

Système de prise d'images RTI

Conception d'un système automatisé pour la prise d'images RTI (Reflectance Transformation Imaging)

Meven Ricchieri
2023

Numéro de projet : 2309

Date de début : 07.08.2023
Date de fin : 11.09.2023

Durée : 5 semaines

Type de projet : Travail de diplôme
Responsable : M. Serge Castoldi
Mandant : MCAH / M. Mathieu Bernard-Reymond



Table des matières

1 Acronymes	4
2 Préface	5
3 Cahier des charges	6
3.1 Introduction	6
3.2 Principe de fonctionnement	6
3.3 Hardware	7
3.4 Firmware	7
3.5 Mécanique	7
4 Pré-étude	8
4.1 Introduction	8
4.2 Planning	8
4.3 Schéma bloc détaillé	9
4.4 Descriptions des blocs détaillées	9
4.4.1 Bloc Afficheur LCD	9
4.4.2 Bloc Interface de contrôle	9
4.4.3 Bloc Moteur de positionnement	10
4.4.4 Bloc Driver(s) moteur	10
4.4.5 Bloc LEDs de puissance	11
4.4.6 Bloc Driver(s) LEDs de puissance	11
4.4.7 Bloc LED IR	12
4.4.8 Bloc Driver LED IR	12
4.4.9 Bloc Optocoupleurs	12
4.4.10 Bloc Système d'indexage	12
4.4.11 Bloc Microcontrôleur	12
4.4.12 Bloc Alimentation secteur	13
4.4.13 Bloc Régulateur(s) de tension	13
4.5 Intégration mécanique	13
4.6 Estimation des coûts en CHF	14
4.7 Faisabilité du projet	14
5 Design électronique	15
5.1 Introduction	15
5.2 Système d'éclairage de puissance	15
5.2.1 Choix principaux	15
5.2.2 Dimensionnements	16
5.2.3 Schémas	17
5.3 Système de commande Nikon D750	18
5.3.1 Choix principaux	18
5.3.2 Dimensionnement	19
5.3.3 Schéma	20
5.4 Système de commande Sony A7R	20
5.4.1 Choix principaux	20
5.4.2 Dimensionnement	20
5.4.3 Schéma	21
5.5 Système de signalisation	21
5.5.1 Choix principaux	21
5.5.2 Dimensionnement	21
5.5.3 Schémas	22
5.6 Système d'indexage	22

5.6.1	Choix principal	22
5.6.2	Dimensionnement	23
5.6.3	Schéma	23
5.7	Écran LCD et rétro-éclairage	23
5.7.1	Choix principaux	23
5.7.2	Dimensionnement	24
5.7.3	Schéma	24
5.8	Système de rotation	25
5.8.1	Choix principaux	25
5.8.2	Dimensionnement	25
5.8.3	Schémas	27
5.9	Interface de commande	28
5.9.1	Choix principaux	28
5.9.2	Dimensionnement	28
5.9.3	Schéma	28
5.10	Système d'alimentation	28
5.10.1	Choix principaux	29
5.10.2	Dimensionnement	29
5.10.3	Schémas	29
5.11	Microcontrôleur	30
5.11.1	Choix principal	30
5.11.2	Dimensionnement	30
5.11.3	Schémas	31
6	Design circuit imprimé	32
6.1	Introduction	32
6.2	Caractéristiques du circuit	32
6.3	Placement et routage des composants	32
6.4	Améliorations	33
7	Design mécanique	34
7.1	Introduction	34
7.2	Design	34
8	Firmware MCU	35
8.1	Introduction	35
8.2	Configuration du system clock	35
8.3	Configuration des Timers	35
8.4	Configuration des MCPWMs	37
8.5	Configuration du SPI	38
8.6	Machine d'état principale	39
8.6.1	Détails	39
8.7	Gestion du moteur stepper	41
8.8	Gestion de la séquence de prise d'image	43
8.9	Gestion du menu	44
9	Mise en service	45
9.1	Introduction	45
9.2	Régulateurs de tension	45
9.2.1	Méthode de mesure	45
9.2.2	Schéma de mesure	45
9.2.3	Résultats	45
9.2.4	Analyse	46
9.3	Driver de moteur	46
9.3.1	Méthode de mesure	46

9.3.2	Schéma de mesure	47
9.3.3	Résultats	48
9.3.4	Analyse	51
9.4	Driver LED de puissance	52
9.4.1	Méthode de mesure	52
9.4.2	Schéma de mesure	52
9.4.3	Résultat	53
9.4.4	Analyse	53
10	Résultat final	54
10.1	Éléments fonctionnels	54
10.2	Éléments à corriger	54
10.3	Problèmes rencontrés	54
11	Conclusion	55
12	Annexes	56
12.1	Cahier des charges	57
12.2	Schémas	61
12.3	Drawings	64
12.4	Bill of material	66
12.5	Codes C et H	67
12.6	Document de modification	112
12.7	Planning	113
12.8	Journal de travail	114
12.9	Mode d'emploi	115
12.10	PV	117

1 Acronymes

CCW : Counter Clock Wise

CW : Clock Wise

IR : Infra Rouge

MCU : Microcontrôleur

MCPWM : Motor Control Pulse-Width Modulation

PCB : Printed Circuit Board

RTI : Reflectance Transformation Imaging

2 Préface

La RTI (Reflectance Transformation Imaging) est une technique photographique informatique qui saisit à la fois la forme et la couleur de la surface d'un sujet, permettant ainsi un ré-éclairage interactif du sujet à partir de toutes les directions possibles. De plus, la RTI offre la possibilité d'améliorer mathématiquement les caractéristiques de forme et de couleur de la surface du sujet. Les fonctions d'amélioration propres à la RTI permettent de révéler des informations de surface qui échappent à une simple observation empirique directe de l'objet physique. Les logiciels RTI actuels et les méthodologies associées ont été élaborés par une équipe internationale de développeurs.

Les images RTI résultent de l'analyse de multiples photographies numériques d'un sujet, capturées à partir d'un appareil photo fixe. Chaque prise de vue projette la lumière depuis une direction différente, précisément connue ou déductible. Cette procédure engendre une séquence d'images du même sujet, chacune présentant des variations d'ombres et de lumières. L'information lumineuse de ces images est synthétisée de façon mathématique pour créer un modèle de surface, permettant ainsi à l'utilisateur de ré-éclairer l'image RTI de manière interactive et d'examiner sa surface sur un écran.

Chaque image RTI ressemble à une photographie conventionnelle en deux dimensions (2D). Cependant, contrairement à une photographie classique, les informations de réflectance sont extrapolées à partir de la forme tridimensionnelle (3D) du sujet et encodées dans chaque pixel de l'image. Cela permet à l'image RTI synthétisée d'incorporer une connaissance intrinsèque sur la manière dont la lumière se reflète sur le sujet. Lorsque l'image RTI est visualisée à l'aide d'un logiciel dédié, chaque pixel peut réagir à une lumière "virtuelle" interactive provenant de différentes positions sélectionnées par l'utilisateur. Ce jeu dynamique d'ombres et de lumières dans l'image dévoile les subtils détails de la forme tridimensionnelle de la surface du sujet.

Texte tiré du site web culturalheritageimaging.org et traduit de l'anglais.

3 Cahier des charges

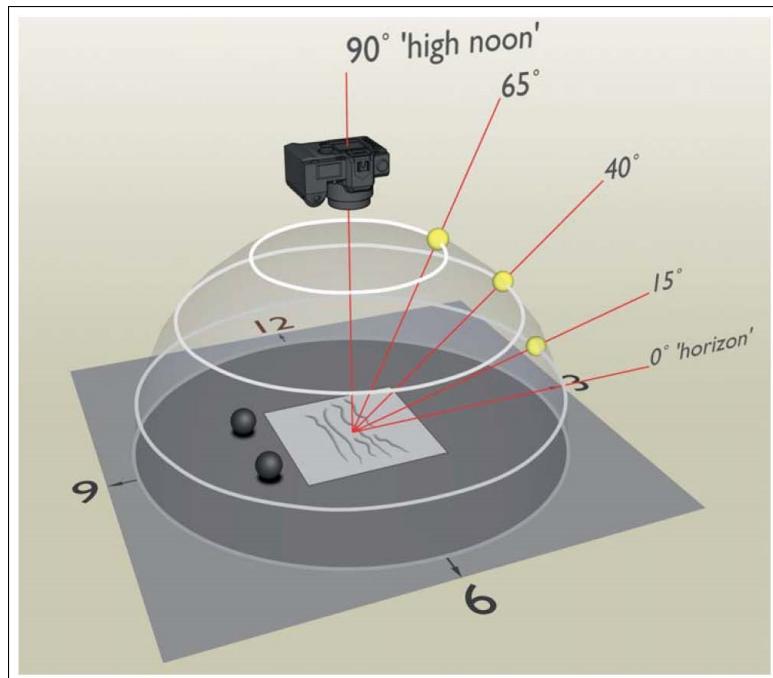
3.1 Introduction

Ce projet de diplôme a pour objectif de concevoir un système automatisé pour faciliter la prise d’images RTI (Reflectance Transformation Imaging). Le travail est segmenté en deux parties distinctes : la première concerne la réalisation électronique tandis que la seconde porte sur la réalisation mécanique. La partie électronique du projet sera prise en charge par moi-même, tandis que la partie mécanique sera réalisée par un diplômant en génie mécanique. L’électronique du système sera responsable de plusieurs tâches essentielles, notamment la gestion des LEDs de puissance pour un éclairage optimal de la cible lors des prises de vue, la commande du moteur pour le positionnement précis, ainsi que la coordination de la séquence et le déclenchement de l’appareil photo.

3.2 Principe de fonctionnement

En supplément de la partie électronique, le système comprendra une armature ainsi qu’un bras motorisé, développé comme mentionné précédemment, par un étudiant en génie mécanique. Les LED haute puissance seront montées sur le bras motorisé de manière à éclairer la cible selon des angles précisément déterminés. Le microcontrôleur sera en charge des mouvements du bras, assurant ainsi une rotation complète avec des arrêts à intervalles réguliers programmables. Après chaque immobilisation du bras, une série de 4 à 5 prises de vue sera réalisée avec activation individuelle d’une seule LED, avant de procéder au changement d’angle suivant. Sur la Figure 1, un schéma de principe est illustré afin de mieux comprendre comment cela fonctionne théoriquement. Sur le plan horizontal une représentation des heures (0 à 12) correspondent aux 360° que le bras motorisé doit parcourir afin de réaliser une séquence d’acquisition d’images complète. Les droites rouges correspondent aux emplacement des différentes LEDs.

FIGURE 1 – Schéma de principe



3.3 Hardware

- Éléments demandés :
 - * Alimentation du système via bloc secteur tension standard DC (5V, 12V ou 24V)
 - * Contrôle du système à l'aide d'un afficheur LCD alphanumérique et des boutons et/ou encodeur rotatif
 - * Buzzer permettant de signaler différents états du système
 - * Connectique robuste, détrompée et simple d'utilisation
- Éléments pilotés :
 - * Moteur de positionnement (choix et dimensionnement en collaboration avec l'étudiant en mécanique selon couple)
 - ◊ La position de l'arbre du moteur doit être connue par le MCU
 - * LEDs de puissance pour l'éclairage de la cible
 - ◊ Nombre de LED à décider entre 4 et 5
 - ◊ Évaluer si les LEDs LXML-PWC2 peuvent être remplacées par d'autres plus puissantes
 - * Appareil photo
 - ◊ Déclenchement commandé via câble optocouplé pour Nikon D750
 - ◊ OPTIONNEL Déclenchement commandé via infrarouge pour Sony A7R

3.4 Firmware

- Éléments demandés :
 - * Contrôle du système simple et intuitif
 - * Émettre un son lorsque la séquence est terminée
 - * Sauvegarde des réglages dans une mémoire non volatile
 - * Réglage des paramètres de la séquence
 - ◊ Temps d'allumage des LEDs pour adaptation au temps de pose
 - ◊ Temps entre chaque changement de LED
 - * 2 modes de fonctionnement
 - ◊ Automatique (séquence complète environ 200 photos)
 - ◊ Manuel
 - * Déclenchement appareil photo Nikon D750
 - * OPTIONNEL Déclenchement appareil photo Sony A7R

3.5 Mécanique

- Élément demandé :
 - * Mise en boîtier

4 Pré-étude

4.1 Introduction

Cette phase de pré-étude consiste à explorer en détail les composants principaux qui constitueront ce projet afin de bien comprendre les différentes contraintes et exigences. Elle permet également d'imaginer un concept réalisable et optimal du produit de sorte à ce que lors de la phase de design, les éléments bloquants soient déjà éliminés. Cela permettra à terme, d'obtenir un produit final fonctionnel.

4.2 Planning

Après avoir examiné le cahier des charges et analysé ses exigences, j'ai établi un planning réaliste pour gérer efficacement le temps qui m'est imparti pour le développement de ce projet. La planification finale du projet sera incluse en annexe, avec des cases mises en évidence en gras là où le travail a été effectivement réalisé. Cela permettra de comparer la planification initiale avec les réalisations concrètes.

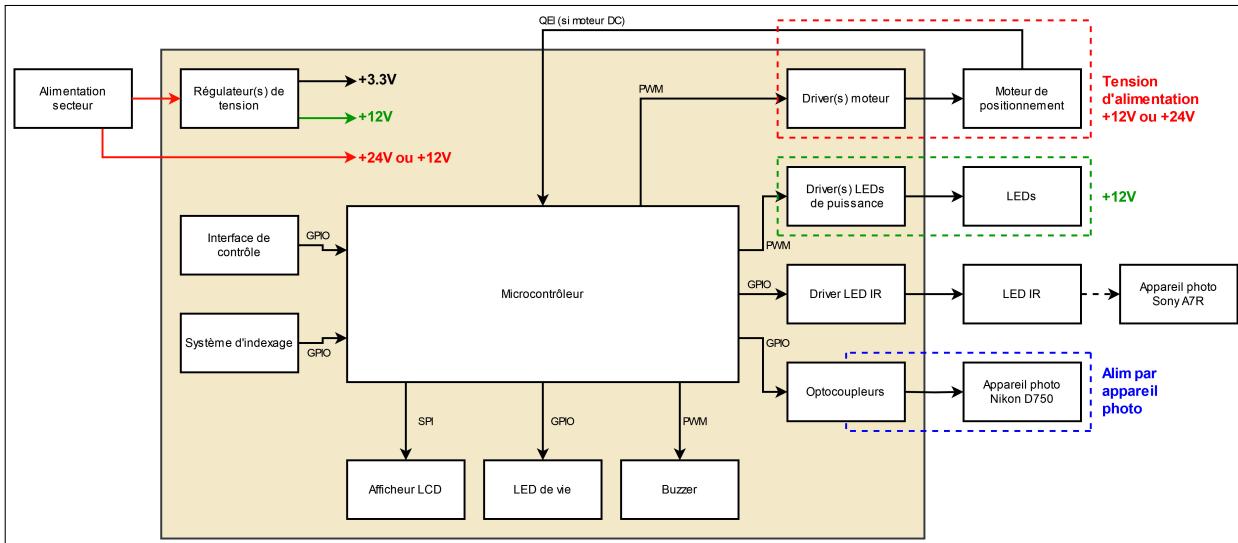
FIGURE 2 – Planning

No jour de travail	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	29	30	31	32	33	34	35	36
Date	lundi, 7. août 2023	mardi, 8. août 2023	mercredi, 9. août 2023	jeudi, 10. août 2023	vendredi, 11. août 2023	samedi, 12. août 2023	dimanche, 13. août 2023	lundi, 14. août 2023	mardi, 15. août 2023	mercredi, 16. août 2023	jeudi, 17. août 2023	vendredi, 18. août 2023	samedi, 19. août 2023	dimanche, 20. août 2023	lundi, 21. août 2023	mardi, 22. août 2023	mercredi, 23. août 2023	jeudi, 24. août 2023	vendredi, 25. août 2023	samedi, 26. août 2023	dimanche, 27. août 2023	lundi, 28. août 2023	mardi, 29. august 2023	mercredi, 30. août 2023	jeudi, 31. août 2023	vendredi, 1. septembre 2023	samedi, 2. septembre 2023	dimanche, 3. septembre 2023	lundi, 4. septembre 2023	mardi, 5. septembre 2023	mercredi, 6. septembre 2023	jeudi, 7. septembre 2023	vendredi, 8. septembre 2023	samedi, 9. septembre 2023	dimanche, 10. septembre 2023	lundi, 11. septembre 2023
Cahier des charges																																				
Pré-étude																																				
Dimensionnement + design + schéma																																				
Design du PCB + BOM																																				
Design de des parties mécaniques																																				
Montage PCB																																				
Réalisation du software																																				
Mise en service et tests																																				
Rédaction rapport																																				
Finalisation/corrections/documentation																																				

4.3 Schéma bloc détaillé

Le schéma bloc détaillé illustré dans la Figure 3 offre une meilleure compréhension de l'interaction entre tous les composants. Il met également en évidence les composants présents sur le circuit imprimé et ceux situés à l'extérieur du circuit.

FIGURE 3 – Schéma bloc détaillé



Pour une meilleure lisibilité du schéma, celui-ci se trouve en annexe.

4.4 Descriptions des blocs détaillées

Dans cette section, chaque bloc du schéma est décrit de sorte à comprendre quelles sont les contraintes et quelles sont les solutions possibles.

4.4.1 Bloc Afficheur LCD

Conformément aux directives du CDC, il est nécessaire que l'écran LCD soit de nature alphanumérique plutôt que graphique. Cette spécification a été choisie pour faciliter considérablement la gestion de l'affichage au niveau du logiciel. Un écran à quatre lignes est un choix idéal pour ce produit, car il s'aligne bien avec son utilisation prévue, qui reste simple en termes de configuration. Selon la sélection de l'afficheur, il pourra être nécessaire d'intégrer un rétro-éclairage. Cette fonctionnalité sera essentielle pour utiliser le système en cas de faible luminosité. L'utilisation d'un écran LCD réfléchissant (reflective LCD) pourrait s'avérer limitée.

4.4.2 Bloc Interface de contrôle

L'idée est de contrôler le démarrage d'une séquence de prise de 5 images au moyen d'un simple bouton poussoir lorsque le mode manuel a été activé. La gestion du menu, des configurations et du moteur sera réalisée à l'aide d'un encodeur rotatif avec un bouton poussoir intégré. L'utilisation de cet encodeur simplifiera considérablement le processus de réglage des paramètres par rapport à l'utilisation de boutons traditionnels.

4.4.3 Bloc Moteur de positionnement

En raison de la réalisation de mon diplôme en avance, la sélection du moteur ne peut être effectuée actuellement (08.08.2023) car ce choix doit être pris conjointement avec l'étudiant responsable de la partie mécanique qui lui effectue un diplôme suivant le calendrier régulier. La raison en est qu'il est nécessaire de déterminer le couple requis par le moteur pour accomplir ses fonctions de manière optimale

Il existe deux variantes de technologies de moteurs appropriées. Ces deux options comprennent d'une part les moteurs pas à pas (stepper), et d'autre part les moteurs à courant continu (DC) équipés d'un encodeur rotatif. Selon moi l'utilisation d'un moteur pas à pas est préférable à un moteur à courant continu (DC) avec encodeur. Cela s'explique par le fait qu'un moteur pas à pas peut fonctionner en boucle ouverte, ce qui signifie qu'il ne nécessite pas de régulation de position complexe. Le fait de manquer quelques pas sous charge ne pose pas de problème, car la précision n'est pas un élément critique. De plus, une opération d'indexation sera effectuée avant le début de chaque séquence complète, ce qui évitera l'accumulation d'erreurs.

4.4.4 Bloc Driver(s) moteur

Comme la technologie du moteur ainsi que le choix spécifique n'ont pas encore été déterminés, j'ai sollicité Mike Jaton, le responsable de l'étudiant en mécanique, pour obtenir une estimation de la puissance minimale requise. Cette information me sera très utile pour la décision du modèle de driver. L'estimation de la puissance minimale est de 50 watts, ce qui signifie que le driver devra être capable au minimum fournir cette puissance au moteur. A l'aide de la formule 1, j'ai pu obtenir le courant minimum que le driver devra être capable de délivrer en fonction du choix de la tension d'alimentation du système.

$$I = \frac{P}{U} \quad (1)$$

où :

- P = Puissance min (W)
- U = Tension d'alimentation (V)
- I = Courant de sortie min (A)

Les résultats sont documentés dans la Table 1.

TABLE 1 – Table de résultats

Puissance	Tension d'alimentation	Courant de sortie
50W	5V	10A
50W	12V	4.16A
50W	24V	2.083A

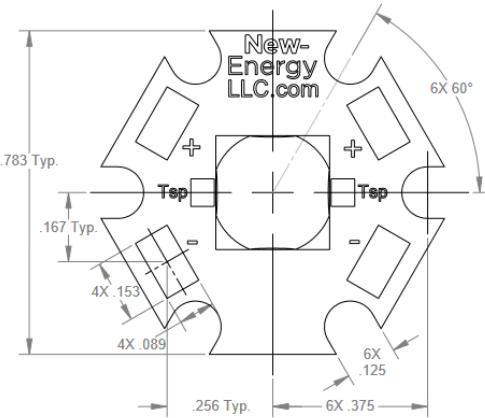
Comme la technologie du moteur n'a pas encore été sélectionnée, j'ai opté pour l'utilisation d'un driver capable de contrôler à la fois un moteur pas à pas et un moteur à courant continu. Cette approche facilitera le choix ultérieur du moteur. Si un moteur pas à pas est choisi, son contrôle adéquat nécessitera l'utilisation de deux ponts en H. En revanche, si un moteur à courant continu est utilisé, un seul pont en H sera requis. En conséquence, il faudra soit un unique driver équipé de deux ponts en H, soit deux drivers distincts, chacun avec un pont en H.

4.4.5 Bloc LEDs de puissance

Selon le CDC, il est fortement conseillé de remplacer les LEDs actuellement utilisées par d'autres plus puissantes. Comme le système final aura des dimensions supérieures à celles de la configuration actuelle, les LEDs seront obligatoirement plus éloignées de l'élément visé. En conséquence, le flux lumineux devra être plus élevé. De plus, il est demandé que le temps d'exposition ne dépasse pas la seconde pour chaque cliché.

Les LEDs devront également être déjà monté sur des PCB aluminium afin qu'elles puissent dissiper correctement la chaleur et surtout qu'elles puissent se fixer facilement au bras mécanisé à l'aide de vis standard. La Figure 4 présente une illustration d'un exemple de modèle de carte de circuit imprimé en aluminium pouvant être utilisé.

FIGURE 4 – Exemple d'un modèle adéquat (StarBoardLED)



Certains aspects stipulés ci-dessous dans la Table 2 ne sont pas mentionnés dans le CDC mais découlent d'un échange de mails avec Monsieur Mathieu Bernard-Reymond, mandant du projet.

TABLE 2 – Valeurs demandées

	Valeurs idéales
Température de couleur	5500K - 6500K
Flux lumineux	>200lm
ICR	≥ 70
Angle de visualisation	≥ 120

D'après mes recherches, les LEDs de puissance répondant aux spécifications du cahier des charges peuvent fonctionner avec une tension de 12V, ce qui permet de considérablement réduire le courant. C'est la raison pour laquelle j'ai choisi d'alimenter ces composants à une tension de 12V.

4.4.6 Bloc Driver(s) LEDs de puissance

Le pilote devra avoir la capacité de fournir le courant et la tension requis par les LEDs qui seront sélectionnées. L'idée consiste à intégrer un driver de LED existant en multiplexant sa sortie sur les 4 ou 5 LEDs de puissance, de manière à utiliser un seul circuit intégré et 5 transistors N-MOSFETs, ce qui devrait en principe être plus économique que d'utiliser 5 drivers séparés.

4.4.7 Bloc LED IR

Après de nombreuses recherches, je n'ai malheureusement pas trouvé d'informations précises concernant la longueur d'onde correspondant à la photodiode de l'appareil photo Sony A7. Par conséquent, il sera nécessaire d'effectuer des tests réels en utilisant plusieurs LEDs ayant toutes des longueurs d'onde différentes pour parvenir au résultat souhaité. Toutefois, la longueur d'onde la plus courante pour les systèmes de commande à distance est d'environ 950 nm.

4.4.8 Bloc Driver LED IR

Le contrôle de la LED infrarouge sera réalisé au moyen d'un N-MOSFET de faible puissance. Un composant supportant un courant drain source d'au minimum 200mA sera parfaitement adaptée, étant donné que les LEDs IR ont une consommation moyenne maximale de 100mA.

4.4.9 Bloc Optocoupleurs

Selon le cahier des charges, il est impératif d'assurer une isolation électrique de l'appareil photo Nikon D750 vis-à-vis du système, afin de le préserver de tout éventuel défaut. Cette exigence peut être satisfaite en ajoutant des optocoupleurs à usage général, lesquels fourniront une isolation efficace et sécurisée entre les deux appareils.

4.4.10 Bloc Système d'indexage

Pour établir la position précise du bras mécanisé au démarrage et avant chaque séquence complète, le microcontrôleur devra initier une rotation du bras vers un dispositif d'indexage. Ce mécanisme peut prendre diverses formes, telles qu'un capteur de fin de course, un contact Reed ou un capteur à effet Hall, pour garantir que le bras atteigne une position déterminée avec exactitude. Dans ce projet, un contact Reed sera le plus adapté du fait qu'il n'y aura pas de contact physique et une mise en place simple.

4.4.11 Bloc Microcontrôleur

En tenant compte des technologies évoquées précédemment, un microcontrôleur appartenant à la famille des Motor Control (PIC32MKxxxxMCF) s'avère être un choix approprié car ce MCU devra gérer entre 5 et 7 PWM, selon le futur choix du moteur et peut être un module QEI. La Table 3 répertorie les diverses configurations de signaux qui pourraient être employées dans le design finale afin de se faire une première idée de la configuration minimale du MCU.

TABLE 3 – Signaux MCU

Système à commander	Nombre d'I/O	Type d'I/O
Buzzer	1	PWM
LEDs de puissance	1	PWM
Commande moteur	4 - 6	PWM
Encodeur moteur	0 - 3	QEI
Afficheur LCD	3	SPI
Autres	15	GPIO

4.4.12 Bloc Alimentation secteur

Conformément aux directives du CDC, l'alimentation du système devra se faire au moyen d'un bloc secteur. Après avoir mené des recherches approfondies sur les composants évoqués précédemment, j'ai finalement décidé de développer un système flexible permettant d'alimenter le circuit soit en 24V soit en 12V. Cela permettra d'avoir une grande liberté au moment du choix du moteur. Ce bloc devra délivrer au minimum 6A si celui-ci fourni une tension de 12V ou alors 4A s'il délivre une tension de 24V. Cette estimation prend en compte le courant du moteur, des LEDs de puissance (1 LED allumée à la fois) et le circuit très basse tension (3.3V). En plus de cela, une marge de sécurité est ajoutée à cette estimation.

4.4.13 Bloc Régulateur(s) de tension

Afin de choisir et de dimensionner correctement le ou les régulateurs de tension, il faut tout d'abord estimer la consommation maximale des composants fonctionnant avec une tension de 3.3V et celle utilisant une tension de 12V si le montage final est alimenté en 24V. Dans ce projet, aucun composant ne sera particulièrement sensible aux perturbations électromagnétiques, ce qui fait qu'une alimentation à découpage de type step-down sera idéale puisque la différence de tension est assez élevée, soit 12V à 3.3V soit de 24V à 12V puis à 3.3V. Dans la Table 4 nous trouvons la consommation maximale totale des composants fonctionnant en 3.3V.

TABLE 4 – Table des consommations électrique 3.3V

Composant	Max
Microcontrôleur	50mA
LCD	0.25mA
Rétro-éclairage	100mA
Buzzer	30mA
LED IR	100mA
LED de vie	15mA
Optocoupleurs	50mA
	345.25mA

Un régulateur de tension 3.3V pouvant fournir un courant d'au moins 500mA sera parfaitement adapté. Dans le cas où l'alimentation du circuit se fera avec une tension de 24V, le régulateur 24-12V devra être capable de délivrer au moins 2A afin qu'il puisse fournir aux LEDs d'éclairage la puissance nécessaire.

4.5 Intégration mécanique

Comme une mise en boîtier m'est demandé, j'ai décidé de dessiner et d'imprimer en PLA mon propre boîtier à la place d'acheter un modèle existant et de l'usiner. Mes compétences en conception 3D me permettent de créer facilement ce type de boîtier et cela garantira une intégration plus facile du PCB et de la connectique, répondant ainsi précisément à mes besoins. Aucune contrainte de taille ne m'est imposée, étant donné que la machine sera de grande envergure. Cependant, le PCB et le boîtier seront de dimensions modestes, ce choix réduira le coût de fabrication du circuit.

4.6 Estimation des coûts en CHF

Ci-dessous, se présente une approximation des dépenses minimales et maximales impliquées dans la conception de ce projet au niveau du système de commande électronique. Si nous prenons le juste milieu de ces deux valeurs, nous obtenons un prix d'environ 365.-.

TABLE 5 – Table de prix

Composant	Min	Max
Alimentation secteur	30.-	50.-
Régulateurs de tension	10.-	15.-
Microcontrôleur	3.-	5.-
Système d'indexage	1.-	10.-
Afficheur LCD	20.-	40.-
LEDs	40.-	60.-
Driver(s) LEDs	10.-	20.-
Moteur	20.-	150.-
Driver(s) moteur	10.-	20.-
Autres composants	15.-	40.-
PCB	50.-	100.-
Boîtier	3.-	7.-
	212.-	517.-

4.7 Faisabilité du projet

Selon les données recueillies, le projet est tout à fait réalisable dans son intégralité. Cependant, des questions subsistent concernant le moteur, bien que cela ne compromette pas sa faisabilité.

5 Design électronique

5.1 Introduction

Dans cette section, nous allons examiner en détail les différentes étapes qui composent la conception du système. Cela englobe la sélection des composants électroniques et quelques fois mécaniques, la phase de dimensionnement et la création des schémas. Les éléments présentés ici ne seront plus systématiquement subdivisés selon le schéma bloc, comme cela a été fait jusqu'à maintenant, mais seront regroupés en ensembles, permettant ainsi une meilleure compréhension.

5.2 Système d'éclairage de puissance

Le système d'éclairage est constitué de 3 parties. La première étant le driver de LED permettant de générer un courant constant et un réglage de l'intensité lumineuse par switching. La seconde partie étant le multiplexage du driver de LED à l'aide de N-MOSFETs et la troisième correspond au connecteur permettant au branchement des 5 LEDs qui seront déportées du PCB et fixées sur le bras mécanisé.

5.2.1 Choix principaux

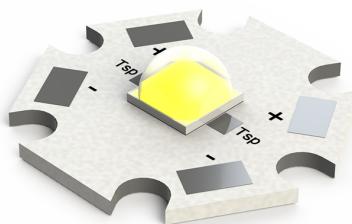
Choix des LEDs :

Les LEDs de puissance choisies pour l'éclairage de la cible, dont les caractéristiques sont répertoriées dans la Table 6, seront capables de produire un flux lumineux pouvant atteindre plus de 2000 lumens. La référence XHP50B-00-0000-0D0BJ40CB correspond à la LED en elle même et la référence se terminant par SB01 correspond à la LED monté sur un PCB aluminium StarBoardLED.

TABLE 6 – Caractéristiques importantes de la LED XHP50B-00-0000-0D0BJ40CB

Type	Valeur
Couleur	Blanc cool
CCT	6500K
Lumens/Watt	148lm/W
Flux lumineux @ 1.5A	2088lm
CRI	70
Angle de vue	120°
Courant max	1.5A
Tension d'alimentation	6V/12V

FIGURE 5 – XHP50B-00-0000-0D0BJ40CB-SB01



Choix du driver de LED :

Le driver de LED à tout d'abord été le MAX16832AASA+T mais le schéma élaboré posait un problème au niveau de la commande des MOSFETs. En conséquence, je l'ai remplacé par un modèle plus puissant capable de fournir un courant de 1,5A, afin de tirer pleinement parti de la puissance des LEDs sélectionnées. Le driver choisi s'intitule TPS92200D1DDCR. Il génère un courant constant pouvant aller jusqu'à 1.5A et possède une entrée logique de variation d'intensité (dimming). Il est constitué de deux MOSFETs qui, une fois combinés, présentent une résistance RDs(ON) de 240mΩ, ce qui se traduit par une chute de tension faible mais néanmoins présente, d'environ 360 mV à pleine puissance.

Choix des MOSFETs :

Les MOSFETs sélectionnés pour permettre le multiplexage du courant constant est le PMV20ENR. Il s'agit de MOSFETs à canal N ayant une capacité maximale de courant pouvant circuler à travers eux de 6A, ce qui dépasse largement le courant réellement prévu pour les traverser. Ces composants ont l'avantage d'être très bon marché, environ 30 centimes l'unité. De ce fait, j'ai décidé de les utiliser pour tout les éléments de commutation de ce circuit afin de réduire le nombre de référence.

TABLE 7 – Caractéristiques importantes du N-MOSFET PMV20ENR

Type	Valeur
Courant drain max continu	6A
Tension de seuil min	1V
Tension de seuil max	2V
Puissance max	510mW
RDS(ON) @ VGS = 3.3V	30mΩ

5.2.2 Dimensionnements

Les formules utilisées dans cette section proviennent tous de la datasheet du composant TPS92200D1DDCR.

Calcul de Rsense :

Le dimensionnement de la résistance Rsense permet de fixer le courant, donc les 700mA qui doivent traverser la LED. Elle se calcule avec la formule 2.

$$R_{SENSE} = \frac{V_{F_BREF}}{I_{LED}} \quad (2)$$

où :

- V_{F_BREF} = Référence de tension donné par le fabricant (V)
- I_{LED} = Courant dans la LED (I)
- R_{SENSE} = Résistance de mesure (Ω)

TABLE 8 – Valeurs utilisées et résultat

V_{F_BREF}	99mV
I_{LED}	1.5A
R_{SENSE}	<u>66mΩ</u>

Étant donné la difficulté à trouver une résistance de 66mΩ, j'ai opté pour une résistance de 68mΩ dans mon circuit.

Calcul de l'inductance :

Pour que le chip puisse réguler le courant à la valeur calculée précédemment, il a besoin d'un certaine fréquence d'opération qui se fixe à l'aide d'une inductance. Cette inductance se calcule à l'aide de la formule 3. La fréquence quant à elle peut monter jusqu'à 2MHz.

$$L = \frac{V_{OUT} * (V_{IN(MAX)} - V_{OUT})}{V_{IN(MAX)} * K_{IND} * I_{LED} * f_{SW}} \quad (3)$$

où :

- V_{OUT} = Tension de sortie (V)
- $V_{IN(MAX)}$ = Tension d'alimentation max (V)
- I_{LED} = Courant max LED (A)
- f_{SW} = Fréquence d'opération (Hz)
- K_{IND} = Coefficient courant d'ondulation (-)
- L = Valeur de l'inductance (H)

TABLE 9 – Valeurs utilisées et résultat

V_{OUT}	11.6V
$V_{IN(MAX)}$	12V
I_{LED}	1.5A
f_{SW}	1MHz
K_{IND}	0.2
L	<u>1.28μH</u>

La valeur finale sera de $1.2\mu\text{H}$, De ce fait le choix sera bien plus grand et de toute manière la tolérance de ce genre de composant est très grande, de l'ordre de $\pm 20\%$.

