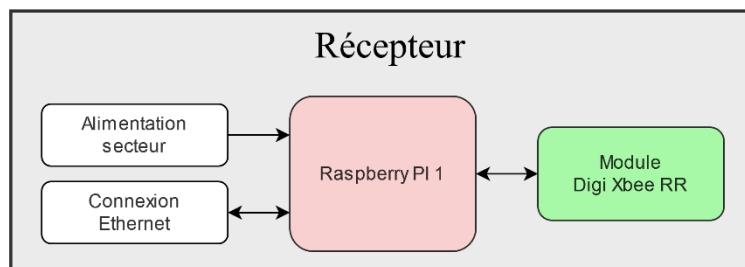
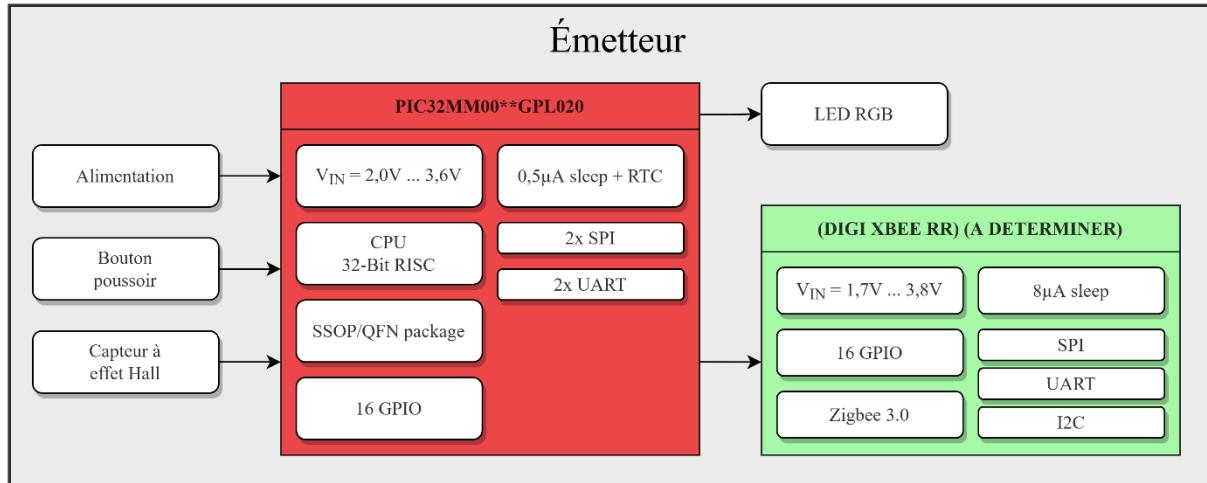
Dans les deux cas, un circuit de mesure de la charge du niveau d'énergie restant devra être réalisé. Il devra faire preuve d'un courant de fuite le plus faible possible.

2.6. Microcontrôleur et module de communication

2.6.1 Méthode PIC32

Pour cette partie, j'ai étudié deux méthodes différentes. La première était basé sur l'utilisation d'un PIC32 couplé à un module Xbee, intégrant un processeur ARM, permettant une communication par le protocole Zigbee.

Après avoir commencé ma pré-étude sur cette partie, j'ai jugé qu'il était peut-être exagéré d'intégrer deux processeurs pour une application qui est peu gourmande en ressources. De plus, le prix unitaire se retrouvait être très élevé.

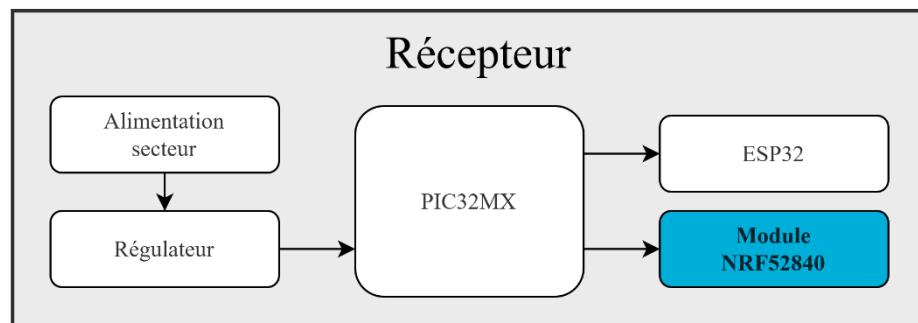
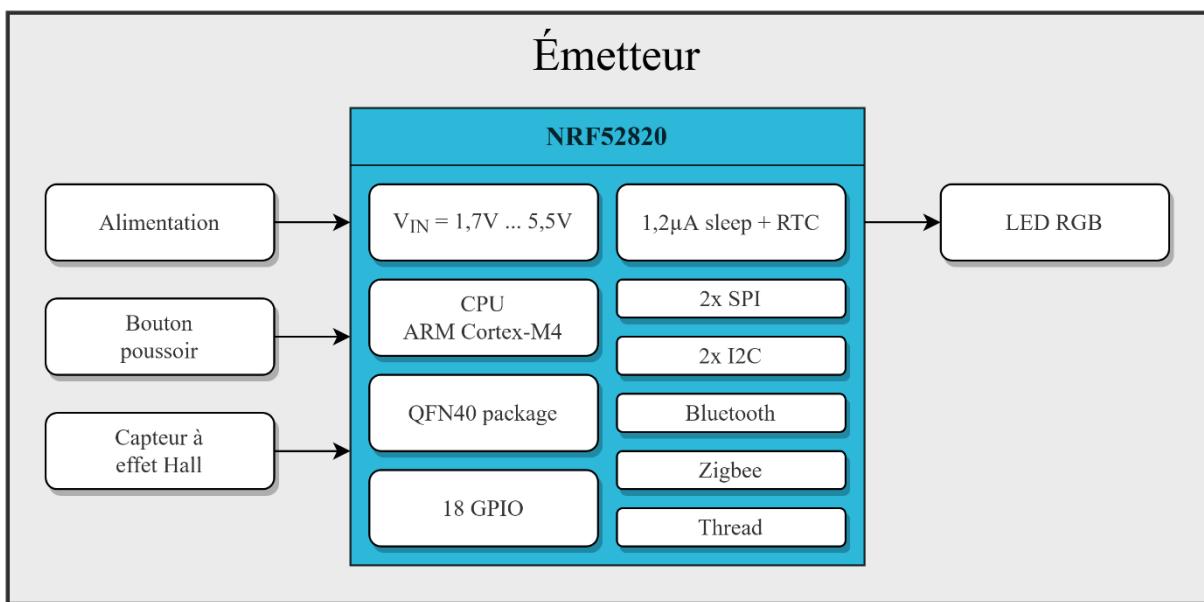


2.6.2 Méthode Nordic

J'ai donc étudié la possibilité d'utiliser un module « NRF52820 » du fabricant « Nordic Semiconductor ». J'ai eu l'occasion de le survoler rapidement lors de mon stage en entreprise. Il a l'avantage d'intégrer à la fois un CPU et un module de communication Zigbee, ainsi que plusieurs autres périphériques. De plus, il offre un prix bien plus avantageux que la méthode avec le PIC32.

Néanmoins, il était demandé d'intégrer un PIC32 dans le projet. Pour cela, j'ai donc décidé de le réaliser dans

Je souhaiterais privilégier cette méthode pour ces avantages techniques et économiques. Néanmoins, elle présente le désavantage d'être plus complexe à implémenter, mais cela me permettrait d'améliorer mes compétences personnelles en prévision de l'entreprise que j'intégrerai à la suite de mes études.



3. Boitiers

La réalisation des boitiers sera faite en impression 3D. De cette manière, une plus grande flexibilité est des designs plus pratiques seront possible pour un coût équivalent en contrepartie de temps supplémentaires pour la conception.

4. Évaluation des coûts

Comme j'ai proposé deux méthodes pour ce projet, j'ai réalisé deux estimations des coûts distinctes. Certaines parties sont communes aux deux méthodes.

J'ai réalisé une estimation par unité, pour une salle complète (en moyenne 15 fenêtres) et une estimation si le projet devait être étendu à l'entièreté de l'école (environ 150 fenêtres) en considérant que chaque salle serait équipé d'un récepteur.

	Méthode PIC32			
	Description	Estimation / unité	Estimation / salle	Estimation / école
Émetteur	Alimentation	10	150	1 400
	Bouton poussoir	1	15	140
	Capteur à effet Hall	1	15	140
	LED RGB	1	15	140
	Boitier	5	75	700
	PIC32MM00**GPL020	2	30	280
	Modules Xbee RR (émetteur)	25	375	3 500
Sous-total émetteur		45	675	6 300
Récepteur	Modules Xbee RR (récepteur)	25	25	200
	Alimentation secteur	20	20	160
	Sous-total récepteur	45	45	360
Total final		90 CHF	720 CHF	6 660 CHF

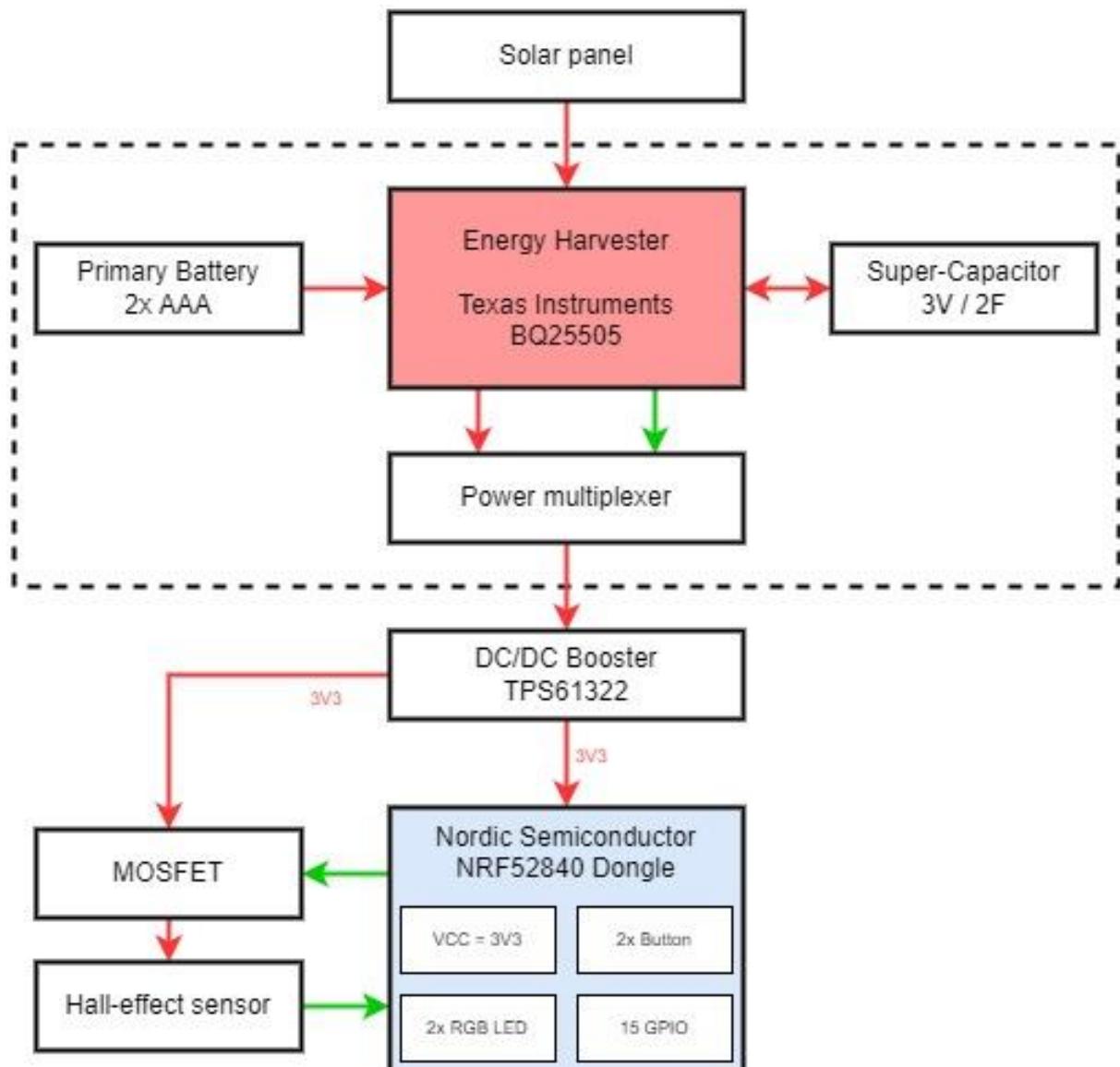
	Méthode Nordic			
	Description	Estimation / unité	Estimation / salle	Estimation / école
Émetteur	Alimentation	10	150	1 400
	Bouton poussoir	1	15	140
	Capteur à effet Hall	1	15	140
	LED RGB	1	15	140
	Boitier	5	75	700
	NRF52820	5	75	700
	Sous-total émetteur	23	345	3 220
Récepteur	Alimentation secteur	20	20	160
	Régulateur	5	5	40
	PIC32MX	15	15	120
	ESP32	2	2	16
	Module NRF52840	10	10	80
	Sous-total récepteur	52	52	416
Total		75 CHF	397 CHF	3 636 CHF

On peut voir que la méthode Nordic offre un avantage financier non négligeable. Ce qui appuie ma volonté d'utiliser cette méthode.

5. Design

5.1. Choix des composants : Émetteur

5.1.1 Schéma bloc de l'émetteur



5.1.2 Cellule photovoltaïque | KXOB25-05X3F-TR



Le panneau solaire représente la source d'énergie principale de l'émetteur. Il devrait lui permettre d'atteindre une autonomie complète sans nécessiter de changement de piles ou batteries.

Composé de cellule photovoltaïque monocrystalline, cela permettra d'exploiter toute la longueur d'onde de la lumière visible et en conséquence la lumière intérieure des néons.

Le rendement élevé (25%) permettra de maximiser la quantité d'énergie captée dans une surface réduite. Les tensions sont parfaitement adaptées à la plage de fonctionnement du circuit de gestion décrit aux points suivants.

5.1.3 Energy harvester | BQ25505

Le BQ25505 de Texas Instrument est un circuit permettant la gestion de source d'énergie de très faible puissance. Il est en mesure de gérer le stockage dans un super-condensateur. En plus de cela, une batterie primaire peut-être relié au circuit pour offrir une source d'énergie supplémentaire et plus stable.

La datasheet fournit une application typique avec une cellule solaire.

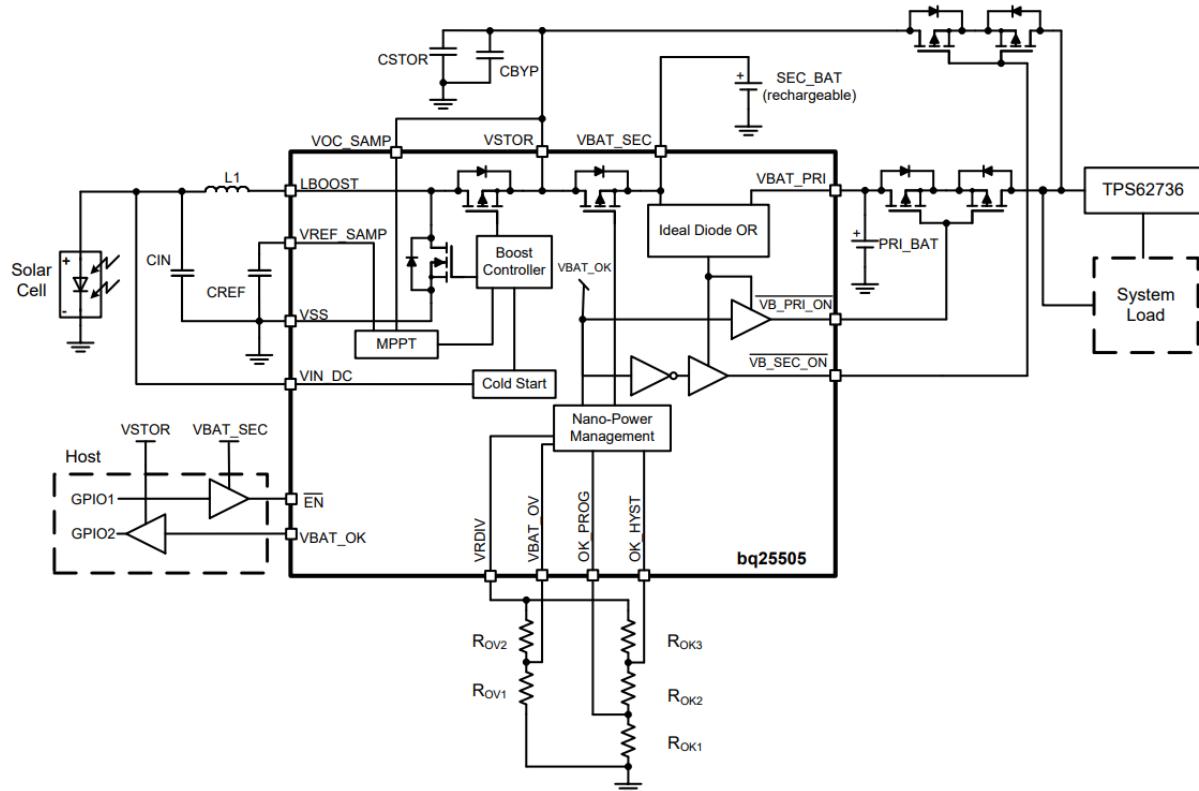
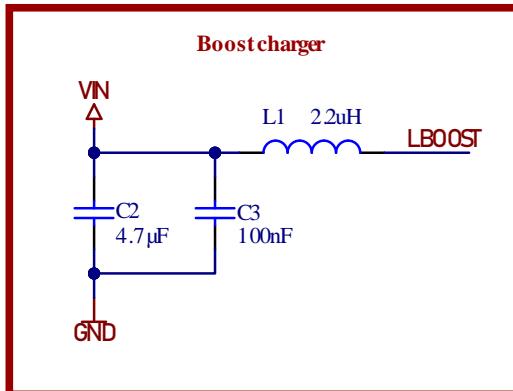


Figure 4 : Application typique BQ25505

Chargeur boost | C2 & L1

En entrée, un boost permet d'élever la tension de la cellule solaire. Les valeurs de composants à utiliser sont fournies par la datasheet (p. 20).



Protection de surtension | R1 & R2

Un pont résistif permet de fixer la tension maximale admissible sur la batterie secondaire (le super-condensateur). La formule est fournie par la datasheet (p. 13).

$$V_{BAT_{OV}} = \frac{3}{2} * V_{BIAS} * \left(1 + \frac{R_{OV2}}{R_{OV1}}\right)$$

$$V_{BIAS} = V_{RDIV} = 1.21 \text{ V}$$

$$R_{OV2} + R_{OV1} = 13 \text{ M}\Omega$$

Dû aux valeurs élevées de résistance et donc de courant très faible, une attention particulière doit être faite concernant le bruit pouvant perturber ce pont. De plus, les produits appliqués sur le circuit tel que le flux de la soudure ou les produits de nettoyages peuvent engendrer une résistance parallèle du même ordre de grandeur. Le fabricant fournit des recommandations dans la datasheet à ce sujet.

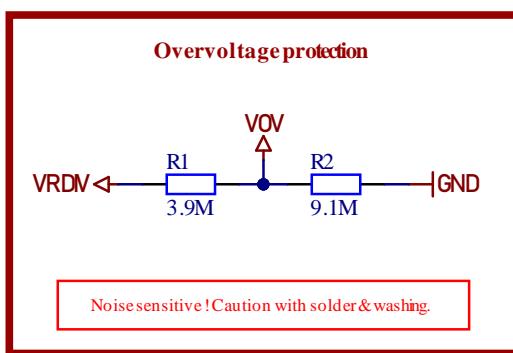


Figure 5 : Pont résistif fixant la valeur de protection de surtension

Stockage d'énergie | B1 & B2

Pour stocker l'énergie, un super-condensateur est utilisé. Celui-ci a été dimensionné grâce à un outil de calcul fourni par Texas Instrument. La plage de tension utile est celle entre la tension minimale de sortie du BQ25505 et la tension maximale du condensateur. Pour maximiser cette plage, il faut donc choisir un condensateur avec la plus haute tension possible. Néanmoins, pour éviter d'avoir des coûts trop élevés, j'ai préféré en choisir un d'une tension de 3V avec une plus grande capacité.

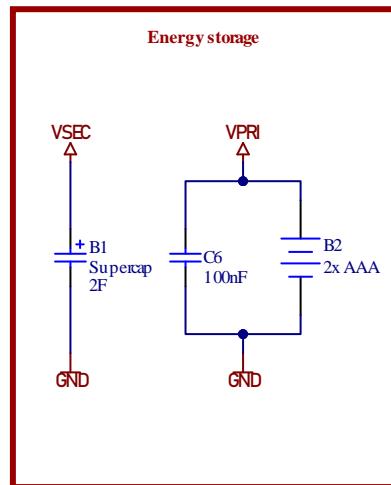


Figure 6 : Stockage d'énergie

Condensateurs | C3 & C4 & C5

Le condensateur C3 est nécessaire pour que le circuit puisse fixer sa tension de référence en interne. Il est particulièrement sensible aux courant de fuites et doit les éviter au maximum tout en étant d'une précision la plus grande possible. Il doit être au minimum de type X7R ou COG.

Les condensateurs C4 et C5 sont les condensateurs de sortie et leur valeur est fixée par la datasheet.

De manière globale, tous les condensateurs reliés au BQ25505 doivent posséder le courant de fuite le plus faible possible afin augmenter le rendement global.

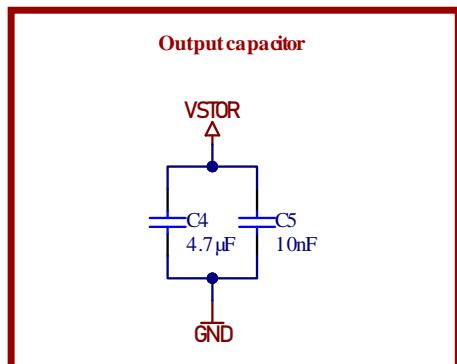


Figure 7 : Condensateurs de sortie

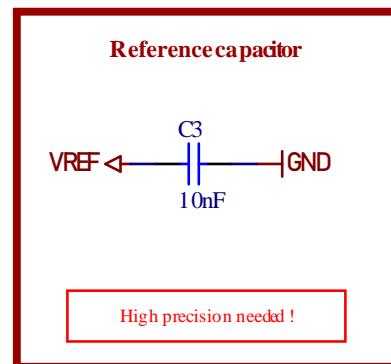
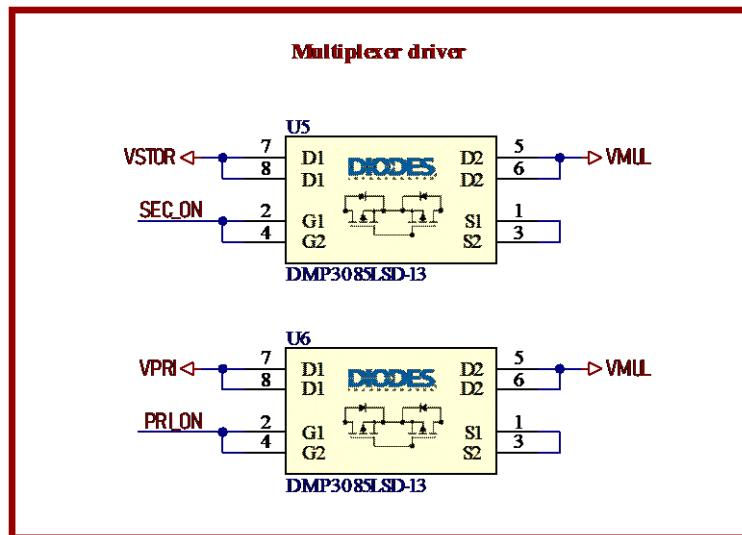


Figure 8 : Condensateur de référence

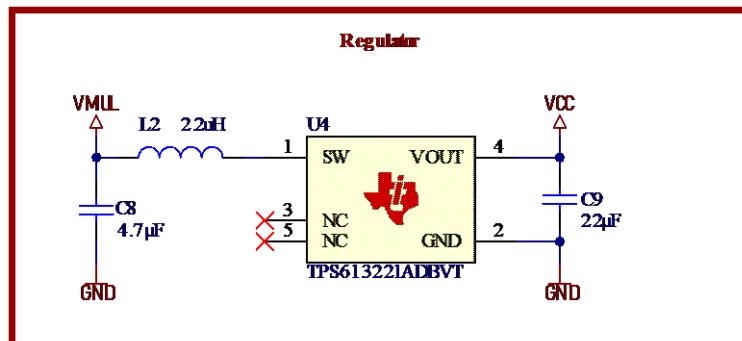
Multiplexeur | U5 & U6

Le BQ25505 offre la possibilité de multiplexer les différentes sources de stockage pour notamment automatiquement commuter sur la batterie primaire lorsque la secondaire n'a plus assez d'énergie ou une tension trop faible.



