

Projet de diplôme

Technicien ES en génie électrique,
spécialisation électronique

2312

Système d'accès par badge pour place de travail

Réalisé par :

Miguel Santos

Expert n° 1 :

Emilien Coulinge

Maître de diplôme :

Philippe Bovey

Expert n° 2 :

Daniel Bommottet

Table des matières

1 Introduction	7
1.1 Contexte	7
1.2 But du projet	7
1.3 Organisation	7
2 Conception	8
2.1 Schéma-bloc du système	8
2.2 Connecteurs 230VAC	9
2.3 Convertisseur AC/DC	10
2.4 Commutation 230VAC	11
2.5 Microcontrôleur	12
2.6 Ethernet	14
2.7 Module Wi-Fi	16
2.8 RFID	17
2.9 LEDs d'interface	18
2.10 LEDs témoins	19
2.11 Buzzer	19
2.12 Points de mesure	20
2.13 Boitier	20
2.14 Serveur externe	21
3 Hardware	22
3.1 Vues des couches du PCB	22
3.2 Vues réalistes du PCB	23
3.3 Spécifications du PCB	24
3.4 Règles de fabrication	24
3.5 Placement des composants	25
3.6 Largeurs de pistes	26
3.7 Pistes d'alimentation +3V3	26
3.8 Pistes de puissance 230 [VAC]	26
3.9 Plan de masse	27
3.10 Oscillateurs externes	27
3.11 Ethernet	28
3.12 ESP32	29
3.13 Boitier	29
4 Firmware	30

4.1 Approche utilisée	30
4.2 Machine d'état global	30
4.3 Librairies utilisées	31
4.4 Librairie : ESP	32
4.5 Librairie : CHU	33
4.6 Librairies : RFIDB1Client	34
4.7 Librairie : BZR	35
4.8 Librairie : TLC5973"	36
4.9 Librairie : SerialTimer	37
4.10 Calculs des timers	37
5 Software	38
6 Mesures	39
6.1 Matériel utilisé	39
6.2 Convertisseur AC/DC	39
6.3 Commutation 230VAC	40
6.4 Module RFID	41
6.5 Module ESP32	42
6.6 TLC5973.	43
6.7 Buzzer	44
7 Etat d'avancement	45
8 Conclusion	46
9 Bibliographie	47
10 Logiciels	49
11 Figures	50
12 Tableaux	51
13 Equations	52
14 Annexes	53
14.1 Cahier des charges	53
14.2 Planification	53
14.3 Journal de travail	53
14.4 Procès-verbaux des séances hebdomadaires	53
14.5 Schémas électroniques	53
14.6 Fichiers de fabrication	53
14.7 Liste des règles Altium	53
14.8 Code source firmware	53

14.9 Fiche de modifications	53
14.10 Mode d'emploi	53

Glossaire

ETML	École Technique et des Métiers de Lausanne
ES	École Supérieure
CDC	Cahier Des Charges
PCB	Printed Circuit Board
RFID	Radio Frequency Identification
PWM	Pulse Width Modulation
GPIO	General Purpose Input Output
RMII	Reduced Media-Independent Interface
MDI	Medium Dependent Interface
ABS	Acrylonitrile Butadiène Styrene
PETG	PolyEthylene Terephthalate Glycol
PLA	Acide Polylactique

1 Introduction

1.1 Contexte

Ce projet est réalisé dans le cadre de la formation en École Supérieure (ES) en génie électrique, spécialisation électronique. Il constitue la validation finale des connaissances et des compétences acquises tout au long de la formation. Sa réussite conduit à l'obtention du diplôme de l'ES.

Ce projet est réalisé en fin de formation et possède une durée de 5 semaines. Un enseignant de l'ES, nommé le Maître de diplôme, en assure le suivi. Ce dernier réalise l'évaluation finale de ce rapport ainsi que de la défense orale, conjointement à deux experts externes.

1.2 But du projet

L'objectif de ce projet est de réaliser un système d'accès par badge aux différents équipements disponibles à l'ES. En fonction des droits accordés à la personne, l'alimentation électrique des différents équipements sera activée ou non. Cela concerne notamment chacune des places de travail des étudiants et le « local de montage » de l'ES.

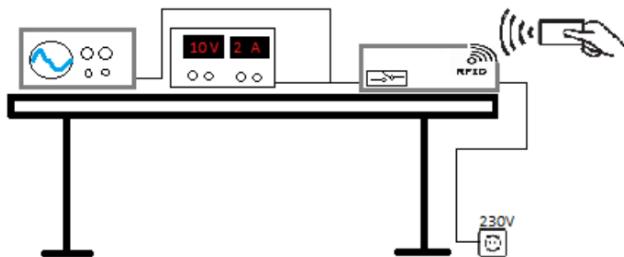


Figure 1 : Illustration du système, issu du CDC

Le principal but est d'améliorer la gestion de la consommation électrique en évitant les oubliers d'extinction des appareils, d'assurer la sécurité des stations de brasage en réduisant les risques d'incendie, de gérer les droits d'utilisations et de fournir un suivi à l'aide d'un journal.

Le système électronique doit être capable de lire un badge RFID, d'activer ou de désactiver un commutateur 230 VAC en fonction des informations stockées dans une base de données accessible via Ethernet ou Wi-Fi. Il devra également gérer le concept de timeout en utilisant des indications lumineuses et/ou sonores. Le dispositif doit avoir une adresse permettant de le relier à la base de données pour enregistrer des informations telles que l'identité de l'utilisateur, la fréquence et la durée d'utilisation.

La conception du système, la réalisation d'un circuit imprimé (PCB) et sa programmation représentent les exigences minimales de la formation.

Le projet est mandaté par l'ES, qui a fourni un cahier des charges (CDC) détaillant les exigences précises du projet (Annexe 14.1).

1.3 Organisation

Le projet a été planifié en différentes phases et une documentation quotidienne des activités a été réalisée dans un journal de travail.

Une réunion hebdomadaire était organisée avec le Maître de diplôme afin d'examiner l'évolution du projet. Un procès-verbal a été dressé à chaque séance.

Ces documents sont disponibles en annexes de ce rapport. (14.2 à 14.4)

2 Conception

Cette partie vise à expliquer les raisons des choix de composants, en mettant en avant leurs avantages, inconvénients et dimensionnement.

Les datasheets des composants sont référencés en bibliographie ou accessibles via les liens dans les tableaux. Les schémas électriques correspondants sont joints en annexe (14.5).

2.1 Schéma-bloc du système

Le schéma-bloc suivant résume les composants du système :

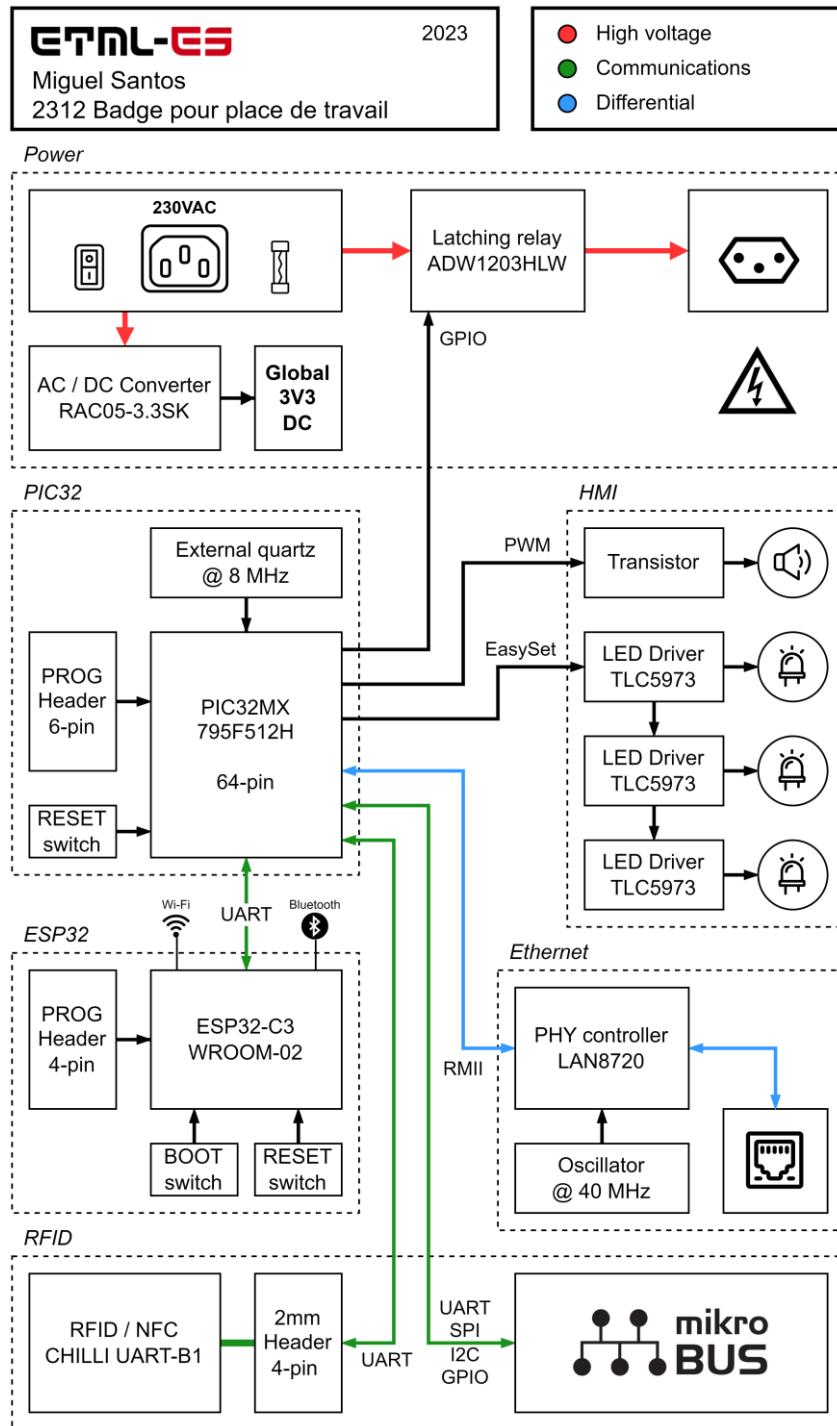


Figure 2 : Schéma-bloc du système

2.2 Connecteurs 230VAC

L'appareil doit être alimenté directement sur le réseau électrique. Il doit aussi pouvoir fournir une sortie 230VAC avec le courant standard d'une prise électrique (10A).

L'entrée est réalisée au moyen d'un câble et d'une prise IEC C14 standard, couramment utilisée au sein de l'ES. Celle-ci est montée sur le boîtier et intègre un porte-fusible et un interrupteur, connectés en interne, pour assurer la sécurité du circuit et de l'utilisateur.

La sortie du circuit est réalisée avec un câble doté d'une prise électrique CH (T13) femelle permettant de connecter un appareil ou une multiprise.

Connecteur d'entrée 230VAC		Datasheet [1]
Fabricant	Schurter	
N° de fabricant	DD11.0111.1111	
Tension nominale	250 RMS	[VAC]
Courant nominale	10	[A]
Taille fusible	5 x 20	[mm]
Puissance fusible	2	[W]
Interrupteur	2 pos. / non-illuminé Disjoncteur thermique	



Tableau 1 : Caractéristiques principales de la prise IEC C14

Les connecteurs externes sont reliés à des borniers pour acheminer la phase et le neutre à travers le PCB, utilisés par le convertisseur AC/DC et la commutation via le relai.

La mise à la terre est établie directement de l'entrée à la sortie, le boîtier ou le circuit ne nécessitant pas de protections particulières contrairement aux appareils connectés à la sortie.

Borniers		Datasheet [2]
Fabricant	Würth Elektronik	
N° de fabricant	69140170000xB	
Tension nominale	300 RMS	[VAC]
Courant nominale	10	[A]
Positions (x)	2 & 4	[-]



Tableau 2 : Caractéristiques principales des borniers

Tous les composants ci-dessus sont conçus pour supporter le courant maximal d'une prise électrique standard (10A) ainsi que la tension du réseau (230VAC RMS).

2.3 Convertisseur AC/DC

L'appareil étant directement alimenté à partir du réseau électrique, il est nécessaire d'employer un convertisseur AC/DC afin de fournir la tension nécessaire au circuit. Celle-ci a été fixée à +3,3VDC car elle est nécessaire aux principaux composants, tel que le microcontrôleur ou le module RFID.

Le courant maximal nécessaire a été déterminé en se basant sur les datasheets des composants, avec une marge de sécurité de 10% en cas d'imprévu (Tableau 3).

Estimation du courant maximal		[mA]
Microcontrôleur	PIC32MX795F512H	100
Modules RFID	CHILLI UART	100
	RFID CLICK	100
Module Wi-fi	ESP32-C3-WROOM	345
Module Ethernet	Module Ethernet	220
Commutation 230VAC	Relai 230V	140
Interfaces	LEDS RGB	180
	Buzzer	90
Total		1285
Avec marge de 10%		1413,5

Tableau 3 : Estimation du courant maximal

Le module choisi (Tableau 4) ne nécessite pas de composants externes et se distingue de ses concurrents par sa taille et son rapport qualité/prix. Il intègre des protections contre les surtensions, les courts-circuits, et un fusible. Bien que son rendement soit relativement bas en utilisation intensive, il se montre plus efficace en cas de faible consommation.

Convertisseur AC/DC		Datasheet [3]
Fabricant	RECOM	
N° de fabricant	RAC05-3.3SK	
Tension d'entrée	85 ~ 264 RMS [VAC]	
Tension de sortie	3.3 [VDC]	
Courant de sortie max.	1515 [mA]	
Fréquence interne	130 [kHz]	

Tableau 4 : Caractéristiques principales du convertisseur AC/DC

Une attention particulière est nécessaire en cas de développements futurs impliquant des technologies RFID proches de la fréquence de fonctionnement de 130kHz. Les badges utilisés dans ce projet ne se situent pas dans cette plage.

Bien que facultatifs, des condensateurs ont été ajoutés en sortie du circuit pour garantir la stabilité de la tension. Les valeurs ont été choisies de manière arbitraire. (Figure 3)

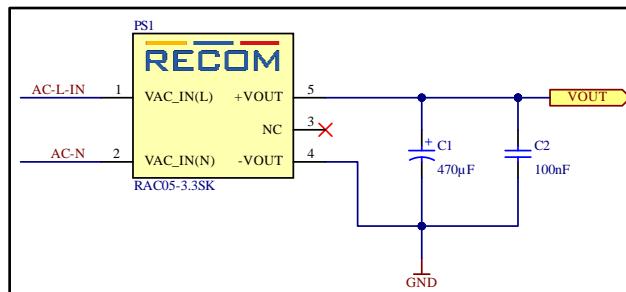


Figure 3 : Schéma du convertisseur AC/DC

2.4 Commutation 230VAC

L'utilisation d'un relais permet de réaliser la commutation de la sortie 230VAC tout en assurant une isolation galvanique entre les sections à haute et basse tension. Contrairement à d'autres alternatives, telles que les optocoupleurs, un relai présente l'avantage de pouvoir commuter des puissances plus élevées. Cependant, les contacts mécaniques ont l'inconvénient de s'user plus rapidement et il nécessite un courant de contrôle plus élevé.

Un relais à verrouillage est employé car il peut maintenir sa position pendant de longues périodes sans nécessiter une alimentation constante, réduisant ainsi la consommation de courant. Il possède aussi deux bobines de contrôle, ce qui élimine la nécessité d'un circuit externe pour inverser la tension des bobines. Il est le moins coûteux parmi les rares modèles de ce type capable de fonctionner à une tension de 3,3VDC.

Relai de puissance		Datasheet [4]
Fabricant	Panasonic	
N° de fabricant	ADW1203HLW	
Tension de contact max.	277 RMS	[VAC]
Courant de contact max.	16	[A]
Tension de bobine nom.	3	[V]
Courant de bobine nom.	133.3	[mA]
Type de bobine	Double bobine, à verrouillage	

Tableau 5 : Caractéristiques principales du relai de puissance

Des diodes de roue libres sont placées en parallèle des bobines pour protéger les composants des surtensions qui surviennent lorsque l'on désactive les bobines. Elles sont capables de supporter le courant et la tension inverse maximal de la bobine. Des diodes Schottky permettent de réagir rapidement aux variations soudaines de tension.

Les bobines sont contrôlées par des transistors externes, les résistances R3 et R4 fournissent un courant de base suffisamment élevé pour garantir la saturation du transistor, calculés comme suit :

$$R_B = \frac{U_{RB}}{I_{RB}} = \frac{(V_{CC} - V_{BE}) * 10}{I_L} = \frac{(3,3 - 0,7) * 10}{0,133} \cong 195 \Omega \rightarrow 180 \Omega \text{ E12}$$

Équation 1 : Résistance de base du transistor

Des résistances de « pull-up » et « pull-down » permettent d'assurer l'état des transistors au démarrage. La datasheet n'étant pas explicite sur quelle bobine est « set » ou « reset », des emplacements sont prévus sur chaque transistor. Cependant, leur nécessité est remise en question car le microcontrôleur peut potentiellement assurer cet état par défaut. Des tests de mise en service sont prévus pour confirmer cela, en particulier pour vérifier l'absence d'impulsions au démarrage.

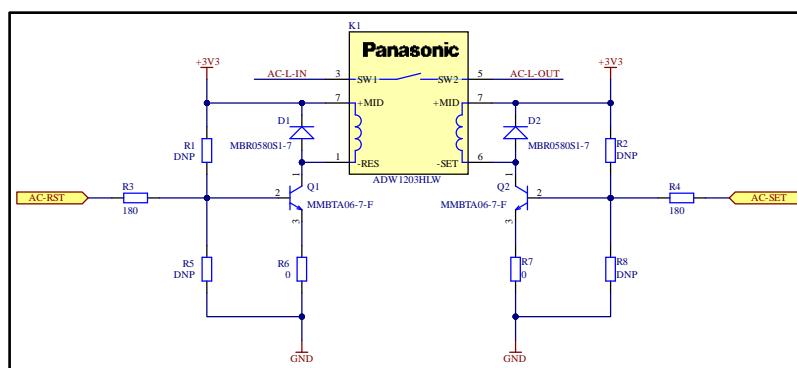


Figure 4 : Schéma du relai de puissance

2.5 Microcontrôleur

Le microcontrôleur a pour but de coordonner et gérer les différents périphériques. Il sert notamment d'interface entre le module RFID, le module Wi-Fi ainsi que les LEDs et le buzzer.

Un microcontrôleur de la famille PIC32 a été choisi en raison de sa popularité à l'ES et sa relative facilité de programmation avec MPLAB X. La série MX795 utilisée à l'ES apparaît comme le choix idéal grâce à son intégration du module Ethernet MAC, essentiel pour la connectivité Ethernet, ainsi que sa gamme étendue de périphériques de communication. Néanmoins, les PIC32 présentent l'inconvénients d'avoir une certaine rigidité dans le choix des broches pour les périphériques, ce qui nécessite une certaine attention lors de la phase de conception.

Microcontrôleur		Datasheet [5]
Fabricant	Microchip	
N° de fabricant	PIC32MX795F512HT-80I/PT	
Boîtier	64-TQFP	
Tension d'alimentation	2,3~3,6 [V]	
Fréquence	80 [MHz]	
Mémoire Flash	512 [ko]	
Mémoire RAM	128 [ko]	
Connectivité	CANbus, Ethernet, I ² C, SPI, UART/USART, USB OTG	
Nombre d'E./S.	53	

Tableau 6 : Caractéristiques principales du microcontrôleur

Il a été déterminé que les périphériques internes suivants sont nécessaires :

Périphériques interne PIC32		
Périphérique	# pin	Fonctionnalité
PGEx	2	Programmation et debug
OSCx	2	Oscillateur externe
Ethernet MAC	11	Connexions avec contrôleur PHY
		Communication avec l'ESP32
UART	6	Communication avec RFID / Chilli UART B1
		Réservé pour mikroBUS
SPI	4	Réservé pour mikroBUS
I2C	2	Réservé pour mikroBUS
OC	2	(Contrôle des LEDs)
		Contrôle du Buzzer
	1	Reset de l'ESP32
GPIO	2	Contrôle du relai
	4	Réservé pour mikroBUS
Total	36	

Tableau 7 : Périphériques et nombre de broches nécessaires pour le PIC32

Ce constat a conduit à l'élimination du boîtier 44 broches en raison de l'espace insuffisant pour les broches d'alimentation, ce qui a entraîné un choix arbitraire en faveur du boîtier 64 broches. Lors de l'assignation des broches, ce choix s'est avéré optimal car la capacité maximale en termes de périphériques a été atteinte (Figure 5).

Le modèle avec la mémoire interne maximale est sélectionné pour prévenir les limitations en programmation, avec la possibilité de la réduire ultérieurement.

Il ne faut pas oublier les résistances pull-up externes pour l'I²C, une valeur de 10 kΩ est recommandée selon les spécifications du standard.



Figure 5 : Assignation des broches du PIC32

La datasheet du fabricant fournit les recommandations en terme de découplage de l'alimentation (Figure 6) et de connexion au port de programmation (Figure 7). Un bouton a été rajouté pour permettre un redémarrage manuel.

Les valeurs des condensateurs nécessaires pour le quartz externe (Figure 8) ont été calculées en utilisant la formule suivante, ainsi que les informations de la datasheet du fabricant [6], en considérant que les deux condensateurs auront la même valeur :

$$C_L = \frac{C_1 * C_2}{C_1 + C_2} + C_0$$

$$C_1 = C_2 = 2 * (C_L - C_0) = 2 * (18 - 7) = 22 \text{ [pF]}$$

Équation 2 : Calcul des condensateurs du quartz externe

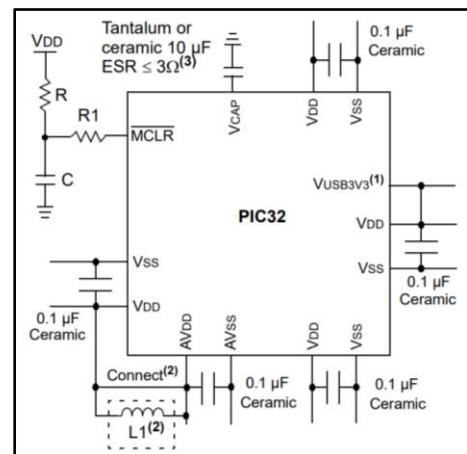


Figure 6 : Condensateurs de découplage PIC32

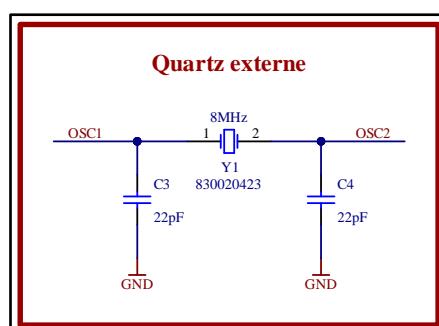


Figure 8 : Schéma du quartz externe PIC32

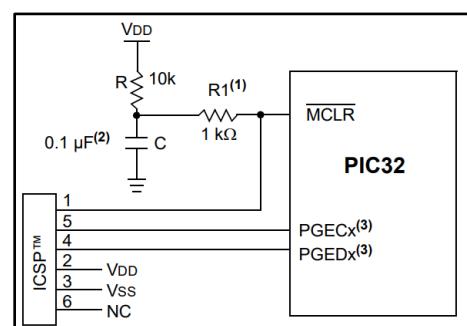


Figure 7 : Port de programmation PIC32

2.6 Ethernet

L'utilisation de l'Ethernet vise à fournir une option plus fiable que le Wi-Fi pour communiquer avec la base de données externe. Cependant, en raison de sa complexité, cette intégration est prévue en dernière priorité.

La première étape dans la mise en place d'une connexion Ethernet consiste à établir la connexion physique (couche OSI 1). Cette tâche est gérée par un contrôleur PHY, qui agit en tant qu'interface entre le connecteur RJ45 et l'Ethernet MAC (couche OSI 2) intégré au PIC32 () .

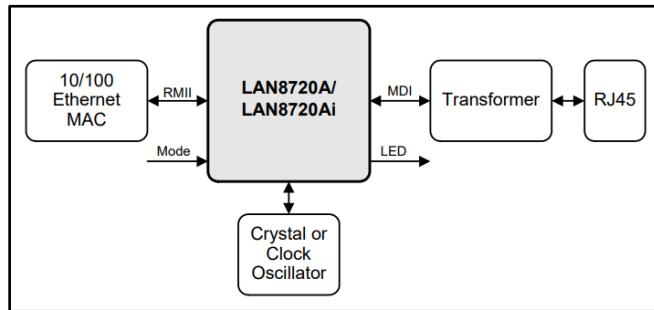


Figure 9 : Principe du contrôleur PHY

Pour simplifier la conception, le schéma du « Ethernet Starter Kit 2 » développé par Microchip a été reproduit [7] et notamment celui de la « PHY Daughter Board » présent sur le kit [8] (Figure 10). L'utilisation de composants du même fabricant garantit une certaine synergie et offre l'avantage d'avoir la configuration du contrôleur PHY directement disponible dans l'IDE.

Contrôleur PHY		Datasheet [9]
Fabricant	Microchip	
N° de fabricant	LAN8720A-CP-TR-ABC	
Boîtier	24-QFN	
Tension d'alimentation	1,62~3,6 [V]	
Interface	RMII	
Ethernet	10/100BASE-T	

Tableau 8 : Caractéristiques principales du contrôleur PHY

Le port RJ45 est doté de transformateurs de couplage. Leur rôle est d'assurer une isolation galvanique entre la ligne Ethernet et le circuit, permettant de le protéger contre les surtensions ou les perturbations électromagnétiques. L'isolation galvanique permet aussi de garantir les caractéristiques d'impédances de la ligne et du circuit.

Connecteur RJ45		Datasheet [10]
Fabricant	CUI devices	
N° de fabricant	CRJ011-ML3-TH	
Blindage	Blindé	
Magnétiques	Intégrés	
Ethernet	10/100BASE-T	

Tableau 9 : Caractéristiques principales du connecteur RJ45

L'oscillateur externe assure le fonctionnement du contrôleur PHY (Figure 11).

Les contraintes de placement et de routage sont détaillées dans la partie hardware.

Dans la figure ci-dessous, les résistances R14 à R17 adaptent l'impédance avec une valeur de 50Ω par fil, 100Ω par ligne différentielle. Les condensateurs C9 à C13, destinés à la réduction du bruit, n'ont pas été inclus dans le circuit, de même que la ferrite isolant l'alimentation analogique du +3V3. Ceci est basé sur l'analyse d'un circuit équivalent à l'ES où ces éléments n'étaient pas présents, mais le fonctionnement restait garanti pour les vitesses de communication concernées.

Les protocoles de communication utilisés que sont le RMII et le MDI nécessitent l'utilisation de paires différentielles. Celles-ci ont pour but de réduire les interférences sur la ligne. Pour cela, elles doivent posséder une impédance caractéristique, détaillé dans la partie hardware.

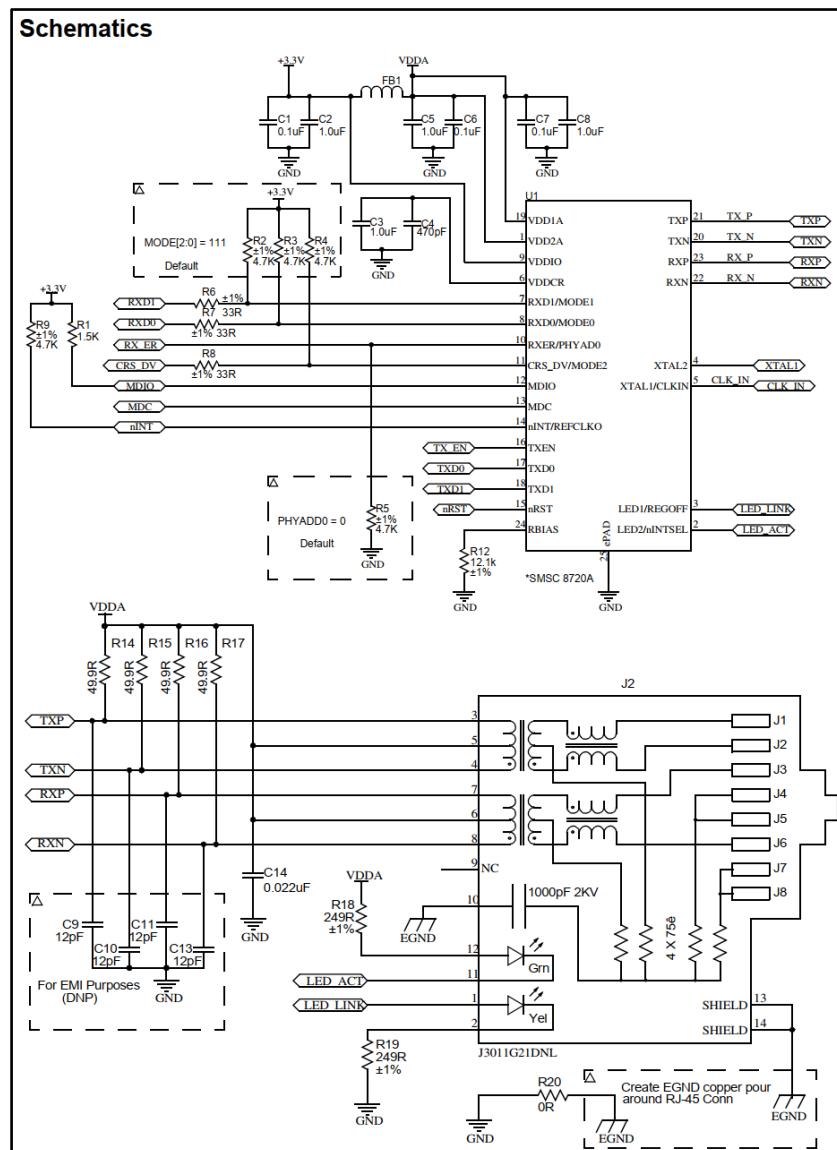


Figure 10 : Schéma de la "PHY Daughter Board"

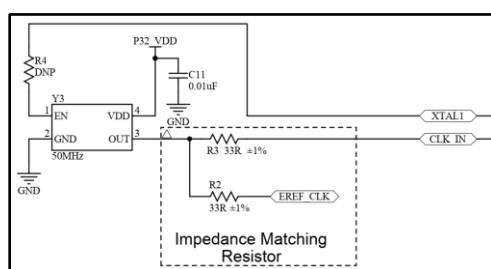


Figure 11 : Schéma de l'oscillateur externe sur le "Ethernet Kit 2"

2.7 Module Wi-Fi

Un module Wi-Fi permet d'assurer la connectivité sans fil avec la base de données externe. Il permet ainsi de vérifier les droits accordés aux badges scannés.

Un module « ESP32-C3 » a été sélectionné parmi les projets réalisés au sein de l'ES. Cela permet ainsi de bénéficier de l'expérience et de la documentation d'anciens étudiants ou des enseignants. Sa popularité et sa documentation abondante participent aussi à en faire un choix idéal.

Son contrôle s'effectue par l'envoi d'instructions AT à l'aide d'une communication UART. Le « ESP-AT User Guide » fournit une liste détaillée des commandes à utiliser [11].

Module Wi-Fi ESP32		Datasheet [12]
Fabricant	Espressif	
N° de fabricant	ESP32-C3-WROOM-02-N4	
Tension d'alimentation	3 ~ 3.6 [VDC]	
Courant de réception	82 ~ 84 [mA]	
Courant de transmission	280 ~ 345 [mA]	
Protocoles	802.11b/g/n, Bluetooth v5.0	

Tableau 10 : Caractéristiques principales du module Wi-Fi

Le module doit être préalablement programmé en mode « Boot Download » via le port de programmation. Pour cela, il suffit d'activer certaines broches à l'état haut lors du démarrage, conformément à la datasheet (Figure 12).

Booting Mode ¹			
Pin	Default	SPI Boot	Download Boot
GPIO2	N/A	1	1
GPIO8	N/A	Don't care	1
GPIO9	Internal weak pull-up	1	0

Figure 12 : Modes de démarrage de l'ESP32

Sur ce circuit, il faut maintenir le bouton « BOOT » enfoncé et redémarrer le module avec le bouton « RESET ». Les broches « IO2 » et « IO8 » peuvent être maintenues à l'état haut en utilisant une résistance de pull-up de $10k\Omega$, comme l'indique le fabricant dans la datasheet.

Des détails supplémentaires concernant la programmation sont disponibles dans la partie software.

(Corrections à apporter : voir fiche de modification)

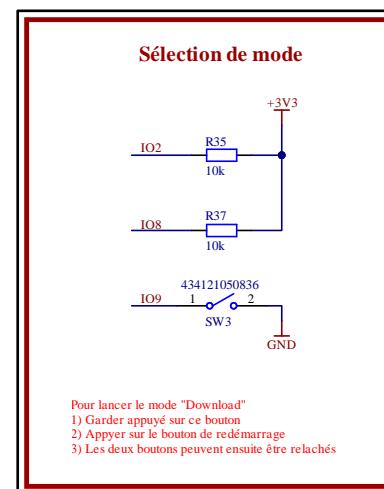


Figure 13 : Schéma de sélection du mode de l'ESP32

2.8 RFID

Des badges sont mis à disposition des élèves de l'ETML-ES pendant toute la durée de leur formation. Ceux-ci sont utilisés dans ce projet afin d'éviter aux élèves la nécessité de multiples badges. Un lecteur doit récupérer l'ID du badge scanné pour déterminer ensuite les droits qui lui sont accordé en interrogeant une base de données externe.

La technologie du badge a été identifiée à l'aide de l'application « NFC Tools » sur un smartphone Samsung S23 Ultra. La figure ci-dessous montre le fabricant et le modèle de la puce interne. Des informations techniques supplémentaires sont disponibles sur le site web du fabricant.[13]



Figure 14 : Technologie du badge RFID de l'ES

En résumé, le badge utilise puce RFID à 13,56MHz et dispose d'une mémoire d'un kilo-octet.

Lors de la recherche d'un lecteur compatible, le choix s'est porté vers un module tout-en-un afin de simplifier la conception, notamment de l'antenne. Deux modules se sont démarqués :

Comparaison des modules RFID		
Nom	RFID CLICK	CHILLI UART B1
Fabricant	MIKROE	Eccel Technology
Liens	Datasheet [14]	Datasheet [15]
Illustration		
Prix	25,23 CHF	34,82 CHF
Interface	UART, SPI	UART, GPIO
Taille	57,15 x 25,4 mm	75 x 50 mm
Connecteur	mikroBUS	Header 2,54 ou 2mm
Avantages	Déjà utilisé au sein de l'ES	Placement libre dans le boîtier Programmation simplifiée
Inconvénients	Antenne à faible portée	Coût plus élevé Taille plus grande

Tableau 11 : Comparaison des modules RFID

Le choix final s'est porté sur le module « Chilli UART B1 » en raison de sa liberté de placement dans le boîtier, prévu sur le dessus, ainsi que de sa portée d'antenne. Il sera connecté au circuit via un connecteur 2mm. Cependant, l'ES a demandé l'intégration d'une empreinte « mikroBUS » dans le système pour d'éventuels développements futurs, sans nécessité d'exécution immédiate.

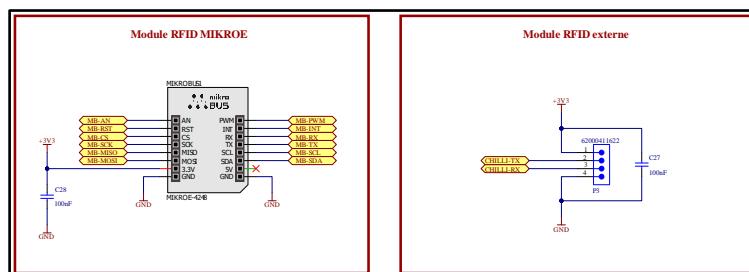


Figure 15 : Schéma des connecteurs des modules RFID

2.9 LEDs d'interface

Trois LEDs extérieurs ont pour but d'informer l'utilisateur sur l'état du système. Notamment pour avertir sur l'extinction du système, si une communication Wi-Fi ou Ethernet a lieu et si le badge RFID a été accepté ou non.

Des LEDs RGB sont utilisées avec des drivers dédiés. Des LEDs bicolores étaient prévues, mais le choix s'est porté sur les LEDs RGB pour plus de possibilités sans différence de coût. Il s'agit d'une décision purement esthétique afin d'offrir plus de possibilités. Leur utilisation pourra être étudié et envisagé dans de futurs projets de l'ES.

LED d'interface RGB (x3)		Datasheet [16]
Fabricant	Würth Elektronik	
N° de fabricant	150141M173100	
Tension directe	R 2,0 - V 3,2 - B 3,2 [V]	
Courant maximal	30 [mA]	

Tableau 12 : Caractéristiques principales des LEDs RGB

Le driver LED comporte 3 canaux, adaptés au contrôle des LEDs RGB, réduisant la charge de courant au microcontrôleur. Le driver utilise la méthode « Grayscale », ajustant l'intensité de chaque couleur avec des signaux PWM pour créer une variété d'effets. La commande s'effectue via une seule ligne en série par un protocole propriétaire « EasySet ».

Une erreur de lecture de la datasheet a conduit à croire que la commande était basée sur des signaux PWM en entrée. La connexion a été établie avec une sortie OC du microcontrôleur, mais il aurait été préférable de la connecter à une sortie SPI pour simplifier le contrôle. **(Voir fiche de modification)**

Le courant de sortie pour chaque canal est fixé par une unique résistance externe. La valeur est déterminée par la formule suivante, fourni par la datasheet :

$$R_{IREF} (\text{k}\Omega) = \frac{V_{IREF} (\text{V})}{I_{OLC} (\text{mA})} \times 43.4$$

Équation 3 : Résistance de limitation de courant du TLC5973

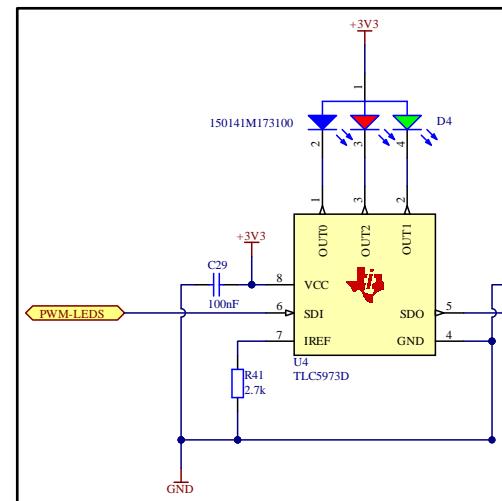


Figure 16 : Schéma des LEDs RGB d'interface

Driver de LED (x3)		Datasheet [17]
Fabricant	Texas Instrument	
N° de fabricant	TLC5973D	
Tension d'alimentation	3 ~ 6 [V]	
Tension par canal	21 [V]	
Courant par canal	50 [mA]	
Interface	3-Mbps « EasySet »	

Tableau 13 : Caractéristiques principales du driver de LED

Le protocole et la méthode de contrôle du circuit sont détaillés dans la partie software.

2.10 LEDs témoins

Deux LEDs sont placées sur le circuit, une indique la présence de l'alimentation, l'autre peut être contrôlée par le microcontrôleur à des fins de débogage. Elles ne sont pas visibles depuis l'extérieur du boîtier. Les LEDs ont été sélectionnées arbitrairement parmi le stock de l'ES. Leur courant et luminosité sont volontairement bas. Les résistances ont été calculés de la manière suivante :

$$R_{LED} = \frac{V_{CC} - U_{LED}}{I_{LED}} = \frac{3,3 - 2}{0,004} = 325 [\Omega] \rightarrow 330 [\Omega] E12$$

Équation 4 : Calcul de la résistance des LEDs témoins

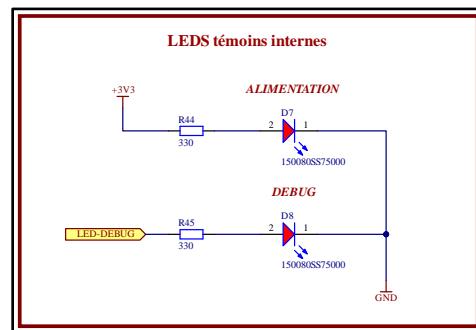


Figure 17 : Schéma LEDs témoins

2.11 Buzzer

Le buzzer a pour but d'informer l'utilisateur que la sortie 230 [VAC] va être désactivée et lui offrir la possibilité de prolonger la durée à l'aide de son badge.

Un buzzer magnétique a été choisi pour son volume sonore élevé par rapport à un buzzer piézoélectrique, malgré une consommation de courant plus élevée. L'objectif est de garantir que l'utilisateur puisse clairement entendre le signal pour éviter toute interruption inattendue de ses appareils. La consommation de courant est brève et aura finalement peu d'impact.

Une résistance placée en série permet de réduire le courant et le volume sonore. Sa valeur actuelle est de 0 [\Omega] pour expérimenter, ajustable si nécessaire.

Le buzzer est contrôlé par un signal PWM à travers un transistor. Pour éviter les effets indésirables de la commutation, une diode de roue libre est placée en parallèle. Les mêmes transistors et diodes que le relais sont choisis pour simplifier le nombre de références de composants en s'assurant qu'ils soient bien adaptés. La formule utilisée est la même (Équation 1).

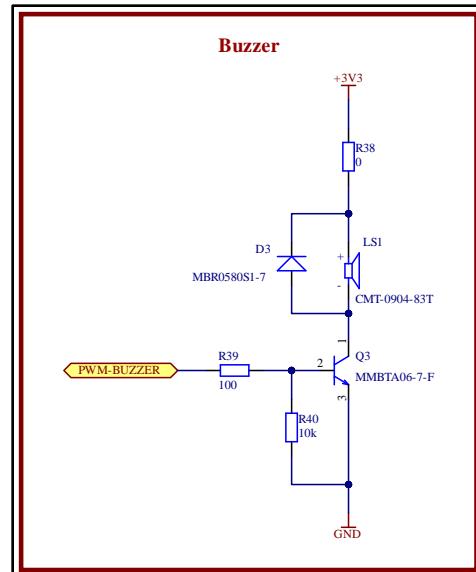


Figure 18 : Schéma du buzzer

Buzzer	Datasheet [18]
Fabricant	CUI Devices
N° de fabricant	CMT-0904-83T
Tension d'alimentation	2 ~ 5 [V]
Courant maximale	90 [mA]
Fréquence nominale	2730 [Hz]

Tableau 14 : Caractéristiques principales du buzzer

2.12 Points de mesure

Divers header répartis sur la carte permettent la mesure de tous les signaux de communications, les PWMs et la plupart des GPIOs ainsi que l'alimentation et le débogage.

Points de mesure			
Composant	Protocole	Signal	Désignateur
ESP32	UART	TX	P8
		RX	
RFID Chilli	UART	TX	P6
		RX	
mikroBUS	UART	TX	P7
		RX	
	I2C	SCL	P7
		SDA	
	SPI	SCK	P10
		MISO	
		MOSI	
		CS	
	GPIO	AN	P9
		PWM	
		INT	
		RST	
LEDs RGB	PWM	OC1	P5
Buzzer	PWM	OC5	
PIC32	GPIO	DEBUG	TP2
AC/DC	-	+3V3	TP1
		(Courant)	P4
		GND	TP3, TP4

Tableau 15 : Liste des points de mesure du système

Il est recommandé de ne placer aucun point de test sur la partie concernant l'Ethernet, car cela présente peu d'intérêt au vu de la complexité des trames qui sont transmises et du risque d'influencer les signaux par le fait de les déporter sur le PCB.

2.13 Boitier

Il était initialement envisagé dans le CDC de se procurer un boitier disponible sur le marché. Après réflexions et négociations avec le mandant du projet, il a été décidé de réaliser un boitier imprimé en 3D. Cette méthode offre l'avantage de réaliser une conception réalisée entièrement sur mesure sans nécessité d'usinage. La charge de travail supplémentaire entraîné par la modélisation 3D sera en partie compensé par le gain de temps sur l'usinage d'un boitier standard.

Le matériau utilisé devra offrir une isolation et une résistance à la chaleur suffisamment élevée. Pour cela, des matériaux comme l'ABS ou le PETG semblent les plus adaptés. Le matériel le plus couramment utilisé dans l'impression 3D qu'est le PLA ne convient pas pour cette application ! En effet, sa tendance à absorber l'humidité et sa faible résistance à la chaleur représente un risque non négligeable lors de l'utilisation de hautes tensions et de courants élevés. (Expérience personnelle)

Des détails supplémentaires sur le boitier sont fournis dans la partie hardware.

2.14 Serveur externe

Un serveur externe est essentiel pour prendre en charge deux fonctionnalités principales : la configuration à distance et la gestion d'une base de données.

Cette base de données permet de stocker les informations concernant les badges d'accès et les autorisations qui leur sont associées. Un serveur web basique facilite l'accès à distance afin de configurer le système et administrer les droits d'accès. La base de donnée pourra être gérée en SQL ou par simple écriture dans un fichier csv.

Dans cette optique, l'utilisation du Raspberry Pi 3B+ se montre particulièrement appropriée. Le langage de programmation privilégié, Python, bénéficie d'un grand nombre de bibliothèques et d'une documentation abondante. La faible consommation de courant en fait une option idéale, car il restera constamment allumé.

L'ES dispose de stocks suffisants compte tenu de la pénurie actuelle



Figure 19 : Image du Raspberry Pi 3B+

3 Hardware

Ce chapitre explore la conception du PCB et la modélisation du boîtier qui l'abrite. Il détaille les raisons derrière les choix de conception et les contraintes de fabrication.

Les fichiers de fabrication sont disponibles en annexe (14.6).