5.2.3 Schémas

FIGURE 6 – Schéma du système d'éclairage LED - partie A

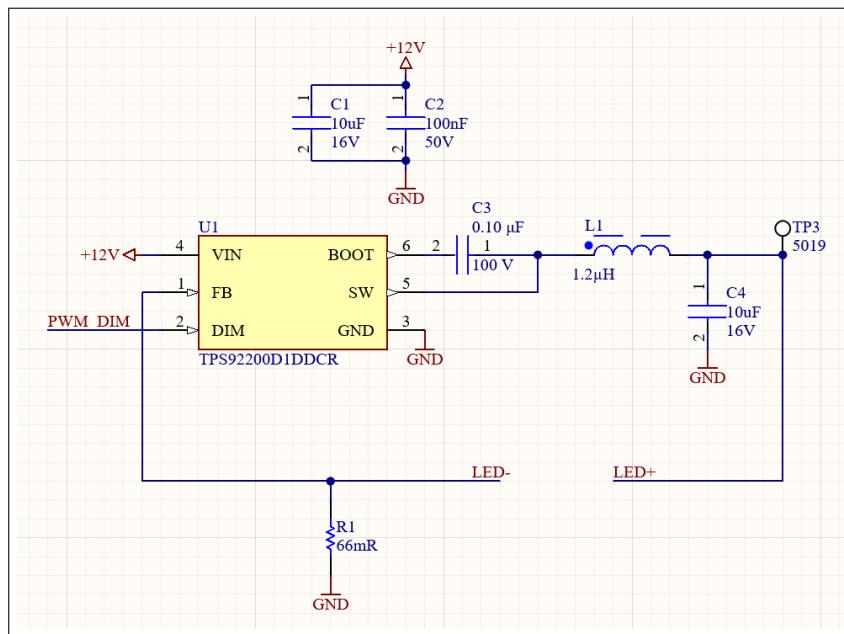


FIGURE 7 – Schéma du système d'éclairage LED - partie B

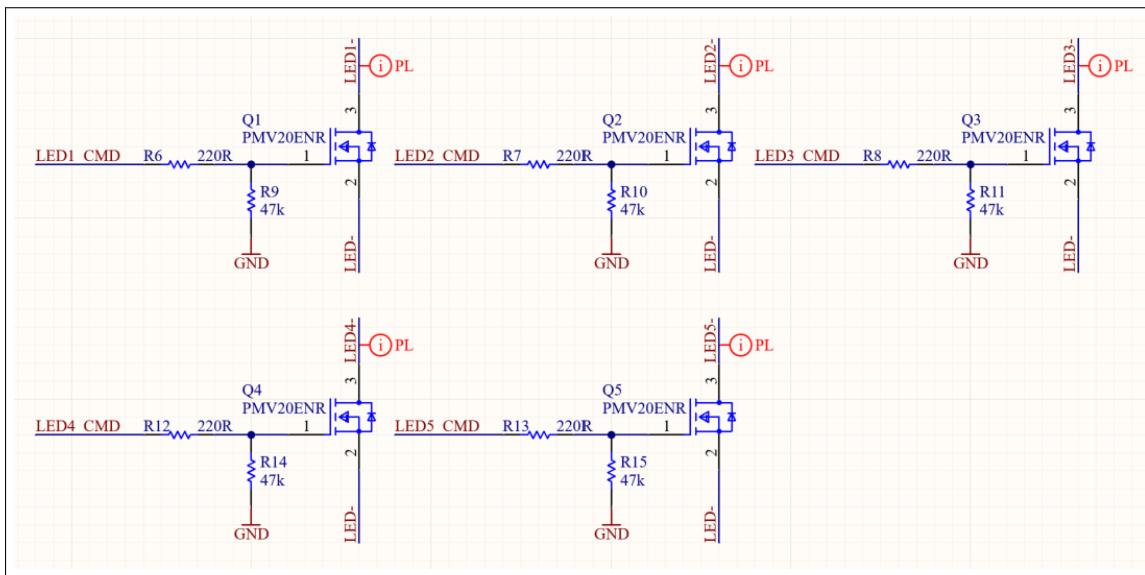
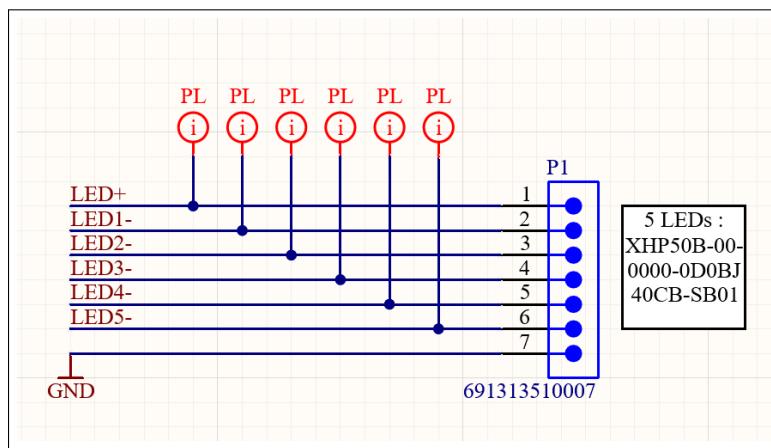


FIGURE 8 – Schéma du système d'éclairage LED - partie C



5.3 Système de commande Nikon D750

Ne connaissant pas le domaine, j'ai effectué différentes recherches afin de comprendre comment un appareil photo, tel que le Nikon D750, se déclenche à distance à l'aide d'un câble. La documentation du fabricant étant absente à ce niveau là, j'ai trouvé quelques informations intéressantes sur des forums de photographes. Grâce à cela j'ai pu comprendre le fonctionnement et ainsi créer un schéma.

5.3.1 Choix principaux

Choix des optocoupleurs :

L'objectif est d'établir la connexion entre la ligne de déclenchement ou de mise au point et la référence commune (GND), le tout étant commandé par le MCU qui est isolé de l'appareil. Dans cette optique, des optocoupleurs polyvalents tels que les 4N25M se révèlent être un choix parfaitement approprié.

Choix du connecteur :

La liaison entre l'appareil photo et le système s'établit à l'aide d'un câble DC2 vers Jack 3.5 mm. En conséquence, le connecteur intégré sur le PCB sera de type Jack 3.5 mm à 3 contacts.

5.3.2 Dimensionnement

Calcul de la résistance de limitation :

La LED intégrée aux optocoupleurs présente une tension directe de 1,2V, tandis que j'ai établi le courant à 10mA. Cette configuration résulte de la nécessité de simplement connecter les deux lignes, sans qu'aucun courant ne soit supposé circuler à travers le phototransistor.

$$R_{limitation} = \frac{V_{CC} - V_F}{I_F} \quad (4)$$

où :

V_{CC}	= Tension d'alimentation (V)
V_F	= Tension directe de la LED (V)
I_F	= Courant de la LED (A)
$R_{limitation}$	= Résistance de mesure (Ω)

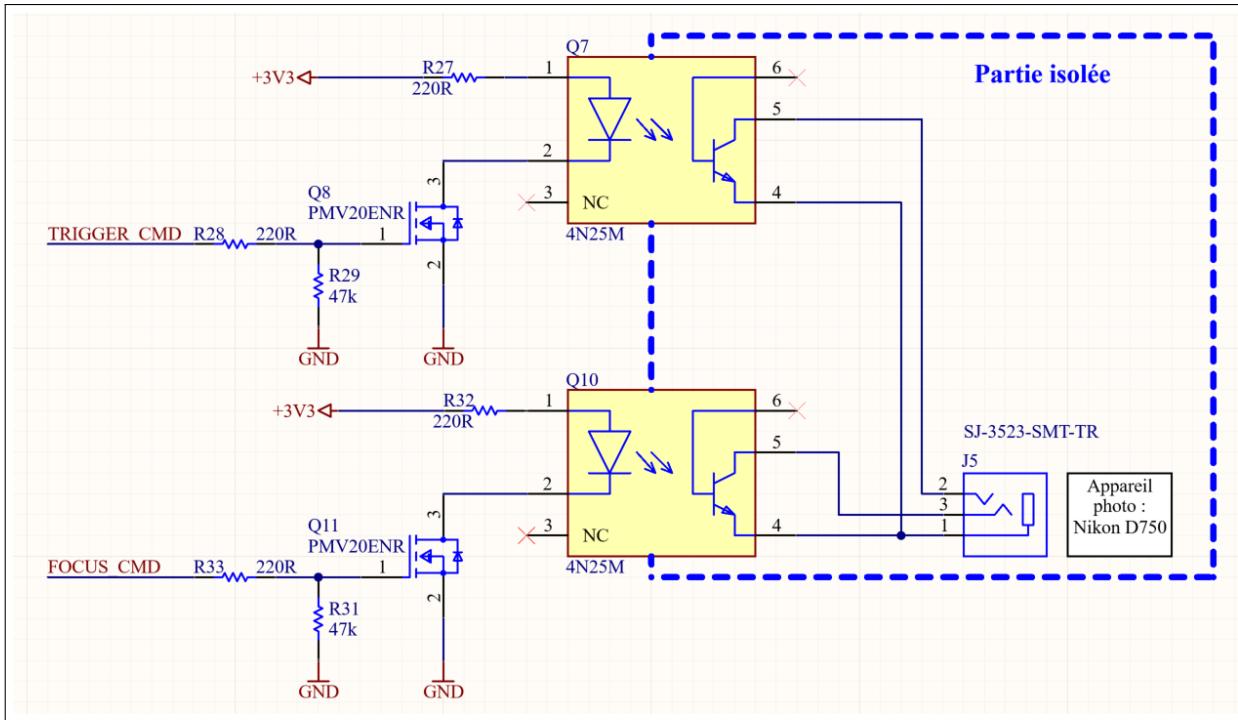
TABLE 10 – Valeurs utilisées et résultat

V_{CC}	3.3V
V_F	1.2V
I_F	10mA
$R_{limitation}$	<u>210</u> Ω

La valeur finale sera de 220Ω , De ce fait les résistances se trouvent dans la série E24.

5.3.3 Schéma

FIGURE 9 – Schéma du système de commande Nikon D750



5.4 Système de commande Sony A7R

Ayant déjà mentionné ce détail plus tôt dans ce rapport, aucune information de longueur d'onde m'est disponible pour la conception de cette partie. De ce fait, j'ai tout de même décidé de la mettre en place et de tester le système une fois terminé. De plus cette partie est optionnelle donc le mauvais fonctionnement de cette partie n'est pas critique.

5.4.1 Choix principaux

Choix de la LED IR :

La LED infrarouge CQY36N s'avère idéale dans le cas où la longueur d'onde de 950nm est correcte. Celle-ci à un angle de visualisation de 110° ce qui offre une certaine liberté au niveau du positionnement de la LED sur le boîtier ou à l'extérieur du boîtier.

Choix du MOSFET :

Étant donné que le courant traversant la LED IR sera de 100 mA, le N-MOSFET sélectionné pour les LEDs de puissance, mentionnées précédemment dans ce document, s'avère être une solution appropriée.

5.4.2 Dimensionnement

Calcul de la résistance de limitation :

La LED IR a une tension directe de 1.3V et j'ai fixé le courant à 100mA afin que l'émission soit la meilleure possible.

Avec la formule 4 :

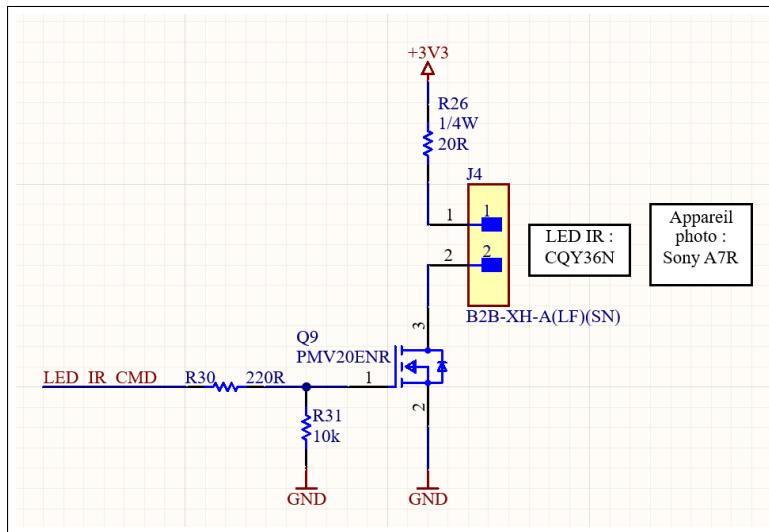
TABLE 11 – Valeurs utilisées et résultat

V_{CC}	3.3V
V_F	1.3V
I_F	100mA
$R_{limitation}$	<u><u>20Ω</u></u>

Cette résistance devra être d'une puissance d'au moins 1/4 de watt pour ne brûler.

5.4.3 Schéma

FIGURE 10 – Schéma du système de commande Sony A7R



5.5 Système de signalisation

5.5.1 Choix principaux

Choix de la LED de vie :

La LED de vie a la référence suivante, LH R974-LP-1. Il s'agit d'une LED rouge disponible dans le stock de l'école.

Choix du buzzer :

Le buzzer choisi, le PKMCS0909E4000-R1, est de type SMD et possède une épaisseur particulièrement réduite. Cette caractéristique permettra de le placer sur la face avant par rapport au boîtier, ce qui vise à améliorer la propagation de l'onde sonore.

5.5.2 Dimensionnement

Le buzzer sélectionné fonctionne avec une tension d'alimentation peut aller jusqu'à 12.5V donc il ne nécessite pas de résistance de limitation. J'ai toutefois placé une résistance de 0Ω afin de pouvoir en braser un si le son est trop fort ou qu'un soucis de consommation se présente.

Calcul de la résistance de limitation de la LED :

La LED étant simplement là pour aider à la réalisation du software, j'ai décidé de l'allumer avec un courant de 10mA. A l'aide de la formule 4, j'ai obtenu une résistance de limitation de 150Ω .

5.5.3 Schémas

FIGURE 11 – Schéma

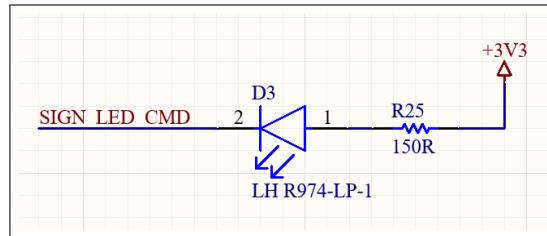
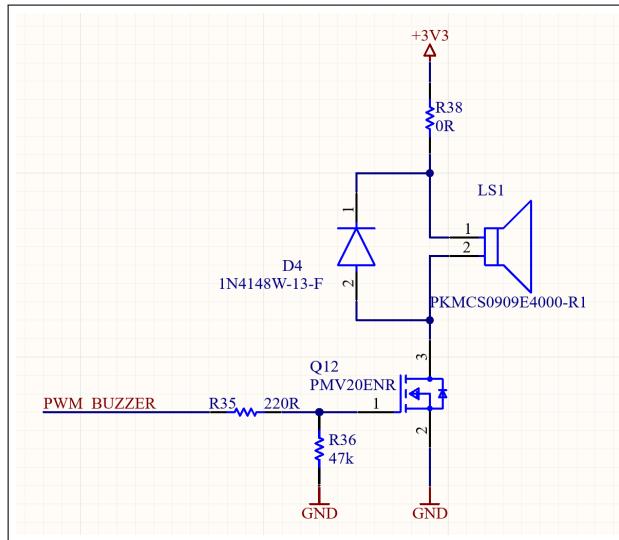


FIGURE 12 – Schéma



5.6 Système d'indexage

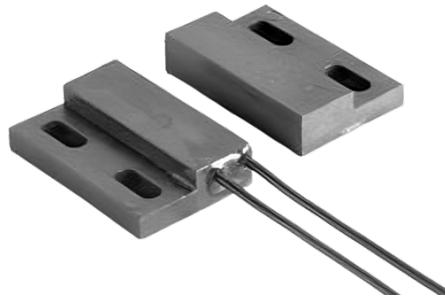
Comme mentionné dans la pré-étude, la détection du point zéro (index) sera réalisé au moyen d'un contact Reed et d'un aimant. Cette décision vient du fait que cette option ne nécessite pas de contact physique entre le bras et un autre composant monté sur l'armature de la machine.

5.6.1 Choix principal

Choix du contact Reed :

Le contact Reed, référencé sous 59145-010, et son aimant, référencé sous 57150-000, sont conçus pour être facilement montés sur des pièces mécaniques à l'aide de vis M3. Une illustration de ce composant et de l'aimant est présentée sur la Figure 13.

FIGURE 13 – Illustration du contact Reed 59145-010 et son aimant 57150-000

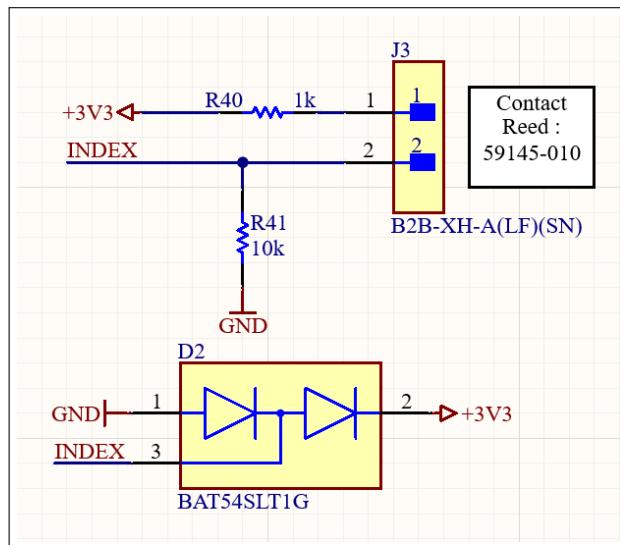


5.6.2 Dimensionnement

Aucun calcul de dimensionnement n'a été requis. La seule particularité est la connexion directe à une entrée GPIO du MCU, ce qui nécessite des mesures de sécurité en raison de la longueur variable de la ligne qui dépendra de la conception mécanique. Raison pour laquelle deux diodes de protection sont présentes sur le schéma.

5.6.3 Schéma

FIGURE 14 – Schéma du système d'indexage



5.7 Écran LCD et rétro-éclairage

5.7.1 Choix principaux

Choix de l'écran LCD :

La référence du LCD sélectionné est la suivante : EA DOGM204W-A. Cet écran à l'avantage d'être extrêmement fin et peut être contrôlé par une communication série telle que l'I2C ou le SPI. Ce modèle est de type transflectif, ce qui signifie qu'il peut être utilisé avec ou sans rétro-éclairage.

Choix du rétro-éclairage :

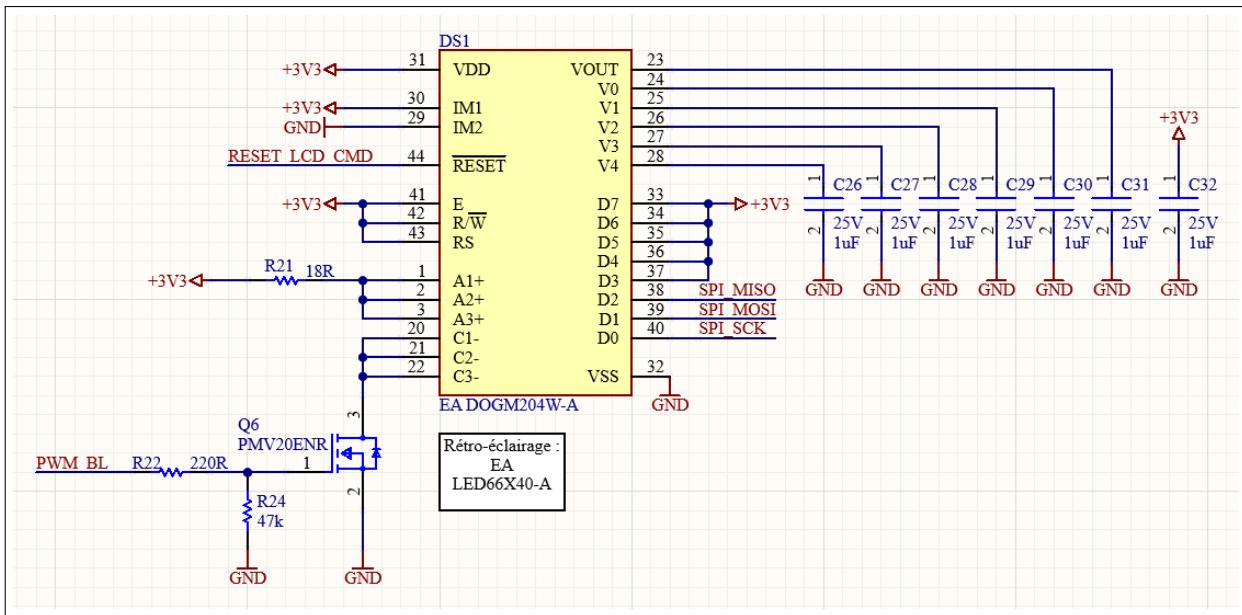
Malgré la capacité de l'écran à fonctionner sans rétro-éclairage sous la dalle LCD, j'ai choisi d'en incorporer un. La gestion de l'alumage et de l'intensité de ce rétro-éclairage pourra être effectuée via le menu de contrôle.

5.7.2 Dimensionnement

Aucun dimensionnement particulier n'a été nécessaire, la documentation de ces deux composants fournit la totalité des informations utiles au bon fonctionnement de ceux-ci. Cependant, en ce qui concerne la gestion de l'intensité du rétro-éclairage, j'ai mis en place un circuit de commutation sur les cathodes des LEDs, ce qui permet une gestion flexible à l'aide d'un signal PWM.

5.7.3 Schéma

FIGURE 15 – Schéma du système de commande Sony A7R



5.8 Système de rotation

5.8.1 Choix principaux

Choix du driver de moteur :

Comme stipulé dans la pré-étude, le driver du moteur doit autant pouvoir contrôler un moteur DC qu'un moteur pas à pas. De ce fait j'ai axé mes recherches sur des composants ayant deux ponts en H intégrés dans le même boîtier et pouvant gérer des courants allant jusqu'à 5A et une tension de plus de 24V. J'ai finalement trouvé un composant répondant parfaitement à ces attentes, il s'agit du DRV8432DKD. Ce composant possède un système de sécurité permettant d'avertir le MCU dans le cas où le chip est en surchauffe, en surintensité ou si la tension d'alimentation est trop faible.

Choix du moteur (**effectué le 05.09.2023**) :

Le moteur a été un choix en collaboration avec Monsieur Jérôme Antonioli qui a du réaliser différents calculs de couple et de vitesse de rotation. Il s'agit du moteur stepper MOT-AN-S-060-005-042-L-A-AAAA du fabricant igus. Son couple nominal est de 0.5 Nm, et sa plage de tension d'alimentation s'étend de 24V à 48V, ce qui fixe la tension de fonctionnement du montage à 24V de manière définitive.

FIGURE 16 – Caractéristiques du moteur MOT-AN-S-060-005-042-L-A-AAAA

technical data					
flange dimension	20(NEMA8)	28(NEMA11)	35(NEMA14)	42(NEMA17)	56(NEMA23)
motor					
max voltage	[VDC]	60	60	60	60
nominal voltage	[VDC]	24-48	24-48	24-48	24-48
intermittent operation	[A] at 25°C	0,6	1,0	1,2	1,8
continuous operation	[A] at 25°C	0,4	0,6	0,7	1,1
holding torque	[Nm]	0,026	0,12	0,2	0,5
detent torque	[Nm]	0,002	0,004	0,010	0,022
step angle	[°]	1,8 ±5%	1,8 ±5%	1,8 ±5%	1,8 ±5%
resistance / phase	[Ω]	5,8 ±10%	2,30 ±10%	2,5 ±10%	1,75 ±10%
inductance / phase	[mH]	2 ±20%	1,80 ±20%	3 ±20%	3,30 ±20%
dielectric strength	[VAC]	500	500	500	500
moment of inertia / rotor	[kgcm²]	0,0032	0,018	0,022	0,082
max. shaft load axial	[N]	4	7	7	15
max. shaft load radial	[N]	10	20	20	52

5.8.2 Dimensionnement

Calculs de dissipation thermique :

Selon les indications fournies dans la documentation technique du driver utilisé, il est fortement recommandé d'installer un dissipateur thermique sur le boîtier HTSSOP, même lorsque la puissance est faible. De ce fait j'ai ajouté un modèle de dissipateur plus ou moins grand ayant une résistance thermique de 23.68°C/W. Les calculs effectués pour obtenir la température de la jonction avec ce dissipateur suivent ce paragraphe. Les données telles que la résistance interne des MOSFETs, les résistances thermiques ou autres, proviennent des documents techniques de composants utilisés, c'est à dire du driver DRV8432DKD, du dissipateur HSB06-181810 et de l'adhésif thermique TG-A1250-15-15-1.0.

$$P = R * I^2 \quad (5)$$

où :

R = Résistance $R_{DS(on)}$ des MOSFETs (Ω)

I = Courant (A)

P = Puissance à dissiper (W)

$$R_{THth} = \frac{Z_{TH}}{S} \quad (6)$$

où :

- Z_{TH} = Impédance thermique ($^{\circ}\text{C in}^2/\text{W}$)
 S = Surface de contact (in^2)
 R_{THha} = Résistance thermique heatsink-ambiant

$$T_J = T_A + P * (R_{THjc} + R_{THth} + R_{THha}) \quad (7)$$

où :

- R_{THjc} = Résistance thermique jonction-case ($^{\circ}\text{C/W}$)
 R_{THth} = Résistance thermique case-heatsink ($^{\circ}\text{C/W}$)
 R_{THha} = Résistance thermique heatsink-ambiant ($^{\circ}\text{C/W}$)
 P = Puissance dissipée (W)
 T_A = Température ambiante ($^{\circ}\text{C}$)
 T_J = Température jonction ($^{\circ}\text{C}$)

A l'aide des formules précédentes, obtenues dans un bulletin technique de 3M nommé "Heat Flow Calculation for 3M Thermally Conductive Tapes 9882, 9885, 9890", j'ai calculé la température de la jonction du chip dans le cas où 4 ampères traversent le composant en continu. Avec la formule 6 j'ai obtenu la résistance thermique de l'adhésif se trouvant entre le dissipateur et le chip étant donné que celle-ci dépend de la surface en contact avec le composant.

TABLE 12 – Valeurs utilisées et résultat

R	0.11Ω
I	4A
P	<u>3.52W</u>

La puissance maximale à dissiper sera de 3.52W si le moteur tourne en continu.

TABLE 13 – Valeurs utilisées et résultat

S	0.2537 in^2
Z_{TH}	0.304 $^{\circ}\text{C in}^2/\text{W}$
R_{THjc}	<u>1.2°C/W</u>

Une fois cette valeur obtenue, j'ai pu calculer la température de la jonction en utilisant la formule 7.

TABLE 14 – Valeurs utilisées et résultat

R_{THjc}	0.4°C/W
R_{THth}	1.2°C/W
R_{THha}	23.68°C/W
P	3.52W
T_A	25°C
T_J	<u>113.99°C</u>

La jonction peut supporter une température maximale de 150°C, ce qui signifie que le dissipateur a une capacité adéquate pour dissiper la puissance. Même si la température peut sembler élevée, il est important de noter que le moteur n'est pas sensé fonctionner en continu. Par conséquent, en théorie, la température ne devra jamais atteindre cette valeur.

Résistance de surintensité :

Comme le chip possède un système de sécurité de surintensité, il est possible de choisir à partir de quel courant, le composant va couper. La programmation de cette valeur se fait au moyen d'une résistance dont les valeurs sont disponibles dans la datasheet du driver. Pour un courant maximum de 4.1A, la résistance à monter est $68\text{k}\Omega$. La valeur de cette résistance sera fixée définitivement lorsque le moteur aura été sélectionné.

5.8.3 Schémas

FIGURE 17 – Schéma du système de commande moteur partie A

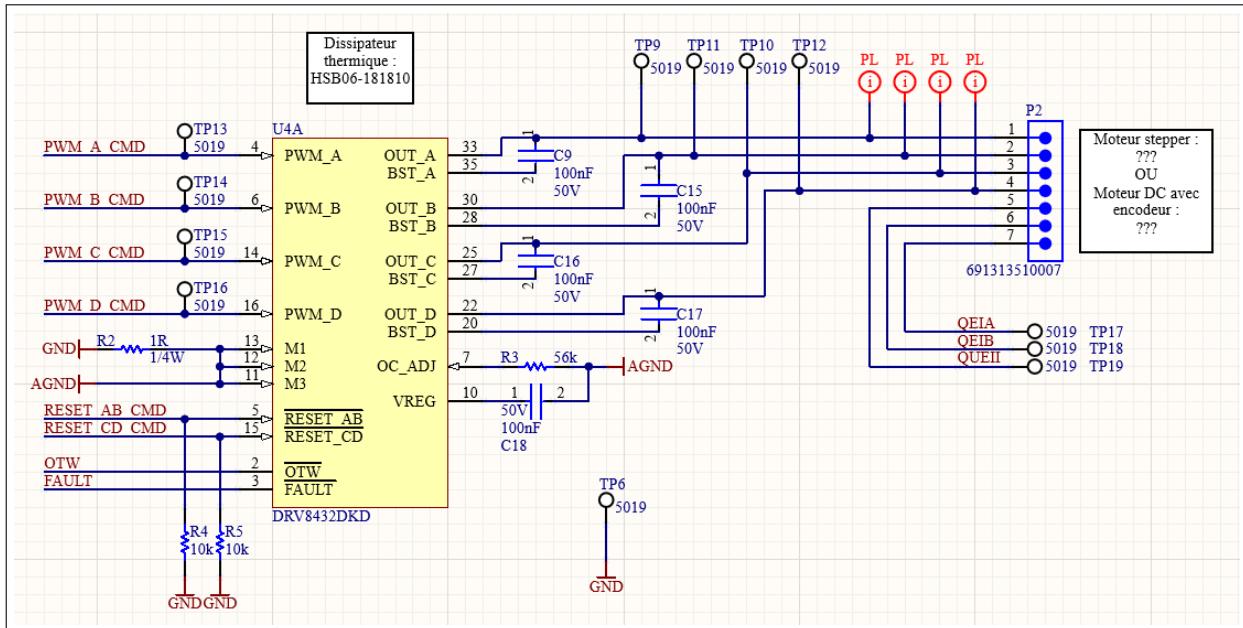
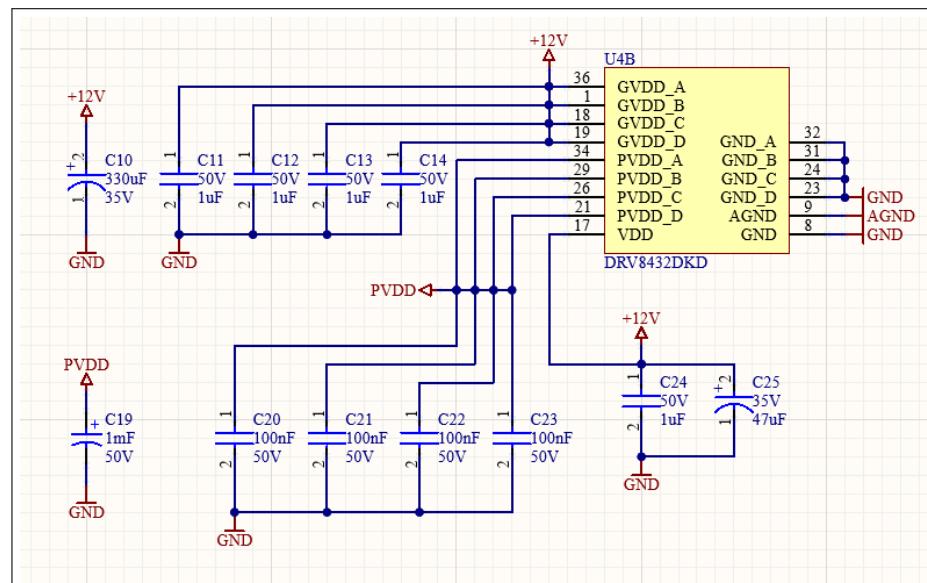


FIGURE 18 – Schéma du système de commande moteur partie B



5.9 Interface de commande

5.9.1 Choix principaux

Choix de l'encodeur rotatif :

Au niveau du contrôle de l'appareil, j'ai choisi un encodeur rotatif PEC12R-4120F-S0012. Celui-ci possède 12 crans par tour, un axe de 6mm de diamètre et un switch intégré.

Choix du switch :

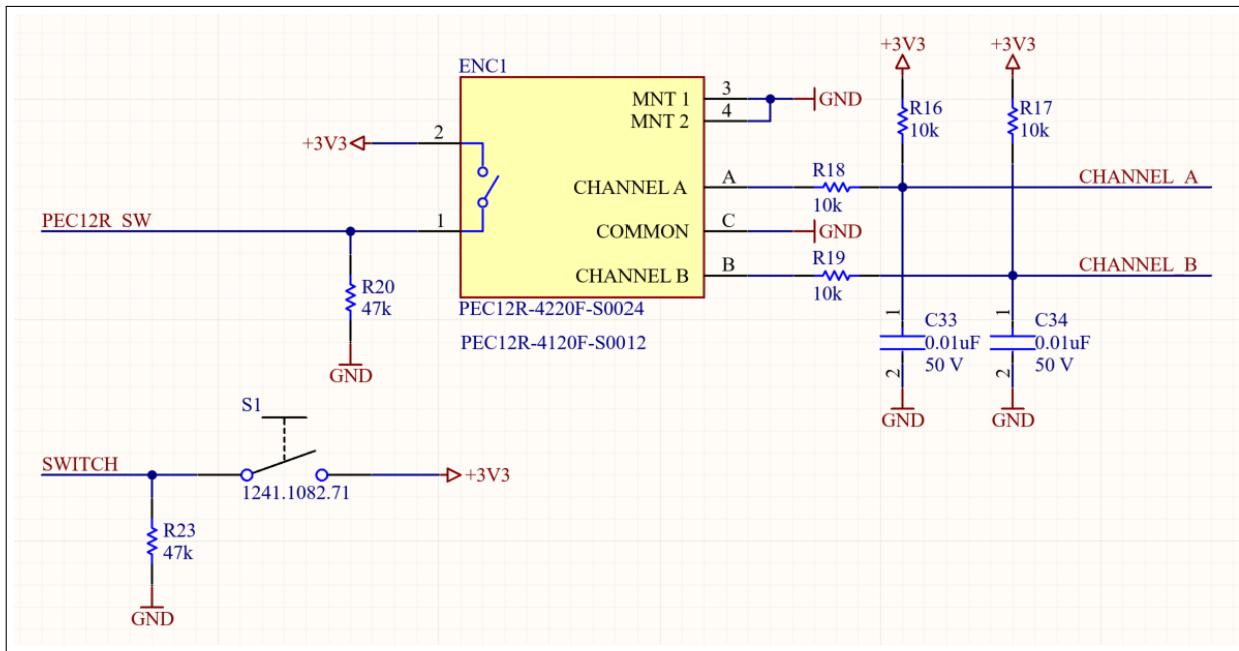
La référence du switch sélectionné est la suivante : 1241.1082.71. Ce composant est un simple bouton non maintenu ayant comme particularité d'avoir des capuchons modulables en fonction de la hauteur désirée.

5.9.2 Dimensionnement

Aucun dimensionnement n'a été requis, à l'exception de la mise en place de filtres passe-bas RC aux bornes du PEC12R. Le schéma de ces filtres est fourni dans la documentation technique du composant. Ils sont destinés à réduire les rebonds, qui peuvent devenir plus fréquents avec le vieillissement de l'encodeur.

5.9.3 Schéma

FIGURE 19 – Schéma de l'interface de commande



5.10 Système d'alimentation

Le montage final pourra être alimenté de deux manières différentes, +24V ou +12V, ce choix dépendra du moteur. Par contre, une fois ce choix réalisé, le montage ne pourra être alimenter que par cette tension car cela nécessite une intervention au niveau du PCB. Trois solder-bridges sont présents et doivent être ou ne pas être court-circuités pour sélectionner la tension d'alimentation.

5.10.1 Choix principaux

Choix du bloc secteur :

Bien que le moteur n'ait été choisi à ce jour, j'ai tout de même effectué deux premiers choix de blocs secteur pouvant être utilisés. Le bloc numéro 1 fourni une tension de 24V avec un courant de 2.5A maximum (puissance de 65W). Le second fourni une tension de 12V avec un courant de 5A. (puissance de 60W).

Choix du régulateur de tension 12V :

Le régulateur sélectionné est le P7812-2000R-S, qui est une alimentation à découpage intégrée dans un boîtier. Son courant de sortie peut atteindre jusqu'à 2A avec une tension de sortie de 12V. Ce courant de 2A sera suffisant pour alimenter à la fois une LED de puissance et le circuit fonctionnant en 3.3V.

Choix du régulateur de tension 3.3V :

Le régulateur choisi permet est le 173950378 de Wurth Elektronik, il s'agit également d'une alimentation à découpage intégrée dans un boîtier. Elle peut fournir un courant allant jusqu'à 500mA avec une tension de sortie de 3.3V.

5.10.2 Dimensionnement

Aucun dimensionnement particulier n'a été nécessaire, toutes les informations concernant les condensateurs de découplage sont disponibles dans les documentations techniques des deux composants, sous forme de tableaux.