5.1.4 Régulateur 3V3 | U4

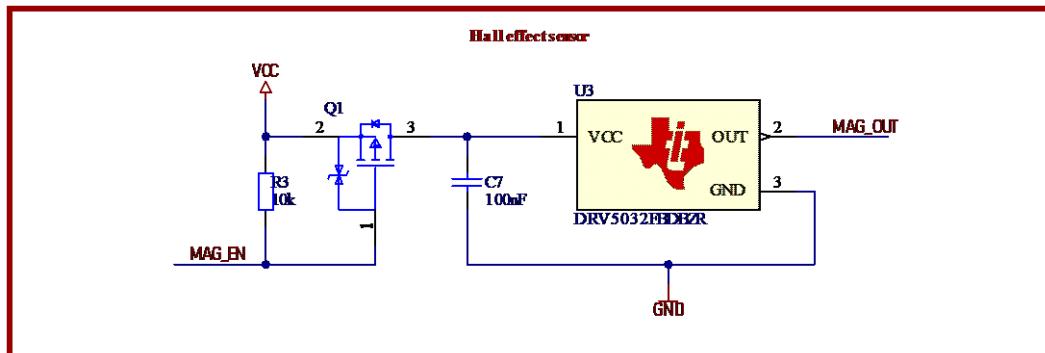
La tension fournit par le super-condensateur sera variable par nature. Pour la stabiliser, il faut veiller à utiliser un régulateur 3V3. Les composants ont été dimensionné grâce à l'utilitaire « WEBENCH » de Texas instrument.



5.1.5 Capteur à effet hall

Pour détecter la présence ou non de la fenêtre, j'ai décidé d'implémenter un capteur à effet hall agissant comme un interrupteur. La présence d'un champ magnétique supérieure à une valeur définie fera passer la sortie à un niveau logique haut.

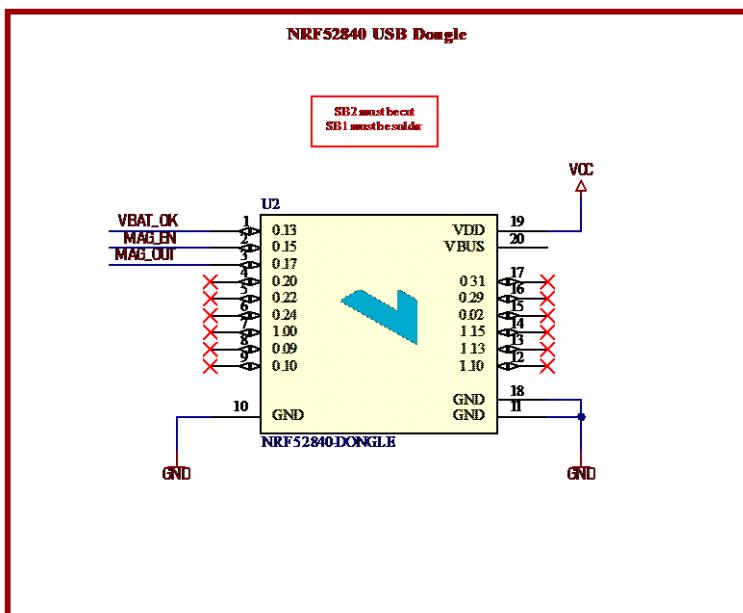
Pour réduire le risque de courant de fuite dans le capteur, un MOSFET est installé sur son alimentation qui sera contrôlé par le processeur.



5.1.6 NRF52840 USB Dongle

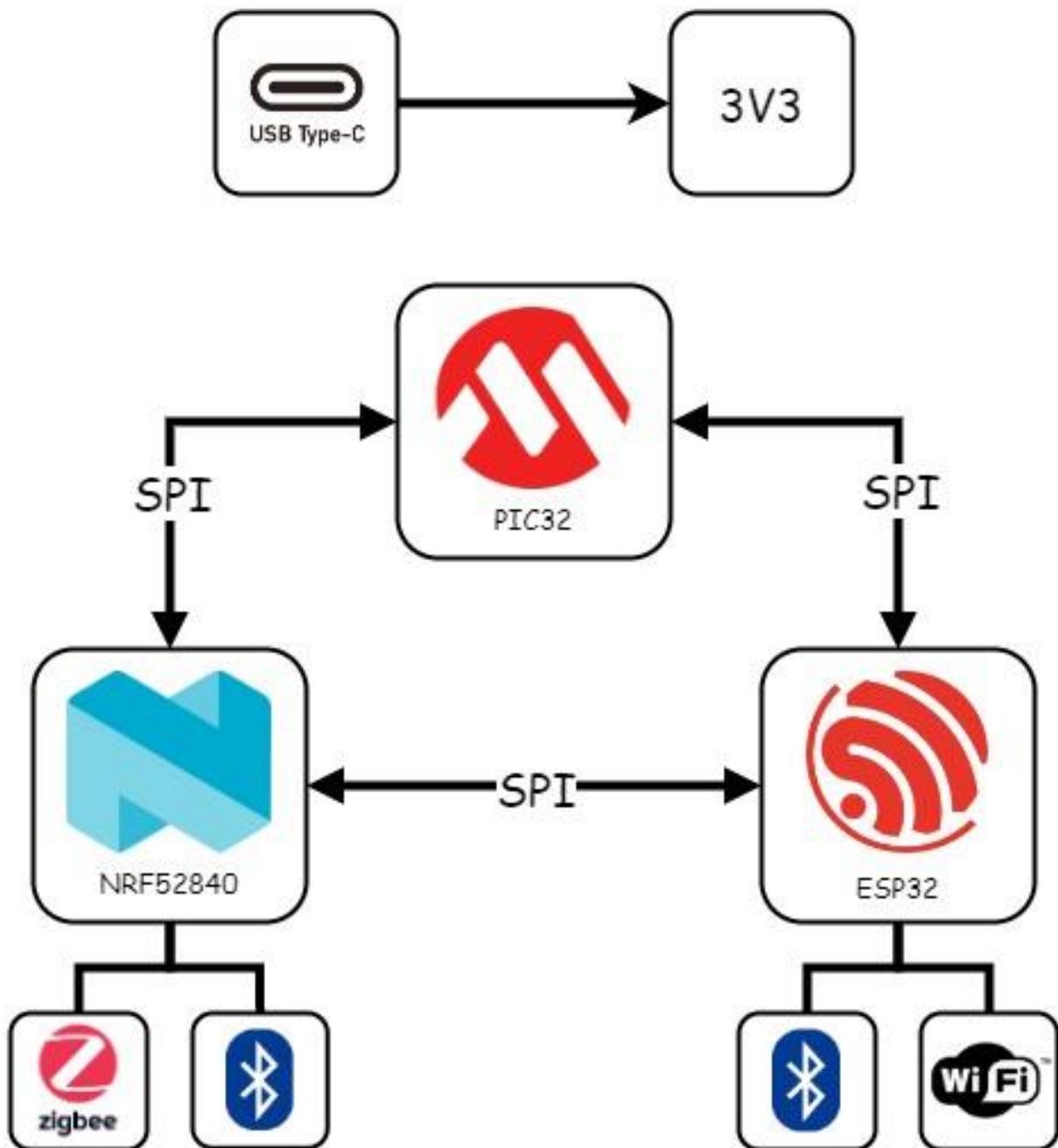
Le processeur utilisé sera le NRF52840 de Nordic Semiconductor. Pour simplifier le développement, j'ai choisi un module tout-en-un. Nordic fournit plusieurs logiciels et exemple de code pour cela.

Le module permettra la communication en Zigbee avec le récepteur. Pour le programmer, une connexion par USB à un ordinateur sera suffisante. Nordic fournit plusieurs logiciels et exemple de code pour cela.



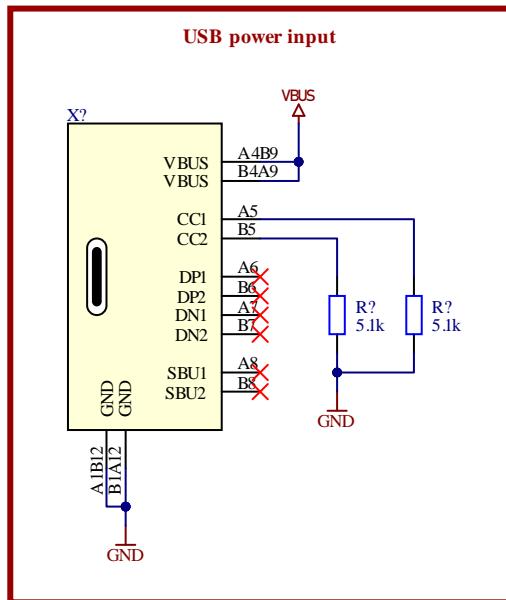
5.2. Choix des composants : Récepteur

5.2.1 Schéma bloc du récepteur



5.2.2 Connecteur USB

Le récepteur sera alimenté à l'aide de la tension de 5V fournit par un port USB-C.



5.2.3 Régulateur 3V3

Pour assurer l'alimentation des différents circuits, une conversion en 3V3 est nécessaire. Mon choix a été mis en commun avec le restant de la classe et en fonction du stock disponible à l'ES.

5.2.4 Processeurs et empreintes MIKROE

Le même module NRF52840 utilisé pour l'émetteur sera présent pour assurer la communication en Zigbee.

Des empreintes MIKROE ont été utilisé afin de permettre une certaine flexibilité dans le choix d'un module pour réaliser la connexion WIFI

Un PIC32 assurera le contrôle central entre les différents modules à l'aide de connexions SPI et UART.

6. Hardware

6.1. PCB de l'émetteur

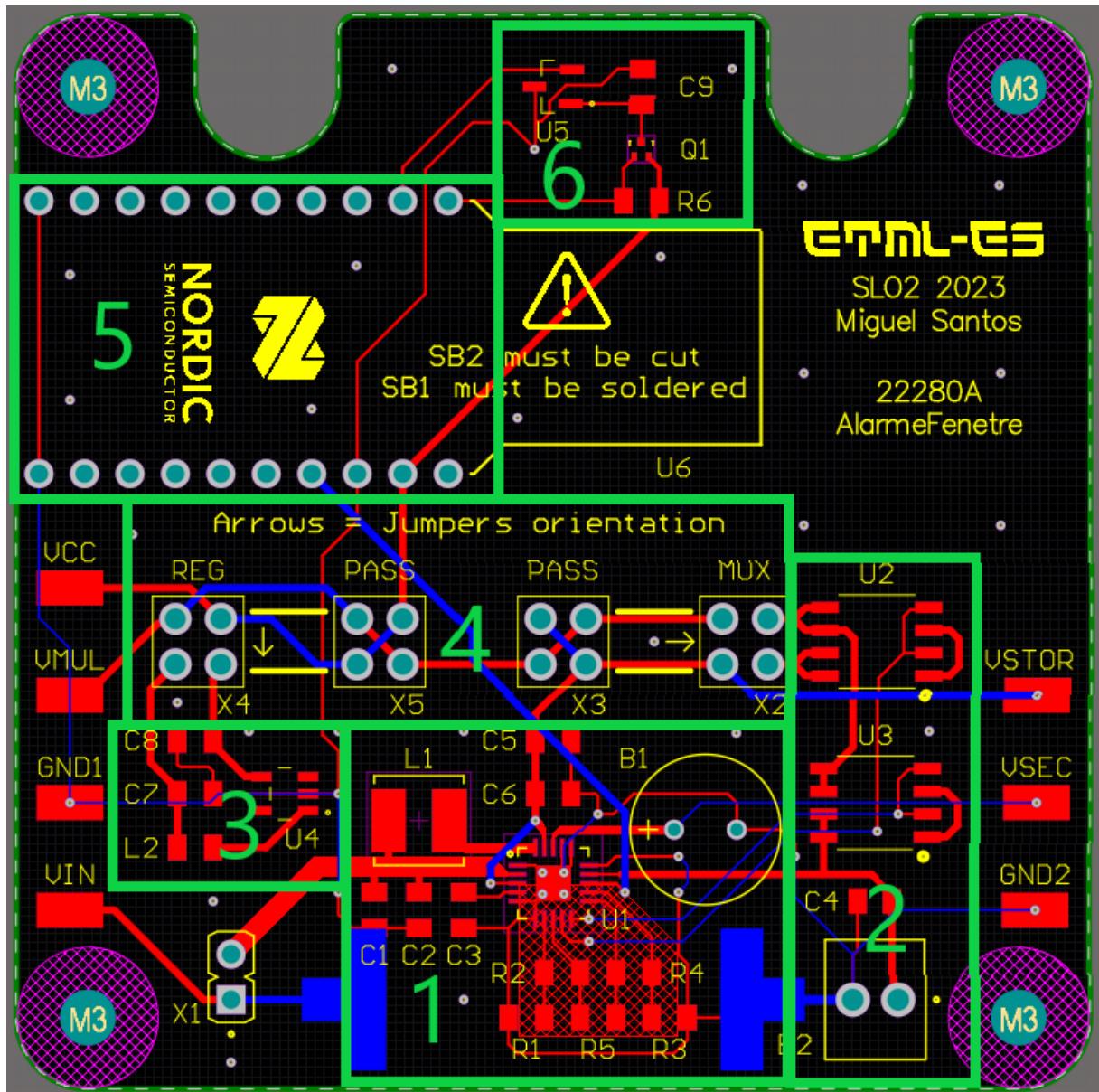


Figure 9 : PCB de l'émetteur

Les plans de masse ont été retirés de l'image pour améliorer la visibilité des pistes et des composants.

Les encoches sur le bord du PCB sont destinées à permettre le passage des colonnettes du boîtier, ce qui permet de gagner de la place. Les trous dans les coins servent à fixer le PCB solidement à l'intérieur du boîtier.

Les pointes de test ont été placées sur les bords du circuit pour faciliter l'accès avec des sondes de mesure lors des tests.

Les éléments identifiés sur l'image (Figure 9) sont décrits ci-dessous :

- 1) Panneau solaire | « Energy harvester » | Super-capacité
 - a. La piste VIN a été conçue aussi large que possible pour minimiser l'impédance. Étant donné les très faibles courants en jeu, la largeur de la piste a peu d'importance
 - b. Le panneau solaire est placé sur la couche inférieure pour qu'il puisse être le plus près possible de l'extérieur du boîtier. La forme en "T" du footprint a été spécifiquement choisie pour faciliter le processus de brassage des pads qui ont leurs connexions uniquement en dessous du composant.
 - c. Une zone de "keep-out" est placée autour des ponts résistifs, qui ont des valeurs élevées, pour éviter les impédances parasites.
 - d. Les condensateurs C1 à C3 sont placés au plus proche de l'entrée du BQ25505.
 - e. Les composants sont légèrement espacés de la bobine L1 pour éviter les perturbations causées par le boost en entrée.
- 2) Multiplexage des sources d'alimentations
 - a. Placé pour garantir le chemin le plus court entre la pile et les jumpers
- 3) Régulateur 3V3
- 4) Bypass des blocs
 - a. L'orientation de placement des jumpers est indiquée par des flèches.
 - b. Les jumpers « PASS » peuvent être placés dans n'importe quel sens
 - c. Les circuits sont protégés physiquement si jumpers placés dans le mauvais sens.
- 5) NRF52840 USB Dongle
 - a. Le fabricant recommande de ne rien placer sous la partie de l'antenne présente sur le module, c'est pour cette raison qu'il dépasser légèrement du bord du PCB.
- 6) Capteur magnétique
 - a. Placé au bord du circuit pour détecter l'aimant externe.

Les différents fichiers nécessaires à la production sont fournis en annexe.

6.2. PCB du récepteur

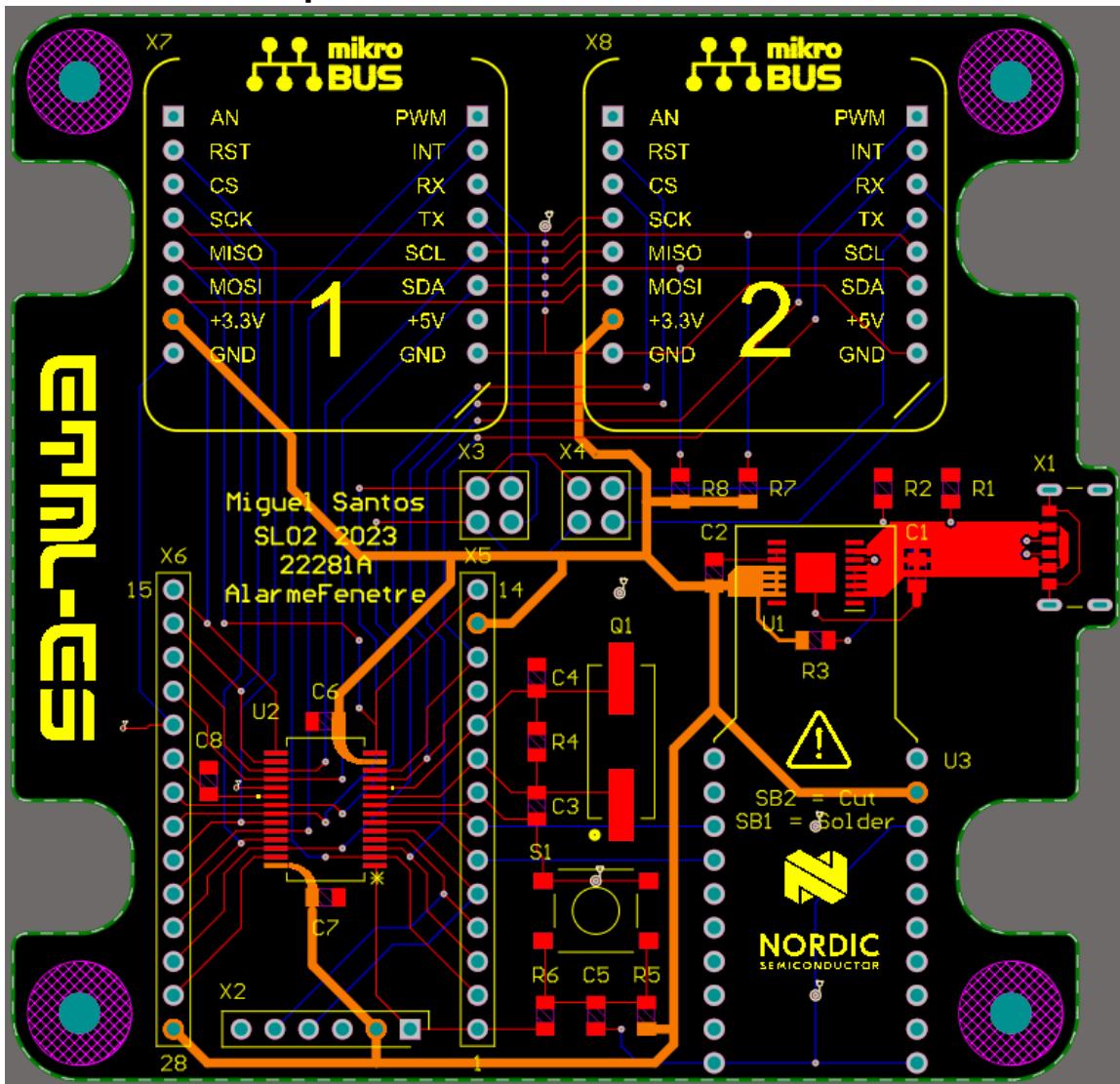


Figure 10 : PCB du récepteur

Les plans de masse ont été retirés de l'image pour améliorer la visibilité des pistes et des composants. Les pistes de couleur orange indiquent la connexion 3V3.

En entrée d'alimentation du port USB, des pistes larges ont été utilisées pour réduire l'impédance.

Des headers ont été utilisés comme pointes de test pour chaque patte du microcontrôleur.

Le NRF52840 a été positionné près du bord, de manière similaire à l'émetteur. Les empreintes mikroBUS ont été placées près du bord pour les mêmes raisons, en prévision d'une éventuelle antenne et pour laisser suffisamment d'espace, étant donné que certains modules Mikroe sont de taille assez importante.

Des encoches et des trous ont été ajoutés, de manière similaire à l'émetteur, pour permettre une fixation éventuelle dans un boîtier (le boîtier est facultatif).

Les différents fichiers nécessaires à la production sont fournis en annexe.

6.3. Boitier de l'émetteur

Un boitier composé d'un boîtier principal et d'un couvercle a été développé pour l'émetteur en utilisant SolidWorks. Il a ensuite été imprimé en 3D en PLA. Une ouverture a été réalisée pour permettre à la cellule solaire l'accès à la lumière. Un « socle » permet aussi d'y fixer un support de pile.

Les fichiers de cotations sont disponibles en annexes.

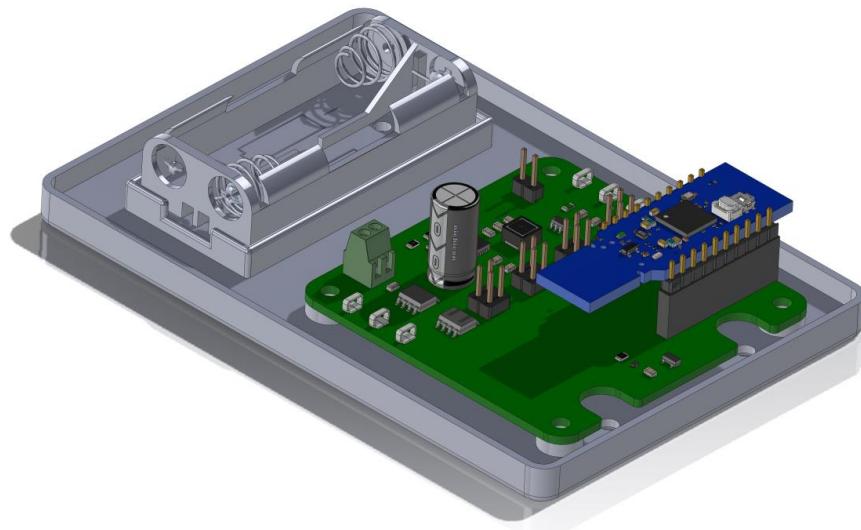


Figure 11 : Intérieur du boitier 3D



Figure 12 : Boitier vu de l'extérieur

7. Montage et mise en service

Après avoir finalisé la conception de mon circuit, j'ai procédé à la commande des différents composants et PCB. J'ai commandé deux unités pour l'émetteur et une unité pour le récepteur.