3.1 Vues des couches du PCB

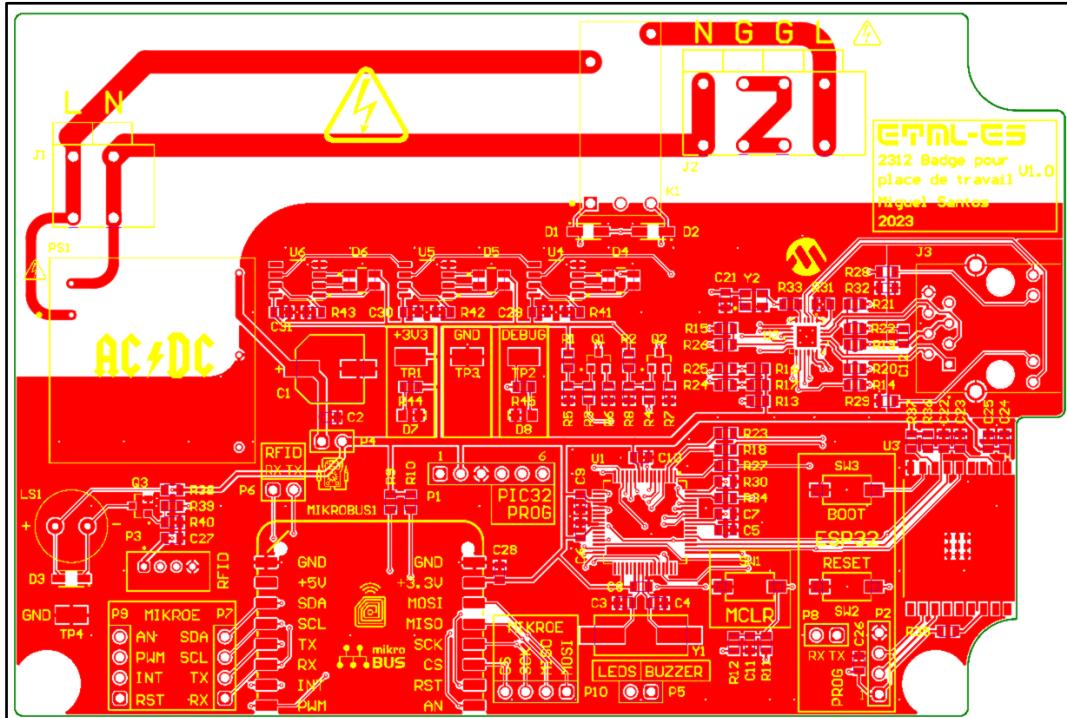


Figure 20 : Vue de la couche TOP

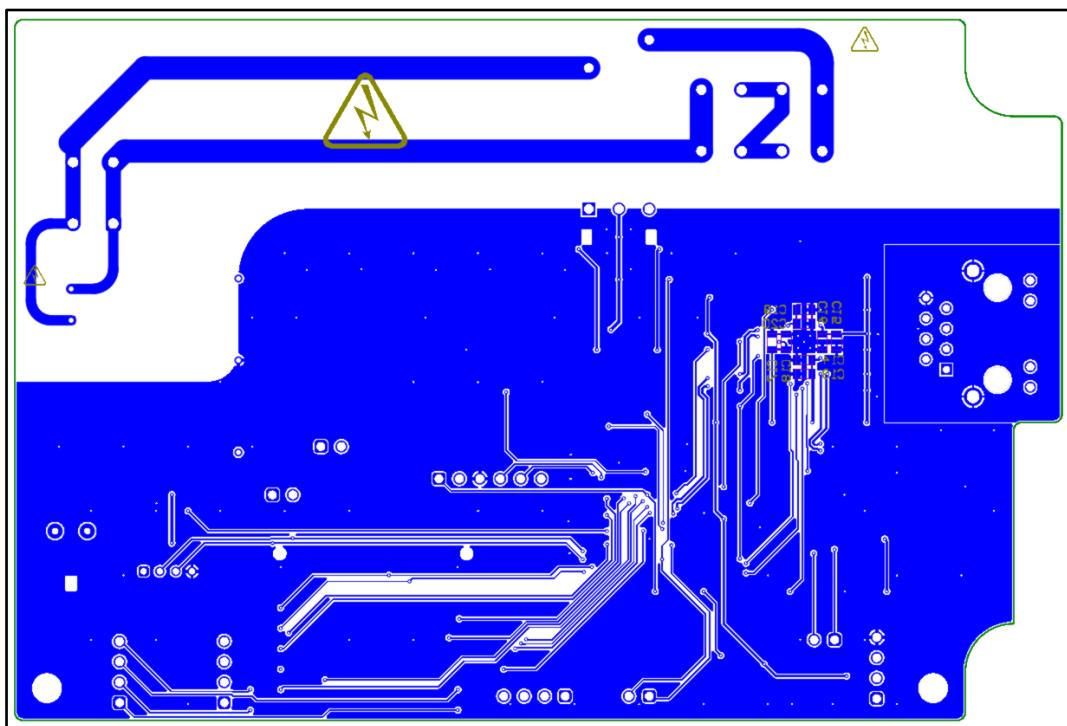


Figure 21 : Vue de la couche BOTTOM

3.2 Vues réalistes du PCB

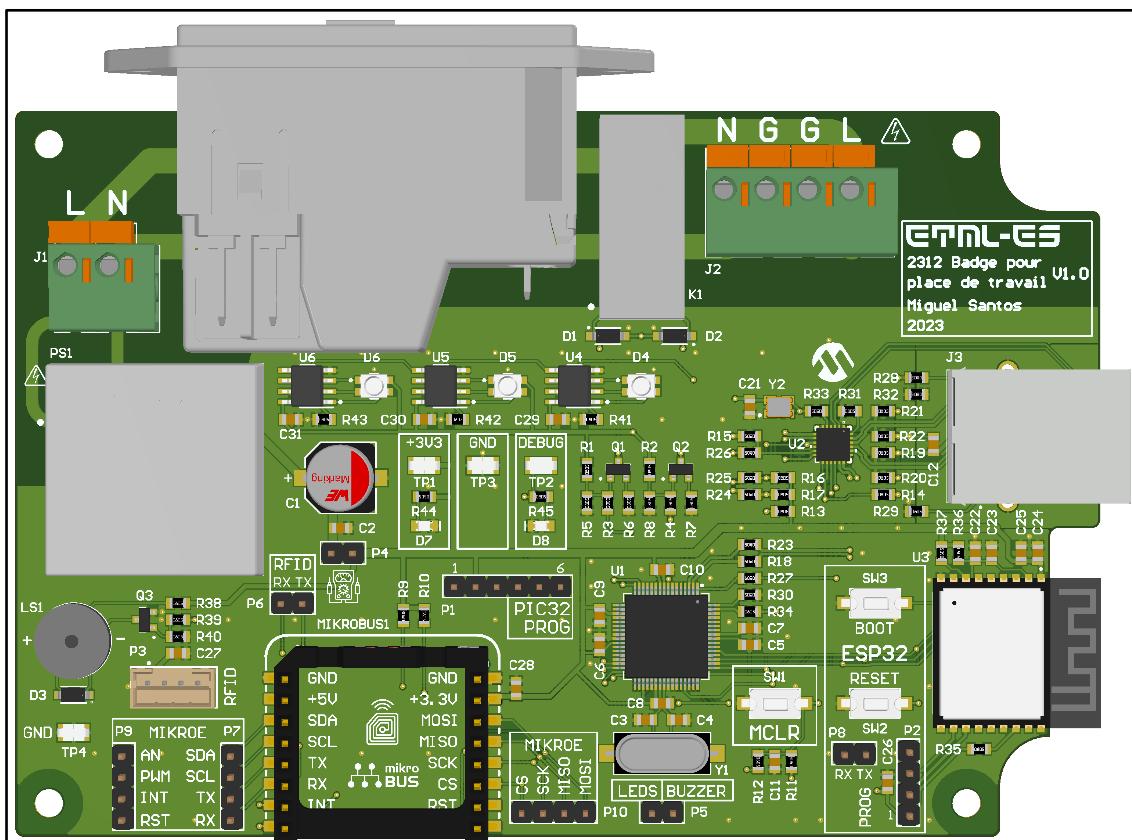


Figure 22 : Vue du dessus du PCB

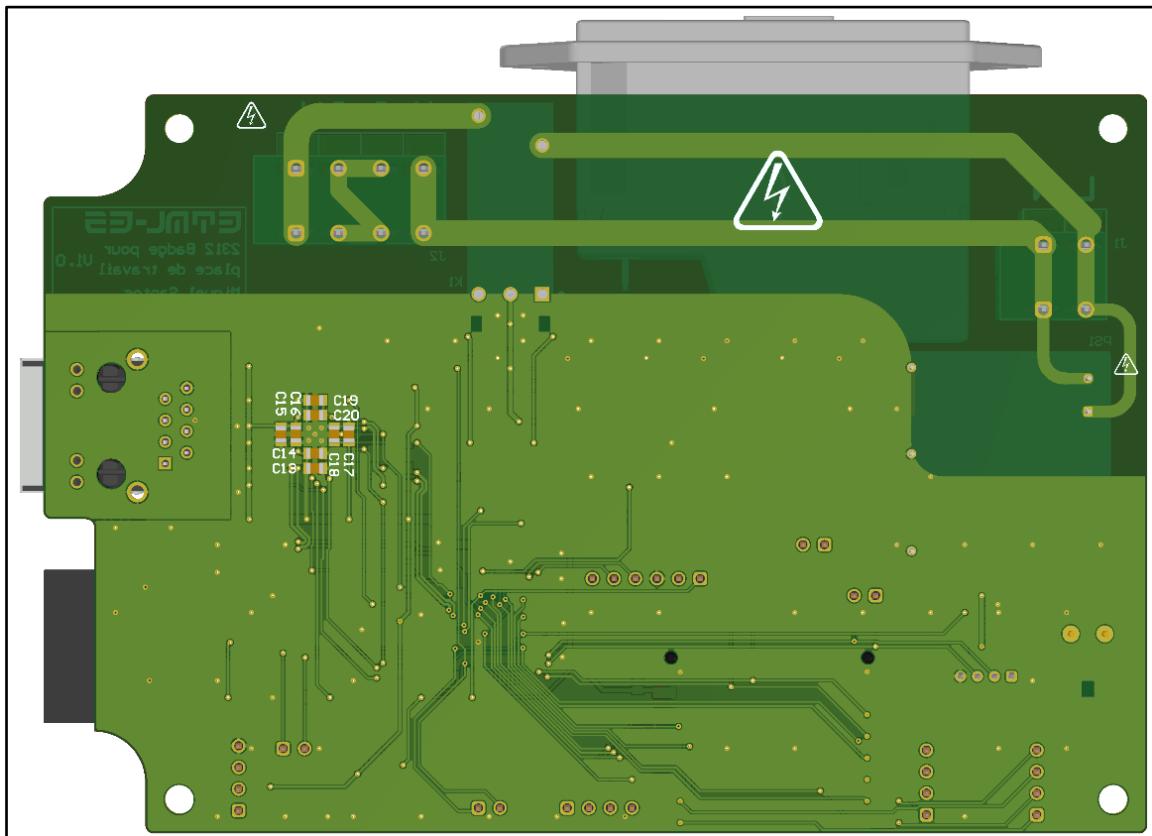


Figure 23 : Vue du dessous du PCB

3.3 Spécifications du PCB

Le PCB a été fabriqué par Eurocircuit, entreprise avec laquelle l'ES a l'habitude de collaborer. Voici ci-dessous (Tableau 16) les caractéristiques clés de ce PCB. Les paramètres à définir lors de la commande étaient fixé par l'ES. Le PCB a été commandé en collaboration avec d'autres étudiants sous la forme d'un panel afin de réduire les coûts.

Eurocircuit : « PCB proto »		
Matériel	FR-4 amélioré	
Nombre de couches	2	
Epaisseur de cuivre	35	[µm]
Longueur	130	[mm]
Largeur	87	[mm]
Epaisseur	1,55	[mm]

Tableau 16 : Spécifications principales du PCB

3.4 Règles de fabrication

Des règles de fabrication (Tableau 17) ont été intégrées dans le logiciel de CAO (Altium Designer) pour répondre aux normes de fabrication minimales de la catégorie « 6C » d'Eurocircuit [19].

Cette classe est choisie en raison de la préférence de l'ES avec ses spécifications. Elle représente le seuil idéal avant une augmentation notable des coûts de fabrication liés au PCB tout en offrant suffisamment de précision pour la plupart des circuits.

Eurocircuit : Classe 6C		
Largeur de piste	0.15	[mm]
Distances pistes et pads	0.15	[mm]
Largeur de pad	0.125	[mm]
Diamètre des perçages	0.35	[mm]

Tableau 17 : Règles principales de fabrication Eurocircuit - Classe 6C

Une liste complète des règles utilisées est disponible en annexe (14.7).

3.5 Placement des composants

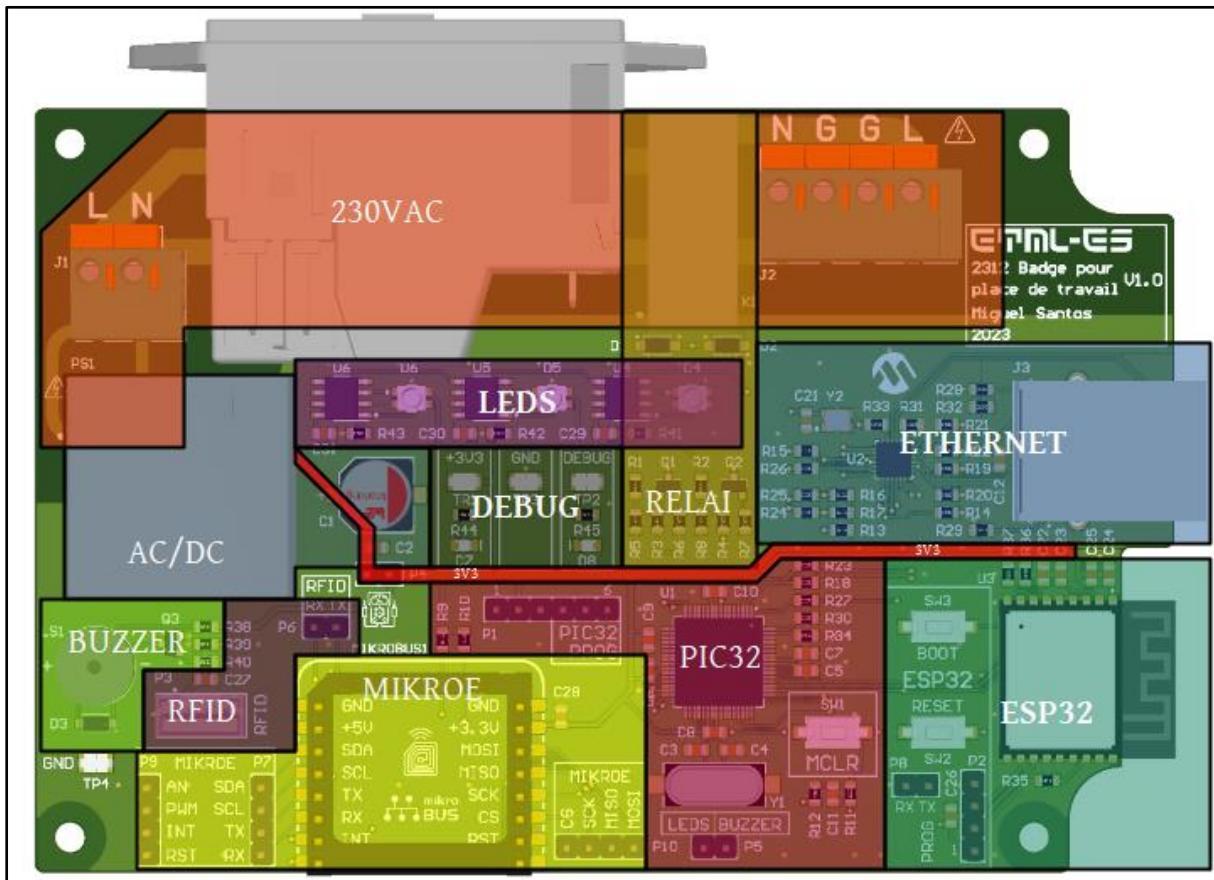


Figure 24 : Stratégie de placement des composants

Les composants liés au 230VAC ont été placés sur la partie supérieure, qui sera située vers l'arrière du boîtier. Cela permet de faciliter les connexions et de réduire les risques liés à la partie 230VAC.

Le convertisseur AC/DC a été placé le plus possible à distance des antennes (RFID et Wi-Fi) pour réduire le risque d'interférences, qui sont courantes avec ce type de composant.

Le PIC32 a été positionné au centre autant que possible au plus près de l'Ethernet et de l'ESP32, car ils nécessitent des vitesses de communication plus élevées.

Le connecteur RJ45 dépasse légèrement du bord du PCB pour être accessible depuis l'extérieur du boîtier par une ouverture dédiée. Le contrôleur PHY a été placé au plus proche du connecteur pour réduire la longueur des pistes différentielles.

L'empreinte « mikroBUS » a été placée à l'avant du boîtier pour permettre à une éventuelle antenne d'un module Mikroe de dépasser et d'être accessible de l'extérieur.

Le connecteur RFID a été placé librement car le module sera déporté sur le dessus du boîtier.

Le relai est placé sur le chemin entre l'entrée et la sortie 230VAC. Son contrôle est partiellement superposé aux LEDs en raison de contraintes physiques de placement.

Les LEDs ont été alignées horizontalement au centre du boîtier. Leur position verticale est influencée par la présence du module RFID sur le dessus et de la partie 230VAC. Elles seront déportées vers le haut avec des guides lumineux pour être visible depuis l'extérieur du boîtier.

Le Buzzer a été placé en fonction de l'espace disponible.

3.6 Largeurs de pistes

Une attention particulière est nécessaire pour les pistes de puissance. Cela implique notamment de garantir des largeurs de pistes suffisantes pour le passage du courant. Celles-ci ont été déterminées à l'aide du calculateur de DigiKey (Tableau 18).

Largeurs des pistes de puissance					Calculateur [20]
Net	Courant	Elévation de température	Largeur minimale	Largeur sélectionnée	
AC-L-IN AC-L-OUT AC-N EARTH	10 [A]	10 [°C]	7,19 [mm]	(8,00) [mm]	
	5,0 [A]	10 [°C]	2,77 [mm]	3,00 [mm]	
+3V3	1,6 [A]	10 [°C]	0,57 [mm]	1,00 [mm]	

Tableau 18 : Largeurs des pistes de puissance

3.7 Pistes d'alimentation +3V3

La piste d'alimentation +3V3 est tracée sous forme d'un réseau arborescent au centre du PCB, se subdivisant en branches connectées aux différents composants. Cette approche permet une distribution efficace de l'alimentation électrique tout en évitant la formation de boucles d'alimentation indésirables, susceptibles de perturber le circuit.

La piste principale est dimensionnée selon les spécifications définies ci-dessus (Tableau 18) afin de garantir le courant maximal que peut fournir le convertisseur AC/DC. Les branches connectées aux composants, moins exigeantes individuellement en courant, sont réduites en taille pour faciliter la connexion aux broches.

3.8 Pistes de puissance 230 [VAC]

Les pistes destinées à véhiculer le 230 [VAC] ont été tracées au bord du PCB afin de minimiser les risques et les éventuelles perturbations avec les autres parties du circuit. Pour réduire la largeur nécessaire pour acheminer un courant de 10 A, la charge a été répartis sur les deux couches du PCB. Cette approche était nécessaire pour maintenir une distance d'isolation suffisante sans occuper une surface excessive. Par conséquent, les calculs ont été effectués en considérant un courant de 5 A.

La distance d'isolation minimale a été calculée en utilisant un outil en ligne qui se base sur la norme IPC-2221. Celle-ci spécifie les distances d'isolation requises pour les pistes d'un PCB. Une marge a été prise pour une sécurité supplémentaire. Cette distance est aussi garantie avec les vis de fixations. (Tableau 19)

Distance d'isolations des pistes 230 [VAC]			Calculateur [21]
Net	Tension crête	Distance minimale	Distance sélectionnée
AC-L-IN AC-L-OUT AC-N EARTH	325 [VAC]	2,49 [mm]	4,00 [mm]

Tableau 19 : Distance d'isolation 230 [VAC]

3.9 Plan de masse

Le PCB comporte un plan de masse GND sur les deux couches pour réduire la résistance du chemin de retour et prévenir les boucles de masse, minimisant ainsi les interférences.

Le plan a été isolé de la partie 230VAC pour prévenir les risques d'incidents.

Les connexions avec les composants suivent les règles Altium établis ci-dessous (Figure 25). Les pads des composants sont connectés avec un « thermal relief » pour faciliter le soudage

manuel. Cette méthode permet de focaliser la dissipation thermique principalement sur le pad plutôt que de la répartir dans tout le plan de masse. La largeur de connexion de 0,4mm sert à éviter de créer un goulot d'étranglement sur le chemin du retour. Les via, non soudés manuellement, sont directement connectés pour minimiser la résistance.

Le stitching permet d'assurer une connexion optimale entre les plans de masse situé sur les différentes couches. Cela permet de garantir le chemin de retour le plus court possible et de réduire le risque de créer des boucles de masse, diminuant ainsi le risque d'interférences.

L'espacement des via de stitching est influencé par divers paramètres, notamment les fréquences de fonctionnement du circuit et les contraintes de fabrication. Cependant, le calcul précis de cet espacement peut s'avérer complexe. Pour simplifier, une valeur arbitraire de 8 mm a été choisie (Figure 26).

Cette valeur semble offrir un compromis satisfaisant entre un espacement suffisamment large pour éviter des problèmes potentiels tout en garantissant une connexion optimale entre les plans de masse.

3.10 Oscillateurs externes

Il faut veiller à ce qu'aucune piste ne traverse les zones sous les oscillateurs externes. Cette mesure est cruciale à la fois pour assurer le fonctionnement optimal des oscillateurs et pour minimiser le risque d'interférences qui pourraient perturber d'autres signaux du circuit.

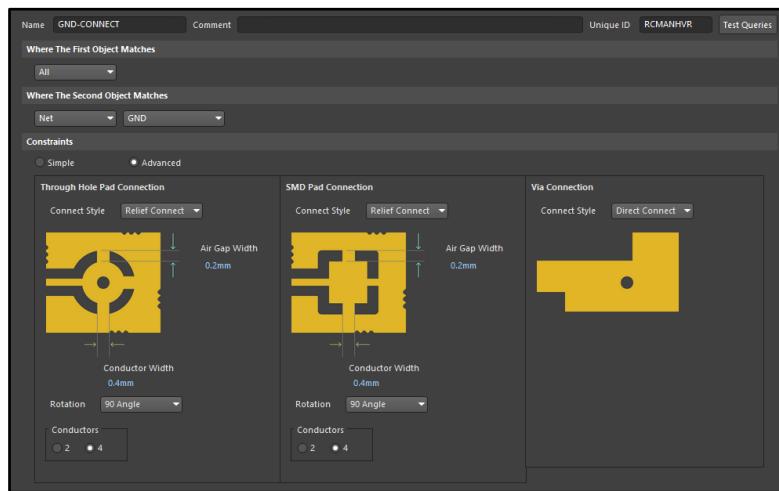


Figure 25 : Règle de connexion au plan de masse

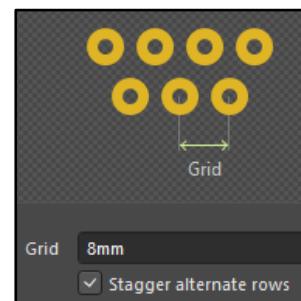


Figure 26 : Règle de stitching du plan de masse

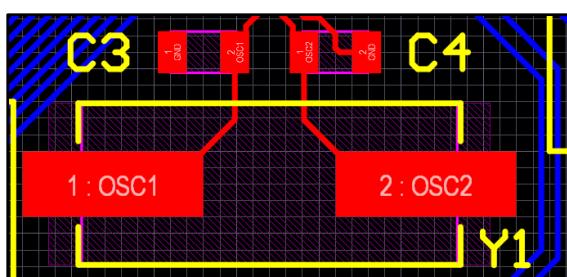


Figure 28 : Vue de l'oscillateur du PIC32

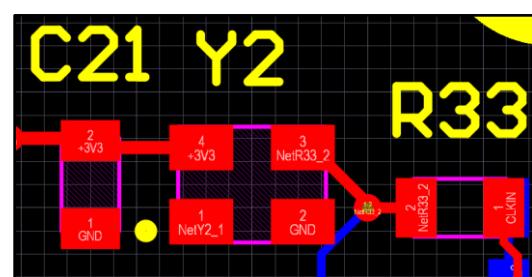


Figure 27 : Vue de l'oscillateur du contrôleur PHY

3.11 Ethernet

Plusieurs signaux de la partie concernant l'Ethernet nécessitent l'utilisation de paires différentielles.

Il est recommandé d'éviter de croiser les paires différentielles, malheureusement cette situation n'a pas pu être évité. Les résistances d'adaptation d'impédance doivent être placé au plus proche du contrôleur PHY.

Le stack manager d'Altium est utilisé pour garantir l'impédance des paires différentielles sur le PCB. Il permet de créer des profils d'impédances en fixant l'espacement des pistes, et Altium calcule automatiquement l'autre valeur nécessaire en fonction de l'impédance qui a été fixée.

L'écartement minimal fixé par les règles de fabrication a été choisi. La valeur de diélectrique a peu d'importance dans notre cas, car le PCB d'Eurocircuit n'assure pas de valeur précise et celle-ci a finalement peu d'impact sur le calcul.

Les normes des protocoles RMII et MDI nécessitent une impédance caractéristique de 100Ω , avec une tolérance de $+/- 10\%$, ce qui est respecté ::

D100 (100OhmDiff)					
#	Name	Dk	Width (W1)	Trace Gap (G)	Impedance (Zdiff)
	Top Overlay				
	Top Solder	3.5			
1	Top Layer		<input checked="" type="checkbox"/> 0.3mm	0.15mm	103.04
	Dielectric1	4.8			
2	Bottom Layer		<input checked="" type="checkbox"/> 0.3mm	0.15mm	103.04
	Bottom Solder	3.5			
	Bottom Overlay				

Figure 30 : Impédance des paires différentielles de l'Ethernet

Conformément à ce qui est indiqué sur la datasheet, un plan de masse isolé a été réalisé autour du connecteur RJ45 (Figure 31). Celui-ci a pour but de réduire le bruit des interférences. Il est relié au plan de masse GND par le biais d'une résistance de 0Ω . La valeur est à adapter de manière empirique en fonction des interférences rencontrées.

Pour permettre le positionnement des composants au plus proche du contrôleur PHY, les condensateurs de découplage ont été placé sur la couche BOTTOM.

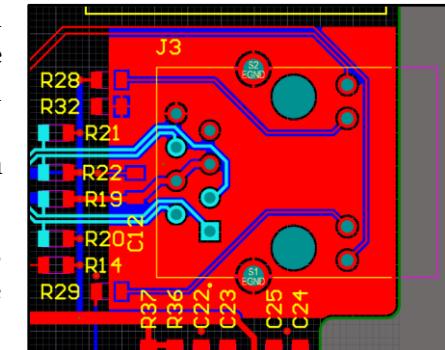


Figure 31 : Plan de masse isolé du connecteur RJ45

3.12 ESP32

Le fabricant du module ESP32 fournit un document détaillant les directives de conception précises [22]. Le positionnement optimal du module ESP32 dépend de l'emplacement de l'antenne, qui est située du côté droit dans le cas de l'ESP32-C3. Par conséquent, le module a été disposé sur le PCB conformément à ces recommandations afin de garantir ses performances (Figure 32).

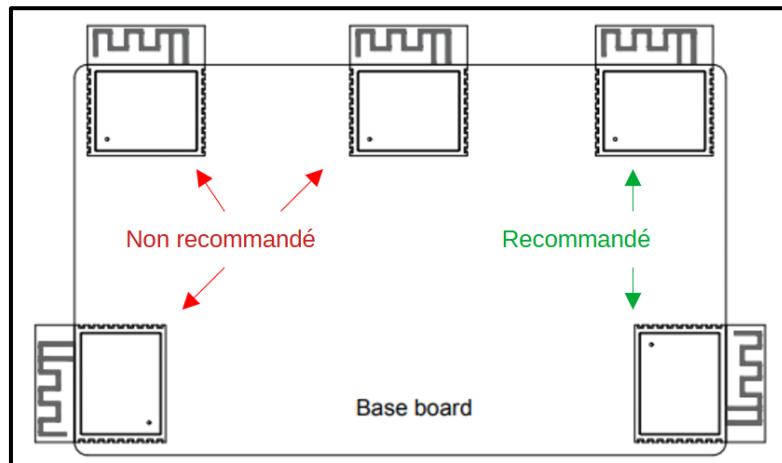


Figure 32 : Recommandations de placement de l'ESP32

Si l'installation du module sur le bord du circuit n'est pas possible, il est essentiel de veiller à ce qu'il n'y ait pas de cuivre autour et sous l'antenne, au minimum. Cependant, le fabricant recommande toujours d'effectuer une découpe du PCB si cela est réalisable. Dans ce cas précis, une découpe selon les dimensions spécifiées par le fabricant (Figure 33). Cela a permis de réduire légèrement la taille maximale du PCB et de rapprocher l'ESP32 du PIC32.

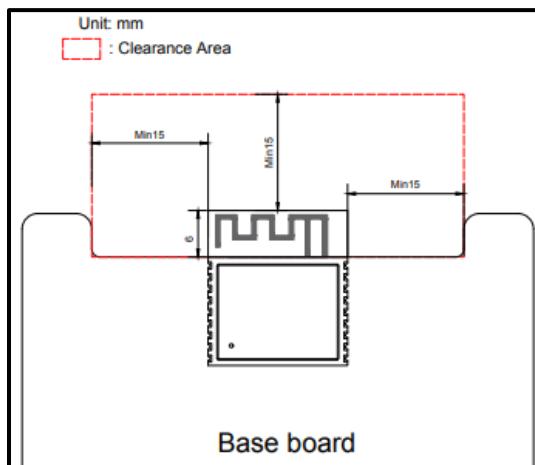


Figure 33 : Recommandation de dégagement pour l'antenne de l'ESP32

3.13 Boîtier

Malheureusement le boîtier n'a pas été réalisé en l'état du projet. Celui-ci est prévu pour être réalisé au plus vite, idéalement pour les portes ouvertes de l'ES.

4 Firmware

La partie firmware décrit la programmation réalisée au niveau du PCB. Elle répertorie les principales informations des différentes librairies.

Un listing complet du code réalisé est disponible en annexe (14.8).

4.1 Approche utilisée

La gestion du système repose sur l'utilisation de plusieurs machines à états fonctionnant simultanément. Cette approche garantit une utilisation efficace des ressources en évitant les blocages de fonctions. Ainsi, il est possible d'accomplir des tâches telles que la lecture d'un son sur le buzzer, l'activation d'une LED, et l'établissement d'une communication UART en parallèle, sans qu'aucune de ces actions n'interfère avec les autres.

4.2 Machine d'état global

La machine d'état global est gérée par la librairie « APP ». C'est elle qui réalise l'envoi des commandes et appels de fonctions nécessaires au systèmes.

Malheureusement celle-ci n'a pas encore été implémenté en l'état actuelle. La dernière phase du projet s'est concentré sur le débogage des erreurs des autres librairies.

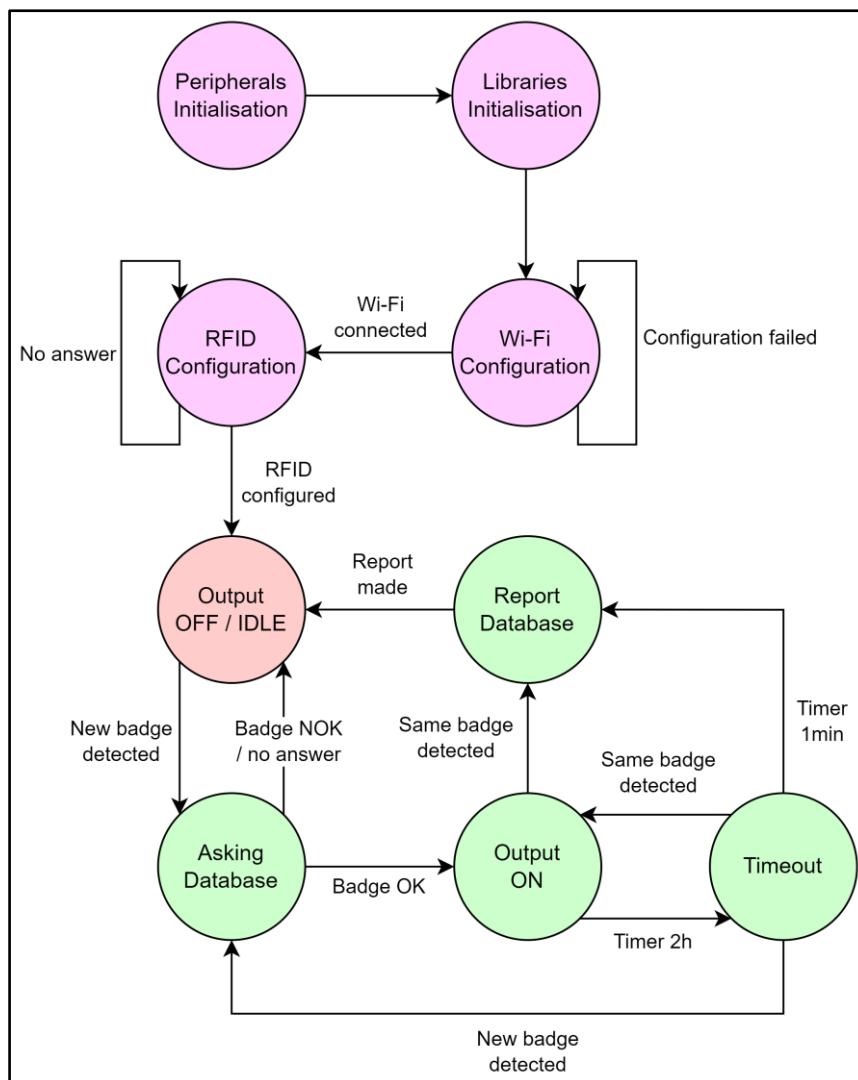


Figure 34 : Machine d'état globale du firmware

4.3 Librairies utilisées

Librairies firmware	
Nom	Description
app	Gestion de la machine d'état global
esp	Interfaçage par machine d'état de la communication avec l'ESP32
chu	Interfaçage par machine d'état de la communication avec le Chilli UART B1
bzr	Gestion par machine d'état des séquences du buzzer
fifo	Gestion de la fifo software utilisé par les UART
counter	Gestion des compteurs de délais non bloquant
SerialTimer	Emulation d'une communication série sur une sortie GPIO avec interruption de timer
TLC5973	Fonctions de contrôle des drivers de LEDs
RFIDB1ClientInterface	Interface d'envoi de commandes pour communiquer avec le Chilli UART B1 (fourni par le fabricant)
RFIDB1ClientProtocol	Gestion des trames UART utilisé par les fonctions de RFIDB1ClientInterface (fourni par le fabricant)
ccittcrc	Calculs du CRC nécessaire au RFIDB1ClientProtocol (fourni par le fabricant)

Tableau 20 : Description des librairies firmware

Les librairies à trois lettres concernent les machines d'état du système. Chacune d'elle possède au minimum les caractéristiques suivantes (non répété dans les tableaux suivants) :

***.c / ***.h	
Types	Description
***_DATA	Structure des données associées à la librairie
***_STATES	Enumération de la machine d'état
Variables	Description
***Data	Variable stockant les données associées à la librairie
Fonctions	Description
***_Initialize	Initialisation de la machine d'état et des variables du fichier, appelée au démarrage du système
***_Tasks	Exécution de la machine d'état, appelée de manière cyclique

Tableau 21 : Composantes principales des machines d'états

Les spécificités propres à chaque machine d'état sont détaillées dans les parties suivantes.

4.4 Librairie : ESP

La librairie ESP a pour but de jouer le rôle d'interface dans la communication UART avec l'ESP32. L'application principale gère les ordres d'envoi et de réception des commandes pendant que la machine d'état établit les différentes communications en parallèle.

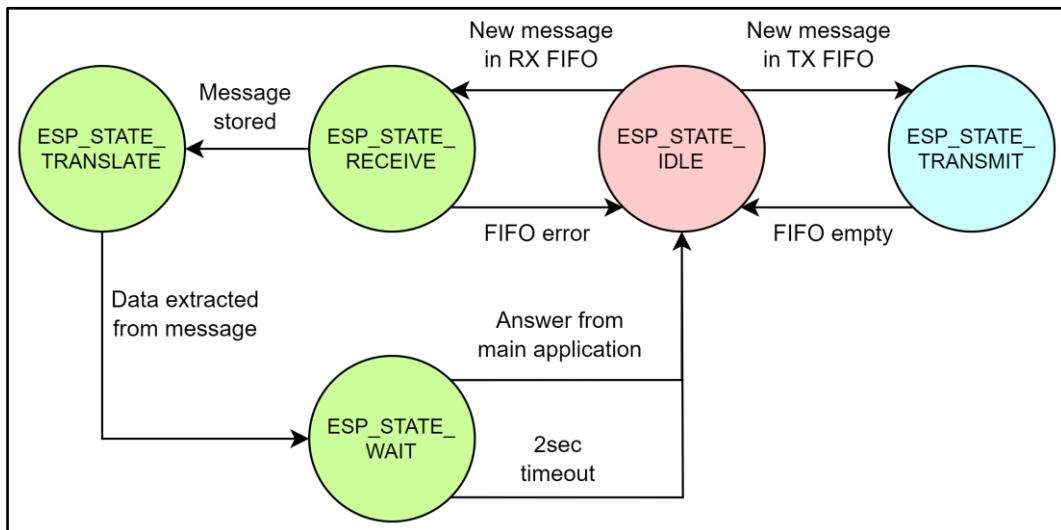


Figure 35 : Machine d'état - ESP

esp.c / esp.h	
Définitions	Description
ESP_USART_ID	ID de l'USART utilisé par l'ESP32
ESP_INT_SOURCE_*	Sources d'interruptions de l'USART utilisé par l'ESP32
AT_CMD_*	Les commandes AT utilisées sont définies sous forme de chaîne de caractères
Types	Description
S_AT_PACKET	Structure décrivant les différentes parties d'une commande AT
Fonctions	Description
ESP_SendCommand	Permet à l'application principale d'envoyer une commande AT en utilisant les définitions, gérée par la machine d'état

Tableau 22 : Spécificités de la librairie ESP

L'USART utilisé par la librairie a été configuré sous Harmony selon les spécifications fournies par le datasheet de l'ESP32.

Ces valeurs peuvent être configurées sur l'ESP32, mais les valeurs par défaut ont été choisies pour plus de facilité. Il s'agit d'un baudrate de 115200, avec 8 bits de données, sans bit de parité et un seul bit de stop. Aucun contrôle de flux, qu'il soit software ou hardware n'est utilisé. (Handshake mode et USART Lines enable)

La priorité de l'interruption a été réglée au niveau maximal car les communications UART ont été jugées comme la partie la plus importante du circuit.

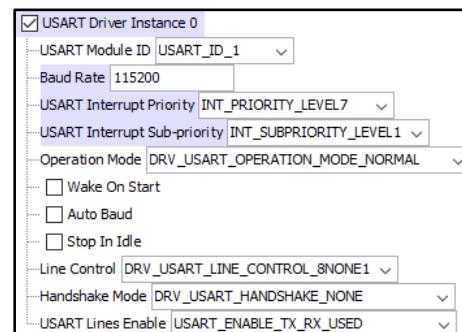


Figure 36 : Configuration harmony USART1

4.5 Librairie : CHU

La librairie CHU (Chilli UART B1) a été établie sur le même modèle que la librairie ESP. Elle gère la communication par UART pendant que l'application principale lui fournit les ordres de commande.

La gestion des trames nécessaires à l'envoi des commandes est gérée par la librairie fourni par le fabricant (RFIDB1Client...).

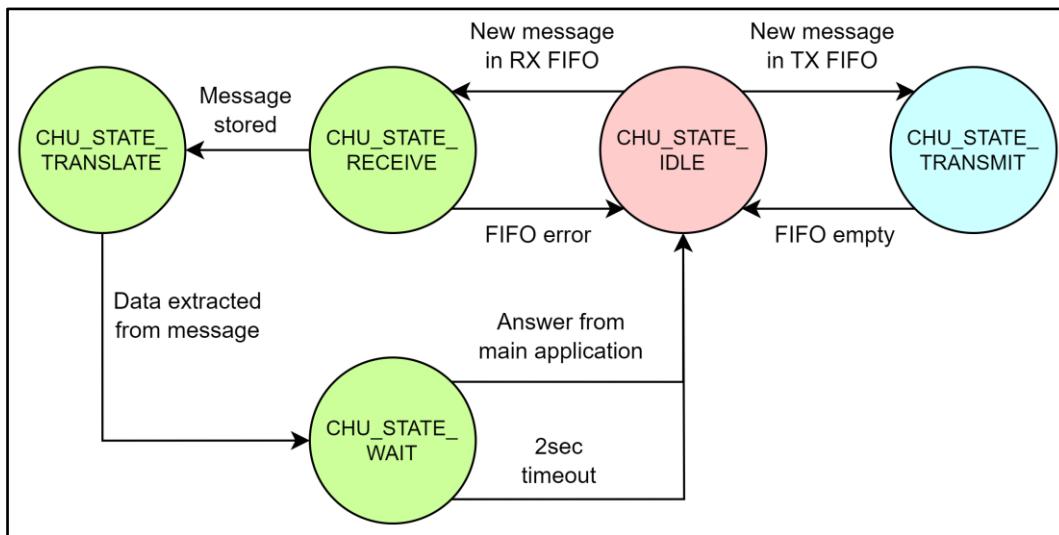


Figure 37 : Machine d'état - CHU

chu.c / chu.h	
Définitions	Description
CHU_USART_ID	ID de l'USART utilisé par le module RFID
CHU_INT_SOURCE_*	Sources d'interruptions de l'USART utilisé par le module RFID
Variables	
chuRfid_config	Variable du type de structure décrivant les paramètres de configuration, notamment les buffers d'entrée et de sortie
chuRfid_interface	Contient des pointeurs de fonctions pour l'interfaçage avec le module RFID
chuRfid_object	Contient les informations liés au module RFID, notamment les fonctions de callback utiliser par la librairie RFIDB1
Fonctions	
CHU_RFID_Polling	Activation du mode « polling » sur le module RFID.

Tableau 23 : Spécificités de la librairie CHU

L'USART utilisé par la librairie a été configuré sous Harmony selon les spécifications fournies par la datasheet du « RFID B1 User Manual ». [23]

Ces valeurs peuvent être modifiées à l'aide de commandes. Il s'agit d'un baudrate de 9600, avec 8 bits de données, sans bit de parité et un seul bit de stop. Aucun contrôle de flux, qu'il soit software ou hardware n'est utilisé. (Handshake mode et USART Lines enable)

La priorité de l'interruption a été réglée au niveau maximal car les communications USART ont été jugé comme la partie la plus importante du circuit.

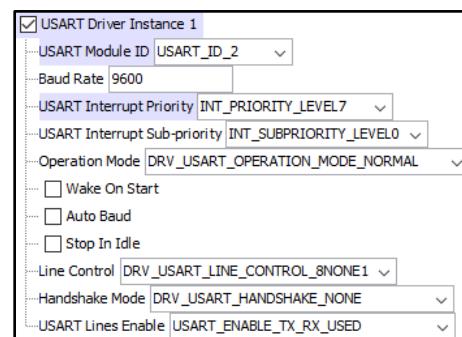


Figure 38 : Configuration Harmony USART2

4.6 Librairies : RFIDB1Client

Le fabricant fourni ses propres librairies pour interfaçer avec le module RFID. Celles-ci ont pour but de gérer la construction et l'envoi des trames.

Pour cela, il est nécessaire de déclarer trois types de variables qui seront utilisées pour cela.

RFIDB1ClientInterface.c / RFIDB1ClientInterface.h	
Types	Description
RFIDB1_InterfaceConfigurationT	Permet de configurer les buffer d'entrée et de sortie utilisées et d'indiquer les fonctions utilisées pour l'envoi et la réception des trames
RFIDB1_InterfaceT	Permet d'accéder aux fonctions de la librairie par le biais de pointeurs de fonctions.
RFIDB1_ObjectT	Contient les divers protocoles utilisées par la librairie.

Tableau 24 : Spécificités de la librairie RFIDB1ClientInterface

Il faut déclarer une variable de chaque type et réaliser les initialisations nécessaires à l'aide des pointeurs de fonctions fourni par l'interface.

La méthode de construction des trames transmises au module RFIDB1 sont détaillées dans le « RFID B1 User Manual ».

En résumé, chaque trame débute par un header, dont deux types sont possibles.

Seul le type A est utilisé dans ce projet. Celui-ci contient un byte de start et le nombre de données à transmettre, l'intégrité est vérifiée par un CRC concernant uniquement ces bytes.

Les données contiennent le numéro de commande et les paramètres nécessaires à la commande quand il y en a. Elles sont vérifiées à nouveau par un CRC appliquée uniquement sur les données.

La librairie offre des moyens d'encrypter les commandes envoyées. Cette possibilité n'a pas été étudié dans ce projet car la sécurité ne faisait pas partie des priorités du CDC.

Les réponses sont envoyées sous le même format, avec un header du type utilisé.

```
/* Config setup for RFIDB1 */
chuRfid_config.InputBuffer = chuData.fifoBuff_tx;
chuRfid_config.InputBufferSize = CHU_FIFO_SIZE;
chuRfid_config.OutputBuffer = chuData.fifoBuff_rx;
chuRfid_config.OutputBufferSize = CHU_FIFO_SIZE;
chuRfid_config.handleResponse = CHU_RFID_Response;
chuRfid_config.handleRequest = CHU_RFID_Request;

/* Initialise RFIDB1 objects */
GetRFIDB1Interface(&chuRfid_interface);
chuRfid_interface.Initialise(&chuRfid_object, &chuRfid_config);
chuRfid_interface.SetPacketHeaderType(&chuRfid_object, HeaderTypeA);
```

Figure 39 : Initialisation librairie RFIDB1

Packet with Type A Header							
Byte number	1	2	3	4	5	6	...
Value	0x02	0x00 - 0xFF	...				
Type	Start Of Text	Data Size (k) LSB	Data Size (k) MSB	Header CRC LSByte	Header CRC MSByte	Data (Plain or Encrypted)	

Figure 40 : RFIDB1 - Construction d'une trame de type A

Plain Command					
Byte number	1	2	...	N + 1	N + 2
Value	0x00 - 0x1E	0x00-0xFF	...	0x00 - 0xFF	0x00 - 0xFF
Type	Command Byte	Parameters Byte 1	...	Parameters Byte N	CRC LSByte

Figure 41 : RFIDB1 - Construction d'une commande non crypté

Plain Response					
Byte number	1	2	...	N + 1	N + 2
Value	0x00 - 0x02, 0xF0 - 0xF3	0x00-0xFF	...	0x00 - 0xFF	0x00 - 0xFF
Type	Response Byte	Parameters Byte 1	...	Parameters Byte N	CRC LSByte

Figure 42 : RFIDB1 - Construction d'une réponse du module

4.7 Librairie : BZR

La librairie BZR réalise le contrôle du buzzer à l'aide d'un signal PWM. L'application principale se charge ensuite d'appeler les séquences à jouer, implémentés sous forme d'un tableau, qui seront ensuite gérés par une machine d'état.