5.10.3 Schémas

FIGURE 20 – Schéma de l'entrée de l'alimentation

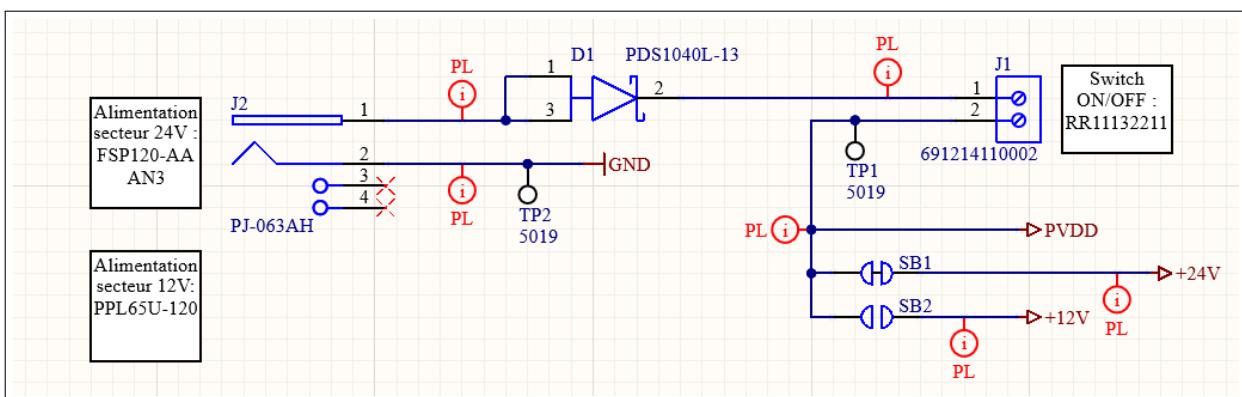


FIGURE 21 – Schéma du régulateur 12V

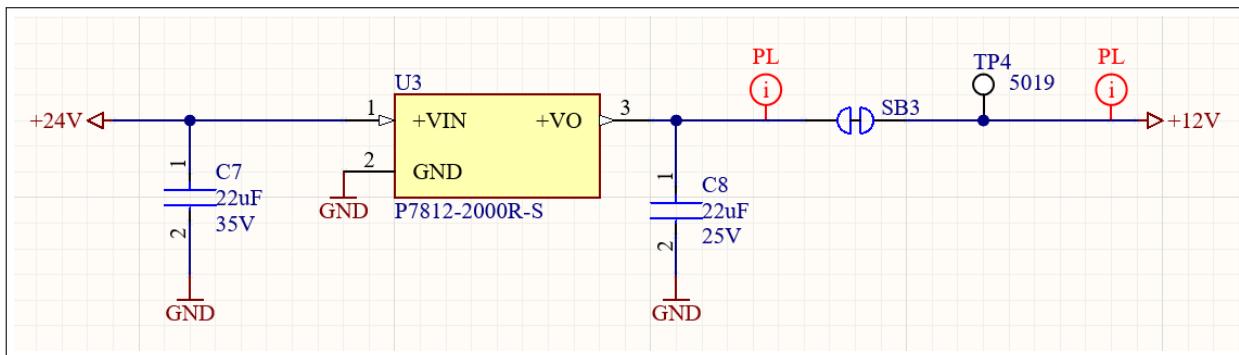
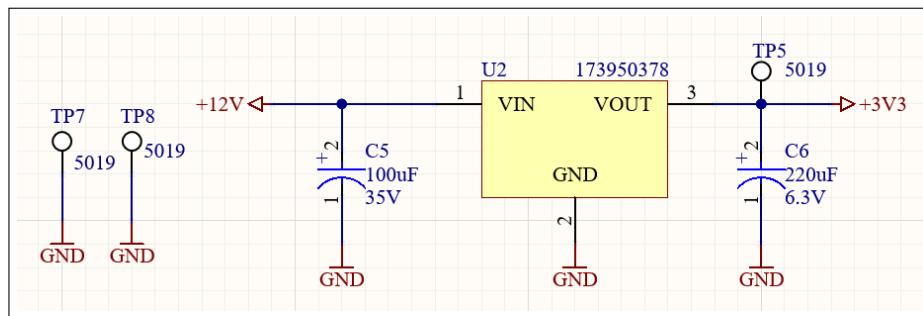


FIGURE 22 – Schéma du régulateur 3.3V



5.11 Microcontrôleur

5.11.1 Choix principal

Choix du microcontrôleur :

Conformément à ce qui a été mentionné dans la pré-étude, un microcontrôleur de la famille Motor Control est l'option idéale en raison de la nécessité de générer plusieurs signaux PWM. C'est pourquoi j'ai sélectionné le PIC32MK0512MCF064-I/PT, qui a été recommandé par Monsieur Serge Castoldi. Ce choix s'avère approprié car ce microcontrôleur peut gérer jusqu'à 12 sorties MCPWM et est compatible avec Harmony v2.06, contrairement à la plupart des autres MCU de cette famille.

5.11.2 Dimensionnement

Aucun calcul n'a été requis, car la documentation du microcontrôleur fournit le schéma minimal nécessaire pour le faire tourner.

5.11.3 Schémas

FIGURE 23 – Schéma du MCU

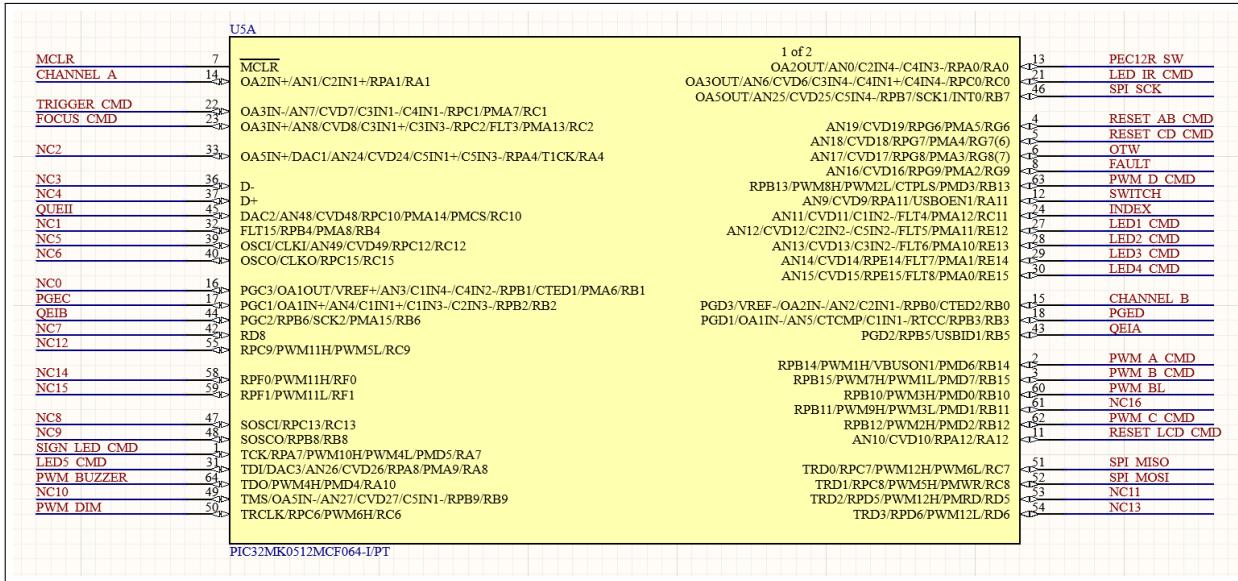


FIGURE 24 – Schéma des alimentations du MCU

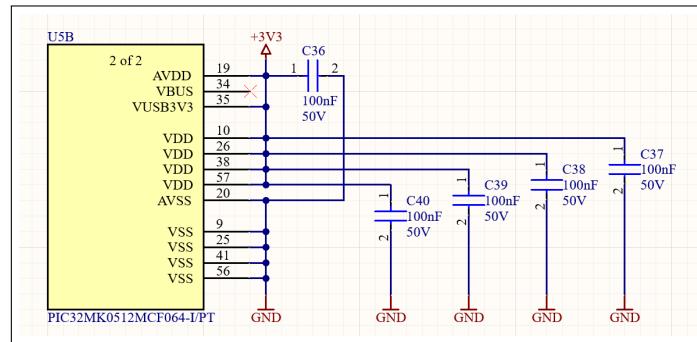
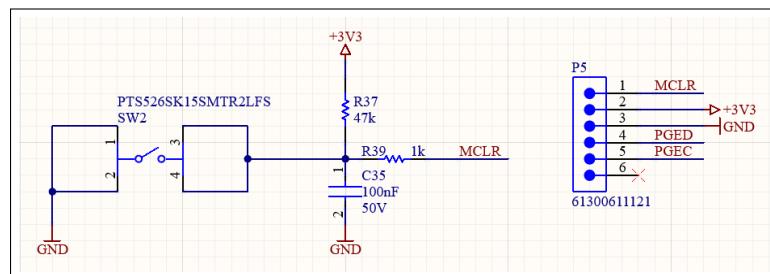


FIGURE 25 – Schéma du bouton reset et du connecteur de programmation



6 Design circuit imprimé

6.1 Introduction

Une fois le schéma terminé et revu par mon collègue Monsieur Ali Zoubir, j'ai débuté la phase de design du circuit imprimé sur Altium. J'ai dû respecter différentes contraintes telles que la taille des pistes en fonction des différents courants, la séparation des plans de masse alimentant le moteur, très bruyant électriquement, et le reste du circuit et la positions des connecteurs afin qu'ils s'intègrent parfaitement dans le boîtier. Les détails sont décrits dans les sections suivantes.

6.2 Caractéristiques du circuit

N'ayant aucune contrainte au niveau des dimensions de la carte, je les ai fixées à 150mm x 90mm (réellement 154mm expliqué plus loin dans ce document). De ce fait j'ai l'espace nécessaire sans exagération afin de placer au mieux tous les composant tout en limitant le prix de production. Ces dimensions m'ont permise d'utiliser uniquement deux couches (top et bottom) de $35\mu\text{m}$ d'épaisseur de cuivre chacune. Le circuit est produit en matériaux FR4 qui est le plus commun pour les circuits de base (circuits n'ayant pas de signaux hautes fréquences ou autres spécialités). La grande majorité des composants sont placés sur la face top du PCB, tandis que les composants d'interface sont positionnés sur la face inférieure. Cette configuration permet d'intégrer plus facilement le circuit dans le boîtier et de façon ergonomique. Notons que le circuit sera intégré dans le boîtier avec la face bottom orientée vers le haut.

La largeur des pistes a été en premier lieu calculé à l'aide de l'utilitaire de Digi-key "PCB Trace Width Calculator", puis j'y ai ajouté une certaine marge supplémentaire lorsque la place le permettait. L'épaisseur des lignes est configurée de la manière suivante : les lignes 3.3V ont une largeur fixe de 0.28mm, tandis que les lignes 12V ont une largeur minimale de 0.48mm et maximale de 2 mm. Pour ce qui est de la ligne 24V, sa largeur varie entre 2mm et 5mm.

6.3 Placement et routage des composants

Le placement des composants est un point très important car une mauvaise disposition peut entraîner des problèmes de routage, affectant ainsi l'intégrité des signaux et la fiabilité du système. De ce fait, j'ai tout d'abord placé le connecteur d'alimentation et les régulateurs en haut à gauche de la carte. Ensuite j'ai placé le driver de moteur dans le coin inférieur droit de la carte de sorte à ce que je puisse faire passer son alimentation de puissance sur le coté de la carte et non pas en plein milieu du circuit. De plus, j'ai séparé le plan de masse permettant le retour de courant du driver de moteur afin de minimiser les perturbations. La raison pour laquelle j'ai opté pour cette configuration réside dans le fait que le moteur est susceptible de générer un fort bruit électrique, pouvant potentiellement causer des interférences aux autres composants présents sur la carte. Sur la Figure 26, nous pouvons constater la séparation des deux plans au niveau du connecteur

d'alimentation. Ces plans incorporent un stitching visant à réduire la distance parcourue par le courant de retour. La sortie du régulateur 12V traverse la partie supérieure de la carte pour alimenter à la fois le driver de LED et l'alimentation basse puissance du driver de moteur. Le centre de la carte fonctionne à une tension de 3.3V, c'est à dire le MCU, l'écran LCD etc. Les connecteurs sont positionnés de manière à dépasser de

4 mm du PCB, alignés avec la surface extérieure du boîtier. Quant au connecteur d'alimentation, d'index et de déclenchement de l'appareil photo, ils sont situés sur une extension du PCB, car il n'est pas possible de les faire dépasser du circuit autrement. Le PCB comporte également cinq trous de perçage de 3,5 mm de diamètre permettant le passage des vis pour fixer le circuit dans le boîtier.

Le schéma d'implantation et les couches top et bottom sont en annexes.

FIGURE 26 – Couche bottom

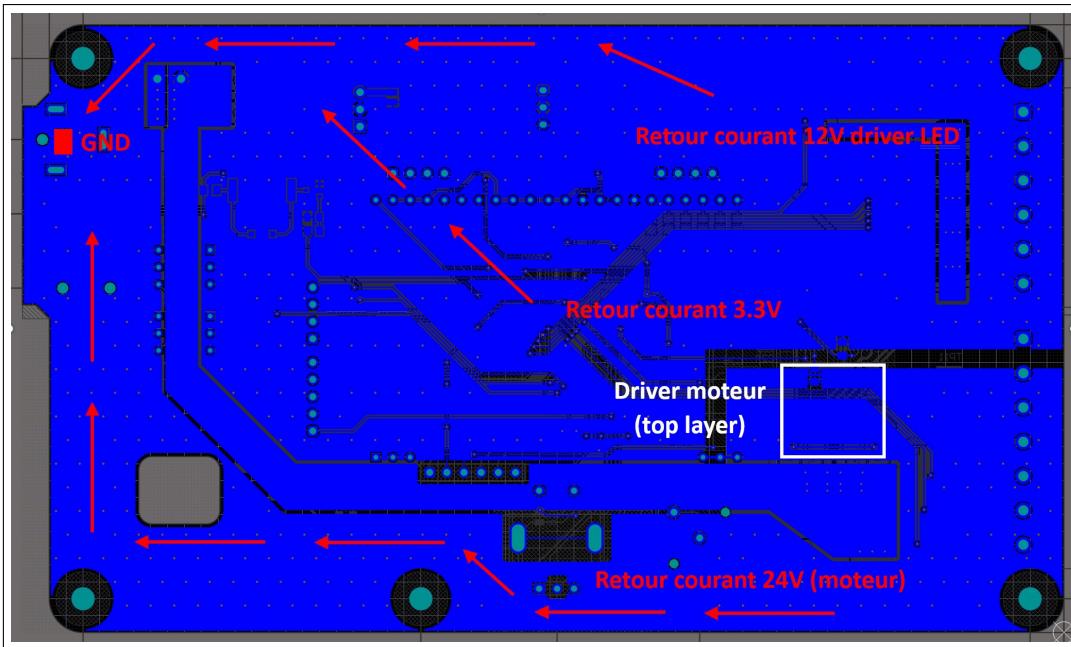
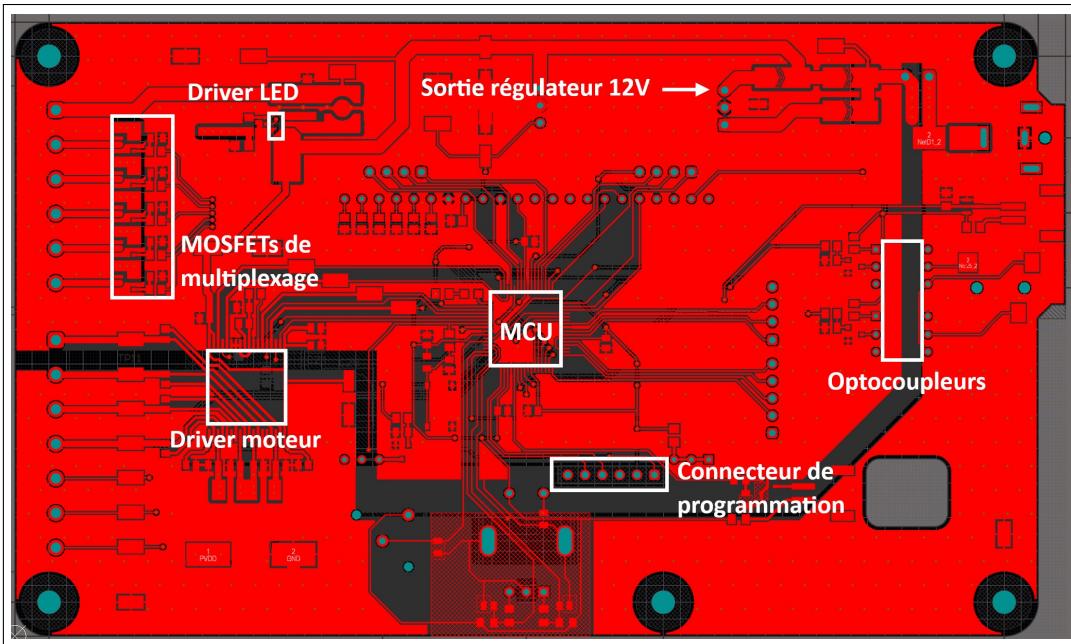


FIGURE 27 – Couche top



6.4 Améliorations

Une fois le PCB terminé, j'ai remarqué que certains aspects n'étaient pas optimaux. Le plus remarquable concerne la ligne d'alimentation du driver de moteur, qui est entrecoupée par des pins de composants traversants. Malheureusement à défaut de temps, je n'ai pas pu le corriger. Notons tout de même qu'il s'agit ici d'un amélioration car la largeur de la piste interrompue est suffisante pour gérer le courant qui la traverse.

7 Design mécanique

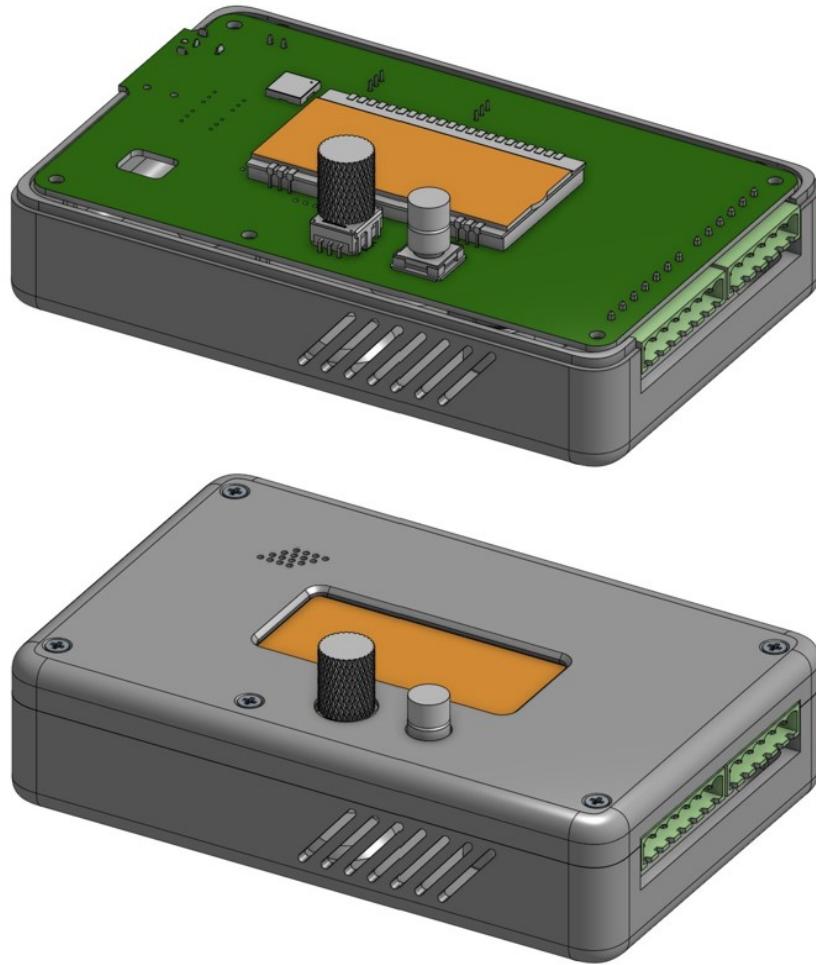
7.1 Introduction

Après avoir achevé la conception du PCB, j'ai réalisé la modélisation du boîtier en utilisant l'application web Onshape. J'ai choisi d'utiliser ce logiciel plutôt que SolidWorks en raison de ma meilleure maîtrise de ses fonctionnalités. Afin d'optimiser le processus de design, j'ai exporté une version 3D du PCB depuis Altium et je l'ai intégré à l'assemblage du projet Onshape.

7.2 Design

J'ai élaboré la conception du boîtier de manière élémentaire, sous la forme d'un rectangle mesurant 158mm sur 102mm. À l'intérieur, j'ai intégré cinq colonnettes de fixation, qui remplissent également la fonction de fermer le boîtier depuis le dessus. Pour garantir un bon refroidissement du driver de moteur, j'ai pris soin d'ajouter deux ouvertures de ventilation de chaque côté du boîtier, permettant d'y faire passer un flux d'air. J'ai également prévu des ouvertures spécifiques pour la connectique, assurant ainsi la facilité de l'interface. Du côté supérieur du boîtier, des ouvertures ont été créées pour accueillir l'encodeur rotatif et le bouton poussoir, garantissant leur accessibilité. Enfin, une grande fenêtre a été conçue pour visualiser l'écran LCD, tandis que des trous ont été prévus pour permettre au son émis par le buzzer de s'échapper de manière efficace.

FIGURE 28 – Boîtier ouvert avec PCB intégré et boîtier fermé



8 Firmware MCU

8.1 Introduction

Dans cette section, nous verrons les paramétrages principaux du MCU et de ses périphériques. Ensuite, nous verrons le fonctionnement principal du programme sous forme de machine d'états, ainsi que des parties de code essentielles à la compréhension du firmware.

8.2 Configuration du system clock

Selon la configuration hardware, la source d'horloge du microcontrôleur est interne car aucun quartz ou oscillateur n'a été prévu à cet effet, étant donné qu'aucune exigence de haute précision n'est nécessaire. De ce fait j'ai la liberté de choisir une fréquence d'horloge lors de la programmation et de l'adapter en fonction des besoins. Le PIC32MK que j'utilise peut monter jusqu'à une fréquence de 120MHz. Étant donné qu'il n'est pas nécessaire d'effectuer des calculs complexes, j'ai réglé cette fréquence à 40 MHz pour obtenir une performance adéquate sans exagération.

8.3 Configuration des Timers

Les paramètres présents dans la configuration des divers Timers utilisés ont été calculés en utilisant la formule 8. Cette équation permet de déterminer la valeur de la période des Timers en fonction de la fréquence souhaitée.

$$PERIOD = \frac{f_{SYS}}{f_{Timer} * PRESCALER} - 1 \quad (8)$$

où :

f_{Timer}	= Fréquence du Timer désirée (Hz)
f_{SYS}	= Fréquence du système (Hz)
$PRESCALER$	= Valeur de prescaler (-)
$PERIOD$	= Période du Timer qui devra être entrée dans Harmony (-)

Timer0 ID1 :

Le Timer0 ID1 a pour but de séquencer les 5 LEDs de puissance. C'est à dire activer un flag de la LED à allumer en fonction de l'instant et de démarrer la séquence de prise d'une image qui elle, est gérée par le Timer4 ID5. Le Timer0 ID1 est activé uniquement lorsqu'un séquence automatique est lancée ou si une simple séquence de 5 prises d'image (mode manuel) est lancée, une fois la séquence terminée, le Timer est stoppé, le reste du temps il est arrêté.

TABLE 15 – Valeurs utilisées et résultat

f_{Timer}	1kHz
f_{SYS}	40MHz
$PRESCALER$	1
$PERIOD$	<u>39999</u>

Timer1 ID2 :

Le Timer1 ID2 est responsable de la mise à jour de la machine d'état. Celle-ci est mise à jours à une fréquence de 10kHz afin de gérer au mieux la communication SPI qui est également sous forme de machine d'état. En d'autre termes, la séquence de service tasks est exécutée à une fréquence de 10kHz.

TABLE 16 – Valeurs utilisées et résultat

f_{Timer}	10kHz
f_{SYS}	40MHz
<i>PRESCALER</i>	1
<i>PERIOD</i>	<u>3999</u>

Timer2 ID3 :

Le Timer2 ID3 est en charge de la séquence du moteur pas à pas. Il ne fonctionne pas à une fréquence fixe, car sa tâche principale est de contrôler la vitesse du moteur. En augmentant la fréquence, le moteur tourne plus rapidement. Lorsque le moteur est actif, chaque interruption provoque l'exécution d'un pas, faisant ainsi tourner le moteur d'un pas à la fois. Même si la vitesse peut varier, la configuration initiale doit être réalisée en utilisant des valeurs qui se situent dans la plage de vitesse souhaitée. La priorité des interruptions de ce Timer est la plus élevée (LEVEL4) car il ne faut pas louper des steps inutilement étant donnée qu'il est drivé en boucle ouverte.

TABLE 17 – Valeurs utilisées et résultat

f_{Timer}	10kHz
f_{SYS}	40MHz
<i>PRESCALER</i>	16
<i>PERIOD</i>	<u>249</u>

Timer3 ID4 :

Le Timer3 ID4 a pour but de stopper le programme durant un certain temps (fonction de délai bloquant) permettant d'initialiser le LCD au démarrage. Une fois lancé, ce Timer génère une interruption toute les millisecondes afin de pouvoir régler le temps d'arrêt du programme avec une précision à la milliseconde. La priorité des interruptions de ce Timer est la plus basse (LEVEL1), car il ne gère pas de composant critique et est utilisé uniquement au démarrage du MCU lorsque tous les autres Timers sont éteints.

TABLE 18 – Valeurs utilisées et résultat

f_{Timer}	1kHz
f_{SYS}	40MHz
<i>PRESCALER</i>	1
<i>PERIOD</i>	<u>39999</u>

Timer4 ID5 :

Le Timer4 ID5 gère la séquence de prise d'une seule image. C'est ce Timer qui permet de gérer le déclenchement de l'appareil photo et gérer l'allumage et l'extinction de la LED flagged par la séquence du Timer0 ID1. Les interruptions de ce Timer sont au LEVEL3, c'est à dire, la priorité juste en dessous de celle du contrôle du moteur car il s'agit ici de gérer un temps à priori précis.

TABLE 19 – Valeurs utilisées et résultat

f_{Timer}	1kHz
f_{SYS}	40MHz
<i>PRESCALER</i>	1
<i>PERIOD</i>	<u>39999</u>

8.4 Configuration des MCPWMs

Le microcontrôleur implémenté appartient à la famille des Motor Control, de ce fait il est équipé de modules MCPWM. Ces modules offrent une gamme plus étendue de paramètres pour générer des signaux PWM par rapport aux fonctions de comparaison de sortie (Output Compare) disponibles sur les microcontrôleurs à usage général. Grâce à ces module, il m'a été possible de générer les signaux nécessaires au bon fonctionnement du stepper moteur.

Configuration générale :

Les valeurs de période ont été calculé à l'aide de la formule 9 qui provient du document de référence Section 44. Motor Control PWM (MCPWM) de Microchip.

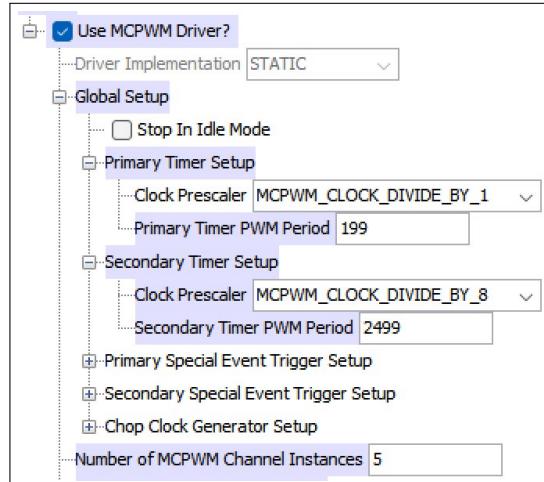
$$PERIOD = \frac{f_{SYS}}{f_{PWM} * PWMInputClockPrescaler} - 1 \quad (9)$$

où :

f_{PWM}	= Fréquence du PWM désirée (Hz)
f_{SYS}	= Fréquence du système (Hz)
$PWMInputClockPrescaler$	= Valeur de prescaler (-)
$PERIOD$	= Période du PWM qui devra être entrée dans Harmony (-)

La fréquence du Timer PWM primaire est fixée à 200kHz. Cette fréquence est celle utilisé pour le contrôle de la puissance du moteur. La fréquence du Timer PWM secondaire est fixée à 2kHz. La sélection de cette fréquence a été basée sur les spécifications fournies dans la datasheet du buzzer, dans le but de générer une intensité sonore assez élevée pour qu'elle puisse être entendue clairement.

FIGURE 29 – Configuration générale MCPWM



MCPWM0 CHANNEL1 :

Ce MCPWM gère les signaux PWM_A_CMD (CHANNEL1 HIGH) et PWM_B_CMD (CHANNEL1 LOW) du moteur, c'est à dire les signaux A et A_. Le mode de génération est en COMPLEMENTARY MODE. La polarité du CHANNEL1 HIGH est en mode ACTIVELOW alors que celle du CHANNEL1 LOW est en mode ACTIVEHIGH.

MCPWM1 CHANNEL2 :

Ce MCPWM gère les signaux PWM_C_CMD (CHANNEL2 HIGH) et PWM_D_CMD (CHANNEL2 LOW) du moteur, c'est à dire les signaux B et B_. Le mode de génération est en COMPLEMENTARY MODE. La polarité du CHANNEL2 HIGH est en mode ACTIVEHIGH alors que celle du CHANNEL2 LOW est en mode ACTIVELOW.

MCPWM2 CHANNEL3 :

Ce module MCPWM gère l'intensité lumineuse du rétro-éclairage du LCD. Le PWM a une fréquence de 2kHz étant donnée que sa source provient du Timer PWM secondaire. Le mode de génération est en mode LOW LATCHED TO ZERO. La sortie du signal se fait sur le CHANNEL3_H, le CHANNEL3_L est désactivé.

MCPWM3 CHANNEL4 :

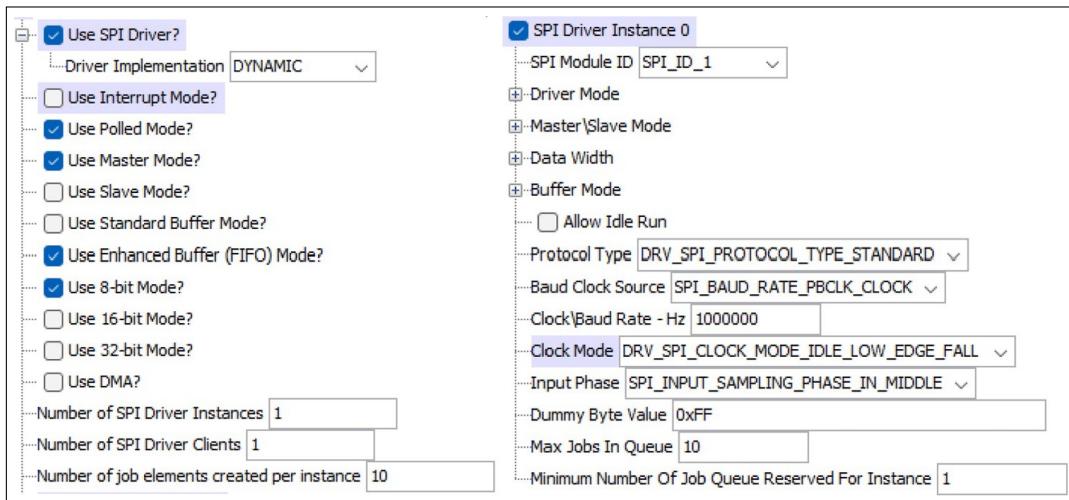
Ce module MCPWM gère l'intensité sonore du buzzer. Le PWM a une fréquence de 2kHz étant donnée que sa source provient du Timer PWM secondaire. Le mode de génération est en mode LOW LATCHED TO ZERO. La sortie du signal se fait sur le CHANNEL4_H, le CHANNEL4_L est désactivé.

MCPWM4 CHANNEL6 :

Ce module MCPWM gère l'intensité lumineuse des LEDs de puissance. Le PWM a une fréquence de 2kHz étant donnée que sa source provient du Timer PWM secondaire. Le mode de génération est en mode LOW LATCHED TO ZERO. La sortie du signal se fait sur le CHANNEL6_H, le CHANNEL6_L est désactivé.

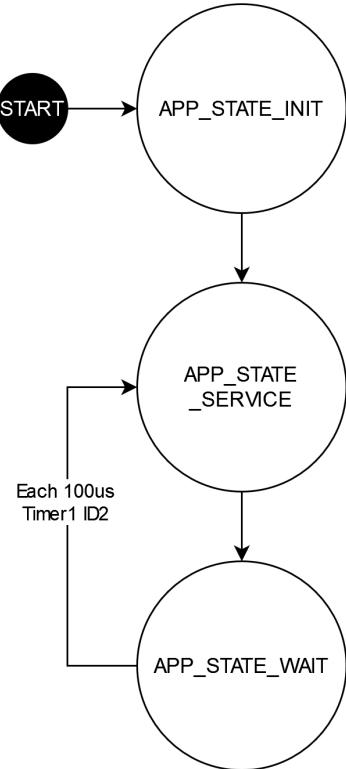
8.5 Configuration du SPI

FIGURE 30 – Configuration SPI



8.6 Machine d'état principale

FIGURE 31 – Machine d'état principale



Dans la Figure 31, une représentation de la machine d'état principale du système est visible. Cette machine est synchronisée par le Timer1 ID2 à une fréquence de 10 kHz. Au démarrage, tous les périphériques du système sont initialisés avant d'entrer pour la première fois dans l'état de service. Après chaque passage dans cet état de service, le programme entre dans l'état d'attente. Dans cet état, le microcontrôleur n'est pas totalement inactif, car il peut potentiellement exécuter des actions telles que le contrôle du moteur ou la gestion d'une séquence en réponse aux déclenchements d'autres Timers.

8.6.1 Détails

Dans la machine d'état principale, la séquence d'initialisation commence par la lecture des données stockées dans l'EEPROM, puis les utilise pour configurer les différents périphériques. En ce qui concerne la séquence de service, elle débute par l'entrée dans la gestion de la séquence de capture d'image en tant que première étape. Cette fonction ne produit aucun effet si aucune séquence n'est en cours. Ensuite, la lecture des boutons est effectuée en fonction de l'état d'un compteur diviseur de temps, suivie de la gestion de l'affichage sur l'écran LCD et de la navigation dans les menus, en fonction d'un autre compteur. Enfin, la séquence se termine par un appel à une autre machine d'état, celle chargée de la gestion du SPI. Les états APP_STATE_INIT et APP_STATE_SERVICE sont décrits en détail dans les organigrammes illustrés sur les Figures 32 et 33.

FIGURE 32 – État d'initialisation

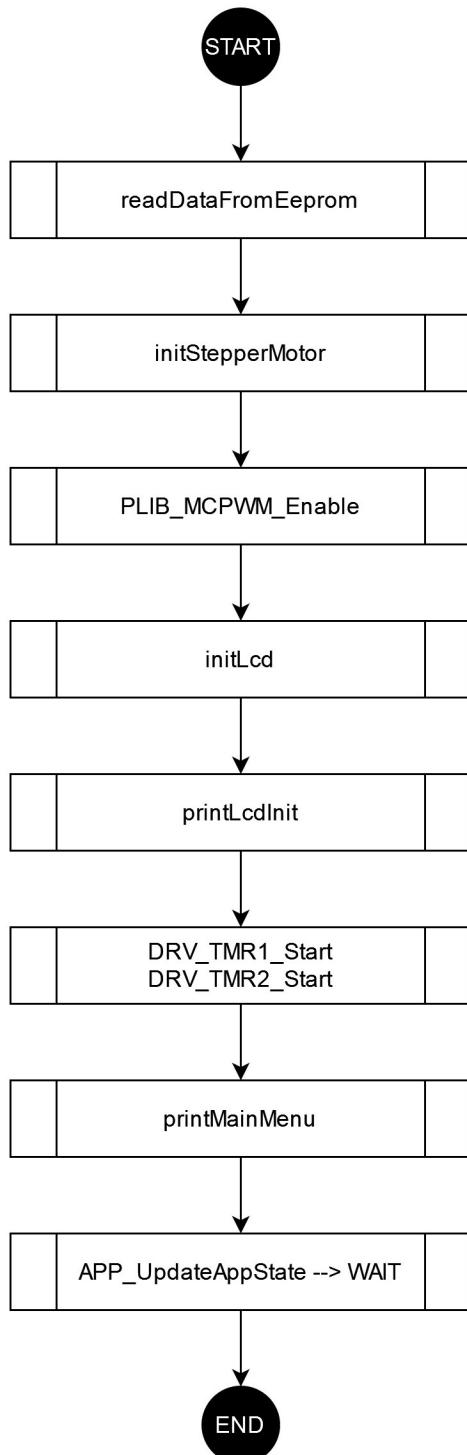
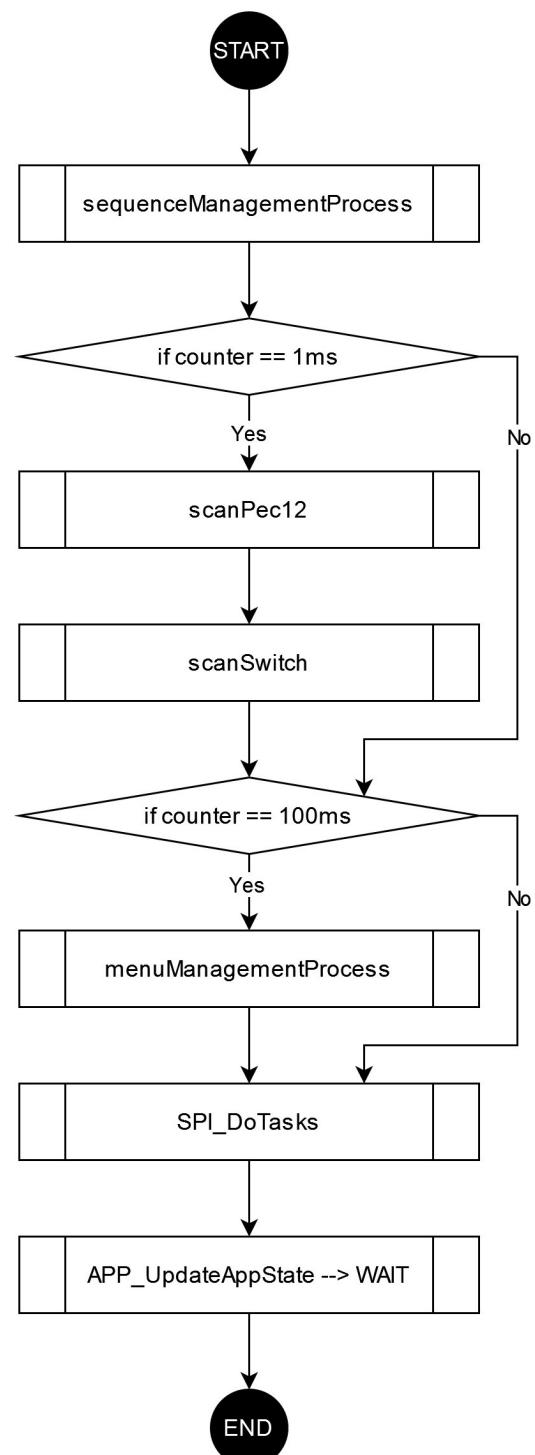


FIGURE 33 – État de service



8.7 Gestion du moteur stepper

La gestion du moteur s'est révélée être une étape complexe, car je n'avais aucune expérience préalable avec les moteurs pas à pas. De plus, la documentation limitée du driver DRV8432 n'a pas facilité l'apprentissage du contrôle d'un moteur de ce type pour la première fois. De plus, la gestion à l'aide des modules MCPWM du PIC32MK représentait également une nouveauté pour moi et n'était pas immédiatement évidente à comprendre. Par conséquent, j'ai dû investir beaucoup de temps dans la recherche et la réalisation de nombreux tests pour parvenir à un résultat satisfaisant.