Le montage des composants a été réalisé par brasage à la main, à l'exception du BQ25505 de l'émetteur que j'ai tenté de souder au four. Malheureusement, cette méthode n'a pas donné de résultats satisfaisants en termes de soudures. J'ai essayé de reprendre les soudures à la main, mais je n'ai pas réussi à obtenir une connexion solide sur ma première carte. J'ai donc réalisé la deuxième carte entièrement à la main, ce qui a abouti à de bons résultats, comme décrit plus loin dans ce rapport.

Pour l'émetteur, il n'était pas possible de monter les composants de manière progressive, mais j'ai utilisé des jumpers pour isoler certaines parties du PCB. Après avoir terminé le montage, j'ai vérifié progressivement l'absence de court-circuit et la présence des tensions appropriées. Tout était conforme, et aucune erreur de conception n'a été détectée. Pour la carte mal soudée, la charge de la super-capacité ne fonctionnait pas, mais comme le multiplexage des sources d'alimentations fonctionne, le circuit est utilisable avec des piles.

En ce qui concerne le récepteur, j'ai d'abord monté le régulateur 3V3 pour vérifier la tension de sortie, ce qui s'est avéré conforme. Ensuite, j'ai procédé au montage du reste du PCB. Lorsque j'ai tenté d'établir une communication avec le microcontrôleur, j'ai remarqué que les broches PGEC et PGED de programmation étaient inversées. Pour résoudre ce problème, j'ai fabriqué un câble croisé pour inverser ces deux broches. Ce câble est inclus dans la boîte du projet avec les circuits.

Les modifications à apporter sont récapitulées en annexe du rapport.

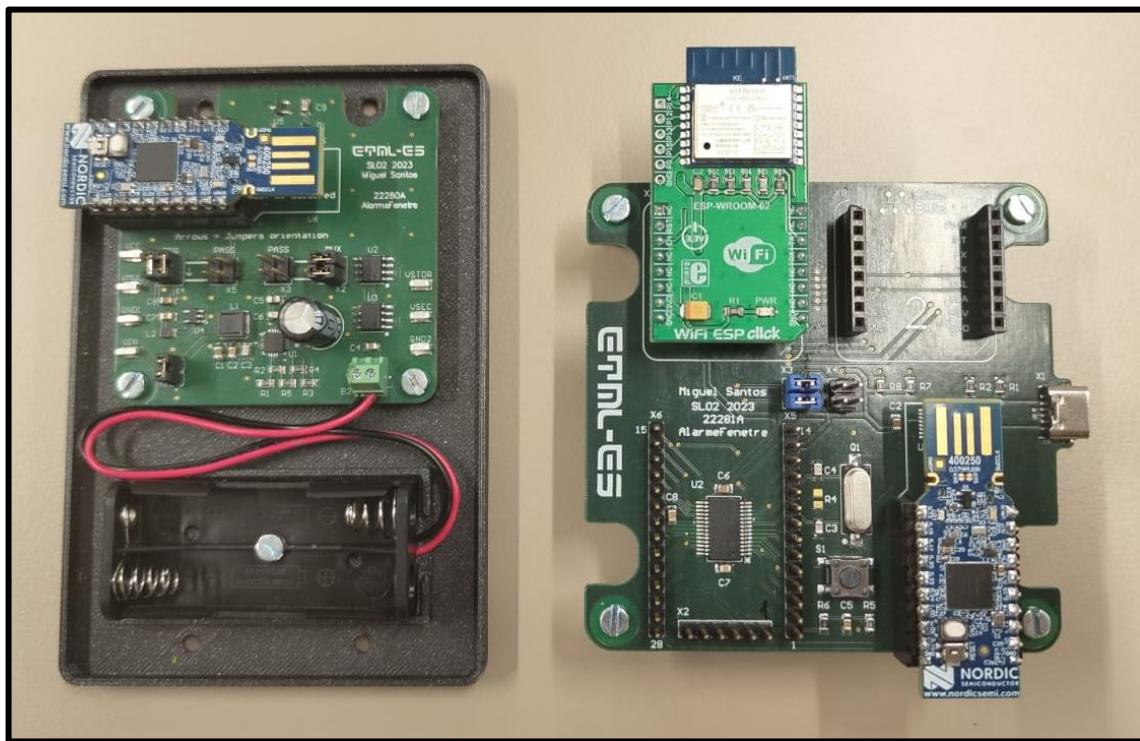


Figure 13 : Photos des circuits montés

8. Software

Dans le cadre de la programmation du NRF52840, une machine virtuelle est mise à disposition dans le dossier "soft" du projet. Cette machine virtuelle est préconfigurée avec tous les logiciels nécessaires, notamment :

- Visual Studio Code : Il s'agit d'un environnement de développement utilisé pour programmer des applications en langage C. Un plugin spécifique à Nordic Semiconductor est installé, ce qui permet d'accéder facilement à de nombreux exemples et de configurer aisément tous les aspects du NRF.
- NRF Connect : C'est un logiciel propriétaire développé par Nordic Semiconductor. Il est utilisé pour installer le NRF SDK et pour programmer le module NRF52840.

8.1. Logique de programmation du NRF52840

Le NRF52840 USB Dongle utilise une approche de programmation différente de celle du PIC32. Il est pris en charge par le projet "Zephyr", qui est un système d'exploitation temps réel (RTOS) conçu spécifiquement pour les systèmes embarqués dotés de microcontrôleurs ou de ressources limitées. Zephyr permet une gestion efficace du matériel grâce à diverses bibliothèques.

Contrairement au PIC32, le système basé sur le NRF52840 s'appuie sur trois types de fichiers principaux :

- Les fichiers "Kconfig" ou ".conf" génèrent les définitions des différentes configurations de notre système. Ces fichiers permettent de spécifier les bibliothèques, les protocoles de communication utilisés et d'autres paramètres de configuration.
- Le "Devicetree" et ses fichiers ".dts" décrivent le matériel présent sur la carte. Ces fichiers fournissent une représentation structurée des périphériques, des broches et des ressources matérielles utilisées par le système.
- Les fichiers sources ".c" et les en-têtes ".h" sont utilisés pour créer l'application spécifique. Ces fichiers contiennent le code source de l'application et les déclarations des fonctions et des structures nécessaires. Le compilateur se charge ensuite de lier ces fichiers pour générer le binaire final.

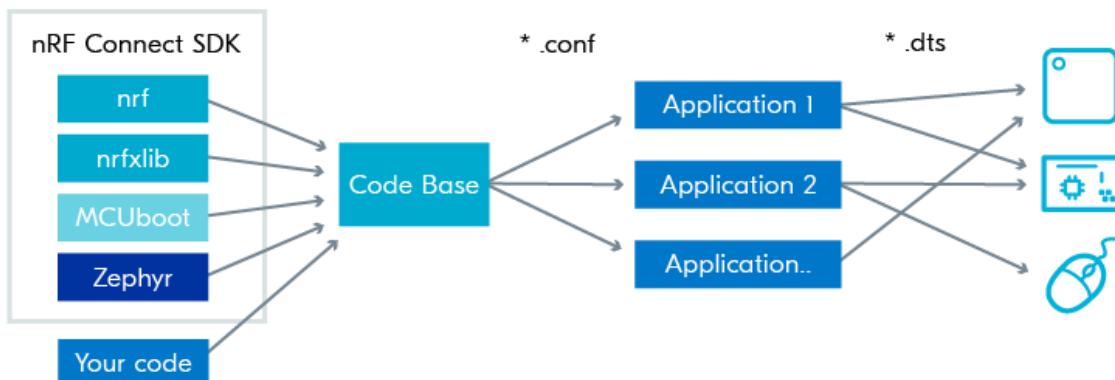


Figure 14 : NRF52840 types de fichiers

Cette méthode permet ainsi de créer une indépendance totale entre le software et le hardware. Les librairies créées peuvent être facilement utilisées sur une autre carte.

8.2. Software émetteur : NRF52840

Le code de l'émetteur est conçu de manière assez simple. Son fonctionnement repose sur la récupération de la valeur du capteur à effet Hall après avoir reçu une interruption via le protocole Zigbee, puis il renvoie cette valeur en utilisant le même protocole.

Cependant, durant le développement, les protocoles de communication se sont avérés plus complexes que prévu. J'ai alors envisagé d'utiliser le Bluetooth, mais j'ai rencontré des difficultés similaires. Les défis auxquels j'ai été confronté comprenaient des problèmes de compilation du code et de compréhension. J'ai tenté d'effectuer des recherches approfondies et étudier attentivement la documentation des protocoles de communication, malheureusement sans succès.

Un listing du fichier main.c est fourni en annexe. J'ai réalisé certains tests avec les LEDS et interruptions, qui sont détaillés dans la section de test de fonctionnement. Voici le schéma bloc du fonctionnement théorique :

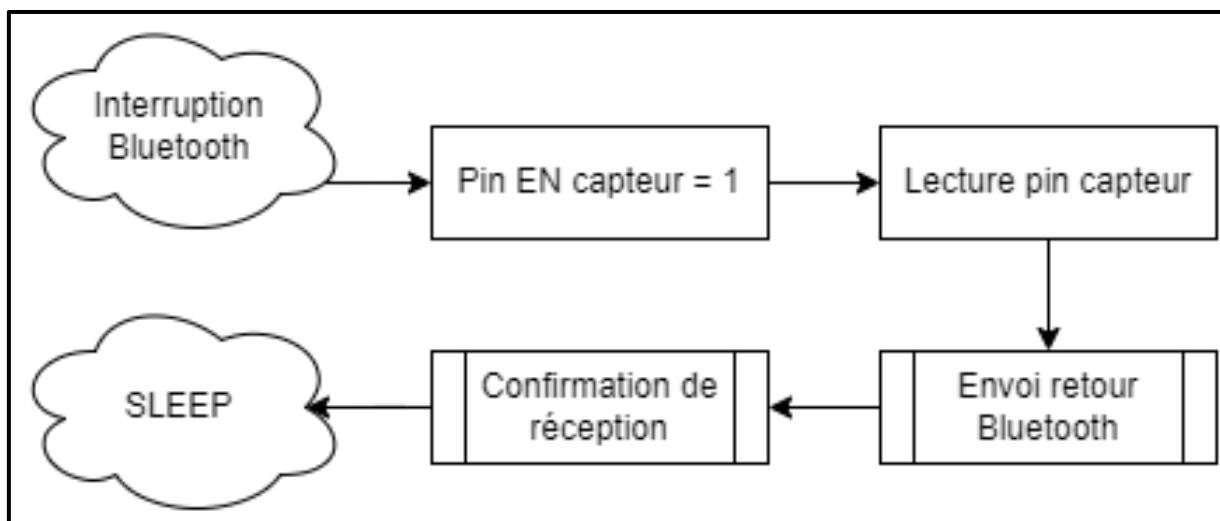


Figure 15 : Schéma-bloc du NRF de l'émetteur

8.3. Software récepteur : NRF52840

Le module NRF utilisé sur le récepteur a présenté des problèmes similaires à ceux rencontrés avec l'émetteur. Un listing du fichier main.c est fourni en annexe.

J'ai effectué quelques tests en utilisant les LED, les interruptions et l'UART, qui sont détaillés dans la section de test de fonctionnement. Pour mieux comprendre le fonctionnement théorique, voici le schéma bloc :

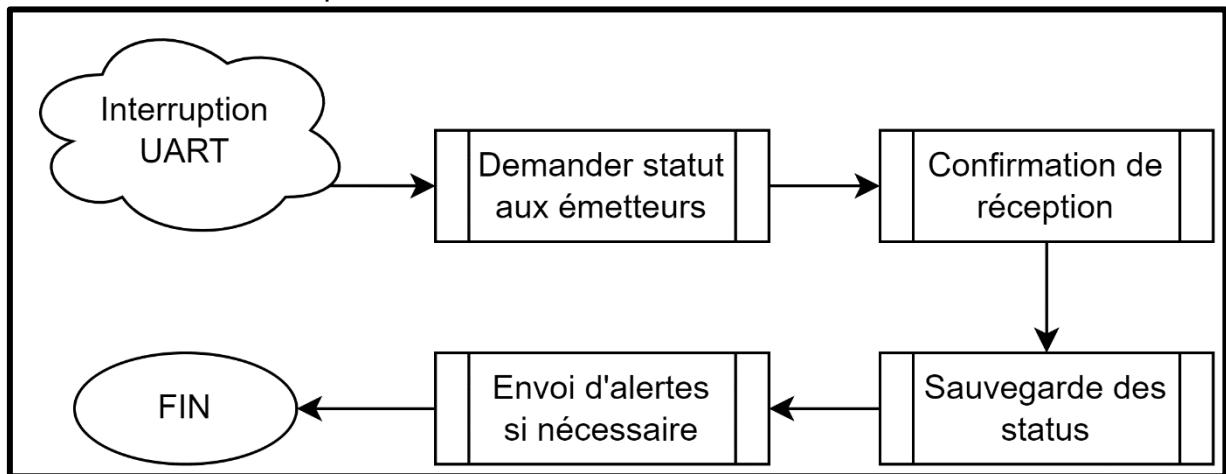


Figure 16 : Schéma bloc du NRF du récepteur

8.4. Software récepteur : PIC32MX130

Le but du PIC32 est d'assurer la communication entre le module NRF et l'éventuelle module de communication Wifi ou SMS. Grâce à la RTCC intégré au PIC32, une alerte peut être programmé à une heure précise pour récupérer l'état des émetteurs et lever une alerte.

Actuellement, seule une application de test de communication en UART a été réalisé. L'UART a été configuré pour correspondre aux paramètres de celui du NRF.

Voici la méthode qui devrait être théoriquement développé :

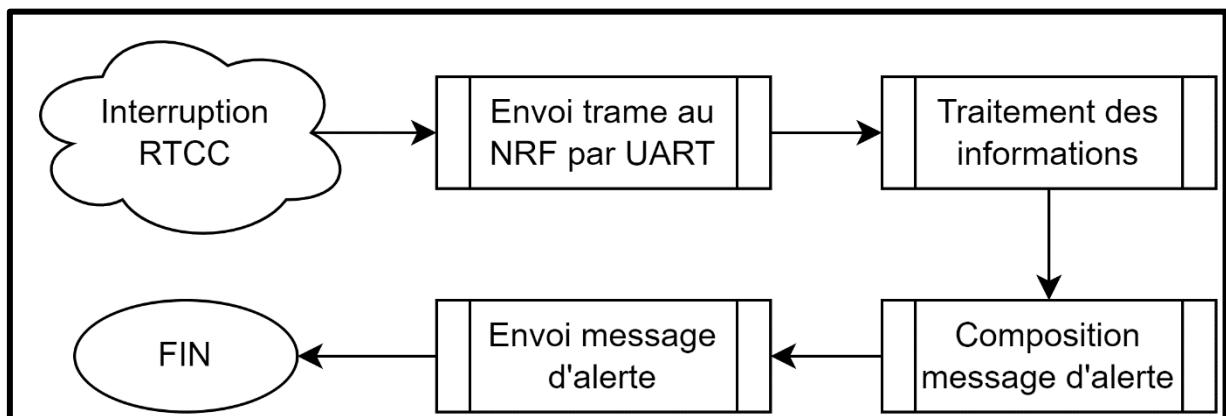


Figure 17 : Schéma bloc du PIC32 du récepteur

Voici comment a été configuré l'UART sur le PIC32 :

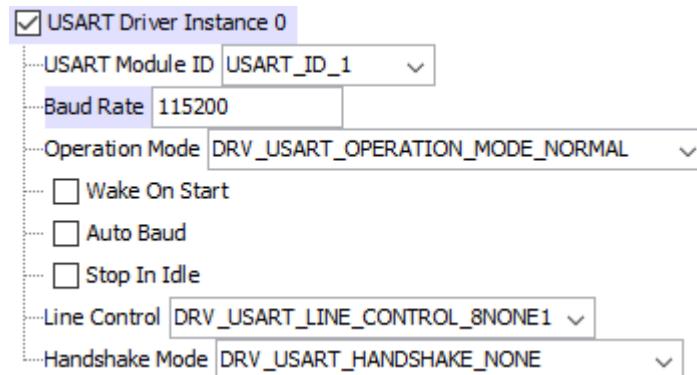


Figure 18 : Configuration UART sur le PIC32

Grâce à Harmony sur MPLAB, j'ai réalisé la configuration des pins du PIC32 en me basant sur le schéma électrique du récepteur.

Pin Number	Pin ID	Voltage Tolerance	Name	Function
1	MCLR	5V		
2	RA0		OC1	OC1
3	RA1		OC2	OC2
4	RB0			Available
5	RB1			Available
6	RB2		U1RX	U1RX
7	RB3		U1TX	U1TX
8	VSS			
9	RA2		OSC1	OSC1
10	RA3		OSC2	OSC2
11	RB4		RST2	GPIO_OUT
12	RA4		INT2	GPIO_OUT
13	VDD			
14	RB5	5V	SDI1	SDI1
15	RB6	5V	SDO1	SDO1
16	RB7	5V	CS1	GPIO_OUT
17	RB8	5V	SCL1	SCL1
18	RB9	5V	SDA1	SDA1
19	VSS			
20	VCAP			
21	RB10	5V	U2TX	U2TX
22	RB11	5V	U2RX	U2RX
23	RB12		RST1	GPIO_OUT
24	RB13		INT1	GPIO_OUT
25	RB14		SCK1	SCK1
26	RB15		CS2	GPIO_OUT
27	AVSS			
28	AVDD			

Figure 19 : Configuration des pins du PIC32

8.5. Tests de fonctionnement

8.5.1 Seuils de tension de la super-capacité

J'ai effectué des mesures pour évaluer le fonctionnement des seuils de sous-tension et de surtension de la super-capacité. Pour ce faire, j'ai développé un programme qui active toutes les LED sur le module NRF afin de consommer le maximum d'énergie. J'ai rechargé manuellement la super-capacité à l'aide d'une alimentation externe pour accélérer le processus. J'ai ensuite mesuré le signal "VBAT_OK" qui indique si la tension de la batterie se situe dans la plage acceptable, en prenant en compte l'effet d'hystérésis.

Channel	Signal
C1	VSEC
C2	VBAT_OK

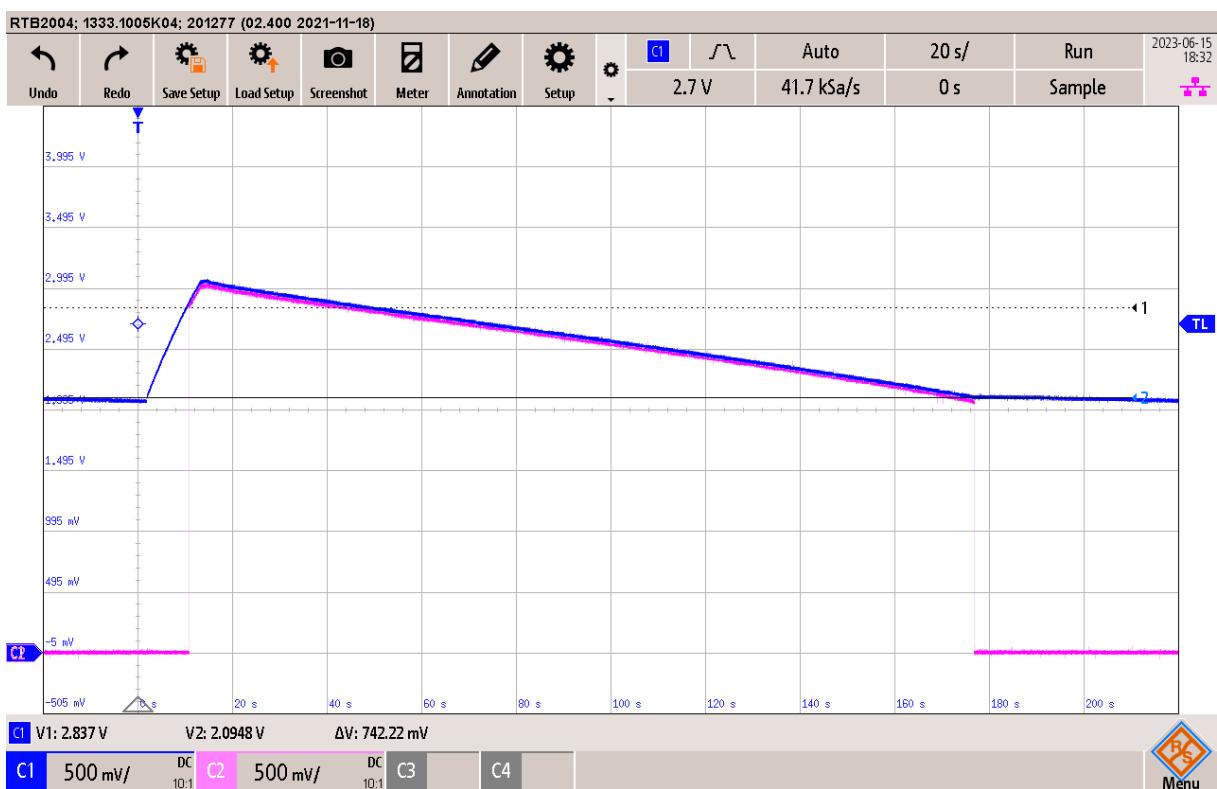


Figure 20 : Mesure de tests des seuils de tensions

Pendant la phase de conception, j'ai dimensionné les résistances pour obtenir des valeurs spécifiques. Les valeurs obtenues sont bien celles qui étaient attendues :

Nom	Valeur théorique	Valeur mesurée	Différence rel.
VBAT_OK	2,1 V	2,09 V	-0,47 %
VBAT_OK_HYST	2,9 V	2,83 V	-2,47 %

La plage de tension de fonctionnement a été légèrement réduite, ce qui entraîne une diminution de l'autonomie de la super-capacité. Cependant, dans ce cas précis, cette réduction est négligeable et n'affecte pas de manière significative les performances globales du système.

8.5.2 Fonctionnement du capteur à effet hall

Afin de tester le fonctionnement du capteur à effet Hall, j'ai développé un programme qui allume une LED lorsqu'un signal est détecté par le capteur. Cela permet de vérifier que le microcontrôleur réagit correctement aux signaux du capteur. Pour effectuer ce test, le capteur était maintenu en marche en continu.