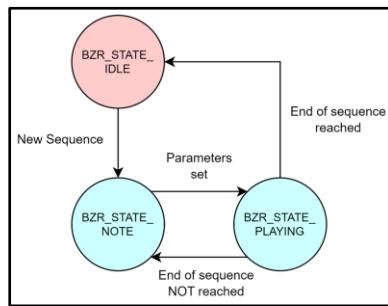


Figure 43 : Machine d'état - BZR

bzs.c / bzs.h	
Définitions	Description
BZR_TMR_ID	ID du timer utilisé par le buzzer
BZR_OC_ID	Sortie PWM utilisé par le buzzer
BZR_VOLUME	Permet de réduire le rapport cyclique et volume du buzzer
NOTE_***	Définitions de la fréquence de chaque note en Hz
Types	Description
E_BZR_SEQ	Enumération des séquences à jouer. Permet d'appeler les séquences en dehors du fichier.
S_BZR_SEQ	Structure décrivant les informations d'une séquence à jouer.
Variables	Description
BZR_SEQUENCE_*	Chaque séquence (musique) est représentée dans un tableau où la note et la durée de la note sont indiquées en alternance.
BZR_SEQUENCES	Tableau contenant les structures de chaque séquence, avec sa taille, son tempo et le pointeur de la séquence correspondante.
Fonctions	Description
BZR_PlaySequence	Permet de lancer une séquence musicale, gérée ensuite par la machine d'état

Tableau 25 : Spécificités de la librairie BZR

Un timer et une sortie OC, configurables sur Harmony, sont utilisés pour générer un signal PWM. Ces éléments sont exploités sans interruptions, car elles ne sont pas nécessaires. Le prescaler a été sélectionné de manière à permettre une génération précise de fréquences comprises entre 20 Hz et 20 kHz. La valeur du "Timer Period" permet de définir la fréquence du signal, tandis que la valeur du "OC Pulse Width" permet de définir le rapport cyclique du signal PWM.

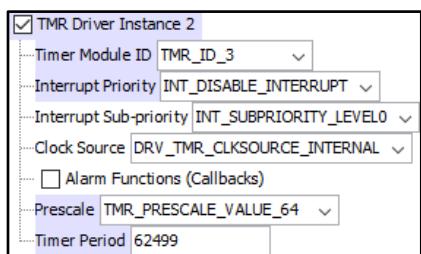


Figure 45 : Configuration Harmony - TMR3

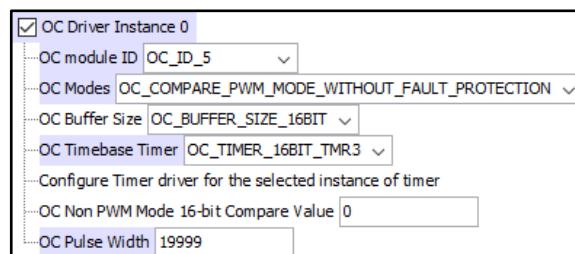


Figure 44 : Configuration Harmony - OC5

4.8 Librairie : TLC5973"

Le protocole de communication propriétaire "EasySet" est utilisé pour les drivers de LEDs TLC5973. Il permet d'envoyer des données sérielles en encodant les données binaires avec des séquences spécifiques, tout en permettant l'enchaînement des drivers. Les sorties sont des signaux PWM, ajustant l'intensité des LEDs en fonction de la valeur fournie au driver.

Chaque bit est encodé avec une impulsion à l'état haut, suivie d'une deuxième impulsion avant la moitié de la période du bit, indiquant ainsi un 1 logique ou l'absence de cette impulsion pour un 0 logique. Une séquence complète pour un driver comprend 48 bits, avec les 12 premiers bits servant de commande d'écriture. Ensuite, chaque sortie est encodée sur 12 bits pour spécifier le rapport cyclique.

Après chaque séquence complète, une pause précise (EOS) est marquée avant d'envoyer la séquence suivante. Une pause plus longue à la fin permet au driver de transférer les données dans ses registres internes (GSLAT).

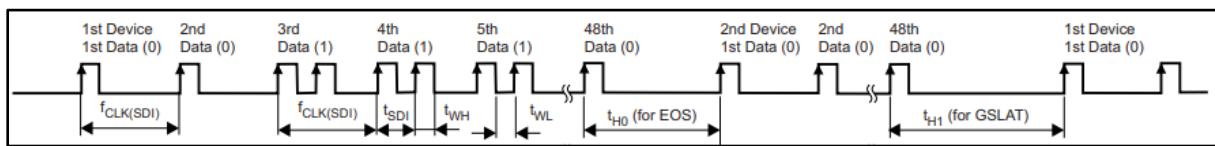


Figure 46 : Séquences de contrôle des LEDs

Des timings d'une certaine précision doivent être respectés, comme indiqué par la datasheet :

AC CHARACTERISTICS					
$f_{CLK(SDI)}$	Data transfer rate	SDI	100	3000	kHz
t_{SDI}	SDI input pulse duration	SDI	60	$0.5 / f_{CLK}$	ns
t_{WH}	Pulse duration, high	SDI	14		ns
t_{WL}	Pulse duration, low	SDI	14		ns
t_{H0}	Hold time: end of sequence (EOS)	$SDI \uparrow$ to $SDI \uparrow$	$3.5 / f_{CLK}$	$5.5 / f_{CLK}$	ns
t_{H1}	Hold time: data latch (GSLAT)	$SDI \uparrow$ to $SDI \uparrow$	$8 / f_{CLK}$		ns

Figure 47 : Timings à respecter - TLC5973

Les séquences peuvent être générées via un protocole série, de préférence SPI. Une erreur de conception avait conduit à utiliser une sortie PWM au lieu de l'encodage souhaité. Pour résoudre cela, une librairie spécifique a été créée (SerialTimer).

TLC5973.c / TLC5973.h	
Définitions	Description
DRIVER_COUNT	Défini le nombre de driver en chaîne
Types	Description
S_TLC_CHANNEL	Décrit la valeur de sortie d'un channel et son encodage
S_TLC_DRIVER	Structure stockant les 3 channels de sortie
Variables	Description
cyclesBuffer	Buffer de stockage des encodages des bits de chaque trame
tlcDrivers	Stockage des structures des drivers
Fonctions	Description
TLC_SetDriver	Fixe les valeurs de sortie d'un driver spécifique
TLC_SetAll	Fixe les mêmes valeurs de sortie pour tous les drivers
TLC_Transmit	Transmet les données des drivers par le port série

Tableau 26 : Spécificités de la librairie TLC5973

4.9 Librairie : SerialTimer

Cette librairie permet d'émuler une communication sérielle sur une sortie GPIO, en utilisant les interruptions d'un timer. Elle permet de contourner l'erreur citée précédemment.

Pour cela, il suffit de remplir son buffer et d'ensuite démarrer le timer. Lorsque toutes les données ont été envoyées, le timer s'arrête automatiquement.

Pour communiquer avec le TLC5973, l'encodage de chaque bit (cycle) est subdivisé en 5 impulsions. La fréquence d'interruption du timer doit donc être au minimum 5 fois supérieure à la fréquence minimale imposée par le TLC5973.

Pour cette raison, la fréquence du timer a été fixé à 500kHz. Au vu de cette rapidité, la fonction d'interruption doit faire preuve d'une grande efficacité. Néanmoins, elle ne sera active que pendant une très courte période de temps, ne devant pas être problématique pour le système.

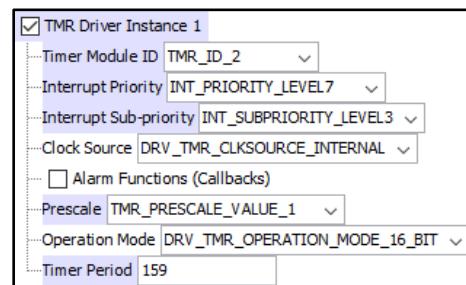


Figure 48 : Configuration Harmony - TMR2

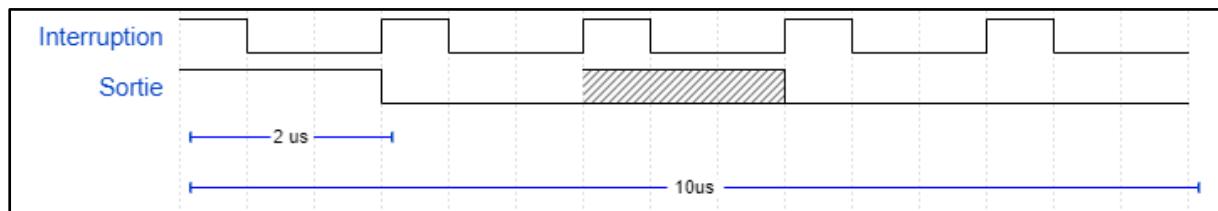


Figure 49 : Représentation temporel de l'encodage des bits

Sa priorité et sous-priorité ont été fixés aux valeurs maximales, dû à la nécessité de respecter des timings très précis. Elle ne semble pas perturber les communications UART, car les LEDs sont allumé uniquement après avoir communiqué avec les différents modules.

4.10 Calculs des timers

Les périodes des timers ont été déterminés par les formules suivantes. Premièrement il faut déterminer le prescaler minimum, sans quoi notre vitesse de comptage sera trop rapide et nous aurons atteint la valeur maximum avant d'avoir atteint la période du timer :

$$\text{Prescaler}_{min} = \frac{f_{SYS}}{2^{16} * f_{TMR}} = \frac{f_{SYS}}{2^{16}} * T_{TMR}$$

Équation 5 : Calcul du prescaler minimum

Cette valeur doit ensuite être arrondie au prescaler supérieur possible. Généralement il s'agit d'une puissance de 2, sauf pour le Timer 1 dont les valeurs peuvent être obtenues dans la datasheet ou sur Harmony.

La période du timer peut ensuite être calculée :

$$\text{Period}_{TMR} = \frac{f_{SYS}}{\text{prescaler} * f_{TMR}} = \frac{f_{SYS}}{\text{prescaler}} * T_{TMR}$$

Équation 6 : Calcul de la période d'un timer

5 Software

Malheureusement, suite à des retards sur le planning, la partie software concernant le Raspberry Pi n'a pas pu être implémentée.

Parmi les concepts à retenir, il est important de noter que le programme sera développé en Python. En ce qui concerne la gestion des données, la base de données sera construite en utilisant la bibliothèque "SQLite". Cette bibliothèque est particulièrement adaptée pour la gestion de bases de données de petite taille en langage SQL.



Figure 50 : Illustration - SQLite

Après avoir effectué quelques recherches, il est fréquemment suggéré dans de multiples sources d'utiliser la bibliothèque « socket » pour la gestion des communications réseau.

Il sera aussi nécessaire de déterminer la trame TCP/IP qui sera utilisée. Il s'agira notamment de pouvoir envoyer diverses commandes par le biais du réseau.

Il est prévu que tout cette partie soit développé d'ici à la défense orale.

6 Mesures

Les mesures visent à prouver le bon fonctionnement du système et à vérifier que le cahier des charges est respecté.

6.1 Matériel utilisé

Matériel de mesure		
Oscilloscope	Rhode&Schwarz	ES.SLO2.05.01.08
Multimètre	Gwinstek	ES.SLO2.00.00.90
Alimentation	Gwinstek	ES.SLO2.00.00.31
Analyseur logique	LogicSniffer	ES.SLO1.04.00.04

Tableau 27 : Liste du matériel de mesure

6.2 Convertisseur AC/DC

Mesure convertisseur AC / DC		
Signal	Point de test	Oscilloscope
+3V3	TP1	CH1

Tableau 28 : Points de tests – Mesure convertisseur AC/DC

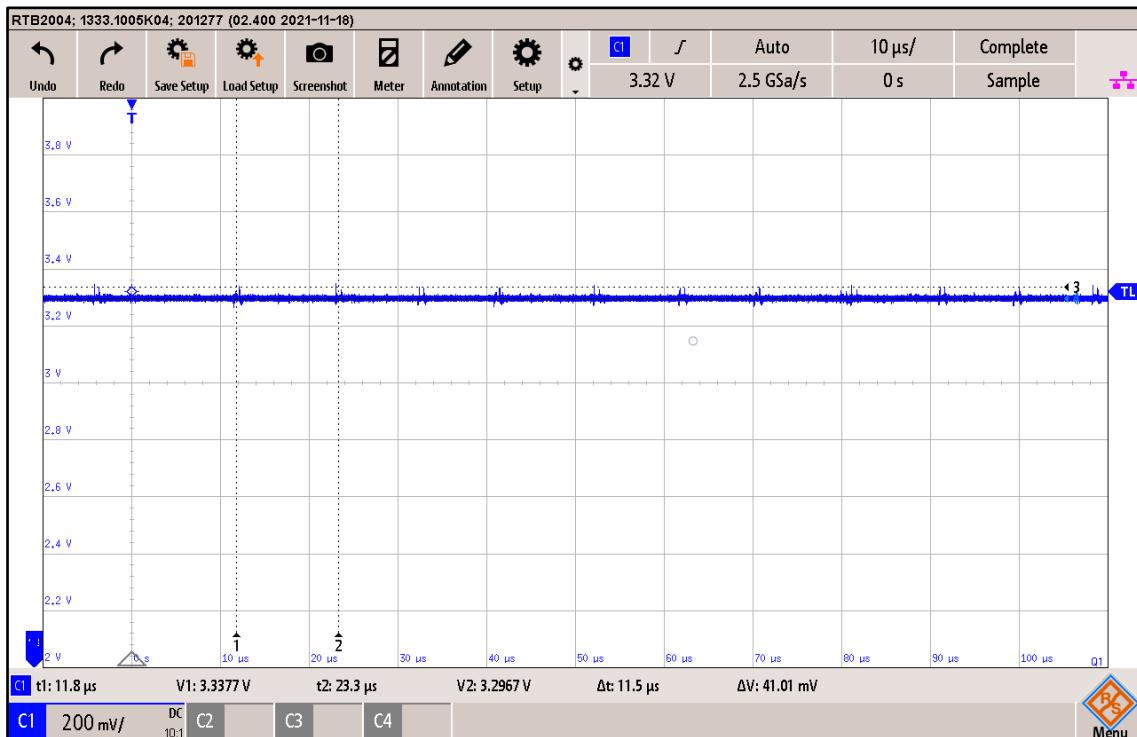


Figure 51 : Mesure du +3V3

On peut constater que le convertisseur AC/DC fournit une tension parfaitement régulée. On peut observer néanmoins que du bruit à ~85kHz est provoqué sur l'alimentation. Ceci est dû à la fréquence de fonctionnement du convertisseur AC/DC. Son amplitude est faible et a peu de risques de provoquer des nuisances au reste du circuit.

La mesure a été réalisée en fonctionnement avec une consommation de courant de ~300mA.

6.3 Commutation 230VAC

Mesure commutation 230VAC		
Signal	Point de test	Oscilloscope
AC-SET	K1 pin -6	CH1

Tableau 29 : Points de tests - Mesure commutation 230VAC

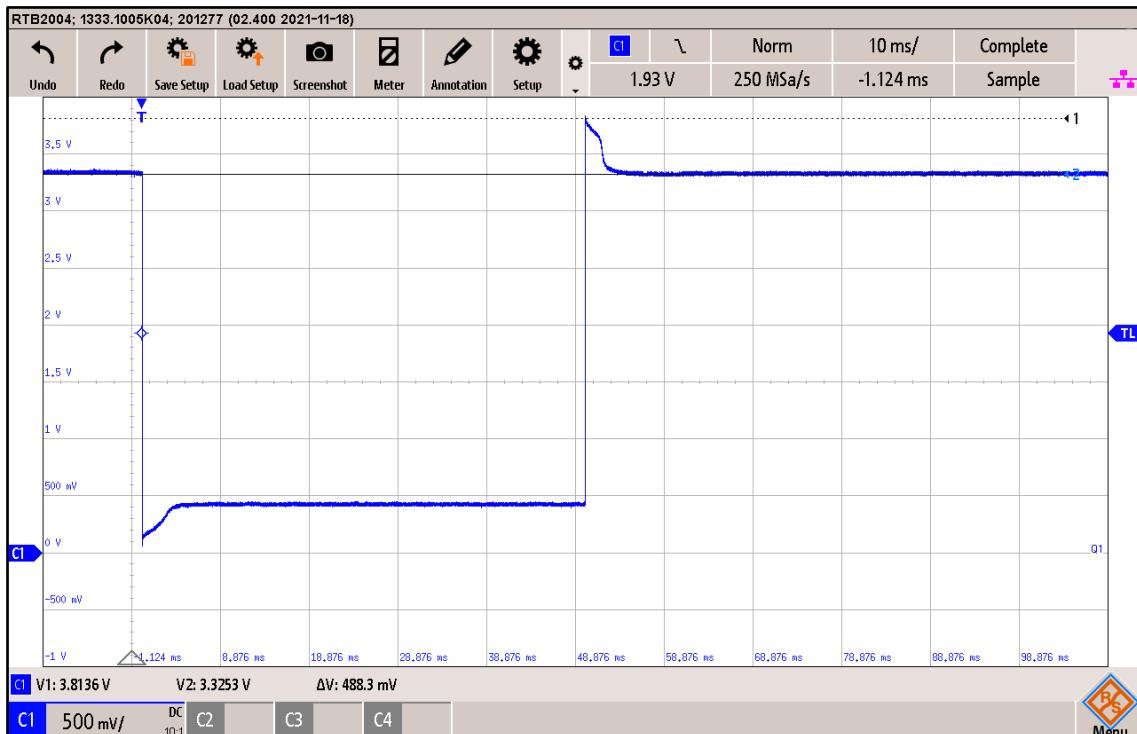


Figure 52 : Mesure de la commutation 230VAC sur le relai

La commutation de la sortie 230VAC est gérée de manière efficace par le microcontrôleur. Une vérification de la continuité a confirmé que la connexion entre la sortie et l'entrée était correcte.

La mesure révèle une tension inverse après la commutation. Cette tension est subie par le transistor qui est capable de la supporter sans problème. La bobine elle ne subit pas de surtension ce qui idéale pour conserver sa durée de vie.

Ce contrôle a été effectué à la fois pour le signal AC-SET et AC-RESET.

6.4 Module RFID

Mesure communication UART RFIDB1		
Signal	Point de test	Analyseur logique
U2TX	P6-1	Channel0
U2RX	P6-2	Channel1

Tableau 30 : Points de tests - Mesure UART RFID

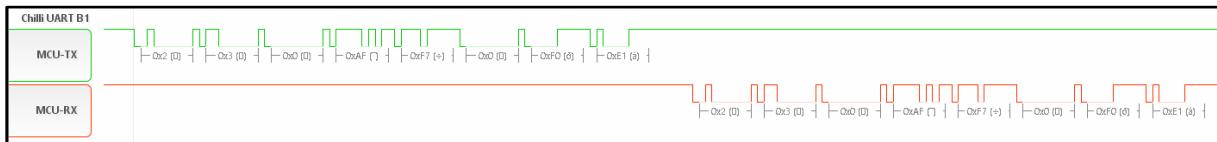


Figure 53 : Mesure trame UART RFID

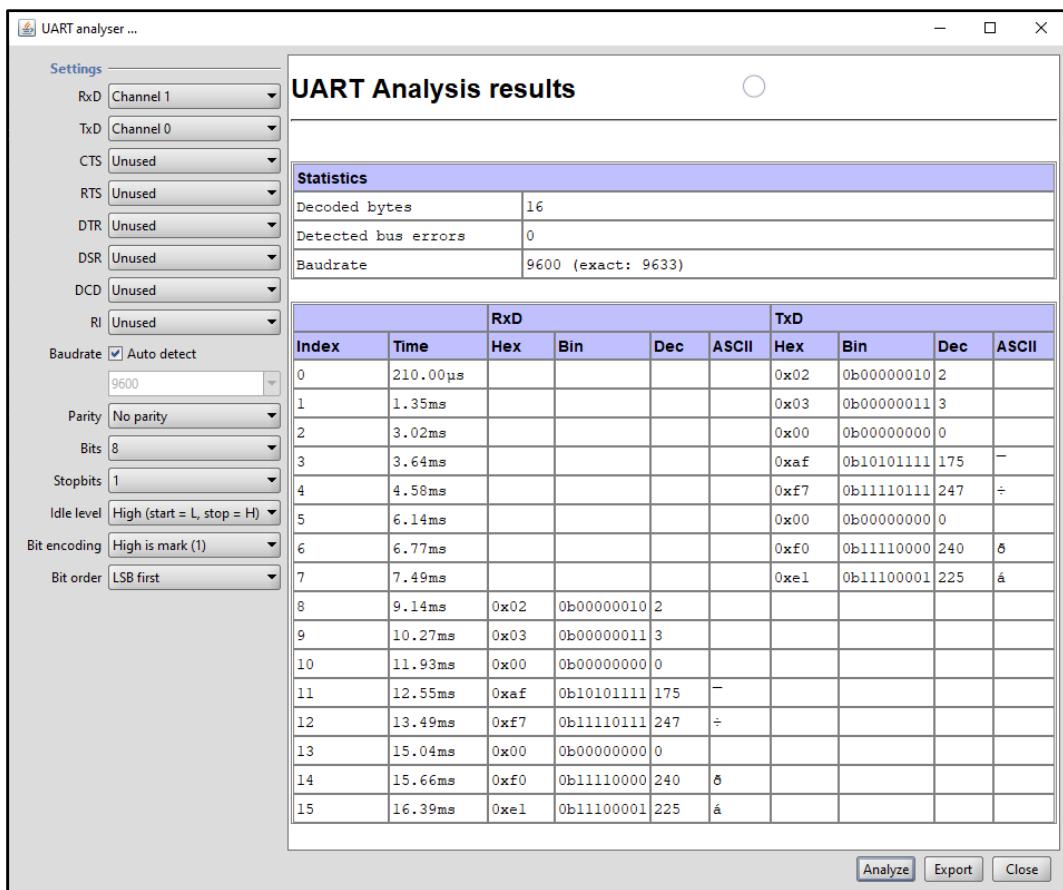


Figure 54 : Analyse trame UART RFID

Pour évaluer la communication avec le module RFID, une procédure simple appelée « Dummy Command » a été utilisée en transmettant les données par l'intermédiaire de l'UART en utilisant les bibliothèques fournies par le fabricant. Le module a répondu avec succès, confirmant ainsi son bon fonctionnement.

Des tests plus approfondis ont été réalisés en utilisant le mode de polling. Cependant, en raison de la complexité des trames, qui étaient nettement plus longues, il a été difficile d'intégrer ces résultats dans le rapport. Malgré cela, ces tests ont confirmé que l'envoi de commandes fonctionne efficacement. Le module réagit aux badges et transmet leurs identifiants via l'UART.

6.5 Module ESP32

Mesure communication UART ESP32		
Signal	Point de test	Analyseur logique
U1TX	P6-1	Channel0
U2RX	P6-2	Channel1

Tableau 31 : Points de tests - Mesure UART ESP32

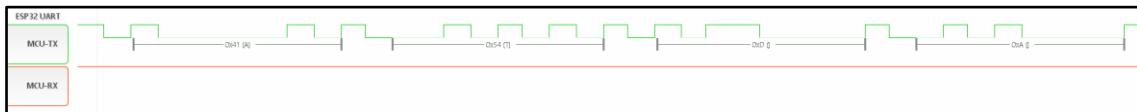


Figure 55 : Mesure trame UART TX ESP32

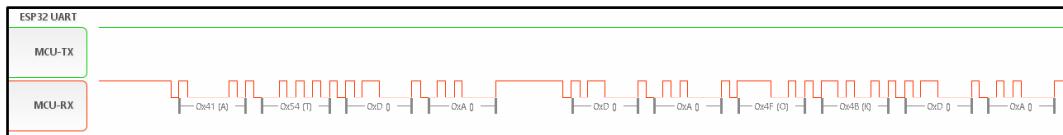


Figure 56 : Mesure trame UART RX ESP32

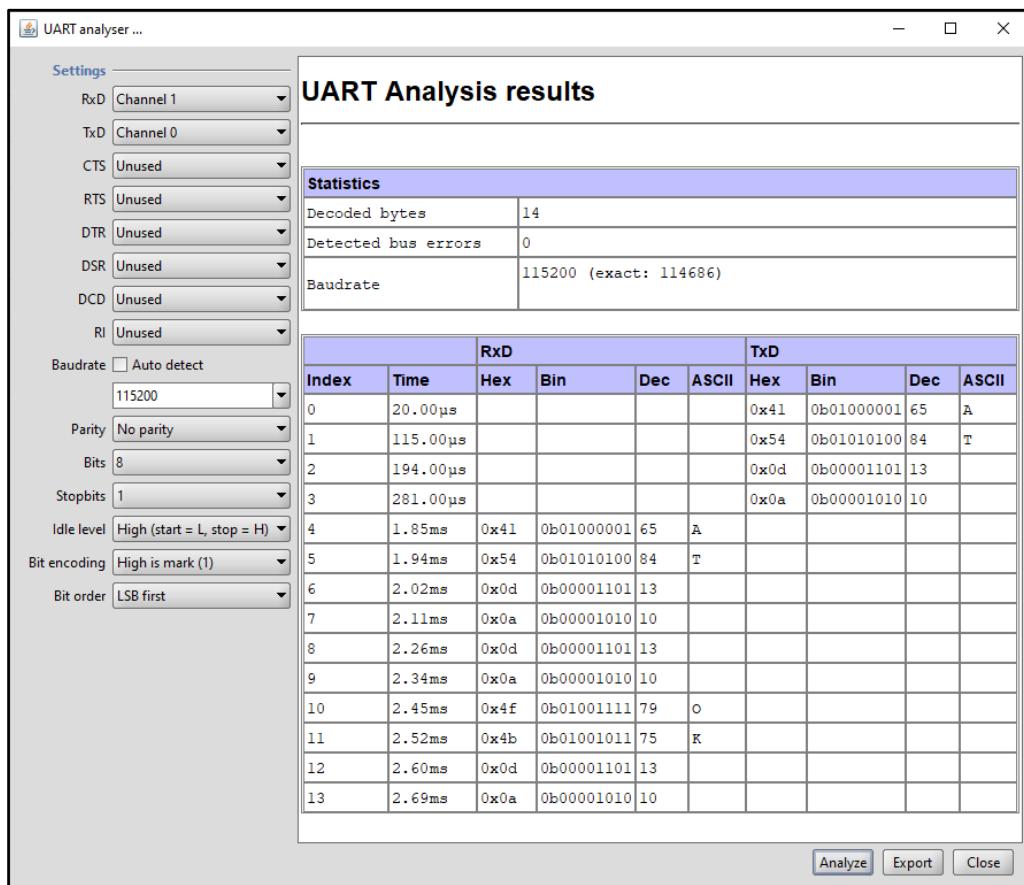


Figure 57 : Analyse trame UART ESP32

La même méthode que le RFID a été appliquée pour le module ESP32. Une simple commande « AT » a été transmise par UART. Le module fournit bien la réponse qui est attendue.

Une connexion Wi-Fi a aussi pu être établis à l'aide d'autres commandes.

Le module fonctionne correctement, aucun problème particulier n'a été rencontré.

6.6 TLC5973.

Mesure TLC5973		
Signal	Point de test	Oscilloscope
RD4	P5-2	CH1

Tableau 32 : Points de tests - Mesure TLC5973

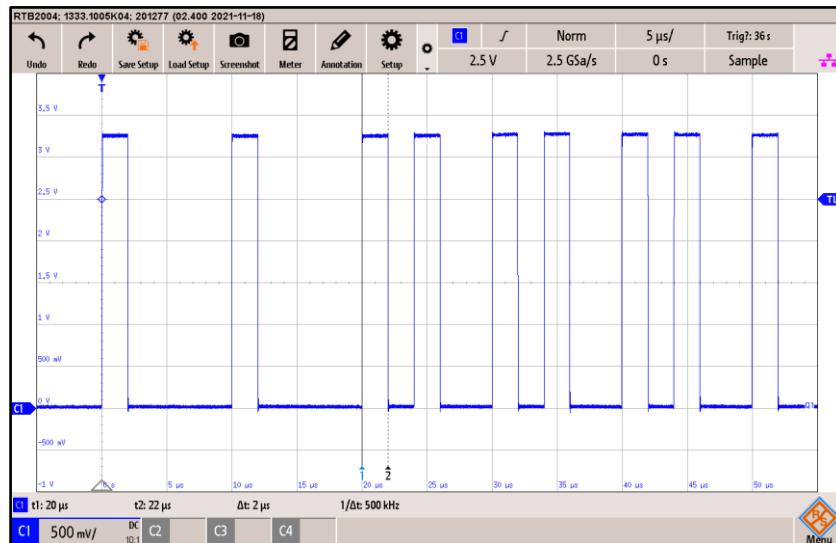


Figure 58 : Mesure fréquences TLC5973

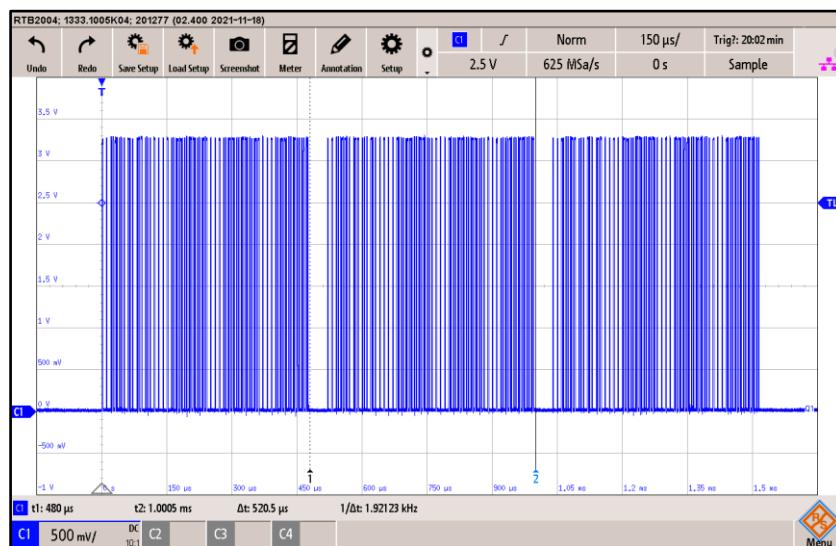


Figure 59 : Mesure de la trame complète du TLC5973

Pour effectuer le test de fonctionnement des LEDs, une commande aléatoire a été transmise au TLC5973. Il est possible de vérifier que la fréquence de 100 kHz requise entre les impulsions de début de bit est réalisée avec précision. La durée de l'impulsion correspond à ce qui a été défini dans l'interruption, c'est-à-dire 2 μs, ce qui équivaut à une fréquence de 500 kHz.

En dézoomant, on peut observer les trois séquences qui sont transmises successivement avec de courtes pauses entre chacune d'elles.

Visuellement, sur le PCB, on peut constater que les LEDs s'allument conformément aux couleurs demandées.

6.7 Buzzer

Mesure Buzzer		
Signal	Point de test	Oscilloscope
RD5	P5-1	CH1

Tableau 33 : Points de tests - Mesure Buzzer

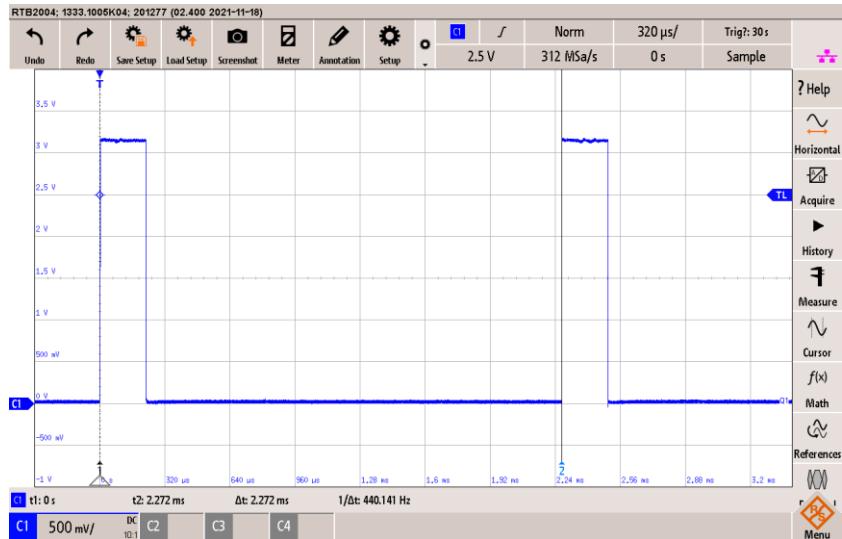


Figure 60 : Mesure de la fréquence de l'OC du buzzer

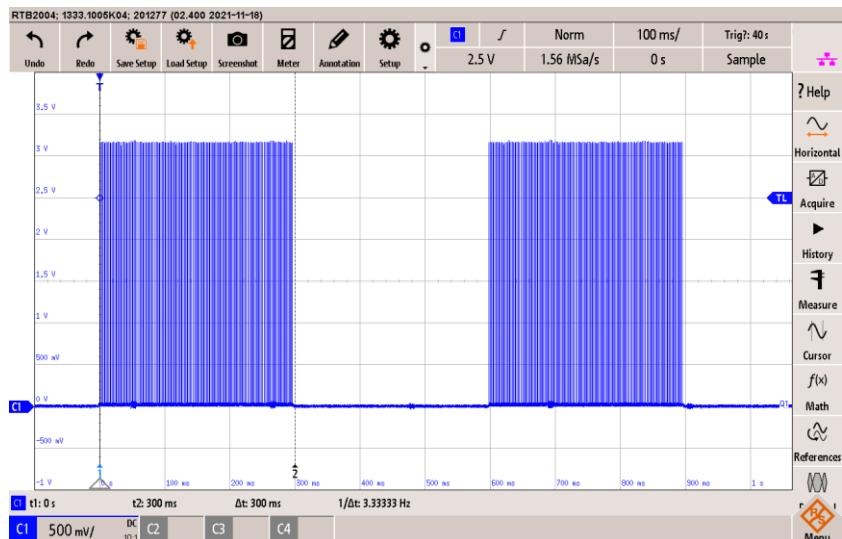


Figure 61 : Mesure du temps des notes du buzzer

Pour tester le buzzer, une séquence composée de deux notes LA (440Hz) avec une pause au milieu a été jouée. En mesurant la fréquence de l'OC générée, on peut constater que la fréquence correspond bien à ce qui a été fixée.

La durée des notes qui ont été fixé dans le code ont été calculées à l'aide de la formule disponible dans la fonction correspondante. On obtient bien la valeur attendue. La librairie et le buzzer fonctionnent comme attendu.

7 Etat d'avancement

Etat d'avancement du projet selon CDC, au 26 septembre 2023		
Exigence	Etat	Remarques
Partie Hardware		
Microcontrôleur	✓	
Lecteur RFID	✓	
Alimentation 230VAC	✓	
Convertisseur AC / DC	✓	
Commutation 230VAC	✓	
Connecteur Ethernet	⚡	Implémenté mais non fonctionnel
Module Wi-Fi	✓	
Buzzer timeout	✓	
LEDs	✓	
Boitier	⚡	Esquisse tracé mais non réalisé
Partie Firmware		
Configuration du module Wi-Fi	✓	
Commandes du module Wi-Fi	✓	
Envoi et réception par réseau	✗	Non testé car manque partie software
Commande du buzzer	✓	
Commande des LEDs	✓	
Commande du commutateur 230VAC	✓	
Configuration des timers	✓	
Partie software		
API Ethernet ou Wi-Fi	✗	La partie software n'a malheureusement pas encore été implémenté. Quelques idées ont été proposées et reste en attente de réalisation.
Réception et affichage du tag RFID	✗	
Base de donnée	✗	
Envoi de commandes	✗	

Tableau 34 : Etat d'avancement du projet

L'objectif est de réaliser une API sur le Raspberry Pi, au mieux pour les présentations des portes ouvertes ou au moins pour la défense orale.

La partie concernant l'Ethernet est réalisée en dernière priorité au vu de la complexité que présente la tâche.

8 Conclusion

Ce projet de diplôme a été une opportunité précieuse pour mettre en pratique et améliorer les compétences que j'ai acquises tout au long de ma formation à l'École Supérieure. Cette expérience concrète m'a permis de consolider mes connaissances théoriques et de les appliquer dans un contexte réel.

Tout d'abord, lors de la phase de conception, j'ai été confronté à la nécessité de rassembler une grande quantité d'informations dans un laps de temps très court. J'ai dû plonger rapidement dans des concepts tels que l'Ethernet et le Wi-Fi, que je n'avais que peu explorés auparavant. Cette phase initiale a engendré un léger retard dès le départ.

Le retard initial s'est progressivement accumulé au cours de la phase de conception du PCB, laquelle s'est avérée être un défi de taille. Cette étape a présenté des difficultés significatives en raison de la nécessité de respecter des contraintes électriques, mécaniques et de placement des composants. L'équilibre entre ces différentes contraintes s'est avéré délicat à atteindre, mais il était essentiel pour garantir la fiabilité et la performance du PCB.

Cependant, malgré les obstacles rencontrés, je ne regrette pas d'avoir investi le temps nécessaire pour réaliser cette phase avec précision. Cette approche méticuleuse a porté ses fruits, car le résultat final du PCB ne présente aucune erreur majeure. Les erreurs mineures qui ont pu survenir ont été corrigées grâce à des ajustements logiciels, ce qui a permis de maintenir la qualité et la propreté du PCB. Cette expérience m'a enseigné l'importance de la rigueur dans la conception matérielle et m'a préparé à faire face à des projets similaires à l'avenir avec une plus grande confiance dans mes compétences techniques.

En ce qui concerne la partie firmware du projet, je reconnais avoir consacré un temps considérable à des aspects qui n'étaient pas prioritaires, tels que la gestion des LEDs. À posteriori, il est clair que ce temps aurait été plus judicieusement investi dans l'amélioration de la gestion des communications UART entre les différents modules RFID et Wi-Fi. Ces communications jouent un rôle central dans le fonctionnement du système, et une optimisation aurait contribué à accroître l'efficacité globale du projet.

Cependant, malgré cette déviation des priorités initiales, cette expérience m'a permis d'approfondir ma maîtrise du langage C. Les divers défis rencontrés ont renforcé mes compétences en programmation. Néanmoins, cette leçon m'a rappelé l'importance de la gestion efficace des ressources et des priorités dans le développement de firmware.

L'accumulation de ces retards a effectivement entraîné des défis dans la réalisation de l'ensemble du cahier des charges dans les délais impartis pour la remise de ce rapport. Cependant, je reste optimiste quant à mes capacités à fournir une version complète et conforme aux exigences lors de la présentation orale.

Lausanne, le 26 septembre 2023

Miguel Santos

9 Bibliographie

- [1] « DD11-IEC-C14.pdf ». Consulté le: 24 septembre 2023. [En ligne]. Disponible sur: https://www.schurter.com/en/datasheet/typ_DD11.pdf
- [2] « WE-6914017000xxB.pdf ». Consulté le: 24 septembre 2023. [En ligne]. Disponible sur: <https://www.we-online.com/components/products/datasheet/69140170002B.pdf>
- [3] « RAC05-K.pdf ». Consulté le: 24 septembre 2023. [En ligne]. Disponible sur: <https://g.recomcdn.com/media/Datasheet/pdf/.f7S7vjMX/.t5ca32a5e916099fe1de3/Datasheet-130/RAC05-K.pdf>
- [4] « ADW1203HLW | Panasonic ». Consulté le: 24 septembre 2023. [En ligne]. Disponible sur: https://www3.panasonic.biz/ac/ae/search_num/index.jsp?c=detail&part_no=ADW1203HLW&large_g_cd=1&medium_g_cd=11&small_g_cd=112&series_cd=2154
- [5] « PIC32MX5XX6XX7XX_Family_Datasheet.pdf ». Consulté le: 24 septembre 2023. [En ligne]. Disponible sur: https://ww1.microchip.com/downloads/aemDocuments/documents/MCU32/ProductDocuments/DataSheets/PIC32MX5XX6XX7XX_Family%29Datasheet_DS60001156K.pdf
- [6] « WE-QUARTZ-830020423.pdf ». Consulté le: 24 septembre 2023. [En ligne]. Disponible sur: <https://www.we-online.com/components/products/datasheet/830020423.pdf>
- [7] « Ethernet Starter Kit II ». Consulté le: 22 août 2023. [En ligne]. Disponible sur: <https://www.microchip.com/en-us/development-tool/dm320004-2>
- [8] « SMSC-LAN8720-PHY-Daughter-Board.pdf ». Consulté le: 24 septembre 2023. [En ligne]. Disponible sur: <https://ww1.microchip.com/downloads/en/DeviceDoc/50002211A.pdf>
- [9] « LAN8720A.pdf ». Consulté le: 24 septembre 2023. [En ligne]. Disponible sur: <https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/DataSheets/00002165B.pdf>
- [10] « CRJ011-ML3-TH.pdf ». Consulté le: 24 septembre 2023. [En ligne]. Disponible sur: <https://www.cuidevices.com/product/resource/crj011-ml3-th.pdf>
- [11] « esp-at-user-guide.pdf ». Consulté le: 23 septembre 2023. [En ligne]. Disponible sur: https://docs.espressif.com/_/downloads/esp-at/en/release-v2.1.0.0_esp32/pdf/
- [12] « esp32-c3-wroom-02_datasheet_en.pdf ». Consulté le: 23 septembre 2023. [En ligne]. Disponible sur: https://www.espressif.com/sites/default/files/documentation/esp32-c3-wroom-02_datasheet_en.pdf
- [13] « MIFARE Classic EV1 ». Consulté le: 22 août 2023. [En ligne]. Disponible sur: https://www.nxp.com/products/rfid-nfc/mifare-hf/mifare-classic/mifare-classic-ev1-1k-4k:MF1S50YYX_V1
- [14] « RFID Click Board », MIKROE. Consulté le: 23 septembre 2023. [En ligne]. Disponible sur: <http://www.mikroe.com/rfid-click>
- [15] « UART-B1-User-manual.pdf ». Consulté le: 24 septembre 2023. [En ligne]. Disponible sur: <https://eccel.co.uk/wp-content/downloads/UART-B1-User-manual.pdf>
- [16] « WE-RGB-150141M173100.pdf ». Consulté le: 24 septembre 2023. [En ligne]. Disponible sur: <https://www.we-online.com/components/products/datasheet/150141M173100.pdf>
- [17] « TLC5973.pdf ». Consulté le: 24 septembre 2023. [En ligne]. Disponible sur: [https://www.ti.com/lit/ds/symlink/tlc5973.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1695554399423&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253Fd\(productId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252Ftlc5973](https://www.ti.com/lit/ds/symlink/tlc5973.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1695554399423&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253Fd(productId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252Ftlc5973)

- [18] « CMT-0904-83T.pdf ». Consulté le: 24 septembre 2023. [En ligne]. Disponible sur: <https://www.cuidevices.com/product/resource/cmt-0904-83t.pdf>
- [19] « Classification », Eurocircuits. Consulté le: 18 septembre 2023. [En ligne]. Disponible sur: <https://www.eurocircuits.com/pcb-design-guidelines/classification/>
- [20] « PCB Trace Width Conversion Calculator | DigiKey ». Consulté le: 18 septembre 2023. [En ligne]. Disponible sur: <https://www.digikey.hk/en/resources/conversion-calculators/conversion-calculator-pcb-trace-width>
- [21] « Utilisation d'un calculateur d'espacement IPC-2221 pour la conception haute tension », Altium. Consulté le: 19 septembre 2023. [En ligne]. Disponible sur: <https://resources.altium.com/fr/p/using-an-ipc-2221-calculator-for-high-voltage-design>
- [22] « esp32_hardware_design_guidelines_en.pdf ». Consulté le: 24 septembre 2023. [En ligne]. Disponible sur: https://www.espressif.com/sites/default/files/documentation/esp32_hardware_design_guidelines_en.pdf
- [23] « RFID-B1-User-Manual.pdf ». Consulté le: 25 septembre 2023. [En ligne]. Disponible sur: <https://eccel.co.uk/wp-content/downloads/RFID-B1-User-Manual.pdf>



ChatGPT a été intégré dans ce rapport exclusivement pour la correction orthographique et la rédaction de reformulations. Toutes les informations contenues dans ce document, sauf indication contraire, sont de la responsabilité de l'auteur du rapport.