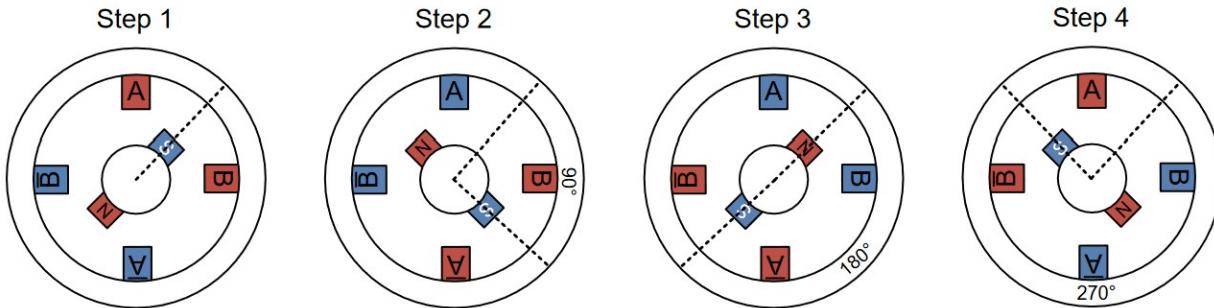
Un rapport d'application de Texas Instruments intitulé "Stepper Motor Driving With Peripheral Drivers (Driver ICs)" m'a été d'une grande utilité pour établir une séquence de contrôle fonctionnelle du moteur, dont la partie principale est présentée dans le Listing 1. Cette séquence permet de contrôler le moteur en utilisant la technique du full step. L'intérêt de ce mode de contrôle réside dans le fait qu'il s'agit du mode le plus couramment utilisé et qu'il offre également le couple le plus élevé.

FIGURE 34 – Séquence logique

	Step Sequence			
	1	2	3	4
A	1	0	0	1
B	1	1	0	0
\bar{A}	0	1	1	0
\bar{B}	0	0	1	1

Pour parvenir à un résultat fonctionnel, j'ai principalement utilisé les informations contenues dans les Figures 34 et 35, lesquelles sont extraites du rapport d'application de TI. Le fonctionnement n'a pas été immédiat car en plus de la gestion de la séquence, l'ajustement précis des MCPWMs est essentiel. Si le cycle de service (duty-cycle) n'est pas suffisamment élevé, le moteur ne se met pas en mouvement au-delà d'une certaine vitesse, tandis que si le cycle de service est trop élevé, le moteur ne fonctionne plus correctement à basse vitesse.

FIGURE 35 – Séquence des coils



Listing 1 – Séquence de contrôle moteur CW

```

switch(step){
    /* Sequence of 4 steps for CW rotation */
    case 1:
        /* A */
        PLIB_MCPWM_Channel1PWMxHEnable (MCPWM_ID_0 ,MCPWM_CHANNEL1);
        /* B */
        PLIB_MCPWM_Channel1PWMxHDisable(MCPWM_ID_0 ,MCPWM_CHANNEL2);
        /* A_ */
        PLIB_MCPWM_Channel1PWMxLDisable(MCPWM_ID_0 ,MCPWM_CHANNEL1);
        /* B_ */
        PLIB_MCPWM_Channel1PWMxLEnable (MCPWM_ID_0 ,MCPWM_CHANNEL2);
        break;

    case 0:
        PLIB_MCPWM_Channel1PWMxHEnable (MCPWM_ID_0 ,MCPWM_CHANNEL1);
        PLIB_MCPWM_Channel1PWMxHEnable (MCPWM_ID_0 ,MCPWM_CHANNEL2);
        PLIB_MCPWM_Channel1PWMxLDisable(MCPWM_ID_0 ,MCPWM_CHANNEL1);
        PLIB_MCPWM_Channel1PWMxLDisable(MCPWM_ID_0 ,MCPWM_CHANNEL2);
        break;

    case 3:
        PLIB_MCPWM_Channel1PWMxHDisable(MCPWM_ID_0 ,MCPWM_CHANNEL1);
        PLIB_MCPWM_Channel1PWMxHEnable (MCPWM_ID_0 ,MCPWM_CHANNEL2);
        PLIB_MCPWM_Channel1PWMxLEnable (MCPWM_ID_0 ,MCPWM_CHANNEL1);
        PLIB_MCPWM_Channel1PWMxLDisable(MCPWM_ID_0 ,MCPWM_CHANNEL2);
        break;

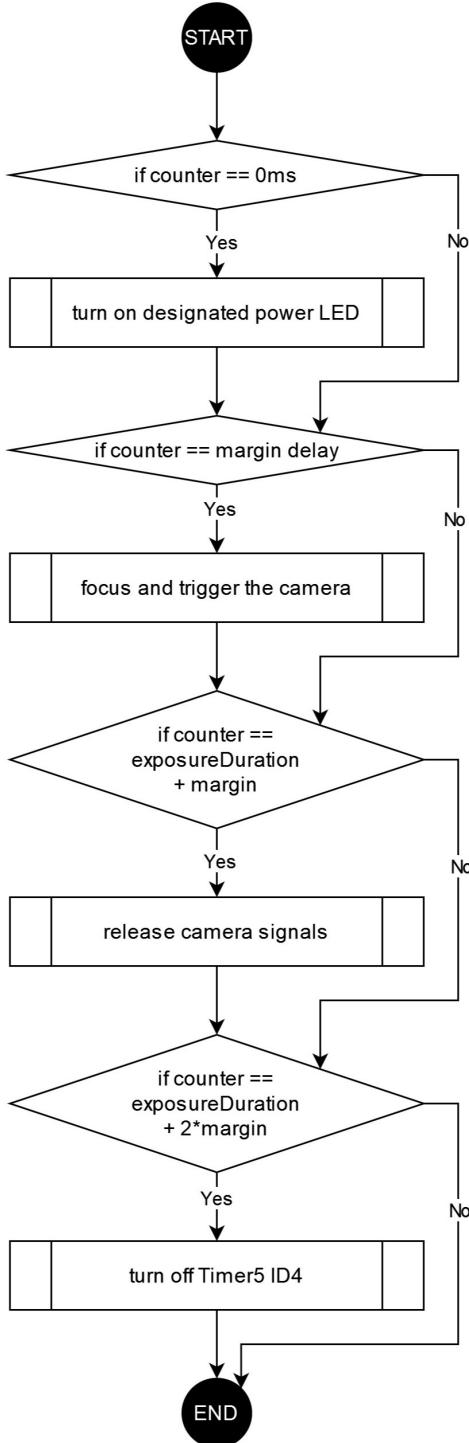
    case 2:
        PLIB_MCPWM_Channel1PWMxHDisable(MCPWM_ID_0 ,MCPWM_CHANNEL1);
        PLIB_MCPWM_Channel1PWMxHDisable(MCPWM_ID_0 ,MCPWM_CHANNEL2);
        PLIB_MCPWM_Channel1PWMxLEnable (MCPWM_ID_0 ,MCPWM_CHANNEL1);
        PLIB_MCPWM_Channel1PWMxLEnable (MCPWM_ID_0 ,MCPWM_CHANNEL2);
        break;
}
step++;

```

Dans le listing ci-dessus, nous pouvons clairement apercevoir la séquence de contrôle des enroulements.

8.8 Gestion de la séquence de prise d'image

FIGURE 36 – Séquence one shot



La gestion de la capture d'images est divisée en deux parties distinctes. La première partie est dédiée à la gestion d'une seule capture d'image, en veillant à respecter un timing précis. À gauche, la Figure 36 représente la séquence de capture d'une seule image sous forme de flowchart. La seconde partie concerne la gestion d'une séquence de 5 captures d'image, ce qui entraîne l'exécution de la première partie cinq fois de suite. Cette seconde partie est responsable de la sélection de la LED à activer.

8.9 Gestion du menu

L'architecture du menu est composée de trois principales fonctions, avec une quatrième fonction chargée de les appeler. La première fonction, appelée "menuActionProcess", gère la navigation dans les menus. La seconde, nommée "menuDataProcess", s'occupe des interactions entre le menu et les réglages du système, y compris les commandes en temps réel telles que la rotation du moteur en mode manuel. Enfin, la troisième fonction appelée "menuPrintProcess" a pour unique responsabilité d'afficher les données sur l'écran LCD.

Listing 2 – Début de la fonction menuActionProcess

```
void menuActionProcess(int32_t pec12RotationValue){  
    /* Menu action switch */  
    if(isInModifMode == false){  
        switch(menu.menuState){  
  
            //-----// Main menu  
            case MAIN_MENU:  
  
                switch(pec12RotationValue){  
  
                    case CAPTURE_MODE_SEL:  
                        menu.menuState = CAPTURE_MODE_MENU;  
                        menu.menuSize = 2;  
                        break;  
  
                    case SETTINGS_SEL:  
                        menu.menuState = SETTINGS_MENU;  
                        menu.menuSize = 5;  
                        break;  
  
                    case ABOUT_SEL:  
                        menu.menuState = ABOUT_MENU;  
                        menu.menuSize = 0;  
                        break;  
                }  
                break;  
    }  
}
```

Le listing ci-dessus illustre une partie de la fonction menuActionProcess qui permet à l'utilisateur de naviguer entre différents menus en utilisant un encodeur rotatif. La gestion du menu est basée sur un ensemble d'énumérations, ce qui rend la lecture du code plus compréhensible et structurée comme nous pouvons le voir.

9 Mise en service

9.1 Introduction

Cette mise en service a pour but de valider les choix de conception et d'assurer que le système électronique répond aux spécifications du cahier des charges au niveau hardware. Cette section se concentrera en détail sur les mesures réalisées et les résultats obtenus pour les composants essentiels, les alimentations, le driver du moteur pas à pas et le driver de LED de puissance. Cette étape est cruciale pour s'assurer que notre système réponde pleinement aux exigences du projet.

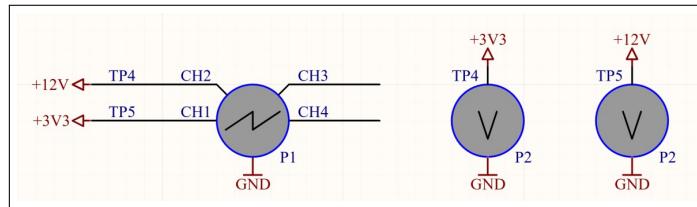
9.2 Régulateurs de tension

9.2.1 Méthode de mesure

Pour ces mesures, j'ai utilisé une alimentation de laboratoire fournissant une tension continu de 24V pour alimenter le montage. Ensuite, j'ai connecté une sonde d'oscilloscope à chaque sortie pour évaluer la stabilité des tensions, tandis qu'un voltmètre a été employé pour obtenir des mesures précises des tensions.

9.2.2 Schéma de mesure

FIGURE 37 – Schéma de mesure des alimentations



9.2.3 Résultats

Oscillogramme

FIGURE 38 – Sortie régulateur de tension 12V et 3.3V



Tableau des mesures

TABLE 20 – Mesures au voltmètre

	Désiré	Mesuré
Régulateur 12V	12V	12.04V
Régulateur 3.3V	3.3V	3.319V

9.2.4 Analyse

Selon les mesures effectuées, nous pouvons voir que les tensions de sortie des deux alimentations sont stables, non bruitées et correspondent bien aux tensions attendues.

9.3 Driver de moteur

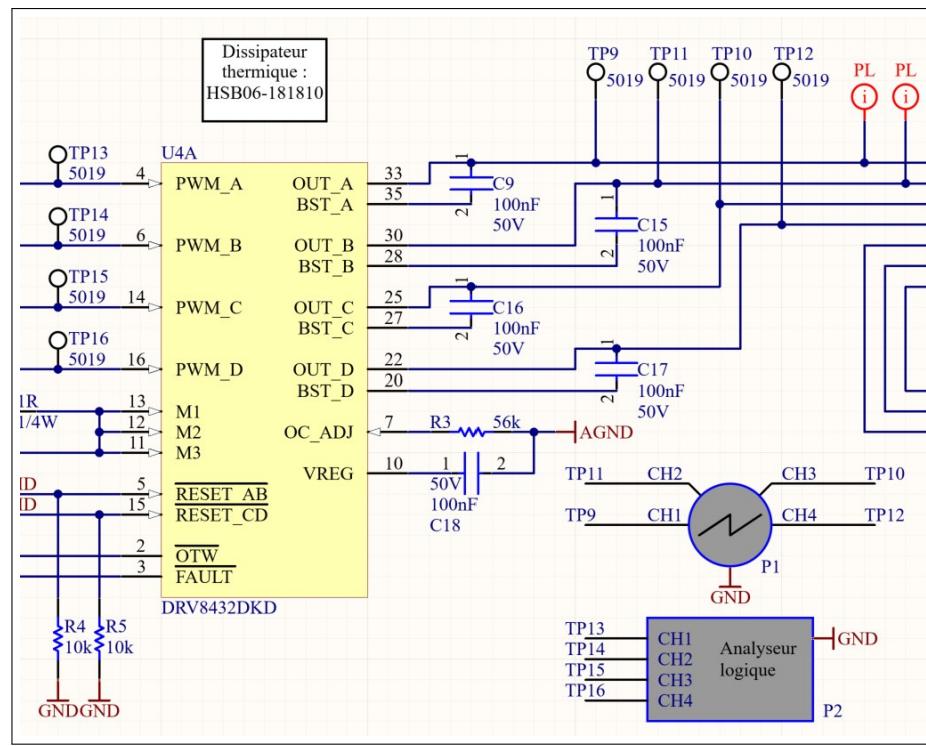
Comme cela a déjà été mentionné précédemment dans ce rapport, la mise en service du moteur a pris beaucoup plus de temps que prévu. Par conséquent, j'ai effectué de nombreuses mesures, et certaines d'entre elles, qui me semblent intéressantes, sont disponibles dans cette section. Tous les tests ont été effectués avec le moteur igus MOT-AN-S-060-005-042-L-A-AAAA.

9.3.1 Méthode de mesure

Afin de mesurer les signaux du moteur stepper, j'ai effectué différentes configurations qui sont spécifiées pour chaque mesure. Pour ce faire, j'ai utilisé un analyseur logique pour les signaux partant du PIC en direction du driver et un oscilloscope pour les signaux partant du driver et allant au moteur pas à pas. De plus, j'ai ajouté 4 sondes de courant sur les 4 câbles du moteur afin de mesurer les courants traversant les enroulements.

9.3.2 Schéma de mesure

FIGURE 39 – Schéma de mesure signaux driver et signaux moteur



9.3.3 Résultats

Oscillogrammes

FIGURE 40 – Signaux sortant du MCU : Duty=75%, f=200kHz, v=300rpm

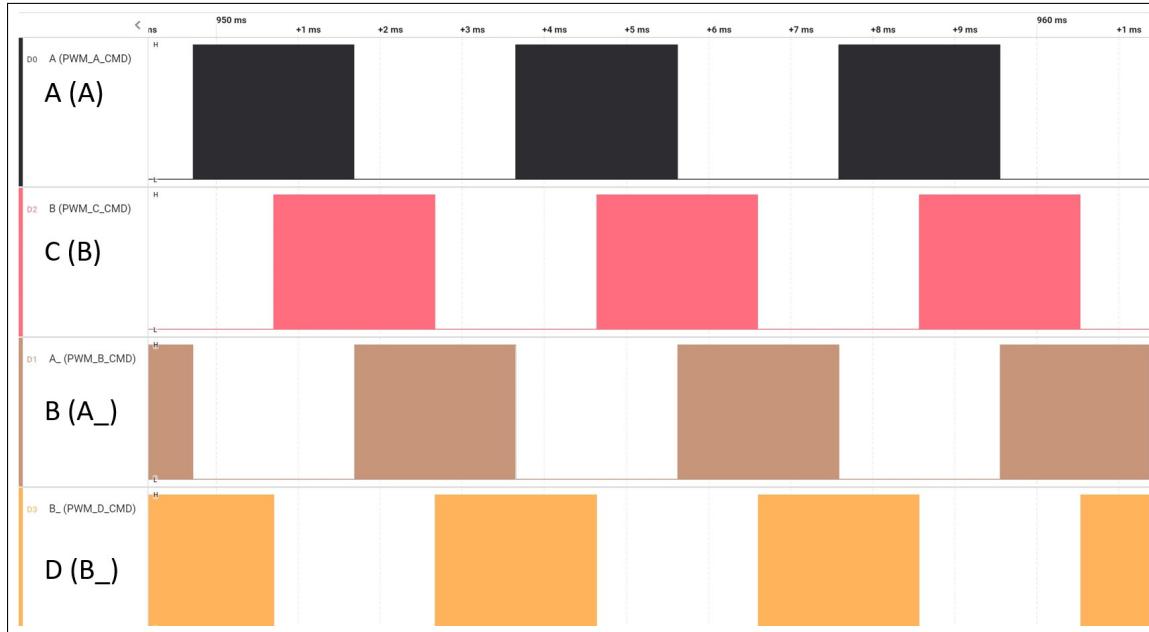


FIGURE 41 – Signaux sortant du driver : Duty=75%, f=200kHz, v=300rpm

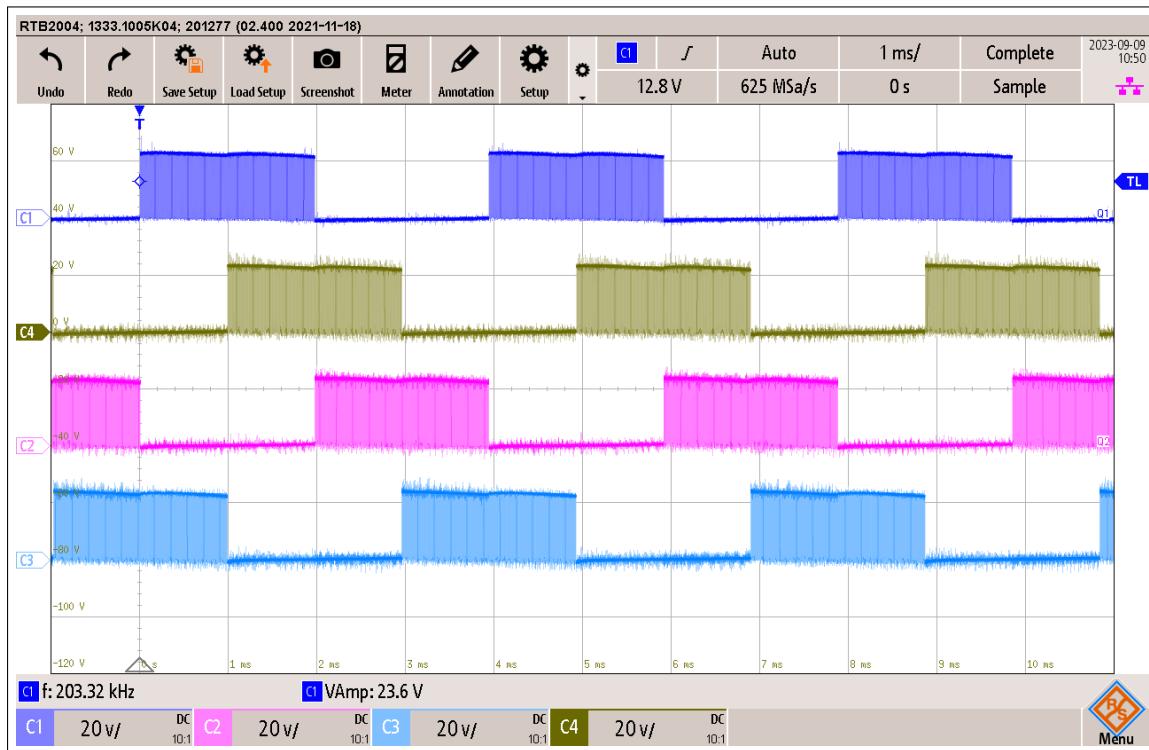


FIGURE 42 – Signaux sortant du driver (zoom) : Duty=75%, f=200kHz, v=300rpm

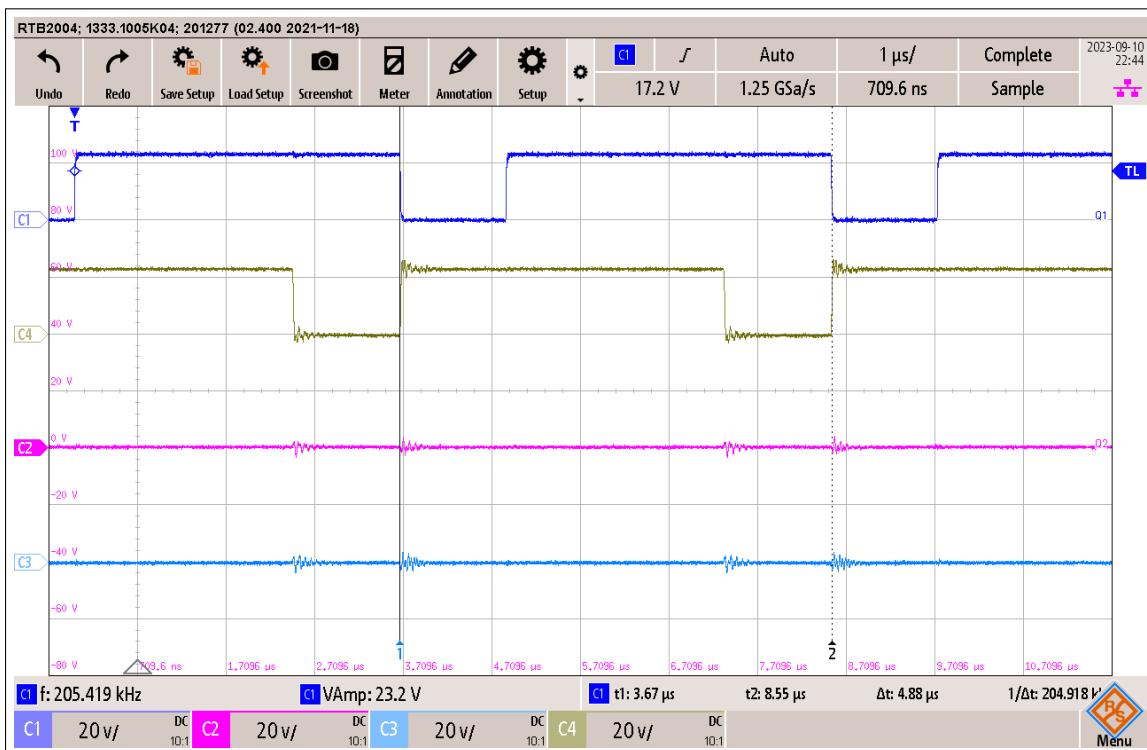


FIGURE 43 – Signaux sortant du driver : Duty=75%, f=200kHz, v=60rpm

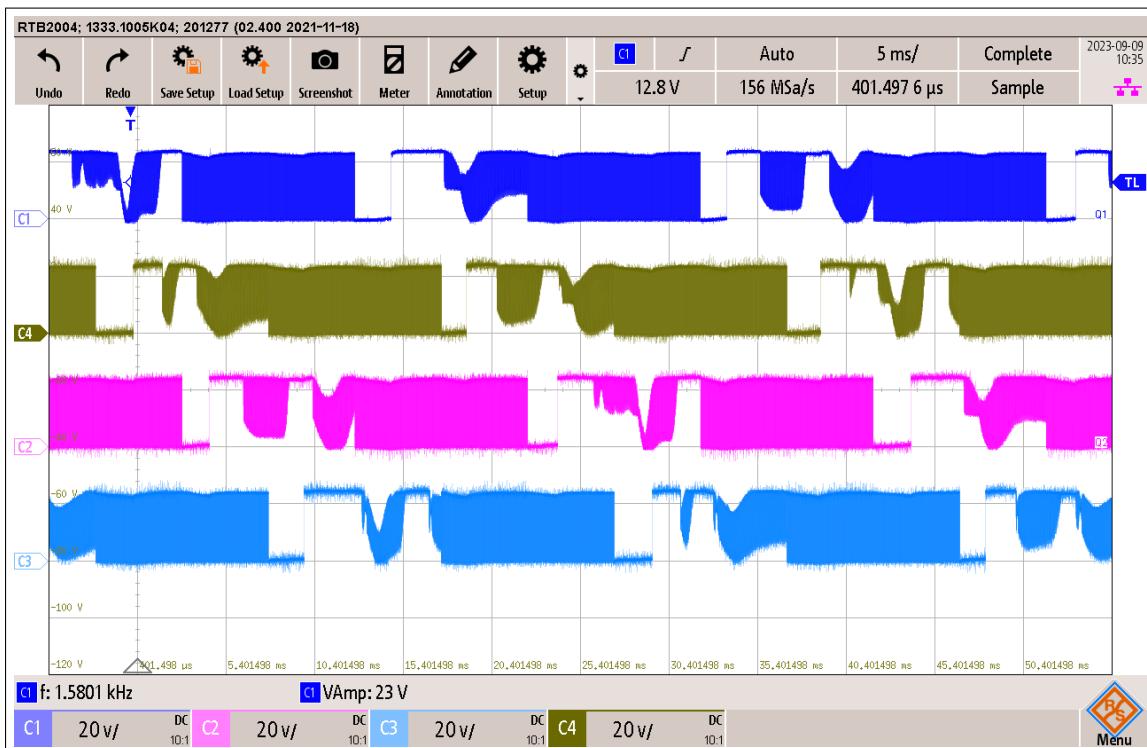


FIGURE 44 – Courants sortant du driver : Duty=95%, f=200kHz, v=300rpm

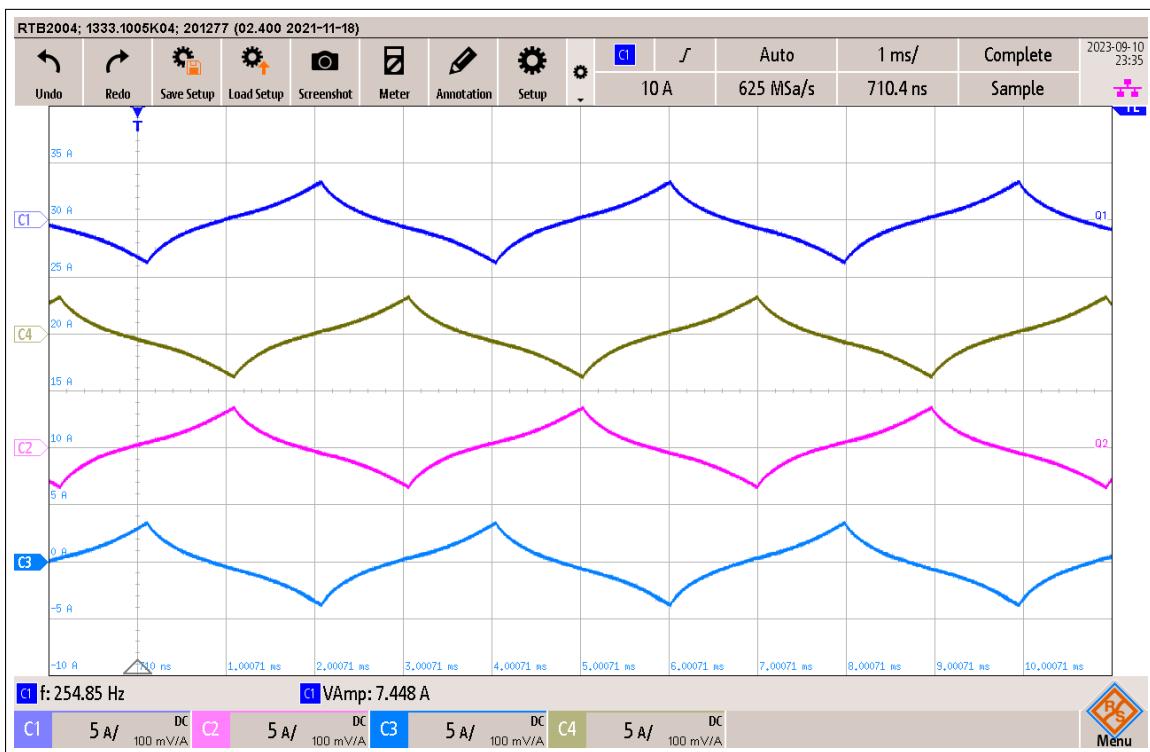
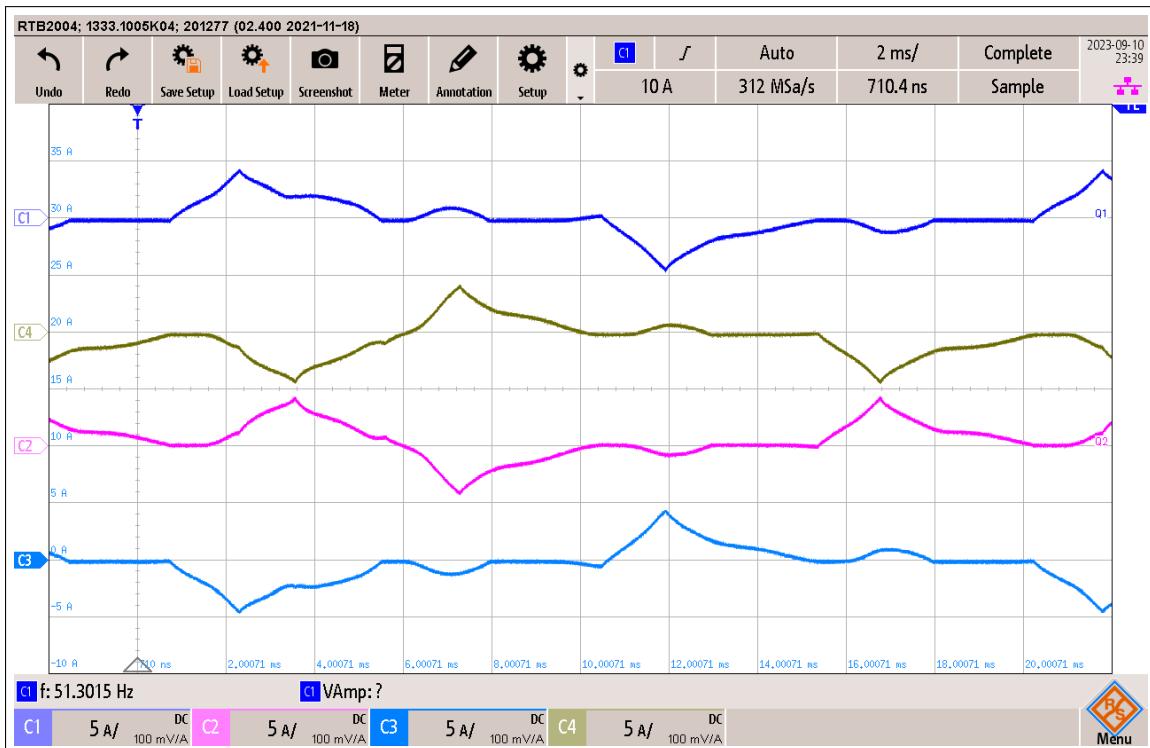


FIGURE 45 – Courants sortant du driver : Duty=95%, f=200kHz, v=60rpm



9.3.4 Analyse

Sur les deux premières Figures, nous pouvons constater une séquence de rotation du moteur en rotation horaire (CW) fonctionnelle. Nous pouvons voir également que les signaux sortant du driver correspondent bien à ceux sortant du MCU. Sur la Figure 43 nous pouvons observer le duty-cycle du PWM et confirmer qu'il est bien à 75% étant donnée que la période est de 4.88us et que la durée haute du signal est de 3.68us ce qui correspond à exactement 0.754%.

Sur la Figure 43, nous pouvons constater une distorsion significative des signaux émis par le driver. Cette capture a été réalisée en modifiant un seul paramètre, à savoir la vitesse de rotation. De plus, le moteur tourne à peine et présente d'importantes vibrations. Cela est vraisemblablement dû à un couple trop élevé à basse vitesse, ce qui rend en quelque sorte le moteur instable.

Les Figures 44 et 45 présentent les courants sortant du driver et entrant dans les enroulements du moteur. Lors de la première mesure, le comportement du moteur est correct, il ne perd pas de pas et tourne de manière fluide. En revanche, lors de la seconde mesure, le moteur tourne à peine et présente des vibrations importantes. Une observation importante est que sur la Figure 44, les signaux sont beaucoup plus réguliers et lisses que sur la Figure 45, où les signaux sont plus abrupts. Cependant, je n'ai pas poussé l'analyse des signaux plus loin en vue de caractériser le moteur, car cette responsabilité devrait incomber au fabricant du moteur. De plus, il est important de noter que le réglage du moteur en fonction du couple et de la vitesse ne devrait nécessiter qu'une seule configuration, et aucun problème significatif ne devrait survenir avec le temps si le réglage est effectué de façon correcte.

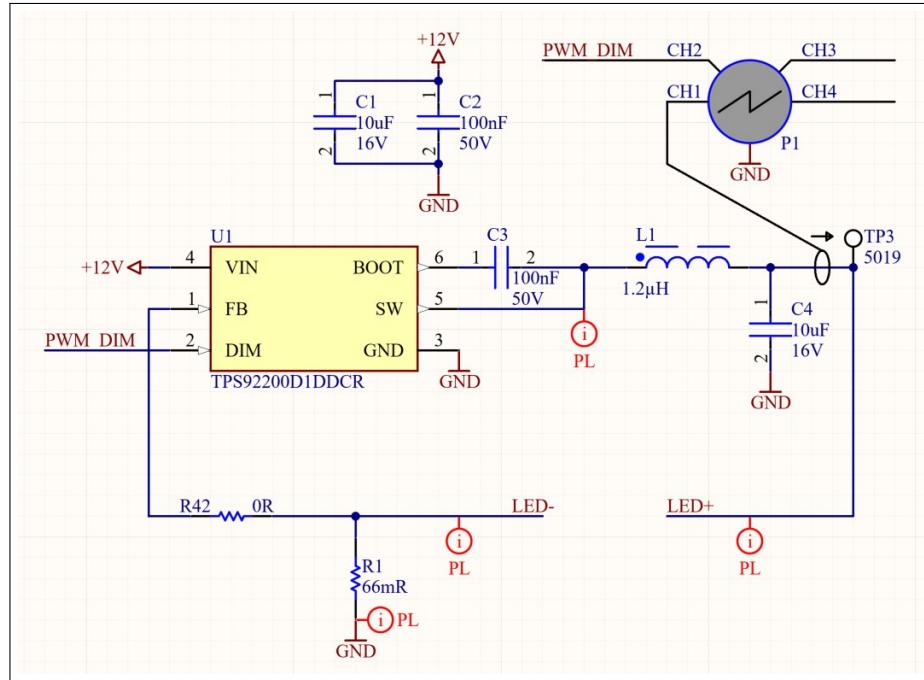
9.4 Driver LED de puissance

9.4.1 Méthode de mesure

Afin de mesurer le courant délivré par le driver, j'ai branché une sonde de courant sur l'anode des LEDs et j'ai lancé une séquence de 5 prises d'image.