Channel	Signal
C1	MAG_OUT
C2	LED0

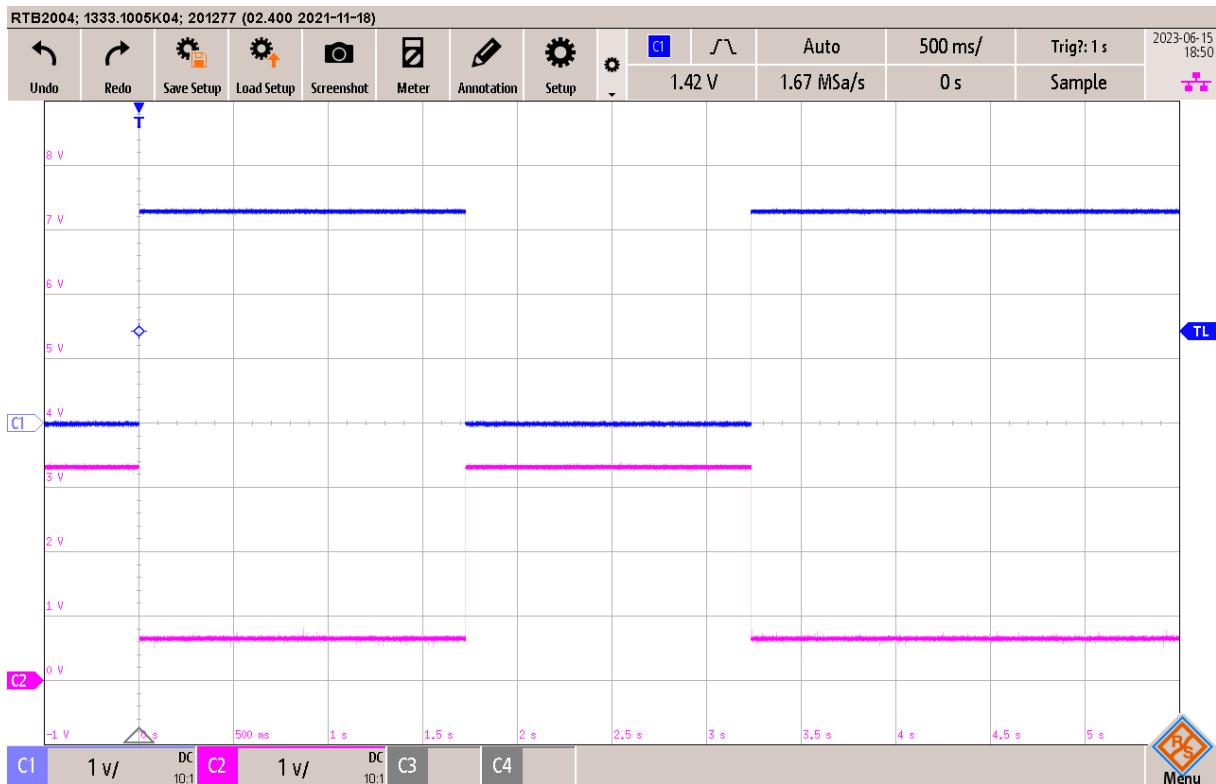


Figure 21 : Mesure de test du capteur à effet Hall

En approchant un aimant du capteur à effet Hall, le capteur réagit en tirant le signal à la masse, ce qui entraîne l'allumage de la LED comme prévu. Cette observation confirme le bon fonctionnement du capteur. Cependant, il convient de noter que l'aimant initialement commandé, tel que spécifié dans la liste des composants (BOM), s'est avéré être insuffisamment puissant. Pour contourner ce problème, j'ai pu utiliser un aimant disponible dans notre classe qui satisfait les besoins du test.

8.5.3 Communication UART

Pour évaluer le fonctionnement de la communication UART entre le PIC32 et le module NRF, j'ai développé un programme sur le module NRF qui réagit à la réception d'une trame spécifique (0x0F) en alternant l'état d'une LED. Cette approche me permet de tester à la fois l'interruption UART et le contrôle des données.

Channel	Signal
C1	(PIC32) TX1
C2	(NRF) LED2

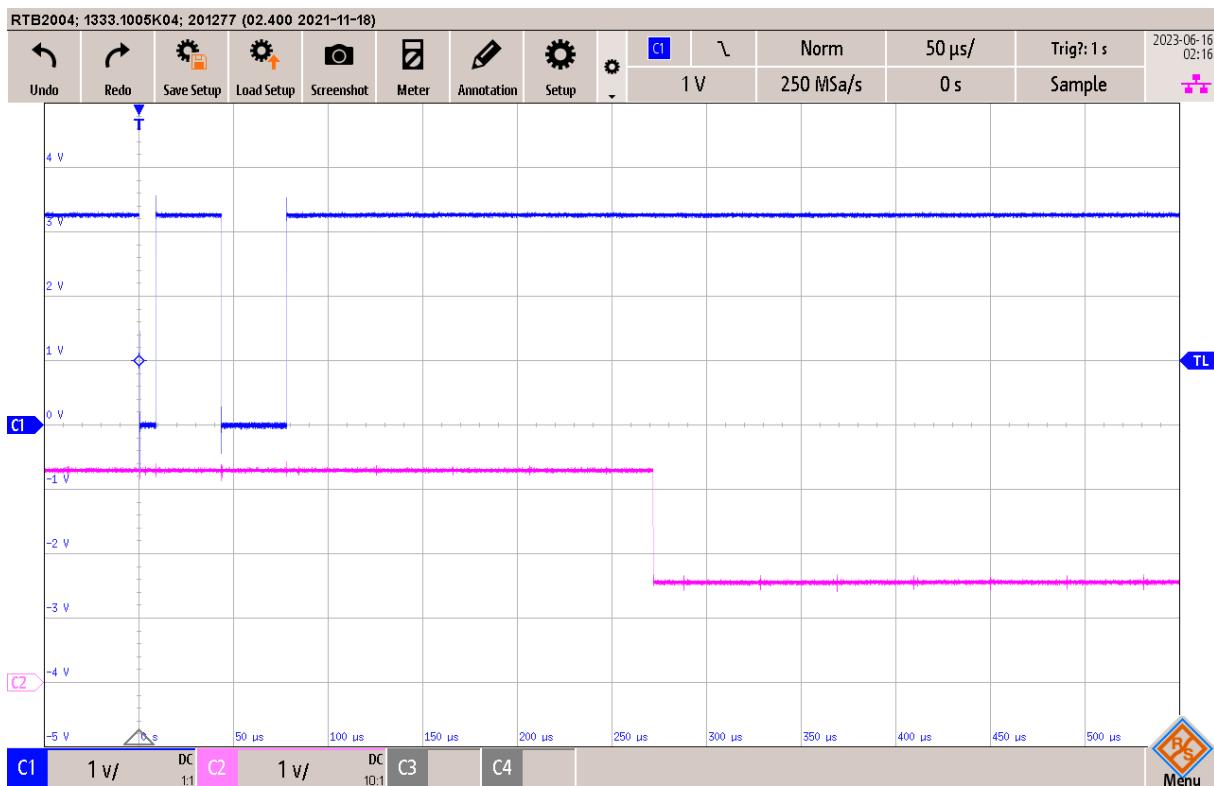


Figure 22 : Mesure de test de l'UART

L'observation de cette approche permet de conclure que la communication entre le PIC32 et le module NRF via l'UART fonctionne correctement. Cela indique également que l'UART a été configuré de manière adéquate sur le module NRF. Les tests effectués ont démontré une réception précise de la trame spécifique (0x0F) par le module NRF, déclenchant ainsi l'interruption UART et contrôlant avec succès l'état de la LED correspondante.

9. État final

Le projet actuel présente un état encourageant au niveau du hardware, car celui-ci est opérationnel dans l'ensemble. Cependant, il y a quelques points qui nécessitent une attention particulière pour améliorer son fonctionnement.

Tout d'abord, la première carte de l'émetteur requiert un changement du circuit BQ25505 afin de rendre opérationnelle la charge de la super capacité.

De plus, une légère modification au niveau du port de programmation du récepteur est nécessaire pour éviter de recourir à un câble croisé.

Cependant, un défi majeur se présente au niveau du logiciel. En effet, il y a une certaine lacune dans l'implémentation des protocoles de communication. Plus précisément, il manque le protocole WiFi sur le module ESP Mikroe, qui limite les possibilités de connectivité sans fil du projet.

De plus, il est également nécessaire d'ajouter le protocole Bluetooth ou idéalement Zigbee sur le module NRF pour offrir des options de communication entre les différents émetteur et récepteurs.

En résumé, bien que le matériel du projet soit en grande partie fonctionnel, des améliorations sont nécessaires pour parvenir à une pleine opérationnalité. Les priorités actuelles incluent les modifications hardware et l'ajout des protocoles WiFi, Bluetooth ou Zigbee pour renforcer les capacités de communication du système.

10. Conclusion

En conclusion, ce projet s'est révélé être un défi complexe et stimulant, malgré son apparence initialement simple. Il s'est articulé autour de deux aspects clés qui étaient la gestion de l'énergie et la communication sans fil via des protocoles spécifiques.

Du côté matériel, j'ai obtenu des résultats encourageants en ce qui concerne le fonctionnement du panneau solaire, la réussite de la charge de la super-capacité, ainsi que la mise en place du mécanisme de transition automatique entre la super-capacité et les piles.

Cependant, j'ai fait preuve d'une ambition excessive en choisissant le module NRF52840, ce qui a été une tâche plus ardue que prévu. J'ai sous-estimé les défis associés à l'apprentissage d'un nouveau microcontrôleur, surtout lorsque celui-ci présente une logique fondamentalement différente de celle à laquelle j'étais habitué. Cela a nécessité un investissement en temps considérable pour me familiariser avec ses fonctionnalités et comprendre son fonctionnement. Malheureusement, cela a entraîné des retards considérables dans l'avancement de mon projet, au point de ne pas pouvoir atteindre les parties relatives aux protocoles de communication. Mon choix ambitieux du module NRF52840 a nécessité un investissement de temps important pour en comprendre les fonctionnalités et le fonctionnement, ce qui a finalement eu un impact négatif sur la réalisation de ces aspects essentiels du projet.

Cette expérience m'a apporté une précieuse leçon sur l'importance de l'évaluation minutieuse des exigences et des compétences nécessaires avant d'entreprendre de nouveaux projets. À l'avenir, je serai plus réaliste dans mes ambitions et j'accorderai une attention particulière à l'analyse approfondie des spécifications techniques et des challenges associés.

Dans l'ensemble, ce projet m'a permis de développer mes compétences techniques, ma résilience face aux obstacles et ma capacité d'adaptation à de nouvelles technologies. Je suis convaincu que les enseignements tirés de cette expérience me seront précieux pour mes projets futurs, en m'a aidant à prendre des décisions plus éclairées et à atteindre des résultats encore plus remarquables.

Lausanne, le 16 juin 2023

Miguel Santos

11. Annexes

- 11.1. Résumé du projet**
- 11.2. Affiche de projet**
- 11.3. Cahier des charges**
- 11.4. Fiche de modification**
- 11.5. Planning**
- 11.6. Journal de travail**
- 11.7. Émetteur : Schéma électrique**
- 11.8. Émetteur : BOM**
- 11.9. Émetteur : Schéma d'implémentation**
- 11.10. Émetteur : Schéma mécanique**
- 11.11. Récepteur : Schéma électrique**
- 11.12. Récepteur : BOM**
- 11.13. Récepteur : Schéma d'implémentation**
- 11.14. Récepteur : Schéma mécanique**
- 11.15. Boitier : Cotation « Top »**
- 11.16. Boitier : Cotation « Bottom »**
- 11.17. Software : NRF52840 Emetteur**
- 11.18. Software : NRF52840 Récepteur**

RESUMÉ – Projet

Miguel Santos
SLO
2023

Titre :

2228 Alarme pour fenêtre ouverte

Contexte et objectifs :

Le projet consiste à développer un dispositif capable de détecter si une fenêtre est restée ouverte pendant certaines heures dans une salle spécifique de l'ETML-ES. L'objectif est d'envoyer une alerte par e-mail dans le cas où une fenêtre reste ouverte. Ce projet s'inspire des solutions domotiques disponibles sur le marché.

Le système sera composé de plusieurs émetteurs, placés à chaque fenêtre à surveiller, et d'un récepteur unique, installé dans la même salle que les fenêtres. Les émetteurs seront responsables de détecter l'état d'ouverture/fermeture des fenêtres à l'aide d'un capteur magnétique et d'un aimant et de transmettre cette information au récepteur.

Résultats obtenus et conclusion :

Le projet actuel présente un état encourageant au niveau du hardware, car celui-ci est opérationnel dans l'ensemble. Cependant, il y a quelques points qui nécessitent une attention particulière pour améliorer son fonctionnement.

nécessitent une attention particulière pour améliorer leur fonctionnement. En effet, il y a une certaine lacune dans l'implémentation des protocoles de communication. Plus précisément, il manque le protocole WiFi sur le module ESP Mikroe, qui limite les possibilités de connectivité sans fil du projet.

Cette expérience m'a apporté une précieuse leçon sur l'importance de l'évaluation minutieuse des exigences et des compétences nécessaires avant d'entreprendre de nouveaux projets. À l'avenir, je serai plus réaliste dans mes ambitions et j'accorderai une attention particulière à l'analyse approfondie des spécifications techniques et des challenges associés.

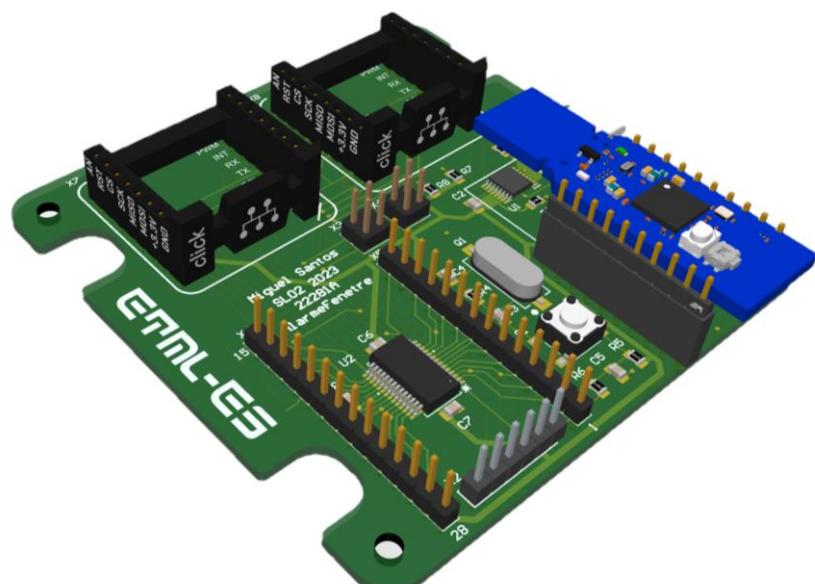
Dans l'ensemble, ce projet m'a permis de développer mes compétences techniques, ma résilience face aux obstacles et ma capacité d'adaptation à de nouvelles technologies. Je suis convaincu que les enseignements tirés de cette expérience me seront précieux pour mes projets futurs, en m'a aidant à prendre des décisions plus éclairées et à atteindre des résultats encore plus remarquables.



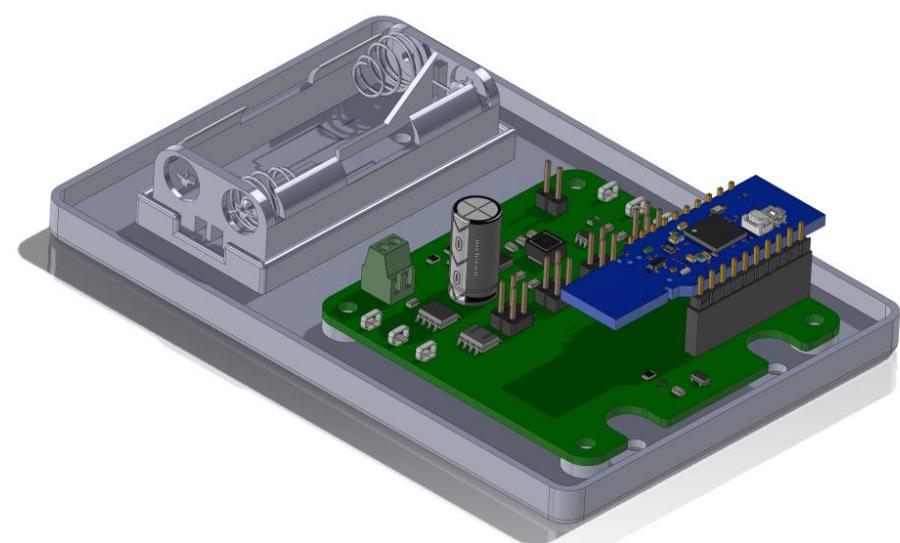
Alarme de fenêtre ouverte

ETML-ES

Le projet consiste à développer un dispositif capable de détecter si une fenêtre est restée ouverte pendant certaines heures dans une salle spécifique de l'ETML-ES. L'objectif est d'envoyer une alerte par e-mail dans le cas où une fenêtre reste ouverte. Ce projet s'inspire des solutions domotiques disponibles sur le marché. Le système sera composé de plusieurs émetteurs, placés à chaque fenêtre à surveiller, et d'un récepteur unique, installé dans la même salle que les fenêtres. Les émetteurs seront responsables de détecter l'état d'ouverture/fermeture des fenêtres à l'aide d'un capteur magnétique et d'un aimant et de transmettre cette information au récepteur.



Récepteur



Émetteur

Cette expérience m'a apporté une précieuse leçon sur l'importance de l'évaluation minutieuse des exigences et des compétences nécessaires avant d'entreprendre de nouveaux projets. À l'avenir, je serai plus réaliste dans mes ambitions et j'accorderai une attention particulière à l'analyse approfondie des spécifications techniques et des challenges associés.

Dans l'ensemble, ce projet m'a permis de développer mes compétences techniques, ma résilience face aux obstacles et ma capacité d'adaptation à de nouvelles technologies. Je suis convaincu que les enseignements tirés de cette expérience me seront précieux pour mes projets futurs, en m'a aidant à prendre des décisions plus éclairées et à atteindre des résultats encore plus remarquables.

Projet ETML-ES - Cahier des charges

Alarme de fenêtre ouverte

2228

Entreprise/Client :	ETML-ES	Département :	SLO
Demandé par (Prénom, Nom) :	Serge Castoldi	Date :	17.11.2022

Auteur (ETML-ES) :	Miguel Santos	Filière :	SLO
		Date :	17.11.2022

1 But du projet

Réalisation d'un dispositif permettant de détecter si une fenêtre est restée ouverte à certaines heures dans une salle de l'ETML-ES. Le cas échéant, une alerte sera envoyée par e-mail.

Ce projet sera notamment inspiré par des produits déjà disponibles sur le marché de la domotique.



Figure 1 : Capteur de détection de l'ouverture d'une fenêtre

2 Spécifications du projet

Pour la réalisation de ce projet, une grande liberté a été donné sur le cahier des charges. Les seules conditions à respecter sont les suivantes :

- Fonctionnement sur piles pendant 1 année.
- Récupération de l'état des fenêtres (ouvertes / fermées) à des heures précises.
- Envoi d'un avertissement (sms ou email) à un responsable déterminé.

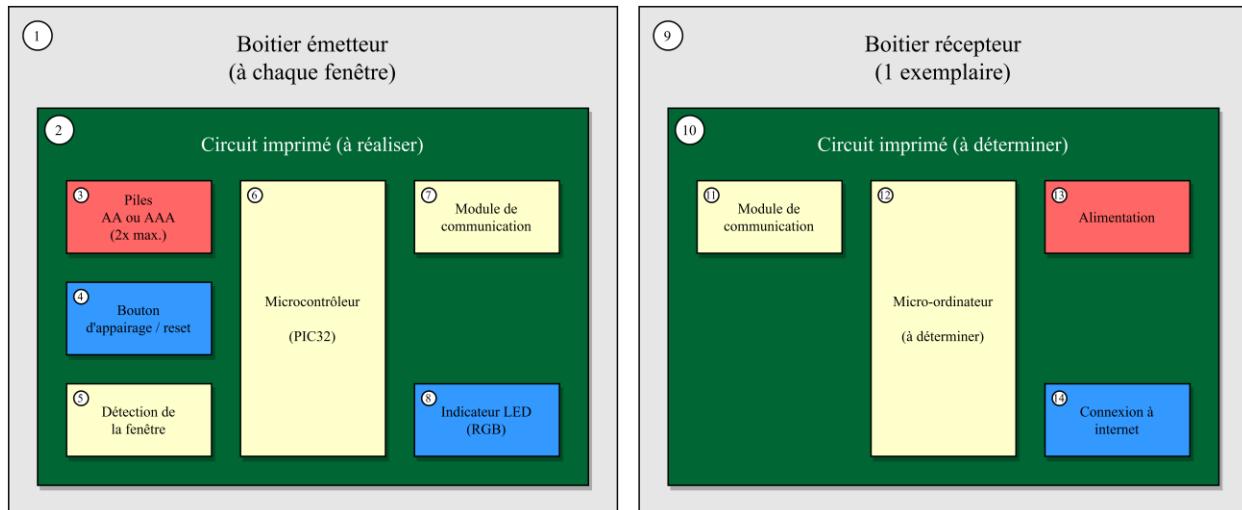


Figure 2 : Schéma général du système

Commentaires :

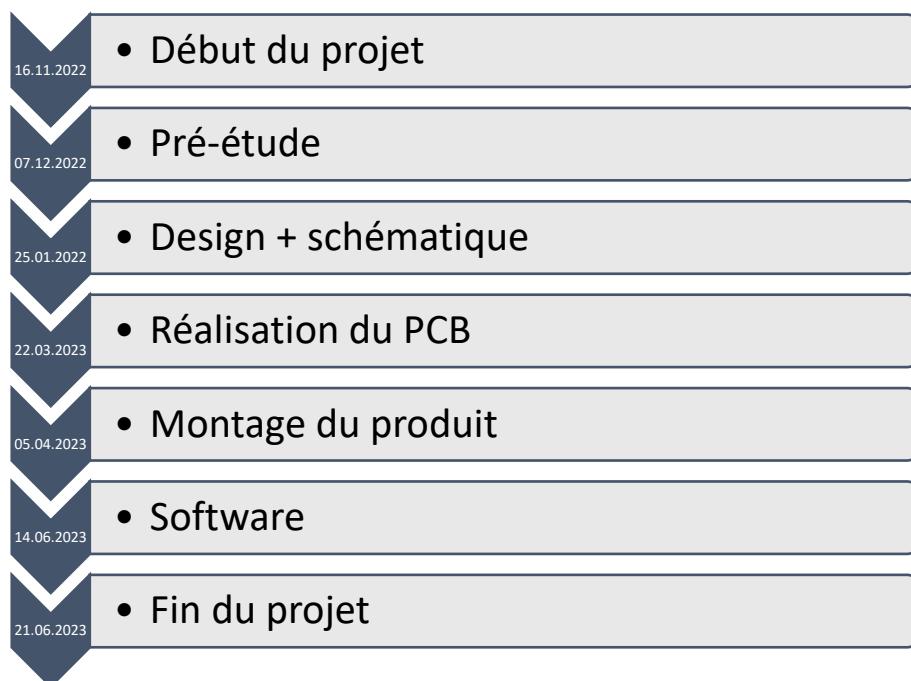
- [1 ; 9] Imprimé en 3D ou préfabriqué et déterminer méthode de fixation.
- [4] Accès à l'extérieur du boîtier.
- [5] Fenêtre ouverte ou fermée.
- [6] PIC32 exigé par l'ES.
- [7 ; 11] Protocole de communication à déterminer.
- [12] Raspberry-Pi privilégié, en fonction des stocks disponibles.
- [13] Alimentation sur secteur

Figure 3 : Commentaires sur le schéma général du système

3 Tâches à réaliser

- Déterminer le protocole de communication à utiliser
- Choix du microcontrôleur
- Choix de la méthode de détection de la fenêtre
- Réalisation du PCB de l'émetteur
- (Réalisation du boîtier de l'émetteur)

4 Jalons principaux



5 Livrables

- Les fichiers sources de CAO électronique des PCB réalisés
- Tout le nécessaire à fabriquer un exemplaire hardware de chaque :
- Fichiers de fabrication (GERBER) / liste de pièces avec références pour commande / implantation (prototype) / modifications / dessins mécaniques, etc
- Les fichiers sources de programmation microcontrôleur (.c / .h)
- Tout le nécessaire pour programmer les microcontrôleurs (logiciel ou fichier .hex)
- Le cas échéant, les fichiers sources de programmation PC/Windows/Linux.
- Le cas échéant, tout le nécessaire à l'installation de programmes sur PC/Windows/Linux.
- Un mode d'emploi du système
- Un calcul / estimation des coûts
- Un rapport contenant les calculs - dimensionnement de composants - structogramme, etc.