*Les icônes utilisées ont été obtenues en libre téléchargement sur le site :
<https://thenounproject.com/>*



*Les diagrammes temporels ont été créés grâce au projet open-source :
<https://wavedrom.com/>*

10 Logiciels

 Windows 10		
Description	Nom	Version
Conception électronique	Altium Designer	22.7.1
Modélisation 3D	SolidWorks	2023
IDE de programmation du PIC32	MPLAB X	5.45
Configuration du PIC32	Harmony	2.06
Compilateur intégré à MPLAB	XC32	2.50
Analyseur logique	LogicSniffer	0.9.7.2
Gestionnaire de versions	GitHub Desktop	3.3.3
Gestionnaire de bibliographie	Zotero	6.0.27

		
Description	Nom	Version
Utilitaire de lecture RFID	NFC Tools	8.9

11 Figures

Figure 1 : Illustration du système, issu du CDC	7
Figure 2 : Schéma-bloc du système	8
Figure 3 : Schéma du convertisseur AC/DC	10
Figure 4 : Schéma du relai de puissance	11
Figure 5 : Assignation des broches du PIC32	13
Figure 6 : Condensateurs de découplage PIC32	13
Figure 7 : Port de programmation PIC32	13
Figure 8 : Schéma du quartz externe PIC32	13
Figure 9 : Principe du contrôleur PHY	14
Figure 10 : Schéma de la "PHY Daughter Board"	15
Figure 11 : Schéma de l'oscillateur externe sur le "Ethernet Kit 2"	15
Figure 12 : Modes de démarrage de l'ESP32	16
Figure 13 : Schéma de sélection du mode de l'ESP32	16
Figure 14 : Technologie du badge RFID de l'ES	17
Figure 15 : Schéma des connecteurs des modules RFID	17
Figure 16 : Schéma des LEDs RGB d'interface	18
Figure 17 : Schéma LEDs témoins	19
Figure 18 : Schéma du buzzer	19
Figure 19 : Image du Raspberry Pi 3B+	21
Figure 20 : Vue de la couche TOP	22
Figure 21 : Vue de la couche BOTTOM	22
Figure 22 : Vue du dessus du PCB	23
Figure 23 : Vue du dessous du PCB	23
Figure 24 : Stratégie de placement des composants	25
Figure 25 : Règle de connexion au plan de masse	27
Figure 26 : Règle de stitching du plan de masse	27
Figure 27 : Vue de l'oscillateur du contrôleur PHY	27
Figure 28 : Vue de l'oscillateur du PIC32	27
Figure 29 : Paires différentielles de la partie Ethernet	28
Figure 30 : Impédance des paires différentielles de l'Ethernet	28
Figure 31 : Plan de masse isolé du connecteur RJ45	28
Figure 32 : Recommandations de placement de l'ESP32	29
Figure 33 : Recommandation de dégagement pour l'antenne de l'ESP32	29
Figure 34 : Machine d'état global du firmware	30

Figure 35 : Machine d'état - ESP	32
Figure 36 : Configuration harmony UART1	32
Figure 37 : Machine d'état - CHU	33
Figure 38 : Configuration Harmony UART2	33
Figure 39 : Initialisation librairie RFIDB1	34
Figure 40 : RFIDB1 - Construction d'une trame de type A	34
Figure 41 : RFIDB1 - Construction d'une commande non crypté	34
Figure 42 : RFIDB1 - Construction d'une réponse du module	34
Figure 43 : Machine d'état - BZR	35
Figure 44 : Configuration Harmony - OC5	35
Figure 45 : Configuration Harmony - TMR3	35
Figure 46 : Séquences de contrôle des LEDs	36
Figure 47 : Timings à respecter - TLC5973	36
Figure 48 : Configuration Harmony - TMR2	37
Figure 49 : Représentation temporel de l'encodage des bits	37
Figure 50 : Illustration - SQLite	38
Figure 51 : Mesure du +3V3	39
Figure 52 : Mesure de la commutation 230VAC sur le relai	40
Figure 53 : Mesure trame UART RFID	41
Figure 54 : Analyse trame UART RFID	41
Figure 55 : Mesure trame UART TX ESP32	42
Figure 56 : Mesure trame UART RX ESP32	42
Figure 57 : Analyse trame UART ESP32	42
Figure 58 : Mesure fréquences TLC5973	43
Figure 59 : Mesure de la trame complète du TLC5973	43
Figure 60 : Mesure de la fréquence de l'OC du buzzer	44
Figure 61 : Mesure du temps des notes du buzzer	44

12 Tableaux

Tableau 1 : Caractéristiques principales de la prise IEC C14	9
Tableau 2 : Caractéristiques principales des borniers	9
Tableau 3 : Estimation du courant maximal	10
Tableau 4 : Caractéristiques principales du convertisseur AC/DC	10
Tableau 5 : Caractéristiques principales du relai de puissance	11
Tableau 6 : Caractéristiques principales du microcontrôleur	12

Tableau 7 : Périphériques et nombre de broches nécessaires pour le PIC32	12
Tableau 8 : Caractéristiques principales du contrôleur PHY	14
Tableau 9 : Caractéristiques principales du connecteur RJ45	14
Tableau 10 : Caractéristiques principales du module Wi-Fi	16
Tableau 11 : Comparaison des modules RFID	17
Tableau 12 : Caractéristiques principales des LEDs RGB	18
Tableau 13 : Caractéristiques principales du driver de LED	18
Tableau 14 : Caractéristiques principales du buzzer	19
Tableau 15 : Liste des points de mesure du système	20
Tableau 16 : Spécifications principales du PCB	24
Tableau 17 : Règles principales de fabrication Eurocircuit - Classe 6C	24
Tableau 18 : Largeurs des pistes de puissance	26
Tableau 19 : Distance d'isolation 230 [VAC]	26
Tableau 20 : Description des librairies firmware	31
Tableau 21 : Composantes principales des machines d'états	31
Tableau 22 : Spécificités de la librairie ESP	32
Tableau 23 : Spécificités de la librairie CHU	33
Tableau 24 : Spécificités de la librairie RFIDB1ClientInterface	34
Tableau 25 : Spécificités de la librairie BZR	35
Tableau 26 : Spécificités de la librairie TLC5973	36
Tableau 27 : Liste du matériel de mesure	39
Tableau 28 : Points de tests – Mesure convertisseur AC/DC	39
Tableau 29 : Points de tests - Mesure commutation 230VAC	40
Tableau 30 : Points de tests - Mesure UART RFID	41
Tableau 31 : Points de tests - Mesure UART ESP32	42
Tableau 32 : Points de tests - Mesure TLC5973	43
Tableau 33 : Points de tests - Mesure Buzzer	44
Tableau 34 : Etat d'avancement du projet	45

13 Equations

Équation 1 : Résistance de base du transistor	11
Équation 2 : Calcul des condensateurs du quartz externe	13
Équation 3 : Résistance de limitation de courant du TLC5973	18
Équation 4 : Calcul de la résistance des LEDs témoins	19
Équation 5 : Calcul du prescaler minimum	37

14 Annexes

14.1 Cahier des charges

14.2 Planification

14.3 Journal de travail

14.4 Procès-verbaux des séances hebdomadaires

14.5 Schémas électroniques

- Schéma bloc
- Puissance
- PIC32
- Ethernet
- ESP32
- RFID
- Interfaces

14.6 Fichiers de fabrication

- Vues des projections du PCB
- Cotations du PCB et composants
- Vues des couches du PCB
- Vues réalistes du PCB
- Liste des composants

14.7 Liste des règles Altium

14.8 Code source firmware

14.9 Fiche de modifications

14.10 Mode d'emploi

CAHIER DES CHARGES

DIPLOME

Badge pour Place de Travail SLO

N° projet 2312

Mandataire

Entreprise/Client:	Ecole supérieure - Lausanne	Département:	SLO
Demandé par (Prénom, Nom):	Section SLO SCA – PBY – JMO	Date:	26.07.2023

1 Objectif - Cahier des charges

Le but de ce travail de diplôme est de concevoir un dispositif électronique permettant à un étudiant, à l'aide d'un badge RFID¹ reçu lors de la première année, de pouvoir activer les appareils se trouvant à sa place de travail (ex : alimentation de laboratoire, oscilloscope, générateur, autres), ou d'activer l'appareillage se trouvant au « local de montage » (station de dessoudage, fer à braser, binoculaire,) pour une durée limitée ; ceci doit permettre une meilleure gestion de la consommation électrique des appareils électriques (éviter les oubli d'extinction des appareils), une sécurité concernant les stations brassage (risque minime d'incendie), une gestion des droits d'utilisation et un suivi (log).

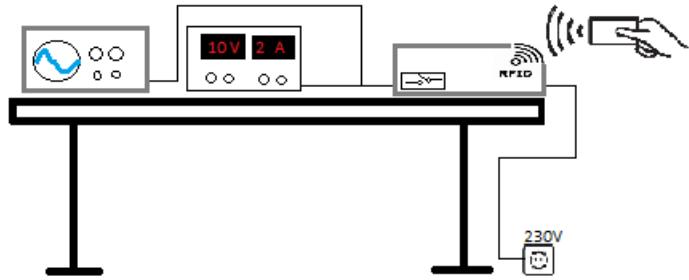


Figure 1: représentation symbolique du dispositif

Le système électronique doit pouvoir lire un badge RFID, activer ou non un commutateur 230 VAC selon une base donnée qui sera lue via le protocole Ethernet ou via le WiFi (heure d'activation, durée, autre).

Le système devra gérer la notion de timeout par une indication lumineuse et/ou sonore.

Le dispositif devra avoir une adresse permettant une mise en relation avec la base de données (logs d'utilisation) : utilisation de la place de travail par qui, combien de fois, combien de temps, etc.

Partie Hardware

L'étudiant devra concevoir une carte électronique basé sur un microcontrôleur de son choix ; cette carte doit contenir au minimum :

- Un microcontrôleur (Microchip PIC32 ou STM32 ARM M0) – attention aux ruptures de stock.

¹ RFID : Radio Frequency Identification

- Une partie RFID permettant de lire des tags RFID (module ou propre conception)

Attention : avant d'implémenter un lecteur, voir la technologie des tags de l'ETML-ES (fréquence, type de datas, etc)
- Une antenne RFID
- Une partie alimentation :
 - Le dispositif sera alimenté en 230V AC
 - Connecteur mâle 230V
 - Convertisseurs AC/DC -> les différentes tension DC dont le système a besoin sont à définir : 3.3V - 5V – autre ?
- Un étage pour l'activation du 230V
 - Connecteur femelle 230V
- Un étage pour la communication Ethernet et Wifi
 - Connecteur Ethernet
- Un étage HMI²
 - Buzzer pour avertissement Timeout
 - Leds pour avertissement Timeout, communication Wifi et/ou Ethernet, tag RFID (autorisé – refusé)

L'étudiant devra choisir un boîtier du commerce et l'adapter (design des ouvertures pour les différents connecteurs 230V / leds / buzzer). Dans l'idéal, le boîtier devra avoir les dimensions suivante (L – l – h) : 170 – 100 – 50 mm. L'étudiant peut s'inspirer des lecteurs RFID utiliser lors des laboratoires d'ELCO³.

Partie Firmware

L'étudiant réalisera un Firmware (partie microcontrôleur) permettant de faire au minimum les tâches suivantes :

- Lecture des informations de tags RFID (ID unique) -> attention à la technologie des tags
- Envoi et réception de données par Ethernet ou Wifi via le uC⁴
 - Configuration du contrôleur– commandes / protocoles
 - Envoi : ID du tag RFID
 - Réception : commande d'activation commutation 230V, durée, autre
- Activation de GPIO⁵ :
 - Commande de leds
 - Commande du buzzer
 - Commande du commutateur pour l'activation du 230V
- Configuration de timers
 - Comptage du Timeout

La système (carte électronique) sera vu comme un client.

² HMI : Human Machine Interface

³ ELCO : cours d'électronique de communication

⁴ uC : microcontrôleur

⁵ GPIO : General Purpose Input/Ouput

Partie Software

L'étudiant devra concevoir une petite application sur PC ou sur un autre système embarqué (tel qu'un Raspberry Pi) permettant de faire au minimum les tâches suivantes :

- Configuration API Ethernet ou Wifi
- Réception et affichage de l'ID du tag RFID
- Base de données (évt. simplifiée) selon l'ID du tag et la place
 - MAJ des logs de la carte
 - Qui -> nom étudiant
 - Emplacement -> n° place de travail / local de montage
 - Durée d'activation
- Envoi commandes -> à définir le format de la trame
 - Activation 230V
 - Badge non reconnu
 - Durée

1.1 Données en lien avec l'objectif – les grandes lignes

- Recherche de solutions **Hardware**
 - Recherche et implémentation d'un microcontrôleur (préférence ci-dessous)
 - Fabriquant Microchip -> famille PIC32
 - Fabriquant ST Electronics -> famille STM32 -> M0
 - Recherche et implémentation de convertisseur de tension AC/DC
 - Carte alimentée en 230V AC
 - Recherche des tensions DC dont le système a besoin : 5V, 3.3V, autre ?
 - Connecteur suisse male pour cordon d'alimentation 230V
 - Recherche et implémentation d'un lecteur RFID
 - Recherche informations sur les badges utilisés par l'école -> compatible au projet ?
 - Préférence pour un module lecteur RFID déjà réalisé
 - Voir si besoin de concevoir une antenne
 - Voir communication avec le uC (UART, SPI, I2C, autre)
 - Recherche et implémentation pour la commande d'activation du 230V
 - Relais ?
 - Optocoupleurs ?
 - Triacs ?
 - Connecteur suisse femelle 230V pour appareils ou multiprise
 - Recherche et implémentation pour communication Ethernet et Wifi
 - Contrôleur Ethernet
 - Communication entre le uC et le contrôleur Ethernet (UART, SPI, I2C, autre)
 - Module Wifi
 - Communication entre le uC et le module Wifi (UART, SPI, I2C, autre)
 - Recherche et implémentation pour l'interface utilisateur :
 - Leds (timeout / alimentation / vie du uC / autorisation / autre)
 - Indicateur sonore pour timeout
- Recherche de composants électroniques autour des composants décrits ci-dessus : AOP, résistances, condensateur, self, autres
- Réalisation de schématique(s) complet et d'un PCB - à réaliser sous ALTIUM de préférence
- Recherche d'un boîtier (achat) – modification sous SolidWorks

- Recherche de solutions au niveau **Firmware**
 - Pilotage de Leds pour informations visuelles (alarme – communication – autre)
 - Pilotage d'un Buzzer
 - Pilotage commande commutation 230V
 - Gestion et configuration de Timers
 - Communication Ethernet (uC <-> PC (application))
 - Communication Ethernet (uC <-> PC (application))
 - Lecture datas RFID
- Recherche de solutions au niveau **Software**
 - Gestion API Ethernet
 - Gestion API Wifi
 - Envoi et réception des données (ID tag RFID, activation pour 230V, autre)
 - Affichage de l'ID d'un tag RFID
 - Simulation MAJ d'une base de donnée
- Démontrer par différentes simulations et mesures que les parties Hardware, Firmware et Software ont été bien implémentées.

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

- Fichiers sources de CAO électronique du PCB réalisé (ALTIUM) + configuration logiciel (version utilisées)
- 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 logiciel (version utilisées)
- Fichiers sources de l'application C# (.cs) + configuration logiciel (version utilisées)
- Tout le nécessaire pour programmer le microcontrôleur (logiciel ou fichiers .hex), sous format numérique => utilisation de la structure de projet fourni par l'ES
- Tout le nécessaire pour programmer l'application C# (logiciel ou fichiers .hex) sous format numérique => utilisation de la structure de projet fourni par l'ES
- Tout le nécessaire à l'installation de programmes sur PC ou autres environnements utilisés durant le travail de diplômes
- Un rapport de diplôme contenant :
 - Les concepts du design **Hardware** :
 - Études des différents systèmes à implémenter (explications)
 - Choix des composants et dimensionnement de ceux-ci (calculs / simulation / autre)
 - Réalisation schématique / PCB / boitier / montage de la carte
 - Les concepts du design **Firmware**
 - Structogramme / flowchart / Pseudocode
 - Explication des algorithmes mise en place
 - Démonstration par calculs / outils de debug / des résultat obtenu
 - Validation des concepts mis en place
 - Les concepts du design **Software**
 - Structogramme / flowchart / Pseudocode
 - Explication des algorithmes mise en place
 - Démonstration par calculs / outils de debug / des résultat obtenu
 - Validation des concepts mis en place
 - Tests & Validation :
 - Méthodologie de tests
 - Mesure(s)
 - Validation des résultats
 - Correction(s) apportée(s) au design (Hardware / Firmware / Software)
 - Estimation des coûts pour le design développé (un prototype)
 - Etat d'avancement & problèmes rencontré
 - Conclusion
 - Bibliographie / webographie / autre sources
- Les annexes :
 - Calculs détaillés des concepts
 - Listings complets des parties Firmware & Hardware que vous avez implémenté
 - Schématique + plan d'implémentation complète du PCB
 - Dessin / schématique du boitier
 - Mesures
 - Pages utilisée des différents datasheets ou documentations exploités

- Mode d'emploi du système développé pendant le diplôme
- Journal de travail
- PV de séances hebdomadaires
- Un prototype

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 mandataire 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 mandataire.

4 Documents de références

Projet ES comme référence :

- 1630 -> Timbreuse à Badge RFID
- 1920 -> Minuterie pour multiprise
- 2130 -> Système de timbrage à Badge RFID
-

Vidéo

- Vidéo sur les datas d'un TAG RFID (en)
https://www.youtube.com/watch?v=crqCPL4Dfl4&ab_channel=GS1US
- Vidéo pour créer un client/Serveur TCP/IP – application C#
https://www.youtube.com/watch?v=ve2LX1tOwIM&ab_channel=FoxLearn

Pages web :

- Article général sur le RFID (fr) :
<https://sbedirect.com/fr/blog/article/comprendre-la-rfid-en-10-points.html>
- Article sur le fonctionnement RFID (fr) :
<https://www.connectwave.fr/techno-appli-iot/rfid/fonctionnement-dun-systeme-rfid/>
- Article sur les Tags RFID – Principe (fr) :
<https://altec.ch/technologies/les-tags-rfid/>

- Article sur les datas d'un tag RFID (en) :
<https://learn.sparkfun.com/tutorials/rfid-basics/tag-memory>
- Article sur la notion d'optocoupleur et de triac pour la commutation du 230V (fr) :
<https://www.abonnel.fr/electronique/lois-et-composants/interface-5v-230v-avec-optocoupleur-et-triac>
- Article sur le fonctionnement d'un triac (fr) :
https://www.sonelec-musique.com/electronique_theorie_triac.html
- Article sur la notion de client / Serveur (en) :
<https://www.zenarmor.com/docs/network-basics/what-is-client-server-network>
- PDF – communication réseau (fr) :
http://pedagogie.ac-limoges.fr/eco-gest/IMG/pdf/communcation_reseau.pdf
- Wiki sur la notion de RFID (fr) :
<https://fr.wikipedia.org/wiki/Radio-identification>

ETNL-ES	21.août	22.août	23.août	24.août	25.août	26.août	27.août	28.août	29.août	30.août	31.août	01.sept	02.sept	03.sept	04.sept	05.sept	06.sept	07.sept	08.sept	09.sept	10.sept	11.sept	12.sept	13.sept	14.sept	15.sept
Pré-étude																										
Design																										
Revue de schémas																										
Commande composants																										
PCB																										
Revue de PCB																										
Commande PCB																										
Montage et tests																										
Modélisation 3D																										
Firmware PCB																										
Software RPI																										
Mise en service																										
Documentation																										
Rendu mémoire																										
Rendu affiche + résumé																										
Portes ouvertes																										
Présentation orale																										

ETNL-ES	16.sept	17.sept	18.sept	19.sept	20.sept	21.sept	22.sept	23.sept	24.sept	25.sept	26.sept	27.sept	28.sept	29.sept	30.sept	01.oct	02.oct	03.oct	04.oct	05.oct	06.oct	07.oct	08.oct
Pré-étude																							
Design																							
Revue de schémas																							
Commande composants																							
PCB																							
Revue de PCB																							
Commande PCB																							
Montage et tests																							
Modélisation 3D																							
Firmware PCB																							
Software RPI																							
Mise en service																							
Documentation																							
Rendu mémoire																							
Rendu affiche + résumé																							
Portes ouvertes																							
Présentation orale																							

2312 Badge pour place de travail
Miguel Santos
Journal de travail

Date	Heure	Tâche réalisée
Semaine 1		
21.08.2023	10:00 - 11:30	Réception du cahier des charges et entretien initial.
	13:00 - 16:00	Recherches d'informations sur la technologie RFID
22.08.2023	08:30 - 10:00	Schéma bloc de principe
	10:00 - 11:30	Recherche des composants à utiliser : - Module RFID - Convertisseur AC/DC - Relais de commutation - Gestion de la base de donnée - Boitier
	12:30 - 16:00	Validation des composants : - Module RFID = MIKROE-1434 - Convertisseur AC/DC = RAC03-3.3SK
23.08.2023	08:30 - 11:30	Choix de divers composants : - Connecteur utilisé pour l'entrée/sortie 230VAC. - Relai de commutation 230VAC - Module RFID - Buzzer - LEDS témoins - Module Wifi
	13:00 - 14:00	Réalisation du schéma-bloc hardware
	14:00 - 16:30	Recherche sur le module WIFI et le module ethernet
	16:30 - 17:30	Séance avec maître de diplôme
24.08.2023	08:00 - 10:30	Dimensionnement des composants externes du circuit d'alimentation
	10:30 - 11:00	Recherche et choix d'un nouveau convertisseur AC/DC : Traco TMPW_5-103 Offre plus de protections, de meilleures certifications EMC, le tout en un seul bloc sans composants externes, entreprise et fabrication Suisse
	13:00 - 16:00	Début de la schématique d'alimentation, recherche des symboles et footprints des composants.
	16:00 - 17:00	Organisation du projet. Ecriture du rapport.
25.08.2023	08:00 - 10:00	Séance hebdomadaire avec maître de diplôme
	10:00 - 12:00	Rédaction du procés-verbal de la séance hebdomadaire
	13:00 - 16:00	Réalisation de la schématique, recherches de footprints et de composants.
	16:00 - 17:00	Réalisation de tests sur la consommation de courant des différents appareils.
26.08.2023	10:00 - 16:00	Réalisation la schématique du connecteur et contrôleur Ethernet. Recherche du fonctionnement et des footprints des composants nécessaires
27.08.2023	10:00 - 19:00	Finalisation de la schématique Ethernet. Schématique des interfaces. Diverses recherches de composants et ajustements des schématiques. Design de composants (HP, LEDS, transistors, etc...)
Semaine 2		
28.08.2023	08:30 - 10:00	Revue de schéma #1 avec maître de diplôme
	10:15 - 12:15	Dimensionnement des résistances pour les transistor de commutation des relais. Corrections du schéma "2312_Puissance.SchDoc" (Schéma finalisé)
	13:30 - 14:30	Schématique "2312_Interface.SchDoc" (Terminé)
	15:00 - 16:00	Schématique "2312_Ethernet.SchDoc" (Terminé)
	16:00 - 18:00	Etude et schématique "2312_ESP32.SchDoc" + "2312_RFID.SchDoc"
	18:00 - 22:00	Schématique "2312_PIC32.SchDoc" et finalisation de toute les schématiques (90% terminées)
29.08.2023	08:00 - 12:00	Finalisation des schématiques à 100%
	13:00 - 15:00	Recherche chez les fournisseurs, réalisation de la bom et commande des composants
	15:00 - 20:00	Mise en page de la schématique et correction de footprints
30.08.2023	08:30 - 10:00	Dimensionnement mécanique du PCB
	10:00 - 12:00	Revue de la schématique de T. Neziri
	13:00 - 20:00	Placement des composants sur le PCB
	21:00 - 23:00	Recherche d'une nouvelle méthode de connexion au 230VAC. Recherche et commande des composants nécessaires.

Date	Heure	Tâche réalisée
31.08.2023	08:00 - 20:00	Placement des composants sur le PCB et début de routage de l'ethernet
01.09.2023	08:00 - 20:00	Routage de l'ethernet et composants annexes au module
02.09.2023	10:00 - 19:00	Routage du PCB en entier
	10:00 - 16:00	Routage finale du PCB
03.09.2023	17:00 - 18:30	Revue du PCB avec maître de diplôme
	18:30 - 21:00	Corrections des erreurs de conception et de règles de fabrication erronées.
Semaine 3		
04.09.2023	08:30 - 10:00	Ajustements finaux sur le PCB (marquage + logos)
	10:00 - 11:30	Vérifications finales et réalisation du panel. Commande du PCB effectuée !
	13:30 - 14:00	Validation des composants reçus sur Saphir.
	14:00 - 15:00	Rapport recommandé sur Word (initialement sur Latex) Mise en page du document Word. Rédaction de l'introduction.
	15:30 - 16:00	Mise à jour du planning (pour inclure week-ends et décallage) Rédaction et envoi du PV.
05.09.2023	08:30 - 11:30	Rédaction de la partie "Design" du rapport
	13:30 - 17:00	Mise en page générale
06.09.2023	08:30 - 10:30	Réalisation d'un câble pour le module RFID
	10:30 - 11:30	Configuration du projet Harmony sur MPLAB et récupération des librairies nécessaires au projet
	13:30 - 20:00	Lecture de la documentation et programmation du module RFID "CHILLI"
07.09.2023	08:30 - 11:30	Programmation du module RFID "CHILLI" avec utilisation des librairies du fabricant
	13:00 - 16:00	
	17:00 - 19:00	Rédaction de rapport (page de titre + design)
08.09.2023	08:30 - 11:30	Rédaction de rapport (design)
	11:30 - 12:30	Séance avec maître de diplôme
	13:30 - 16:00	Programmation module RFID
Semaine 4		
11.09.2023	08:30 - 12:00	
	13:30 - 20:00	Montage de tous les composants SMD du PCB + tests si courts-circuits
12.09.2023	08:30 - 11:30	Montage des composants traversants + corrections erreur de montage des LEDS. PCB entièrement monté et pas d'erreur hardware repérée.
	13:00 - 16:30	Création du projet sous harmony + configuration de base des pins
		Configuration des différents drivers harmony nécessaire
		Programmation du buzzer
	Documentation sur le module ESP32 pour le programmer	
13.09.2023	08:30 - 12:00	
	13:30 - 20:00	Réalisation des librairies de contrôle des LEDS
	15:30 - 16:30	Séance hebdomadaire avec maître de diplôme
14.09.2023	08:30 - 12:00	
	13:30 - 20:00	Design et réalisation des différentes machines d'états du système
15.09.2023	08:30 - 12:00	Programmation du module ESP32. Utilisation d'une machine virtuelle avec programmes dédiés à la programmation du module
	14:30 - 15:00	Première séance avec un expert (Mr. Coulinge Emilien)
	15:00 - 20:00	Documentation sur la partie software (diagrammes d'états, etc...)
16.09.2023		
17.09.2023	10:00 - 22:00	Rédaction du rapport et des fichiers annexes
Semaine 5		
18.09.2023	10:00 - 22:00	Rédaction du rapport et des fichiers annexes Création des fichiers de fabrication (fichier Draftsman) sur Altium
19.09.2023	08:00 - 20:00	Programmation et design des machines d'états Rédaction du rapport et des fichiers annexes
20.09.2023	07:00 - 20:00	Programmation du module RFID et début ESP32.
21.09.2023	08:00 - 20:00	Programmation de l'ESP32 avec connexion Wi-Fi. Mise à jour de divers documents annexes
22.09.2023	08:00 - 20:00	Programmation du module RFID et ESP32 + autres machines d'états
23.09.2023	08:00 - 00:00	Rédaction du rapport et de la documentation
24.09.2023	10:00 - 00:00	Rédaction du rapport et de la documentation
Semaine 6		
25.09.2023	08:00 - 00:00	Rédaction du rapport et de la documentation

Date	Heure	Tâche réalisée
26.09.2023	10:00	Rendu du rapport final
27.09.2023	16:30	Rendu de l'affiche et du résumé

Diplôme SLO
SANTOS Miguel
 2312 Badge pour place de travail

Procès-verbal du vendredi 25 août 2023

Présences	État des lieux
<ul style="list-style-type: none"> • Mr. Santos Miguel • Mr. Bovey Philippe 	Fin de la pré-étude avec une journée de retard, selon le planning initial. Début de la réalisation du design et de la schématique.
Problèmes rencontrés	Solutions proposées
Méthode de connexion de l'entrée et de la sortie 230VAC au PCB.	Utilisation de borniers sur le PCB et de fiches déportés sur le boitier
Aucun moyen d'éteindre complètement l'appareil	Ajout d'un interrupteur 230V externe ou intégré à la fiche d'entrée
Choix entre deux possibilités de module RFID	Utilisation du module « Chilli UART » mais implémenter l'empreinte du module Mikroe par précaution.
Décisions prises	Objectifs jusqu'à prochaine réunion
Commandes de divers composants : <ul style="list-style-type: none"> - Convertisseur AC/DC - LEDS bicolores - Buzzer - Microcontrôleur PIC32 - Module RFID « CHILLI » 	Réalisation des schémas électronique et revue de schéma le lundi 28.08. Développer PCB pour en réaliser, si possible, une revue lors de la prochaine réunion.
Prochaine réunion planifiée le :	
Vendredi 1^{er} septembre 2023 08h00 – 10h00 Salle R110 ES	
Destinataires du procès-verbal :	
Grégoire Rossier Doyen de l'ETML-ES	Philippe Bovey Maître de diplôme
Lausanne, le 28 août 2023	

Diplôme SLO
SANTOS Miguel
 2312 Badge pour place de travail

Procès-verbal du vendredi 01 septembre 2023

Présences	État des lieux
<ul style="list-style-type: none"> Mr. Santos Miguel Mr. Bovey Philippe 	Fin de la partie de design et des schématiques avec une journée de retard. Conception du PCB en cours, retard d'un jour et demi, causé par les difficultés sur les paires différentielles, le placement des composants et des erreurs de règles chez le fabricant.
Problèmes rencontrés	Solutions proposées
Méthode de routage des paires différentielles.	Utilisation des outils intégrés à Altium.
Difficultés de placement des composants en respectant les recommandations du fabricant.	Placement d'une partie des composants sur la face inférieure.
Difficultés de routage du microcontrôleur.	Recherche et utilisation de techniques de routage.
Décisions prises	Objectifs jusqu'à prochaine réunion
Commande du PCB le lundi 04.09 avant 12h00.	Respect des délais fixés dans les décisions prises.
Revue intermédiaire du rapport pour mardi 05.09 au soir.	Réalisation de la partie software autant que possible.
Prochaine réunion planifiée le :	
Vendredi 08 septembre 2023 12h00 – 13h00 Salle R110 ES	
Destinataires du procès-verbal :	
Grégoire Rossier Doyen de l'ETML-ES	Philippe Bovey Maître de diplôme
Lausanne, le 4 septembre 2023	

Diplôme SLO
SANTOS Miguel
 2312 Badge pour place de travail

Procès-verbal du vendredi 08 septembre 2023

Présences	État des lieux
<ul style="list-style-type: none"> • Mr. Santos Miguel • Mr. Bovey Philippe 	<ul style="list-style-type: none"> - Finalisation du routage du PCB avec validation et commande du PCB sous forme de Panel (04.09) - Finalisation des commandes de composants (05.09) - Avancement du rapport sur les parties pré-étude et design - Prototype du software sur le kit PIC32 -> développement RFID.
Problèmes rencontrés	Solutions proposées
Planning pas suffisamment précis (limité par le logiciel)	Changement pour un planning réalisé sur excel offrant plus de flexibilité.
Décisions prises	Objectifs jusqu'à prochaine réunion
<ul style="list-style-type: none"> - Réaliser les différents diagrammes software : <ul style="list-style-type: none"> o Machines d'états o Tableau descriptifs des fonctions utilisées o Tableau descriptifs des variables principales o Flowchart des fonctions perso. - Ordre de priorité du software : <ul style="list-style-type: none"> o RFID o ESP32 o GPIO et PWM o Raspberry Pi o Ethernet 	<ul style="list-style-type: none"> - Finir le rapport jusqu'à la partie PCB minimum - Montage et tests du PCB - Diagrammes des machines d'états - Début du firmware selon les priorités fixés
Prochaine réunion planifiée le :	
Mercredi 13 septembre 2023 15h30 Salle R110 ES	
Destinataires du procès-verbal :	
Grégoire Rossier Doyen de l'ETML-ES	Philippe Bovey Maître de diplôme
Lausanne, le 8 septembre 2023	

Diplôme SLO
SANTOS Miguel
 2312 Badge pour place de travail

Procès-verbal du mercredi 13 septembre 2023

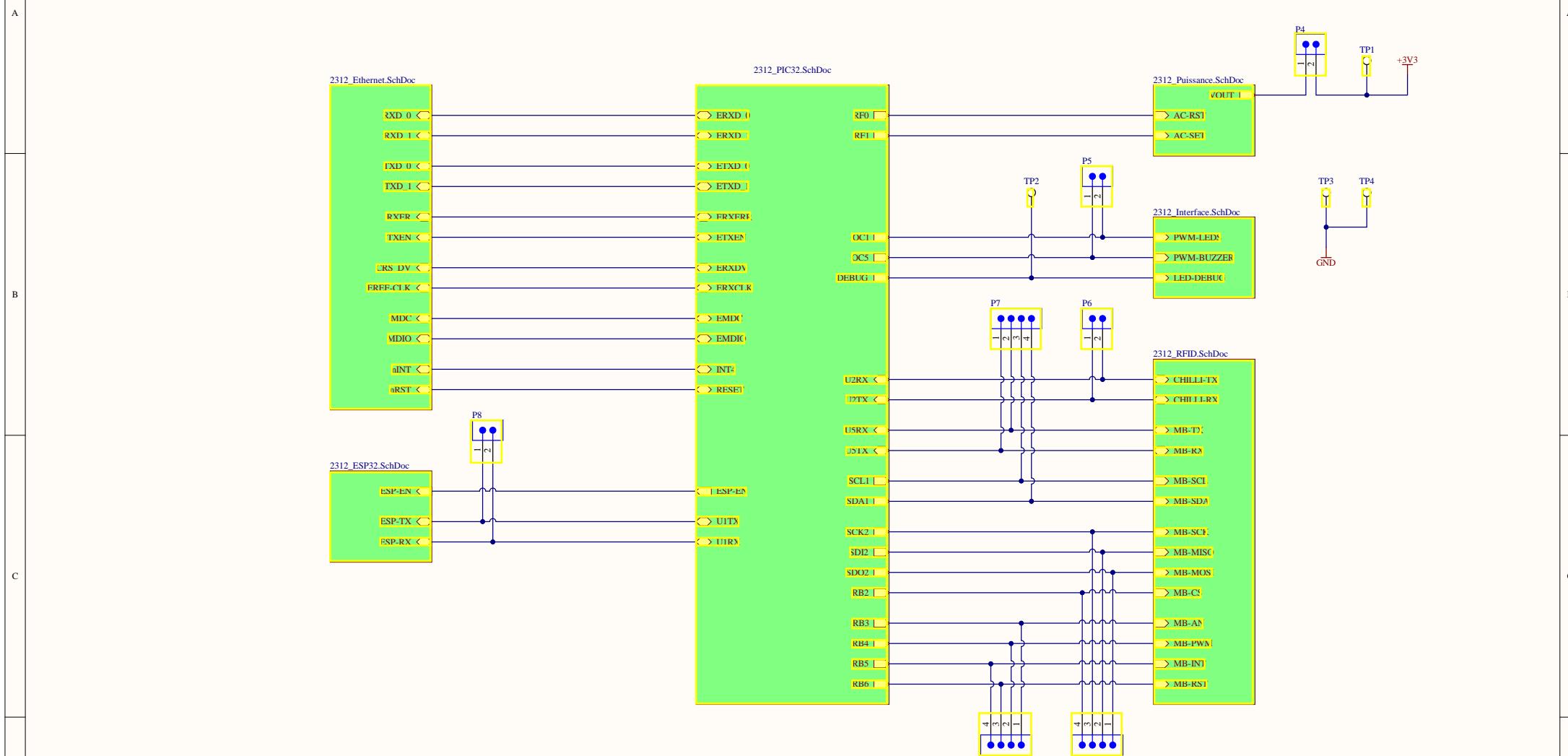
Présences	État des lieux
<ul style="list-style-type: none"> • Mr. Santos Miguel • Mr. Bovey Philippe 	PCB entièrement monté sans problème apparent (reste à tester certaines lignes de communication)
Problèmes rencontrés	Solutions proposées
Court-circuit entre le VCC et le GND	Un pont entre deux vias s'était formés sous le microcontrôleur [Problème réglé]
Décisions prises	Objectifs jusqu'à prochaine réunion
<p>Finir au plus vite la documentation sur le hardware et commencer celle de la partie software</p> <p>Réalisation du software selon priorités de la séance précédente</p>	<ul style="list-style-type: none"> - Finir la documentation sur la partie hardware - Réaliser les diagrammes d'états du software - Mettre en lien les différentes parties du software pour pouvoir réaliser une démonstration - Etablir une communication avec le serveur externe
Prochaine réunion planifiée le :	
Mercredi 20 septembre 2023 15h00 – 16h00 Salle R110 ES	
Destinataires du procès-verbal :	
Grégoire Rossier Doyen de l'ETML-ES	Philippe Bovey Maître de diplôme
Lausanne, le 19 septembre 2023	

Diplôme SLO
 SANTOS Miguel
 2312 Badge pour place de travail

Procès-verbal du mercredi 20 septembre 2023

Présences	État des lieux
<ul style="list-style-type: none"> Mr. Santos Miguel Mr. Bovey Philippe 	Programmation du module RFID réalisé. Configuration et programmation du module ESP32.
Problèmes rencontrés	Solutions proposées
<ul style="list-style-type: none"> Problèmes de « exception générale » sur le PIC32 Problèmes de communication avec les modules UART. 	Analyse et débogage approfondi. [Problèmes résolus]
Décisions prises	Objectifs jusqu'à prochaine réunion
Réaliser la programmation du PCB jusqu'au vendredi et se concentrer sur la rédaction du rapport ensuite jusqu'au rendu. Préparer une démonstration pour les portes ouvertes	Rendu du rapport le 26/09 Rendu de l'affiche + résumé le 27/09 Portes ouvertes le 29/09
Prochaine réunion planifiée le :	
N/A Projet terminé	
Destinataires du procès-verbal :	
Grégoire Rossier Doyen de l'ETML-ES	Philippe Bovey Maître de diplôme
Lausanne, le 21 septembre 2023	

2312 Badge pour place de travail



Fichier : 2312_SchemaBloc.SchDoc

Date : 26.09.2023 Version : 1.0

Heure : 08:37:55 Auteur : Miguel Santos



Projet : 2312_BadgePlaceTravail.PrjPcb

Chemin : C:\microchip\harmony\v2_06\apps\2312_BadgePlace\hard\2312_BadgePlaceTravail\2312_SchemaBloc.SchDoc

Modif. a
b
c
dNo :
Nb feuillets 7 Feuille n° 1

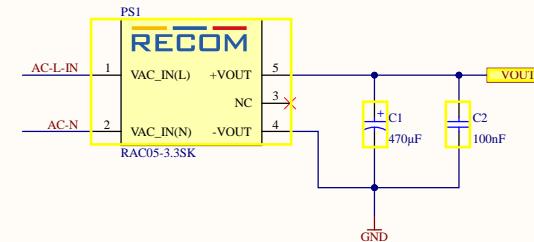
230VAC / Alimentation

A

Connecteurs 230VAC



Convertisseur AC/DC

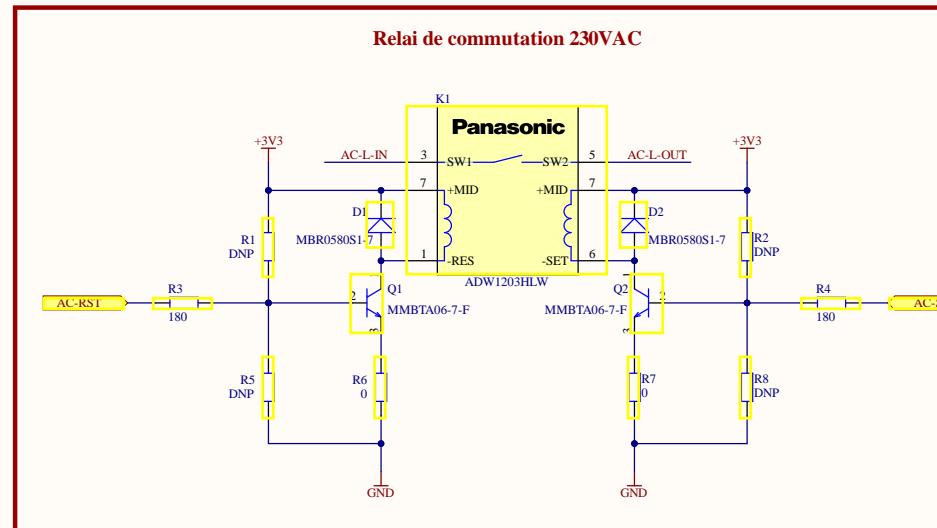


B

C

D

Relai de commutation 230VAC



Fichier : 2312_Puissance.SchDoc

Date : 26.09.2023 Version : 1.0

Heure : 08:37:55 Auteur : Miguel Santos

ETNL-ES
Avenue Recordon 1
1004 Lausanne
Switzerland

Projet : 2312_BadgePlaceTravail.PrjPcb

Chemin : C:\microchip\harmony\v2_06\apps\2312_BadgePlace\hard\2312_BadgePlaceTravail\2312_Puissance.SchDoc

a

b

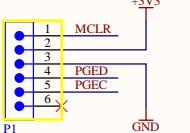
c

d

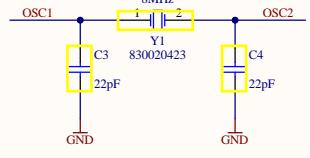
No :
Nb feuillets 7 Feuille n° 2

PIC32

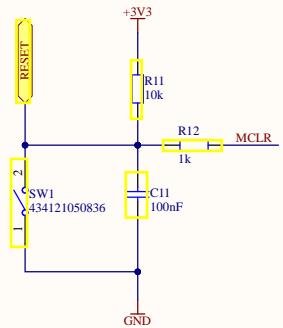
Port de programmation



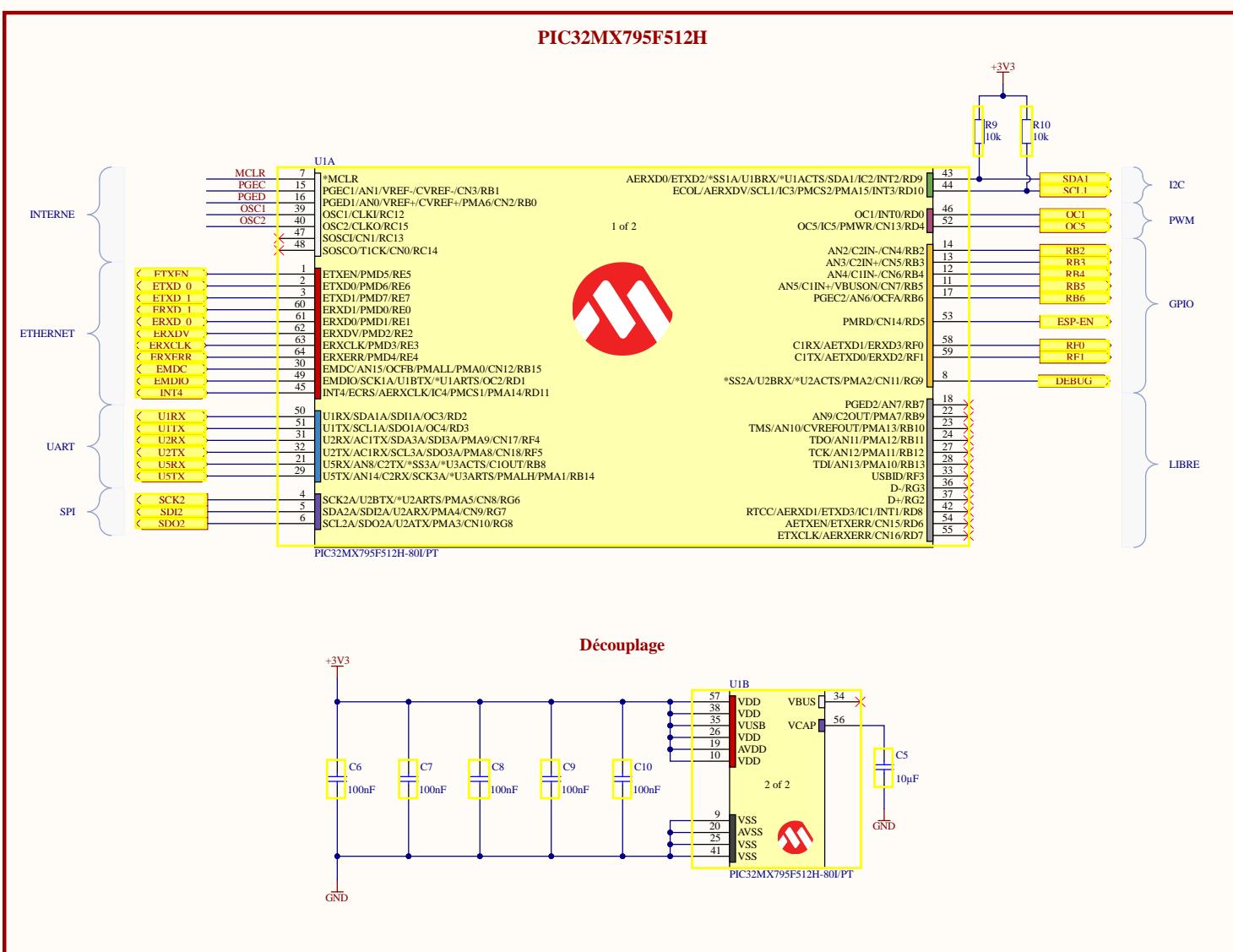
Quartz externe



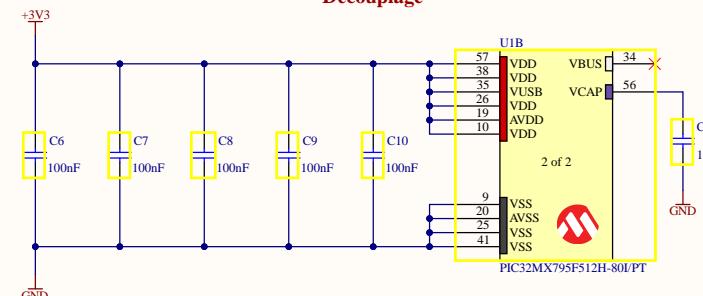
Bouton de redémarrage



PIC32MX795F512H



Découplage



Fichier : 2312_PIC32.SchDoc

Date : 26.09.2023 Version : 1.0

Heure : 08:37:55 Auteur : Miguel Santos

ETNL-ES
Avenue Recordon 1
1004 Lausanne
Switzerland

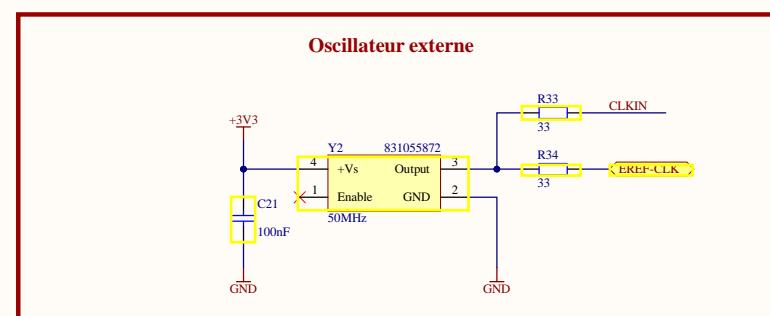
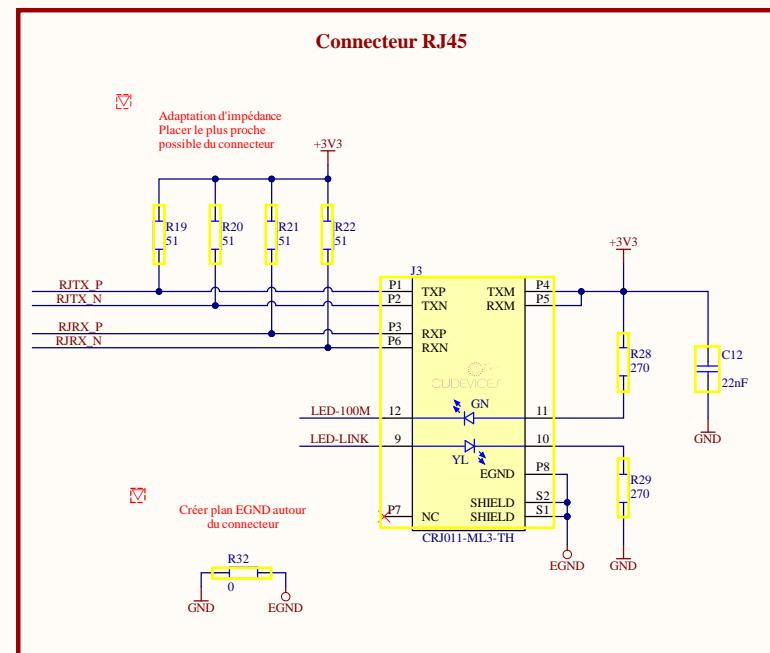
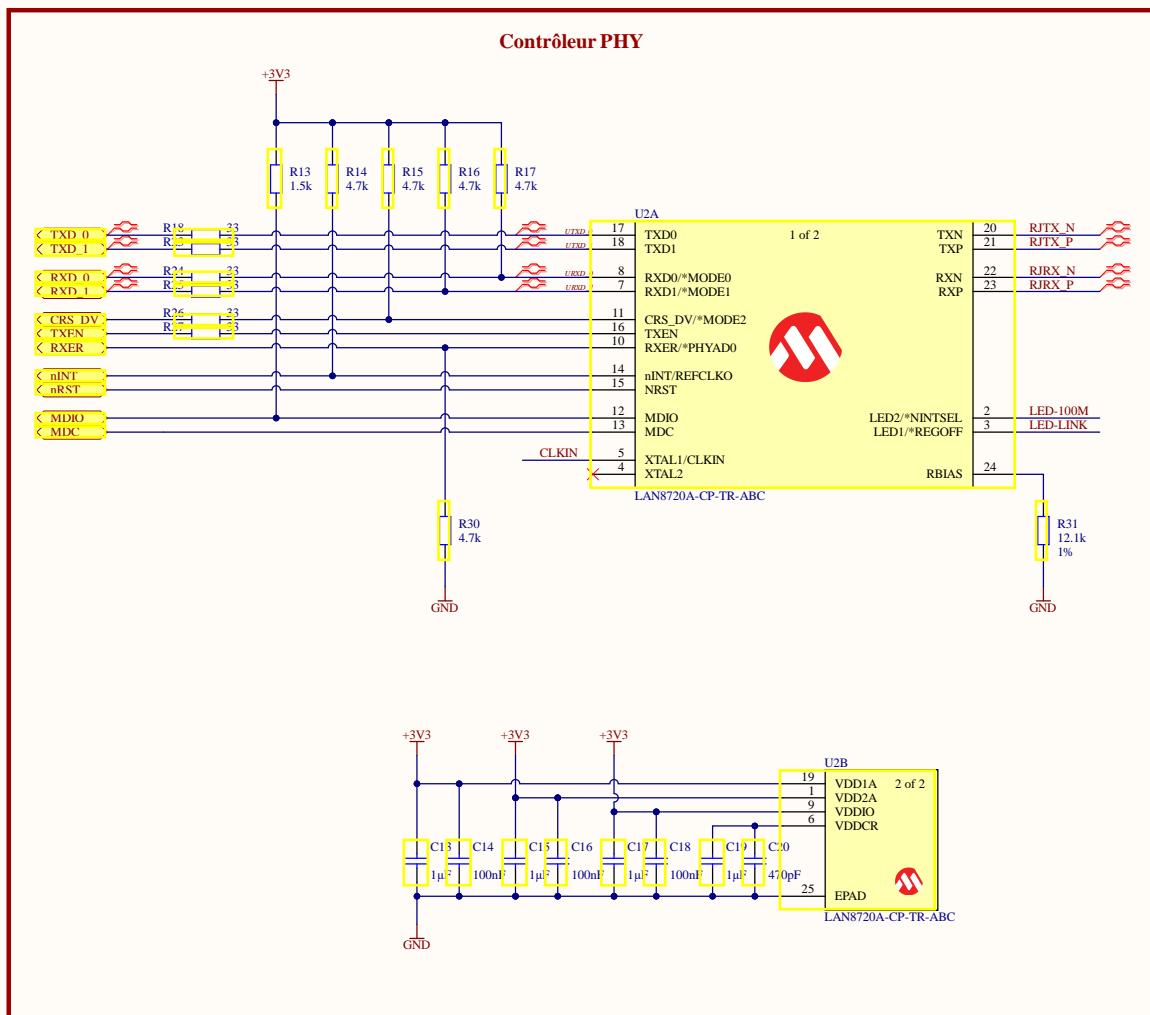
Projet : 2312_BadgePlaceTravail.PrjPcb

