

Rainette

Travail de Bachelor

Département TIN
Filière Génie Électrique
Orientation Électronique embarquée et Mécatronique

Damien Pomelli

25 juillet 2025

Supervisé par :
Prof. P. Favrat (HEIG-VD)

Préambule

Ce travail de Bachelor (ci-après **TB**) est réalisé en fin de cursus d'études, en vue de l'obtention du titre de Bachelor of Science HES-SO en Ingénierie.

En tant que travail académique, son contenu, sans préjuger de sa valeur, n'engage ni la responsabilité de l'auteur, ni celles du jury du travail de Bachelor et de l'École.

Toute utilisation, même partielle, de ce TB doit être faite dans le respect du droit d'auteur.

HEIG-VD
Le Chef du Département

Yverdon-les-Bains, le 25 juillet 2025

Authentification

Je soussigné, Damien Pomelli, atteste par la présente avoir réalisé seul ce travail et n'avoir utilisé aucune autre source que celles expressément mentionnées.

Damien Pomelli



Yverdon-les-Bains, le 25 juillet 2025

Résumé

Ce rapport présente l'ensemble du travail réalisé durant le TB *Rainette*. Le projet consiste à concevoir une carte électronique capable de collecter des mesures et de commander des actionneurs pour réguler une serre aquaponique. Il s'agit d'un système embarqué communiquant sans fil avec une unité centrale et pouvant fonctionner sur batterie. Le travail englobe les phases d'analyse préliminaire, de conception matérielle et logicielle, ainsi que la mise en service.

**

This report presents all the work carried out as part of the TB *Rainette*. The project involves designing an electronic board capable of collecting measurements and controlling actuators to regulate an aquaponic greenhouse. It is an embedded system that communicates wirelessly with a central unit and can operate on battery power. The work includes the phases of preliminary analysis, hardware and software design, as well as commissioning.

Glossaire et abréviations

- ADC** Convertisseur analogique-numérique (Analog-to-Digital Converter), utilisé pour transformer un signal analogique en signal numérique.
- Altium** Logiciel de conception de circuits imprimés (PCB) utilisé pour le design électronique professionnel.
- API** Interface de programmation applicative (Application Programming Interface), ensemble normalisé de fonctions, classes ou protocoles permettant à des logiciels d'interagir entre eux ou avec des composants matériels.
- ARM** Architecture de processeur (Advanced RISC Machines) largement utilisée pour sa faible consommation et ses hautes performances.
- BLE** Bluetooth Low Energy, technologie de communication sans fil à faible consommation d'énergie.
- DC** Courant continu (Direct Current), type de courant électrique dont la polarité reste constante dans le temps.
- DRC** Design Rule Check, processus automatisé de vérification dans les logiciels de conception de circuits imprimés (PCB) permettant de s'assurer que la topologie du circuit respecte les règles de fabrication et de conception.
- ESP32** Microcontrôleur à faible consommation d'énergie avec Wi-Fi et Bluetooth intégrés, développé par Espressif Systems.
- Filtre RC** Circuit composé d'une résistance (R) et d'un condensateur (C), utilisé pour filtrer certaines fréquences d'un signal.
- Firmware** Logiciel embarqué stocké dans une mémoire non volatile, assurant le contrôle bas niveau d'un matériel électronique et souvent mis à jour pour corriger des bugs ou ajouter des fonctionnalités.
- Gerber** Format de fichier standard utilisé pour la fabrication des circuits imprimés (PCB), contenant les données graphiques décrivant les différentes couches du circuit (pistes, plans de masse, sérigraphie, perçages, etc.).
- GND** Masse électrique (Ground), point de référence de tension dans un circuit, souvent connecté au zéro volt.
- GPIO** Broche d'entrée/sortie programmable (General Purpose Input/Output), utilisée pour la communication avec des périphériques externes.
- Handle** Référence opaque utilisée pour accéder à une ressource ou un objet géré par une bibliothèque ou un système.
- HCI** Interface de communication entre le contrôleur Bluetooth et l'hôte, permettant la gestion de la pile Bluetooth et des applications associées.
- HEIG-VD** Haute École d'Ingénierie et de Gestion du canton de Vaud.
- IC** Circuit intégré (Integrated Circuit), composant électronique miniaturisé regroupant plusieurs fonctions sur une seule puce de semi-conducteur.