Projet ETML-ES – Modification

PROJET:	Alarme pour fenêtre ouverte		
Entreprise/Client:	Serge Castoldi	Département:	SLO
Demandé par (Prénom, Nom):	-	Date:	16.06.23
Objet (No ou réf, pièce, PCB...)	PCB Récepteur		
Version à modifier:	1		

Auteur (ETML-ES):		Filière:	SLO
Nouvelle version:		Date:	05.12.2018

1 Description ou justification

Corrections de quelques erreurs de conception du PCB.

2 Référence conception

Projet «Altium 2228_AlarmeFenetre_Recepteur» disponible sous :

*\2228_AlarmeFenetreOuverte\hard\2228_Altium\2228_AlarmeFenetre_Recepteur

3 Détail des modifications

#	Description	Fait	Approuvé
1	Intervertir les pins PGEC et PGED sur le port de programmation.	NOK	
2	Rajouté indications dans quelle direction doivent être mis les jumpers pour le multiplexage des UART (Voir exemple sur le projet de l'émetteur)	NOK	

4 Remarques

Pour contourner l'erreur sur le port de programmation, les deux fils concernés ont été croisés directement sur le câble qui est fournis dans la boîte du projet.

ETNL-ES	Planning de projet																									
	N° de projet :		2228						Auteur :		Miguel Santos															
	Nom du projet :		Alarme pour fenêtre ouverte						Année :		2022-2023															

No de semaine projet	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28					
Date	16.11.2022	23.11.2022	30.11.2022	07.12.2022	14.12.2022	21.12.2022	28.12.2022	04.01.2023	11.01.2023	18.01.2023	25.01.2023	01.02.2023	08.02.2023	15.02.2023	22.02.2023	01.03.2023	08.03.2023	15.03.2023	22.03.2023	29.03.2023	05.04.2023	12.04.2023	19.04.2023	26.04.2023	03.05.2023	10.05.2023	17.05.2023	24.05.2023	31.05.2023	07.06.2023	14.06.2023	21.06.2023	28.06.2023
Tâches				R P							R P																						
Pré-étude																																	
Design + schéma																																	
PCB																																	
Montage																																	
Software																																	
Test et mise au point																																	
Rédaction du rapport																																	
Préparation présentation + demo																																	
Présentations finales																																	
Finalisation/Corrections/Documentation																																	

 = Tâches effectives



Journal de travail

2228

AlarmeFenetreOuverte

15.06.2023

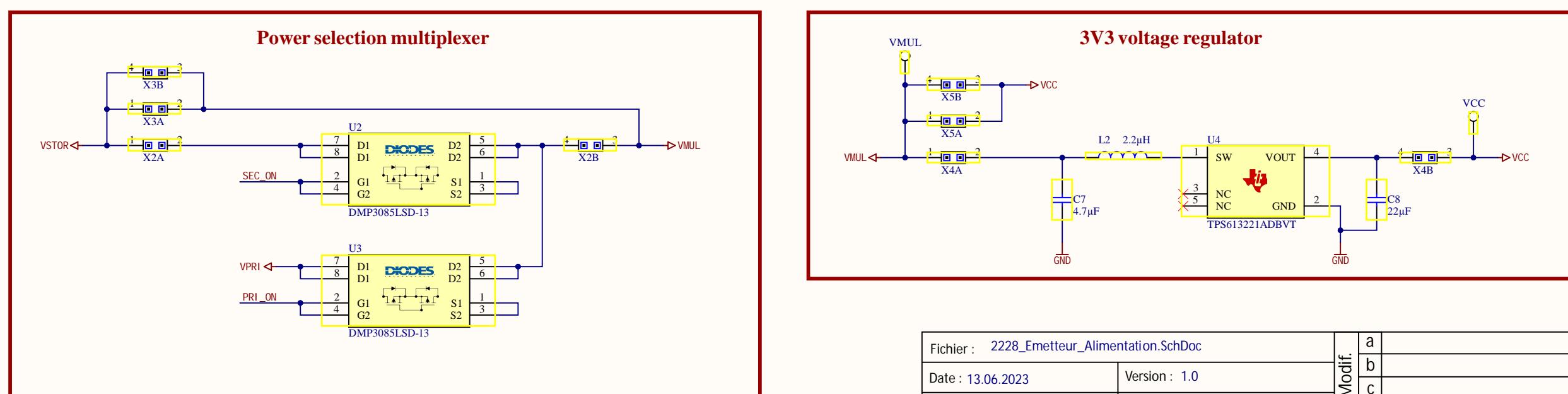
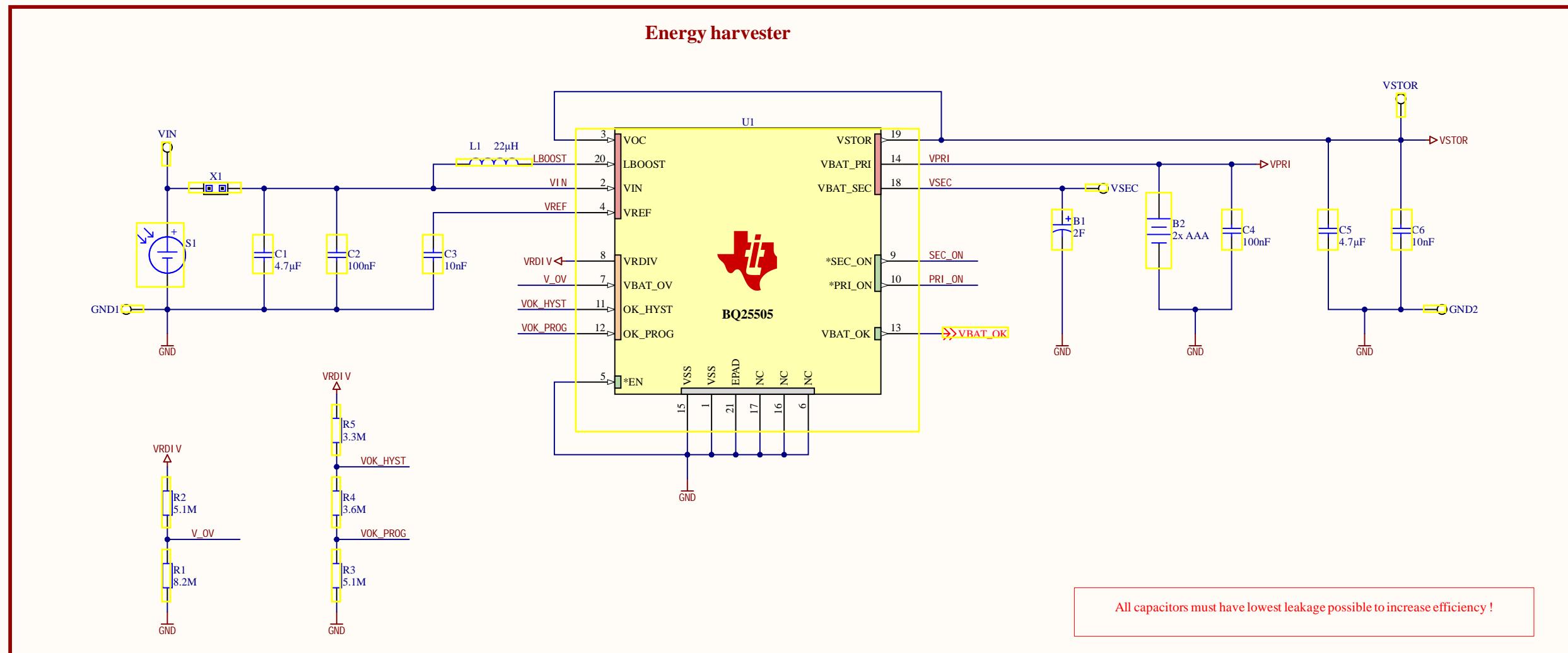
Miguel Santos

Date	Étape	Tâches réalisées	Tâches à faire
01.02.2023	Design	Configuration de l'active BOM	Ajout des footprints manquants
		Ajout des capacités et résistances dans la BOM	Changement du régulateur 3V3
		Changement du connecteur USB	
		Ajout des fabricants et MPN à la BOM	
		Calculs des ponts résistifs	
05.02.2023	Design	Correction erreur USB (schéma + footprint)	Vérifier les disponibilités en stock
		Ajout de différents composants à la BOM	Demander pour courants de fuite sur le mosfet
		MAJ Footprint du multiplexer	
		Etude de l'application note du BQ25505	
08.02.2023	Design	Changement des valeurs de résistances pour du standard	Confirmer le footprint du BQ25505 (pad thermique)
		Footprint panneau solaire + 3D	
		Footprint BQ25505	
		Réorganisation du schéma - Séparation en deux schémas - Merge des blocs pour plus de lisibilité - Ajout des jumpers de bypass	
22.02.2023	Design	Footprints des composants passifs	Sérigraphie jumpers
		Footprint NRF52840 + 3D	Vérification pins NRF
01.03.2023	Design	Finitions des footprints	
		Recherche du boitier	
02.03.2023	Design	Recherche d'un boitier - Boitier 3D décidé !	Finir le schéma du récepteur
			Modification footprint BQ25505
04.03.2023	Réalisation	Placement des composants sur le PCB de l'émetteur	Vérifier le bornier de piles
		Backup du projet !	Trous de passage des colonettes du boitier
			Sérigraphie info du projet
			Sérigraphie d'avertissement de l'alim du NRF
			Review du placement émetteur
05.03.2023	Réalisation	Review du placement par Alexandre	

05.03.2023	Réalisation	Routage de l'émetteur - Design rules check OK !	
08.03.2023	Réalisation	Correction du PCB émetteur - Trous de fixation - Ecart avec les jumpers Correction footprints Correction schéma récepteur	
14.03.2023	Réalisation	Ajouts footprints récepteur - MIKROE - PIC32 + quartz - etc... Vérification finale émetteur - Pas d'erreur Altium ni EC !	Boitier émetteur
15.03.2023	Réalisation	Boitier émetteur terminé Réalisation du panel avec émetteur + commande Configuration pour conversion automatique de l'active BOM en commande Saphir Réalisation active BOM + commande	Placement et routage du récepteur
16.03.2023	Réalisation	Placement du récepteur (80%)	
18.03.2023	Réalisation	Placement du récepteur terminé Routage du récepteur terminé	Demander influence de l'oscillateur sur l'UART proche Demander pour les polygons et tailles de pistes
29.03.2023	Réalisation	Boitier émetteur recommandé - Ajout support de pile - Ajout texte info projet - Assemblage virtuel Envois fichiers 3D à Mr. Muller	
26.04.2023	Montage	Montage du 1er émetteur terminé	
03.05.2023	Montage	Recherche des raisons des erreurs de fonctionnement Analyse du datasheet du BQ25505	
10.05.2023	Montage	Montage du récepteur terminé Montage du 2eme émetteur terminé Tests de fonctionnement du récepteur Commande en urgence de panneaux solaire détruits lors du montage (température trop haute)	Installation environnements de programmation NRF
17.05.2023	Software	Installation environnement de programmation NRF - NRF Connect - NRF SDK	

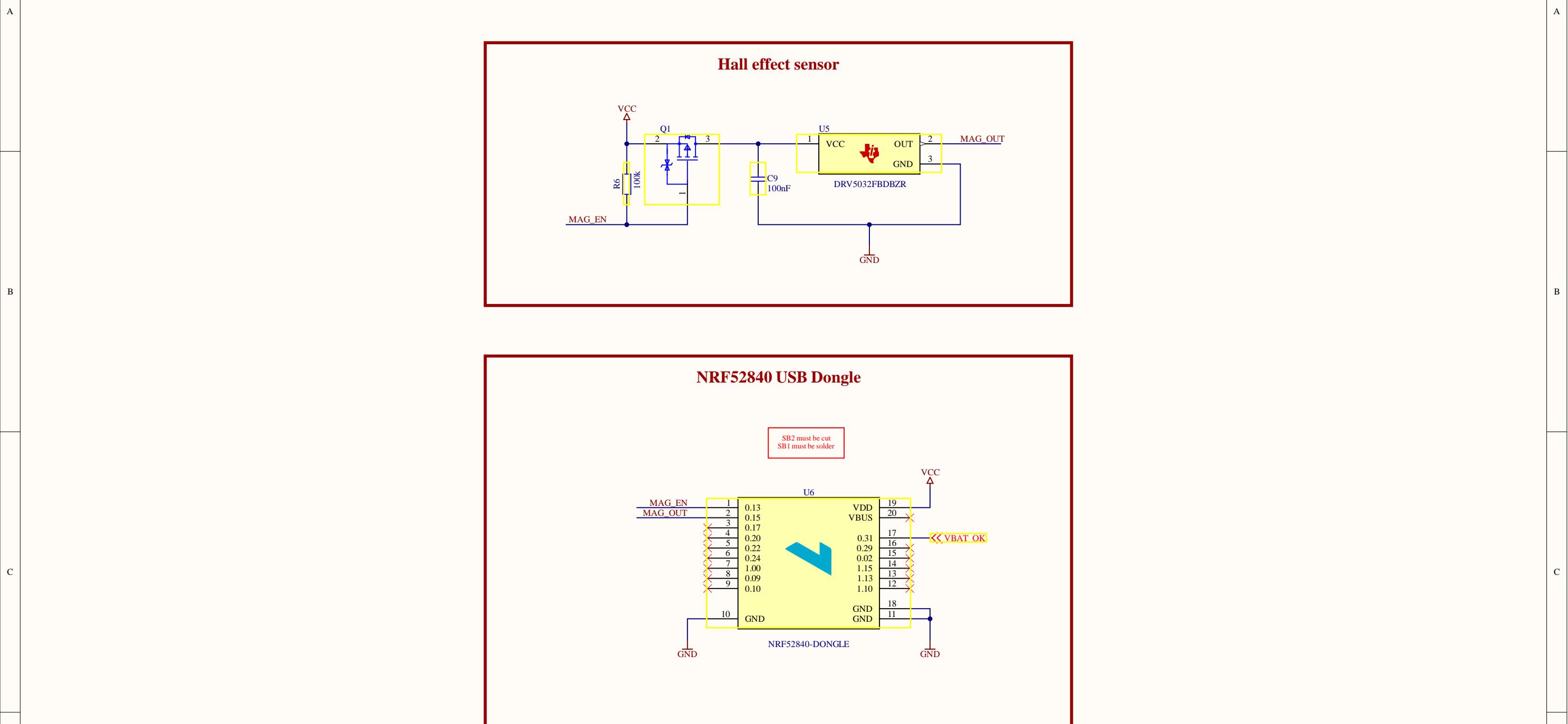
		Création du projet dans VS Code	
20.05.2023	Software	Etude du fonctionnement du SDK de Nordic	
21.05.2023	Software	Création d'une application basique avec les GPIO	
23.05.2023	Software	Tentative de créer une application Zigbee - Beaucoup d'erreurs de compilation, trop complexe à debugger	
24.05.2023	Software	Création d'une application bluetooth à partir d'exemple fourni - Communication réussie !	
27.05.2023	Software	Configuration Harmony, pins et drivers PIC32 du récepteur	
31.05.2023	Software	Tentative de créer une application Zigbee sur le NRF - Échoué ! -> Utiliser BT	
07.06.2023	Software	Création des projets NRF pour l'émetteur et le récepteur	
	Software	Configuration des différents GPIO nécessaires au NRF et programme	
12.06.2023	Rapport	Organisation de la documentation et dossiers	
14.06.2023	Software	Finalisation des différents programmes NRF et PIC32 pour le récepteur et émetteur	
14.06.2023	Rapport	Ecriture du rapport et différents documents demandés	

Power Management



Fichier : 2228_Emetteur_Alimentation.SchDoc	a
Date : 13.06.2023	b
Version : 1.0	c
Heure : 13:35:32	d
Auteur : Miguel Santos	
ETNL-ES Avenue Recordon 1 1004 Lausanne Switzerland	No : A
Chemin : C:\Users\migsantos\MIGSAN_GIT\2228_AlarmeFenetre\2228_AlarmeFenetreOuverte\hard\2228_Altium\2228_AlarmeFenetre_Emetteur	Nb feuillets 2
	Feuille n° 1

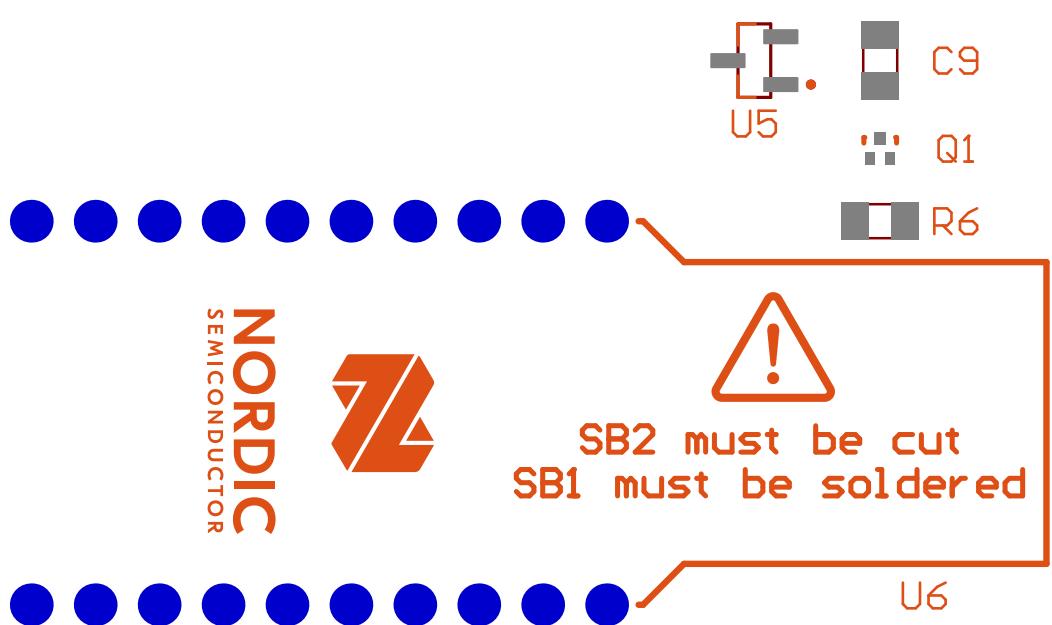
System





Bill of materials

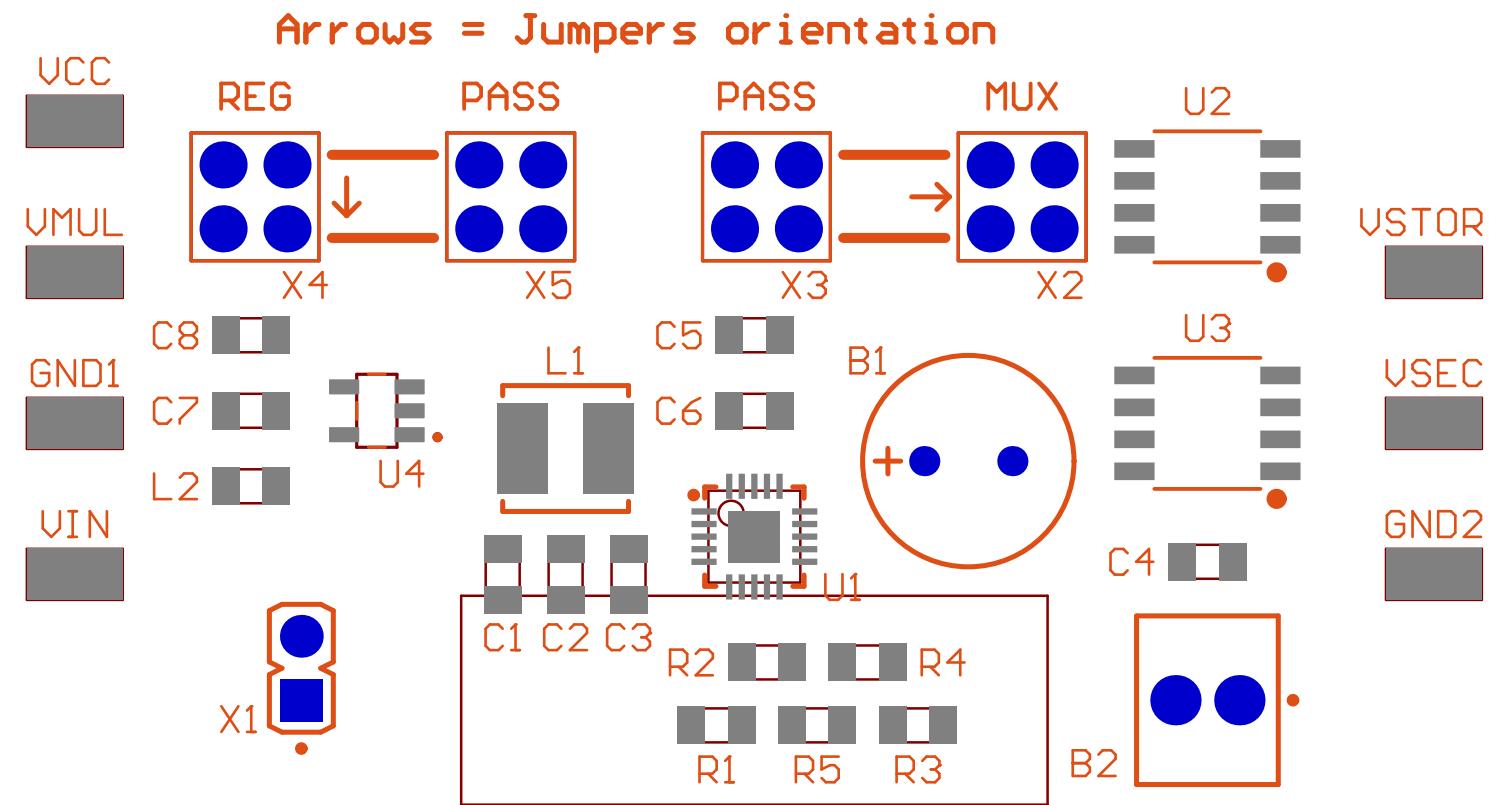
Name of project :		Alarme pour fenêtre ouverte			Author :	Miguel Santos		PCB :	A	
N° of project :		2228			Date :	13.06.2023		Version :	1.0	
Description	Designator	Value	Quantity	Manufacturer 1	Manufacturer Part Number 1	Supplier 1	Supplier Part Number 1	In stock	Price U.	Price total
	B2A		1	Keystone Electronics	2469	Digi-Key	36-2469-ND	No	1,20 CHF	1,20 CHF
Test Point, 1 Position SMD, RoHS, Tape and Reel	GND1, GND2, VCC, VIN, VMUL, VSEC, VSTOR		7	Keystone Electronics	5019	Digi-Key	36-5019CT-ND	Yes		
Low power energy harvester	U1		1	Texas Instruments	BQ25505RGRR	Digi-Key	296-37078-1-ND	No	4,21 CHF	4,21 CHF
	U5		1	Texas Instruments	DRV5032FBDBZR	Digi-Key	296-47323-1-ND	No	0,60 CHF	0,60 CHF
	U4		1	Texas Instruments	TPS13221ADBV7	Digi-Key	296-49467-1-ND	No	1,05 CHF	1,05 CHF
Polarized SuperCapacitor (Radial)	B1	2F	1	Cornell Dubilier	DSF205Q3R0	Digi-Key	338-DSF205Q3R0-ND	No	1,20 CHF	1,20 CHF
	M1		1	Radial Magnet	8189	Digi-Key	469-1059-ND	No	0,21 CHF	0,21 CHF
Capacitor	C8	22µF	1	Murata	GRM21BZ71A226ME15L	Digi-Key	490-GRM21BZ71A226ME15LCT-ND	No	0,36 CHF	0,36 CHF
	R4	3.6M	1	Vishay	CRCW08053M60FKEA	Digi-Key	541-3.60MCCT-ND	Yes		
	R2, R3	5.1M	2	Vishay	CRCW08055M10FKEA	Digi-Key	541-5.10MCCT-ND	Yes		
	R1	8.2M	1	Vishay	CRCW08058M20FKEA	Digi-Key	541-8.20MCCT-ND	Yes		
	R6	100k	1	Vishay	CRCW0805100KFKEB	Digi-Key	541-4318-1-ND	Yes		
	X2, X3, X4, X5		4	Wurth Electronics	61300421121	Mouser	710-61300421121	No	1,78 CHF	7,12 CHF
	X1		1	Wurth Electronics	61300211121	Digi-Key	732-5315-ND	Yes		
Capacitor	C3, C6	10nF	2	Wurth Electronics	8,85012E+11	Digi-Key	732-7579-1-ND	No	0,37 CHF	0,74 CHF
Capacitor	C1, C5, C7	4.7µF	3	Wurth Electronics	8,85012E+11	Digi-Key	732-7666-1-ND	No	1,11 CHF	3,33 CHF
Capacitor	C2, C4, C9	100nF	3	Wurth Electronics	8,85012E+11	Digi-Key	732-8026-1-ND	Yes		
Inductor SMD	L1	22µH	1	Wurth Electronics	74406043220	Digi-Key	732-10821-1-ND	No	0,96 CHF	0,96 CHF
Multicell Battery	B2		1	Wurth Electronics	6,91211E+11	Digi-Key	732-691210910002-ND	No	1,06 CHF	1,06 CHF
Inductor SMD	L2	2.2µH	1	Walsin Technologies	WLPH201610M2R2PP	Mouser	791-WLPH201610M2R2PP	No	0,10 CHF	0,10 CHF
	U6		1	Nordic Semiconductor	NRF52840-DONGLE	Digi-Key	1490-1073-ND	No	8,99 CHF	8,99 CHF
	PCB0		1	Eurocircuits	2228_Emetteur_PCB0	Eurocircuits	2228_Emetteur_PCB0	No	15,00 CHF	15,00 CHF
Anysolar KXOB25-05X3F	S1		1	ANY SOLAR	KXOB25-03X4F-TR	Digi-Key	2994-KXOB25-03X4F-TRCT-ND	No	2,85 CHF	2,85 CHF
	U2, U3		2	Diodes	DMP3085LSD-13	Digi-Key	DMP3085LSD-13DICT-ND	No	0,69 CHF	1,38 CHF
	R5	3.3M	1	Rohm	KTR10EZPF3304	Digi-Key	RHM3.30MAHCT-ND	Yes		
MOSFET P-CH 20V 0.1A VMT3	Q1		1	Rohm	RZM001P02T2L	Digi-Key	RZM001P02T2LCT-ND	No	0,24 CHF	0,24 CHF
								Total	50,60 CHF	