Chemin : C:\microchip\harmony\v2_06\apps\2312_BadgePlace\hard\2312_BadgePlaceTravail\2312_PIC32.SchDoc

a
b
c
d

No :
Nb feuilles 7 Feille n° 3

Ethernet

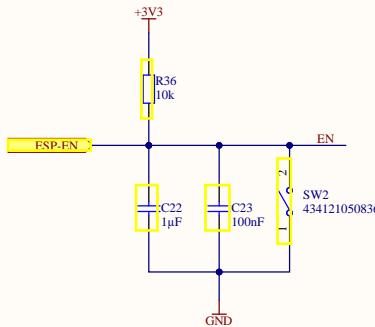


Fichier :	2312_Ethernet.SchDoc	Modif.	a
Date :	26.09.2023		b
Heure :	08:37:56		c
	Auteur : Miguel Santos		d
 Avenue Recordon 1 1004 Lausanne Switzerland	Projet : 2312_BadgePlaceTravail.PrjPcb		No : Nb feuilles Feuille n° 7 4
Chemin : C:\microchip\harmony\v2_06\apps\2312_BadgePlace\hard\2312_BadgePlaceTravail\2312_Ethernet.SchDoc			

ESP32

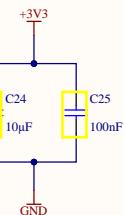
A

Bouton de redémarrage

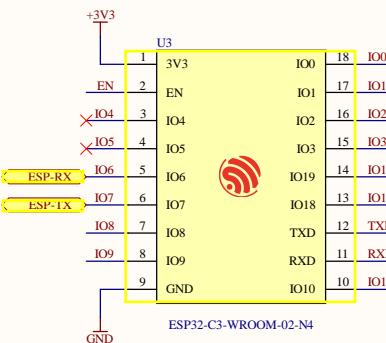


B

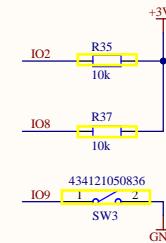
Découplage



ESP32-C3-WROOM-02-N4



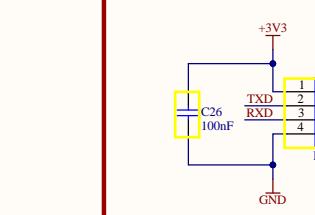
Sélection de mode



Pour lancer le mode "Download"
 1) Garder appuyé sur ce bouton
 2) Appuyer sur le bouton de redémarrage
 3) Les deux boutons peuvent ensuite être relâchés

C

Port de programmation



D

Fichier : 2312_ESP32.SchDoc

Modif.

a

b

c

d

Date : 26.09.2023

Version : 1.0

Heure : 08:37:57

Auteur : Miguel Santos

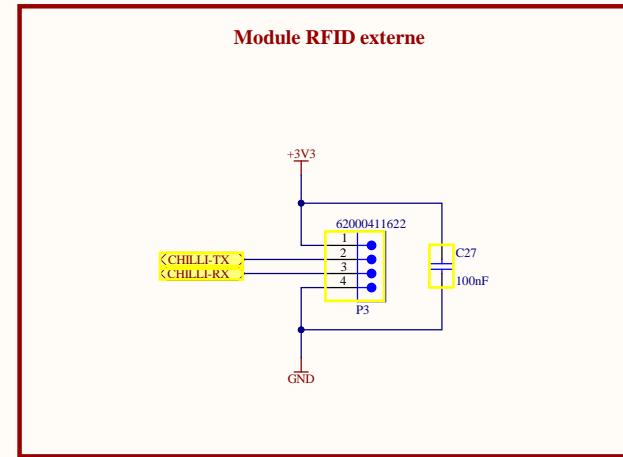
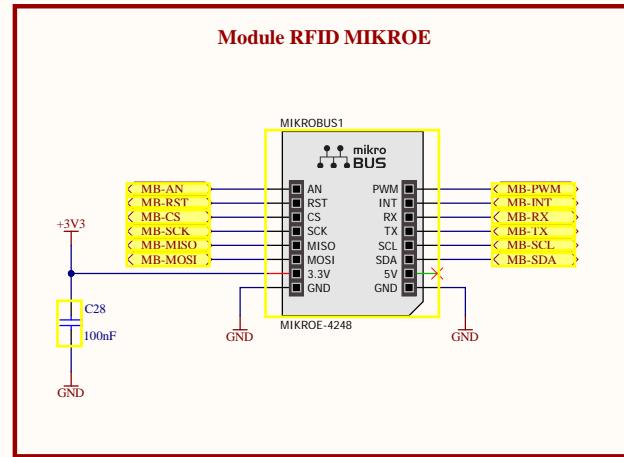
ETNL-ES
 Avenue Recordon 1
 1004 Lausanne
 Switzerland

Projet : 2312_BadgePlaceTravail.PrjPcb

 No :
 Nb feuillets 7 Feuille n° 5

Chemin : C:\microchip\harmony\v2_06\apps\2312_BadgePlace\hard\2312_BadgePlaceTravail\2312_ESP32.SchDoc

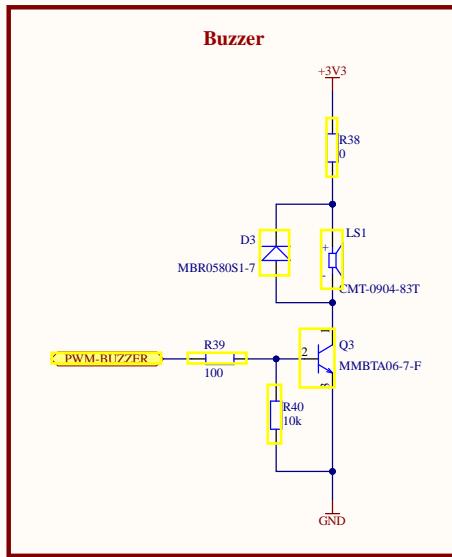
RFID



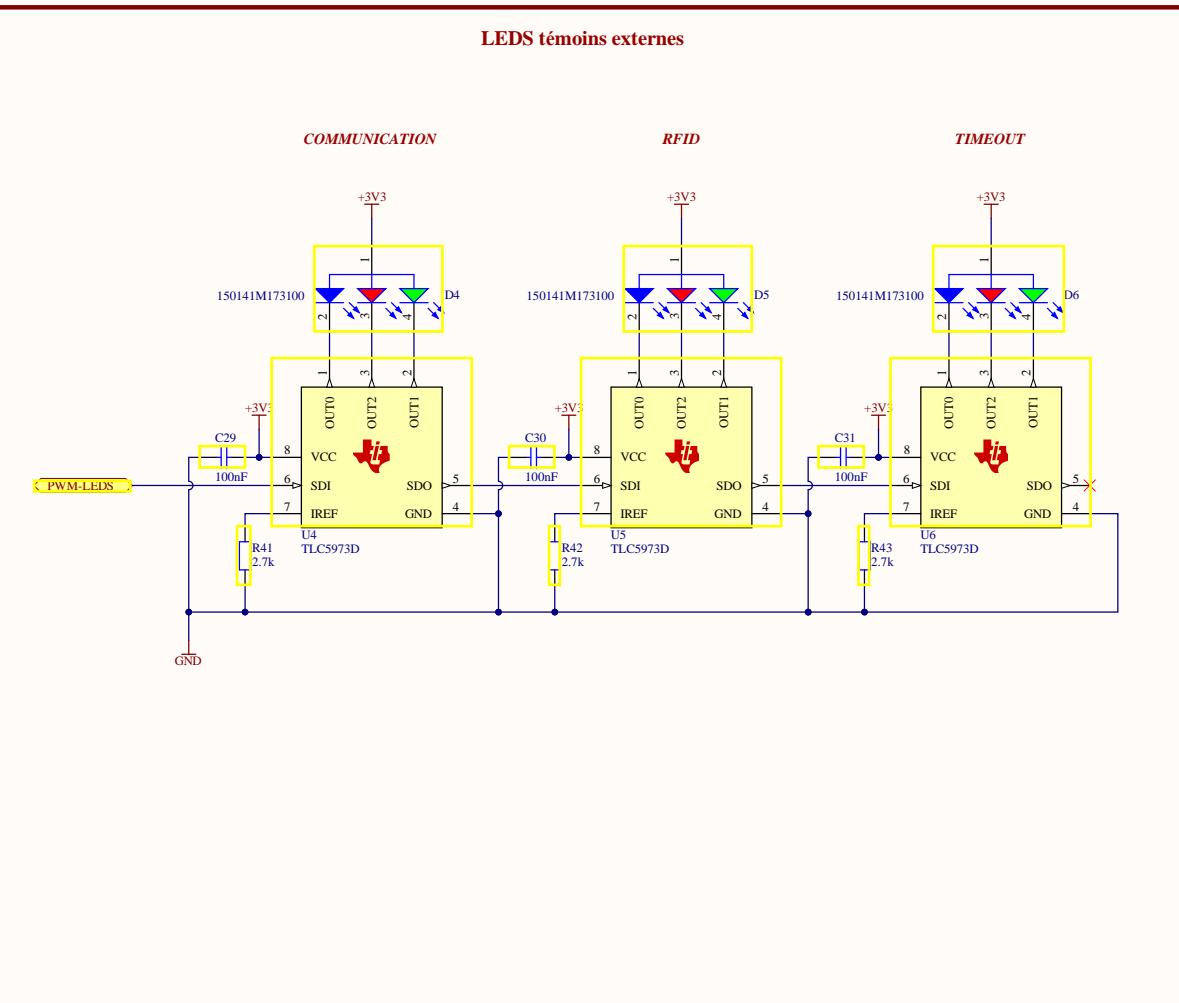
Fichier : 2312_RFID.SchDoc	a
Date : 26.09.2023	b
Version : 1.0	c
Heure : 08:37:57	d
Auteur : Miguel Santos	
ETNL-ES Avenue Recordon 1 1004 Lausanne Switzerland	Projet : 2312_BadgePlaceTravail.PrjPcb
Chemin : C:\microchip\harmony\v2_06\apps\2312_BadgePlace\hard\2312_BadgePlaceTravail\2312_RFID.SchDoc	No : Nb feuillets 7 Feuille n° 6

Interface

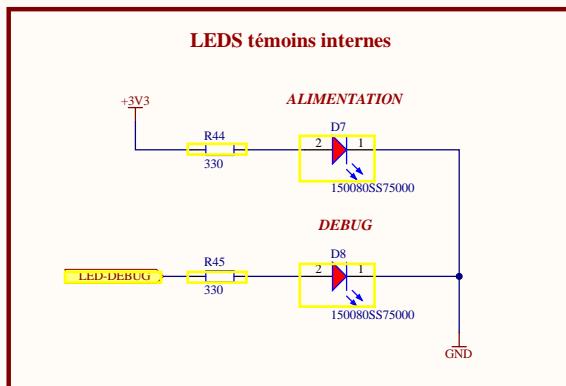
A



B



C



D

Fichier : 2312_Interface.SchDoc	a
Date : 26.09.2023	b
Heure : 08:37:58	c
Auteur : Miguel Santos	d
ETNL-ES Avenue Recordon 1 1004 Lausanne Switzerland	Modif.
Projet : 2312_BadgePlaceTravail.PrjPcb	Nb feuillets 7
Chemin : C:\microchip\harmony\v2_06\apps\2312_BadgePlace\hard\2312_BadgePlaceTravail\2312_Interface.SchDoc	Feuille n° 7

A

B

C

D

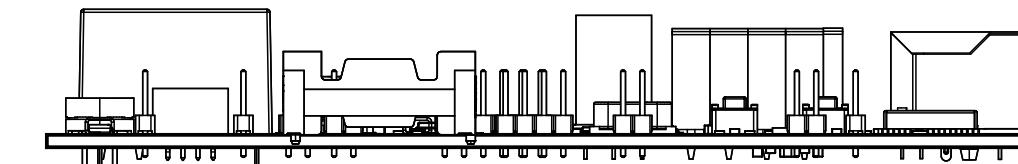
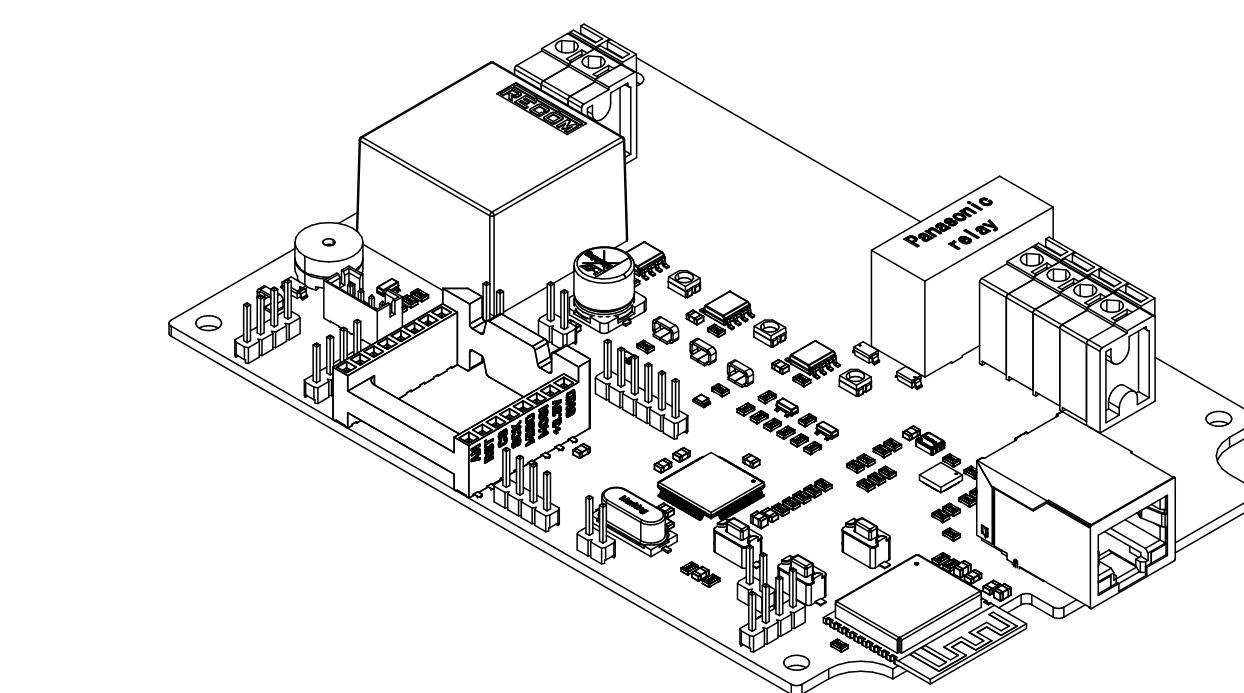
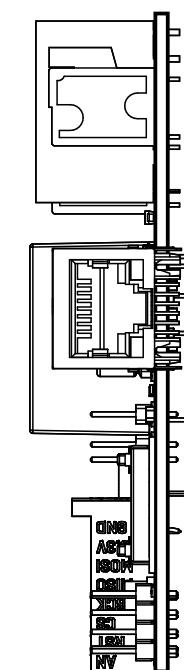
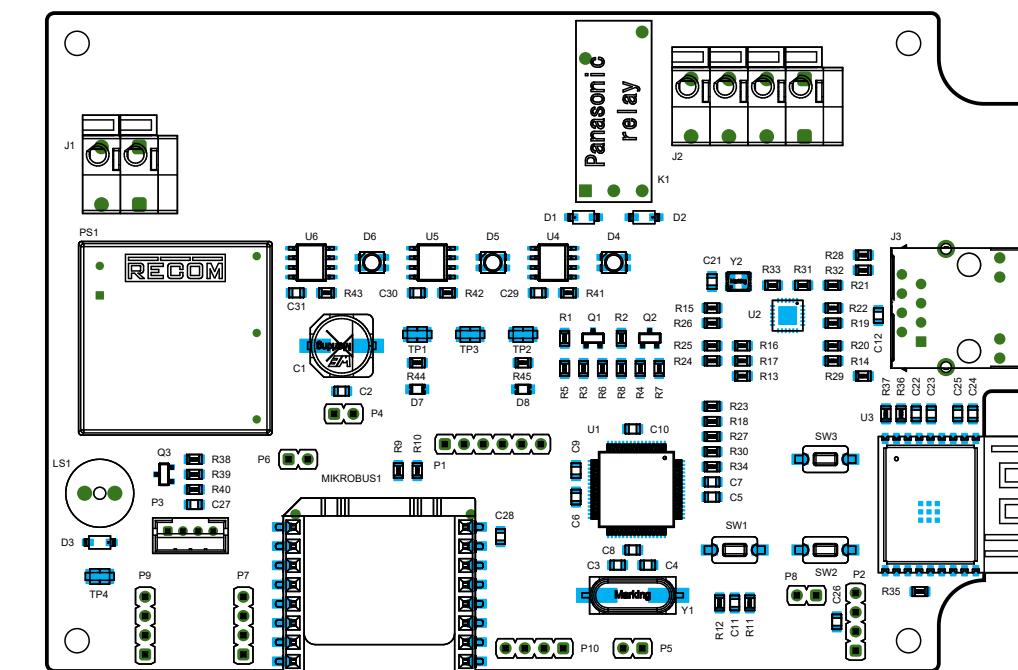
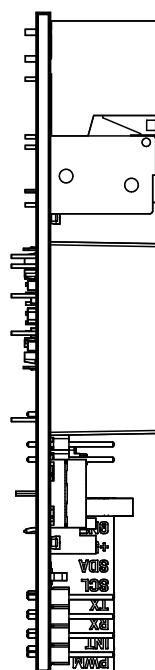
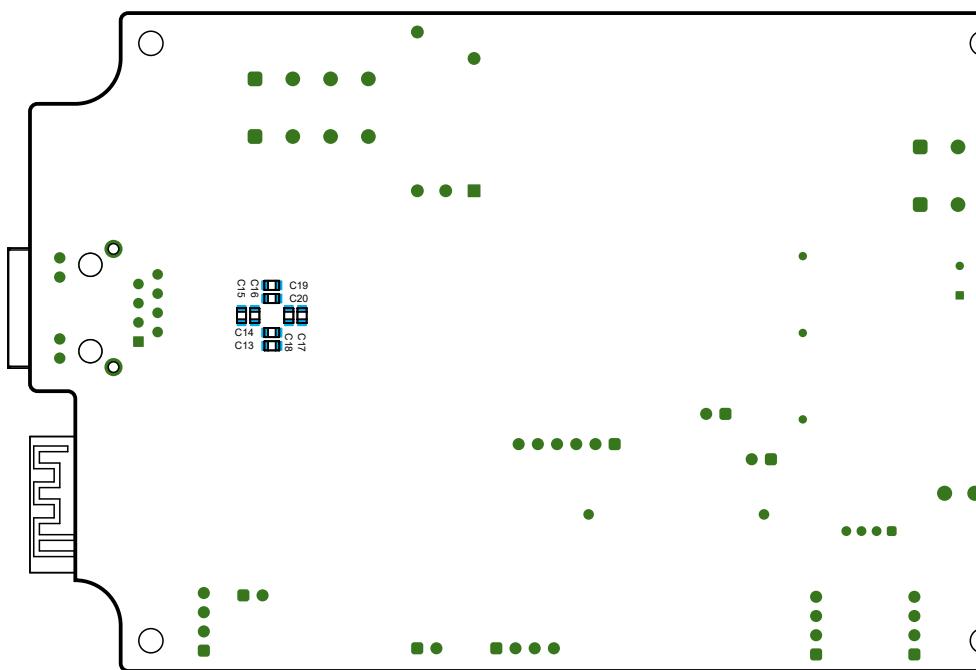
E

F

G

H

1



Scale (1:1)

Fichier : 2312_BadgePlaceTravail.PCBDwf

Date 26.09.2023

Version 1.0

Heure 08:30

Auteur Miguel Santos

ETNL-ES
Avenue Recordon 1
1004 Lausanne
Switzerland

Projet 2312_BadgePlaceTravail.PrjPcb

Chemin C:\microchip\harmony\v2_06\apps\2312_BadgePlace\hard\2312_BadgePlaceTravail\2312_BadgePlaceTr

a	=Modifa
b	=Modifb
c	=Modifc
d	=Modifd

No :	
Nb feuillets	5
Feuille n°	1

A

B

C

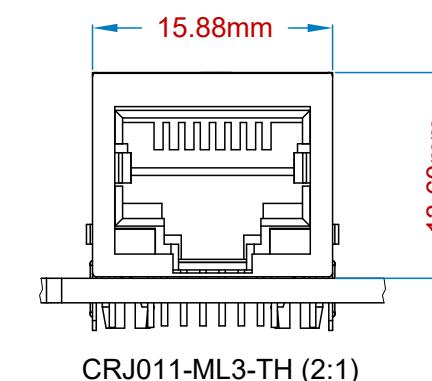
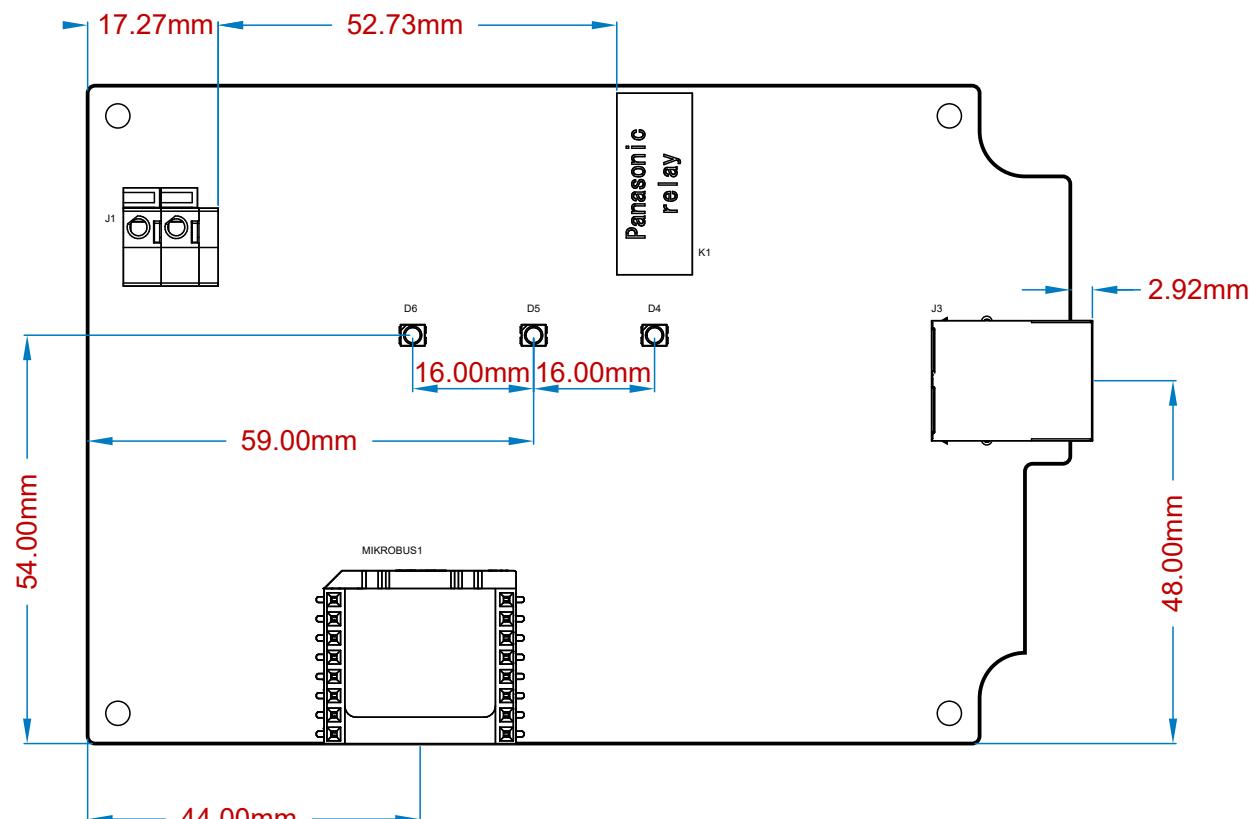
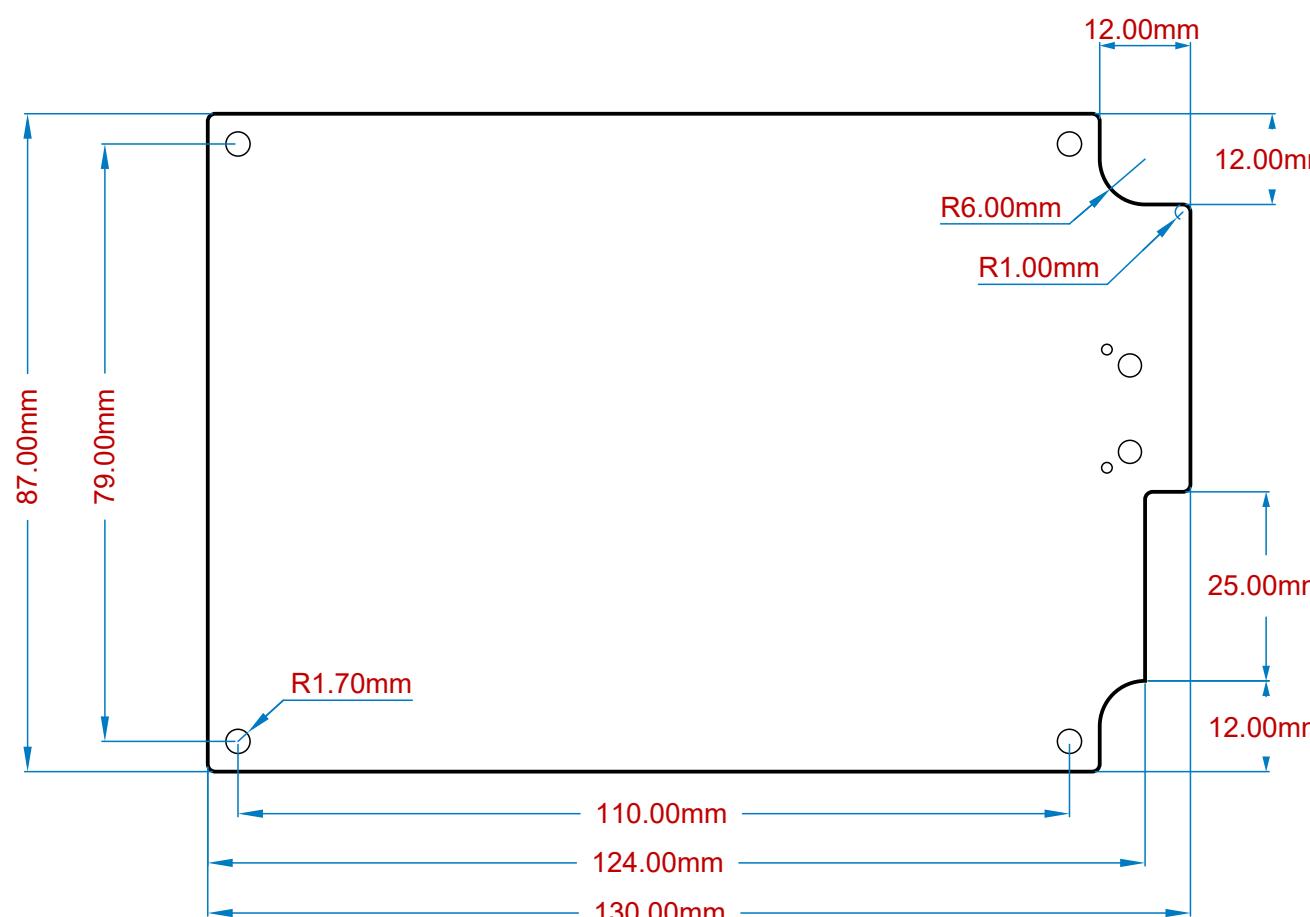
D

E

F

G

H



Scale (1:1)

Fichier : 2312_BadgePlaceTravail.PCBDwf

Date 26.09.2023

Version 1.0

Heure 08:30

Auteur Miguel Santos

a	=Modifa
b	=Modifb
c	=Modifc
d	=Modifd

ETNL-ES
Avenue Recordon 1
1004 Lausanne
Switzerland

Projet 2312_BadgePlaceTravail.PrjPcb

No :	
Nb feuillets	5
Feuille n°	2

Chemin C:\microchip\harmony\v2_06\apps\2312_BadgePlace\hard\2312_BadgePlaceTravail\2312_BadgePlaceTravai

A

B

C

D

E

F

G

H

A

B

C

D

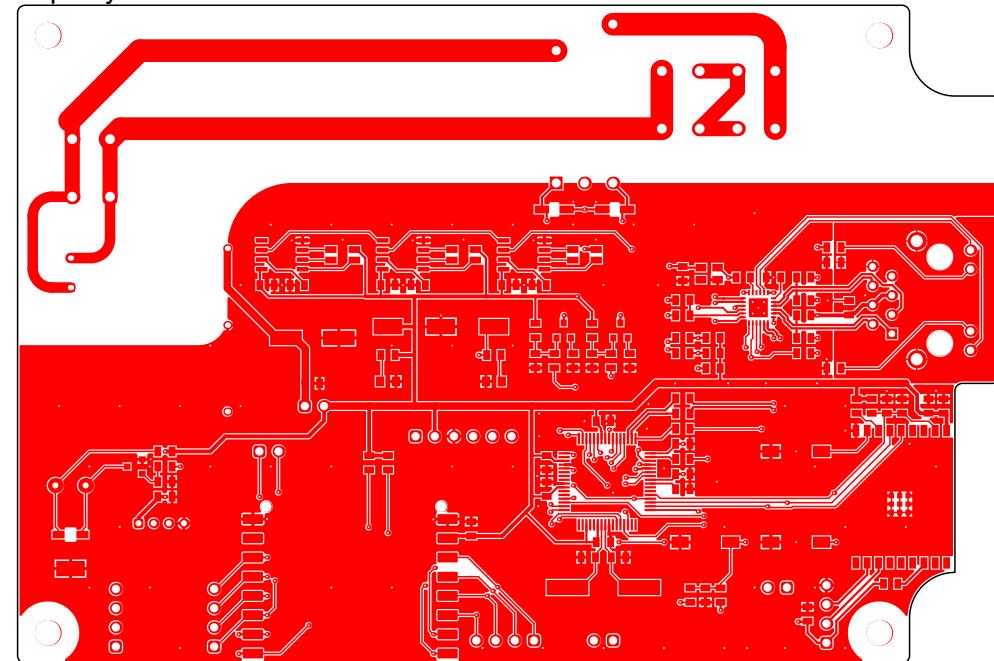
E

F

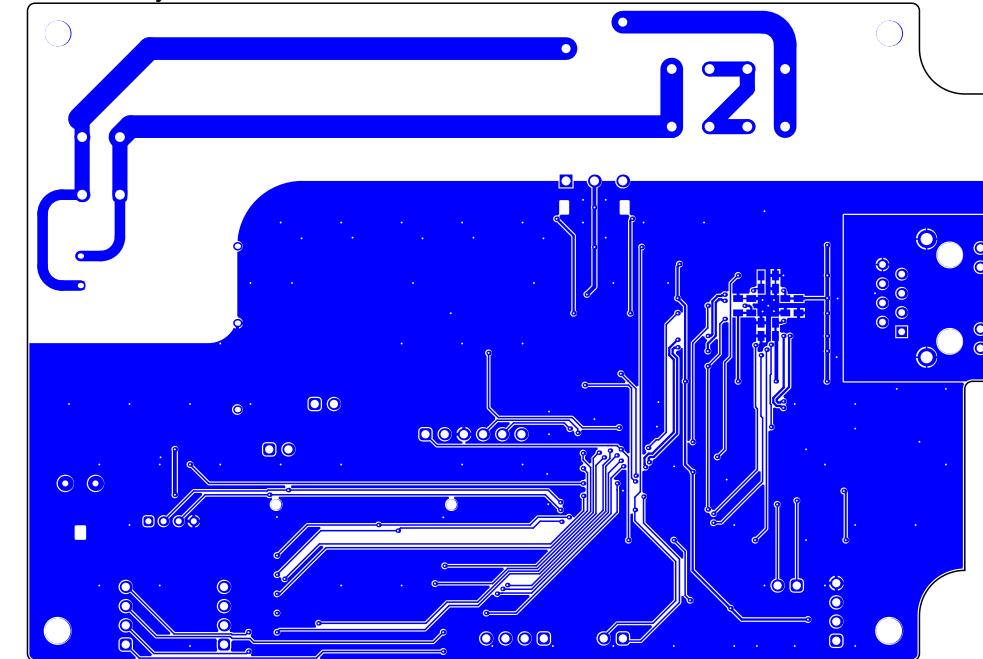
G

H

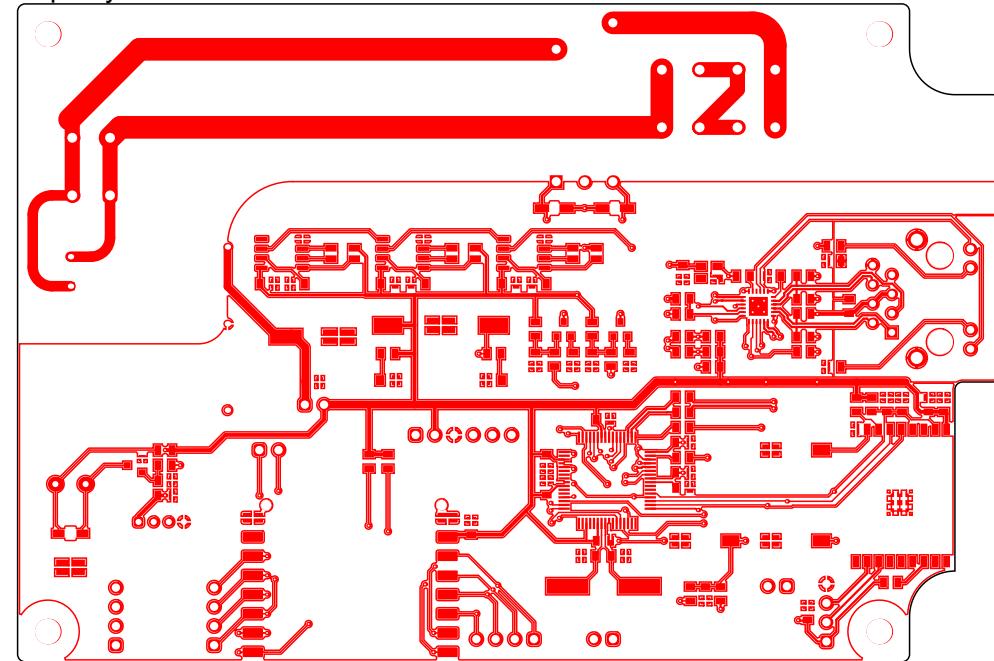
Top Layer



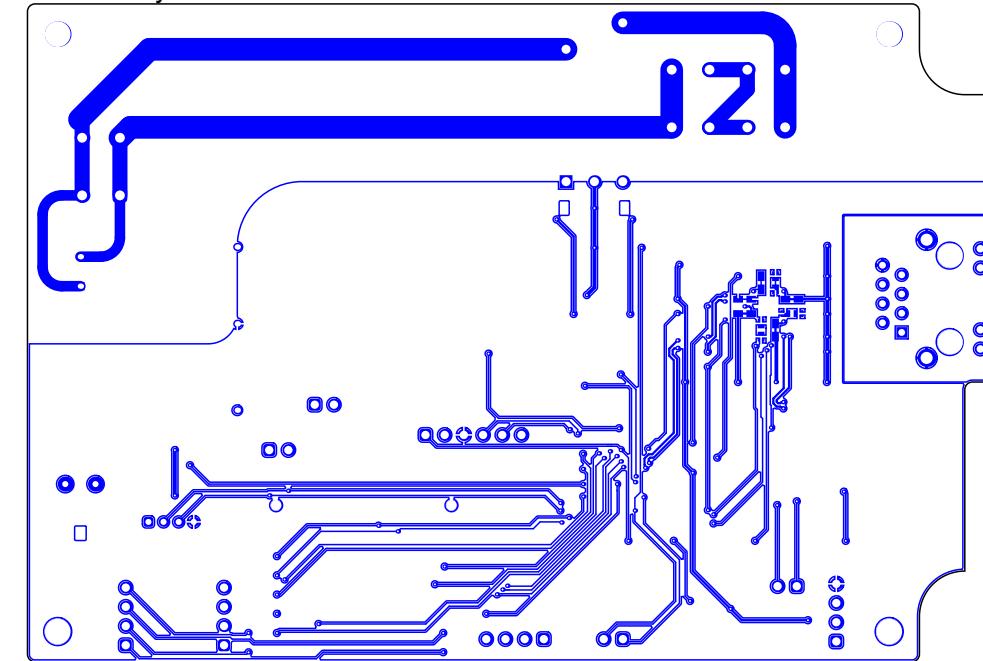
Bottom Layer



Top Layer



Bottom Layer



Scale (1:1)

Fichier : 2312_BadgePlaceTravail.PCBDwf

Date 26.09.2023

Version 1.0

Heure 08:30

Auteur Miguel Santos

a	=Modifa
b	=Modifb
c	=Modifc
d	=Modifd

ETNL-ES
Avenue Recordon 1
1004 Lausanne
Switzerland

Projet 2312_BadgePlaceTravail.PrjPcb

No :	
Nb feuillets	5
Feuille n°	3

Chemin C:\microchip\harmony\v2_06\apps\2312_BadgePlace\hard\2312_BadgePlaceTravail\2312_BadgePlaceTr