9.4.2 Schéma de mesure

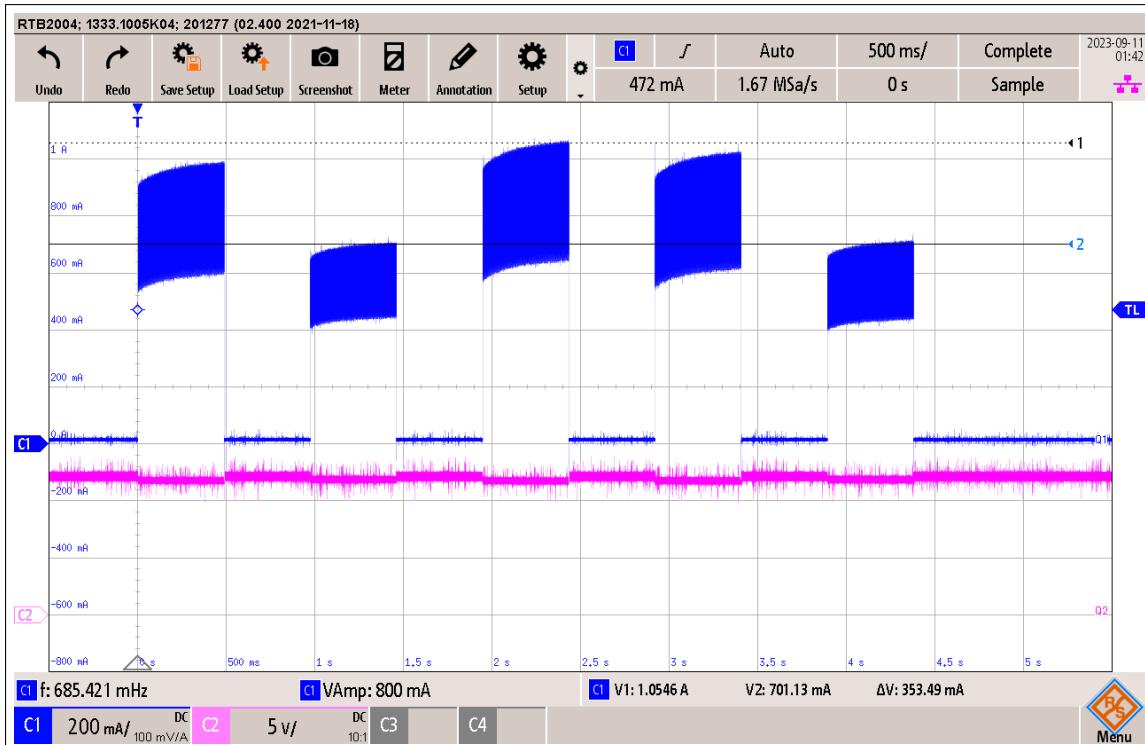
FIGURE 46 – Schéma de mesure



9.4.3 Résultat

Oscilloscopogramme

FIGURE 47 – Courant et tension dans les 5 LEDs



9.4.4 Analyse

Sur la Figure 47, une observation importante concerne le courant. Normalement, il devrait atteindre 1,5A, mais dans la capture, nous constatons qu'il atteint tout juste 1A pour 3 LEDs, tandis que les 2 autres sont à 700mA. Bien que ce problème n'ait pas été analysé en détail en raison de contraintes de temps, une première hypothèse est que la tension d'alimentation du driver est de 12V, alors que les LEDs nécessitent 11,6V pour fonctionner à leur puissance maximale. Cette légère différence de 0,4V pourrait être à l'origine du problème, et par conséquent, une solution potentielle consisterait à alimenter le driver avec la tension d'entrée du montage, soit 24V, afin d'obtenir une plus grande différence de tension et permettre ainsi au driver de fournir davantage de puissance. Le driver TPS92299D1DDCR peut être alimenté avec une tension allant jusqu'à 30V. Notons également que le courant de ripple est anormalement grand.

10 Résultat final

10.1 Éléments fonctionnels

Selon les critères définis dans le cahier des charges, le système est actuellement achevé à environ 90%. Les modes de fonctionnement manuel et automatique sont pleinement opérationnels, permettant à l'utilisateur de sélectionner et de configurer les paramètres de la séquence ainsi que du moteur. De plus, la fonction de sauvegarde des paramètres dans l'EEPROM fonctionne correctement. Toutefois, un élément demeure en attente de résolution : lors d'un premier test, le déclenchement de l'appareil photo Nikon n'a pas fonctionné comme prévu. Il est toutefois probable que ce problème soit attribuable à l'état désactivé du signal de mise au point (focus) au moment du test.

En termes généraux, l'intégration de la partie électronique dans le boîtier s'est effectuée sans encombre. Le circuit ne présente pas de problèmes majeurs, à l'exception du courant délivré par le driver de LED, qui est inférieur aux attentes. Quelques détails nécessitent encore des ajustements, tels que la résolution de quelques problèmes techniques et l'incorporation de nouveaux paramètres de contrôle, pour parvenir à une réalisation complète et optimale du projet.

10.2 Éléments à corriger

Une modification potentielle au niveau hardware pour une éventuelle nouvelle version concerne le filtre de l'encodeur rotatif PEC12R. Le schéma fourni par le fabricant a été mal recopié sur Altium, et les résistances de pull-up R16 et R17 ne sont pas correctement positionnées. Au lieu d'être connectées après R18 et R19, elles auraient dû être placées avant, directement à la sortie du canal A et du canal B du PEC12R.

10.3 Problèmes rencontrés

Pendant la réalisation de ce travail de diplôme, j'ai été confronté à divers défis auxquels j'ai réussi à faire face avec des résultats plus ou moins bons.

Le premier défi a été de concevoir le système tout en ayant des incertitudes quant au choix du moteur, qui n'a été fixé que vers le début de la dernière semaine de mon travail. Cette situation découle du fait que le choix du moteur devait être décidé conjointement avec le diplômé en mécanique, qui n'a pas commencé son travail en avance contrairement à moi. J'ai réussi à résoudre ce problème en développant un système flexible capable d'utiliser deux types de moteurs complètement différents, ainsi qu'une alimentation du système variable.

En deuxième lieu, la gestion du temps s'est avérée complexe pour moi, car la recherche et la documentation sur les composants demandent beaucoup de temps, en particulier lorsque les documentations sont longues et que les points essentiels ne sont pas forcément connus d'avance, en particulier lors de l'utilisation de technologies jamais explorées auparavant.

Ensuite, la gestion du moteur a été un défi en raison des nombreux paramètres des modules MCPWM, qui offrent d'un côté une grande flexibilité mais rendent la configuration complexe, surtout lorsqu'on les utilise pour la première fois. De plus, les moteurs pas à pas ne disposent pas d'une documentation très détaillée en ce qui concerne leur contrôle. Les informations disponibles sont davantage axées sur l'aspect mécanique que sur l'aspect électronique.

11 Conclusion

Ce projet de diplôme avait pour objectif de concevoir un système automatisé visant à simplifier la capture d’images pour la Reflectance Transformation Imaging (RTI). Il a été divisé en deux parties distinctes, avec la réalisation électronique relevant de ma responsabilité, tandis que la partie mécanique a été confiée à un diplômé en génie mécanique.

L’électronique du système a été chargée de plusieurs tâches cruciales, notamment la gestion des LEDs de puissance pour un éclairage optimal lors des prises de vue, la commande précise du moteur pour le positionnement, ainsi que la coordination de la séquence et le déclenchement de l’appareil photo.

Ce projet a été l’occasion de mettre en pratique des compétences techniques avancées dans le domaine de l’électronique et de la mécanique. Il a également illustré l’importance de la collaboration interdisciplinaire, où l’expertise de chacun a contribué à la réalisation de l’ensemble du système.

Dans l’ensemble, je considère que le développement de ce système a été globalement positif, bien qu’il nécessite encore quelques heures de travail pour résoudre certains problèmes de firmware et hardware et ajouter certaines fonctionnalités.

Je tiens à exprimer ma gratitude envers Monsieur Serge Castoldi pour son précieux accompagnement tout au long de ce projet. Un grand merci également à Monsieur Mathieu Bernard-Reymond, le mandant du projet, pour sa confiance et son soutien constants. Ce projet n’aurait pas pu se concrétiser sans leur précieuse contribution.

Lausanne, le 11 septembre 2023

Meven Ricchieri

12 Annexes

- Cahier des charges
- Schémas
- Drawings
- Bill of material
- Codes C et H
- Document de modification
- Planning
- Journal de travail
- Mode d'emploi
- Références

12.1 Cahier des charges


CAHIER DES CHARGES
ETNL-ES
DIPLOME

**Automatisation d'un système de prise
d'image RTI
N° de projet 2309**

1 Objectif

Sur la base d'un concept mécanique en cours de conception (diplôme mécanique) et en collaboration avec l'étudiant exécutant ce travail de diplôme, réaliser une carte électronique permettant de piloter un dispositif permettant une prise d'image RTI.

La commande des éléments suivants devra être séquencée :

- pilotage de sources lumineuses (LED de puissance),
- positionnement via moteur,
- prise d'image (appareil photo).

1.1 Données en lien avec l'objectif

Il s'agit de réaliser une carte électronique basée sur un microcontrôleur PIC32 permettant la commande du système.

Au niveau hardware :

- Alimentation via bloc secteur tension standard DC (5V, 12V ou 24V).
- Prévoir un LCD alphanumérique et des boutons et/ou encodeur afin de régler les paramètres de la séquence, la démarrer/stopper et afficher l'état.
- Un buzzer permettant de signaler un bip sonore de signalisation.
- Les éléments suivant devront être pilotés :
 - Sources lumineuses : LED de puissance.
 - Probablement 4 ou 5 LED, nombre précis à définir par concept du travail de diplôme mécanique.
 - Voir [1] pour les modèles de LED et driver actuellement utilisés.
 - Évaluer possibilité autre modèle plus puissant.
 - Moteur de positionnement.
 - Choix et dimensionnement du modèle en collaboration avec travail de diplôme mécanique.
 - La position actuelle doit pouvoir être connue par le MCU (moteur pas à pas ou retour encodeur + index)
 - Déclenchement appareil photo :
 - Commande via câble, optocouplée.
 - En option et en supplément à la commande câblée : commande IR.
Implémenter le système en priorité pour un Canon D750. Compatibilité également souhaitée pour un Sony A7R.
Voir [2] pour des exemples de codes IR pour le Sony A7R..





- Au niveau connectique, prévoir de la connectique robuste, détrompée et facile à brancher/connecter.

Au niveau firmware :

- Viser une commande simple et intuitive.
- Émettre un bip à la fin de la série de prises de vue.
- Sauvegarder les réglages en mémoire non volatile, pour restauration à la prochaine mise sous tension.
- Les fonctionnalités suivantes devront être implémentées :
 - Réglage du temps d'allumage des LED pour s'adapter au temps de pose.
 - Réglage de la durée entre chaque changement de LED.
 - 2 modes de fonctionnement :
 - tout automatique (avance auto). Appui sur un bouton pour déclenchement de la séquence de déplacement et prises de vue tous les 15 à 20 degrés environ (en fonction du concept mécanique). Une séquence complète ~ 200 photos.
 - ou manuel (juste une séquence avec une image par LED, sans déplacement automatique)
 - Déclenchement de l'appareil photo pour la prise de vue (câblé, et en option IR – voir hardware ci-dessus)

Au niveau mécanique :

Une mise en boîtier est demandée.

1.2 Découpage des tâches proposée

- Etudier ce qui existe actuellement (voir [1]).
- Proposer un concept global, en collaboration avec le travail de diplôme de mécanique.
- Si nécessaire faire des tests de composants (led, moteur, déclenchement appareil photo)
- Designer la carte en prévoyant une mise en boîtier et la connectique.
- Faire fabriquer la carte, approvisionner les composants.
- Pendant fabrication : débuter programmation sur kit
- Assembler la carte
- Mettre en service carte et implémenter le FW
- Documentation et rapport à prévoir tout au long du projet.

2 A l'issue du projet de diplôme, l'étudiant fournira (liste non exhaustive)

- Les fichiers sources de CAO électronique du/des PCB réalisé(s) + configuration logicielle (versions utilisées)
Les symboles schématiques, footprints et modèles 3D doivent être inclus dans une librairie locale.
- Tout le nécessaire pour fabriquer un exemplaire hardware :
Fichiers de fabrication (GERBER) / liste de pièces avec références pour commande (BOM) / implantation (prototype) / modifications, etc
- Fichiers sources de programmation microcontrôleur (.c / .h) + configuration logicielle (versions utilisées).
- Tout le nécessaire pour programmer le microcontrôleur (logiciel et fichiers .hex).
- Le cas échéant, fichiers sources d'application Windows/Linux/Android + configuration logicielle (versions utilisées)
- Tout le nécessaire à l'installation de programmes ou autres environnements utilisés.
- Le cas échéant, les dessins mécaniques des pièces réalisées.
- Un mode d'emploi du système
- Un calcul / estimation des coûts
- Un rapport complet contenant les concepts, calculs, dimensionnement, structogrammes, etc. et toutes les annexes nécessaires à la compréhension, réalisation et reprise du projet.
- Le/les HW réalisé(s) avec toutes les fonctions disponibles et dans l'état final (câblé(s), programmé(s), mis en boîtier, etc.)



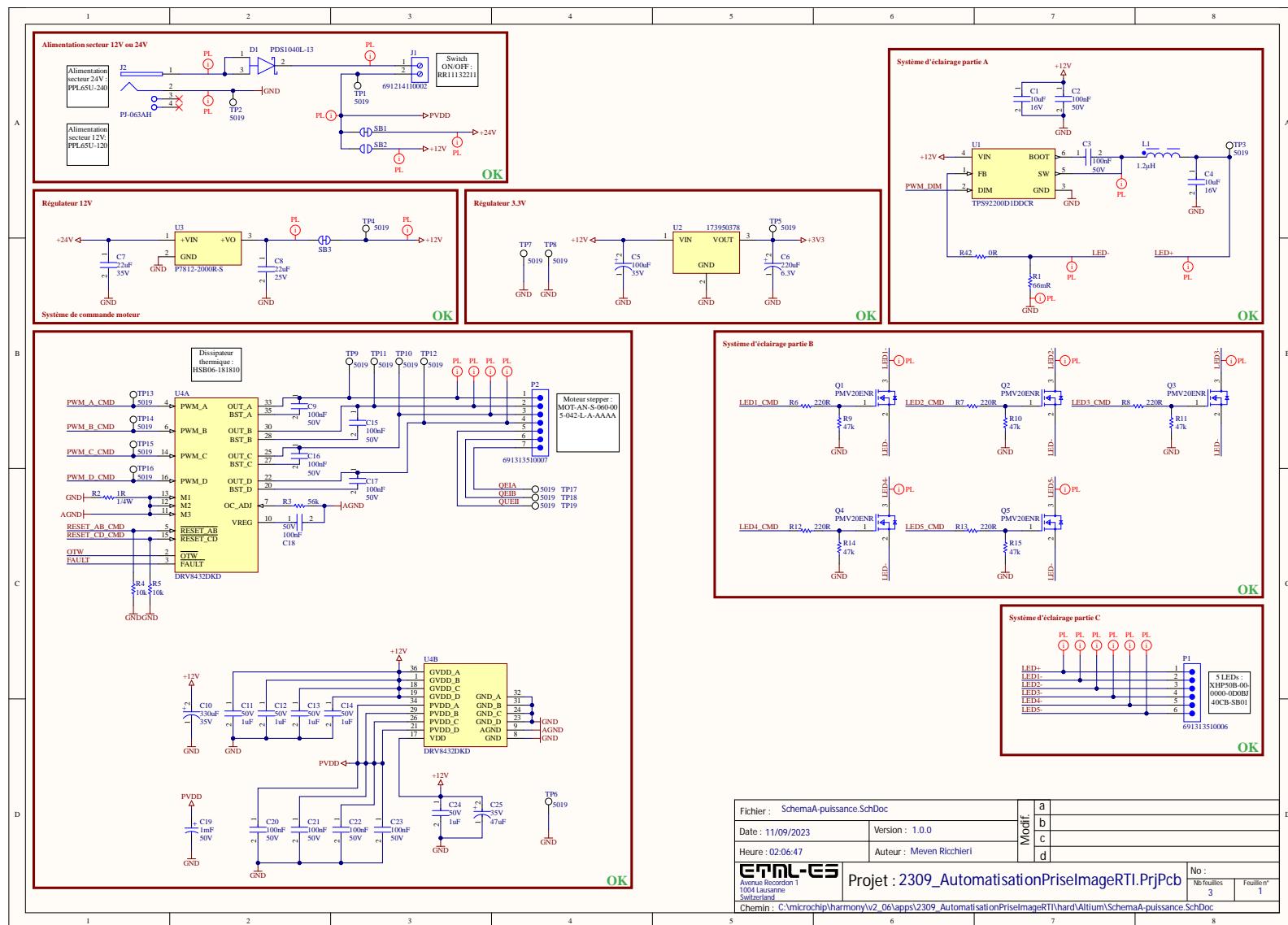
3 Autres demandes / contraintes / conseils

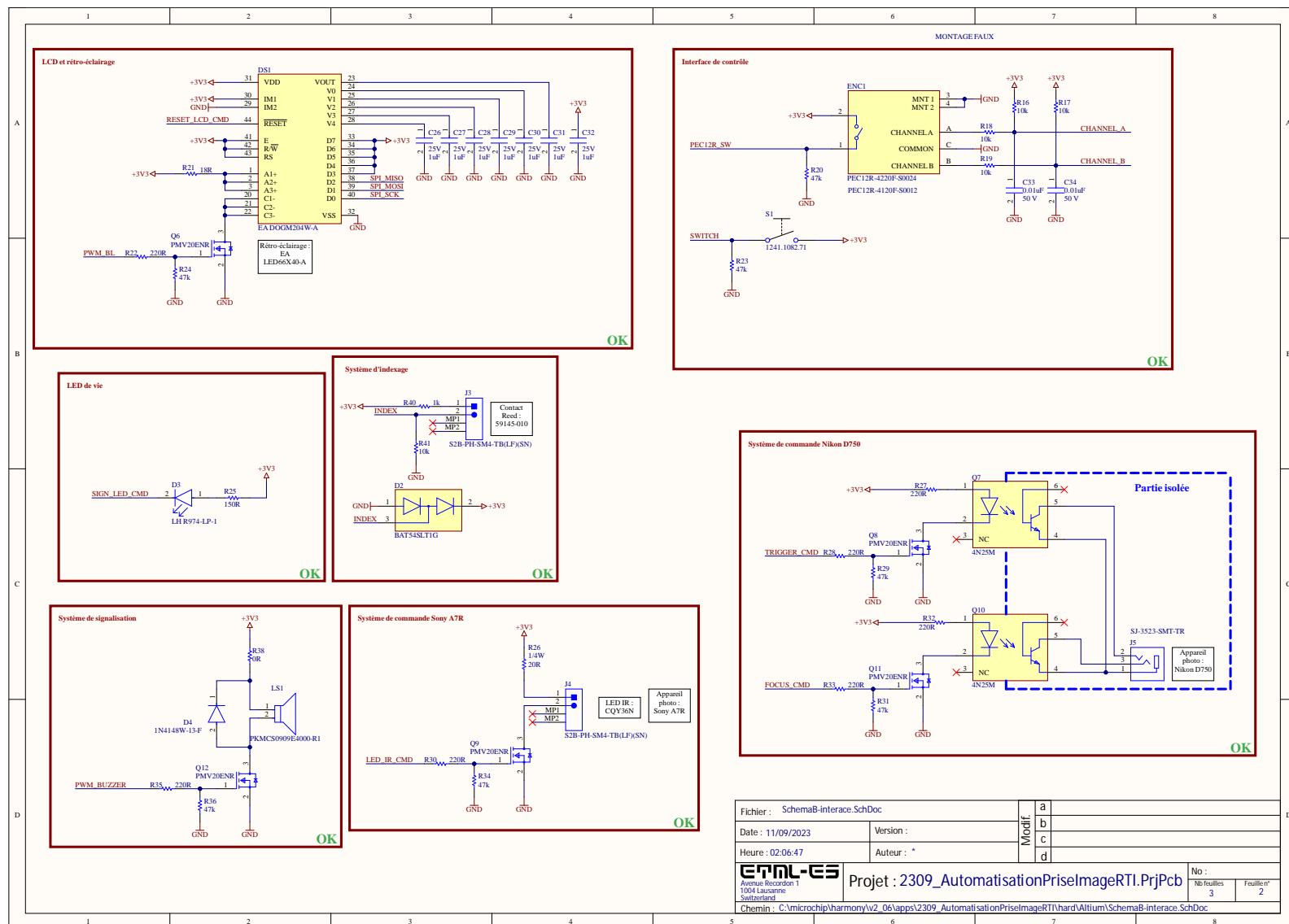
- **Planifier** dans le détail les travaux demandés.
- Se référer au planning régulièrement, **vérifier son avancement**, rédiger son **journal de projet** quotidiennement.
- Commencer à **rédiger le rapport de diplôme le plus tôt possible**, et régulièrement tout au long du travail de diplôme.
- Prendre du temps, préparer sa réflexion, rechercher des apports théoriques et des exemples pratiques, **envisager plusieurs possibilités** avant de finaliser une solution.
- **Numéroter et dater tous les documents**
- En cas de **problème** (retard, objectif à revoir, difficulté rencontrée, etc.), se référer à l'enseignant et au mandant au plus vite.
- Toutes les **décisions importantes**, tant au niveau technique qu'organisationnel, doivent être posées **par écrit** dans le PV de séance, le rapport de diplôme et /ou figurer dans le journal de projet, après discussion avec l'enseignant / le mandant.

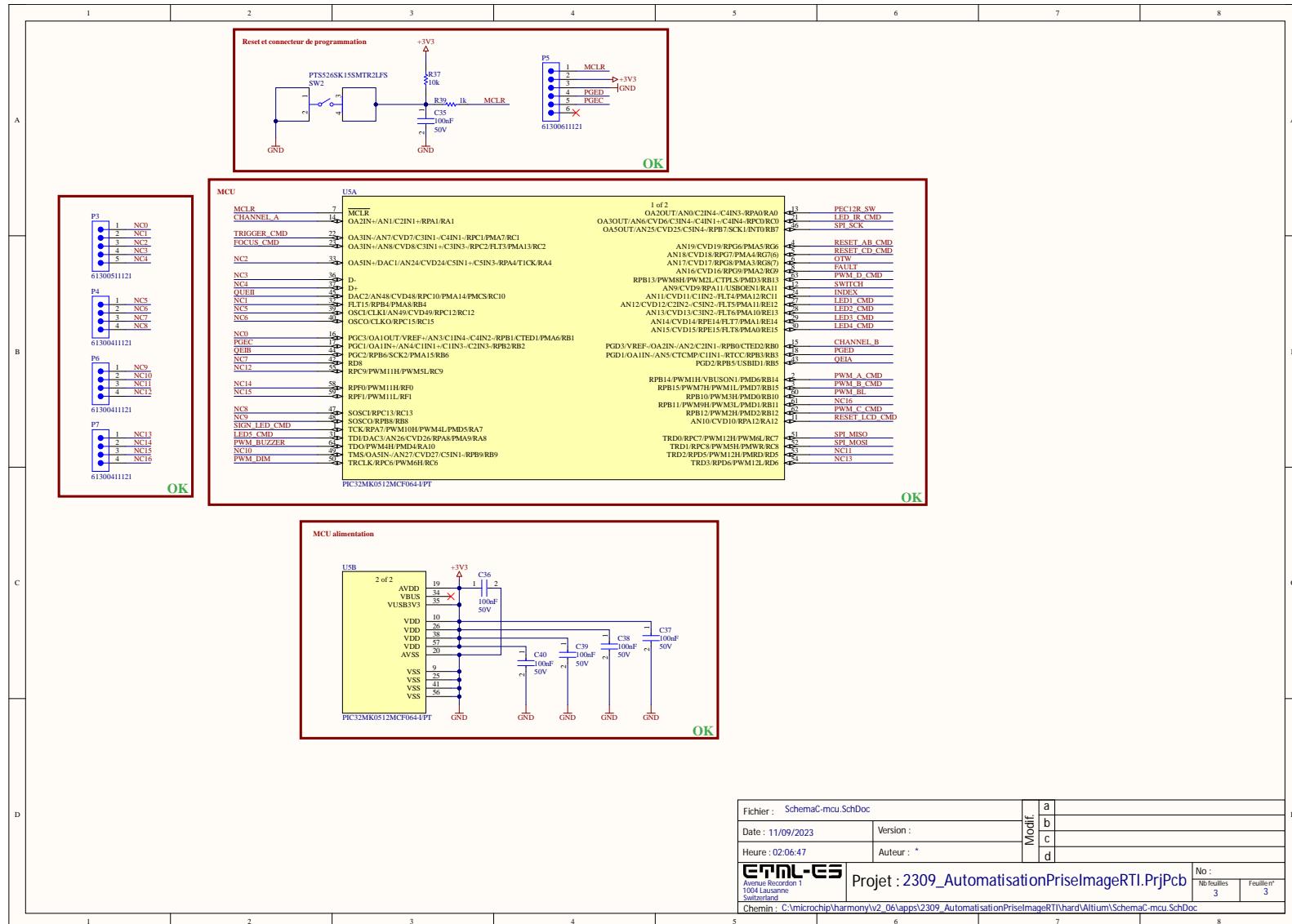
4 Documents de références

- [1] : Description du projet actuel et du matériel utilisé, fonctionnalités voulues.
<https://matbdrd.notion.site/Projet-RTI-au-MCAH-5ca2b01f4e9e4958a5f73d3760e3fb7e>
- [2] IR codes pour Sony A7r
<https://diydrones.com/forum/topics/sony-a7-infrared-codes>