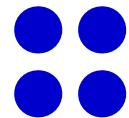
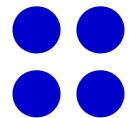
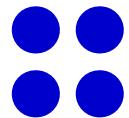
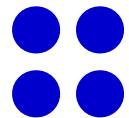


ETNL-ES

SLO2 2023
Miguel Santos

22280A
AlarmeFenetre





• •

• •



A

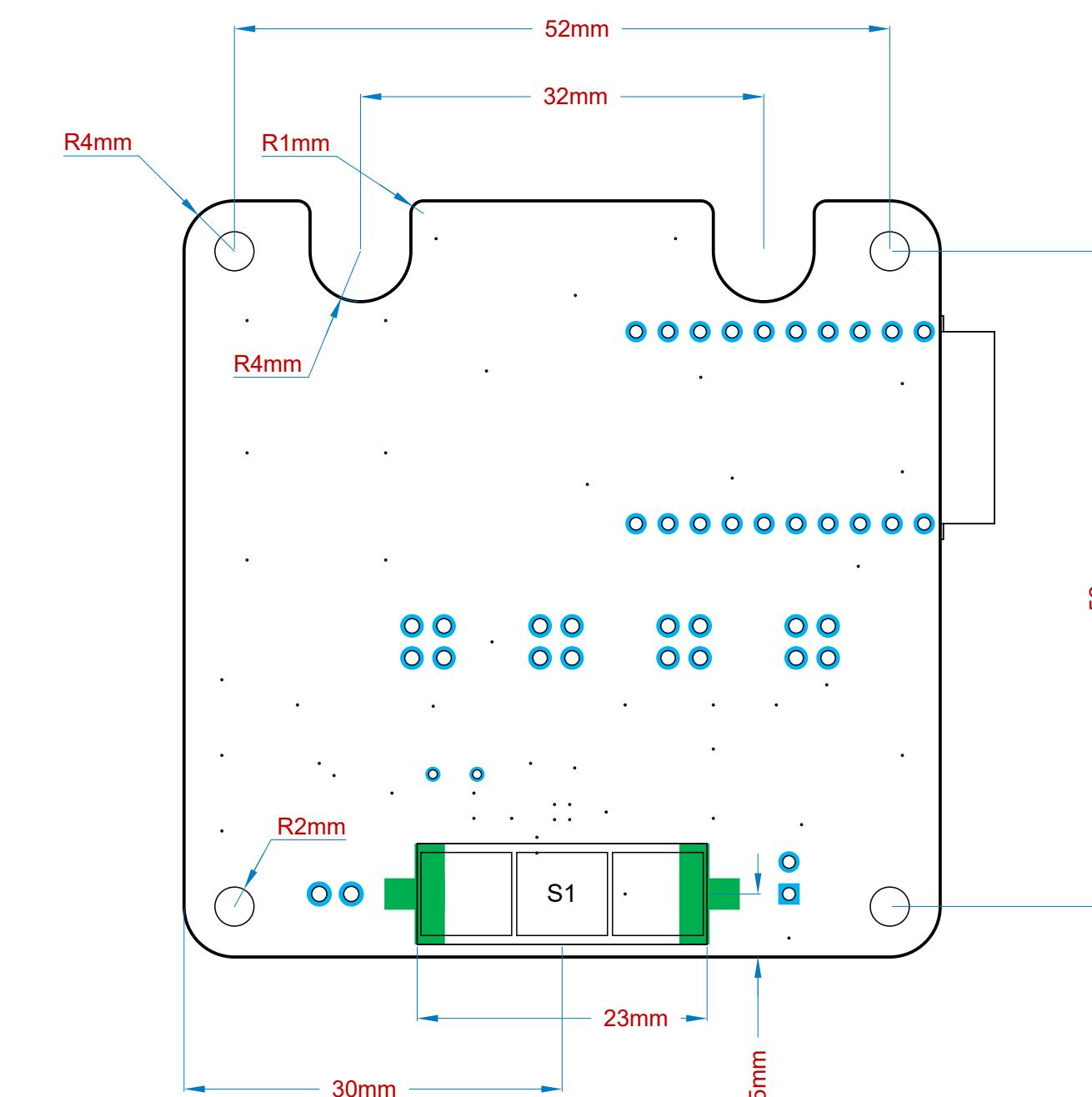
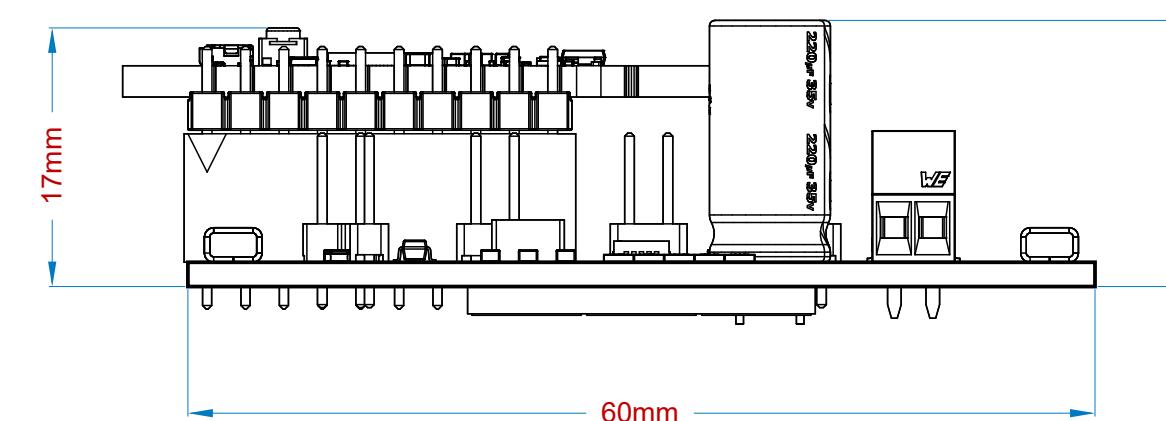
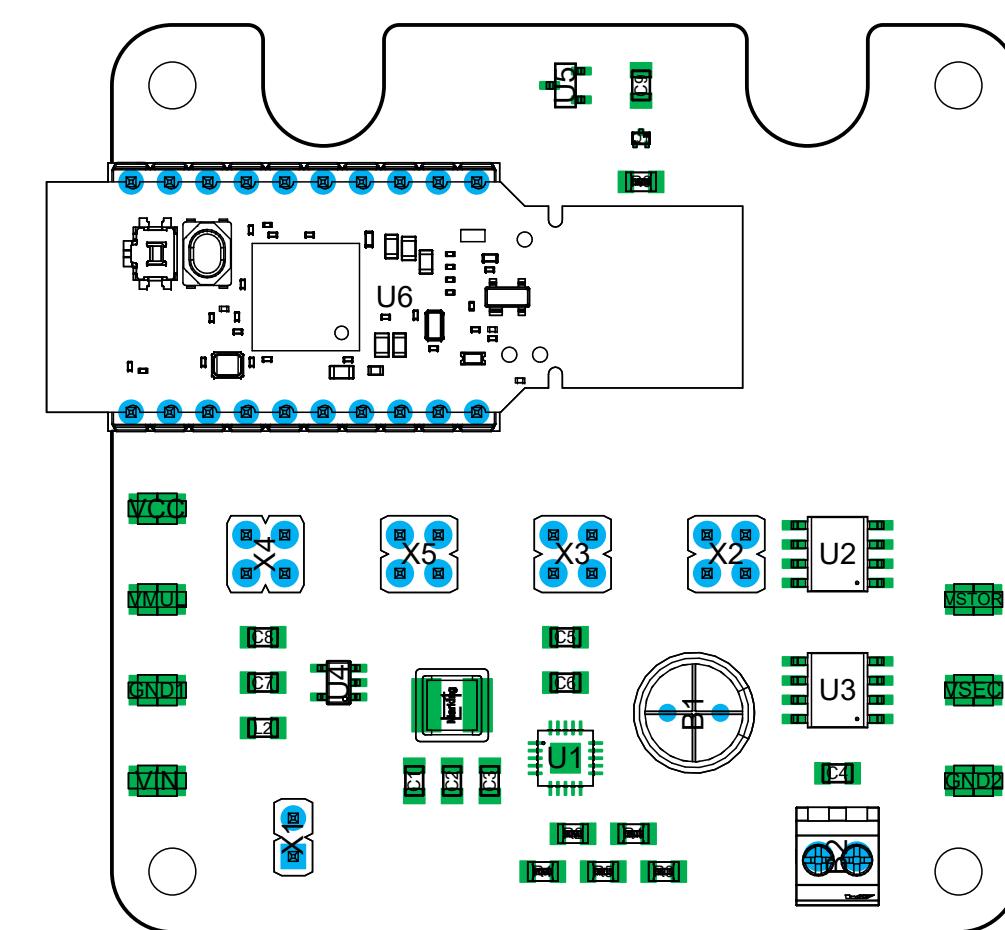
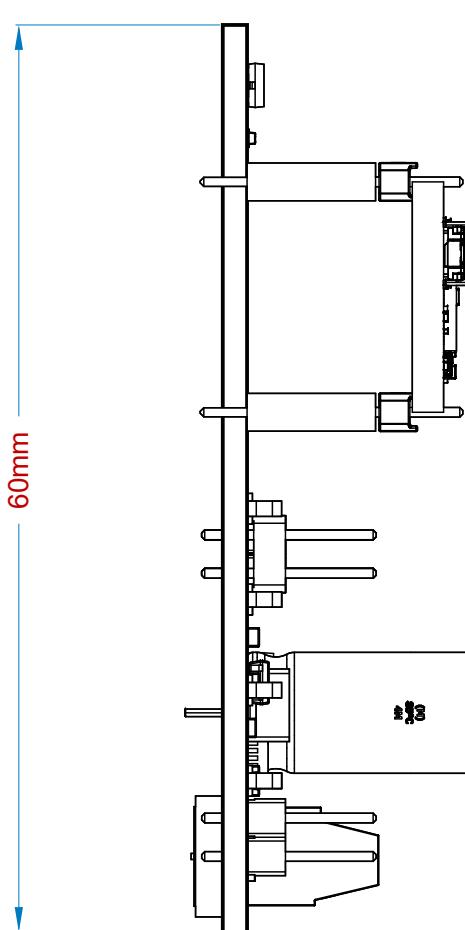
B

C

D

E

F



THE INFORMATION CONTAINED IN
THIS DRAWING IS THE SOLE
PROPERTY OF
ETML-ES. ANY REPRODUCTION IN
PART OR AS A WHOLE WITHOUT
THE WRITTEN PERMISSION OF
PROPRIETARY AND CONFIDENTIAL

		UNLESS OTHERWISE SPECIFIED:	NAME	DATE	ETML-ES	
		DIMENSIONS ARE IN INCHES	DRAWN	15.06.2023	TITLE	
		TOLERANCES: FRACTIONAL: ± ANGULAR: MACH± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	CHECKED		2228_Emetteur	
		INTERPRET GEOMETRIC TOLERANCING PER:	ENG APPR.			
		MATERIAL	MFG APPR.			
		NEXT ASSY	USED ON	Q.A.	COMMENTS:	
		FINISH		MATERIAL		
		APPLICATION		DO NOT SCALE DRAWING		
SCALE:	2:1	WEIGHT:		SHEET 1 OF 1		

A

A

B

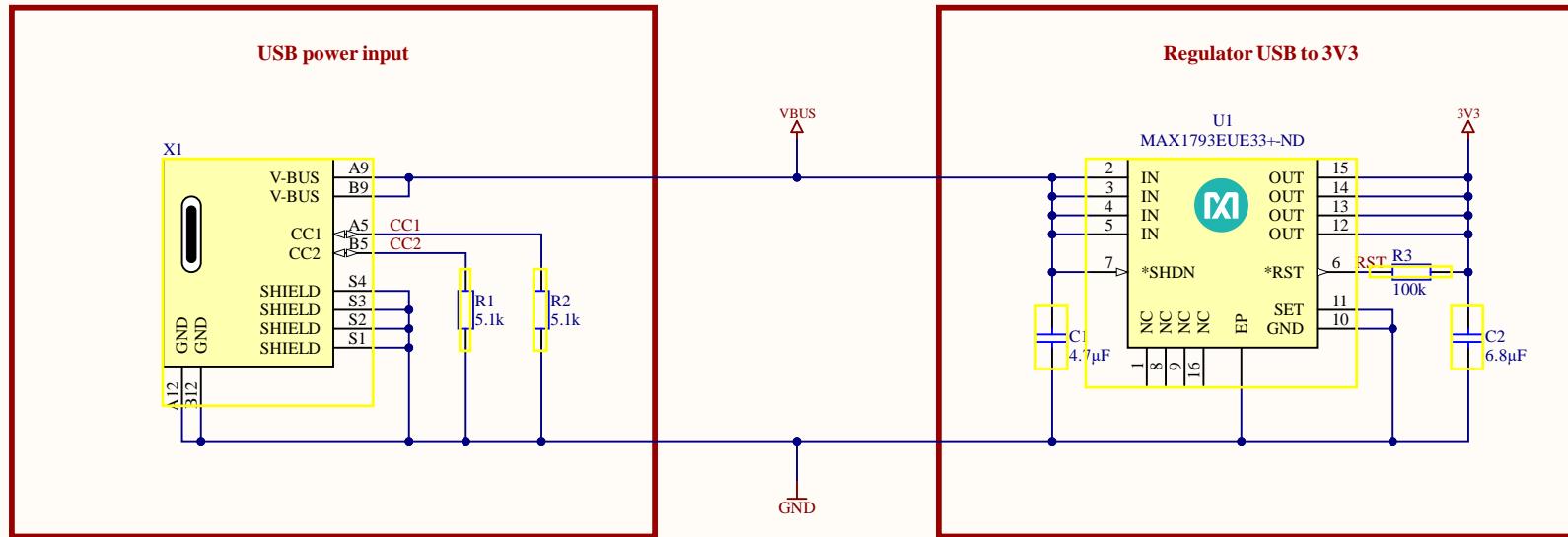
B

C

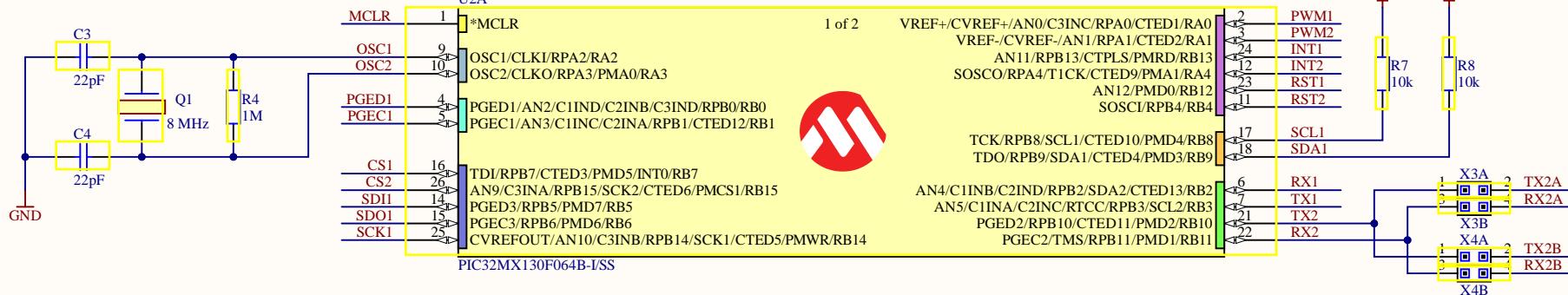
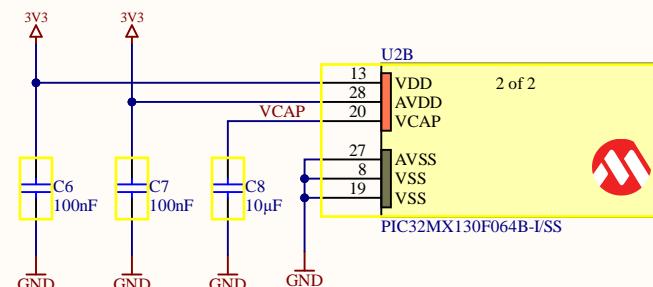
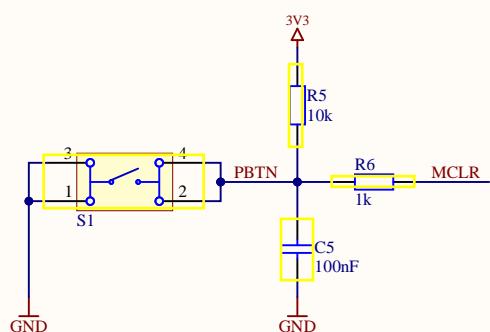
C

D

D



Fichier : 2228_Recepteur_Alimentation.SchDoc	a
Date : 13.06.2023	b
Heure : 13:44:05	c
Auteur : Miguel Santos	d
Modif.	