A

B

C

D

E

F

G

H

A

B

C

D

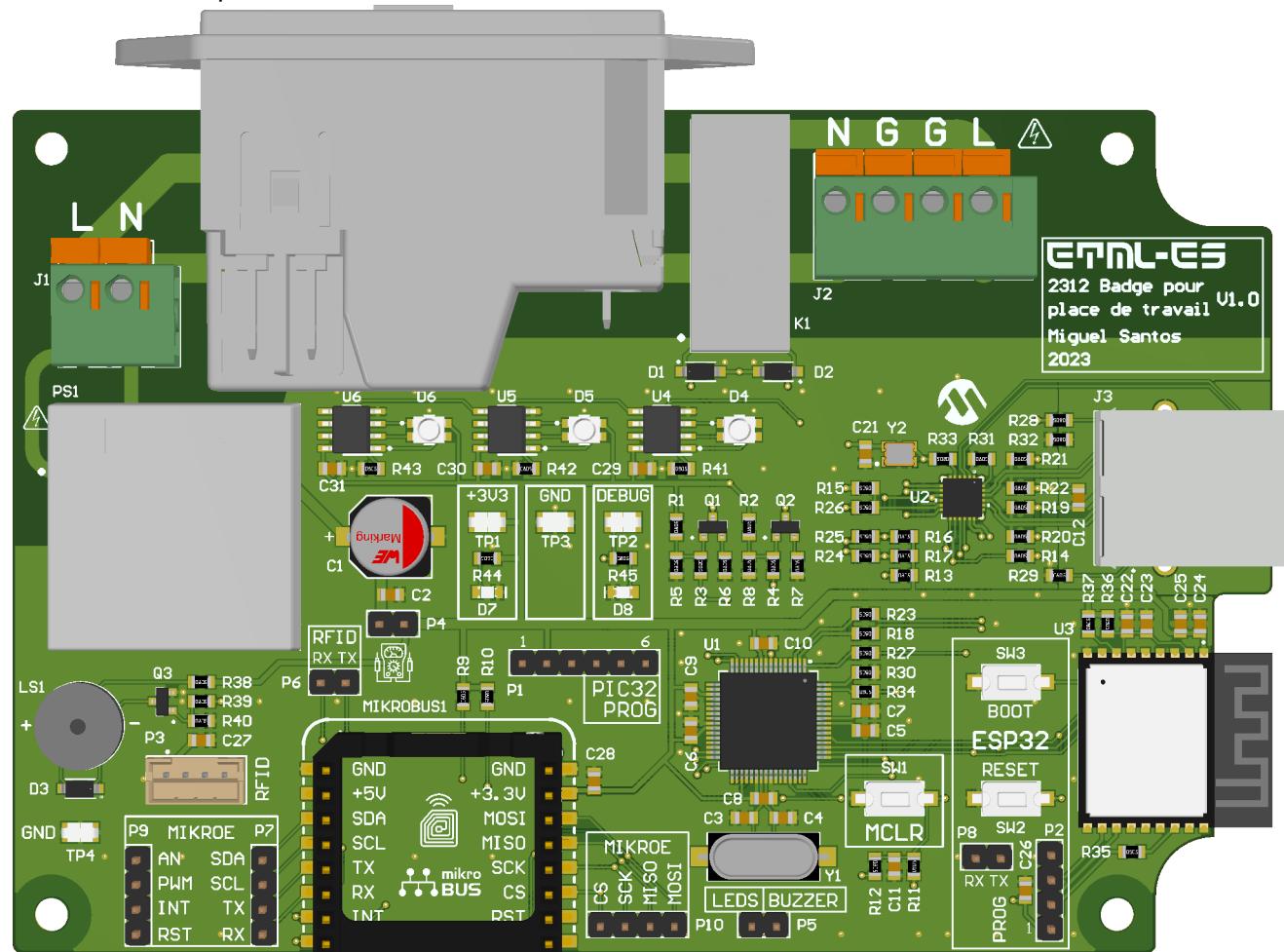
E

F

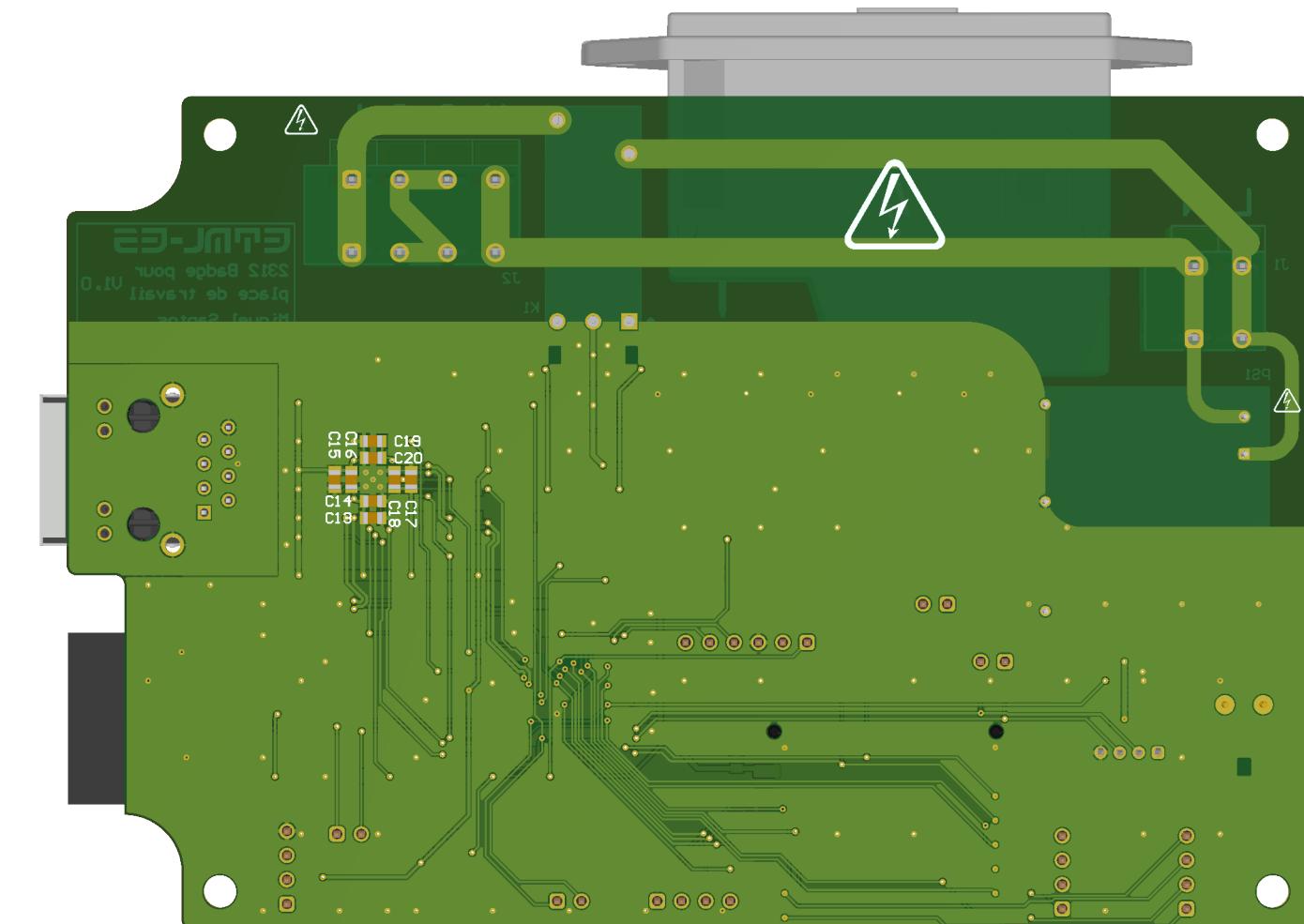
G

H

Realistic View : Top



Realistic View : Bottom



Fichier : 2312_BadgePlaceTravail.PCBDwf

Date 26.09.2023

Version 1.0

Heure 08:30

Auteur Miguel Santos

ETNL-ES
Avenue Recordon 1
1004 Lausanne
Switzerland

Projet 2312_BadgePlaceTravail.PrjPcb

No :
Nb feuillets 5 Feuille n° 4

Chemin C:\microchip\harmony\v2_06\apps\2312_BadgePlace\hard\2312_BadgePlaceTravail\2312_BadgePlaceTr

Modif.
a = Modifa
b = Modifb
c = Modifc
d = Modifd

Bill Of Materials

Line #	Designator	Quantity	Description	Manufacturer 1	Manufacturer Part Number	Supplier 1	Supplier Unit Price 1	Supplier Subtotal 1	Stock ES ?
1	D7, D8	2	WL-SMCW Mono-color Chip LED Waterclear, size 0805, Super Red, 1.9V, 140°	Wurth Electronics	150080SS75000	Digi-Key	0.19	0.38	
	J1	1	WR-TBL Series 401B Vertical Entry, THT, pitch 5mm, 2p	Wurth Electronics	691401700002B	Digi-Key	1.06	1.06	
	J2	1	WR-TBL Series 401B Vertical Entry, THT, pitch 5mm, 4p	Wurth Electronics	691401700004B	Digi-Key	2.12	2.12	
	J3	1	Modular Jack	CUI Devices		Digi-Key	5.48	5.48	
	MIKROBUS1	1	mikroBUS socket	mikroElektronika		Mouser	0.9	0.9	
	R44, R45	2	SMD Resistor 0805, 330Ω						Y
	U1	1	No Description Available	Microchip		Digi-Key	11.89	11.89	
	U2	1	No Description Available	Microchip		Digi-Key	1.29	1.29	
	U4, U5, U6	3	No Description Available	Texas Instruments		Digi-Key	1.33	3.99	
	C1	1	WCAP-PSHP Aluminum Polymer Capacitor, V-Chip, SMT, D8 x H7.7mm, 470μF, 6.3V	Wurth Electronics	875115150005	Digi-Key	1.06	1.06	
2	C2, C6, C7, C8, C9, C10, C11, C14, C16, C18, C21, C23, C25, C26, C27, C28, C29, C30, C31	19	WCAP-CSGP Multilayer Ceramic Chip Capacitor, General Purpose, size 0805, X7R, 100nF, 10VDC	Wurth Electronics	885012207016	Digi-Key	0.047	0.893	
	C3, C4	2	WCAP-CSGP Multilayer Ceramic Chip Capacitor, General Purpose, size 0805, NP0, 22pF, 10VDC	Wurth Electronics	885012007002	Digi-Key	0.1	0.2	
	C5	1	WCAP-CSGP Multilayer Ceramic Chip Capacitor, General Purpose, size 0805, X7R, 10μF, 6.3VDC	Wurth Electronics	885012207003	Digi-Key	0.56	0.56	
	C12	1	WCAP-CSGP Multilayer Ceramic Chip Capacitor, General Purpose, size 0805, X7R, 22nF, 10VDC	Wurth Electronics	885012207013	Digi-Key	0.1	0.1	
	C13, C15, C17, C19, C22	5	WCAP-CSGP Multilayer Ceramic Chip Capacitor, General Purpose, size 0805, X7R, 1μF, 10VDC	Wurth Electronics	885012207022	Digi-Key	0.1	0.5	
	C20	1	WCAP-CSGP Multilayer Ceramic Chip Capacitor, General Purpose, size 0805, NP0, 470pF, 10VDC	Wurth Electronics	885012007007	Digi-Key	0.1	0.1	
	C24	1	WCAP-CSGP Multilayer Ceramic Chip Capacitor, General Purpose, size 0805, X5R, 10μF, 6.3VDC	Wurth Electronics	885012107004	Digi-Key	0.2	0.2	
	D4, D5, D6	3	WL-SFTW Full-color TOP LED Waterclear, size 3528, Red & Green & Blue, 120°	Wurth Electronics	150141M173100	Digi-Key	0.5	1.5	
	P1	1	WR-PHD Pin Header, THT, Vertical, pitch 2.54mm, 1 Row, 6P	Wurth Electronics	61300611121	Digi-Key	0.35	0.35	Y
	P2, P7, P9, P10	4	WR-PHD Pin Header, THT, Vertical, pitch 2.54mm, 1 Row, 4P	Wurth Electronics	61300411121	Digi-Key	0.19	0.76	Y
	P3	1	WR-WTB Male Vertical Shrouded Header, pitch 2mm, 4pins	Wurth Electronics	62000411622	Digi-Key	0.21	0.21	
	P4, P5, P6, P8	4	WR-PHD Pin Header, THT, Vertical, pitch 2.54mm, 1 Row, 2P	Wurth Electronics	61300211121	Digi-Key	0.13	0.52	Y
	SW1, SW2, SW3	3	WS-TASV SMT Tact Switch, L6 x W3.5 x H5mm, 360g, SPST-NO, 12V, 50mA	Wurth Electronics	434121050836	Digi-Key	0.42	1.26	
	Y1	1	WE-XTAL Quartz Crystal, SMT, HC49/4HSMX, 8MHz, +/-30ppm	Wurth Electronics	830020423	Digi-Key	0.37	0.37	
	Y2	1	WE-SPXO Simple Packaged Quartz Oscillator, SMT, CFPS-39, 50MHz, 3.3V	Wurth Electronics	831055872	Digi-Key	1.62	1.62	
	R6, R7	2	SMD Resistor 0805, 2Ω	Vishay Dale		Digi-Key	0.1	0.2	
	R31	1	SMD Resistor 0805, 12.1kΩ	Vishay		Digi-Key	0.1	0.1	
3	X1	1	Power RCPT IEC320	Schurter		Digi-Key	1.15	1.15	
	PS1	1	5W AC/DC-Converter 'POWERLINE' 1x1 4kV reg	Recom		Digi-Key	11.63	11.63	
	K1	1	RELAY GEN PURPOSE SPST 16A 3V	Panasonic	ADW1203HLW	Digi-Key	6.01	6.01	
	TP1, TP2, TP3, TP4	4	Test Point, 1 Position SMD, RoHS, Tape and Reel	Keystone Electronics		Digi-Key	0.34	1.36	
	U3	1	WiFi Modules (802.11) (Engineering Samples) SMD module, ESP32-C3, 4MB SPI flash, PCB antenna, -40 C +85 C	Espressif Systems		Digi-Key	1.9	1.9	
	UX1	1	CHILLI UART B1 READER BOARD	Eccel		Digi-Key	40.02	40.02	
	Q1, Q2, Q3	3	Bipolar (BJT) Transistor NPN 80V 500mA 100MHz 225mW Surface Mount SOT-23-3 (TO-236)	Diodes		Digi-Key	0.22	0.66	
	D1, D2, D3	3	Diode Schottky, 500mA, 80V	Diodes		Digi-Key	0.29	0.87	
	LS1	1	Buzzer Magnetic, 15Ω	CUI Devices		Digi-Key	0.83	0.83	
	LP1, LP2, LP3	3	LIGHT PIPE CLEAR FLEXIBLE 6"	Bivar		Digi-Key	1.99	5.97	
	R1, R8	2	SMD Resistor 0805, 1.4kΩ						Y
	R2, R5	2	SMD Resistor 0805, DNP						Y
	R3, R4, R39	3	SMD Resistor 0805, 140Ω						Y
	R9, R10, R11, R35, R36, R37, R40	7	SMD Resistor 0805, 10kΩ						Y
	R12	1	SMD Resistor 0805, 1kΩ						Y
	R13	1	SMD Resistor 0805, 1.5kΩ						Y
	R14, R15, R16, R17, R30	5	SMD Resistor 0805, 4.7kΩ						Y
	R19, R20, R21, R22	4	SMD Resistor 0805, 51Ω						Y
	R18, R23, R24, R25, R26, R27, R33, R34	8	SMD Resistor 0805, 33Ω						Y
	R28, R29	2	SMD Resistor 0805, 270Ω						Y
	R32, R38	2	SMD Resistor 0805, 0Ω						Y
	R41, R42, R43	3	SMD Resistor 0805, 2.6kΩ						Y

Report Generated From Altium Designer

Name	Priority	Enabled	Type	Category	Scope	Attributes
AssemblyTestpoint	1	True	Assembly Testpoint Style	Testpoint	All	Under Comp - Allow Sides - Top, Bottom Pref Size = 1.524mm Pref Hole Size = 0.813mm Using Grid = Yes Grid = 0.025mm Grid Tolerance = 0mm
AssemblyTestPointUsage	1	True	Assembly Testpoint Usage	Testpoint	All	Testpoint - One Required Multiple - Not Allowed
FabricationTestpoint	1	True	Fabrication Testpoint Style	Testpoint	All	Under Comp - Allow Sides - Top, Bottom Pref Size = 1.524mm Pref Hole Size = 0.813mm Using Grid = Yes Grid = 0.025mm Grid Tolerance = 0mm
FabricationTestPointUsage	1	True	Fabrication Testpoint Usage	Testpoint	All	Testpoint - One Required Multiple - Not Allowed
SMDEntry	2	True	SMD Entry	SMT	All	Side = Allowed Corner = Allowed Any Angle = Not Allowed Ignore First Corner = Allowed
SMDEntry-GND	1	True	SMD Entry	SMT	InNet('GND')	Side = Allowed Corner = Not Allowed Any Angle = Not Allowed Ignore First Corner = Allowed
DiffPairsRouting	2	True	Differential Pairs Routing	Routing	All	Pref Gap = 0.2mm Min Gap = 0.15mm Max Gap = 0.25mm Pref Width = 0.25mm Min Width = 0.15mm Max Width = 0.25mm
EthernetDiffPair	1	True	Differential Pairs Routing	Routing	InDifferentialPairClass('All Differential Pairs')	Pref Gap = 0.151mm Min Gap = 0.151mm Max Gap = 0.151mm Pref Width = 0.3mm Min Width = 0.3mm Max Width = 0.3mm
Fanout_BGA	1	True	Fanout Control	Routing	IsBGA	Style - Auto Direction - Alternating In and Out Via Grid = 0.025mm
Fanout_Default	5	True	Fanout Control	Routing	All	Style - Auto Direction - Alternating In and Out Via Grid = 0.025mm
Fanout_LCC	2	True	Fanout Control	Routing	IsLCC	Style - Auto Direction - Alternating In and Out Via Grid = 0.025mm
Fanout_Small	4	True	Fanout Control	Routing	(CompPinCount < 5)	Style - Auto Direction - Out Then In Via Grid = 0.025mm
Fanout_SOIC	3	True	Fanout Control	Routing	IsSOIC	Style - Auto Direction - Alternating In and Out Via Grid = 0.025mm
RoutingCorners	1	True	Routing Corners	Routing	All	Style - 45 Degree Min Setback = 2.5mm Max Setback = 2.5mm
RoutingLayers	1	True	Routing Layers	Routing	All	TopLayer - Enabled BottomLayer - Enabled
RoutingPriority	1	True	Routing Priority	Routing	All	Priority = 0
RoutingTopology	1	True	Routing Topology	Routing	All	Topology - Shortest
RoutingViasES	1	True	Routing Via Style	Routing	All	Pref Size = 0.51mm Pref Hole Size = 0.25mm
Width-230VAC	1	True	Width	Routing	InNetClass('AC-230V')	Pref Width = 1.5mm Min Width = 1mm Max Width = 10mm
Width-3V3	2	True	Width	Routing	InNet('+3V3')	Pref Width = 0.3mm Min Width = 0.25mm Max Width = 1mm
Width-xTW	3	True	Width	Routing	All	Pref Width = 0.2mm Min Width = 0.15mm Max Width = 1.5mm
GND-CONNECT	1	True	Polygon Connect Style	Plane	All - InNet('GND')	Advanced settings
PlaneClearance-IPI	1	True	Power Plane Clearance	Plane	All	Clearance = 0.25mm
PlaneConnect	1	True	Power Plane Connect Style	Plane	All	Style - Relief Connect Expansion = 0.175mm Width = 0.2mm Gap = 0.2mm # Entries = 4

Name	Priority	Enabled	Type	Category	Scope	Attributes
PolygonConnect-xTW	2	True	Polygon Connect Style	Plane	All - All	Advanced settings
ComponentClearance	1	True	Component Clearance	Placement	All - All	Horizontal Clearance = 0.1mm Vertical Clearance = 0.25mm
Height	1	True	Height	Placement	All	Pref Height = 12.5mm Min Height = 0mm Max Height = 25mm
PasteMaskExpansion	1	True	Paste Mask Expansion	Mask	All	Expansion = -0.025mm
SolderMaskExpansion	1	True	Solder Mask Expansion	Mask	All	Expansion = 0mm
BoardOutlineClearance	1	True	Board Outline Clearance	Manufacturing	All	Generic clearance = 0.25mm, and 9 value(s) for objects
HoleSize-PTH	1	True	Hole Size	Manufacturing	All	Min = 0.25mm Max = 10mm
HoleToHoleClearance	1	True	Hole To Hole Clearance	Manufacturing	All - All	Hole To Hole Clearance = 0.35mm
LayerPairs	1	True	Layer Pairs	Manufacturing	All	Layer Pairs - Enforce
Minimum IAR >0.45mm	1	True	Minimum Annular Ring	Manufacturing	((ObjectKind = 'Pad') OR (ObjectKind = 'Via')) And (Layer = 'MultiLayer') And (HoleDiameter > AsMM(0.45))	Min = 0.15mm
Minimum IAR <=0.45mm	2	True	Minimum Annular Ring	Manufacturing	((ObjectKind = 'Pad') OR (ObjectKind = 'Via')) And (Layer = 'MultiLayer') And (HoleDiameter <= AsMM(0.45))	Min = 0.125mm
MinimumSolderMaskSliver	1	True	Minimum Solder Mask Sliver	Manufacturing	All - All	Minimum Solder Mask Sliver = 0.08mm
NetAntennae	1	True	Net Antennae	Manufacturing	All	Net Antennae Tolerance = 2mm
SilkscreenOverComponentPads	1	True	Silk To Solder Mask Clearance	Manufacturing	IsPad - All	Silk To Solder Mask Clearance = 0.2mm
SilkToSilkClearance	1	True	Silk To Silk Clearance	Manufacturing	All - All	Silk to Silk Clearance = 0.2mm
Clearance 230VAC-230VAC	1	True	Clearance	Electrical	InNetClass('AC-230V') - InNetClass('AC-230V')	Clearance = 2.5mm
Clearance-230VAC-ALL	2	True	Clearance	Electrical	All - InNetClass('AC-230V')	Clearance = 4mm
Clearance-xPP-xTP-xTT	3	True	Clearance	Electrical	All - All	Clearance = 0.151mm
ShortCircuit	1	True	Short-Circuit	Electrical	All - All	Short Circuit - Not Allowed
UnpouredPolygon	1	True	Modified Polygon	Electrical	All	Allow modified - No Allow shelved - No
UnRoutedNet	1	True	Un-Routed Net	Electrical	All	(No Attributes)

```

1 ****
2 MPLAB Harmony Project Main Source File
3
4 Company:
5 Microchip Technology Inc.
6
7 File Name:
8 main.c
9
10 Summary:
11 This file contains the "main" function for an MPLAB Harmony project.
12
13 Description:
14 This file contains the "main" function for an MPLAB Harmony project. The
15 "main" function calls the "SYS_Initialize" function to initialize the state
16 machines of all MPLAB Harmony modules in the system and it calls the
17 "SYS_Tasks" function from within a system-wide "super" loop to maintain
18 their correct operation. These two functions are implemented in
19 configuration-specific files (usually "system_init.c" and "system_tasks.c")
20 in a configuration-specific folder under the "src/system_config" folder
21 within this project's top-level folder. An MPLAB Harmony project may have
22 more than one configuration, each contained within its own folder under
23 the "system_config" folder.
24 ****
25
26 // DOM-IGNORE-BEGIN
27 ****
28 Copyright (c) 2013-2014 released Microchip Technology Inc. All rights reserved.
29
30 //Microchip licenses to you the right to use, modify, copy and distribute
31 Software only when embedded on a Microchip microcontroller or digital signal
32 controller that is integrated into your product or third party product
33 (pursuant to the sublicense terms in the accompanying license agreement).
34
35 You should refer to the license agreement accompanying this Software for
36 additional information regarding your rights and obligations.
37
38 SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,
39 EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF
40 MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.
41 IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER
42 CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR
43 OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES
44 INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR
45 CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF
46 SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES
47 (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.
48 ****
49 // DOM-IGNORE-END
50
51
52 // ****
53 // ****
54 // Section: Included Files
55 // ****
56 // ****
57
58 #include <stddef.h> // Defines NULL
59 #include <stdbool.h> // Defines true
60 #include <stdlib.h> // Defines EXIT_FAILURE
61 #include "system/common/sys_module.h" // SYS function prototypes
62
63
64 // ****
65 // ****
66 // Section: Main Entry Point
67 // ****
68 // ****
69
70 int main ( void )
71 {
72     /* Initialize all MPLAB Harmony modules, including application(s). */
73     SYS_Initialize ( NULL );

```

```
74
75
76     while ( true )
77     {
78         /* Maintain state machines of all polled MPLAB Harmony modules. */
79         SYS_Tasks ( );
80     }
81
82     /* Execution should not come here during normal operation */
83
84     return ( EXIT_FAILURE );
85 }
86
87
88 *****
89 End of File
90 */
91
92
```

```

1  ****
2  *
3  *   _____| |_____| |_____| |_____| \ / | |_____| |_____| |_____| | . ' _____\ |
4  *  | |_____| |_____| |_____| |_____| | \ / | |_____| |_____| |_____| | | ( _____) |
5  *  | |_____| |_____| |_____| |_____| | \ / | |_____| |_____| |_____| | | \ _____) |
6  *  | |_____| |_____| |_____| |_____| | \ / | |_____| |_____| |_____| | | \ _____) |
7  *  | |_____| |_____| |_____| |_____| | \ / | |_____| |_____| |_____| | | \ _____) |
8  *
9  ****
10 *
11 * File          : app.c
12 * Version       : 1.0
13 *
14 ****
15 *
16 * Description  : Managing global state machine
17 *
18 ****
19 *
20 * Author        : Miguel Santos
21 * Date          : 25.09.2023
22 *
23 ****
24 *
25 * MPLAB X      : 5.45
26 * XC32          : 2.50
27 * Harmony        : 2.06
28 *
29 ****
30
31 #include "app.h"
32 #include "modules/TLC5973.h"
33 #include "driver/tmr/drv_tmr_mapping.h"
34
35 #define LED_WIFI TLC_DRV_ID_0
36
37 ****
38
39 /* Timeout for turning off output (in milliseconds) */
40 #define APP_TIME_OUT_MS 7200000
41
42 /* Wait time to allow user to react (in milliseconds) */
43 #define APP_TIME_WAIT_MS 60000
44
45 ****
46
47 /* Declaration of global application data */
48
49 APP_DATA appData;
50
51 ****
52
53 /**
54 * @brief APP_Initialize
55 *
56 * Initialize APP state machine
57 *
58 * @param void
59 * @return void
60 */
61 void APP_Initialize ( void )
62 {
63     /* Place the App state machine in its initial state. */
64     appData.state = APP_STATE_INIT;
65
66     /* Initialize TLC5973 interface */
67     TLC_Initialize();
68
69     /* Start application's timer */
70     DRV_TMR0_Start();
71
72     /* Initialize timeout counter */
73     CNT_Initialize(&appData.timeOut, 50);

```

```

74 }
75
76 /*****
77 */
78 /**
79 * @brief APP_Tasks
80 *
81 * Execute APP state machine, should be called cyclically
82 *
83 * @param void
84 * @return void
85 */
86 void APP_Tasks ( void )
87 {
88     /* Check the application's current state. */
89     switch ( appData.state )
90     {
91         /* Application's initial state. */
92         case APP_STATE_INIT:
93         {
94             AC_SETOn();
95             appData.state = APP_STATE_IDLE;
96             break;
97         }
98
99         case APP_STATE_IDLE:
100        {
101            if(CNT_Check(&appData.timeOut))
102            {
103                AC_SETOff();
104                appData.state = APP_STATE_SETUP_WIFI;
105            }
106            break;
107        }
108
109        case APP_STATE_SETUP_WIFI:
110        {
111            break;
112        }
113
114        case APP_STATE_SETUP_RFID:
115        {
116            break;
117        }
118
119        case APP_STATE_ASKING:
120        {
121            break;
122        }
123
124        case APP_STATE_OFF:
125        {
126            break;
127        }
128
129        case APP_STATE_ON:
130        {
131            break;
132        }
133
134     /* The default state should never be executed. */
135     default:
136     {
137         /* TODO: Handle error in application's state machine. */
138         break;
139     }
140 }
141
142 /*****
143 */
144 /* End of File *****/
145

```

```

1  ****
2  *
3  *   _____| |_____| |_____| |_____| \ / | |_____| |_____| |_____| | . ' _____\ |
4  *  | |_____| |_____| |_____| |_____| | \ / | |_____| |_____| |_____| | | |_____| | |
5  *  | |_____| |_____| |_____| |_____| | \ / | |_____| |_____| |_____| | | |_____| | |
6  *  | |_____| |_____| |_____| |_____| | \ / | |_____| |_____| |_____| | | |_____| | |
7  *  | |_____| |_____| |_____| |_____| | \ / | |_____| |_____| |_____| | | |_____| | |
8  *
9  ****
10 *
11 * File           : app.h
12 * Version        : 1.0
13 *
14 ****
15 *
16 * Description   : Managing global state machine
17 *
18 ****
19 *
20 * Author         : Miguel Santos
21 * Date          : 25.09.2023
22 *
23 ****
24 *
25 * MPLAB X       : 5.45
26 * XC32          : 2.50
27 * Harmony        : 2.06
28 *
29 ****
30
31 #ifndef _APP_H
32 #define _APP_H
33
34 ****
35
36 #include <stdint.h>
37 #include <stdbool.h>
38 #include <stddef.h>
39 #include <stdlib.h>
40 #include "system_config.h"
41 #include "system_definitions.h"
42 #include "modules/counter.h"
43
44 ****
45
46 /* Application states */
47 typedef enum
48 {
49     APP_STATE_INIT,
50     APP_STATE_IDLE,
51     APP_STATE_SETUP_WIFI,
52     APP_STATE_SETUP_RFID,
53     APP_STATE_ASKING,
54     APP_STATE_OFF,
55     APP_STATE_ON,
56
57 } APP_STATES;
58
59 ****
60
61 /* Structure to hold application data */
62 typedef struct
63 {
64     /* The application's current state */
65     APP_STATES state;
66
67     /* Timeout counter used to turn off output */
68     S_Counter timeOut;
69
70 } APP_DATA;
71
72 ****
73

```

```
74 /**
75 * @brief APP_Initialize
76 *
77 * Initialize APP state machine
78 *
79 * @param void
80 * @return void
81 */
82 void APP_Initialize ( void );
83
84 /*****
85 /**
86 * @brief APP_Tasks
87 *
88 * Execute APP state machine, should be called cyclically
89 *
90 * @param void
91 * @return void
92 */
93
94 void APP_Tasks ( void );
95
96 *****/
97
98 #endif /* _APP_H */
99
100 /* End of File *****/
101
```

```

1  ****
2  *
3  *      [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
4  *      [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
5  *      [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
6  *      [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
7  *      [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
8  *
9  ****
10 *
11 * File          : esp.c
12 * Version       : 1.0
13 *
14 ****
15 *
16 * Description   : Managing ESP32 state machine and commands
17 *
18 ****
19 *
20 * Author        : Miguel Santos
21 * Date          : 25.09.2023
22 *
23 ****
24 *
25 * MPLAB X       : 5.45
26 * XC32          : 2.50
27 * Harmony        : 2.06
28 *
29 ****
30
31 #include "esp.h"
32
33 ****
34
35 /* Enable or disable debug of specific parts by (un)comment */
36 #ifndef DEBUG_LED
37     #define DEBUG_LED PORTGbits.RG9
38 #endif
39 // #define DEBUG_ESP_UART
40 // #define DEBUG_ESP_GET
41 // #define DEBUG_ESP_SEND
42
43 ****
44
45 /* Define UART and interrupts used by ESP32 */
46 #define ESP_USART_ID           USART_ID_1
47 #define ESP_INT_SOURCE_USART_ERROR INT_SOURCE_USART_1_ERROR
48 #define ESP_INT_SOURCE_USART_RECEIVE INT_SOURCE_USART_1_RECEIVE
49 #define ESP_INT_SOURCE_USART_TRANSMIT INT_SOURCE_USART_1_TRANSMIT
50
51 ****
52
53 /* Time to wait for fifo to be filled */
54 #define ESP_COUNT_RECEIVE_MS 20
55
56 /* Time waiting for a response of main app */
57 #define ESP_COUNT_WAIT_MS 2000
58
59 ****
60
61 /* Declaration of global application data */
62 ESP_DATA espData;
63
64 ****
65
66 /**
67 * @brief ESP_Initialize
68 *
69 * Initialize ESP32 state machine, counters and FIFOs
70 *
71 * @param void
72 * @return void
73 */

```

```

74 void ESP_Initialize ( void )
75 {
76     /* Place the App state machine in its default state. */
77     espData.state = ESP_STATE_IDLE;
78
79     /* Initial flags values */
80     espData.transmit = false;
81     espData.receive = false;
82
83     /* Initialize timing counters */
84     CNT_Initialize(&espData.cntReceive, ESP_COUNT_RECEIVE_MS);
85     CNT_Initialize(&espData.cntWait, ESP_COUNT_WAIT_MS);
86
87     /* Initialize FIFO descriptors */
88     FIFO_Initialize(&espData fifoDesc_tx, ESP_FIFO_SIZE,
89                     espData.fifoBuff_tx, 0x00);
90     FIFO_Initialize(&espData fifoDesc_rx, ESP_FIFO_SIZE,
91                     espData.fifoBuff_rx, 0x00);
92 }
93
94 ****
95
96 /**
97 * @brief ESP_Tasks
98 *
99 * Execute ESP32 state machine, should be called cyclically
100 *
101 * @param void
102 * @return void
103 */
104 void ESP_Tasks ( void )
105 {
106     /* Check current state. */
107     switch ( espData.state )
108     {
109         /* Waiting for next state */
110         case ESP_STATE_IDLE:
111         {
112             /* Something to transmit in FIFO */
113             if(espData.transmit)
114             {
115                 espData.state = ESP_STATE_TRANSMIT;
116             }
117             /* Something to receive in FIFO */
118             else if(espData.receive)
119             {
120                 CNT_Reset(&espData.cntReceive);
121                 espData.state = ESP_STATE_RECEIVE;
122             }
123             break;
124         }
125
126         /* A new message has to be transmitted */
127         case ESP_STATE_TRANSMIT:
128         {
129             /* Check if data still available in FIFO */
130             if(FIFO_GetReadSpace(&espData.fifoDesc_tx))
131             {
132                 /* Enable the interrupt if needed */
133                 if(!SYS_INT_SourceEnabled(ESP_INT_SOURCE_USART_TRANSMIT))
134                 {
135                     SYS_INT_SourceEnable(ESP_INT_SOURCE_USART_TRANSMIT);
136                 }
137             }
138             else
139             {
140                 /* Leave state when fifo is empty */
141                 espData.transmit = false;
142                 espData.state = ESP_STATE_IDLE;
143             }
144             break;
145         }
146     }

```

```

147 /* Receiving a message by UART */
148 case ESP_STATE_RECEIVE:
149 {
150     /* Waiting for fifo to be filled */
151     if(CNT_Check(&espData.cntReceive))
152     {
153         if(FIFO_GetBuffer(&espData fifoDesc_rx, (uint8_t*)&espData.resBuffer))
154         {
155             /* Get first line of message */
156             espData.p_resBuffer = strtok(espData.resBuffer, "\r\n");
157
158             /* Flag state */
159             espData.translate = true;
160
161             /* Change directly to translate state */
162             espData.state = ESP_STATE_TRANSLATE;
163         }
164     else
165     {
166         /* An error occurred, going back to IDLE */
167         espData.state = ESP_STATE_IDLE;
168     }
169
170     /* Flag state */
171     espData.receive = false;
172 }
173 break;
174 }

175 /* Translating different parts of the message received */
176 case ESP_STATE_TRANSLATE:
177 {
178     /* A command is detected */
179     if(espData.p_resBuffer[0] == 'A' && espData.p_resBuffer[1] == 'T')
180     {
181         strcpy(espData.atResponse.command, espData.p_resBuffer);
182     }
183     /* Acknowledge detected */
184     if(strcmp(espData.p_resBuffer, AT_ACK_OK) ||
185        strcmp(espData.p_resBuffer, AT_ACK_ERROR))
186     {
187         strcpy(espData.atResponse.ack, espData.p_resBuffer);
188     }
189     /* Data is detected */
190     else
191     {
192         strcpy(espData.atResponse.data, espData.p_resBuffer);
193     }
194
195     /* Get next line in string */
196     espData.p_resBuffer = strtok(NULL, "\r\n");
197
198     /* Leave state when no more lines */
199     if(espData.p_resBuffer == NULL)
200     {
201         /* Reset buffer */
202         memset(espData.resBuffer, 0x00, sizeof(espData.resBuffer));
203
204         /* Machine states and flags */
205         espData.newMessage = true;
206         espData.translate = false;
207         espData.wait = true;
208         espData.state = ESP_STATE_WAIT;
209
210         CNT_Reset(&espData.cntWait);
211     }
212     break;
213 }

214 /* Waiting for main application to answer */
215 case ESP_STATE_WAIT:
216 {
217     if(CNT_Check(&espData.cntWait))
218

```

```

220
221     {
222         espData.newMessage = false;
223         espData.state = ESP_STATE_IDLE;
224     }
225     break;
226 }
227 /* The default state should never be executed. */
228 default:
229 {
230     /* TODO: Handle error in application's state machine. */
231     break;
232 }
233 }
234 }
235
236 /***** *****
237 /**
238 * @brief ESP_SendCommand
239 *
240 * Send a command to the ESP32, managed by state machine
241 *
242 * @param char Command to send ; Use constant definitions
243 * @return bool True = command send ; False = Not allowed to send a command
244 */
245
246 bool ESP_SendCommand( char *p_command )
247 {
248     /* Local variables */
249     S_Fifo *p_fifoDesc;
250     uint8_t commandSize;
251     uint8_t i_string;
252     bool commandStatus;
253
254     /* DEBUG */
255     #ifdef DEBUG_ESP_SEND
256         DEBUG_LED = true;
257     #endif
258
259     /* Default command status */
260     commandStatus = false;
261
262     /* Dont send a command if not IDLE */
263     if(espData.state == ESP_STATE_IDLE)
264     {
265         /* Point to the desired FIFO */
266         p_fifoDesc = &espData.fifoDesc_tx;
267
268         /* Get number of characters to send */
269         commandSize = strlen(p_command);
270
271         /* Check if enough space in FIFO */
272         if(FIFO_GetWriteSpace(p_fifoDesc) >= (commandSize + 2))
273         {
274             /* Loop to add command */
275             for(i_string = 0; i_string < commandSize; i_string++)
276             {
277                 FIFO_Add(p_fifoDesc, (uint8_t) (p_command[i_string]));
278             }
279
280             /* Add CR and LF suffix to FIFO */
281             FIFO_Add(p_fifoDesc, (uint8_t) ('\r'));
282             FIFO_Add(p_fifoDesc, (uint8_t) ('\n'));
283
284             /* Command added to FIFO */
285             espData.transmit = true;
286             commandStatus = true;
287         }
288     }
289     /* DEBUG */
290     #ifdef DEBUG_ESP_SEND
291         DEBUG_LED = false;
292     #endif

```

```

293
294     /* Feedback */
295     return commandStatus;
296 }
297
298 /***** *****
299 /**
300 * @brief _IntHandlerDrvUsartInstance0
301 *
302 * Interrupt instance to manage UART communication to ESP32
303 *
304 */
305
306 void __ISR(_UART_1_VECTOR, ipl7AUTO) _IntHandlerDrvUsartInstance0(void)
307 {
308     S_Fifo *RX_fifoDescriptor;
309     S_Fifo *TX_fifoDescriptor;
310     uint8_t TX_size;
311     uint8_t TX_BufferFull;
312     uint8_t dataFifo;
313     USART_ERROR usartStatus;
314
315     /* Pointers to fifo descriptors */
316     RX_fifoDescriptor = &espData.fifoDesc_rx;
317     TX_fifoDescriptor = &espData.fifoDesc_tx;
318
319
320     /* DEBUG */
321     #ifdef DEBUG_ESP_UART
322         DEBUG_LED = true;
323     #endif
324
325     /* Reading the error interrupt flag */
326     if(SYS_INT_SourceStatusGet(ESP_INT_SOURCE_USART_ERROR))
327     {
328         /* Clear up the error interrupt flag */
329         SYS_INT_SourceStatusClear(ESP_INT_SOURCE_USART_ERROR);
330     }
331
332     /* Reading the receive interrupt flag */
333     if(SYS_INT_SourceStatusGet(ESP_INT_SOURCE_USART_RECEIVE))
334     {
335         /* Checks overrun or parity error */
336         usartStatus = PLIB_USART_ErrorsGet(ESP_USART_ID);
337
338         if(usartStatus)
339         {
340             /* Errors are auto cleaned when read, except overrun */
341             if ( usartStatus & USART_ERROR_RECEIVER_OVERRUN )
342             {
343                 PLIB_USART_ReceiverOverrunErrorClear(ESP_USART_ID);
344             }
345         }
346         else
347         {
348             espData.receive = true;
349
350             while(PLIB_USART_ReceiverDataIsAvailable(ESP_USART_ID))
351             {
352                 dataFifo = PLIB_USART_ReceiverByteReceive(ESP_USART_ID);
353                 FIFO_Add(RX_fifoDescriptor, dataFifo);
354             }
355
356             /* Clear up the interrupt flag when buffer is empty */
357             SYS_INT_SourceStatusClear(ESP_INT_SOURCE_USART_RECEIVE);
358         }
359     }
360
361     /* Reading the transmit interrupt flag */
362     if(SYS_INT_SourceStatusGet(ESP_INT_SOURCE_USART_TRANSMIT))
363     {
364         TX_size = FIFO_GetReadSpace(TX_fifoDescriptor);
365         TX_BufferFull = PLIB_USART_TransmitterBufferIsFull(ESP_USART_ID);

```

```
366
367     while(TX_size && !TX_BufferFull)
368     {
369         FIFO_GetData(TX_fifoDescriptor, &dataFifo);
370         PLIB_USART_TransmitterByteSend(ESP_USART_ID, dataFifo);
371         TX_size = FIFO_GetReadSpace(TX_fifoDescriptor);
372         TX_BufferFull = PLIB_USART_TransmitterBufferIsFull(ESP_USART_ID);
373     }
374
375     /* Disable the interrupt, to avoid calling ISR continuously*/
376     SYS_INT_SourceDisable(ESP_INT_SOURCE_USART_TRANSMIT);
377
378     /* Clear up the interrupt flag */
379     SYS_INT_SourceStatusClear(ESP_INT_SOURCE_USART_TRANSMIT);
380 }
381
382 /* DEBUG */
383 #ifdef DEBUG_ESP_UART
384     DEBUG_LED = false;
385 #endif
386 }
387
388 /***** End of File *****/
389
390 /* End of File *****/
391
```

```

1  ****
2  *
3  *      [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
4  *      [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
5  *      [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
6  *      [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
7  *      [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
8  *
9  ****
10 *
11 * File           : esp.h
12 * Version        : 1.0
13 *
14 ****
15 *
16 * Description   : Managing ESP32 state machine and commands
17 *
18 ****
19 *
20 * Author         : Miguel Santos
21 * Date          : 25.09.2023
22 *
23 ****
24 *
25 * MPLAB X       : 5.45
26 * XC32          : 2.50
27 * Harmony        : 2.06
28 *
29 ****
30
31 #ifndef _ESP_H
32 #define _ESP_H
33
34 ****
35
36 #include <stdint.h>
37 #include <stdbool.h>
38 #include "system_config.h"
39 #include "system_definitions.h"
40 #include "modules/fifo.h"
41 #include "modules/counter.h"
42
43 ****
44
45 /* Buffers sizes */
46
47 #define ESP_FIFO_SIZE 50
48 #define AT_CMD_SIZE 50
49 #define AT_DATA_SIZE 50
50 #define AT_ACK_SIZE 10
51
52 ****
53
54 /* AT commands to send */
55 #define AT_CMD_AT      "AT"
56 #define AT_CMD_RST     "AT+RST"
57 #define AT_CMD_CWMODEIS "AT+CWMODE=1"
58 #define AT_CMD_CWMODE   "AT+CWMODE?"
59 #define AT_CMD_CWJAP    "AT+CWJAP=\"ES-SLO-2\", \"slo-etml-es\""
60
61 /* AT acknowledge responses */
62 #define AT_ACK_OK      "OK"
63 #define AT_ACK_ERROR   "ERROR"
64
65 ****
66
67 /* Structure of packets communication as defined by ESP32 datasheet */
68 typedef struct
69 {
70     char command[AT_CMD_SIZE];
71     char data[AT_DATA_SIZE];
72     char ack[AT_ACK_SIZE];
73 }
```

```

74 } S_AT_PACKET;
75
76 /*****
77
78 /* ESP state machine */
79 typedef enum
80 {
81     /* Waiting for a command */
82     ESP_STATE_IDLE,
83
84     /* Transmitting a command */
85     ESP_STATE_TRANSMIT,
86
87     /* Receiving a command */
88     ESP_STATE_RECEIVE,
89
90     /* Translating the UART message */
91     ESP_STATE_TRANSLATE,
92
93     /* Waiting for main application */
94     ESP_STATE_WAIT,
95
96 } E_ESP_STATES;
97
98 /*****
99
100 /* ESP32 structure of global data */
101 typedef struct
102 {
103     /* Application's current states */
104     E_ESP_STATES state;
105
106     /* Applications's flags */
107     bool transmit;
108     bool receive;
109     bool translate;
110     bool wait;
111     bool newMessage;
112
113     /* Applications COUNTERS */
114     S_Counter cntReceive;
115     S_Counter cntWait;
116
117     /* Application's FIFOs descriptors */
118     S_Fifo fifoDesc_tx;
119     S_Fifo fifoDesc_rx;
120
121     /* Application's FIFOs buffers */
122     uint8_t fifoBuff_tx[ESP_FIFO_SIZE];
123     uint8_t fifoBuff_rx[ESP_FIFO_SIZE];
124
125     /* Buffer to store response commands */
126     char resBuffer[ESP_FIFO_SIZE];
127
128     /* Pointer to last char received */
129     char *p_resBuffer;
130
131     /* Store response as different fields */
132     S_AT_PACKET atResponse;
133
134 } ESP_DATA;
135
136 /*****
137
138 /**
139 * @brief ESP_Initialize
140 *
141 * Initialize ESP32 state machine, counters and FIFOs
142 *
143 * @param void
144 * @return void
145 *
146 void ESP_Initialize ( void );

```

```
147
148 //*****
149
150 /**
151 * @brief ESP_Tasks
152 *
153 * Execute ESP32 state machine, should be called cyclically
154 *
155 * @param void
156 * @return void
157 */
158 void ESP_Tasks( void );
159
160 //*****
161
162 /**
163 * @brief ESP_SendCommand
164 *
165 * Send a command to the ESP32, managed by state machine
166 *
167 * @param char Command to send ; Use constant definitions
168 * @return bool True = command send ; False = Not allowed to send a command
169 */
170 bool ESP_SendCommand( char *command );
171
172 //*****
173
174 #endif /* _ESP_H */
175
176 /* End of File *****/
177
```

```

1  ****
2  *
3  *      [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
4  *      [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
5  *      [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
6  *      [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
7  *      [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
8  *
9  ****
10 *
11 * File          : chu.c
12 * Version       : 1.0
13 *
14 ****
15 *
16 * Description   : Managing Chilli UART b1 state machine and commands
17 *                   Manufacturer library is needed to interface it
18 *                   https://eccel.co.uk/wp-content/downloads/C-library-for-B1.zip
19 *
20 ****
21 *
22 * Author        : Miguel Santos
23 * Date          : 25.09.2023
24 *
25 ****
26 *
27 * MPLAB X       : 5.45
28 * XC32          : 2.50
29 * Harmony        : 2.06
30 *
31 ****
32
33 #include "chu.h"
34 #include "modules/ccittcrc.h"
35
36 ****
37
38 /* Enable or disable debug of specific parts by (un)comment */
39 #ifndef DEBUG_LED
40     #define DEBUG_LED PORTGbits.RG9
41 #endif
42 // #define DEBUG_CHU_UART
43
44 ****
45
46 /* Define UART and interrupts used by Chilli */
47 #define CHU_USART_ID           USART_ID_2
48 #define CHU_INT_SOURCE_USART_ERROR INT_SOURCE_USART_2_ERROR
49 #define CHU_INT_SOURCE_USART_RECEIVE INT_SOURCE_USART_2_RECEIVE
50 #define CHU_INT_SOURCE_USART_TRANSMIT INT_SOURCE_USART_2_TRANSMIT
51
52 ****
53
54 /* Declaration of global application data */
55 CHU_DATA chuData;
56
57 /* RFID B1 UART objects */
58 RFIDB1_InterfaceConfigurationT chuRfid_config;
59 RFIDB1_InterfaceT chuRfid_interface;
60 RFIDB1_ObjectT chuRfid_object;
61
62 ****
63
64 /**
65 * @brief CHU_Initialize
66 *
67 * Initialize Chilli state machine, counters and FIFOs
68 * Setup objects needed for RFIDB1 interface
69 *
70 * @param void
71 * @return void
72 */
73 void CHU_Initialize ( void )