- IDE** Environnement de développement intégré (Integrated Development Environment), application regroupant outils de programmation tels qu'éditeur de code, compilateur, débogueur et gestionnaire de projet dans une interface unifiée.
- IoT** Internet des objets (Internet of Things), réseau d'objets connectés communiquant entre eux via Internet.
- ISR** Routine de service d'interruption (Interrupt Service Routine), fonction exécutée automatiquement en réponse à une interruption matérielle.
- I2C** Bus série (Inter-Integrated Circuit) à deux fils, utilisé pour la communication entre circuits intégrés.
- JTAG** Interface normalisée (Joint Test Action Group) utilisée pour le test, le débogage et la programmation des circuits intégrés.
- Layout** Disposition physique des composants et pistes sur un circuit imprimé (PCB).
- LDO** Régulateur de tension à faible dropout (Low Dropout Regulator), composant permettant de maintenir une tension de sortie stable avec une faible différence entre l'entrée et la sortie.
- Linux** Système d'exploitation libre et open source, largement utilisé sur serveurs, ordinateurs et systèmes embarqués.
- MBed OS** Système d'exploitation temps réel open source pour microcontrôleurs ARM, développé par Arm Ltd, conçu pour les applications IoT embarquées.
- MCU** Microcontrôleur (Microcontroller Unit), circuit intégré regroupant un processeur, de la mémoire et des périphériques sur une seule puce.
- Mémoire Flash** Mémoire non volatile permettant de stocker des données même sans alimentation.
- MOSFET** Transistor à effet de champ (Metal-Oxide-Semiconductor Field-Effect Transistor), utilisé pour la commutation et l'amplification des signaux électroniques.
- OS** Système d'exploitation (Operating System), logiciel gérant les ressources matérielles et l'exécution des programmes.
- PCB** Circuit imprimé (Printed Circuit Board), support physique qui connecte électriquement les composants électroniques via des pistes conductrices.
- PPK2** Power Profiler Kit 2, outil de mesure de consommation électrique produit par Nordic Semiconductor.
- Protections ESD** Dispositifs protégeant les circuits contre les décharges électrostatiques (ElectroStatic Discharge), afin d'éviter les dommages aux composants sensibles.
- Raspberry Pi** Ordinateur monocarte compact et abordable, populaires pour les projets électroniques et informatiques.
- Ressources CAD** Fichiers ou bibliothèques utilisés dans la conception assistée par ordinateur (Computer-Aided Design), comme les empreintes de composants ou les schémas électroniques.
- SPI** Bus série synchrone (Serial Peripheral Interface) utilisé pour la communication rapide entre microcontrôleurs et périphériques.
- ST-Link** Outil matériel et interface de programmation/débogage développé par STMicroelectronics, utilisé pour programmer et déboguer les microcontrôleurs STM32 via JTAG ou SWD.
- STM32** Famille de microcontrôleurs 32 bits basée sur l'architecture ARM Cortex, développée par STMicroelectronics, utilisée dans les systèmes embarqués.

- SWD** Serial Wire Debug, protocole de débogage série développé par ARM, utilisant deux fils pour accéder aux fonctions de débogage internes des microcontrôleurs, en alternative au JTAG.
- Timeout** Délai d'attente au-delà duquel une opération ou une communication est automatiquement interrompue faute de réponse.
- Toolchain** Chaîne d'outils de compilation, regroupant les programmes nécessaires à la construction d'un logiciel, tels que le compilateur, l'éditeur de liens, l'assembleur et les utilitaires de génération de fichiers binaires.
- USART** Interface de communication série (Universal Synchronous/Asynchronous Receiver Transmitter), utilisée pour transmettre et recevoir des données en mode synchrone ou asynchrone.
- USB** Bus universel en série (Universal Serial Bus), norme de communication pour connecter des périphériques à un ordinateur.
- VCC** Tension d'alimentation positive d'un circuit, souvent utilisée pour désigner l'alimentation principale.
- 3V3** Tension d'alimentation de 3,3 volts, couramment utilisée dans les circuits électroniques modernes.