PIC32MX130**RESET Button**

A

A

B

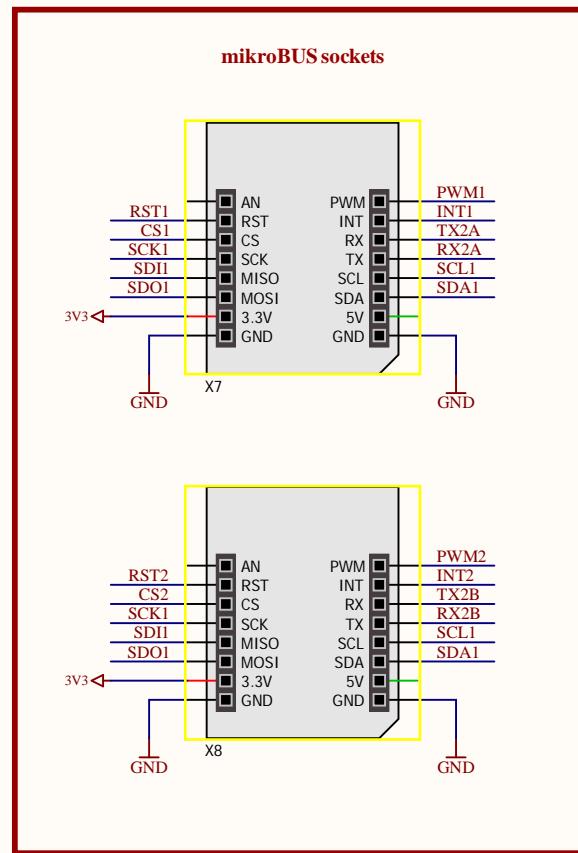
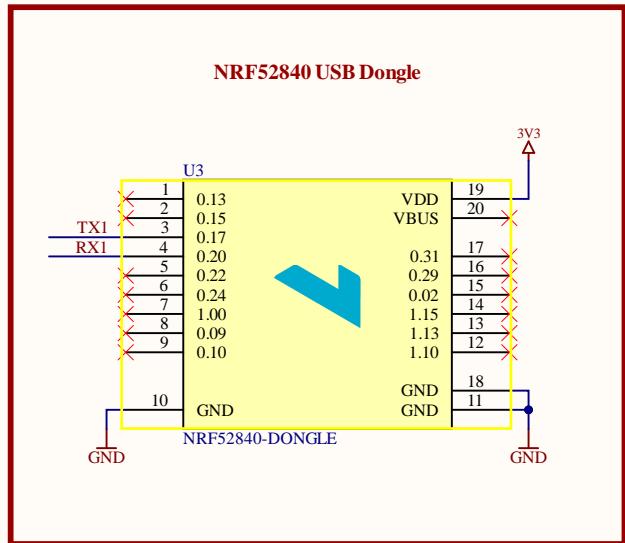
B

C

C

D

D



Fichier : 2228_Recepteur_MIKROE_NRF.SchDoc

Date : 13.06.2023 Version : 1.0

Heure : 13:44:07 Auteur : Miguel Santos

Modif.
a
b
c
d
ETNL-ES
 Avenue Recordon 1
 1004 Lausanne
 Switzerland

Projet : 2228_AlarmeFenetre_Recepteur.PrjPcb

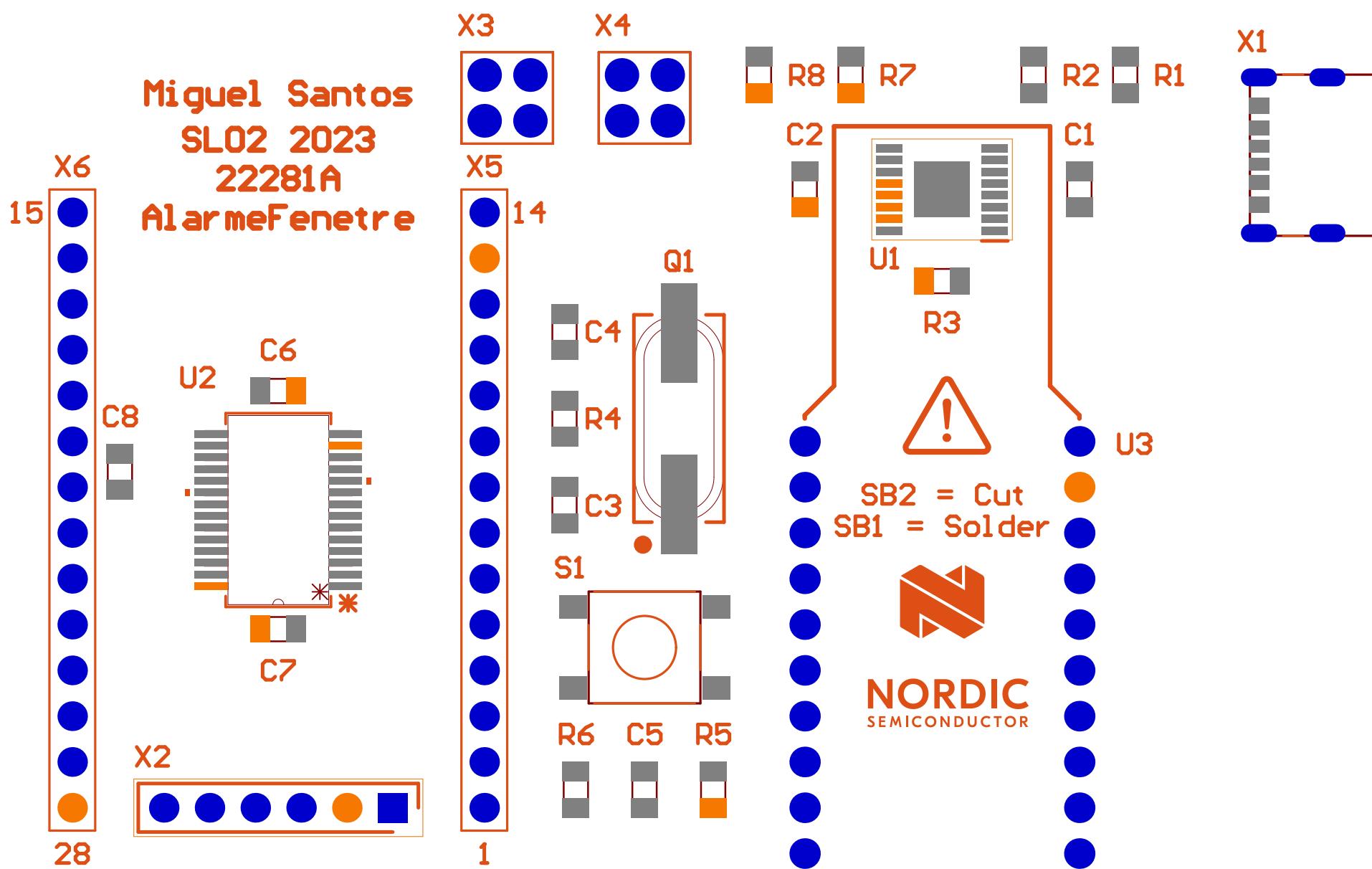
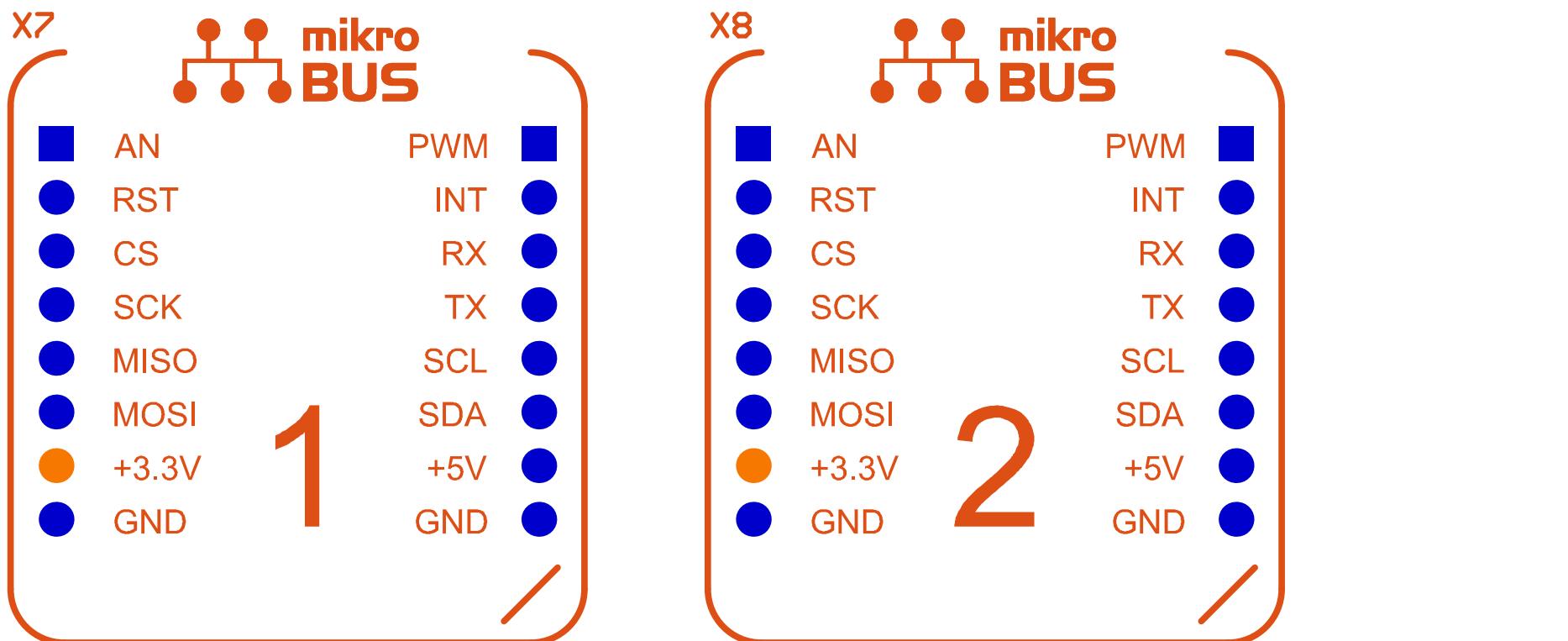
 No : B
 Nb feuillets 3 Feuilles n° 3

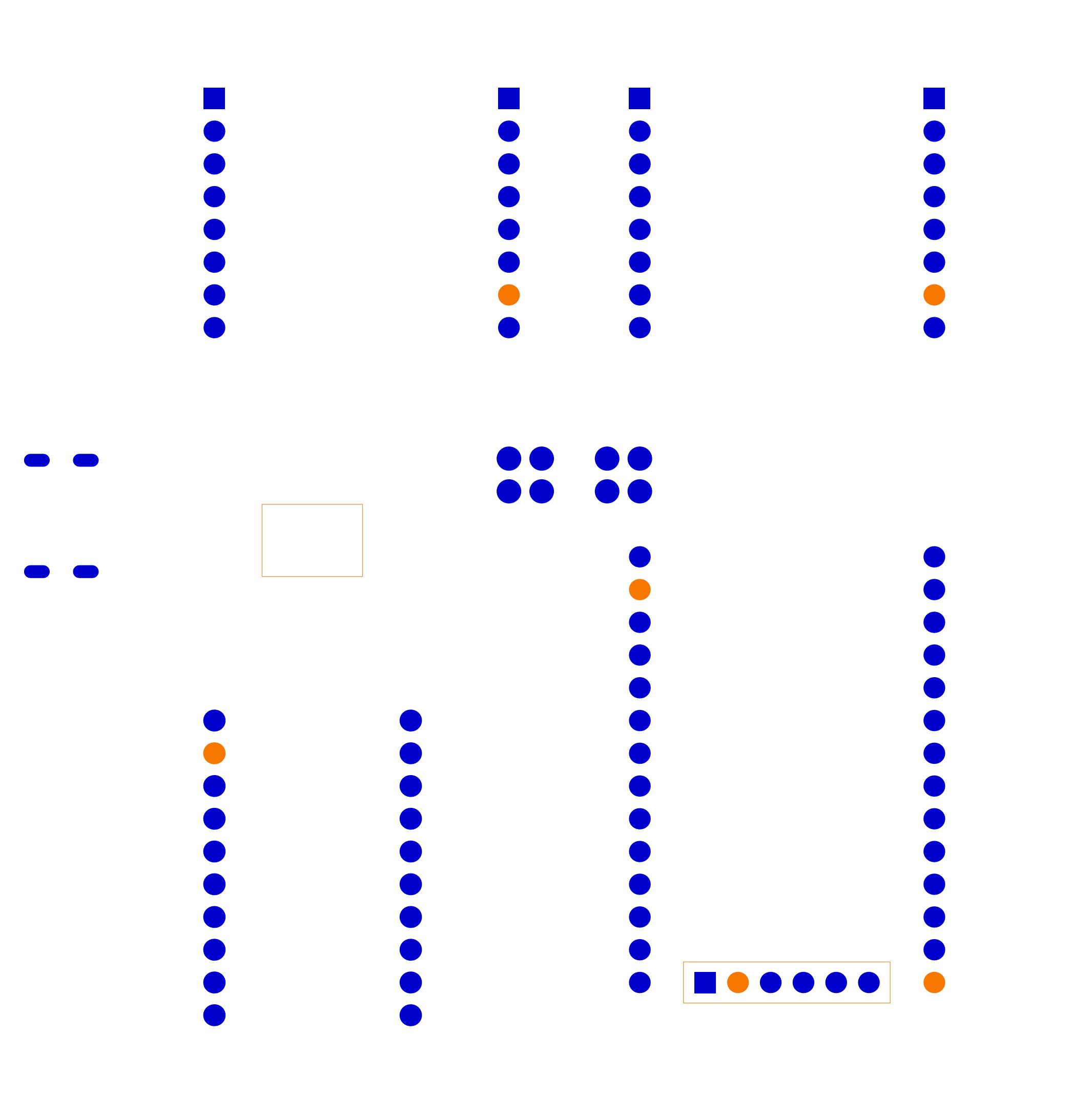
Chemin : C:\Users\migsantos\MIGSAN_GIT\2228_AlarmeFenetre\2228_AlarmeFenetreOverte\hard\2228_Altium\2228_AlarmeFenetre_Recepteur



Bill of materials

Name of project :		Alarme pour fenêtre ouverte			Author :	Miguel Santos		PCB :	B	
N° of project :		2228			Date :	13.06.2023		Version :	1.0	
Description	Designator	Value	Quantity	Manufacturer 1	Manufacturer Part Number 1	Supplier 1	Supplier Part Number 1	In stock	Price U.	Price total
Ceramic capacitor	C1	4.7µF	1	Wurth Electronics	8,85012E+11	Digi-Key	732-7666-1-ND	No	0,36 CHF	0,36 CHF
Ceramic capacitor	C2	6.8µF	1	TDK	CGA4J1X7R0J685K125AC	Digi-Key	445-12777-1-ND	No	0,36 CHF	0,36 CHF
Ceramic capacitor	C3, C4	22pF	2	Wurth Electronics	8,85012E+11	Digi-Key	732-7807-1-ND	No	0,09 CHF	0,18 CHF
Ceramic capacitor	C5, C6, C7	100nF	3	Wurth Electronics	8,85012E+11	Digi-Key	732-8026-1-ND	Yes		
Ceramic capacitor	C8	10µF	1	Wurth Electronics	8,85012E+11	Digi-Key	732-7620-1-ND	No	0,18 CHF	0,18 CHF
IQD FREQUENCY PRODUCTS - LFXTAL003151REEL - CRYSTAL, 8MHZ, 16PF, 11.4 X 4.9MM	Q1		1	IQD	LFXTAL003151REEL	Digi-Key	1923-1454-1-ND	Yes		
	R1, R2	5.1k	2	Stackpole Electronics	RMCF0805FT5K10	Digi-Key	RMCF0805FT5K10CT-ND	Yes		
	R3	100k	1	Stackpole Electronics	RMCF0805FT100K	Digi-Key	RMCF0805FT100KCT-ND	Yes		
	R4	1M	1	Stackpole Electronics	RMCF0805FT1M00	Digi-Key	RMCF0805FT1M00CT-ND	Yes		
	R5, R7, R8	10k	3	Stackpole Electronics	RNCP0805FTD10K0	Digi-Key	RNCP0805FTD10K0CT-ND	Yes		
	R6	1k	1	Stackpole Electronics	RNCP0805FTD1K00	Digi-Key	RNCP0805FTD1K00CT-ND	Yes		
WS-TASV SMD Tact Switch 6X6 mm	S1		1	Wurth Electronics	4,30182E+11	Digi-Key	732-7004-1-ND	Yes		
	U1		1	Analog Devices	MAX1793EUE33+	Digi-Key	MAX1793EUE33+-ND	Yes		
	U2		1	Microchip	PIC32MX130F064B-I/SS	Digi-Key	PIC32MX130F064B-I/SS-ND	Yes		
	U3		1	Nordic Semiconductor	NRF52840-DONGLE	Digi-Key	1490-1073-ND	No	8,99 CHF	8,99 CHF
Type C, 3 A, Right Angle, Surface Mount (SMT), Power-Only USB Receptacle	X1		1	CUI Devices	UJC-HP-3-SMT-TR	Digi-Key	2223-UJC-HP-3-SMT-CT-ND	No	1,04 CHF	1,04 CHF
	X2		1	Wurth Electronics	61300611121	Digi-Key	732-5319-ND	Yes		
	X3, X4		2	Wurth Electronics	61300421121	ouser	710-61300421121	Yes		
	X5, X6		2	Wurth Electronics	61301411121	Digi-Key	732-5325-ND	Yes		
	X7, X8		2	mikroElektronika	MIKROE-4247	ouser	932-MIKROE-4247	Yes		
								Total	11,11 CHF	





A

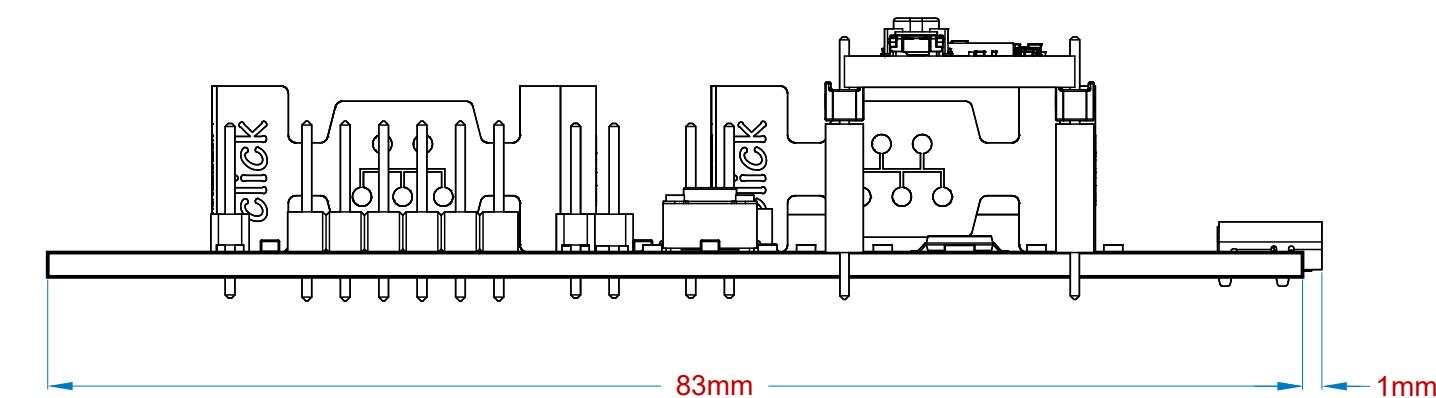
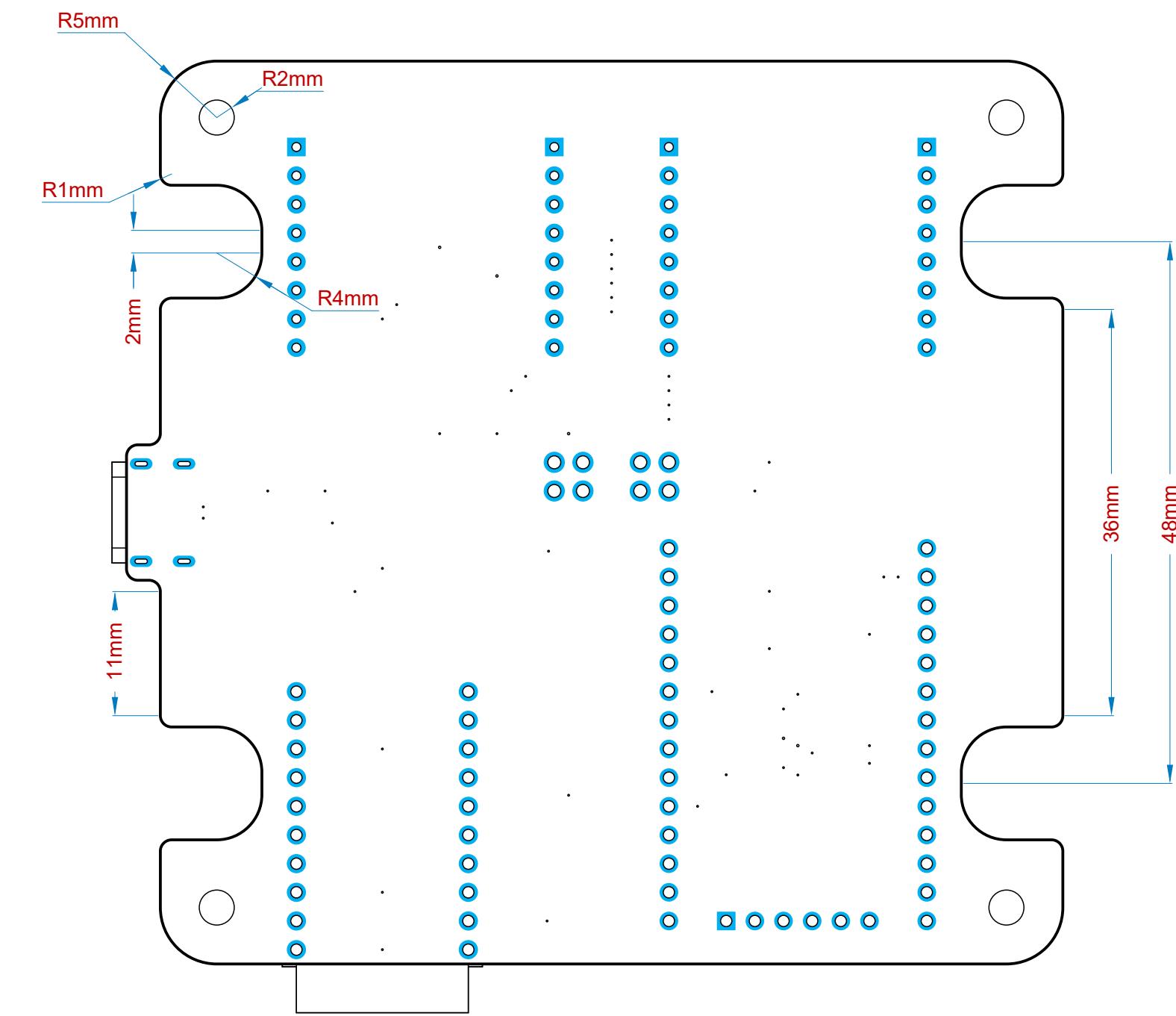
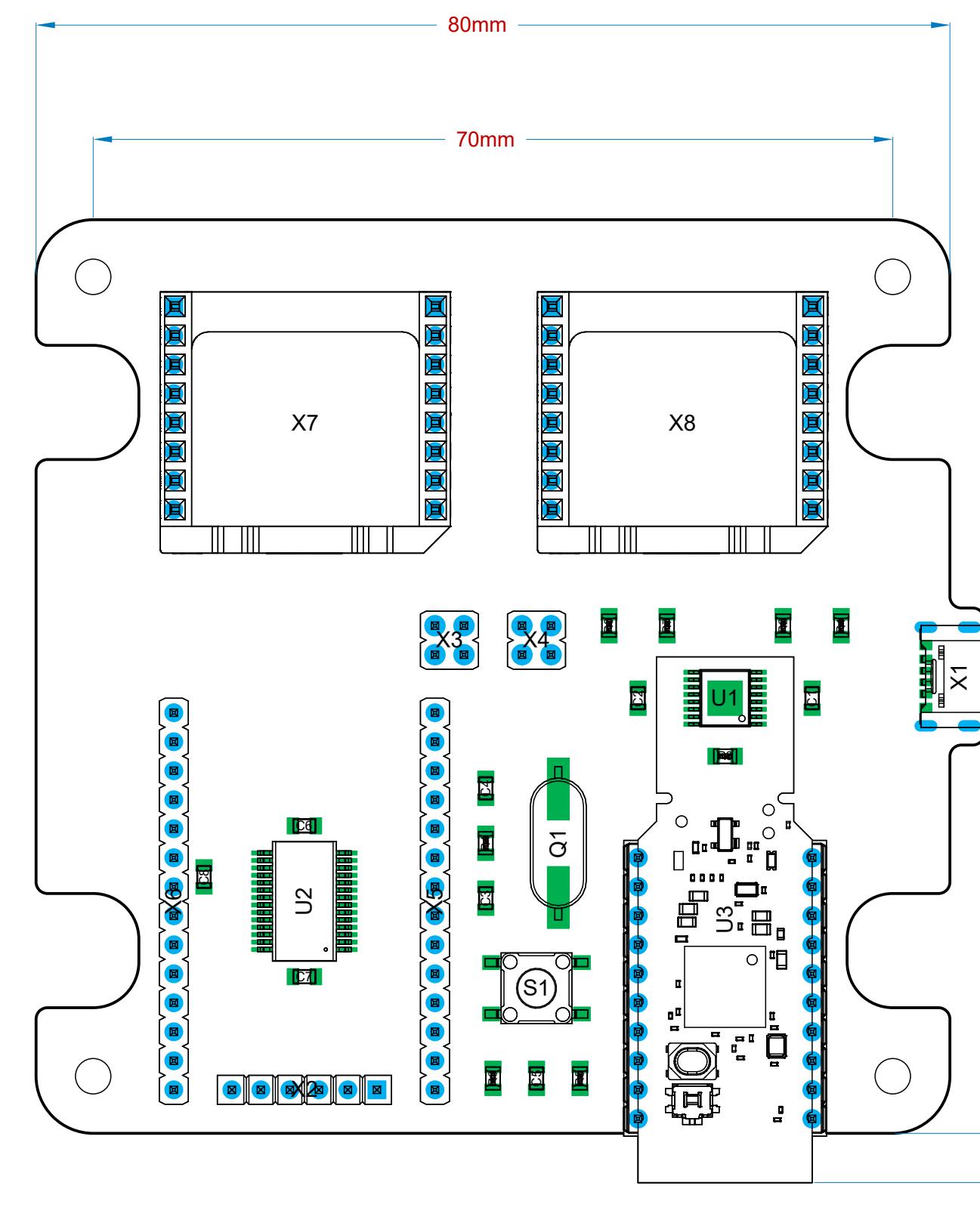
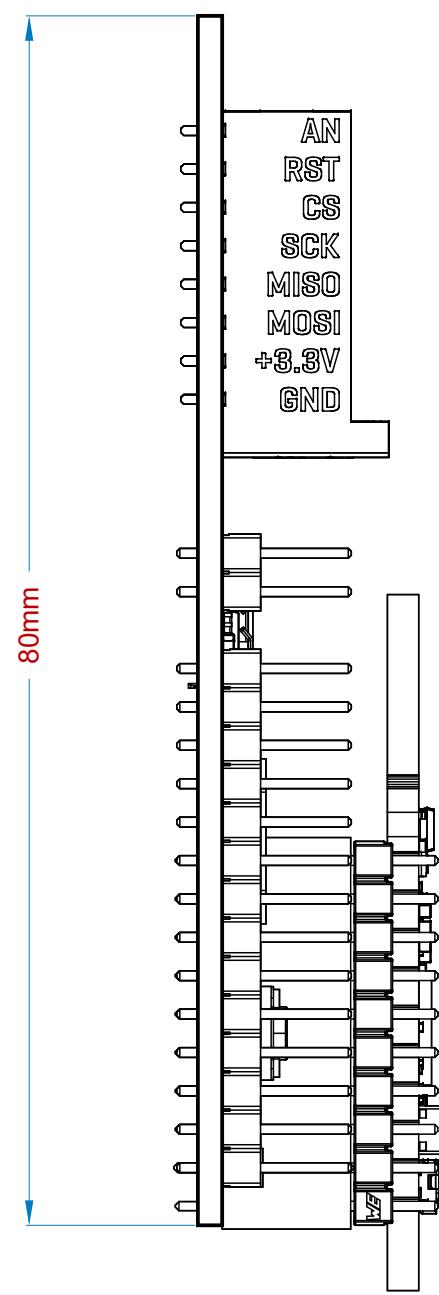
B