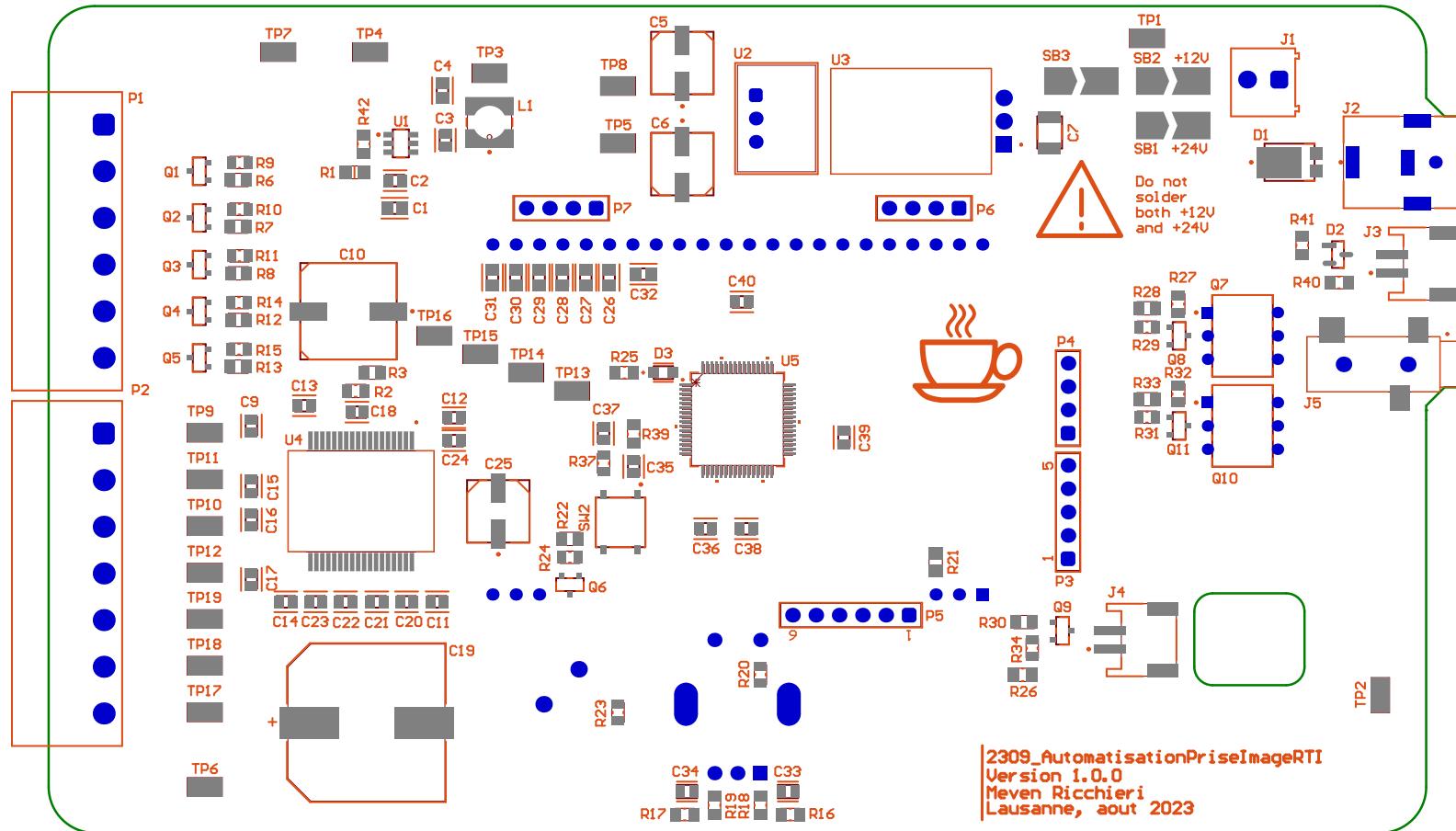
12.2 Schémas

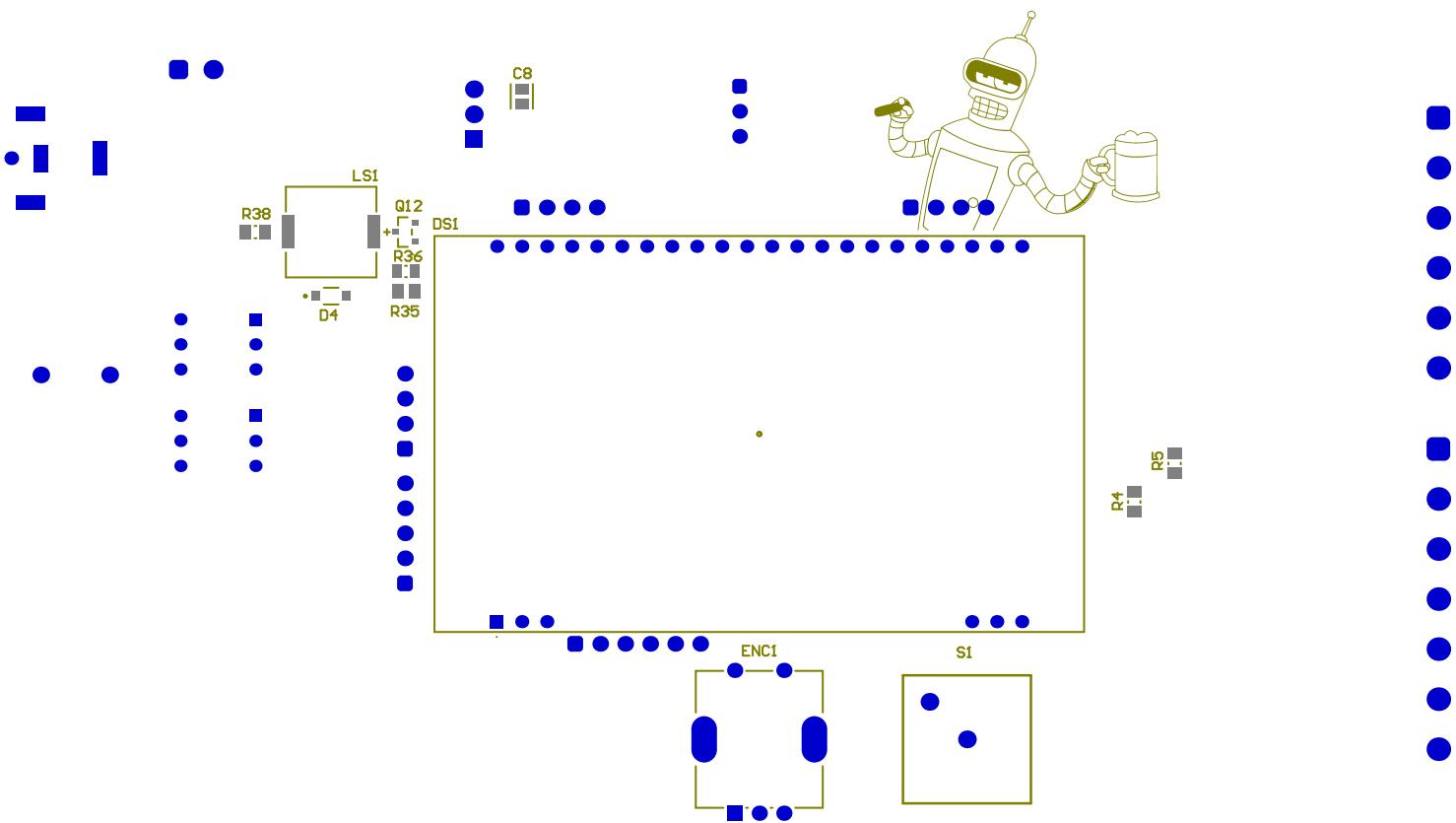






12.3 Drawings





12.4 Bill of material

Name	Description	Designator	sur PCB	Stock ES	Manufacturer 1	Manufacturer Part Number 1	Manufacturer Lifecycle 1	Supplier 1	Supplier Part Number 1	Quantity	Supplier Unit Price 1	Supplier Subtotal 1
Solder bridge 3mm close		SB1, SB3		Y						2		
Solder bridge 3mm open		SB2		Y						1		
4714	JST-PH 2-PIN JUMPER CABLE - 100M		N	N	Adafruit Industrie	4714	Unknown	Mouser	485-4714	0	0	0
CRCW0805210RKFKA		R27, R32		Y	Bourns	CR0805-FX-2100ELF	Volume Production	Digi-Key	11B-CR0805-FX-2100ELF	2	0.1	0.2
PEC12R-4220F-S0024	ROTARY ENCODER MECHANICAL 24PPR	ENC1		Y	Bourns	PEC12R-4220F-S0024	Volume Production	Digi-Key	PEC12R-4220F-S0024-ND	1	1.62	1.62
P7812-200R-S		U3		N	CUI	P7812-200R-S	Unknown	Digi-Key	102-P7812-200R-S-ND	1	5.96	5.96
H5B06-181810	Dissipateur thermique BGA Alliage d'aluminium 3,2W à 75°C Montage supérieur		N	N	CUI Devices	H5B06-181810	Unknown	Digi-Key	2223-H5B06-181810-ND	0	0	0
PI-063AH	Through Hole Right Angle DC Power Jack, 8 A, 2 mm Center Pin, 2 Position, -25 to 85 degC, RoHS, Bulk	J2		N	CUI Devices	PI-063AH	Volume Production	Digi-Key	2223-PI-063AH-ND	1	1.55	1.55
SI-3523-SMT-TR		J5		N	CUI Devices	SI-3523-SMT-TR	Volume Production	Digi-Key	CP-3523SMTK-ND	1	1.01	1.01
1N4148W-13-F	DIODE GEN PURP 100V 150MA SOD123	D4		N	Diodes	1N4148W-13-F	Volume Production	Digi-Key	1N4148W-13FDIET-ND	1	0.18	0.18
PDS1040L-13	DIODE SCHOTTKY 40V 10A POWERDIS	D1		N	Diodes	PDS1040L-13	Volume Production	Digi-Key	PDS1040LDICT-ND	1	1	1
EA DOGM204W-A		DS1		Y	Display Visions	EADOGM204W-A	Unknown	Digi-Key	1481-1087-ND	1	26.64	26.64
EA LED66X40-A		Y		EA LED66X40-A	EA LED66X40-A	Manual Solution	Digi-Key	1481-1166-ND	1	8.07	8.07	
RR11132211		N		E-Switch	RR11132211	Volume Production	Mouser	612-RR11132211	1	0.49	0.49	
PTS5265KSMTR2LF5	TACT 5.2 X 5.2, 1.5 MM H, 260GF,	SW2		N	ITT & K	PTS5265KSMTR2LF5	Unknown	Newark	35AH5137	1	0.055	0.055
C0805C103ISRAC7210	CAP CER SMD 0805 .01UF X7R 5% 50	C33, C34		N	KEMET	C0805C103ISRAC7210	Volume Production	Digi-Key	399-17615-1-ND	2	0.1	0.2
5019	Test Point, 1 Position SMD, RoHS, Tape and Reel	TP1, TP2, TP3, TP4, TP5, TP6, TP7, TP8, TP9, TP10	N	N	Keystone Electrof	5019	Volume Production	Digi-Key	36-5019CT-ND	19	0.34	6.46
RK73B2ATTDTD1R0J		R2		N	KOA Speer	RK73B2ATTDT1R0J	Volume Production	Digi-Key	2019-RK73B2ATTDTD1R0J	1	0.1	0.1
RK73B2ATTDTD200J		R26		N	KOA Speer	RK73B2ATTDTD200J	Volume Production	Digi-Key	2019-RK73B2ATTDTD200J	1	0.1	0.1
59145-010	Reed contact		N	N	Littlefuse Hamlin	59145-010	Volume Production	Digi-Key	F8075-ND	1	3.36	3.36
PIC32MK0512MCF064-I/P		U5		Y	Microchip	PIC32MK0512MCF064-I/PT	Volume Production	Digi-Key	PIC32MK0512MCF064-I/	1	9.26	9.26
PKMCS0909E-4000-R1	AUDIO PIEZO TRANSDUCER 12.5V SMD	L51		N	Murata	PKMCS0909E-4000-R1	Volume Production	Digi-Key	490-9647-1-ND	1	0.88	0.88
PMV20ENR	MOSFET N-Ch 30V SOT23	Q1, Q2, Q3, Q4, Q5, Q6, Q8, Q9, Q11, Q12	N	Nexperia	PMV20ENR	Volume Production	Digi-Key	1727-2301-1-ND	10	0.288	2.88	
BAT54SLT1G	DIODE ARRAY SCHOTTKY 30V SOT23-3	D2		N	ON Semiconductor	BAT54SLT1G	Volume Production	Digi-Key	BAT54SLT1G-0GSC7-ND	1	0.15	0.15
4N25M	4N25M Optocoupler DC-IN 1-CH Transistor With Base DC-OUT 6-Pin PDIP Tube - Bulk	Q7, Q10		N	ON Semiconductor	4N25M	Volume Production	Newark	98K9084	2	0.229	0.458
LH R974-LP-1	LED RED DIFUSED 0805 SMD	D3		Y	Osram Opto	LH R974-LP-1	Volume Production			1		
EEE-1VA01XP	CAP ALUM 100uF 20% 35V SMD	C5		N	Panasonic	EEE-1VA01XP	Volume Production	Digi-Key	PCE3951CT-ND	3	0.5	0.5
EEE-1VA331P	CAP ALUM 330uF 20% 35V SMD	C10		N	Panasonic	EEE-1VA331P	Volume Production	Mouser	667-EE-1VA331P	1	0.66	0.66
EEE-FX0221P	CAP ALUM 220uF 20% 6.3V SMD	C6		N	Panasonic	EEE-FX0221P	Volume Production	Mouser	667-EE-FX0221P	1	0.42	0.42
EEEFP1V470AP	CAP ALUM 47uF 20% 35V SMD	C25		N	Panasonic	EEEFP1V470AP	Volume Production	Digi-Key	PCE45520KR-ND	1	0.52	0.52
ER16ENF1880V		R21		Y	Panasonic	ER1-6ENF1880V	Volume Production	Digi-Key	P18.0CCT-ND	1	0.1	0.1
ER16ENF2200V		R6, R7, R8, R12, R13, R22, R28, R30, R33, R35	Y	N	Panasonic	ER1-6ENF2200V	Volume Production	Digi-Key	P20.0CCT-ND	10	0.079	0.79
ER16ENF5602V		R3		Y	Panasonic	ER1-6ENF5602V	Volume Production	Digi-Key	P56.0KCT-ND	1	0.1	0.1
S2B-PH-SM4-TB(LF)(SN)	Male Header, Pitch 2 mm, 1x 2 Position, Height 5.5 mm, -25 to 85 degC, RoHS, Tape and Reel	J3, J4		N	S2B-PH-SM4-TB	Manual Solution	Digi-Key	455-S2B-PH-SM4-TBDR	2	0.53944	1.08	
CL21A106KQDNNE	CL21 Series 0805 10uF 10V ±10% Tolerance X5R Multilayer Ceramic Chip Capacitor	C1, C4		N	Samsung	CL21A106KQDNNE	Volume Production	Digi-Key	1276-1095-1-ND	2	0.1	0.2
CL21A226MAYNNNE	CL21 Series 0805 22uF 25V ±20% Tolerance X5R Multilayer Ceramic Chip Capacitor	C8		N	Samsung	CL21A226MAYNNNE	Volume Production	Digi-Key	1276-1095-1-ND	1	0.25	0.25
CL21B104KBCNNNC	CL21 Series 0805 100nF 50V ±10% Tolerance X7R Multilayer Ceramic Chip Capacitor	C2, C3, C9, C15, C16, C17, C18, C20, C21, C22, C23	N	Samsung	CL21B104KBCNNNC	Volume Production	Digi-Key	1276-1003-1-ND	1	0.044	0.044	
CL21B105KBPNNN	CL21 Series 0805 1uF 50V ±10% Tolerance X7R Multilayer Ceramic Chip Capacitor	C11, C12, C13, C14, C24	N	Samsung	CL21B105KBPNNN	Volume Production	Digi-Key	1276-6470-1-ND	5	0.11	0.55	
CL21F105ZAPFNNNC	CL21 Series 0805 1uF 25V -20/+80% Tolerance Y5V Multilayer Ceramic Chip Capacitor	C26, C27, C28, C29, C30, C31, C32	Y	Samsung	CL21F105ZAPFNNNC	Volume Production			7			
1241.1082_71		S1		Y	Schurter	1241.1082_71	Volume Production	Digi-Key	486-7033-ND	1	4.17	4.17
ERJ-060U66MV	Resistor	R1		N	Susumu	ERJ-060U66MV	Volume Production	Digi-Key	RL12201-R068-G	1	0.38	0.38
GMK32B226MM-P	None	C7		N	Taiyo Yuden	GMK32B226MM-P	Unknown	Digi-Key	587-5853-1-ND	1	0.96	0.96
DRV8432D0K	12A Dual Brushed DC or Single Bipolar Stepper Motor Driver (PWM Ctrl), 0 to 52.5V, -40 to 85 degC	U4		N	Texas Instruments	DRV8432D0K	Volume Production			1		
TPS92200DDCR		U1		N	Texas Instruments	TPS92200DDCR	Volume Production	Digi-Key	296-TPS92200DDCR	1	1.2	1.2
TG-A1250-15-15-10	THERM PAD 15MMX15MM GREEN		N	N	T-Global Techno	TG-A1250-15-15-1-0	Unknown	Digi-Key	1168-TG-A1250-15-15-1	0	0	0
CQY36N			N	N	Vishay Semicond	CQY36N	Volume Production	Arrow Electron	CQY36N	1	0.5953	0.5953
173959378	173950378 Module DC-DC 1-OUT 3.3V 0.5A 3-Pin SIP Module Tube	U2		N	Wurth Electronics	173959378	Volume Production	Digi-Key	732-8242-5-ND	1	6.34	6.34
7440430012		L1		N	Wurth Electronics	7440430012	Volume Production	Digi-Key	732-1093-1-ND	1	1.53	1.53
61300411121		P4, P6, P7		N	Wurth Electronics	61300411121	Volume Production	Digi-Key	732-5317-ND	3	0.19	0.57
61300511121		P3		N	Wurth Electronics	61300511121	Volume Production	Digi-Key	732-5318-ND	1	0.26	0.26
61300611121		P5		N	Wurth Electronics	61300611121	Volume Production	Digi-Key	732-5319-ND	1	0.35	0.35
691214110002		J1		N	Wurth Electronics	691214110002	Volume Production	Digi-Key	732-2747-ND	1	0.88	0.88
691313510007	6 poles connector	P1		N	Wurth Electronics	691313510007	Volume Production	Digi-Key	732-2074-ND	1	1.32	1.32
691313510007	7 poles connector	P2		N	Wurth Electronics	691313510007	Volume Production	Digi-Key	732-691313510007-ND	1	1.69	1.69
691351500006	6 poles connector		N	N	Wurth Electronics	691351500006	Volume Production	Digi-Key	732-2044-ND	1	3.82	3.82
691351500007	7 poles connector		N	N	Wurth Electronics	691351500007	Volume Production	Digi-Key	732-691351500007-ND	1	4.35	4.35
865080663020	865080663020 WCAP-ASL1 Aluminum Electrolytic Capacitors	C19		N	Wurth Electronics	865080663020	Volume Production	Digi-Key	732-8465-1-ND	1	2.92	2.92
XHP508-00-0000-000B04C0	Power LED mounted on a PCB		N	N	XHP508-00-0000-XHP508-00-0000-000B04C0	Manual Solution	Digi-Key	1672-1076-ND	5	12.29	61.44	
RC0805FR-071KL	Chip Resistor, 1 KOhm, +/- 1%, 125 mW, -55 to 155 degC, 0805 (2012 Metric), RoHS, Tape and Reel	R39, R40		Y	Yageo	RC0805FR-071KL	Volume Production	Digi-Key	311-1.00KCRCT-ND	2	0.1	0.2
RC0805FR-0747KL	Chip Resistor, 47 KOhm, +/- 1%, 0.125 W, -55 to 155 degC, 0805 (2012 Metric), RoHS, Tape and Reel	R9, R10, R11, R14, R15, R20, R23, R24, R29, R31, R3Y	Y	Yageo	RC0805FR-0747KL	Volume Production	Digi-Key	311-47.0KCRCT-ND	12	0.024	0.288	
RC0805JR-070RL	Chip Resistor, 0 Ohm, Jumper, +/- 5%, 0.125 W, -55 to 155 degC, 0805 (2012 Metric), RoHS, Tape and Reel	R38, R42		Y	Yageo	RC0805JR-070RL	Volume Production	Digi-Key	311-0.0ARCT-ND	2	0.1	0.2
RC0805JR-0710KL	Chip Resistor, 10 KOhm, +/- 5%, 0.125 W, -55 to 155 degC, 0805 (2012 Metric), RoHS, Tape and Reel	R4, R5, R16, R17, R18, R19, R37, R41	Y	Yageo	RC0805JR-0710KL	Volume Production	Digi-Key	311-10KARCT-ND	8	0.022	0.22	
RC0805JR-0775RL		R25		Y	Yageo	RC0805JR-0775RL	Volume Production	Digi-Key	311-75ARCT-ND	1	0.1	0.1

12.5 Codes C et H

```

1  ****
2  MPLAB Harmony Application Source File
3
4  Company:
5      Microchip Technology Inc.
6
7  File Name:
8      app.c
9
10 Summary:
11     This file contains the source code for the MPLAB Harmony application.
12
13 Description:
14     This file contains the source code for the MPLAB Harmony application. It
15     implements the logic of the application's state machine and it may call
16     API routines of other MPLAB Harmony modules in the system, such as drivers,
17     system services, and middleware. However, it does not call any of the
18     system interfaces (such as the "Initialize" and "Tasks" functions) of any of
19     the modules in the system or make any assumptions about when those functions
20     are called. That is the responsibility of the configuration-specific system
21     files.
22 ****
23
24 // DOM-IGNORE-BEGIN
25 ****
26 Copyright (c) 2013-2014 released Microchip Technology Inc. All rights reserved.
27
28 Microchip licenses to you the right to use, modify, copy and distribute
29 Software only when embedded on a Microchip microcontroller or digital signal
30 controller that is integrated into your product or third party product
31 (pursuant to the sublicense terms in the accompanying license agreement).
32
33 You should refer to the license agreement accompanying this Software for
34 additional information regarding your rights and obligations.
35
36 SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,
37 EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF
38 MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.
39 IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER
40 CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR
41 OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES
42 INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR
43 CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF
44 SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES
45 (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.
46 ****
47 // DOM-IGNORE-END
48
49
50 // ****
51 // ****
52 // Section: Included Files
53 // ****
54 // ****
55
56 #include "app.h"
57 #include "Mc32_spi_sm.h"
58 #include "lcd_spi.h"
59 #include "system/devcon/src/sys_devcon_local.h"
60 #include "pec12.h"
61 #include "menu.h"
62
63
64
65 // ****
66 // ****
67 // Section: Global Data Definitions
68 // ****
69 // ****

```

```

70 // **** Application Data ****
71 /* Application Data
72
73     Summary:
74         Holds application data
75
76     Description:
77         This structure holds the application's data.
78
79     Remarks:
80         This structure should be initialized by the APP_Initialize function.
81
82         Application strings and buffers are be defined outside this structure.
83 */
84
85 APP_DATA appData;
86 SW S1;
87
88 // **** Application Callback Functions ****
89 // **** Application Local Functions ****
90 // **** Application Initialization and State Machine Functions ****
91
92 // **** Section: Application Callback Functions ****
93 // **** Section: Application Local Functions ****
94 // **** Section: Application Initialization and State Machine Functions ****
95
96 /* TODO: Add any necessary callback functions.
97 */
98
99 // **** Section: Application Local Functions ****
100 // **** Section: Application Initialization and State Machine Functions ****
101 // **** Section: Application Initialization and State Machine Functions ****
102 // **** Section: Application Initialization and State Machine Functions ****
103
104
105 /* TODO: Add any necessary local functions.
106 */
107
108
109 // **** Application Initialization and State Machine Functions ****
110 // **** Application Initialization and State Machine Functions ****
111 // **** Section: Application Initialization and State Machine Functions ****
112 // **** Application Initialization and State Machine Functions ****
113 // **** Application Initialization and State Machine Functions ****
114
115
116 //-----//  

APP_UpdateAppState  

117 void APP_UpdateAppState(APP_STATES newState){  

118
119     appData.appState = newState;  

120 }
121
122 //-----//  

APP_Initialize  

123 void APP_Initialize ( void )  

124 {  

125     SPI_Init();  

126     /* Place the App state machine in its initial state. */  

127     appData.appState = APP_STATE_INIT;  

128     appData.msCounter = 0;  

129     appData.backLightIntensity = 2500; /* 100% */  

130     appData.lightIntensity = 2500; /* 100% */  

131     appData.exposureDuration = 100;  

132     appData.timeBetweenPictures = 1000;  

133     appData.isFiveShotsSeqEnable = false;  

134     appData.seqClock1_ms = 0;  

135     appData.angleBwEachSeq = 10;  

136     appData.nbrOfShotsPerformed = 0;

```

```

137     appData.buzzerIntensity = 2500;
138     appData.valSeq = 0;
139
140     initMenuParam();
141     initStepperParam();
142 }
143
144
145
146 //-----//  

APP_Tasks  

147 void APP_Tasks ( void ){  

148
149     static uint16_t counter1 = 0;
150     static uint16_t counter2 = 0;
151
152     /* Main state machine */
153     switch(appData.appState){  

154         //-----//  

155         APP_STATE_INIT
156         case APP_STATE_INIT:
157             /* Read data from EEPROM to restore presets */
158             readDataFromEeprom(getMyStepperStruct());
159             /* Initialization of the motor */
160             initStepperMotor();
161             /* Update MCPWM Duty-cycle of other PWM with EEPROM data */
162             updateMcpwmDuty();
163             /* Turn on MCPWM */
164             PLIB_MCPWM_Enable(MCPWM_ID_0);
165             /* Initialization sequence */
166             initLcd();
167             /* Print initialization menu */
168             printLcdInit();
169             /* Start useful Timers */
170             DRV_TMR1_Start();
171             DRV_TMR2_Start();
172             /* Print main menu once all peripherals are configured */
173             printMainMenu();
174             /* States machines update */
175             APP_UpdateAppState(APP_STATE_WAIT);
176             break;
177
178         //-----//  

179         APP_STATE_SERVICE_TASKS
180         /* Frequency = 10'000Hz */
181         case APP_STATE_SERVICE_TASKS:  

182
183             /* Process who is responsible of the sequence, motor orders and
184             * lights orders. */
185
186             SIGN_LED_CMDToggle();
187             sequenceManagementProcess();
188
189             if(counter2 >= 10){
190                 /* Frequency = 1'000Hz */
191                 counter2 = 0;
192                 /* Scan the activity of the rotary encoder */
193                 scanPec12();
194                 /* Scan the activity of the switch S1 */
195                 scanSwitch();
196             }
197             if(counter1 >= 1000){
198                 /* Frequency = 10Hz */
199                 counter1 = 0;
200                 menuManagementProcess();
201             }
202             counter1++;
203             counter2++;

```

```

203
204     // Calls the SPI do task state machine
205     SPI_DoTasks();
206
207     /* States machines update */
208     APP_UpdateAppState(APP_STATE_WAIT);
209     break;
210
211     //-----// APP_STATE_WAIT
212     case APP_STATE_WAIT:
213         /* Nothing is supposed to happen here */
214         break;
215
216     //-----// default
217     default:
218         break;
219     }
220
221
222 //-----//
223 APP_Delay_ms
224 void APP_Delay_ms(uint32_t ms){
225
226     DRV_TMR3_Start();
227     // SIGN_LED_CMDToggle();
228     while(appData.msCounter < ms){
229
230         // SIGN_LED_CMDToggle();
231
232         DRV_TMR3_Stop();
233         appData.msCounter = 0;
234     }
235
236 //-----//
237 setBlIntensity
238 void setBlIntensity(int32_t *backLightIntensity){
239
240     // Limit values to avoid problems
241     if(*backLightIntensity < BACKLIGHT_INTENSITY_MIN) *backLightIntensity
242         = BACKLIGHT_INTENSITY_MIN;
243     if(*backLightIntensity > BACKLIGHT_INTENSITY_MAX) *backLightIntensity
244         = BACKLIGHT_INTENSITY_MAX;
245
246     /* 25 = 2500 / 100 */
247     appData.backLightIntensity = *backLightIntensity * 25;
248     PLIB_MCPWM_ChannelPrimaryDutyCycleSet(MCPWM_ID_0, PWM_BL_CH,
249                                         appData.backLightIntensity);
250
251     int32_t getBlIntensity(void){
252
253         return appData.backLightIntensity / 25;
254     }
255
256 //-----//
257 scanSwitch
258 void scanSwitch(void){
259
260     // Save old states for debounce
261     S1.state[3] = S1.state[2];
262     S1.state[2] = S1.state[1];
263     S1.state[1] = S1.state[0];
264     S1.state[0] = SWITCHStateGet();
265
266     // Check if switch is pressed
267     if(S1.state[0] == 1 && S1.state[1] == 1
268         && S1.state[2] == 0 && S1.state[3] == 0){

```

```

268         S1.isPressed = true;
269     }
270 }
//-----
272 getSwitchEvent
273 bool getSwitchEvent(void){
274
275     bool isPressed = S1.isPressed;
276     S1.isPressed = 0;
277
278     return isPressed;
279 }
//-----
281 initLcdSeq
282 void initLcd(void){
283
284     RESET_LCD_CMDOff();
285     APP_Delay_ms(1);
286     RESET_LCD_CMDOOn();
287     APP_Delay_ms(10);
288     initDisp1();
289     /* Create degree symbol for LCD uses */
290     CreateLcdDegreeSymbol(0x01);
291 }
292
293 void updateMcpwmDuty(void){
294
295     /* Update PWMs DutyCycle with data from EEPROM */
296     PLIB_MCPWM_ChannelPrimaryDutyCycleSet(MCPWM_ID_0, PWM_BL_CH,
297                                             appData.backLightIntensity);
298     PLIB_MCPWM_ChannelPrimaryDutyCycleSet(MCPWM_ID_0, PWM_BUZZER_CH,
299                                             appData.buzzerIntensity);
300     PLIB_MCPWM_ChannelPrimaryDutyCycleSet(MCPWM_ID_0, PWM_DIM_CH,
301                                             appData.lightIntensity);
302 }
303
304
305
306
307
308 ****
309 End of File
310 */
311

```

```

1  ****
2  MPLAB Harmony Application Header File
3
4  Company:
5      Microchip Technology Inc.
6
7  File Name:
8      app.h
9
10 Summary:
11     This header file provides prototypes and definitions for the application.
12
13 Description:
14     This header file provides function prototypes and data type definitions for
15     the application. Some of these are required by the system (such as the
16     "APP_Initialize" and "APP_Tasks" prototypes) and some of them are only used
17     internally by the application (such as the "APP_STATES" definition). Both
18     are defined here for convenience.
19 ****
20
21 //DOM-IGNORE-BEGIN
22 ****
23 Copyright (c) 2013-2014 released Microchip Technology Inc. All rights reserved.
24
25 Microchip licenses to you the right to use, modify, copy and distribute
26 Software only when embedded on a Microchip microcontroller or digital signal
27 controller that is integrated into your product or third party product
28 (pursuant to the sublicense terms in the accompanying license agreement).
29
30 You should refer to the license agreement accompanying this Software for
31 additional information regarding your rights and obligations.
32
33 SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,
34 EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF
35 MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.
36 IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER
37 CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR
38 OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES
39 INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR
40 CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF
41 SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES
42 (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.
43 ****
44 //DOM-IGNORE-END
45
46 #ifndef _APP_H
47 #define _APP_H
48
49 // ****
50 // ****
51 // Section: Included Files
52 // ****
53 // ****
54
55 #include "stepperDriver.h"
56 #include "lights.h"
57 #include <stdint.h>
58 #include <stdbool.h>
59 #include <stddef.h>
60 #include <stdlib.h>
61 #include <stdio.h>
62 #include "system_config.h"
63 #include "system_definitions.h"
64
65 // DOM-IGNORE-BEGIN
66 #ifdef __cplusplus // Provide C++ Compatibility
67
68 extern "C" {
69

```

```

70 #endif
71 // DOM-IGNORE-END
72
73 // ****
74 // Section: Type Definitions
75 // ****
76 // ****
77 // ****
78
79 #define SYS_CLK 40000000
80
81 #define PWM_A_CMD_CH MCPWM_CHANNEL1
82 #define PWM_B_CMD_CH MCPWM_CHANNEL1
83 #define PWM_C_CMD_CH MCPWM_CHANNEL2
84 #define PWM_D_CMD_CH MCPWM_CHANNEL2
85 #define PWM_BL_CH MCPWM_CHANNEL3
86 #define PWM_BUZZER_CH MCPWM_CHANNEL4
87 #define PWM_DIM_CH MCPWM_CHANNEL6
88
89 #define MARGIN_LED_DELAY 50
90
91 /* Intensity in percent */
92 #define BACKLIGHT_INTENSITY_MIN 0
93 #define BACKLIGHT_INTENSITY_MAX 100
94
95
96 /* Value used to check if the EEPROM is already writtent by this code */
97 #define CONTROL_VALUE 0x11223344
98
99 // ****
100 /* Application states
101
102 Summary:
103 Application states enumeration
104
105 Description:
106 This enumeration defines the valid application states. These states
107 determine the behavior of the application at various times.
108 */
109
110 typedef enum
111 {
112     /* Application's state machine's initial state. */
113     APP_STATE_INIT=0,
114     APP_STATE_SERVICE_TASKS,
115     APP_STATE_SERVICE_CAPTURE,
116     APP_STATE_WAIT,
117 } APP_STATES;
118
119 typedef enum{
120     SYS_STATE_MENU = 0,
121     SYS_STATE_MANUAL,
122     SYS_STATE_AUTO
123 } SYSTEM_STATES;
124
125
126
127 // ****
128 /* Application Data
129
130 Summary:
131 Holds application data
132
133 Description:
134 This structure holds the application's data.
135
136 Remarks:
137 Application strings and buffers are be defined outside this structure.
138

```

```

139     */
140
141 typedef enum{
142
143     ALL_LED_DISABLE = 0,
144     PWR_LED1,
145     PWR_LED2,
146     PWR_LED3,
147     PWR_LED4,
148     PWR_LED5,
149
150 }LED_ID;
151
152
153 typedef struct
154 {
155     /* The application's current state */
156     APP_STATES appState;
157     SYSTEM_STATES systemState;
158     LED_ID ledId;
159     uint32_t msCounter;
160
161     /* LED config */
162     uint16_t lightIntensity;
163     uint16_t timeBetweenPictures;
164     uint16_t exposureDuration;
165
166     /* Auto mode param */
167     uint8_t angleBwEachSeq;
168
169     uint32_t seqClock1_ms;
170     uint32_t seqClock2_ms;
171     bool isFiveShotsSeqEnable;
172     bool isFullImaginSeqEnable;
173     bool isFirstPass;
174     uint16_t nbrOfShotsPerformed;
175     uint8_t valSeq;
176
177     uint16_t backLightIntensitiy;
178
179     uint16_t buzzerIntensity;
180
181 } APP_DATA;
182
183 typedef struct
184 {
185     bool state[4];
186     bool isPressed;
187
188 } SW;
189
190 typedef struct{
191
192     /* Motor data */
193     int16_t stepPerSec;
194     uint16_t stepPerTurn;
195     uint16_t gearValue;
196     float anglePerStep;
197
198     /* LEDs data */
199     uint16_t lightIntensity;
200     uint16_t timeBetweenPictures;
201     uint16_t exposureDuration;
202
203     uint16_t backLightIntensitiy;
204
205     /* Security value */
206     uint32_t controlValue;
207

```

```

208     } DATA_IN_EEPROM;
209 // ****
210 // Section: Application Callback Routines
211 // ****
212 // ****
213 // ****
214 /* These routines are called by drivers when certain events occur.
215 */
216 // ****
217 // ****
218 // Section: Application Initialization and State Machine Functions
219 // ****
220 // ****
221 // ****
222 // ****
223 // ****
224 Function:
225     void APP_Initialize ( void )
226
227 Summary:
228     MPLAB Harmony application initialization routine.
229
230 Description:
231     This function initializes the Harmony application. It places the
232     application in its initial state and prepares it to run so that its
233     APP_Tasks function can be called.
234
235 Precondition:
236     All other system initialization routines should be called before calling
237     this routine (in "SYS_Initialize").
238
239 Parameters:
240     None.
241
242 Returns:
243     None.
244
245 Example:
246     <code>
247         APP_Initialize();
248     </code>
249
250 Remarks:
251     This routine must be called from the SYS_Initialize function.
252 */
253
254 void APP_Initialize ( void );
255
256 // ****
257 Function:
258     void APP_Tasks ( void )
259
260 Summary:
261     MPLAB Harmony Demo application tasks function
262
263 Description:
264     This routine is the Harmony Demo application's tasks function. It
265     defines the application's state machine and core logic.
266
267 Precondition:
268     The system and application initialization ("SYS_Initialize") should be
269     called before calling this.
270
271 Parameters:
272     None.
273
274 Returns:
275     None.
276

```

```
277
278     Example:
279     <code>
280     APP_Tasks();
281     </code>
282
283     Remarks:
284     This routine must be called from SYS_Tasks() routine.
285 */
286
287     void APP_Tasks( void );
288     void APP_Delay_ms(uint32_t ms);
289
290     void setBlIntensity(int32_t *backLightIntensitiy);
291     int32_t getBlIntensity(void);
292
293     void scanSwitch(void);
294     bool getSwitchEvent(void);
295
296     void initLcd(void);
297
298     void updateMcpwmDuty(void);
300
301
302 #endif /* _APP_H */
304 //DOM-IGNORE-BEGIN
305 #ifdef __cplusplus
306 }
308#endif
309//DOM-IGNORE-END
310
311 ****
312 End of File
313 */
314
315
```

```

1  /*
2   * File: stepperDriver.c
3   * Author: ricch
4   *
5   * Created on August 30, 2023, 10:39 PM
6   *
7   * DRV8432 driver 2H bridge
8   */
9
10
11 #include <stepperDriver.h>
12
13 static STEPPER_DATA stepperData;
14 extern APP_DATA appData;
15
16 //-----
17 //----- initStepperData
18 void initStepperParam(void) {
19
20     stepperData.isAtHomeInCW      = false;
21     stepperData.isAtHomeInCCW    = false;
22     stepperData.isIndexed        = false;
23     stepperData.isInAutoHomeSeq = false;
24
25     stepperData.performedSteps  = 0;
26     stepperData.stepToReach     = 0;
27
28     stepperData.stepPerSec       = 1000;
29
30     stepperData.stepPerTurn     = 200;
31     stepperData.gearValue       = 200;
32
33     stepperData.anglePerStep    = 1.8;
34
35     stepperData.dutyCycleStepper = 30;
36 }
37
38 void initStepperMotor() {
39
40     //setStepperPower(&stepperData, &stepperData.dutyCycleStepper);
41
42     /* Disable RESET on both H bridge */
43     RESET_AB_CMDOn();
44     RESET_CD_CMDOn();
45 }
46
47 //-----
48 //----- turnOffStepperPwms
49 /* Disable all PWMs for motor control */
50 void turnOffStepperPwms(void) {
51
52     /* A */
53     PLIB_MCPWM_ChannelPWMxHENable (MCPWM_ID_0 , MCPWM_CHANNEL1);
54     /* B */
55     PLIB_MCPWM_ChannelPWMxHENable (MCPWM_ID_0 , MCPWM_CHANNEL2);
56     /* A */
57     PLIB_MCPWM_ChannelPWMMxLEnable (MCPWM_ID_0 , MCPWM_CHANNEL1);
58     /* B */
59     PLIB_MCPWM_ChannelPWMMxLEnable (MCPWM_ID_0 , MCPWM_CHANNEL2);
60 }
61
62 //-----
63 //----- changeSpeed
64 void changeSpeed(STEPPER_DATA *pStepperData) {
65
66     uint16_t tmrPeriod = 0;
67     uint16_t frequency = 0;
68     //uint16_t presc = 0;

```

```

67     frequency = pStepperData->stepPerSec;
68     //presc = TMR_PrescaleGet_Default(TMR_ID_3);
69     tmrPerdiode = SYS_CLK / (frequency * 16) - 1;
70     PLIB_TMR_Counter16BitClear(TMR_ID_3);
71     PLIB_TMR_Period16BitSet(TMR_ID_3, tmrPerdiode);
72 }
73 }
74 //-----
75 //-----processStepper
76 void processStepper(STEPPER_DATA *pStepperData){
77     static uint8_t step = 0;
78     //----- Counter clockwise CCW
79     if(pStepperData->performedSteps > pStepperData->stepToReach){
80         if(pStepperData->isAtHomeInCCW == false){
81             switch(step){
82                 /* Sequence of 4 steps for CCW rotation */
83                 case 1:
84                     /* A */
85                     PLIB_MCPWM_ChannelPWMxHEnable (MCPWM_ID_0 ,MCPWM_CHANNEL1);
86                     /* B */
87                     PLIB_MCPWM_ChannelPWMxHDisable(MCPWM_ID_0 ,MCPWM_CHANNEL2);
88                     /* A */
89                     PLIB_MCPWM_ChannelPWMxLDisable(MCPWM_ID_0 ,MCPWM_CHANNEL1);
90                     /* B */
91                     PLIB_MCPWM_ChannelPWMxLEnable (MCPWM_ID_0 ,MCPWM_CHANNEL2);
92                     break;
93
94                 case 2:
95                     PLIB_MCPWM_ChannelPWMxHEnable (MCPWM_ID_0 ,MCPWM_CHANNEL1);
96                     PLIB_MCPWM_ChannelPWMxHEnable (MCPWM_ID_0 ,MCPWM_CHANNEL2);
97                     PLIB_MCPWM_ChannelPWMxLDisable(MCPWM_ID_0 ,MCPWM_CHANNEL1);
98                     PLIB_MCPWM_ChannelPWMxLDisable(MCPWM_ID_0 ,MCPWM_CHANNEL2);
99                     break;
100
101                 case 3:
102                     PLIB_MCPWM_ChannelPWMxHDisable(MCPWM_ID_0 ,MCPWM_CHANNEL1);
103                     PLIB_MCPWM_ChannelPWMxHEnable (MCPWM_ID_0 ,MCPWM_CHANNEL2);
104                     PLIB_MCPWM_ChannelPWMxLEnable (MCPWM_ID_0 ,MCPWM_CHANNEL1);
105                     PLIB_MCPWM_ChannelPWMxLDisable(MCPWM_ID_0 ,MCPWM_CHANNEL2);
106                     break;
107
108                 case 0:
109                     PLIB_MCPWM_ChannelPWMxHDisable(MCPWM_ID_0 ,MCPWM_CHANNEL1);
110                     PLIB_MCPWM_ChannelPWMxHDisable(MCPWM_ID_0 ,MCPWM_CHANNEL2);
111                     PLIB_MCPWM_ChannelPWMxLEnable (MCPWM_ID_0 ,MCPWM_CHANNEL1);
112                     PLIB_MCPWM_ChannelPWMxLEnable (MCPWM_ID_0 ,MCPWM_CHANNEL2);
113                     break;
114             }
115             step++;
116         }
117     }/* Four steps performed in CCW */
118     if(step == 4){
119
120         step = 0;
121         pStepperData->performedSteps -= 4;
122     }
123     /* Index is reach in CCW */
124     if(INDEXStateGet() && pStepperData->isAtHomeInCW == false){
125
126         pStepperData->isAtHomeInCCW = true;
127         pStepperData->stepToDoReach = pStepperData->performedStep;
128
129         if(pStepperData->isInAutoHomeSeq == true){
130
131             pStepperData->stepToReach = 0;
132             pStepperData->performedSteps = 0;
133             pStepperData->isIndexed = true;
134

```

```

135         pStepperData->isInAutoHomeSeq = false;
136     }
137   }
138   else pStepperData->isAtHomeInCCW = false;
139 }
//-----// Clockwise CW
140 else if(pStepperData->performedSteps < pStepperData->stepToReach) {
141   if(pStepperData->isAtHomeInCW == false){
142     switch(step){
143       /* Sequence of 4 steps for CW rotation */
144       case 1:
145         /* A */
146         PLIB_MCPWM_ChannelPWMxHEnable (MCPWM_ID_0 ,MCPWM_CHANNEL1);
147         /* B */
148         PLIB_MCPWM_ChannelPWMxHDisable(MCPWM_ID_0 ,MCPWM_CHANNEL2);
149         /* A */
150         PLIB_MCPWM_ChannelPWMxLDisable(MCPWM_ID_0 ,MCPWM_CHANNEL1);
151         /* B */
152         PLIB_MCPWM_ChannelPWMxLEnable (MCPWM_ID_0 ,MCPWM_CHANNEL2);
153         break;
154
155       case 0:
156         PLIB_MCPWM_ChannelPWMxHEnable (MCPWM_ID_0 ,MCPWM_CHANNEL1);
157         PLIB_MCPWM_Channel1PWMxHEnable (MCPWM_ID_0 ,MCPWM_CHANNEL2);
158         PLIB_MCPWM_Channel1PWMxLDisable(MCPWM_ID_0 ,MCPWM_CHANNEL1);
159         PLIB_MCPWM_Channel1PWMxLDisable(MCPWM_ID_0 ,MCPWM_CHANNEL2);
160         break;
161
162       case 3:
163         PLIB_MCPWM_Channel1PWMxHDisable(MCPWM_ID_0 ,MCPWM_CHANNEL1);
164         PLIB_MCPWM_Channel1PWMxHEnable (MCPWM_ID_0 ,MCPWM_CHANNEL2);
165         PLIB_MCPWM_Channel1PWMxLEnable (MCPWM_ID_0 ,MCPWM_CHANNEL1);
166         PLIB_MCPWM_Channel1PWMxLDisable(MCPWM_ID_0 ,MCPWM_CHANNEL2);
167         break;
168
169       case 2:
170         PLIB_MCPWM_Channel1PWMxHDisable(MCPWM_ID_0 ,MCPWM_CHANNEL1);
171         PLIB_MCPWM_Channel1PWMxHDisable(MCPWM_ID_0 ,MCPWM_CHANNEL2);
172         PLIB_MCPWM_Channel1PWMxLEnable (MCPWM_ID_0 ,MCPWM_CHANNEL1);
173         PLIB_MCPWM_Channel1PWMxLEnable (MCPWM_ID_0 ,MCPWM_CHANNEL2);
174         break;
175     }
176     step++;
177   }
178   /* Four steps performed in CW */
179   if(step == 4){
180
181     step = 0;
182     pStepperData->performedSteps += 4;
183   }
184   /* Index is reach in CW */
185   if(INDEXStateGet() && pStepperData->isAtHomeInCCW == false){
186
187     pStepperData->isAtHomeInCW = true;
188     /* Stop the automatic sequence */
189     appData.isFullImaginSeqEnable = false;
190     /* Stop the motor */
191     pStepperData->stepToReach = pStepperData->performedSteps;
192   }
193   else pStepperData->isAtHomeInCW = false;
194 }
195
196
197 // The motor reach its desired position
198 // if(pStepperData->performedSteps == pStepperData->stepToReach) {
199 //   turnOffStepperPwms();
200 // } else {
201 //   PLIB_MCPWM_Enable (MCPWM_ID_0 );
202
203

```

```

204 //      }
205 }
206
207
208
209 //-----// setSpeed
210 void setSpeed(STEPPER_DATA *pStepperData, uint32_t *pStepPerSec){
211
212     // Limit values to avoid problems
213     if(*pStepPerSec < STEP_PER_SEC_MIN) *pStepPerSec = STEP_PER_SEC_MIN;
214     if(*pStepPerSec > STEP_PER_SEC_MAX) *pStepPerSec = STEP_PER_SEC_MAX;
215
216     // Save data
217     pStepperData->stepPerSec = *pStepPerSec;
218 }
219
220 int32_t getSpeed(STEPPER_DATA *pStepperData){
221
222     return pStepperData->stepPerSec;
223 }
224
225 //-----// setGearReduction
226 void setGearReduction(STEPPER_DATA *pStepperData, uint32_t *pGearValue){
227
228     // Limit values to avoid problems
229     if(*pGearValue < GEAR_VALUE_MIN) *pGearValue = GEAR_VALUE_MIN;
230     if(*pGearValue > GEAR_VALUE_MAX) *pGearValue = GEAR_VALUE_MAX;
231
232     // Save data
233     pStepperData->gearValue = *pGearValue;
234 }
235
236 //-----// getGearReduction
237 uint32_t getGearReduction(STEPPER_DATA *pStepperData){
238
239     return pStepperData->gearValue;
240 }
241
242 //-----// setAnglePerStep
243 void setAnglePerStep(STEPPER_DATA *pStepperData, uint32_t *pAnglePerStep){
244
245     float temp = (*pAnglePerStep / 10.0);
246
247     // Limit values to avoid problems
248     if(temp < ANGLE_PER_STEP_MIN) temp = (ANGLE_PER_STEP_MIN);
249     if(temp > ANGLE_PER_STEP_MAX) temp = (ANGLE_PER_STEP_MAX);
250     *pAnglePerStep = temp * 10;
251
252     // Save data
253     pStepperData->anglePerStep = temp;
254 }
255
256 //-----// getAnglePerStep
257 uint32_t getAnglePerStep(STEPPER_DATA *pStepperData){
258
259     // x10 ???
260     return pStepperData->anglePerStep * 10;
261 }
262
263 //-----// getPerformedSteps
264 int32_t getPerformedSteps(STEPPER_DATA *pStepperData){
265
266     return pStepperData->performedSteps / pStepperData->stepPerTurn;
267 }