```

```

74 {
75     /* Place the state machine in its initial state. */
76     chuData.state = CHU_STATE_IDLE;
77
78     /* Initial flags values */
79     chuData.transmit = false;
80     chuData.receive = false;
81
82     /* Initialize UART communication fifos */
83     FIFO_Initialize(&chuData fifoDesc_rx, CHU_FIFO_SIZE,
84                     chuData fifoBuff_rx, 0x00);
85     FIFO_Initialize(&chuData fifoDesc_tx, CHU_FIFO_SIZE,
86                     chuData fifoBuff_tx, 0x00);
87
88     /* Config setup for RFIDB1 */
89     chuRfid_config.InputBuffer = chuData fifoBuff_tx;
90     chuRfid_config.InputBufferSize = CHU_FIFO_SIZE;
91     chuRfid_config.OutputBuffer = chuData fifoBuff_rx;
92     chuRfid_config.OutputBufferSize = CHU_FIFO_SIZE;
93     chuRfid_config.handleResponse = CHU_RFID_Response;
94     chuRfid_config.handleRequest = CHU_RFID_Request;
95
96     /* Initialise RFIDB1 objects */
97     GetRFIDB1Interface(&chuRfid_interface);
98     chuRfid_interface.Initialise(&chuRfid_object, &chuRfid_config);
99     chuRfid_interface.SetPacketHeaderType(&chuRfid_object, HeaderTypeA);
100 }
101
102 /*****
103 ****
104 /**
105 * @brief CHU_Tasks
106 *
107 * Execute Chilli state machine, should be called cyclically
108 *
109 * @param void
110 * @return void
111 */
112 void CHU_Tasks ( void )
113 {
114     /* Check the application's current state. */
115     switch ( chuData.state )
116     {
117         /* Application's initial state. */
118         case CHU_STATE_IDLE:
119         {
120             chuRfid_interface.SendDummyCommand(&chuRfid_object);
121             chuData.state = CHU_STATE_TRANSLATE;
122             break;
123         }
124
125         case CHU_STATE_TRANSLATE:
126         {
127             break;
128         }
129
130         case CHU_STATE_RECEIVE:
131         {
132             break;
133         }
134
135         case CHU_STATE_WAIT:
136         {
137             break;
138         }
139
140         /* The default state should never be executed. */
141         default:
142             break;
143
144     }
145 }
```

```

147
148     {
149         /* TODO: Handle error in application's state machine. */
150         break;
151     }
152 }
153
154 /*****
155 /**
156 /**
157 * @brief CHU_RFID_Response
158 *
159 * Function used by interface library to get a command received by UART
160 * Should not be called by user !
161 *
162 * @param RFIDB1_ObjectT* rfid_object Pointer to RFIDB1 object used by Chilli
163 * @param uint8_t *data Output buffer of data to be receive by UART
164 * @param uint16_t size Size of the buffer
165 */
166 void CHU_RFID_Response( RFIDB1_ObjectT* rfid_object, uint8_t *data, uint16_t size )
167 {
168
169 }
170
171 /*****
172 /**
173 /**
174 * @brief CHU_RFID_Request
175 *
176 * Function used by interface library to send a command by UART
177 * Should not be called by user !
178 *
179 * @param RFIDB1_ObjectT* rfid_object Pointer to RFIDB1 object used by Chilli
180 * @param uint8_t *data Input buffer of data to be send by UART
181 * @param uint16_t size Size of the buffer
182 */
183 void CHU_RFID_Request( RFIDB1_ObjectT* rfid_object, uint8_t *data, uint16_t size )
184 {
185     /* Local variables declaration */
186     uint8_t i_data;
187     uint8_t freeSize;
188     uint8_t fifoData;
189
190     /* Get the space available in fifo */
191     freeSize = FIFO_GetWriteSpace(&chuData.fifoDesc_tx);
192
193     if(freeSize >= size)
194     {
195         /* Add buffer to fifo */
196         for(i_data = 0; i_data < size; i_data++)
197         {
198             fifoData = *(data + i_data);
199             FIFO_Add(&chuData.fifoDesc_tx, fifoData);
200         }
201
202         /* Enable the transmission interrupt */
203         SYS_INT_SourceEnable(CHU_INT_SOURCE_USART_TRANSMIT);
204     }
205 }
206
207 /*****
208 /**
209 /**
210 * @brief CHU_RFID_EnablePolling
211 *
212 * Send a raw command to enable polling
213 * Modifiy function as needed, based on datasheet
214 *
215 * @param void
216 * @return void
217 */
218 void CHU_RFID_Polling( void )
219 {

```

```

220  /* Local variables declaration */
221  uint8_t packet[26];
222  uint16_t crcHeader;
223  uint16_t crcData;
224
225  /* Header informations */
226  packet[0] = 0x02;
227  packet[1] = 0x15;
228  packet[2] = 0x00;
229
230  /* CRC header */
231  crcHeader = GetCCITT_CRC(&packet[0], 3);
232  packet[3] = crcHeader & 0x00FF;
233  packet[4] = (crcHeader & 0xFF00) >> 8;
234
235  /* Command : Polling */
236  packet[5] = 0x22;
237
238  /* Polling period */
239  packet[6] = (uint8_t)(CHU_POLLING_PERIOD_MS / 100);
240
241  /* Defined tags */
242  packet[7] = 0x00; // Number
243  packet[8] = 0x00; // ASYNC mode
244  packet[9] = 0x00; // IO Config
245  packet[10] = 0x00; // PWM
246  packet[11] = 0x00; // PWM duty
247  packet[12] = 0x00; // PWM period (msb)
248  packet[13] = 0x00; // PWM period
249  packet[14] = 0x00; // PWM period (lsb)
250  packet[15] = 0x00; // Timeout [100ms]
251
252  /* Undefined tags */
253  packet[16] = 0x03; // ASYNC mode
254  packet[17] = 0x22; // IO Config
255  packet[18] = 0x00; // PWM
256  packet[19] = 0x32; // PWM Duty
257  packet[20] = 0x00; // PWM period (msb)
258  packet[21] = 0x3E; // PWM period
259  packet[22] = 0xE8; // PWM period (lsb)
260  packet[23] = 0x32; // Timeout [100ms]
261
262  /* CRC data */
263  crcData = GetCCITT_CRC(&packet[5], 19);
264  packet[24] = crcData & 0x00FF;
265  packet[25] = (crcData & 0xFF00) >> 8;
266
267  /* Send command through interface */
268  chuRfid_interface.SendRawDataCommand(&chuRfid_object, packet, 26);
269 }
270
271 ****
272 /**
273 * @brief _IntHandlerDrvUsartInstance1
274 *
275 * Interrupt instance to manage UART communication to RFIDB1 Chilli
276 *
277 */
278
279 void __ISR(_UART_2_VECTOR, ipl7AUTO) _IntHandlerDrvUsartInstance1(void)
280 {
281  /* Local variables declaration */
282  S_Fifo *RX_fifoDescriptor;
283  S_Fifo *TX_fifoDescriptor;
284  uint8_t TX_size;
285  uint8_t TX_BufferFull;
286  uint8_t dataFifo;
287  USART_ERROR usartStatus;
288
289  /* Pointers to fifo descriptors */
290  RX_fifoDescriptor = &chuData.fifoDesc_rx;
291  TX_fifoDescriptor = &chuData.fifoDesc_tx;
292

```

```

293 /* DEBUG */
294 #ifdef DEBUG_CHU_UART
295     DEBUG_LED = true;
296 #endif
297
298 /* Reading the error interrupt flag */
299 if(SYS_INT_SourceStatusGet(CHU_INT_SOURCE_USART_ERROR))
300 {
301     /* Clear up the error interrupt flag */
302     SYS_INT_SourceStatusClear(CHU_INT_SOURCE_USART_ERROR);
303 }
304
305 /* Reading the receive interrupt flag */
306 if(SYS_INT_SourceStatusGet(CHU_INT_SOURCE_USART_RECEIVE))
307 {
308     /* Checks overrun or parity error */
309     usartStatus = PLIB_USART_ErrorsGet(CHU_USART_ID);
310
311     if(usartStatus)
312     {
313         /* Errors are auto cleaned when read, except overrun */
314         if ( usartStatus & USART_ERROR_RECEIVER_OVERRUN )
315         {
316             PLIB_USART_ReceiverOverrunErrorClear(CHU_USART_ID);
317         }
318     }
319     else
320     {
321         chuData.receive = true;
322
323         while(PLIB_USART_ReceiverDataIsAvailable(CHU_USART_ID))
324         {
325             dataFifo = PLIB_USART_ReceiverByteReceive(CHU_USART_ID);
326             FIFO_Add(RX_fifoDescriptor, dataFifo);
327         }
328
329         /* Clear up the interrupt flag when buffer is empty */
330         SYS_INT_SourceStatusClear(CHU_INT_SOURCE_USART_RECEIVE);
331     }
332 }
333
334 /* Reading the transmit interrupt flag */
335 if(SYS_INT_SourceStatusGet(CHU_INT_SOURCE_USART_TRANSMIT))
336 {
337     TX_size = FIFO_GetReadSpace(TX_fifoDescriptor);
338     TX_BufferFull = PLIB_USART_TransmitterBufferIsFull(CHU_USART_ID);
339
340     while(TX_size && !TX_BufferFull)
341     {
342         FIFO_GetData(TX_fifoDescriptor, &dataFifo);
343         PLIB_USART_TransmitterByteSend(CHU_USART_ID, dataFifo);
344         TX_size = FIFO_GetReadSpace(TX_fifoDescriptor);
345         TX_BufferFull = PLIB_USART_TransmitterBufferIsFull(CHU_USART_ID);
346     }
347
348     /* Disable the interrupt, to avoid calling ISR continuously*/
349     SYS_INT_SourceDisable(CHU_INT_SOURCE_USART_TRANSMIT);
350
351     /* Clear up the interrupt flag */
352     SYS_INT_SourceStatusClear(CHU_INT_SOURCE_USART_TRANSMIT);
353 }
354
355 /* DEBUG */
356 #ifdef DEBUG_CHU_UART
357     DEBUG_LED = false;
358 #endif
359 }
360
361 ****
362 /* End of File ****
363
364

```

```

1  ****
2  *
3  *      [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
4  *      [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
5  *      [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
6  *      [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
7  *      [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
8  *
9  ****
10 *
11 * File           : chu.h
12 * Version        : 1.0
13 *
14 ****
15 *
16 * Description   : Managing Chilli UART b1 state machine and commands
17 *                   Manufacturer library is needed to interface it
18 *                   https://eccel.co.uk/wp-content/downloads/C-library-for-B1.zip
19 *
20 ****
21 *
22 * Author         : Miguel Santos
23 * Date          : 25.09.2023
24 *
25 ****
26 *
27 * MPLAB X       : 5.45
28 * XC32          : 2.50
29 * Harmony        : 2.06
30 *
31 ****
32
33 #ifndef _CHU_H
34 #define _CHU_H
35
36 ****
37
38 #include <stdint.h>
39 #include <stdbool.h>
40 #include <stddef.h>
41 #include <stdlib.h>
42 #include "system_config.h"
43 #include "system_definitions.h"
44 #include "modules/fifo.h"
45 #include "modules/RFIDB1ClientInterface.h"
46
47 ****
48
49 /* Fifo buffer size */
50 #define CHU_FIFO_SIZE 16
51
52 /* Polling period must be a 100ms multiple */
53 #define CHU_POLLING_PERIOD_MS 500
54
55 ****
56
57 /* Chilli UART State machine*/
58 typedef enum
59 {
60     /* Waiting for a command */
61     CHU_STATE_IDLE,
62
63     /* Transmitting a command */
64     CHU_STATE_TRANSMIT,
65
66     /* Receiving a command */
67     CHU_STATE_RECEIVE,
68
69     /* Translating the UART message */
70     CHU_STATE_TRANSLATE,
71
72     /* Waiting for main application */
73     CHU_STATE_WAIT,
74
75     /* Application specific states */
76     CHU_STATE_APP1,
77     CHU_STATE_APP2,
78     CHU_STATE_APP3,
79
80     /* Total number of states */
81     CHU_STATE_TOTAL
82 }
83
84 ****
85
86 ****
87
88 ****
89
90 ****
91
92 ****
93
94 ****
95
96 ****
97
98 ****
99
100 ****
101
102 ****
103
104 ****
105
106 ****
107
108 ****
109
110 ****
111
112 ****
113
114 ****
115
116 ****
117
118 ****
119
120 ****
121
122 ****
123
124 ****
125
126 ****
127
128 ****
129
130 ****
131
132 ****
133
134 ****
135
136 ****
137
138 ****
139
140 ****
141
142 ****
143
144 ****
145
146 ****
147
148 ****
149
150 ****
151
152 ****
153
154 ****
155
156 ****
157
158 ****
159
160 ****
161
162 ****
163
164 ****
165
166 ****
167
168 ****
169
170 ****
171
172 ****
173
174 ****
175
176 ****
177
178 ****
179
180 ****
181
182 ****
183
184 ****
185
186 ****
187
188 ****
189
190 ****
191
192 ****
193
194 ****
195
196 ****
197
198 ****
199
200 ****
201
202 ****
203
204 ****
205
206 ****
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
299 ****
300
300 ****
301
301 ****
302
302 ****
303
303 ****
304
304 ****
305
305 ****
306
306 ****
307
307 ****
308
308 ****
309
309 ****
310
310 ****
311
311 ****
312
312 ****
313
313 ****
314
314 ****
315
315 ****
316
316 ****
317
317 ****
318
318 ****
319
319 ****
320
320 ****
321
321 ****
322
322 ****
323
323 ****
324
324 ****
325
325 ****
326
326 ****
327
327 ****
328
328 ****
329
329 ****
330
330 ****
331
331 ****
332
332 ****
333
333 ****
334
334 ****
335
335 ****
336
336 ****
337
337 ****
338
338 ****
339
339 ****
340
340 ****
341
341 ****
342
342 ****
343
343 ****
344
344 ****
345
345 ****
346
346 ****
347
347 ****
348
348 ****
349
349 ****
350
350 ****
351
351 ****
352
352 ****
353
353 ****
354
354 ****
355
355 ****
356
356 ****
357
357 ****
358
358 ****
359
359 ****
360
360 ****
361
361 ****
362
362 ****
363
363 ****
364
364 ****
365
365 ****
366
366 ****
367
367 ****
368
368 ****
369
369 ****
370
370 ****
371
371 ****
372
372 ****
373
373 ****
374
374 ****
375
375 ****
376
376 ****
377
377 ****
378
378 ****
379
379 ****
380
380 ****
381
381 ****
382
382 ****
383
383 ****
384
384 ****
385
385 ****
386
386 ****
387
387 ****
388
388 ****
389
389 ****
390
390 ****
391
391 ****
392
392 ****
393
393 ****
394
394 ****
395
395 ****
396
396 ****
397
397 ****
398
398 ****
399
399 ****
400
400 ****
401
401 ****
402
402 ****
403
403 ****
404
404 ****
405
405 ****
406
406 ****
407
407 ****
408
408 ****
409
409 ****
410
410 ****
411
411 ****
412
412 ****
413
413 ****
414
414 ****
415
415 ****
416
416 ****
417
417 ****
418
418 ****
419
419 ****
420
420 ****
421
421 ****
422
422 ****
423
423 ****
424
424 ****
425
425 ****
426
426 ****
427
427 ****
428
428 ****
429
429 ****
430
430 ****
431
431 ****
432
432 ****
433
433 ****
434
434 ****
435
435 ****
436
436 ****
437
437 ****
438
438 ****
439
439 ****
440
440 ****
441
441 ****
442
442 ****
443
443 ****
444
444 ****
445
445 ****
446
446 ****
447
447 ****
448
448 ****
449
449 ****
450
450 ****
451
451 ****
452
452 ****
453
453 ****
454
454 ****
455
455 ****
456
456 ****
457
457 ****
458
458 ****
459
459 ****
460
460 ****
461
461 ****
462
462 ****
463
463 ****
464
464 ****
465
465 ****
466
466 ****
467
467 ****
468
468 ****
469
469 ****
470
470 ****
471
471 ****
472
472 ****
473
473 ****
474
474 ****
475
475 ****
476
476 ****
477
477 ****
478
478 ****
479
479 ****
480
480 ****
481
481 ****
482
482 ****
483
483 ****
484
484 ****
485
485 ****
486
486 ****
487
487 ****
488
488 ****
489
489 ****
490
490 ****
491
491 ****
492
492 ****
493
493 ****
494
494 ****
495
495 ****
496
496 ****
497
497 ****
498
498 ****
499
499 ****
500
500 ****
501
501 ****
502
502 ****
503
503 ****
504
504 ****
505
505 ****
506
506 ****
507
507 ****
508
508 ****
509
509 ****
510
510 ****
511
511 ****
512
512 ****
513
513 ****
514
514 ****
515
515 ****
516
516 ****
517
517 ****
518
518 ****
519
519 ****
520
520 ****
521
521 ****
522
522 ****
523
523 ****
524
524 ****
525
525 ****
526
526 ****
527
527 ****
528
528 ****
529
529 ****
530
530 ****
531
531 ****
532
532 ****
533
533 ****
534
534 ****
535
535 ****
536
536 ****
537
537 ****
538
538 ****
539
539 ****
540
540 ****
541
541 ****
542
542 ****
543
543 ****
544
544 ****
545
545 ****
546
546 ****
547
547 ****
548
548 ****
549
549 ****
550
550 ****
551
551 ****
552
552 ****
553
553 ****
554
554 ****
555
555 ****
556
556 ****
557
557 ****
558
558 ****
559
559 ****
560
560 ****
561
561 ****
562
562 ****
563
563 ****
564
564 ****
565
565 ****
566
566 ****
567
567 ****
568
568 ****
569
569 ****
570
570 ****
571
571 ****
572
572 ****
573
573 ****
574
574 ****
575
575 ****
576
576 ****
577
577 ****
578
578 ****
579
579 ****
580
580 ****
581
581 ****
582
582 ****
583
583 ****
584
584 ****
585
585 ****
586
586 ****
587
587 ****
588
588 ****
589
589 ****
590
590 ****
591
591 ****
592
592 ****
593
593 ****
594
594 ****
595
595 ****
596
596 ****
597
597 ****
598
598 ****
599
599 ****
600
600 ****
601
601 ****
602
602 ****
603
603 ****
604
604 ****
605
605 ****
606
606 ****
607
607 ****
608
608 ****
609
609 ****
610
610 ****
611
611 ****
612
612 ****
613
613 ****
614
614 ****
615
615 ****
616
616 ****
617
617 ****
618
618 ****
619
619 ****
620
620 ****
621
621 ****
622
622 ****
623
623 ****
624
624 ****
625
625 ****
626
626 ****
627
627 ****
628
628 ****
629
629 ****
630
630 ****
631
631 ****
632
632 ****
633
633 ****
634
634 ****
635
635 ****
636
636 ****
637
637 ****
638
638 ****
639
639 ****
640
640 ****
641
641 ****
642
642 ****
643
643 ****
644
644 ****
645
645 ****
646
646 ****
647
647 ****
648
648 ****
649
649 ****
650
650 ****
651
651 ****
652
652 ****
653
653 ****
654
654 ****
655
655 ****
656
656 ****
657
657 ****
658
658 ****
659
659 ****
660
660 ****
661
661 ****
662
662 ****
663
663 ****
664
664 ****
665
665 ****
666
666 ****
667
667 ****
668
668 ****
669
669 ****
670
670 ****
671
671 ****
672
672 ****
673
673 ****
674
674 ****
675
675 ****
676
676 ****
677
677 ****
678
678 ****
679
679 ****
680
680 ****
681
681 ****
682
682 ****
683
683 ****
684
684 ****
685
685 ****
686
686 ****
687
687 ****
688
688 ****
689
689 ****
690
690 ****
691
691 ****
692
692 ****
693
693 ****
694
694 ****
695
695 ****
696
696 ****
697
697 ****
698
698 ****
699
699 ****
700
700 ****
701
701 ****
702
702 ****
703
703 ****
704
704 ****
705
705 ****
706
706 ****
707
707 ****
708
708 ****
709
709 ****
710
710 ****
711
711 ****
712
712 ****
713
713 ****
714
714 ****
715
715 ****
716
716 ****
717
717 ****
718
718 ****
719
719 ****
720
720 ****
721
721 ****
722
722 ****
723
723 ****
724
724 ****
725
725 ****
726
726 ****
727
727 ****
728
728 ****
729
729 ****
730
730 ****
731
731 ****
732
732 ****
733
733 ****
734
734 ****
735
735 ****
736
736 ****
737
737 ****
738
738 ****
739
739 ****
740
740 ****
741
741 ****
742
742 ****
743
743 ****
744
744 ****
745
745 ****
746
746 ****
747
747 ****
748
748 ****
749
749 ****
750
750 ****
751
751 ****
752
752 ****
753
753 ****
754
754 ****
755
755 ****
756
756 ****
757
757 ****
758
758 ****
759
759 ****
760
760 ****
761
761 ****
762
762 ****
763
763 ****
764
764 ****
765
765 ****
766
766 ****
767
767 ****
768
768 ****
769
769 ****
770
770 ****
771
771 ****
772
772 ****
773
773 ****
774
774 ****
775
775 ****
776
776 ****
777
777 ****
778
778 ****
779
779 ****
780
780 ****
781
781 ****
782
782 ****
783
783 ****
784
784 ****
785
785 ****
786
786 ****
787
787 ****
788
788 ****
789
789 ****
790
790 ****
791
791 ****
792
792 ****
793
793 ****
794
794 ****
795
795 ****
796
796 ****
797
797 ****
798
798 ****
799
799 ****
800
800 ****
801
801 ****
802
802 ****
803
803 ****
804
804 ****
805
805 ****
806
806 ****
807
807 ****
808
808 ****
809
809 ****
810
810 ****
811
811 ****
812
812 ****
813
813 ****
814
814 ****
815
815 ****
816
816 ****
817
817 ****
818
818 ****
819
819 ****
820
820 ****
821
821 ****
822
822 ****
823
823 ****
824
824 ****
825
825 ****
826
826 ****
827
827 ****
828
828 ****
829
829 ****
830
830 ****
831
831 ****
832
832 ****
833
833 ****
834
834 ****
835
835 ****
836
836 ****
837
837 ****
838
838 ****
839
839 ****
840
840 ****
841
841 ****
842
842 ****
843
843 ****
844
844 ****
845
845 ****
846
846 ****
847
847 ****
848
848 ****
849
849 ****
850
850 ****
851
851 ****
852
852 ****
853
853 ****
854
854 ****
855
855 ****
856
856 ****
857
857 ****
858
858 ****
859
859 ****
860
860 ****
861
861 ****
862
862 ****
863
863 ****
864
864 ****
865
865 ****
866
866 ****
867
867 ****
868
868 ****
869
869 ****
870
870 ****
871
871 ****
872
872 ****
873
873 ****
874
874 ****
875
875 ****
876
876 ****
877
877 ****
878
878 ****
879
879 ****
880
880 ****
881
881 ****
882
882 ****
883
883 ****
884
884 ****
885
885 ****
886
886 ****
887
887 ****
888
888 ****
889
889 ****
890
890 ****
891
891 ****
892
892 ****
893
893 ****
894
894 ****
895
895 ****
896
896 ****
897
897 ****
898
898 ****
899
899 ****
900
900 ****
901
901 ****
902
902 ****
903
903 ****
904
904 ****
905
905 ****
906
906 ****
907
907 ****
908
908 ****
909
909 ****
910
910 ****
911
911 ****
912
912 ****
913
913 ****
914
914 ****
915
915 ****
916
916 ****
917
917 ****
918
918 ****
919
919 ****
920
920 ****
921
921 ****
922
922 ****
923
923 ****
924
924 ****
925
925 ****
926
926 ****
927
927 ****
928
928 ****
929
929 ****
930
930 ****
931
931 ****
932
932 ****
933
933 ****
934
934 ****
935
935 ****
936
936 ****
937
937 ****
938
938 ****
939
939 ****
940
940 ****
941
941 ****
942
942 ****
943
943 ****
944
944 ****
945
945 ****
946
946 ****
947
947 ****
948
948 ****
949
949 ****
950
950 ****
951
951 ****
952
952 ****
953
953 ****
954
954 ****
955
955 ****
956
956 ****
957
957 ****
958
958 ****
959
959 ****
960
960 ****
961
961 ****
962
962 ****
963
963 ****
964
964 ****
965
965 ****
966
966 ****
967
967 ****
968
968 ****
969
969 ****
970
970 ****
971
971 ****
972
972 ****
973
973 ****
974
974 ****
975
975 ****
976
976 ****
977
977 ****
978
978 ****
979
979 ****
980
980 ****
981
981 ****
982
982 ****
983
983 ****
984
984 ****
985
985 ****
986
986 ****
987
987 ****
988
988 ****
989
989 ****
990
990 ****
991
991 ****
992
992 ****
993
993 ****
994
994 ****
995
995 ****
996
996 ****
997
997 ****
998
998 ****
999
999 ****
1000
1000 ****
1001
1001 ****
1002
1002 ****
1003
1003 ****
1004
1004 ****
1005
1005 ****
1006
1006 ****
1007
1007 ****
1008
1008 ****
1009
1009 ****
1010
1010 ****
1011
1011 ****
1012
1012 ****
1013
1013 ****
1014
1014 ****
1015
1015 ****
1016
1016 ****
1017
1017 ****
1018
1018 ****
1019
1019 ****
1020
1020 ****
1021
1021 ****
1022
1022 ****
1023
1023 ****
1024
1024 ****
1025
1025 ****
1026
1026 ****
1027
1027 ****
1028
1028 ****
1029
1029 ****
1030
1030 ****
1031
1031 ****
1032
1032 ****
1033
1033 ****
1034
1034 ****
1035
1035 ****
1036
1036 ****
1037
1037 ****
1038
1038 ****
1039
1039 ****
1040
1040 ****
1041
1041 ****
1042
1042 ****
1043
1043 ****
1044
1044 ****
1045
1045 ****
1046
1046 ****
1047
1047 ****
1048
1048 ****
1049
1049 ****
1050
1050 ****
1051
1051 ****
1052
1052 ****
1053
1053 ****
1054
1054 ****
1055
1055 ****
1056
1056 ****
1057
1057 ****
1058
1058 ****
1059
1059 ****
1060
1060 ****
1061
1061 ****
1062
1062 ****
1063
1063 ****
1064
1064 ****
1065
1065 ****
1066
1066 ****
1067
1067 ****
1068
1068 ****
1069
1069 ****
1070
1070 ****
1071
1071 ****
1072
1072 ****
1073
1073 ****
1074
1074 ****
1075
1075 ****
1076
1076 ****
1077
1077 ****
1078
1078 ****
1079
1079 ****
1080
1080 ****
1081
1081 ****
1082
1082 ****
1083
1083 ****
1084
1084 ****
1085
1085 ****
1086
1086 ****
1087
1087 ****
1088
1088 ****
1089
1089 ****
1090
1090 ****
1091
1091 ****
1092
1092 ****
1093
1093 ****
1094
1094 ****
1095
1095 ****
1096
1096 ****
1097
1097 ****
1098
1098 ****
1099
1099 ****
1100
1100 ****
1101
1101 ****
1102
1102 ****
1103
1103 ****
1104
1104 ****
1105
1105 ****
1106
1106 ****
1107
1107 ****
1108
1108 ****
1109
1109 ****
1110
1110 ****
1111
1111 ****
1112
1112 ****
1113
1113 ****
1114
1114 ****
1115
1115 ****
1116
1116 ****
1117
1117 ****
1118
1118 ****
1119
1119 ****
1120
1120 ****
1121
1121 ****
1122
1122 ****
1123
1123 ****
1124
1124 ****
1125
1125 ****
1126
1126 ****
1127
1127 ****
1128
1128 ****
1129
1129 ****
1130
1130 ****
1131
1131 ****
1132
1132 ****
1133
1133 ****
1134
1134 ****
1135
1135 ****
1136
1136 ****
1137
1137 ****
1138
1138 ****
1139
1139 ****
1140
1140 ****
1141
1141 ****
1142
1142 ****
1143
1143 ****
1144
1144 ****
1145
1145 ****
1146
1146 ****
1147
1147 ****
1148
1148 ****
1149
1149 ****
1150
1150 ****
1151
1151 ****
1152
1152 ****
1153
1153 ****
1154
1154 ****
1155
1155 ****
1156
1156 ****
1157
1157 ****
1158
1158 ****
1159
1159 ****
1160
1160 ****
1161
1161 ****
1162
1162 ****
1163
1163 ****
1164
1164 ****
1165
1165 ****
1166
1166 ****
1167
1167 ****
1168
1168 ****
1169
1169 ****
1170
1170 ****
1171
1171 ****
1172
1172 ****
1173
1173 ****
1174
1174 ****
1175
1175 ****
1176
1176 ****
1177
1177 ****
1178
1178 ****
1179
1179 ****
1180
1180 ****
1181
1181 ****
1182
1182 ****
1183
1183 ****
1184
1184 ****
1185
1185 ****
1186
1186 ****
1187
1187 ****
1188
1188 ****
1189
1189 ****
1190
1190 ****
1191
1191 ****
1192
1192 ****
1193
1193 ****
1194
1194 ****
1195
1195 ****
1196
1196 ****
1197
1197 ****
1198
1198 ****
1199
1199 ****
1200
1200 ****
1201
1201 ****
1202
1202 ****
1203
1203 ****
1204
1204 ****
1205
1205 ****
1206
1206 ****
1207
1207 ****
1208
1208 ****
1209
1209 ****
1210
1210 ****
1211
1211 ****
1212
1212 ****
1213
1213 ****
1214
1214 ****
1215
1215 ****
1216
1216 ****
1217
1217 ****
1218
1218 ****
1219
1219 ****
1220
1220 ****
1221
1221 ****
1222
1222 ****
1223
1223 ****
1224
1224 ****
1225
1225 ****
1226
1226 ****
1227
1227 ****
1228
1228 ****
1229
1229 ****
1230
1230 ****
1231
1231 ****
1232
1232 ****
1233
1233 ****
1234
1234 ****
1235
1235 ****
1236
1236 ****
1237
1237 ****
1238
1238 ****
1239
1239 ****
1240
1240 ****
1241
1241 ****
1242
1242 ****
1243
1243 ****
1244
1244 ****
1245
1245 ****
1246
1246 ****
1247
1247 ****
1248
1248 ****
1249
1249 ****
1250
1250 ****
1251
1251 ****
1252
1252 ****
1253
1253 ****
1254
1254 ****
1255
1255 ****
1256
1256 ****
1257
1257 ****
1258
1258 ****
1259
1259 ****
1260
1260 ****
1261
1261 ****
1262
1262 ****
1263
1263 ****
1264
1264 ****
1265
1265 ****
1266
1266 ****
1267
1267 ****
1268
1268 ****
1269
1269 ****
1270
1270 ****
1271
1271 ****
1272
1272 ****
1273
1273 ****
1274
1274 ****
1275
1275 ****
1276
1276 ****
1277
1277 ****
1278
1278 ****
1279
1279 ****
1280
1280 ****
1281
1281 ****
1282
1282 ****
1283
1283 ****
1284
1284 ****
1285
1285 ****
1286
1286 ****
1287
1287 ****
1288
1288 ****
1289
1289 ****
1290
1290 ****
1291
1291 ****
1292
1292 ****
1293
1293 ****
1294
1294 ****
1295
1295 ****
1296
1296 ****
1297
1297 ****
1298
1298 ****
1299
1299 ****
1300
1300 ****
1301
1301 ****
1302
1302 ****
1303
1303 ****
1304
1304 ****
1305
1305 ****
1306
1306 ****
1307
1307 ****
1308
1308 ****
1309
1309 ****
1310
1310 ****
1311
1311 ****
1312
1312 ****
1313
1313 ****
1314
1314 ****
1315
1315 ****
1316
1316 ****
1317
1317 ****
1318
1318 ****
1319
1319 ****
1320
1320 ****
1321
1321 ****
1322
1322 ****
1323
1323 ****
1324
1324 ****
1325
1325 ****
1326
1326 ****
1327
1327 ****
1328
1328 ****
1329
1329 ****
1330
1330 ****
1331
1331 ****
1332
1332 ****
1333
1333 ****
1334
1334 ****
1335
1335 ****
1336
1336 ****
1337
1337 ****
1338
1338 ****
1339
1339 ****
1340
1340 ****
1341
1341 ****
1342
1342 ****
1343
1343 ****
1344
1344 ****
1345
1345 ****
1346
1346 ****
1347
1347 ****
1348
1348 ****
1349
1349 ****
1350
1350 ****
1351
1351 ****
1352
1352 ****
1353
1353 ****
1354
1354 ****
1355
1355 ****
1356
1356 ****
1357
1357 ****
1358
1358 ****
1359
1359 ****
1360
1360 ****
1361
1361 ****
1362
1362 ****
1363
1363 ****
1364
1364 ****
1365
1365 ****
13
```

```

74
75 } CHU_STATES;
76
77 /***** *****/
78
79 /* Chilli UART structure of global data */
80 typedef struct
81 {
82     /* The application's current state */
83     CHU_STATES state;
84
85     /* Applications's flags */
86     bool transmit;
87     bool receive;
88
89     /* Application's FIFOs descriptors */
90     S_Fifo fifoDesc_tx;
91     S_Fifo fifoDesc_rx;
92
93     /* Application's FIFOs buffers */
94     uint8_t fifoBuff_tx[CHU_FIFO_SIZE];
95     uint8_t fifoBuff_rx[CHU_FIFO_SIZE];
96
97 } CHU_DATA;
98
99 /*****
100 /**
101 * @brief CHU_Initialize
102 *
103 * Initialize Chilli state machine, counters and FIFOs
104 * Setup objects needed for RFIDB1 interface
105 *
106 * @param void
107 * @return void
108 */
109 void CHU_Initialize ( void );
110
111 /*****
112 /**
113 /**
114 * @brief CHU_Tasks
115 *
116 * Execute Chilli state machine, should be called cyclically
117 *
118 * @param void
119 * @return void
120 */
121 void CHU_Tasks( void );
122
123 /*****
124 /**
125 /**
126 * @brief CHU_RFID_Response
127 *
128 * Function used by interface library to get a command received by UART
129 * Should not be called by user !
130 *
131 * @param RFIDB1_ObjectT* rfid_object Pointer to RFIDB1 object used by Chilli
132 * @param uint8_t *data Output buffer of data to be receive by UART
133 * @param uint16_t size Size of the buffer
134 */
135 void CHU_RFID_Response( RFIDB1_ObjectT* rfid_object, uint8_t *data, uint16_t size );
136
137 /*****
138 /**
139 /**
140 * @brief CHU_RFID_Request
141 *
142 * Function used by interface library to send a command by UART
143 * Should not be called by user !
144 *
145 * @param RFIDB1_ObjectT* rfid_object Pointer to RFIDB1 object used by Chilli
146

```

```
147 * @param uint8_t *data Input buffer of data to be send by UART
148 * @param uint16_t size Size of the buffer
149 */
150 void CHU_RFID_Request( RFIDB1_ObjectT* rfid_object, uint8_t *data, uint16_t size );
151
152 /*****
153 /**
154 * @brief CHU_RFID_EnablePolling
155 *
156 * Send a raw command to enable polling
157 * Modifiy function as needed, based on datasheet
158 *
159 * @param void
160 * @return void
161 */
162 void CHU_RFID_Polling( void );
163
164 /*****
165 #endif /* _CHU_H */
166
167 /* End of File *****/
168
169
170
```

```

1  ****
2  *
3  *      [-----] | [-----] | [-----] \ / [-----] | [-----] | [-----] | . [-----]
4  *      | [-----] | [-----] | [-----] | [-----] | [-----] | [-----] | [-----] | [-----] | .
5  *      | [-----] | [-----] | [-----] | [-----] | [-----] | [-----] | [-----] | [-----] | .
6  *      | [-----] | [-----] | [-----] | [-----] | [-----] | [-----] | [-----] | [-----] | .
7  *      | [-----] | [-----] | [-----] | [-----] | [-----] | [-----] | [-----] | [-----] | .
8  *
9  ****
10 *
11 * File          : buzzer.c
12 * Version       : 1.0
13 *
14 ****
15 *
16 * Description   : Managing buzzer state machine and sequences
17 *
18 *           Thanks to robsoncouto on GitHub for musics !
19 *           https://github.com/robsoncouto/arduino-songs.git
20 *
21 ****
22 *
23 * Author        : Miguel Santos
24 * Date          : 25.09.2023
25 *
26 ****
27 *
28 * MPLAB X       : 5.45
29 * XC32          : 2.50
30 * Harmony        : 2.06
31 *
32 ****
33
34 #include "bzr.h"
35 #include "peripheral/oc/plib_oc.h"
36
37 ****
38
39 /* Define the timer used by the buzzer */
40 #define BZR_TMR_ID TMR_ID_3
41
42 /* Define the OC output used by the buzzer */
43 #define BZR_OC_ID OC_ID_5
44
45 /* Set the volume of the buzzer by changing duty cycle */
46 #define BZR_VOLUME 0.1
47
48 ****
49
50 /* Define frequencies of notes in Hz */
51 #define NOTE_B0 31
52 #define NOTE_C1 33
53 #define NOTE_CS1 35
54 #define NOTE_D1 37
55 #define NOTE_DS1 39
56 #define NOTE_E1 41
57 #define NOTE_F1 44
58 #define NOTE_FS1 46
59 #define NOTE_G1 49
60 #define NOTE_GS1 52
61 #define NOTE_A1 55
62 #define NOTE_AS1 58
63 #define NOTE_B1 62
64 #define NOTE_C2 65
65 #define NOTE_CS2 69
66 #define NOTE_D2 73
67 #define NOTE_DS2 78
68 #define NOTE_E2 82
69 #define NOTE_F2 87
70 #define NOTE_FS2 93
71 #define NOTE_G2 98
72 #define NOTE_GS2 104
73 #define NOTE_A2 110

```

```

74 #define NOTE_AS2 117
75 #define NOTE_B2 123
76 #define NOTE_C3 131
77 #define NOTE_CS3 139
78 #define NOTE_D3 147
79 #define NOTE_DS3 156
80 #define NOTE_E3 165
81 #define NOTE_F3 175
82 #define NOTE_FS3 185
83 #define NOTE_G3 196
84 #define NOTE_GS3 208
85 #define NOTE_A3 220
86 #define NOTE_AS3 233
87 #define NOTE_B3 247
88 #define NOTE_C4 262
89 #define NOTE_CS4 277
90 #define NOTE_D4 294
91 #define NOTE_DS4 311
92 #define NOTE_E4 330
93 #define NOTE_F4 349
94 #define NOTE_FS4 370
95 #define NOTE_G4 392
96 #define NOTE_GS4 415
97 #define NOTE_A4 440
98 #define NOTE_AS4 466
99 #define NOTE_B4 494
100 #define NOTE_C5 523
101 #define NOTE_CS5 554
102 #define NOTE_D5 587
103 #define NOTE_DS5 622
104 #define NOTE_E5 659
105 #define NOTE_F5 698
106 #define NOTE_FS5 740
107 #define NOTE_G5 784
108 #define NOTE_GS5 831
109 #define NOTE_A5 880
110 #define NOTE_AS5 932
111 #define NOTE_B5 988
112 #define NOTE_C6 1047
113 #define NOTE_CS6 1109
114 #define NOTE_D6 1175
115 #define NOTE_DS6 1245
116 #define NOTE_E6 1319
117 #define NOTE_F6 1397
118 #define NOTE_FS6 1480
119 #define NOTE_G6 1568
120 #define NOTE_GS6 1661
121 #define NOTE_A6 1760
122 #define NOTE_AS6 1865
123 #define NOTE_B6 1976
124 #define NOTE_C7 2093
125 #define NOTE_CS7 2217
126 #define NOTE_D7 2349
127 #define NOTE_DS7 2489
128 #define NOTE_E7 2637
129 #define NOTE_F7 2794
130 #define NOTE_FS7 2960
131 #define NOTE_G7 3136
132 #define NOTE_GS7 3322
133 #define NOTE_A7 3520
134 #define NOTE_AS7 3729
135 #define NOTE_B7 3951
136 #define NOTE_C8 4186
137 #define NOTE_CS8 4435
138 #define NOTE_D8 4699
139 #define NOTE_DS8 4978
140 #define REST 0
141
142 /* ***** */
143
144 /* All sequences are defined here */
145
146 /* Test sequence */

```

```

147 int16_t BZR_SEQUENCE_TEST[] = {
148     NOTE_A4, 4, REST, 4, NOTE_A4, 4,
149 };
150
151 /* Super Mario Bros theme - by Koji Kondo*/
152 int16_t BZR_SEQUENCE_MARIO[] = {
153     NOTE_E5, 8, NOTE_E5, 8, REST, 8, NOTE_E5, 8, REST, 8, NOTE_C5, 8, NOTE_E5, 8, //1
154     NOTE_G5, 4, REST, 4, NOTE_G4, 8, REST, 4,
155     NOTE_C5, -4, NOTE_G4, 8, REST, 4, NOTE_E4, -4, // 3
156     NOTE_A4, 4, NOTE_B4, 4, NOTE_AS4, 8, NOTE_A4, 4,
157     NOTE_G4, -8, NOTE_E5, -8, NOTE_G5, -8, NOTE_A5, 4, NOTE_F5, 8, NOTE_G5, 8,
158     REST, 8, NOTE_E5, 4, NOTE_C5, 8, NOTE_D5, 8, NOTE_B4, -4,
159     NOTE_C5, -4, NOTE_G4, 8, REST, 4, NOTE_E4, -4, // repeats from 3
160     NOTE_A4, 4, NOTE_B4, 4, NOTE_AS4, 8, NOTE_A4, 4,
161     NOTE_G4, -8, NOTE_E5, -8, NOTE_G5, -8, NOTE_A5, 4, NOTE_F5, 8, NOTE_G5, 8,
162     REST, 8, NOTE_E5, 4, NOTE_C5, 8, NOTE_D5, 8, NOTE_B4, -4,
163     REST, 4, NOTE_G5, 8, NOTE_FS5, 8, NOTE_F5, 8, NOTE_DS5, 4, NOTE_E5, 8, //7
164     REST, 8, NOTE_GS4, 8, NOTE_A4, 8, NOTE_C4, 8, REST, 8, NOTE_A4, 8, NOTE_C5, 8, NOTE_D5, 8,
165     REST, 4, NOTE_DS5, 4, REST, 8, NOTE_D5, -4,
166     NOTE_C5, 2, REST, 2,
167     REST, 4, NOTE_G5, 8, NOTE_FS5, 8, NOTE_F5, 8, NOTE_DS5, 4, NOTE_E5, 8, //repeats from 7
168     REST, 8, NOTE_GS4, 8, NOTE_A4, 8, NOTE_C4, 8, REST, 8, NOTE_A4, 8, NOTE_C5, 8, NOTE_D5, 8,
169     REST, 4, NOTE_DS5, 4, REST, 8, NOTE_D5, -4,
170     NOTE_C5, 2, REST, 2,
171     NOTE_C5, 8, NOTE_C5, 4, NOTE_C5, 8, REST, 8, NOTE_C5, 8, NOTE_D5, 4, //11
172     NOTE_E5, 8, NOTE_C5, 4, NOTE_A4, 8, NOTE_G4, 2,
173     NOTE_C5, 8, NOTE_C5, 4, NOTE_C5, 8, REST, 8, NOTE_C5, 8, NOTE_D5, 8, NOTE_E5, 8, //13
174     REST, 1,
175     NOTE_C5, 8, NOTE_C5, 4, NOTE_C5, 8, REST, 8, NOTE_C5, 8, NOTE_D5, 4,
176     NOTE_E5, 8, NOTE_C5, 4, NOTE_A4, 8, NOTE_G4, 2,
177     NOTE_E5, 8, NOTE_E5, 8, REST, 8, NOTE_E5, 8, REST, 8, NOTE_C5, 8, NOTE_E5, 4,
178     NOTE_G5, 4, REST, 4, NOTE_G4, 4, REST, 4,
179     NOTE_C5, -4, NOTE_G4, 8, REST, 4, NOTE_E4, -4, // 19
180     NOTE_A4, 4, NOTE_B4, 4, NOTE_AS4, 8, NOTE_A4, 4,
181     NOTE_G4, -8, NOTE_E5, -8, NOTE_G5, -8, NOTE_A5, 4, NOTE_F5, 8, NOTE_G5, 8,
182     REST, 8, NOTE_E5, 4, NOTE_C5, 8, NOTE_D5, 8, NOTE_B4, -4,
183     NOTE_C5, -4, NOTE_G4, 8, REST, 4, NOTE_E4, -4, // repeats from 19
184     NOTE_A4, 4, NOTE_B4, 4, NOTE_AS4, 8, NOTE_A4, 4,
185     NOTE_G4, -8, NOTE_E5, -8, NOTE_G5, -8, NOTE_A5, 4, NOTE_F5, 8, NOTE_G5, 8,
186     REST, 8, NOTE_E5, 4, NOTE_C5, 8, NOTE_D5, 8, NOTE_B4, -4,
187     NOTE_E5, 8, NOTE_C5, 4, NOTE_G4, 8, REST, 4, NOTE_GS4, 4, //23
188     NOTE_A4, 8, NOTE_F5, 4, NOTE_F5, 8, NOTE_A4, 2,
189     NOTE_D5, -8, NOTE_A5, -8, NOTE_A5, -8, NOTE_A5, -8, NOTE_G5, -8, NOTE_F5, -8,
190     NOTE_E5, 8, NOTE_C5, 4, NOTE_A4, 8, NOTE_G4, 2, //26
191     NOTE_E5, 8, NOTE_C5, 4, NOTE_G4, 8, REST, 4, NOTE_GS4, 4,
192     NOTE_A4, 8, NOTE_F5, 4, NOTE_F5, 8, NOTE_A4, 2,
193     NOTE_B4, 8, NOTE_F5, 4, NOTE_F5, 8, NOTE_F5, -8, NOTE_E5, -8, NOTE_D5, -8,
194     NOTE_C5, 8, NOTE_E4, 4, NOTE_E4, 8, NOTE_C4, 2,
195     NOTE_E5, 8, NOTE_C5, 4, NOTE_G4, 8, REST, 4, NOTE_GS4, 4, //repeats from 23
196     NOTE_A4, 8, NOTE_F5, 4, NOTE_F5, 8, NOTE_A4, 2,
197     NOTE_D5, -8, NOTE_A5, -8, NOTE_A5, -8, NOTE_A5, -8, NOTE_G5, -8, NOTE_F5, -8,
198     NOTE_E5, 8, NOTE_C5, 4, NOTE_A4, 8, NOTE_G4, 2, //26
199     NOTE_E5, 8, NOTE_C5, 4, NOTE_G4, 8, REST, 4, NOTE_GS4, 4,
200     NOTE_A4, 8, NOTE_F5, 4, NOTE_F5, 8, NOTE_A4, 2,
201     NOTE_B4, 8, NOTE_F5, 4, NOTE_F5, 8, NOTE_F5, -8, NOTE_E5, -8, NOTE_D5, -8,
202     NOTE_C5, 8, NOTE_E4, 4, NOTE_E4, 8, NOTE_C4, 2,
203     NOTE_C5, 8, NOTE_C5, 4, NOTE_C5, 8, REST, 8, NOTE_C5, 8, NOTE_D5, 8, NOTE_E5, 8,
204     REST, 1,
205     NOTE_C5, 8, NOTE_C5, 4, NOTE_C5, 8, REST, 8, NOTE_C5, 8, NOTE_D5, 4, //33
206     NOTE_E5, 8, NOTE_C5, 4, NOTE_A4, 8, NOTE_G4, 2,
207     NOTE_E5, 8, NOTE_E5, 8, REST, 8, NOTE_E5, 8, REST, 8, NOTE_C5, 8, NOTE_E5, 4,
208     NOTE_G5, 4, REST, 4, NOTE_G4, 4, REST, 4,
209     NOTE_E5, 8, NOTE_C5, 4, NOTE_G4, 8, REST, 4, NOTE_GS4, 4,
210     NOTE_A4, 8, NOTE_F5, 4, NOTE_F5, 8, NOTE_A4, 2,
211     NOTE_D5, -8, NOTE_A5, -8, NOTE_A5, -8, NOTE_A5, -8, NOTE_G5, -8, NOTE_F5, -8,
212     NOTE_E5, 8, NOTE_C5, 4, NOTE_A4, 8, NOTE_G4, 2, //40
213     NOTE_E5, 8, NOTE_C5, 4, NOTE_G4, 8, REST, 4, NOTE_GS4, 4,
214     NOTE_A4, 8, NOTE_F5, 4, NOTE_F5, 8, NOTE_A4, 2,
215     NOTE_B4, 8, NOTE_F5, 4, NOTE_F5, 8, NOTE_F5, -8, NOTE_E5, -8, NOTE_D5, -8,
216     NOTE_C5, 8, NOTE_E4, 4, NOTE_E4, 8, NOTE_C4, 2,
217     //game over sound
218     NOTE_C5, -4, NOTE_G4, -4, NOTE_E4, 4, //45
219     NOTE_A4, -8, NOTE_B4, -8, NOTE_A4, -8, NOTE_GS4, -8, NOTE_AS4, -8, NOTE_GS4, -8,