C

D

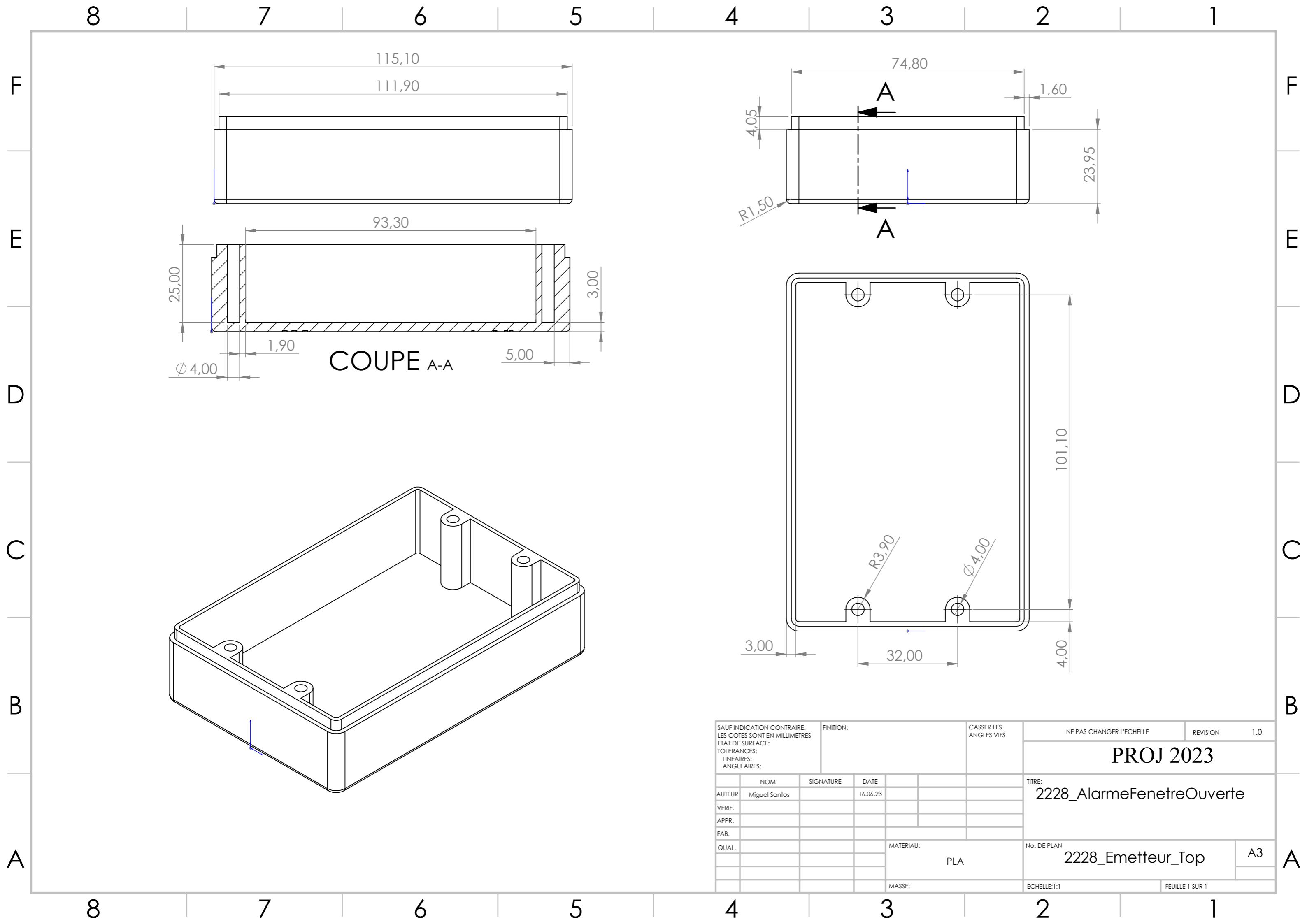
E

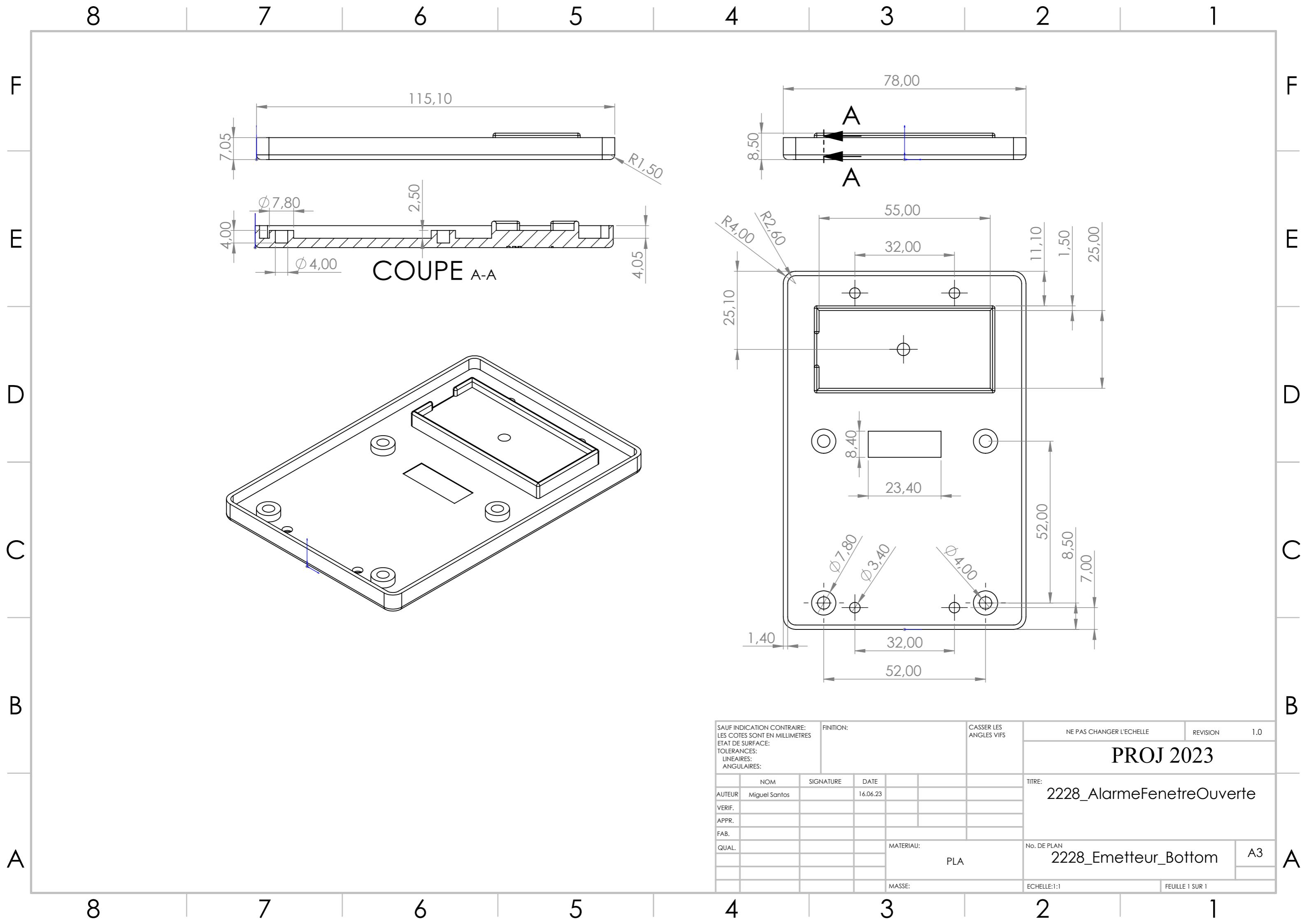
F



THE INFORMATION CONTAINED IN
THIS DRAWING IS THE SOLE
PROPERTY OF
ETML-ES. ANY REPRODUCTION IN
PART OR AS A WHOLE WITHOUT
THE WRITTEN PERMISSION OF
PROPRIETARY AND CONFIDENTIAL

		UNLESS OTHERWISE SPECIFIED:		NAME	DATE	ETML-ES	
		DIMENSIONS ARE IN INCHES				TITLE	
		TOLERANCES:		CHECKED		2228_Récepteur	
		FRACTIONAL:	ANGULAR: MACH \pm BEND \pm	ENG APPR.			
		TWO PLACE DECIMAL \pm	THREE PLACE DECIMAL \pm	MFG APPR.			
		INTERPRET GEOMETRIC TOLERANCING PER:		Q.A.			
		MATERIAL		COMMENTS:			
		NEXT ASSY	USED ON	FINISH			
		APPLICATION		DO NOT SCALE DRAWING			
		SCALE: 1:2		WEIGHT:		SHEET 1 OF 1	





```

1  /*
2   * ETML-ES
3   *
4   * 2228_AlarmeFenetreOuverte
5   *
6   * Main NRF Emetteur
7   *
8   * Miguel Santos
9   * 2023
10 */
11
12 #include <zephyr/kernel.h>
13 #include <zephyr/drivers/gpio.h>
14
15
16 /* Main loop sleeping time in ms */
17 #define SLEEP_TIME_MS    100000
18
19 /* Devicetree nodes identifiers */
20 #define LED0_NODE DT_ALIAS(led0)
21 #define LED1_NODE DT_ALIAS(led1)
22 #define LED2_NODE DT_ALIAS(led2)
23 #define LED3_NODE DT_ALIAS(led3)
24
25 #define SW0_NODE DT_ALIAS(sw0)
26
27 #define VBAT_OK_NODE DT_ALIAS(vbat_ok)
28 #define MAG_OUT_NODE DT_ALIAS(mag_out)
29 #define MAG_EN_NODE DT_ALIAS(mag_en)
30
31
32 /* GPIO specifications */
33 static const struct gpio_dt_spec led0 = GPIO_DT_SPEC_GET(LED0_NODE, gpios);
34 static const struct gpio_dt_spec led1 = GPIO_DT_SPEC_GET(LED1_NODE, gpios);
35 static const struct gpio_dt_spec led2 = GPIO_DT_SPEC_GET(LED2_NODE, gpios);
36 static const struct gpio_dt_spec led3 = GPIO_DT_SPEC_GET(LED3_NODE, gpios);
37
38 static const struct gpio_dt_spec btn0 = GPIO_DT_SPEC_GET(SW0_NODE, gpios);
39
40 static const struct gpio_dt_spec batteryOk = GPIO_DT_SPEC_GET(VBAT_OK_NODE, gpios);
41 static const struct gpio_dt_spec sensorPin = GPIO_DT_SPEC_GET(MAG_OUT_NODE, gpios);
42 static const struct gpio_dt_spec sensorEnable = GPIO_DT_SPEC_GET(MAG_EN_NODE, gpios);
43
44
45 /* Define a variable of type static struct gpio_callback */
46 static struct gpio_callback btn0_cb_data;
47 static struct gpio_callback sensor_cb_data;
48
49
50 /* Check if devices are available
51  * if not, programm should be stop */
52 bool check_devices()
53 {
54     bool device_error = false;
55
56     if (!device_is_ready(led0.port)) {
57         device_error = true;
58     }
59
60     if (!device_is_ready(led1.port)) {
61         device_error = true;
62     }
63
64     if (!device_is_ready(led2.port)) {
65         device_error = true;
66     }
67
68     if (!device_is_ready(led3.port)) {
69         device_error = true;
70     }
71
72     if (!device_is_ready(btn0.port)) {
73         device_error = true;

```

```

74     }
75
76     if (!device_is_ready(batteryOk.port)) {
77         device_error = true;
78     }
79
80     if (!device_is_ready(sensorPin.port)) {
81         device_error = true;
82     }
83
84     if (!device_is_ready(sensorEnable.port)) {
85         device_error = true;
86     }
87
88     return device_error;
89 }
90
91 /* Configure pins to input or outputs */
92 bool configure_devices()
93 {
94     bool device_error = false;
95     int device_return;
96
97     device_return = gpio_pin_configure_dt(&led0, GPIO_OUTPUT_INACTIVE);
98     if (device_return < 0) {
99         device_error = true;
100    }
101
102    device_return = gpio_pin_configure_dt(&led1, GPIO_OUTPUT_INACTIVE);
103    if (device_return < 0) {
104        device_error = true;
105    }
106
107    device_return = gpio_pin_configure_dt(&led2, GPIO_OUTPUT_INACTIVE);
108    if (device_return < 0) {
109        device_error = true;
110    }
111
112    device_return = gpio_pin_configure_dt(&led3, GPIO_OUTPUT_INACTIVE);
113    if (device_return < 0) {
114        device_error = true;
115    }
116
117    device_return = gpio_pin_configure_dt(&btn0, GPIO_INPUT);
118    if (device_return < 0) {
119        device_error = true;
120    }
121
122    device_return = gpio_pin_configure_dt(&batteryOk, GPIO_INPUT);
123    if (device_return < 0) {
124        device_error = true;
125    }
126
127    device_return = gpio_pin_configure_dt(&sensorPin, GPIO_INPUT);
128    if (device_return < 0) {
129        device_error = true;
130    }
131
132    device_return = gpio_pin_configure_dt(&sensorEnable, GPIO_OUTPUT_INACTIVE);
133    if (device_return < 0) {
134        device_error = true;
135    }
136
137    return device_error;
138 }
139
140 void ISR_btn0(const struct device *dev, struct gpio_callback *cb, uint32_t pins)
141 {
142     if(gpio_pin_get_dt(&btn0))
143     {
144         gpio_pin_set_dt(&led0, true);
145     }
146     else

```

```

147     {
148         gpio_pin_set_dt(&led0, false);
149     }
150 }
151
152 void ISR_sensor(const struct device *dev, struct gpio_callback *cb, uint32_t pins)
153 {
154     if(gpio_pin_get_dt(&sensorPin))
155     {
156         gpio_pin_set_dt(&led3, false);
157     }
158     else
159     {
160         gpio_pin_set_dt(&led3, true);
161     }
162 }
163
164 bool ISR_btn0_configure()
165 {
166     bool int_return;
167
168     int_return = gpio_pin_interrupt_configure_dt(&btn0, GPIO_INT_EDGE_BOTH);
169
170     /* Initialize the static struct gpio_callback variable */
171     gpio_init_callback(&btn0_cb_data, ISR_btn0, BIT(btn0.pin));
172
173     /* Add the callback function by calling gpio_add_callback() */
174     gpio_add_callback(btn0.port, &btn0_cb_data);
175
176     return int_return;
177 }
178
179 bool ISR_sensor_configure()
180 {
181     bool int_return;
182
183     int_return = gpio_pin_interrupt_configure_dt(&sensorPin, GPIO_INT_EDGE_BOTH);
184
185     /* Initialize the static struct gpio_callback variable */
186     gpio_init_callback(&sensor_cb_data, ISR_sensor, BIT(sensorPin.pin));
187
188     /* Add the callback function by calling gpio_add_callback() */
189     gpio_add_callback(sensorPin.port, &sensor_cb_data);
190
191     return int_return;
192 }
193
194
195 void main(void)
196 {
197     //int gpio_return;
198     bool error_return;
199
200     error_return = check_devices();
201     if(error_return){
202         return;
203     }
204
205     error_return = configure_devices();
206     if(error_return){
207         return;
208     }
209
210     ISR_btn0_configure();
211     ISR_sensor_configure();
212
213     while (true) {
214         k_msleep(SLEEP_TIME_MS);
215     }
216 }
217

```

```

1  /*
2   *  ETML-ES
3   *
4   *  2228_AlarmeFenetreOuverte
5   *
6   *  Main NRF Récepteur
7   *
8   *  Miguel Santos
9   *  2023
10  */
11 #include <zephyr/kernel.h>
12 #include <zephyr/drivers/gpio.h>
13 #include <zephyr/drivers/uart.h>
14
15 /* Main loop sleeping time in ms */
16 #define SLEEP_TIME_MS    100000
17
18 /* Action sent by uart on PIC32 */
19 #define ACTION1 0x0F
20
21 /* Devicetree nodes identifiers */
22 #define LED0_NODE DT_ALIAS(led0)
23 #define LED1_NODE DT_ALIAS(led1)
24 #define LED2_NODE DT_ALIAS(led2)
25 #define LED3_NODE DT_ALIAS(led3)
26 #define SW0_NODE DT_ALIAS(sw0)
27
28 #define UART_NODE DT_NODENAME(uart0)
29
30 /* GPIO specifications */
31 static const struct gpio_dt_spec led0 = GPIO_DT_SPEC_GET(LED0_NODE, gpios);
32 static const struct gpio_dt_spec led1 = GPIO_DT_SPEC_GET(LED1_NODE, gpios);
33 static const struct gpio_dt_spec led2 = GPIO_DT_SPEC_GET(LED2_NODE, gpios);
34 static const struct gpio_dt_spec led3 = GPIO_DT_SPEC_GET(LED3_NODE, gpios);
35 static const struct gpio_dt_spec btn0 = GPIO_DT_SPEC_GET(SW0_NODE, gpios);
36
37 /* Infos about UART */
38 static const struct device *uart = DEVICE_DT_GET(UART_NODE);
39
40 /* Buffer to store incoming UART data*/
41 static uint8_t rx_buffer[10] = {0};
42
43 /* Infos about btn0 isr callback */
44 static struct gpio_callback btn0_cb_data;
45
46 /* Check if devices are available
47  * if not, programm should be stop */
48 bool check_devices()
49 {
50     bool device_error = false;
51
52     if (!device_is_ready(led0.port)) {
53         device_error = true;
54     }
55
56     if (!device_is_ready(led1.port)) {
57         device_error = true;
58     }
59
60     if (!device_is_ready(led2.port)) {
61         device_error = true;
62     }
63
64     if (!device_is_ready(led3.port)) {
65         device_error = true;
66     }
67
68     if (!device_is_ready(btn0.port)) {
69         device_error = true;
70     }
71
72     if (!device_is_ready(uart)) {
73         device_error = true;

```

```

74     }
75
76     return device_error;
77 }
78
79 /* Configure pins to input or outputs */
80 bool configure_devices()
81 {
82     bool device_error = false;
83     int device_return;
84
85     device_return = gpio_pin_configure_dt(&led0, GPIO_OUTPUT_INACTIVE);
86     if (device_return < 0) {
87         device_error = true;
88     }
89
90     device_return = gpio_pin_configure_dt(&led1, GPIO_OUTPUT_INACTIVE);
91     if (device_return < 0) {
92         device_error = true;
93     }
94
95     device_return = gpio_pin_configure_dt(&led2, GPIO_OUTPUT_INACTIVE);
96     if (device_return < 0) {
97         device_error = true;
98     }
99
100    device_return = gpio_pin_configure_dt(&led3, GPIO_OUTPUT_INACTIVE);
101   if (device_return < 0) {
102       device_error = true;
103   }
104
105   device_return = gpio_pin_configure_dt(&btn0, GPIO_INPUT);
106   if (device_return < 0) {
107       device_error = true;
108   }
109
110   return device_error;
111 }
112
113 /* Callback function of btn0 ISR */
114 void btn0_cb(const struct device *dev, struct gpio_callback *cb, uint32_t pins)
115 {
116     if(gpio_pin_get_dt(&btn0))
117     {
118         gpio_pin_set_dt(&led0, true);
119     }
120     else
121     {
122         gpio_pin_set_dt(&led0, false);
123     }
124 }
125
126 /* Configure interrupt service routine */
127 bool btn0_configure_isr()
128 {
129     bool int_return;
130
131     /* Assign an interrupt to a pin and trigger edge */
132     int_return = gpio_pin_interrupt_configure_dt(&btn0, GPIO_INT_EDGE_BOTH);
133
134     /* Initialize the static struct gpio_callback variable */
135     gpio_init_callback(&btn0_cb_data, btn0_cb, BIT(btn0.pin));
136
137     /* Add the callback function by calling gpio_add_callback() */
138     gpio_add_callback(btn0.port, &btn0_cb_data);
139
140     return int_return;
141 }
142
143 static void uart_cb(const struct device *dev, struct uart_event *evt, void *user_data)
144 {
145     switch (evt->type) {
146         case UART_TX_DONE:

```

```

147         // do something
148         break;
149     case UART_TX_ABORTED:
150         // do something
151         break;
152     case UART_RX_RDY:
153         if(evt->data.rx.buf[evt->data.rx.offset] == ACTION1)
154         {
155             gpio_pin_toggle_dt(&led1);
156         }
157         break;
158     case UART_RX_BUF_REQUEST:
159         // do something
160         break;
161     case UART_RX_BUF_RELEASED:
162         // do something
163         break;
164     case UART_RX_DISABLED:
165         uart_rx_enable( uart, rx_buffer, sizeof(rx_buffer), 100);
166         break;
167     case UART_RX_STOPPED:
168         // do something
169         break;
170     default:
171         break;
172     }
173 }
174
175 void main(void)
176 {
177     bool error_return;
178
179     /* Check if devices are configured right */
180     error_return = check_devices();
181     if(error_return){
182         return;
183     }
184
185     /* Configure GPIO to input or output */
186     error_return = configure_devices();
187     if(error_return){
188         return;
189     }
190
191     /* Configure UART callback routine */
192     error_return = uart_callback_set(uart, uart_cb, NULL);
193     if (error_return) {
194         return;
195     }
196
197     /* Enable UART */
198     uart_rx_enable(uart ,rx_buffer ,sizeof(rx_buffer) ,100);
199
200     /* Configure interrupt related to btn0*/
201     btn0_configure_isr();
202
203     while (true) {
204         k_msleep(SLEEP_TIME_MS);
205     }
206 }
207

```