```

```

268 //-----
269 void setRotationToDo(STEPPER_DATA *pStepperData, int32_t *pRotationToDo){
270
271     // Limit values to avoid problems
272     if(*pRotationToDo < ROTATION_TO_DO_MIN) *pRotationToDo = ROTATION_TO_DO_MIN;
273     if(*pRotationToDo > ROTATION_TO_DO_MAX) *pRotationToDo = ROTATION_TO_DO_MAX;
274
275     // Save data
276     pStepperData->stepToReach = *pRotationToDo * pStepperData->stepPerTurn;
277 }
278 //-----
279 int32_t getRotationToDo(STEPPER_DATA *pStepperData){
280
281     return pStepperData->stepToReach / pStepperData->stepPerTurn;
282 }
283
284 //-----
285 void startAutoHome(STEPPER_DATA *pStepperData){
286
287     pStepperData->isInAutoHomeSeq = true;
288     // Check if the arm is not at home
289     if(pStepperData->isAtHomeInCCW == false){
290         // Put steps to do for returning home in CCW
291         pStepperData->stepToReach = -50000; // DEFINE? STEP_TO_DO_MAX
292     }
293 }
294
295 //-----
296 void setStepperPower(STEPPER_DATA *pStepperData, uint16_t *pDutyCycleStepper){
297
298     uint16_t dutyValCh1 = 0;
299
300     // Limit values to avoid problems
301     if(*pDutyCycleStepper < MCPWM_DUTYCYCLE_MIN) *pDutyCycleStepper
302         = MCPWM_DUTYCYCLE_MIN;
303     if(*pDutyCycleStepper > MCPWM_DUTYCYCLE_MAX) *pDutyCycleStepper
304         = MCPWM_DUTYCYCLE_MAX;
305
306     /* Save configuration in the structure */
307     pStepperData->dutyCycleStepper = *pDutyCycleStepper;
308
309     /* Must be the inverse of the CHANNEL 1 */
310     dutyValCh1 = MCPWM_PRIMARY_PERIOD - *pDutyCycleStepper;
311
312     PLIB_MCPWM_ChannelPrimaryDutyCycleSet(MCPWM_ID_0, MCPWM_CHANNEL1,
313                                             dutyValCh1);
314     PLIB_MCPWM_ChannelPrimaryDutyCycleSet(MCPWM_ID_0, MCPWM_CHANNEL2,
315                                             *pDutyCycleStepper);
316 }
317
318 int16_t getStepperPower(STEPPER_DATA *pStepperData){
319
320     return pStepperData->dutyCycleStepper;
321 }
322
323
324 //-----
325 STEPPER_DATA* getMyStepperStruct(void){
326
327     /* Return the address of the structure */
328     return &stepperData;
329 }
```

```

1  /*
2   * File:    stepperDriver.h
3   * Author:  ricch
4   *
5   * Created on August 30, 2023, 10:39 PM
6   */
7
8 #ifndef STEPPERDRIVER_H
9 #define STEPPERDRIVER_H
10
11 #ifdef __cplusplus
12 extern "C" {
13#endif
14
15 #include <stdint.h>
16 #include "system_definitions.h"
17
18 // Defines
19 #define STEP_PER_SEC_MIN 40
20 #define STEP_PER_SEC_MAX 1000
21
22 #define GEAR_VALUE_MIN 1
23 #define GEAR_VALUE_MAX 1000
24
25 #define STEP_PER_TURN_MIN 4
26 #define STEP_PER_TURN_MAX 400
27
28 #define ANGLE_PER_STEP_MIN 0.1
29 #define ANGLE_PER_STEP_MAX 10.0
30
31 #define ROTATION_TO_DO_MIN -50000
32 #define ROTATION_TO_DO_MAX 50000
33
34 /* Period for 50kHz PWMs */
35 #define MCPWM_PRIMARY_PERIOD 199
36 #define MCPWM_DUTYCYCLE_MIN 9
37 #define MCPWM_DUTYCYCLE_MAX 189
38
39 // Structures
40 typedef struct{
41
42     /* Motion motor data */
43     bool      isAtHomeInCW;
44     bool      isAtHomeInCCW;
45     bool      isIndexed;
46     bool      isInAutoHomeSeq;
47
48     int32_t   performedSteps;
49     int32_t   stepToReach;
50
51     /* Motor characteristics */
52     int16_t   stepPerSec;
53
54     uint16_t  stepPerTurn;
55     uint16_t  gearValue;
56
57     float    anglePerStep;
58
59     uint16_t  dutyCycleStepper;
60
61 } STEPPER_DATA;
62
63
64
65 // Prototypes
66 void initStepperParam(void);
67 void turnOffStepperPwms(void);
68 void changeSpeed(STEPPER_DATA *pStepperData);
69 void processStepper(STEPPER_DATA *pStepperData);

```

```
70
71
72     void setSpeed(STEPPER_DATA *pStepperData, uint32_t *pStepPerSec);
73     int32_t getSpeed(STEPPER_DATA *pStepperData);
74
75     void setGearReduction(STEPPER_DATA *pStepperData, uint32_t *pGearValue);
76     uint32_t getGearReduction(STEPPER_DATA *pStepperData);
77
78     void setAnglePerStep(STEPPER_DATA *pStepperData, uint32_t *pAnglePerStep);
79     uint32_t getAnglePerStep(STEPPER_DATA *pStepperData);
80
81     void setRotationToDo(STEPPER_DATA *pStepperData, int32_t *pRotationToDo);
82     int32_t getRotationToDo(STEPPER_DATA *pStepperData);
83     void startAutoHome(STEPPER_DATA *pStepperData);
84
85     STEPPER_DATA* getMyStepperStruct(void);
86
87 #ifdef __cplusplus
88 }
89#endif
90
91#endif /* STEPPERDRIVER_H */
92
93
94
```

```
1  /*
2   * File: menu.c
3   * Author: ricch
4   *
5   * Created on August 30, 2023, 10:15 AM
6   */
7
8 #include "menu.h"
9 #include "lcd_spi.h"
10 #include "stdio.h"
11 #include "pec12.h"
12 #include "stepperDriver.h"
13
14
15 MENU menu;
16 extern APP_DATA appData;
17 bool isInModifMode = 0;
18 bool isFirstDataProcessPass = false;
19
20
21 void initMenuParam() {
22
23     menu.menuPage = 0;
24     menu.menuSize = 2;
25     menu.menuState = MAIN_MENU;
26 }
27
28
29 //_
// menuManagementProcess
30 void menuManagementProcess(void){
31
32     static int32_t pec12RotationValue = 0;
33
34     /* Get PEC12 increments or decrements if there are */
35     int incrOrDecr = getPec12IncrOrDecr();
36     //-----
37     isInModifMode == true
38     if(isInModifMode){
39
40         pec12RotationValue += incrOrDecr;
41         menuDataProcess(&pec12RotationValue, getMyStepperStruct());
42         menuPrintProcess(getMyStepperStruct());
43
44         /* PEC12 switch pressed */
45         if(getPec12SwitchEvent()){
46
47             /* Leave modification mode */
48             isInModifMode = false;
49             /* Put the cursor on the first line */
50             pec12RotationValue = 0;
51         }
52     }
53     //-----
54     isInModifMode == false
55     else if(isInModifMode == false){
56
57         pec12RotationValue += incrOrDecr;
58
59         if(pec12RotationValue > menu.menuSize) pec12RotationValue =
60             menu.menuSize;
61         else if(pec12RotationValue < 0) pec12RotationValue = 0;
62
63         if(pec12RotationValue <= 3){
64
65             menu.menuPage = 0;
66             menuPrintProcess(getMyStepperStruct());
67             printCursor(pec12RotationValue);
68         }
69     }
70 }
```

```

67         else{
68             menu.menuPage = 1;
69             menuPrintProcess(getMyStepperStruct());
70             printCursor(pec12RotationValue - 4);
71         }
72     }
73 
74     /* PEC12 switch pressed */
75     if(getPec12SwitchEvent()){
76 
77         menuActionProcess(pec12RotationValue);
78         menuDataProcess(&pec12RotationValue, getMyStepperStruct());
79         menuPrintProcess(getMyStepperStruct());
80 
81         /* Put the cursor on the first line */
82         pec12RotationValue = 0;
83     }
84 }
85 if(getSwitchEvent()){
86 
87     //startFiveShotsSequence();
88     startFullImagingSequence();
89 
90     switch (menu.menuState){
91 
92         case MANUAL_MODE_MENU:
93             /* Start a sequence of 5 pictures */
94             //startFiveShotsSequence();
95             startFiveShotsSeqProcess();
96             break;
97 
98         default:
99             break;
100    }
101 }
102 }
103 
104 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
105 void menuActionProcess(int32_t pec12RotationValue){
106 
107     /* Menu action switch */
108     if(isInModifMode == false){
109         switch(menu.menuState){
110 
111             //-----// Main
112             menu
113             case MAIN_MENU:
114 
115                 switch(pec12RotationValue){
116 
117                     case CAPTURE_MODE_SEL:
118                         menu.menuState = CAPTURE_MODE_MENU;
119                         menu.menuSize = 2;
120                         break;
121 
122                     case SETTINGS_SEL:
123                         menu.menuState = SETTINGS_MENU;
124                         menu.menuSize = 5;
125                         break;
126 
127                     case ABOUT_SEL:
128                         menu.menuState = ABOUT_MENU;
129                         menu.menuSize = 0;
130                         break;
131                 }
132                 break;
133 
134             //-----// Main

```

```

135     menu -> Choice_menu
136
137     case CAPTURE_MODE_MENU:
138
139         switch(pec12RotationValue){
140
141             case RETURN_SEL:
142                 menu.menuState = MAIN_MENU;
143                 menu.menuSize = 2;
144                 break;
145
146             case MANUAL_MODE_SEL:
147                 menu.menuState = MANUAL_MODE_MENU;
148                 menu.menuSize = 2;
149                 break;
150
151             case AUTOMATIC_MODE_SEL:
152                 menu.menuState = AUTOMATIC_MODE_MENU;
153                 menu.menuSize = 1;
154                 break;
155
156         }
157         break;
158
159 //-----// Main
160 menu -> Choice_menu -> Manual Mode
161 case MANUAL_MODE_MENU:
162     switch(pec12RotationValue){
163
164         case RETURN_SEL:
165             menu.menuState = CAPTURE_MODE_MENU;
166             menu.menuSize = 2;
167             break;
168
169         case AUTO_HOME_SEL:
170             menu.menuState = AUTO_HOME_MENU;
171             menu.menuSize = 1;
172             break;
173
174         case ANGLE_SEL:
175             menu.modifState = ANGLE_MODIF;
176             isInModifMode = true;
177             isFirstDataProcessPass = true;
178             break;
179
180     }
181     break;
182
183 //-----// Main
184 menu -> Auto menu
185 case AUTOMATIC_MODE_MENU:
186     switch(pec12RotationValue){
187
188         case RETURN_SEL:
189             menu.menuState = CAPTURE_MODE_MENU;
190             menu.menuSize = 2;
191             break;
192
193         case AUTOMATIC_MODE_START_SEL:
194             menu.modifState = AUTOMATIC_MODE_START;
195             isInModifMode = true;
196             isFirstDataProcessPass = true;
197             break;
198
199     }
200     break;
201
202 //-----// Main
203 menu -> Manual menu -> Auto home
204 case AUTO_HOME_MENU:
205     switch(pec12RotationValue){
206
207         case RETURN_SEL:

```

```

200             menu.menuState = MANUAL_MODE_MENU;
201             menu.menuSize = 2;
202             break;
203
204         case AUTO_HOME_START_SEL:
205             menu.modifState = AUTO_HOME_START;
206             isInModifMode = true;
207             isFirstDataProcessPass = true;
208             break;
209         }
210     break;
211
212 //returnToHome(); PEUT ETRE METTRE AILLEUR
213 break;
214
215 //-----// Main
216 menu -> Settings menu
217 case SETTINGS_MENU:
218
219     switch(pec12RotationValue){
220
221         case RETURN_SEL:
222             menu.menuState = MAIN_MENU;
223             menu.menuSize = 2;
224             break;
225
226         case MOTOR_SEL:
227             menu.menuState = MOTOR_MENU;
228             menu.menuSize = 4;
229             break;
230
231         case LEDS_SEL:
232             menu.menuState = LIGHT_MENU;
233             menu.menuSize = 1;
234             break;
235
236         case BACKLIGHT_SEL:
237             menu.menuState = BACKLIGHT_MENU;
238             menu.menuSize = 1;
239             break;
240
241         case CAMERA_SEL:
242             menu.menuState = CAMERA_MENU;
243             menu.menuSize = 3;
244             break;
245
246         case SAVE_DATA_SEL:
247             menu.menuState = SAVE_DATA_MENU;
248             menu.menuSize = 1;
249             break;
250     }
251     break;
252
253 //-----// Main
254 menu -> Settings menu -> Motor menu
255 case MOTOR_MENU:
256     switch(pec12RotationValue){
257
258         case RETURN_SEL:
259             menu.menuState = SETTINGS_MENU;
260             menu.menuSize = 5;
261             break;
262
263         case SPEED_SEL:
264             menu.modifState = SPEED_MODIF;
265             isInModifMode = true;
266             isFirstDataProcessPass = true;
267             break;

```

```

267
268     case GEAR_SEL:
269         menu.modifState = GEAR_MODIF;
270         isInModifMode = true;
271         isFirstDataProcessPass = true;
272         break;
273
274     case STEP_PER_TURN_SEL:
275         menu.modifState = STEP_PER_TURN_MODIF;
276         isInModifMode = true;
277         isFirstDataProcessPass = true;
278         break;
279
280     case POWER_SEL:
281         menu.modifState = POWER_MODIF;
282         isInModifMode = true;
283         isFirstDataProcessPass = true;
284         break;
285     }
286     break;
287
288 //-----// Main
289 menu -> Settings menu -> Light menu
290 case LIGHT_MENU:
291     switch(pec12RotationValue){
292
293         case RETURN_SEL:
294             menu.menuState = SETTINGS_MENU;
295             menu.menuSize = 5;
296             break;
297
298         case LIGHT_INTENSITY_SEL:
299             menu.modifState = LIGHT_INTENSITY_MODIF;
300             isInModifMode = true;
301             isFirstDataProcessPass = true;
302             break;
303
304         // // // // // in camera param
305         case LIGHT_TIME_SEL:
306             menu.modifState = LIGHT_TIME_MODIF;
307             isInModifMode = true;
308             isFirstDataProcessPass = true;
309             break;
310
311 //-----// Main
312 menu -> Settings menu -> Back-light menu
313 case BACKLIGHT_MENU:
314     switch(pec12RotationValue){
315
316         case RETURN_SEL:
317             menu.menuState = SETTINGS_MENU;
318             menu.menuSize = 5;
319             break;
320
321         case LIGHT_INTENSITY_SEL:
322             menu.modifState = BL_INTENSITY_MODIF;
323             isInModifMode = true;
324             isFirstDataProcessPass = true;
325             break;
326
327
328 //-----// Main
329 menu -> Settings menu -> Camera
330 case CAMERA_MENU:
331     switch(pec12RotationValue){
332
333         case RETURN_SEL:

```

```

333             menu.menuState = SETTINGS_MENU;
334             menu.menuSize = 5;
335             break;
336
337         case EXPOSURE_TIME_SEL:
338             menu.modifState = EXPOSURE_TIME_MODIF;
339             isInModifMode = true;
340             break;
341
342         case TIME_BW_PICTURES_SEL:
343             menu.modifState = TIME_BW_PICTURES_MODIF;
344             isInModifMode = true;
345             break;
346     }
347     isFirstDataProcessPass = true;
348     break;
349
350 //-----// Main
351 menu -> Settings menu -> Save data
352 case SAVE_DATA_MENU:
353     switch(pec12RotationValue){
354
355         case RETURN_SEL:
356             menu.menuState = SETTINGS_MENU;
357             menu.menuSize = 5;
358             break;
359
360         case SAVE_DATA_SEL - 4:
361             menu.modifState = SAVE_DATA_START;
362             isInModifMode = true;
363             break;
364     }
365     isFirstDataProcessPass = true;
366     break;
367
368 //-----// Main
369 menu -> About menu
370 case ABOUT_MENU:
371     switch(pec12RotationValue){
372
373         case RETURN_SEL:
374             menu.menuState = MAIN_MENU;
375             menu.menuSize = 2;
376             break;
377     }
378     break;
379
380     default:
381         break;
382     }
383 }
384
385 //////////////////////////////////////////////////////////////////
386 void menuDataProcess(int32_t *pec12RotationValue, STEPPER_DATA *pStepperData) {
387
388     /* Data action switch */
389     if(isInModifMode){
390         switch(menu.modifState){
391
392             //-----//
393             case ANGLE_MODIF:
394                 if(isFirstDataProcessPass){
395
396                     isFirstDataProcessPass = false;
397                     /* A TESTER ET VALIDER, PERTE DE PAS POSSIBLE */
398                     *pec12RotationValue = getRotationToDo(pStepperData);

```

```

399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
}
setRotationToDo(pStepperData, pec12RotationValue);
break;
//-----
SPEED_MODIF
case SPEED_MODIF:
if(isFirstDataProcessPass){
    isFirstDataProcessPass = false;
    *pec12RotationValue = getSpeed(pStepperData);
}
setSpeed(pStepperData, pec12RotationValue);
break;
//-----
GEAR_MODIF
case GEAR_MODIF:
if(isFirstDataProcessPass){
    isFirstDataProcessPass = false;
    *pec12RotationValue = getGearReduction(pStepperData);
}
setGearReduction(pStepperData, pec12RotationValue);
break;
//-----
STEP_PER_TURN_MODIF
case STEP_PER_TURN_MODIF :
if(isFirstDataProcessPass){
    isFirstDataProcessPass = false;
    *pec12RotationValue = getAnglePerStep(pStepperData);
}
setAnglePerStep(pStepperData, pec12RotationValue);
break;
//-----
POWER_MODIF
case POWER_MODIF:
if(isFirstDataProcessPass){
    isFirstDataProcessPass = false;
    *pec12RotationValue = getStepperPower(pStepperData);
}
setStepperPower(pStepperData, (uint16_t*)pec12RotationValue); /**
????_____Dwaf-ad-*** */
break;
//-----
BL_INTENSITY_MODIF
case BL_INTENSITY_MODIF :
if(isFirstDataProcessPass){
    isFirstDataProcessPass = false;
    *pec12RotationValue = getBlIntensity();
}
setBlIntensity(pec12RotationValue);
break;
//-----
LIGHT_INTENSITY_MODIF
case LIGHT_INTENSITY_MODIF:
if(isFirstDataProcessPass){
    isFirstDataProcessPass = false;
    *pec12RotationValue = getLightIntensity();
}
setLightIntensity(pec12RotationValue);

```

```

461         break;
462
463 //-----//
464 case EXPOSURE_TIME_MODIF:
465     if(isFirstDataProcessPass){
466
467         isFirstDataProcessPass = false;
468         *pec12RotationValue = getExposureTime();
469     }
470     setExposureTime(pec12RotationValue);
471     break;
472
473 //-----//
474 case TIME_BW_PICTURES_MODIF:
475     if(isFirstDataProcessPass){
476
477         isFirstDataProcessPass = false;
478         *pec12RotationValue = getTimeBwPictures();
479     }
480     setTimeBwPictures(pec12RotationValue);
481     break;
482
483 //-----//
484 case SAVE_DATA_START:
485     if(isFirstDataProcessPass){
486
487         isFirstDataProcessPass = false;
488         isInModifMode = false; // AFFICHER .. ECRAN
489         saveDataInEeprom(pStepperData);
490         /* Once the data are saved, back to previous menu */
491         isInModifMode = false;
492         menu.menuState = SETTINGS_MENU;
493         menu.menuSize = 5;
494     }
495     break;
496
497 //-----//
498 case AUTO_HOME_START:
499     if(isFirstDataProcessPass){
500
501         isFirstDataProcessPass = false;
502         /* Start the auto home seq. */
503         startAutoHome(pStepperData);
504         /* Once auto home seq. is started, back to previous menu */
505         isInModifMode = false;
506         menu.menuState = MANUAL_MODE_MENU;
507         menu.menuSize = 2;
508     }
509     break;
510
511 //-----//
512 case AUTOMATIC_MODE_START:
513     if(isFirstDataProcessPass){
514
515         isFirstDataProcessPass = false;
516         /* Start the auto home seq. */
517         startFullImagingSequence();
518         /* Once auto home seq. is started, back to previous menu */
519         isInModifMode = false;
520         menu.menuState = AUTOMATIC_MODE_MENU;
521         menu.menuSize = 1;
522     }
523     break;
524 }

```

```

525     }
526 }
527
528
529
530
531 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
532 void menuPrintProcess(STEPPER_DATA *pStepperData){
533 /* Print switch */
534 switch(menu.menuState){
535
536     case MAIN_MENU:
537         printMainMenu();
538         break;
539
540     case SETTINGS_MENU:
541         switch (menu.menuPage){
542             case 0: printParameterMenuPage0();
543             break;
544             case 1: printParameterMenuPage1();
545             break;
546         }
547         break;
548
549     case MOTOR_MENU:
550         switch (menu.menuPage){
551             case 0: printMotorMenu0(pStepperData);
552             break;
553             case 1: printMotorMenu1(pStepperData);
554             break;
555         }
556         break;
557
558     case LIGHT_MENU:
559         printLedsMenu();
560         break;
561
562     case BACKLIGHT_MENU:
563         printBackLightMenu();
564         break;
565
566     case CAMERA_MENU:
567         printCameraMenu();
568         break;
569
570     case SAVE_DATA_MENU:
571         printSaveDataMenu();
572         break;
573
574     case CAPTURE_MODE_MENU:
575         printChoiceSeqMenu();
576         break;
577
578     case MANUAL_MODE_MENU:
579         printManualModeMenu(pStepperData);
580         break;
581
582     case AUTOMATIC_MODE_MENU:
583         printAutoModeMenu(pStepperData);
584         break;
585
586     case ABOUT_MENU:
587         printAboutMenu();
588         break;
589
590     case AUTO_HOME_MENU:
591         printAutoHomeMenu();
592         break;
593

```

```

594         default:
595             break;
596     }
597 }
598
599
600
601
602
603
604 void printLcdInit(void){
605
606     char str[2];
607     ClrDisplay();
608     DisplayOnOff(DISPLAY_ON); //Disable cursor
609     SetPosition(LINE1);
610     WriteString("Auto RTI Capt System");
611     SetPosition(LINE2);
612     WriteString("08-09 2023");
613     SetPosition(LINE3);
614     WriteString("Meven Ricchieri");
615     SetPosition(LINE4);
616
617     int i;
618     for (i = 0; i < 20; i++){
619
620         APP_Delay_ms(75);
621         SetPosition(LINE4 + i);
622         sprintf(str, "%c", 0xD0);
623         WriteString(str);
624     }
625     APP_Delay_ms(150);
626 }
627
628 void printMainMenu(void){
629
630     ClrDisplay();
631     SetPosition(LINE1);
632     WriteString(" Capture mode");
633     SetPosition(LINE2);
634     WriteString(" Settings");
635     SetPosition(LINE3);
636     WriteString(" About");
637     SetPosition(LINE4);
638     WriteString(" ");
639 }
640
641 void printParameterMenuPage0(void){
642
643     ClrDisplay();
644     SetPosition(LINE1);
645     WriteString(" Return");
646     SetPosition(LINE2);
647     WriteString(" Motor");
648     SetPosition(LINE3);
649     WriteString(" Power light");
650     SetPosition(LINE4);
651     WriteString(" Back-light");
652 }
653
654 void printParameterMenuPage1(void){
655
656     ClrDisplay();
657     SetPosition(LINE1);
658     WriteString(" Camera");
659     SetPosition(LINE2);
660     WriteString(" Save data");
661 }
662

```

```

663 void printMotorMenu0(STEPPER_DATA *pStepperData){
664
665     char str[21];
666     ClrDisplay();
667     SetPosition(LINE1);
668     WriteString("  Return");
669     SetPosition(LINE2);
670     sprintf(str, "  Speed: %4dsteps/s", pStepperData->stepPerSec);
671     WriteString(str);
672     SetPosition(LINE3);
673     sprintf(str, "  Gear:      1:%3d", pStepperData->gearValue);
674     WriteString(str);
675     SetPosition(LINE4);
676     sprintf(str, "  Step angle: %1.2f%c", pStepperData->anglePerStep, 0x01);
677     WriteString(str);
678 }
679
680 void printMotorMenu1(STEPPER_DATA *pStepperData){
681
682     char str[21];
683     ClrDisplay();
684     SetPosition(LINE1);
685     /* A changer, en Duty pure, ensuite en %% */
686     sprintf(str, "  Power : %03d", pStepperData->dutyCycleStepper);
687     WriteString(str);
688 }
689
690 void printLedsMenu(void){
691
692     char str[21];
693     ClrDisplay();
694     SetPosition(LINE1);
695     WriteString("  Return");
696     SetPosition(LINE2);
697     /* 0.04 = 100 / 2500 */
698     sprintf(str, "  Intensity : %03.0f%%", ((float)appData.lightIntensity * 0.04));
699     WriteString(str);
700     //  SetPosition(LINE3);
701     //  sprintf(str, "  Light time: %03dms", appData.lightTime);
702     //  WriteString(str);
703 }
704
705 void printChoiceSeqMenu(void){
706
707     ClrDisplay();
708     SetPosition(LINE1);
709     WriteString("  Return");
710     SetPosition(LINE2);
711     WriteString("  Manual mode");
712     SetPosition(LINE3);
713     WriteString("  Auto mode");
714     SetPosition(LINE4);
715     WriteString("  ");
716 }
717
718 void printAboutMenu(void){
719
720     ClrDisplay();
721     SetPosition(LINE1);
722     WriteString("  Return");
723     SetPosition(LINE2);
724     WriteString("  Version 1.0.0");
725     SetPosition(LINE3);
726     WriteString("  Meven Ricchieri");
727     SetPosition(LINE4);
728     WriteString("  08-09 2023");
729 }
730
731 void printManualModeMenu(STEPPER_DATA *pStepperData){

```

```

732     char str[21];
733     ClrDisplay();
734     SetPosition(LINE1);
735     WriteString("  Return");
736     SetPosition(LINE2);
737     if(pStepperData->isIndexed == true) {
738         sprintf(str, "  Auto home    :%s", "DONE");
739     } else {
740         sprintf(str, "  Auto home    :%s", "NOK");
741     }
742     WriteString(str);
743     SetPosition(LINE3);
744     sprintf(str, "  Des. angle :%03.1f%c", (((float)pStepperData->stepToReach * 1.8)
745         / pStepperData->gearValue), 0x01);
746 //    sprintf(str, "  Steps      :%05d", stepperData.stepToDoReach);
747     WriteString(str);
748     SetPosition(LINE4);
749     sprintf(str, "  Real angle :%03.1f%c", (((float)pStepperData->performedSteps * 1.8)
750         / pStepperData->gearValue), 0x01);
751 //    sprintf(str, "  Steps      :%05d", pStepperData->performedStep);
752     WriteString(str);
753 }
754
755 void printAutoModeMenu(STEPPER_DATA *pStepperData){
756
757     char str[21];
758     ClrDisplay();
759     SetPosition(LINE1);
760     WriteString("  Return");
761     SetPosition(LINE2);
762     if(appData.isFullImaginSeqEnable == false) {
763         sprintf(str, "  Start sequence");
764     } else {
765         sprintf(str, "  Sequence is ON");
766     }
767     WriteString(str);
768     SetPosition(LINE3);
769     sprintf(str, "  Pictures:    %03d", appData.nbrOfShotsPerformed);
770     WriteString(str);
771     SetPosition(LINE4);
772     sprintf(str, "  Real angle :%03.1f%c", (((float)pStepperData->performedSteps * 1.8)
773         / pStepperData->gearValue), 0x01);
774     WriteString(str);
775 }
776
777 void printAutoHomeMenu(void){
778
779     ClrDisplay();
780     SetPosition(LINE1);
781     WriteString("  Return");
782     SetPosition(LINE2);
783     WriteString("  Press to index");
784     SetPosition(LINE3);
785     WriteString("  ");
786     SetPosition(LINE4);
787     WriteString("  ");
788 }
789
790 void printBackLightMenu(void){
791
792     char str[21];
793     ClrDisplay();
794     SetPosition(LINE1);
795     WriteString("  Return");
796     SetPosition(LINE2);
797     /* 0.04 = 100 / 2500 */
798     sprintf(str, "  Intensity : %03.0f%%", ((float)appData.backLightIntensitiy * 0.04));
799     WriteString(str);
800 }
```

```

801 }
802
803 void printCameraMenu(void){
804
805     char str[21];
806     ClrDisplay();
807     SetPosition(LINE1);
808     WriteString(" Return");
809     SetPosition(LINE2);
810     sprintf(str, " Expos time: %04dms", appData.exposureDuration);
811     WriteString(str);
812     SetPosition(LINE3);
813     sprintf(str, " Time bw pic:%04dms", appData.timeBetweenPictures);
814     WriteString(str);
815     SetPosition(LINE4);
816     WriteString(" Trigger : cable"); // <-- or IR but not ready
817 }
818
819 void printSaveDataMenu(){
820
821     ClrDisplay();
822     SetPosition(LINE1);
823     WriteString(" Return");
824     SetPosition(LINE2);
825     WriteString(" Confirm to save");
826     SetPosition(LINE3);
827     WriteString(" ! Old values will ");
828     SetPosition(LINE4);
829     WriteString(" be overwritten ! ");
830 }
831
832
833
834
835 /* Clear the first row all 4 lines */
836 void clearFirstRow(void){
837
838     SetPosition(LINE1);
839     WriteString(" ");
840     SetPosition(LINE2);
841     WriteString(" ");
842     SetPosition(LINE3);
843     WriteString(" ");
844     SetPosition(LINE4);
845     WriteString(" ");
846 }
847
848 /* Print cursor */
849 void printCursor(int32_t cursor){
850
851     char str[2];
852     clearFirstRow();
853     SetPosition(cursor * 0x20);
854     sprintf(str, "%c", RIGHT_ARROW);
855     WriteString(str);
856 }
857
858
859
860 //-----//  

861 saveDataInEeprom
862 bool saveDataInEeprom(STEPPER_DATA *pStepperData){
863
864     DATA_IN_EEPROM dataToSaveInEeprom;
865
866     /* Set the structure value for saving in EEPROM */
867     dataToSaveInEeprom.stepPerSec = pStepperData->stepPerSec;
868     dataToSaveInEeprom.stepPerTurn = pStepperData->stepPerTurn;
869     dataToSaveInEeprom.gearValue = pStepperData->gearValue;

```

```

869     dataToSaveInEeprom.anglePerStep = pStepperData->anglePerStep;
870
871     dataToSaveInEeprom.lightIntensity      = appData.lightIntensity;
872     dataToSaveInEeprom.timeBetweenPictures = appData.timeBetweenPictures;
873     dataToSaveInEeprom.exposureDuration   = appData.exposureDuration;
874
875     dataToSaveInEeprom.backLightIntensitiy = appData.backLightIntensitiy;
876
877     dataToSaveInEeprom.controlValue = CONTROL_VALUE;
878
879     Init_DataBuff();
880     /* Write in the EEPROM */
881     NVM_WriteBlock((uint32_t*)&dataToSaveInEeprom, sizeof(dataToSaveInEeprom));
882
883     return 0;
884 }
885
886 //-----//
887 /* Read the parameters from the EEPROM */
888 bool readDataFromEeprom(STEPPER_DATA *pStepperData){
889
890     DATA_IN_EEPROM dataReadFromEeprom;
891
892     Init_DataBuff();
893     /* Read in the EEPROM */
894     NVM_ReadBlock((uint32_t*)&dataReadFromEeprom, sizeof(dataReadFromEeprom));
895
896     /* Check if the control value is already inside the EEPROM */
897     if(dataReadFromEeprom.controlValue == CONTROL_VALUE){
898
899         /* Save data from EEPROM */
900         pStepperData->stepPerSec    = dataReadFromEeprom.stepPerSec;
901         pStepperData->stepPerTurn   = dataReadFromEeprom.stepPerTurn;
902         pStepperData->gearValue     = dataReadFromEeprom.gearValue;
903         pStepperData->anglePerStep = dataReadFromEeprom.anglePerStep;
904
905         appData.lightIntensity      = dataReadFromEeprom.lightIntensity;
906         appData.timeBetweenPictures = dataReadFromEeprom.timeBetweenPictures;
907         appData.exposureDuration   = dataReadFromEeprom.exposureDuration;
908
909         appData.backLightIntensitiy = dataReadFromEeprom.backLightIntensitiy;
910
911     } else {
912
913         // SAVE INIT VAL
914         saveDataInEeprom(pStepperData);
915     }
916 }
```

```

1  /*
2   * File:    menu.h
3   * Author:  ricch
4   *
5   * Created on August 30, 2023, 10:17 AM
6   */
7
8 #ifndef MENU_H
9 #define MENU_H
10
11 #ifdef __cplusplus
12 extern "C" {
13 #endif
14
15     #include <stdbool.h>
16     #include <stdint.h>
17     #include <stepperDriver.h>
18     #include "Mc32NVMUtil.h"
19
20     #define RIGHT_ARROW 0x10
21
22     // Enumerations
23
24     /* All menus */
25     typedef enum
26     {
27         MAIN_MENU = 0,
28         CAPTURE_MODE_MENU,
29         SETTINGS_MENU,
30         ABOUT_MENU,
31         MOTOR_MENU,
32         MANUAL_MODE_MENU,
33         LIGHT_MENU,
34         BACKLIGHT_MENU,
35         CAMERA_MENU,
36         SAVE_DATA_MENU,
37         AUTO_HOME_MENU,
38         AUTOMATIC_MODE_MENU,
39
40     } MENU_STATE;
41
42
43
44
45     typedef enum{
46
47         RETURN_SEL = 0,
48
49     } COMMON;
50
51     typedef enum{
52
53         CAPTURE_MODE_SEL = 0,
54         SETTINGS_SEL,
55         ABOUT_SEL,
56
57     } MAIN_MENU_LIST;
58
59     typedef enum{
60
61         LIGHT_INTENSITY_SEL = 1,
62         LIGHT_TIME_SEL,
63         TIME_BW_PICTURES,
64
65     } LEDS_MENU_LIST;
66
67     typedef enum{
68
69         AUTO_HOME_START_SEL = 1,

```

```

70
71     } AUTO_HOME_MENU_LIST;
72
73 typedef enum{
74
75     MANUAL_MODE_SEL = 1,
76     AUTOMATIC_MODE_SEL,
77
78 } CHOICE_SEQ_MENU_LIST;
79
80 typedef enum{
81
82     AUTO_HOME_SEL = 1,
83     ANGLE_SEL,
84
85 } MANUAL_MODE_MENU_LIST;
86
87 typedef enum{
88
89     AUTOMATIC_MODE_START_SEL = 1,
90
91 } AUTO_MODE_MENU_LIST;
92
93 typedef enum{
94
95     MOTOR_SEL = 1,
96     LEDS_SEL,
97     BACKLIGHT_SEL,
98     CAMERA_SEL,
99     SAVE_DATA_SEL,
100
101 } SETTINGS_MENU_LIST;
102
103 typedef enum{
104
105     SPEED_SEL = 1,
106     GEAR_SEL,
107     STEP_PER_TURN_SEL,
108     POWER_SEL,
109
110 } MOTOR_MENU_LIST;
111
112 typedef enum{
113
114     BACKLIGHT_INTENSITY_SEL = 1,
115
116 } BACKLIGHT_MENU_LIST;
117
118 typedef enum{
119
120     EXPOSURE_TIME_SEL = 1,
121     TIME_BW_PICTURESEL,
122
123 } CAMERA_MENU_LIST;
124
125
126
127 typedef enum{
128
129     ANGLE_MODIF = 0,
130     SPEED_MODIF,
131     GEAR_MODIF,
132     STEP_PER_TURN_MODIF,
133     POWER_MODIF,
134     BL_INTENSITY_MODIF,
135     LIGHT_INTENSITY_MODIF,
136     LIGHT_TIME_MODIF,
137     EXPOSURE_TIME_MODIF,
138     TIME_BW_PICTURE_MODIF,