```

```

220     NOTE_G4,8, NOTE_D4,8, NOTE_E4,-2,
221
222 };
223 /* Dart Vader theme (Imperial March) - Star wars */
224 int16_t BZR_SEQUENCE_IMPERIAL[] = {
225     NOTE_A4, -4, NOTE_A4, -4, NOTE_A4, 16, NOTE_A4, 16,
226     NOTE_A4, 16, NOTE_A4, 16, NOTE_F4, 8, REST, 8,
227     NOTE_A4, -4, NOTE_A4, -4, NOTE_A4, 16, NOTE_A4, 16,
228     NOTE_A4, 16, NOTE_A4, 16, NOTE_F4, 8, REST, 8,
229     NOTE_A4, 4, NOTE_A4, 4, NOTE_A4, 4,
230     NOTE_F4, -8, NOTE_C5, 16, NOTE_A4, 4,
231     NOTE_F4, -8, NOTE_C5, 16, NOTE_A4, 2,
232     NOTE_E5, 4, NOTE_E5, 4, NOTE_E5, 4,
233     NOTE_F5, -8, NOTE_C5, 16, NOTE_A4, 4,
234     NOTE_F4, -8, NOTE_C5, 16, NOTE_A4, 2,
235     NOTE_A5, 4, NOTE_A4, -8, NOTE_A4, 16,
236     NOTE_A5, 4, NOTE_GS5, -8, NOTE_G5, 16,
237     NOTE_DS5, 16, NOTE_D5, 16, NOTE_DS5, 8, REST, 8,
238     NOTE_A4, 8, NOTE_DS5, 4, NOTE_D5, -8, NOTE_CS5, 16,
239     NOTE_C5, 16, NOTE_B4, 16, NOTE_C5, 16, REST, 8,
240     NOTE_F4, 8, NOTE_GS4, 4, NOTE_F4, -8, NOTE_A4, -16,
241     NOTE_C5, 4, NOTE_A4, -8, NOTE_C5, 16, NOTE_E5, 2,
242     NOTE_A5, 4, NOTE_A4, -8, NOTE_A4, 16,
243     NOTE_A5, 4, NOTE_GS5, -8, NOTE_G5, 16,
244     NOTE_DS5, 16, NOTE_D5, 16, NOTE_DS5, 8, REST, 8,
245     NOTE_A4, 8, NOTE_DS5, 4, NOTE_D5, -8, NOTE_CS5, 16,
246     NOTE_C5, 16, NOTE_B4, 16, NOTE_C5, 16, REST, 8,
247     NOTE_F4, 8, NOTE_GS4, 4, NOTE_F4, -8, NOTE_A4, -16,
248     NOTE_A4, 4, NOTE_F4, -8, NOTE_C5, 16,
249     NOTE_A4, 2,
250 };
251
252 ****
253
254 /* Hold informations about sequences */
255 S_BZR_SEQ BZR_SEQUENCES[] = {
256     /* TESTING */
257     {
258         .tempo = 200,
259         .size = sizeof(BZR_SEQUENCE_TEST),
260         .notes = BZR_SEQUENCE_TEST,
261     },
262     /* MARIO BROS SONG */
263     {
264         .tempo = 200,
265         .size = sizeof(BZR_SEQUENCE_MARIO),
266         .notes = BZR_SEQUENCE_MARIO,
267     },
268     /* IMPERIAL MARCH */
269     {
270         .tempo = 120,
271         .size = sizeof(BZR_SEQUENCE_IMPERIAL),
272         .notes = BZR_SEQUENCE_IMPERIAL,
273     },
274 };
275
276 ****
277
278 /* Declaration of global application data */
279 BZR_DATA bsrData;
280
281 ****
282
283 /* Static functions declaration */
284
285 static void BZR_SetFrequency(uint16_t frequency);
286 static void BZR_SetCounter(int8_t tempo);
287
288 ****
289
290 /**
291 * @brief BZR_Initialize
292 *

```

```

293 * Initialize buzzer state machine
294 *
295 * @param void
296 * @return void
297 */
298 void BZR_Initialize ( void )
299 {
300     /* Place the buzzer state machine in its initial state. */
301     bsrData.state = BZR_STATE_IDLE;
302
303     /* Flag to indicate a new sequence available */
304     bsrData.newSequence = false;
305
306     /* Calculate timer frequency only one time */
307     bsrData.tmrFrequency = (uint32_t)(SYS_CLK_FREQ /
308                                         PLIB_TMR_PrescaleGet(BZR_TMR_ID));
309
310     /* Init counter used to count time of notes */
311     CNT_Initialize(&bsrData.counterPlay, 0x00);
312 }
313
314 /*****
315 /**
316 * @brief BZR_Tasks
317 *
318 * Execute buzzer state machine, should be called cyclically
319 *
320 * @param void
321 * @return void
322 */
323 void BZR_Tasks ( void )
324 {
325
326     /* Check the application's current state. */
327     switch ( bsrData.state )
328     {
329         /* Buzzer waiting for new sequence */
330         case BZR_STATE_IDLE:
331         {
332             if(bsrData.newSequence)
333             {
334                 bsrData.state = BZR_STATE_NOTE;
335             }
336             break;
337         }
338
339
340         /* Buzzer getting the note to play */
341         case BZR_STATE_NOTE:
342         {
343             BZR_SetFrequency(bsrData.currentNote[0]);
344             BZR_SetCounter(bsrData.currentNote[1]);
345             bsrData.state = BZR_STATE_PLAYING;
346             break;
347         }
348
349         /* Buzzer playing the note and waiting */
350         case BZR_STATE_PLAYING:
351         {
352             if(CNT_Check(&bsrData.counterPlay))
353             {
354                 if(bsrData.currentNote >= bsrData.lastNote)
355                 {
356                     PLIB_TMR_Stop(BZR_TMR_ID);
357                     PLIB_OC_Disable(BZR_OC_ID);
358                     bsrData.newSequence = false;
359                     bsrData.state = BZR_STATE_IDLE;
360                 }
361                 else
362                 {
363                     bsrData.currentNote += 2;
364                     bsrData.state = BZR_STATE_NOTE;
365                 }
366             }
367         }
368     }
369 }

```

```

366         }
367     break;
368 }
369 
370     /* The default state should never be executed. */
371 default:
372 {
373     /* TODO: Handle error in application's state machine. */
374     break;
375 }
376 }
377 }
378 
379 /*****
380 /**
381 * @brief BZR_PlaySequence
382 *
383 * Play a music sequence using the state machine
384 *
385 * @param E_BZR_SEQ song Call the music you wann play !
386 * @return void
387 */
388 void BZR_PlaySequence(E_BZR_SEQ song)
389 {
390     bsrData.sequence = BZR_SEQUENCES[song].notes;
391 
392     bsrData.tempo = BZR_SEQUENCES[song].tempo;
393 
394     bsrData.currentNote = bsrData.sequence;
395 
396     bsrData.lastNote = bsrData.sequence + BZR_SEQUENCES[song].size / 2 - 2;
397 
398     bsrData.newSequence = true;
399 }
400 
401 *****/
402 /**
403 /**
404 * @brief BZR_SetFrequency
405 *
406 * Set the frequency of the timer to play a note
407 * Start the timer and OC except if frequency is 0
408 *
409 * @param uint16_t frequency Frequency to play
410 * @return void
411 */
412 static void BZR_SetFrequency(uint16_t frequency)
413 {
414     uint16_t period_tmr;
415 
416     PLIB_TMR_Stop(BZR_TMR_ID);
417     PLIB_OC_Disable(BZR_OC_ID);
418 
419     if(frequency != 0)
420     {
421         period_tmr = (uint16_t)( bsrData.tmrFrequency / frequency);
422 
423         PLIB_TMR_Period16BitSet(BZR_TMR_ID, period_tmr);
424         PLIB_OC_PulseWidth16BitSet(BZR_OC_ID, period_tmr * BZR_VOLUME);
425 
426         PLIB_TMR_Start(BZR_TMR_ID);
427         PLIB_OC_Enable(BZR_OC_ID);
428     }
429 }
430 
431 *****/
432 /**
433 * @brief BZR_SetCounter
434 *
435 * Set the duration of the note, using musical tempo
436 * (whole, half, quarter, eighth, ...)
437 */
438 
```

```

439 *
440 * @param int8_t tempo note tempo
441 * @return void
442 */
443 static void BZR_SetCounter(int8_t tempo)
444 {
445     uint32_t counter_ms;
446
447     if(tempo > 0)
448     {
449         counter_ms = (uint32_t)((60000 * 4 / bzsData.tempo) / tempo);
450     }
451     else if(tempo < 0)
452     {
453         counter_ms = (uint32_t)((60000 * 4 / bzsData.tempo) / abs(tempo));
454         counter_ms = counter_ms * 1.5;
455     }
456     else
457     {
458         counter_ms = 0;
459     }
460
461     CNT_Set(&bzsData.counterPlay, counter_ms);
462     CNT_Reset(&bzsData.counterPlay);
463 }
464
465 /***** End of File *****/
466
467 /* End of File *****/
468

```



```

74  /* Buzzer state machine */
75  typedef enum
76  {
77      /* Buzzer waiting for new sequence */
78      BZR_STATE_IDLE,
79
80      /* Buzzer getting the note to play */
81      BZR_STATE_NOTE,
82
83      /* Buzzer playing the note and waiting */
84      BZR_STATE_PLAYING,
85
86 } BZR_STATES;
87
88 /*****
89
90 /* Buzzer structure of global datas */
91 typedef struct
92 {
93     /* The buzzer current state */
94     BZR_STATES state;
95
96     bool newSequence;
97
98     uint32_t tmrFrequency;
99
100    uint16_t tempo;
101    int16_t *currentNote;
102    int16_t *lastNote;
103    int16_t *sequence;
104
105    S_Counter counterPlay;
106
107 } BZR_DATA;
108
109 *****/
110
111 /**
112 * @brief BZR_Initialize
113 *
114 * Initialize buzzer state machine
115 *
116 * @param void
117 * @return void
118 */
119 void BZR_Initialize ( void );
120
121 *****/
122
123 /**
124 * @brief BZR_Tasks
125 *
126 * Execute buzzer state machine, should be called cyclically
127 *
128 * @param void
129 * @return void
130 */
131 void BZR_Tasks ( void );
132
133 *****/
134
135 /**
136 * @brief BZR_PlaySequence
137 *
138 * Play a music sequence using the state machine
139 *
140 * @param E_BZR_SEQ song Call the music you wann play !
141 * @return void
142 */
143 void BZR_PlaySequence(E_BZR_SEQ song);
144
145 *****/
146

```

```
147 #endif /* _BZR_H */  
148 /* End of File *****/
```

```

1  ****
2  *
3  *      [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
4  *      | | \_ | / | \_ | / | \_ | / | \_ | / | \_ | / | \_ | / | \_ | / | \_ | / | \_ | / | \_
5  *      | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
6  *      | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
7  *      | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
8  *
9  ****
10 *
11 * File           : TLC5973.c
12 * Version        : 1.0
13 *
14 ****
15 *
16 * Description   : Interface and data conversion for TLC5973
17 *                   Number of drivers in serie has to be defined by user
18 *                   A cycle represent a bit encoded as defined by datasheet
19 *
20 ****
21 *
22 * Author         : Miguel Santos
23 * Date          : 14.09.2023
24 *
25 ****
26 *
27 * MPLAB X       : 5.45
28 * XC32          : 2.50
29 * Harmony        : 2.06
30 *
31 ****
32
33 #include "TLC5973.h"
34 #include "SerialTimer.h"
35
36 ****
37
38 /* Number of drivers connected in series */
39 #define DRIVER_COUNT 3
40
41 *-----
42
43 /* Number of channels per driver */
44 #define CHANNEL_COUNT 3
45
46 /* Single field cycles count */
47 #define FLD_CYCLE_COUNT 12
48
49 /* Sequences cycles count */
50 #define DWS_CYCLE_COUNT 48
51 #define DWS_TOTAL_COUNT (DWS_CYCLE_COUNT * DRIVER_COUNT)
52 #define EOS_CYCLE_COUNT 4
53 #define EOS_TOTAL_COUNT (EOS_CYCLE_COUNT * (DRIVER_COUNT - 1))
54 #define GSL_CYCLE_COUNT 10
55
56 /* Size of the buffer to store cycles */
57 #define TLC_BUFFER_SIZE (DWS_TOTAL_COUNT + EOS_TOTAL_COUNT + GSL_CYCLE_COUNT)
58
59 /* Buffer offset for each driver */
60 #define DWS_OFFSET (DWS_CYCLE_COUNT + EOS_CYCLE_COUNT)
61
62 /* Each driver starts with a write command */
63 #define WRITE_COMMAND 0x3AA
64
65 /* Cycles values encoding */
66 #define CYCLE_CODE_HIGH 0x05
67 #define CYCLE_CODE_LOW 0x01
68 #define CYCLE_CODE_SKIP 0x00
69
70 /* Mask to MSB bit in field */
71 #define FIELD_MASK 0x800
72
73 ****

```

```

74
75 /* Struct of a single channel */
76 typedef struct
77 {
78     /* Output value of the channel */
79     uint16_t out;
80
81     /* Pointers where values will be stored in CYCLE buffer */
82     uint8_t *p_out;
83
84     /* Flag new value set on channel */
85     bool newValue;
86
87 }S_TLC_CHANNEL;
88
89 /* Struct of a single driver */
90 typedef struct
91 {
92     /* Each driver has 3 output channel */
93     S_TLC_CHANNEL channel[CHANNEL_COUNT];
94
95     /* Flag new value set on driver */
96     bool newValue;
97
98 }S_TLC_DRIVER;
99
100 /*****
101 S_TLC_CHANNEL tlcCommands[DRIVER_COUNT];
102
103 S_TLC_DRIVER tlcDrivers[DRIVER_COUNT];
104
105 uint8_t cyclesBuffer[TLC_BUFFER_SIZE];
106
107 *****/
108 /*****
109
110 static bool TLC_SetChannel(S_TLC_CHANNEL *channel, uint16_t value);
111 static bool TLC_TranslateChannel(S_TLC_CHANNEL *channel);
112 static bool TLC_TranslateDriver(S_TLC_DRIVER *driver);
113 static bool TLC_TranslateAll(void);
114
115 *****/
116
117 void TLC_Initialize(void)
118 {
119     uint8_t i_buff;
120     uint8_t i_drv;
121     uint8_t i_cha;
122
123     /* Initialize TLC_Buffer with CYCLE_SKIP */
124     for(i_buff = 0; i_buff < TLC_BUFFER_SIZE; i_buff++)
125     {
126         cyclesBuffer[i_buff] = CYCLE_CODE_SKIP;
127     }
128
129     /* Initialize TLC_Drivers with default values */
130     for(i_drv = 0; i_drv < DRIVER_COUNT; i_drv++)
131     {
132         /* Commands are static channels in buffer */
133         tlcCommands[i_drv].out = WRITE_COMMAND;
134         tlcCommands[i_drv].p_out = cyclesBuffer +
135                                     (DWS_OFFSET * i_drv);
136         tlcCommands[i_drv].newValue = true;
137         TLC_TranslateChannel(&tlcCommands[i_drv]);
138
139         for(i_cha = 0; i_cha < CHANNEL_COUNT; i_cha++)
140         {
141             tlcDrivers[i_drv].channel[i_cha].out = 0x00;
142             tlcDrivers[i_drv].channel[i_cha].newValue = true;
143             tlcDrivers[i_drv].channel[i_cha].p_out = cyclesBuffer +
144                                             (DWS_OFFSET * i_drv) +
145                                             (FLD_CYCLE_COUNT * (i_cha + 1));
146         }
147     }
148 }
```

```

147         tlcDrivers[i_drv].newValue = true;
148     }
149
150     TLC_TranslateAll();
151
152     /* Initialize the serial timer */
153     STR_Init();
154 }
155 /*************************************************************************/
156
157 bool TLC_Transmit( void )
158 {
159     bool status;
160
161     status = TLC_TranslateAll();
162
163     if(status)
164     {
165         STR_AddBuffer(cyclesBuffer, TLC_BUFFER_SIZE);
166         STR_Start();
167     }
168
169     return status;
170 }
171
172 /*************************************************************************/
173
174 bool TLC_SetAll(uint16_t out0, uint16_t out1, uint16_t out2)
175 {
176     bool status;
177     uint8_t i_drv;
178
179     status = false;
180
181     for(i_drv = 0; i_drv < DRIVER_COUNT; i_drv++)
182     {
183         status |= TLC_SetDriver(i_drv, out0, out1, out2);
184     }
185
186     return status;
187 }
188
189 /*************************************************************************/
190
191 bool TLC_SetDriver(E_TLC_DRV_ID driver, uint16_t out0, uint16_t out1, uint16_t out2)
192 {
193     bool status;
194
195     status = false;
196
197     if(driver < DRIVER_COUNT)
198     {
199         /* Set channel 0 */
200         status |= TLC_SetChannel(&tlcDrivers[driver].channel[0], out0);
201         /* Set channel 1 */
202         status |= TLC_SetChannel(&tlcDrivers[driver].channel[1], out1);
203         /* Set channel 2 */
204         status |= TLC_SetChannel(&tlcDrivers[driver].channel[2], out2);
205
206         tlcDrivers[driver].newValue = status;
207     }
208
209     return status;
210 }
211
212 /*************************************************************************/
213
214 static bool TLC_SetChannel(S_TLC_CHANNEL *channel, uint16_t value)
215 {
216     bool status;
217
218     /* Detect if there's a new value */
219     status = (channel->out != value);

```

```

220
221     if(status)
222     {
223         channel->out = value;
224         channel->newValue = true;
225     }
226
227     return status;
228 }
229
230 *****
231
232 static bool TLC_TranslateAll( void )
233 {
234     bool status;
235     uint8_t i_drv;
236
237     status = false;
238
239     for(i_drv = 0; i_drv < DRIVER_COUNT; i_drv++)
240     {
241         if(tlcDrivers[i_drv].newValue)
242         {
243             status |= TLC_TranslateDriver(&tlcDrivers[i_drv]);
244             tlcDrivers[i_drv].newValue = false;
245         }
246     }
247
248     return status;
249 }
250
251 *****
252
253 static bool TLC_TranslateDriver(S_TLC_DRIVER *driver)
254 {
255     bool status;
256     uint8_t i_cha;
257
258     status = true;
259
260     for(i_cha = 0; i_cha < CHANNEL_COUNT; i_cha++)
261     {
262         if(driver->channel[i_cha].newValue)
263         {
264             status &= TLC_TranslateChannel(&driver->channel[i_cha]);
265             driver->channel[i_cha].newValue = false;
266             driver->newValue = true;
267         }
268     }
269
270     return status;
271 }
272
273 *****
274
275 static bool TLC_TranslateChannel(S_TLC_CHANNEL *channel)
276 {
277     bool status;
278     uint8_t i_bits;
279
280     status = false;
281
282     /* Watchdog */
283     if(channel->p_out != NULL)
284     {
285         for(i_bits = 0 ; i_bits < FLD_CYCLE_COUNT ; i_bits++)
286         {
287             if( channel->out & (FIELD_MASK >> i_bits) )
288             {
289                 channel->p_out[i_bits] = CYCLE_CODE_HIGH;
290             }
291             else
292             {

```

```
293         channel->p_out[i_bits] = CYCLE_CODE_LOW;
294     }
295 }
296 status = true;
297 }
298
299 return status;
300 }
301 /* **** End of File ****
302 */
303
304
305
```

```

1  ****
2  *
3  *   [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
4  *   [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
5  *   [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
6  *   [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
7  *   [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
8  *
9  ****
10 *
11 * File           : TLC5973.c
12 * Version        : 1.0
13 *
14 ****
15 *
16 * Description   : Interface and data conversion for TLC5973
17 *
18 ****
19 *
20 * Author         : Miguel Santos
21 * Date          : 14.09.2023
22 *
23 ****
24 *
25 * MPLAB X       : 5.45
26 * XC32          : 2.50
27 * Harmony        : 2.06
28 *
29 ****
30
31 #ifndef TLC5973_H
32 #define TLC5973_H
33
34 ****
35
36 #include <stdint.h>
37 #include <stdbool.h>
38 #include <stddef.h>
39
40 ****
41
42 typedef enum{
43     TLC_DRV_ID_0 = 0x00,
44     TLC_DRV_ID_1 = 0x01,
45     TLC_DRV_ID_2 = 0x02,
46 }E_TLC_DRV_ID;
47
48 ****
49
50 void TLC_Initialize( void );
51
52 ****
53
54 bool TLC_Transmit( void );
55
56 ****
57
58 bool TLC_SetAll(uint16_t out0, uint16_t out1, uint16_t out2);
59
60 ****
61
62 bool TLC_SetDriver(E_TLC_DRV_ID driver, uint16_t out0, uint16_t out1, uint16_t out2);
63
64 ****
65
66 #endif /* TLC5973_H */
67
68 /* ****
69 End of File
70 */
71

```

```

1  ****
2  *
3  *      _____| |_____| |_____| |_____| \ / | |_____| |_____| |_____| | . ' _____\ |
4  *     | |_____| |_____| |_____| |_____| | \ / | |_____| |_____| |_____| | | ( _____) |
5  *     | |_____| |_____| |_____| |_____| | \ / | |_____| |_____| |_____| | | \ _____) |
6  *     | |_____| |_____| |_____| |_____| | \ / | |_____| |_____| |_____| | | \ _____) |
7  *     | |_____| |_____| |_____| |_____| | \ / | |_____| |_____| |_____| | | \ _____) |
8  *
9  ****
10 *
11 * File           : SerialTimer.c
12 * Version        : 1.0
13 *
14 ****
15 *
16 * Description   : Emulate a serial port on a GPIO pin using a timer
17 *
18 ****
19 *
20 * Author         : Miguel Santos
21 * Date          : 14.09.2023
22 *
23 ****
24 *
25 * MPLAB X       : 5.45
26 * XC32          : 2.50
27 * Harmony        : 2.06
28 *
29 ****
30
31 #include "system_definitions.h"
32 #include "SerialTimer.h"
33
34 /* Pin where the serial datas will be output */
35 #define SERIAL_PIN LATDbits.LATD0
36
37 /* Pin to control the timer related to the serial emulator */
38 #define STR_TMR_ID DRV_TMR_INDEX_1
39
40 /* Number of bits to transmit per data */
41 #define SERIAL_BITS 5
42
43 ****
44
45 uint8_t strBuffer[STR_BUFFER_SIZE];
46 uint8_t *strPointer;
47 uint8_t *strLast;
48 uint8_t bitPosition;
49
50 ****
51
52 void STR_Init( void )
53 {
54     uint8_t i_buffer;
55
56     for(i_buffer = 0 ; i_buffer < STR_BUFFER_SIZE ; i_buffer++)
57     {
58         strBuffer[i_buffer] = STR_BUFFER_DEFAULT;
59     }
60     strPointer = strBuffer;
61     bitPosition= 0;
62 }
63
64 ****
65
66 void STR_AddBuffer(uint8_t *data, uint8_t size)
67 {
68     uint8_t i_data;
69
70     for(i_data = 0; i_data < size ; i_data++)
71     {
72         strBuffer[i_data] = *(data + i_data);
73     }

```

```

74     strLast = strBuffer + size - 1;
75 }
76
77 /****** */
78 void STR_Start( void )
79 {
80     DRV_TMR_Start(STR_TMR_ID);
81 }
82 /****** */
83
84 void STR_CallBack( void )
85 {
86     SERIAL_PIN = (*strPointer >> bitPosition) & 0x01;
87
88     bitPosition++;
89
90     /* Go back to lsb when reached limit */
91     if(bitPosition>= SERIAL_BITS)
92     {
93         bitPosition= 0;
94         strPointer++;
95     }
96
97     /* Disable timer when reached end of buffer */
98     if(strPointer > strLast)
99     {
100        DRV_TMR_Stop(STR_TMR_ID);
101        strPointer = strBuffer;
102    }
103
104 }
105 /* End of File ***** */
106
107
108
109

```

```

1  ****
2  *
3  *   _____| |_____| |_____| |_____| \ / | |_____| |_____| |_____| | . ' _____\ |
4  *  | |_____| |_____| |_____| |_____| | \ / | |_____| |_____| |_____| | | ( _____) |
5  *  | |_____| |_____| |_____| |_____| | \ / | |_____| |_____| |_____| | | \ _____) |
6  *  | |_____| |_____| |_____| |_____| | \ / | |_____| |_____| |_____| | | \ _____) |
7  *  | |_____| |_____| |_____| |_____| | \ / | |_____| |_____| |_____| | | \ _____) |
8  *
9  ****
10 *
11 * File           : SerialTimer.h
12 * Version        : 1.0
13 *
14 ****
15 *
16 * Description   : Emulate a serial port on a GPIO pin using a timer
17 *
18 ****
19 *
20 * Author         : Miguel Santos
21 * Date          : 14.09.2023
22 *
23 ****
24 *
25 * MPLAB X       : 5.45
26 * XC32          : 2.50
27 * Harmony        : 2.06
28 *
29 ****
30 /* Guard against multiple inclusion */
31 #ifndef SERIAL_TIMER_H
32 #define SERIAL_TIMER_H
33
34 ****
35 //*****
36
37 #include <stdint.h>
38 #include <stdbool.h>
39
40 //*****
41
42 /* Size of the buffer */
43 #define STR_BUFFER_SIZE 0xFF
44
45 /* Default value of the buffer */
46 #define STR_BUFFER_DEFAULT 0x00
47
48 //*****
49
50 void STR_Init( void );
51
52 //*****
53
54 void STR_AddBuffer(uint8_t *data, uint8_t size);
55
56 //*****
57
58 void STR_Start( void );
59
60 //*****
61
62 void STR_CallBack( void );
63
64 #endif /* SERIAL_TIMER_H */
65
66 /* End of File *****/
67

```

```

1  ****
2  *
3  *      _____| |_____| |_____| |_____| \ / | |_____| |_____| |_____| | . ' _____\ |
4  *     | |_____| |_____| |_____| |_____| | \ / | |_____| |_____| |_____| | . ' _____\ |
5  *     | |_____| |_____| |_____| |_____| | \ / | |_____| |_____| |_____| | . ' _____\ |
6  *     | |_____| |_____| |_____| |_____| | \ / | |_____| |_____| |_____| | . ' _____\ |
7  *     | |_____| |_____| |_____| |_____| | \ / | |_____| |_____| |_____| | . ' _____\ |
8  *
9  ****
10 *
11 * File          : fifo.c
12 * Version       : 1.0
13 *
14 ****
15 *
16 * Description   : Managing a FIFO using descriptor and pointers
17 *                   Maximal size of FIFO is 255
18 *
19 ****
20 *
21 * Author        : Miguel Santos
22 * Date          : 25.09.2023
23 *
24 ****
25 *
26 * MPLAB X       : 5.45
27 * XC32          : 2.50
28 * Harmony        : 2.06
29 *
30 ****
31
32 #include "FIFO.h"
33
34 ****
35
36 /**
37 * @brief FIFO_Init
38 *
39 * This function initializes a FIFO with the provided parameters,
40 * setting its size, start address, and initializing all elements
41 * to the given initial value.
42 *
43 * @param fifoDescriptor Pointer to the FIFO descriptor structure.
44 * @param fifoSize       The size of the FIFO.
45 * @param fifoStart      Pointer to the beginning of the FIFO memory.
46 * @param initialValue   The initial value to set for all elements in the FIFO.
47 * @return void
48 */
49 void FIFO_Initialize( S_Fifo *fifoDescriptor, uint16_t fifoSize,
50                      uint8_t *fifoStart, uint8_t initialValue )
51 {
52     /* Local variables declaration */
53     uint8_t *fifoPosition;
54     uint16_t i;
55
56     fifoPosition = fifoStart;
57
58     /* Fifo descriptor values initialisation */
59     fifoDescriptor->size    = fifoSize;
60     fifoDescriptor->start   = fifoStart;
61     fifoDescriptor->end     = fifoStart + fifoSize - 1;
62     fifoDescriptor->write   = fifoStart;
63     fifoDescriptor->read    = fifoStart;
64
65     /* Loop through entire fifo to set initial value */
66     for( i = 0 ; i < fifoSize ; i++ )
67     {
68         fifoPosition[i] = initialValue;
69     }
70 }
71
72 ****
73

```

```

74 /**
75 * @brief FIFO_GetWriteSpace
76 *
77 * This function calculates the available space for writing
78 * in the provided FIFO descriptor,
79 * taking into account the current read and write positions.
80 *
81 * @param fifoDescriptor Pointer to the FIFO descriptor structure.
82 * @return The available space for writing in the FIFO.
83 */
84 uint8_t FIFO_GetWriteSpace( S_Fifo *fifoDescriptor )
85 {
86     /* Local variables declaration */
87     int32_t writeSpace;
88
89     /* Calculate space available */
90     writeSpace = fifoDescriptor->read - fifoDescriptor->write - 1;
91
92     /* Adjust to positive if needed */
93     if (writeSpace < 0)
94     {
95         writeSpace = writeSpace + fifoDescriptor->size;
96     }
97
98     /* Return value */
99     return (uint16_t)writeSpace;
100 }
101
102 /*****
103 ****
104 /**
105 * @brief FIFO_GetReadSpace
106 *
107 *
108 * This function calculates the available space for reading
109 * from the provided FIFO descriptor,
110 * taking into account the current read and write positions.
111 *
112 * @param fifoDescriptor Pointer to the FIFO descriptor structure.
113 * @return The available space for reading from the FIFO.
114 */
115 uint8_t FIFO_GetReadSpace( S_Fifo *fifoDescriptor )
116 {
117     /* Local variables declaration */
118     int32_t readSpace;
119
120     /* Calculate space available */
121     readSpace = fifoDescriptor->write - fifoDescriptor->read;
122
123     /* Adjust to positive if needed */
124     if (readSpace < 0)
125     {
126         readSpace = readSpace + fifoDescriptor->size;
127     }
128
129     /* Return value */
130     return (uint16_t)readSpace;
131 }
132
133 /*****
134 ****
135 /**
136 * @brief FIFO_Add
137 *
138 * This function attempts to put the specified character into the FIFO.
139 * If the FIFO is full, returns 0 (FIFO FULL),
140 * otherwise, it puts the character and returns 1 (OK).
141 *
142 * @param fifoDescriptor Pointer to the FIFO descriptor structure.
143 * @param value           The value to add to the FIFO.
144 * @return true if (OK), false if (FIFO FULL).
145 */
146 bool FIFO_Add( S_Fifo *fifoDescriptor , uint8_t value )

```

```

147 {
148     /* Local variables declaration */
149     bool writeStatus;
150
151     /* True = space available ; False = FIFO full */
152     writeStatus = FIFO_GetWriteSpace(fifoDescriptor);
153
154     if (writeStatus)
155     {
156         /* Write the value into the FIFO */
157         *(fifoDescriptor->write) = value;
158
159         /* Increment the write pointer */
160         fifoDescriptor->write++;
161
162         /* Handle wrap-around */
163         if (fifoDescriptor->write > fifoDescriptor->end)
164         {
165             fifoDescriptor->write = fifoDescriptor->start;
166         }
167     }
168
169     /* Return status */
170     return writeStatus;
171 }
172
173 /*****
174 ****
175 /**
176 * @brief FIFO_GetData
177 *
178 * This function attempts to get a value from the FIFO.
179 * If the FIFO is empty, returns 0 (FIFO EMPTY),
180 * otherwise, it gets the value and returns 1 (OK).
181 *
182 * @param fifoDescriptor Pointer to the FIFO descriptor structure.
183 * @param value           Pointer to store the retrieved value.
184 * @return true if (OK), false if (FIFO EMPTY).
185 */
186 bool FIFO_GetData( S_Fifo *fifoDescriptor , uint8_t *value )
187 {
188     /* Local variables declaration */
189     bool readStatus;
190
191     /* True = values in FIFO ; False = FIFO empty */
192     readStatus = FIFO_GetReadSpace(fifoDescriptor);
193
194     if (readStatus)
195     {
196         /* Read value in FIFO */
197         *value = *(fifoDescriptor->read);
198
199         /* Increment read pointer */
200         fifoDescriptor->read++;
201
202         /* Handle wrap-around */
203         if (fifoDescriptor->read > fifoDescriptor->end)
204         {
205             fifoDescriptor->read = fifoDescriptor->start;
206         }
207     }
208     else
209     {
210         /* Value read = NULL */
211         *value = 0;
212     }
213
214     /* Return status */
215     return readStatus;
216 }
217
218 *****/

```

```

220 /**
221 * @brief FIFO_GetBuffer
222 *
223 * This function attempts to get all the FIFO in a buffer.
224 * If the FIFO is empty, returns 0 (FIFO EMPTY),
225 * otherwise, it gets the value and returns 1 (OK).
226 *
227 * @param fifoDescriptor Pointer to the FIFO descriptor structure.
228 * @param buffer Pointer to the buffer to store the FIFO.
229 * @return true if (OK), false if (FIFO EMPTY).
230 */
231 bool FIFO_GetBuffer( S_Fifo *fifoDescriptor , uint8_t *buffer )
232 {
233     /* Local variables declaration */
234     bool readStatus;
235     uint8_t value;
236     uint8_t *p_buffer;
237
238     readStatus = false;
239     value = 0x00;
240     p_buffer = buffer;
241
242     /* True = values in FIFO ; False = FIFO empty */
243     while(FIFO_GetData(fifoDescriptor, &value))
244     {
245         *p_buffer = value;
246         p_buffer++;
247         readStatus = true;
248     }
249
250     /* Return status */
251     return readStatus;
252 }
253
254 /***** End of File *****/
255
256 /* End of File *****/
257

```



```

74 * in the provided FIFO descriptor,
75 * taking into account the current read and write positions.
76 *
77 * @param fifoDescriptor Pointer to the FIFO descriptor structure.
78 * @return The available space for writing in the FIFO.
79 */
80 uint8_t FIFO_GetWriteSpace( S_Fifo *fifoDescriptor );
81
82 /*****
83 /**
84 * @brief FIFO_GetReadSpace
85 *
86 *
87 * This function calculates the available space for reading
88 * from the provided FIFO descriptor,
89 * taking into account the current read and write positions.
90 *
91 * @param fifoDescriptor Pointer to the FIFO descriptor structure.
92 * @return The available space for reading from the FIFO.
93 */
94
95 uint8_t FIFO_GetReadSpace( S_Fifo *fifoDescriptor );
96
97 /*****
98 /**
99 * @brief FIFO_Add
100 *
101 * This function attempts to put the specified character into the FIFO.
102 * If the FIFO is full, returns 0 (FIFO FULL),
103 * otherwise, it puts the character and returns 1 (OK).
104 *
105 * @param fifoDescriptor Pointer to the FIFO descriptor structure.
106 * @param value           The value to add to the FIFO.
107 * @return 1 if (OK), 0 if (FIFO FULL).
108 */
109
110 bool FIFO_Add( S_Fifo *fifoDescriptor , uint8_t value );
111
112 /*****
113 /**
114 * @brief FIFO_GetData
115 *
116 * This function attempts to get a value from the FIFO.
117 * If the FIFO is empty, returns 0 (FIFO EMPTY),
118 * otherwise, it gets the value and returns 1 (OK).
119 *
120 * @param fifoDescriptor Pointer to the FIFO descriptor structure.
121 * @param value           Pointer to store the retrieved value.
122 * @return 1 if (OK), 0 if (FIFO EMPTY).
123 */
124
125 bool FIFO_GetData( S_Fifo *fifoDescriptor , uint8_t *value );
126
127 /*****
128 /**
129 * @brief FIFO_GetBuffer
130 *
131 * This function attempts to get all the FIFO in a buffer.
132 * If the FIFO is empty, returns 0 (FIFO EMPTY),
133 * otherwise, it gets the value and returns 1 (OK).
134 *
135 * @param fifoDescriptor Pointer to the FIFO descriptor structure.
136 * @param buffer          Pointer to the buffer to sore the FIFO.
137 * @return true if (OK), false if (FIFO EMPTY).
138 */
139
140 bool FIFO_GetBuffer( S_Fifo *fifoDescriptor , uint8_t *buffer );
141
142 /*****
143 /**
144 * End of File ****
145 */

```

```

1  ****
2  *
3  *   _____| |_____| |_____| |_____| \ / | |_____| |_____| |_____| | . ' _____\ |
4  *   | |_____| |_____| |_____| |_____| | \ / | |_____| |_____| |_____| | | ( _____) |
5  *   | |_____| |_____| |_____| |_____| | | \ / | |_____| |_____| |_____| | | | \ _____| |
6  *   | |_____| |_____| |_____| |_____| | | | \ / | |_____| |_____| |_____| | | | | \ _____| |
7  *   | |_____| |_____| |_____| |_____| | | | | \ / | |_____| |_____| |_____| | | | | | \ _____| |
8  *
9  ****
10 *
11 * File          : counter.c
12 * Version       : 1.0
13 *
14 ****
15 *
16 * Description   : Managing time counters for application
17 *                   based on timer (1 by default)
18 *
19 ****
20 *
21 * Author        : Miguel Santos
22 * Date          : 14.09.2023
23 *
24 ****
25 *
26 * MPLAB X      : 5.45
27 * XC32          : 2.50
28 * Harmony        : 2.06
29 *
30 ****
31
32 #include "counter.h"
33
34 ****
35
36 /* Global infinite system counter */
37 uint32_t SYS_counter = 0;
38
39 ****
40
41 void CNT_Initialize(S_Counter *counter, uint32_t target)
42 {
43     counter->value = SYS_counter;
44     counter->target = target;
45 }
46
47 ****
48
49 bool CNT_Check( S_Counter *counter )
50 {
51     bool checkStatus;
52
53     checkStatus = ((SYS_counter - counter->value) >= counter->target);
54
55     if(checkStatus)
56     {
57         CNT_Reset(counter);
58     }
59
60     return checkStatus;
61 }
62
63 ****
64
65 void CNT_Set( S_Counter *counter, uint32_t target )
66 {
67     counter->target = target;
68 }
69
70 ****
71
72 void CNT_Reset( S_Counter *counter )
73 {

```

```
74     counter->value = SYS_counter;
75 }
76
77 /****** */
78
79 void CNT_CallBack( void )
80 {
81     SYS_counter++;
82 }
83
84 /****** */
85
86 /* End of File ***** */
87
```

```

1  ****
2  *
3  *   _____| |_____| |_____| |_____| \ / | |_____| |_____| |_____| | . ' _____\ |
4  *  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
5  *  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
6  *  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
7  *  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
8  *
9  ****
10 *
11 * File           : counter.c
12 * Version        : 1.0
13 *
14 ****
15 *
16 * Description   : Managing time counters for application
17 *                   based on timer (1 by default)
18 *
19 ****
20 *
21 * Author         : Miguel Santos
22 * Date          : 14.09.2023
23 *
24 ****
25 *
26 * MPLAB X       : 5.45
27 * XC32          : 2.50
28 * Harmony        : 2.06
29 *
30 ****
31
32 #ifndef COUNTER_H
33 #define COUNTER_H
34
35 ****
36
37 #include <stdint.h>
38 #include <stdbool.h>
39
40 ****
41
42 #define TIMER_PERIOD_MS 1
43
44 ****
45
46 typedef struct{
47     uint32_t value;
48     uint32_t target;
49 }S_Counter;
50
51 ****
52
53 void CNT_Initialize(S_Counter *counter, uint32_t target);
54
55 ****
56
57 bool CNT_Check( S_Counter *counter );
58
59 ****
60 void CNT_Set( S_Counter *counter, uint32_t target );
61 void CNT_Reset( S_Counter *counter );
62
63 ****
64
65 void CNT_CallBack( void );
66
67 ****
68
69 #endif /* COUNTER_H */
70
71 /* End of File ****
72

```

Projet ETML-ES – Modification

PROJET:	Badge pour place de travail		
Entreprise/Client:	ETML-ES	Département:	SLO
Demandé par (Prénom, Nom):	-	Date:	25.09.23
Objet (No ou réf, pièce, PCB...)	PCB BadgePlace		
Version à modifier:	1		

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

1 Description ou justification

Corrections des erreurs de connexions de certains pins.

2 Référence conception

Projet Altium «2312_BadgePlaceTravail.PrjPcb» disponible sous :

*\2312_BadgePlaceTravail\hard\2312_BadgePlace

3 Détail des modifications

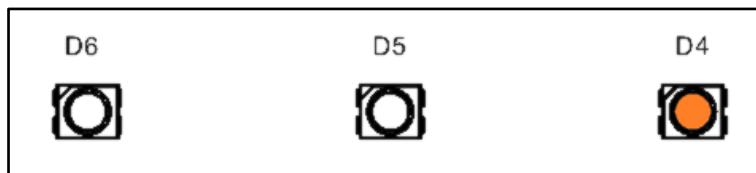
#	Description	Fait	Approuvé
1	Intervertir les pins ESP-TX et ESP-RX.	NOK	
2	Supprimer la connexion du driver de led avec l'OC1 du microcontrôleur. Le connecter au MOSI du SPI2 (SDO2, pin 6 MCU)	NOK	

4 Remarques

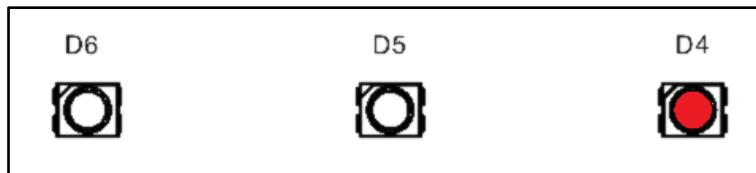
D'autres alternatives peuvent être envisagé pour connecter le driver de LEDs. L'utilisation d'un port permettant une communication série est préférée.

2312

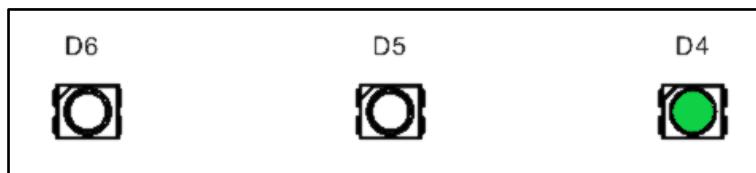
Mode d'emploi



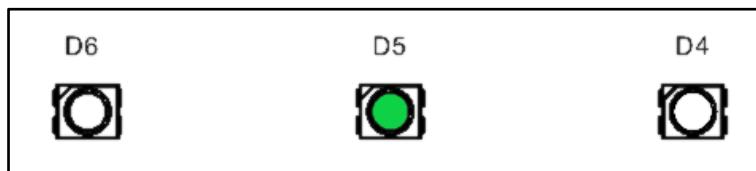
Connexion / Communication Wi-Fi en cours



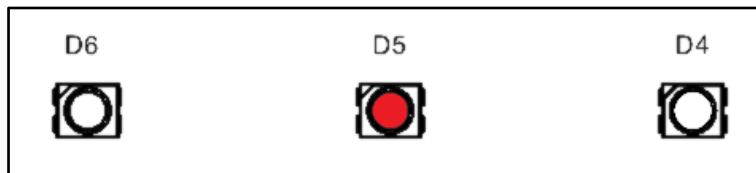
Erreur réseau / Wi-Fi introuvable



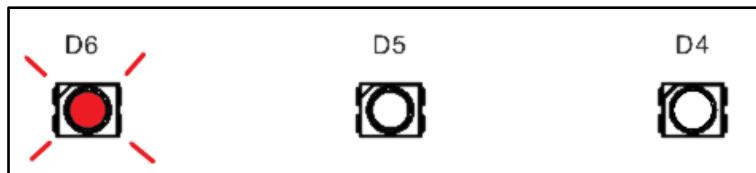
Connexion au Wi-Fi réussi



Badge RFID accepté



Badge RFID refusé



Avertissement de timeout