```

```

139
140     SAVE_DATA_START,
141     AUTO_HOME_START,// INTERACT
142     AUTOMATIC_MODE_START,
143
144 } MODIF_LIST;
145
146
147
148 // Structures
149 typedef struct{
150
151     uint8_t menuPage;
152     uint8_t menuSize;
153     MENU_STATE menuState;
154     MODIF_LIST modifState;
155
156 } MENU;
157
158
159
160
161
162 // Prototypes
163 void printLcdInit(void);
164 void printMainMenu(void);
165 void printParameterMenuPage0(void);
166 void printParameterMenuPage1();
167 void printMotorMenu0(STEPPER_DATA *pStepperData);
168 void printMotorMenu1(STEPPER_DATA *pStepperData);
169 void printLedsMenu(void);
170 void printChoiceSeqMenu(void);
171 void printAboutMenu(void);
172 void printManualModeMenu(STEPPER_DATA *pStepperData);
173 void printAutoModeMenu(STEPPER_DATA *pStepperData);
174 void printAutoHomeMenu(void);
175 void printBackLightMenu(void);
176 void printCameraMenu(void);
177 void printSaveDataMenu(void);
178
179 void menuManagementProcess(void);
180 void menuActionProcess(int32_t pec12RotationValue);
181 void menuDataProcess(int32_t*pec12RotationValue, STEPPER_DATA *pStepperData);
182 void menuPrintProcess(STEPPER_DATA *pStepperData);
183
184 void clearFirstRow(void);
185 void printCursor(int32_t cursor);
186
187 bool saveDataInEeprom(STEPPER_DATA *pStepperData);
188 bool readDataFromEeprom(STEPPER_DATA *pStepperData);
189
190
191 #ifdef __cplusplus
192 }
193 #endif
194
195 #endif /* MENU_H */
196
197

```

```

1  /*
2   * File:    lights.c
3   * Author: ricch
4   *
5   * Created on September 5, 2023, 7:15 PM
6   */
7
8 #include "app.h"
9
10 extern APP_DATA appData;
11
12 //-----
13 lightManagementProcess
14 void sequenceManagementProcess(void){
15     static int32_t order = 5; //= angleDesired / gear;
16
17     if(appData.isFiveShotsSeqEnable){
18
19         /* Sequence of 5 pictures is enable */
20         //fiveShotsSeqProcess();
21         //startFiveShotsSeqProcess();
22     }
23     if(appData.isFullImaginSeqEnable){
24
25         /* Full sequence is enable */
26         switch (appData.valSeq){
27
28             case 0:
29                 appData.valSeq += fiveShotsSeqProcess();
30                 break;
31             case 1:
32                 setRotationToDo(getMyStepperStruct(), &order);
33                 if(getPerformedSteps(getMyStepperStruct()) == order){
34                     order += 5; // appData.angleBwEachSeq;
35                     appData.valSeq = 0;
36                     appData.seqClock1_ms = 0;
37                     appData.seqClock2_ms = 0;
38                     startFiveShotsSeqProcess();
39                 }
40                 break;
41         }
42     }
43 }
44
45
46 //-----
47 turnOffAllPwrLeds
48 void turnOffAllPwrLeds(void){
49
50     /* Turn off all power LED */
51     LED1_CMDOff();
52     LED2_CMDOff();
53     LED3_CMDOff();
54     LED4_CMDOff();
55     LED5_CMDOff();
56 }
57 //-----
58 startFiveShotsSequence
59 /* Start a sequence for 5 shots */
60 void startFiveShotsSequence(void){
61
62     appData.seqClock1_ms = 0;
63     appData.seqClock2_ms = 0;
64     appData.isFiveShotsSeqEnable = true;
65 }
66 //-----

```

```

startFullImagingSequence
67 void startFullImagingSequence(void){
68
69     appData.seqClock1_ms = 0;
70     appData.seqClock2_ms = 0;
71     appData.isFullImaginSeqEnable = true;
72     appData.valSeq = 0;
73     appData.nbrOfShotsPerformed = 0;
74     startFiveShotsSeqProcess();
75 }
76
77 //-----
78 simpleShotProcess
79 void startSimpleShotProcess(void){
80
81     appData.seqClock2_ms = 0;
82     DRV_TMR4_Start();
83 }
84 void startFiveShotsSeqProcess(void){
85
86     appData.seqClock1_ms = 0;
87     DRV_TMR0_Start();
88 }
89
90 //-----
91 imagingSeqProcess
92 /* This function takes 5 pictures with 5 different LEDs */
93 bool fiveShotsSeqProcess(void){
94
95     if(appData.seqClock1_ms == 0){
96         appData.ledId = PWR_LED1;
97         startSimpleShotProcess();
98     } else if(appData.seqClock1_ms == 1 * appData.timeBetweenPictures){
99         appData.ledId = PWR_LED2;
100        startSimpleShotProcess();
101    } else if(appData.seqClock1_ms == 2 * appData.timeBetweenPictures){
102        appData.ledId = PWR_LED3;
103        startSimpleShotProcess();
104    } else if(appData.seqClock1_ms == 3 * appData.timeBetweenPictures){
105        appData.ledId = PWR_LED4;
106        startSimpleShotProcess();
107    } else if(appData.seqClock1_ms == 4 * appData.timeBetweenPictures){
108        appData.ledId = PWR_LED5;
109        startSimpleShotProcess();
110    } else if(appData.seqClock1_ms >= 5 * appData.timeBetweenPictures){
111        appData.seqClock1_ms = 0;
112        startSimpleShotProcess();
113    }
114    if(appData.seqClock1_ms >= 5 * appData.timeBetweenPictures){
115
116        appData.seqClock1_ms = 0;
117        appData.seqClock2_ms = 0;
118        appData.isFiveShotsSeqEnable = false;
119        return 1;
120    }
121    return 0;
122 }
123
124
125 //-----
126 setLighIntensity
127 void setLightIntensity(int32_t *lightIntensity){
128
129     // Limit values to avoid problems
130     if(*lightIntensity < LIGHT_INTENSITY_MIN) *lightIntensity
131         = LIGHT_INTENSITY_MIN;
132     if(*lightIntensity > LIGHT_INTENSITY_MAX) *lightIntensity

```

```

132             = LIGHT_INTENSITY_MAX;
133
134     /* 25 = 2500 / 100 */
135     appData.lightIntensity = *lightIntensity * 25;
136     PLIB_MCPWM_ChannelPrimaryDutyCycleSet(MCPWM_ID_0, PWM_DIM_CH, appData.lightIntensity
137 );
138 }
139 int32_t getLightIntensity(void){
140     return appData.lightIntensity / 25;
141 }
142 //-----
143 //-----setTimeBwPictures
144 void setTimeBwPictures(int32_t *timeBwPictures){
145
146     int32_t time_bw_pictures_min = appData.exposureDuration +
147         3 * MARGIN_LED_DELAY;
148     // Limit values to avoid problems
149     if(*timeBwPictures < time_bw_pictures_min) *timeBwPictures
150         = time_bw_pictures_min;
151     if(*timeBwPictures > TIME_BW_PICTURES_MAX) *timeBwPictures
152         = TIME_BW_PICTURES_MAX;
153
154     appData.timeBetweenPictures = *timeBwPictures;
155 }
156 int32_t getTimeBwPictures(void){
157     return appData.timeBetweenPictures;
158 }
159 //-----
160 //-----setExposureTime
161 void setExposureTime(int32_t *exposureTime){
162
163     // Limit values to avoid problems
164     if(*exposureTime < EXPOSURE_TIME_MIN) *exposureTime = EXPOSURE_TIME_MIN;
165     if(*exposureTime > EXPOSURE_TIME_MAX) *exposureTime = EXPOSURE_TIME_MAX;
166
167     appData.exposureDuration = *exposureTime;
168 }
169 int32_t getExposureTime(void){
170     return appData.exposureDuration;
171 }
172
173 }
```

```
1  /*
2   * File:    lights.h
3   * Author:  ricch
4   *
5   * Created on September 5, 2023, 7:16 PM
6   */
```

```

1  /*
2   * File:    pec12.c
3   * Author:  ricch
4   *
5   * Created on August 30, 2023, 9:15 AM
6   */
7
8 #include "app.h"
9 #include "pec12.h"
10 #include "lcd_spi.h"
11
12 PEC12 pec12;
13
14 void scanPec12(void){
15
16     // Save old states for debounce
17     pec12.chA.state[3] = pec12.chA.state[2];
18     pec12.chA.state[2] = pec12.chA.state[1];
19     pec12.chA.state[1] = pec12.chA.state[0];
20     pec12.chA.state[0] = CHANNEL_AStateGet();
21
22     pec12.chB.state[1] = pec12.chB.state[0];
23     pec12.chB.state[0] = CHANNEL_BStateGet();
24
25     pec12.chC.state[3] = pec12.chC.state[2];
26     pec12.chC.state[2] = pec12.chC.state[1];
27     pec12.chC.state[1] = pec12.chC.state[0];
28     pec12.chC.state[0] = PEC12R_SWStateGet();
29
30     // Check if PEC12 is in rotation
31     if(pec12.chA.state[0] == 0 && pec12.chA.state[1] == 0
32         && pec12.chA.state[2] == 1 && pec12.chA.state[3] == 1) {
33
34         // Check direction of rotation
35         if(pec12.chB.state[0] == 1 && pec12.chB.state[1] == 1) {
36
37             // CW
38             pec12.incrOrDecr++;
39             // SetPosition(LINE3);
40             // sprintf(a_toPrint, "counter = %d",counter);
41             // WriteString(a_toPrint);
42         }
43         else{
44
45             //CCW
46             pec12.incrOrDecr--;
47             // SetPosition(LINE3);
48             // sprintf(a_toPrint, "counter = %d",counter);
49             // WriteString(a_toPrint);
50         }
51     }
52     // Check if PEC12 switch is pressed
53     if(pec12.chC.state[0] == 0 && pec12.chC.state[1] == 0
54         && pec12.chC.state[2] == 1 && pec12.chC.state[3] == 1) {
55
56         pec12.isPressed = true;
57     }
58 }
59
60 int8_t getPec12IncrOrDecr(void){
61
62     int8_t incrOrDecr = pec12.incrOrDecr;
63     pec12.incrOrDecr = 0;
64
65     return incrOrDecr;
66 }
67
68 int8_t getPec12SwitchEvent(void){
69

```

```
70     int8_t isPressed = pec12.isPressed;
71     pec12.isPressed = 0;
72
73     return isPressed;
74 }
```

```
1  /*
2   * File:    pec12.h
3   * Author:  ricch
4   *
5   * Created on August 30, 2023, 9:15 AM
6   */
7
8 #ifndef PEC12_H
9 #define PEC12_H
10
11 #ifdef __cplusplus
12 extern "C" {
13#endif
14
15 #include <stdbool.h>
16
17 typedef struct
18 {
19     bool state[4];
20
21 } CHANNEL;
22
23 typedef struct
24 {
25     CHANNEL chA;
26     CHANNEL chB;
27     CHANNEL chC;
28     int8_t incrOrDecr;
29     bool isPressed;
30
31 } PEC12;
32
33
34
35
36 void scanPec12(void);
37 int8_t getPec12IncrOrDecr(void);
38
39
40
41
42 #ifdef __cplusplus
43 }
44#endif
45
46 #endif /* PEC12_H */
```

```

1  ****
2  System Interrupts File
3
4  File Name:
5      system_interrupt.c
6
7  Summary:
8      Raw ISR definitions.
9
10 Description:
11     This file contains a definitions of the raw ISRs required to support the
12     interrupt sub-system.
13
14 Summary:
15     This file contains source code for the interrupt vector functions in the
16     system.
17
18 Description:
19     This file contains source code for the interrupt vector functions in the
20     system. It implements the system and part specific vector "stub" functions
21     from which the individual "Tasks" functions are called for any modules
22     executing interrupt-driven in the MPLAB Harmony system.
23
24 Remarks:
25     This file requires access to the systemObjects global data structure that
26     contains the object handles to all MPLAB Harmony module objects executing
27     interrupt-driven in the system. These handles are passed into the individual
28     module "Tasks" functions to identify the instance of the module to maintain.
29 ****
30
31 // DOM-IGNORE-BEGIN
32 ****
33 Copyright (c) 2011-2014 released Microchip Technology Inc. All rights reserved.
34
35 Microchip licenses to you the right to use, modify, copy and distribute
36 Software only when embedded on a Microchip microcontroller or digital signal
37 controller that is integrated into your product or third party product
38 (pursuant to the sublicense terms in the accompanying license agreement).
39
40 You should refer to the license agreement accompanying this Software for
41 additional information regarding your rights and obligations.
42
43 SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,
44 EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF
45 MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.
46 IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER
47 CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR
48 OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES
49 INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR
50 CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF
51 SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES
52 (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.
53 ****
54 // DOM-IGNORE-END
55
56 // ****
57 // ****
58 // Section: Included Files
59 // ****
60 // ****
61
62 #include "system/common/sys_common.h"
63 #include "app.h"
64 #include "system_definitions.h"
65 #include "lcd_spi.h"
66 #include "pec12.h"
67
68 // ****
69 // ****

```

```

70 // Section: System Interrupt Vector Functions
71 // ****
72 // ****
73
74 extern APP_DATA appData;
75 extern STEPPER_DATA stepperData;
76
77 //-----// TMR ID 1
78 /* This timer clocks the capture sequence */
79 /* Frequency = 1000Hz */
80 void __ISR(_TIMER_1_VECTOR, ipl1AUTO) IntHandlerDrvTmrInstance0(void){
81     PLIB_INT_SourceFlagClear(INT_ID_0, INT_SOURCE_TIMER_1);
82
83     /* Control sequence of the 5 LEDs */
84     if(appData.seqClock1_ms == 0){
85         appData.ledId = PWR_LED1;
86         startSimpleShotProcess();
87
88     } else if(appData.seqClock1_ms == 1 * appData.timeBetweenPictures){
89         appData.ledId = PWR_LED2;
90         startSimpleShotProcess();
91
92     } else if(appData.seqClock1_ms == 2 * appData.timeBetweenPictures){
93         appData.ledId = PWR_LED3;
94         startSimpleShotProcess();
95
96     } else if(appData.seqClock1_ms == 3 * appData.timeBetweenPictures){
97         appData.ledId = PWR_LED4;
98         startSimpleShotProcess();
99
100    } else if(appData.seqClock1_ms == 4 * appData.timeBetweenPictures){
101        appData.ledId = PWR_LED5;
102        startSimpleShotProcess();
103    }
104
105    if(appData.seqClock1_ms >= 5 * appData.timeBetweenPictures){
106
107        DRV_TMR0_Stop();
108        appData.seqClock1_ms = 0;
109        appData.isFiveShotsSeqEnable = false;
110        appData.valSeq = 1;
111        //return 1;
112    } else {
113        appData.seqClock1_ms++;
114    }
115}
116
117 //-----// TMR ID 2
118 /* This timer clocks the main state machine */
119 /* Frequency = 10000Hz */
120 void __ISR(_TIMER_2_VECTOR, ipl1AUTO) IntHandlerDrvTmrInstance1(void){
121
122     PLIB_INT_SourceFlagClear(INT_ID_0, INT_SOURCE_TIMER_2);
123
124     /* States machines update */
125     APP_UpdateAppState(APP_STATE_SERVICE_TASKS);
126 }
127
128 //-----// TMR ID 3
129 /* This timer clocks the stepper sequence */
130 /* Variable frequency */
131 void __ISR(_TIMER_3_VECTOR, ipl4AUTO) IntHandlerDrvTmrInstance2(void){
132
133     SIGN_LED_CMDOff();
134     PLIB_INT_SourceFlagClear(INT_ID_0, INT_SOURCE_TIMER_3);
135
136     changeSpeed(getMyStepperStruct());
137     processStepper(getMyStepperStruct());
138     SIGN_LED_CMDOn();

```

```

139 }
140 //-----// TMR ID 4
141 /* This timer is used for the APP_Delay_ms() function */
142 /* Frequency = 1000Hz */
143 void __ISR(_TIMER_4_VECTOR, ipl1AUTO) IntHandlerDrvTmrInstance3(void){
144
145     PLIB_INT_SourceFlagClear(INT_ID_0, INT_SOURCE_TIMER_4);
146
147     /* Timer for ms delay */
148     appData.msCounter++;
149 }
150
151 //-----// TMR ID 5
152 /* Frequency = 1000Hz */
153 void __ISR(_TIMER_5_VECTOR, ipl3AUTO) IntHandlerDrvTmrInstance4(void)
154 {
155     PLIB_INT_SourceFlagClear(INT_ID_0, INT_SOURCE_TIMER_5);
156
157     //-----// Start
158     //-----// of sequence
159     if(appData.seqClock2_ms == 0) {
160
161         switch (appData.ledId) {
162             /* Turn on LED */
163             case PWR_LED1:
164                 turnOffAllPwrLeds();
165                 LED1_CMDOn();
166                 break;
167
168             case PWR_LED2:
169                 turnOffAllPwrLeds();
170                 LED2_CMDOn();
171                 break;
172
173             case PWR_LED3:
174                 turnOffAllPwrLeds();
175                 LED3_CMDOn();
176                 break;
177
178             case PWR_LED4:
179                 turnOffAllPwrLeds();
180                 LED4_CMDOn();
181                 break;
182
183             case PWR_LED5:
184                 turnOffAllPwrLeds();
185                 LED5_CMDOn();
186                 break;
187         }
188     }
189     if(appData.seqClock2_ms == MARGIN_LED_DELAY) {
190
191         /* Capture the target */
192         FOCUS_CMDOn();
193         TRIGGER_CMDOn();
194         appData.nbrOfShotsPerformed++;
195         SIGN_LED_CMDOn();
196     }
197     if(appData.seqClock2_ms == appData.exposureDuration + MARGIN_LED_DELAY) {
198
199         TRIGGER_CMDOff();
200         FOCUS_CMDOFF();
201     }
202     //-----// End of
203     //-----// sequence
204     if(appData.seqClock2_ms >= appData.exposureDuration + (2 * MARGIN_LED_DELAY)) {
205
206         /* Turn off TMR4 */
207
208     }
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2196
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2287
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2296
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
230
```

```
206     DRV_TMR4_Stop();
207     turnOffAllPwrLeds();
208     appData.seqClock2_ms = 0;
209     appData.ledId = ALL_LED_DISABLE;
210 } else {
211     appData.seqClock2_ms++;
212 }
213 */
214 ****
215 End of File
216 */
217
```

12.6 Document de modification



ELCO – SLO

Projet ETML-ES – Modification

PROJET:	2309_AutomatisationPriseImageRTI_v1.0.0		
Entreprise/Client:	MCAH / M. Mathieu Bernard-Reymond	Département:	SLO
Demandé par (Prénom, Nom):	M. Serge Castoldi	Date:	11.09.2023
Objet (No ou réf, pièce, PCB...)	2309		
Version à modifier:	V1		

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

1 Description ou justification

La modifications 1 va permettre d'obtenir un filtre passe bas fonctionnel permettant de limiter le nombre de rebonds du PEC12.

2 Référence conception

K:\ES\PROJETS\SLO\2309_AutomatisationPriseImageRTI

3 Détail des modifications

#	Description	Fait	Approuvé
1	Modifier le schéma du filtre du PEC12, R16 doit être connecté directement à la sortie du CHANNEL A et R17 doit être connecté directement à la sortie du CHANNEL B.	NOK	
2	Rechercher l'origine du courant réduit du driver de LED puis effectuer la modification. Piste : Tension d'alimentation trop basse, tester avec alimentation 24V	NOK	
3			
4			
5			
6			

4 Remarques

-

12.7 Planning

No jour de travail	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	29	30	31	32	33	34	35	36
Date	lundi , 7. août 2023	mardi , 8. août 2023	mercredi , 9. août 2023	jeudi , 10. août 2023	vendredi , 11. août 2023	samedi , 12. août 2023	dimanche , 13. août 2023	lundi , 14. août 2023	mardi , 15. août 2023	mercredi , 16. août 2023	jeudi , 17. août 2023	vendredi , 18. août 2023	samedi , 19. août 2023	dimanche , 20. août 2023	lundi , 21. août 2023	mardi , 22. août 2023	mercredi , 23. août 2023	jeudi , 24. août 2023	vendredi , 25. août 2023	samedi , 26. août 2023	dimanche , 27. août 2023	lundi , 28. août 2023	mardi , 29. août 2023	mercredi , 30. août 2023	jeudi , 31. août 2023	vendredi , 1. septembre 2023	samedi , 2. septembre 2023	dimanche , 3. septembre 2023	lundi , 4. septembre 2023	mardi , 5. septembre 2023	mercredi , 6. septembre 2023	jeudi , 7. septembre 2023	vendredi , 8. septembre 2023	samedi , 9. septembre 2023	dimanche , 10. septembre 2023	lundi , 11. septembre 2023
Cahier des charges																																				
Pré-étude																																				
Dimensionnement + design + schéma																																				
Design du PCB + BOM																																				
Design de des parties mécaniques																																				
Montage PCB																																				
Réalisation du software																																				
Mise en service et tests																																				
Rédaction rapport																																				
Finalisation/corrections/documentation																																				

12.8 Journal de travail

Jour	Date	Nbr période	Tâches effectuées	Nbr de café
Lundi	07/08/2023	7	Réception et analyse du cahier des charges, préparation des documents de base, recherche d'informations sur la puissance du moteur et des LEDs.	5
Mardi	08/08/2023	10	Correction du planning, recherche d'informations et de composants pour la pré-étude, réunion d'une heure.	3
Mercredi	09/08/2023	9	Recherche d'informations pour la pré-étude.	4
Jeudi	10/08/2023	9	Recherche d'informations pour la pré-étude, début de la phase de conception avec le développement de la partie driver des LEDs.	4
Vendredi	11/08/2023	9	Dimensionnement du driver des LEDs et de la communication avec l'appareil photo Nikon D750.	3
Samedi	12/08/2023			
Dimanche	13/08/2023	3	Choix du LCD et du rétro-éclairage.	2
Lundi	14/08/2023	9	Dimensionnement des systèmes de déclenchement d'appareils photo.	5
Mardi	15/08/2023	9	Dimensionnement du driver de moteur et changement du driver de LED de puissance, choix du PIC et réunion de deux heures.	4
Mercredi	16/08/2023	9	Dimensionnement du driver de moteur et du PIC32MK.	4
Jeudi	17/08/2023	8	Correction de détails sur le schéma et conception du PCB.	3
Vendredi	18/08/2023	10	Conception du PCB, revue du PCB par M. Zoubir et validation du PCB sur Eurocircuit.	4
Samedi	19/08/2023			3
Dimanche	20/08/2023			
Lundi	21/08/2023	9	Réalisation de la BOM, ajout des composants hors PCB, début du design mécanique.	4
Mardi	22/08/2023	8	Commande des composants, conception mécanique du boîtier.	4
Mercredi	23/08/2023	9	Finalisation de la conception du boîtier, réunion de 45 minutes.	3
Jeudi	24/08/2023	8	Réception du PCB, montage du PIC32MK sur le PCB, début de programmation (génération de PWM).	2
Vendredi	25/08/2023	8	Programmation pour test des PWMs, montage de composants sur le PCB.	3
Samedi	26/08/2023			3
Dimanche	27/08/2023			
Lundi	28/08/2023	9	Montage du PCB complet, programmation de test et installation du driver LCD.	4
Mardi	29/08/2023	9	Programmation pour tester le LCD, les LED et le driver du moteur pas à pas.	5
Mercredi	30/08/2023	10	Envoi du boîtier à imprimer, test des optocoupleurs, essai du contact Reed, programmation de l'affichage et des menus.	4
Jeudi	31/08/2023	10	Programmation des menus et correction de bugs.	5
Vendredi	01/09/2023	9	Programmation des menus et correction de bugs.	5
Samedi	02/09/2023			
Dimanche	03/09/2023	5	Rattrapage du retard dans la rédaction du rapport.	3
Lundi	04/09/2023	10	Rattrapage du retard dans la rédaction du rapport et programmation des menus.	5
Mardi	05/09/2023	10	Programmation de la séquence et des menus de configuration.	4
Mercredi	06/09/2023	10	Création d'un câble de commande d'appareil photo.	3
Jeudi	07/09/2023	10	Test du déclenchement de l'appareil photo au palais de Rumine (musée), correction de bugs au niveau de la séquence des LEDs et programmation de la séquence complète.	3
Vendredi	08/09/2023	10	Programmation des derniers points importants de la séquence.	4
Samedi	09/09/2023	8	Mise en service et mesures des différents composants critiques du système.	3
Dimanche	10/09/2023	8	Finalisation de la documentation et du rapport.	3
Lundi	11/09/2023	1	Rendu du rapport.	
Total		253	Total cafés :	114

12.9 Mode d'emploi

2309_AutomatisationPriseImagesRTI-Mode_d_Emlpoi_v1.0.0

Mode d'emploi du système :

Le fonctionnement du système est intuitif et facile à maîtriser. Le menu principal propose deux modes d'opération : le mode capture et le mode réglage. Un troisième menu permet de consulter les informations du système sans possibilité d'interaction.

Dans le mode capture, vous pouvez prendre des images de l'objet préalablement placé au centre de la machine. Vous avez le choix entre deux modes de capture : manuel et automatique.

Le mode manuel vous donne un contrôle total sur l'angle du bras motorisé équipé des LEDs de puissance. Vous pouvez également choisir quand déclencher une séquence de prise de 5 images. Dans ce mode, vous avez la possibilité d'indexer la machine pour obtenir l'angle 0 à l'emplacement de l'index. Pour ajuster l'angle du bras, activez la modification de position et tournez l'encodeur. Pour lancer la séquence, appuyez simplement une fois sur le bouton poussoir.

Le mode automatique vous permet de lancer une séquence complète de prise d'images, généralement environ 200 photos de l'objet.

Le menu des réglages vous permet de personnaliser le comportement de la machine pour qu'elle réponde à vos besoins spécifiques. Il est crucial de noter que certaines configurations incorrectes, notamment celles liées au moteur, peuvent rendre le système inutilisable. Une fois que vous avez effectué vos réglages, vous avez la possibilité de sauvegarder la configuration pour qu'elle soit conservée en mémoire et chargée automatiquement au démarrage de la machine.

La Figure ci-dessous présente une vue d'ensemble de tous les menus disponibles.

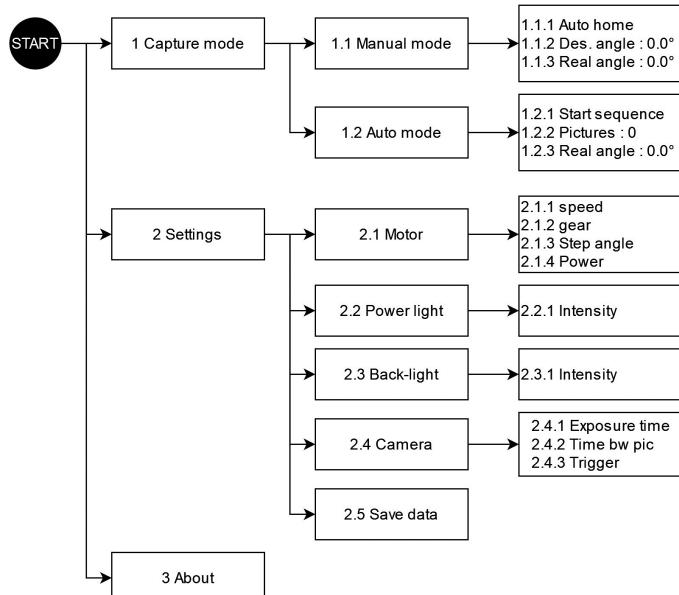
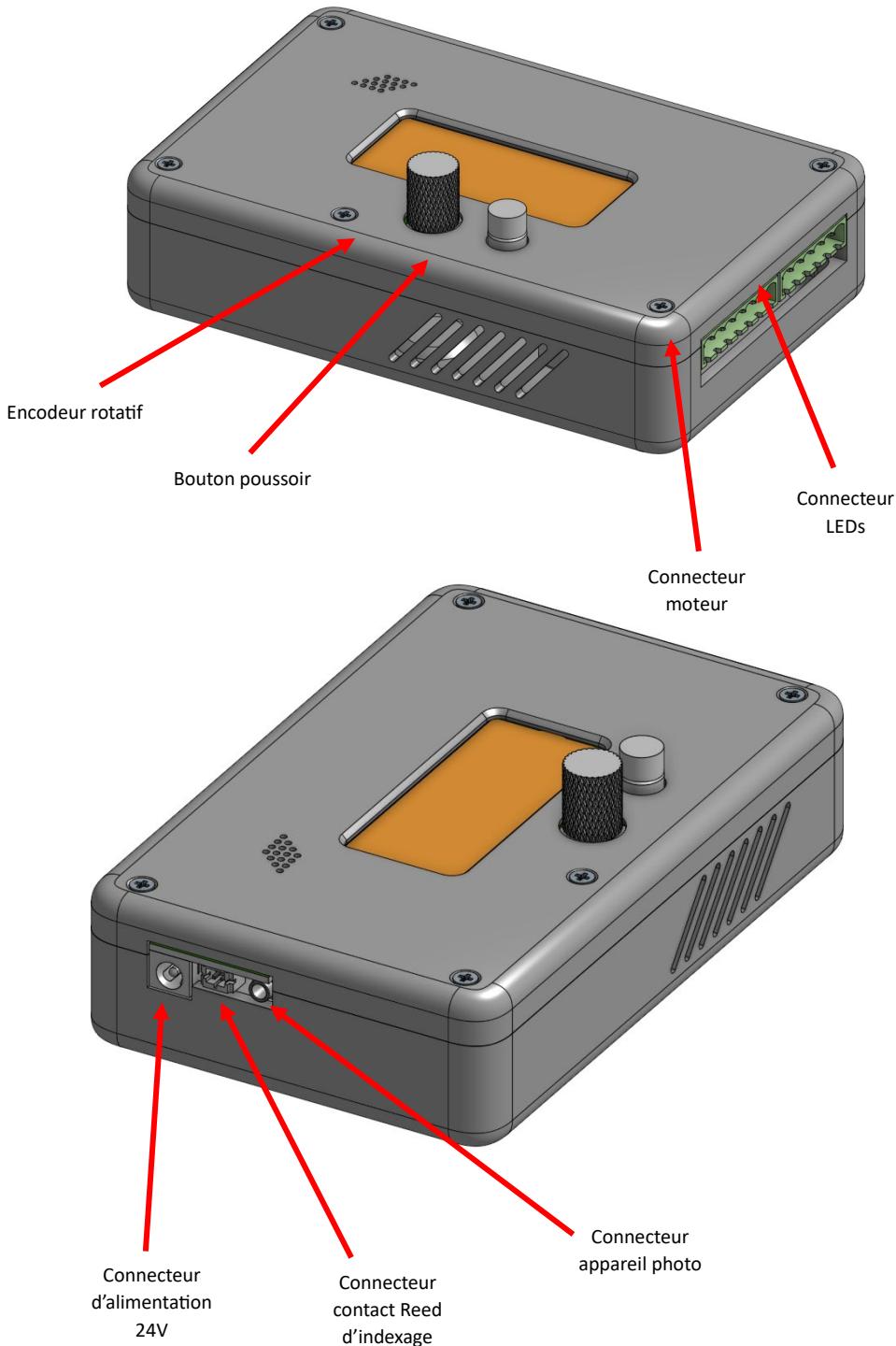


Figure 1 – Menus

Meven Ricchieri



Meven Ricchieri

12.10 PV



Procès-verbal du 08.08.2023

Présents

- M. Castoldi
- M. Ricchieri

État des lieux

- Cahier des charges validé (sauf erreur de marque d'appareil photo Canon au lieu de Nikon), planning terminé, schéma bloc en cours, recherche de composants en cours

Problèmes rencontrés

-

Solutions proposées

-

Décisions prises

- Driver de moteur simple sans micros (soit 1 double pont en H soit 2 pont en H)

Suite du projet / objectifs - jusqu'au mardi 15 août

- Choix des différents composants principaux, création de la schématique / revue schéma

Prochaine réunion :

15.08.2023, 15h00, ETML-ES

Destinataires de ce PV

Serge Castoldi

Lausanne le 08.08.2023

Meven Ricchieri



Ecole supérieure, école des métiers Lausanne
Rue de Sébeillon 12
CH-1004 Lausanne
PV_MevenRicchieri_08.08.2023.docx

www.etml-es.ch
| 1 / 1 |

PV de séance

diplôme***Procès-verbal du 15.08.2023******Présents***

- M. Castoldi
- M. Ricchieri

État des lieux

- Schéma terminé à 75%, la partie PIC32 et la partie du driver de moteur ne sont pas encore réalisées.

Problèmes rencontrés

- Connectique avec appareil photo Nikon D750 inconnue
- Modèle de moteur ne peut pas être décidé

Solutions proposées

- Demander par mail au mandant du projet
- Possibilité de commander un moteur DC ou stepper

Décisions prises

- Famille du PIC : PIC32MK, changer le driver de LEDs, ajouter certaines sécurités au schéma et les trois entrées QEI (Quadrature Encoder Interface)

Suite du projet / objectifs - jusqu'au mercredi 23 août

- D'ici vendredi 18, PCB commandé
- D'ici mercredi 23, composants commandés et design boîtier commencé

Prochaine réunion :

23.08.2023, 15h15, ETML-ES

Destinataires de ce PV

Ecole supérieure, école des métiers Lausanne
 Rue de Sébeillon 12
 CH-1004 Lausanne
 PV_MevenRicchieri_15.08.2023.docx

www.etml-es.ch
 | 1 / 2 |



ETML-ES

Serge Castoldi

Lausanne le 15.08.2023

Meven Ricchieri



PV de séance



Procès-verbal du 23.08.2023

Présents

- M. Castoldi
- M. Ricchieri

État des lieux

- PCB envoyé à produire chez Eurocircuit, composants commandés chez Digi-Key et design du boîtier terminé.

Problèmes rencontrés

- Commande PCB (UPS) retardée donc éventuels retards par rapport au planning.

Solutions proposées

- -

Décisions prises

- Commencer le software sur un autre PCB (ancien projet utilisant un pic de la même famille ou un PIM)

Suite du projet / objectifs - jusqu'au mercredi 23 août

- Terminer le montage du PCB et contrôle de fonctionnement des parties non software.

Prochaine réunion :

30.08.2023, 15h15, ETML-ES

Destinataires de ce PV

Serge Castoldi



Ecole supérieure, école des métiers Lausanne
Rue de Sébeillon 12
CH-1004 Lausanne
PV_MevenRicchieri_23.08.2023.docx

www.etml-es.ch
| 1 / 2 |



Lausanne le 23.08.2023

Meven Ricchieri



Procès-verbal du 30.08.2023

Présents

- M. Castoldi
- M. Ricchieri

État des lieux

- PCB monté, hardware testé à 75%

Problèmes rencontrés

- Courant dans les LEDs de puissance moins élevé que prévu (1A au lieu de 1.5)

Solutions proposées

- Effectuer des mesures pour connaître l'origine du problème

Décisions prises

- Ne pas mettre la priorité sur le courant des LEDs

Suite du projet / objectifs - jusqu'au mercredi 23 août

- Finir de tester le hardware et programmer les éléments essentiels

Prochaine réunion :

06.09.2023, 15h15, ETML-ES

Destinataires de ce PV

Serge Castoldi

Lausanne le 23.08.2023

Meven Ricchieri



Ecole supérieure, école des métiers Lausanne
Rue de Sébeillon 12
CH-1004 Lausanne
PV_MevenRicchieri_30.08.2023.docx

www.etml-es.ch
| 1 / 1 |

PV de séance

diplôme

Procès-verbal du 06.09.2023

Présents

- M. Castoldi
- M. Ricchieri

État des lieux

- Hardware testé et fonctionnel

Problèmes rencontrés

- Courant dans les LEDs de puissance moins élevée que prévu (1A au lieu de 1.5)

Solutions proposées

- Effectuer des mesures pour connaître l'origine du problème

Décisions prises

- Ne pas mettre la priorité sur le courant des LEDs

Suite du projet / objectifs - jusqu'au 11 septembre 2023

- Terminer de programmer la séquence et tester la prise d'image si possible avec un appareil photo, ensuite régler le courant dans le moteur et résoudre le problème du courant dans les LEDs

Prochaine réunion :

Destinataires de ce PV

Serge Castoldi

Lausanne le 06.09.2023



Ecole supérieure, école des métiers Lausanne
 Rue de Sébeillon 12
 CH-1004 Lausanne
 PV_MevenRicchieri_06.09.2023.docx

www.etml-es.ch
 | 1 / 2 |



ETML-ES

Meven Ricchieri

Références

- [1] Reflectance Transformation Imaging - Home. URL : <https://vcg.isti.cnr.it/rti/index.php> (visité le 07/08/2023).