Table des matières

Préambule	i
Authentification	iii
Résumé	v
Glossaire et abréviations	vii
1 Introduction	1
1.1 Contexte	1
1.2 Problématique	2
1.3 Objectifs	3
1.4 Méthodologie	4
2 Analyse préliminaire	5
2.1 Mesure des grandeurs physiques	5
2.2 Produits existants	6
2.3 Technologies et méthodes existantes	9
2.3.1 Fonctionnement global du système	9
2.3.2 Communication sans fil	9
2.4 Nouvelle version du cahier des charges	11
3 Conception Hardware	13
3.1 Connexions externes	13
3.2 Analyse fonctionnelle	14
3.3 Analyse matérielle	15
3.3.1 Mesure du niveau de charge de la pile	16
3.3.2 Régulateurs de tension	16
3.3.3 Module STM32	17
3.3.4 Sondes analogiques	19
3.4 Schéma électrique	20
3.4.1 Ressources composants	20
3.4.2 Décomposition du schéma	21
3.4.3 Choix généraux	21
3.4.4 Power Supply	22
3.4.5 Analog - Sensors	26
3.4.6 Digital - GPIO	28
3.4.7 Communications	29
3.4.8 STM32	34

TABLE DES MATIÈRES

3.4.9 Estimation de la consommation	34
3.5 Conception du PCB	36
3.5.1 Premières considérations	36
3.5.2 Placement des composants	38
3.5.3 Routage et plans de masse	42
3.5.4 Vérification et exportation des fichiers Gerber	44
3.5.5 Commande et coûts totaux	46
4 Développement logiciel	49
4.1 Système d'exploitation	49
4.2 Fonctionnement global du programme	52
4.2.1 Organigramme	52
4.2.2 Bluetooth Low Energy	54
4.3 Paramétrage	55
4.3.1 Paramètres du projet Mbed OS	55
4.3.2 Paramètres du BLE	56
4.4 Développement des fonctionnalités	58
4.4.1 Carte Disco	59
4.4.2 Carte Rainette	65
4.5 Programmation du Raspberry Pi	69
4.5.1 Carte utilisée	69
4.5.2 Informations générales sur le code	69
4.5.3 Spécificités du programme	71
4.5.4 Interface utilisateur	72
5 Mise en service et tests	73
5.1 Contrôle qualité	73
5.1.1 Premier contrôle visuel	73
5.1.2 Deuxième contrôle visuel	74
5.1.3 Contrôle des sources d'alimentation	74
5.2 Tests fonctionnels de la carte	77
5.2.1 Connecteur ST-Link	77
5.2.2 Programmation de la carte	77
5.2.3 Validation des fonctionnalités	78
5.3 Mesure de consommation	82
5.4 Revue des objectifs	85
5.5 Perspectives	85
6 Conclusion	87
Appendices	95
A Général	95
A.1 Codes source	95
A.2 Schéma électrique <i>Rainette</i>	95
A.3 Devis PCBWay	95
A.4 Planification	95
A.5 Schéma électrique <i>STM32WB5MM-DK</i>	95

Table des figures

1.1	Principe de fonctionnement de l'aquaponie	2
2.1	Réalisation hardware du projet ponics32	6
2.2	Réalisation hardware du projet PiPonics	7
2.3	Kit pour aquaponie de Atlas Scientific	8
2.4	Exemple de l'interface utilisateur Home Assistant	8
3.1	Schéma pour analyse connexions externes	14
3.2	Schéma pour analyse fonctionnelle	15
3.3	Header 3x2	16
3.4	Schéma pour analyse matérielle	19
3.5	TVS bidirectionnelle et unidirectionnelle	21
3.6	Configuration des MOSFET pour mesure de la tension de pile	23
3.7	Simulation V_{ADC} en fonction de V_{batt}	24
3.8	Application typique du TPS63001 selon son datasheet	26
3.9	Application typique du SHT41-AD1B-R2 selon son datasheet	27
3.10	Application typique du VEML6030 selon son datasheet	28
3.11	Accès au sélecteur de MCU	31
3.12	MCU sélectionné	31
3.13	Paramètres minimum sur l'outil de devis de PCBWay	37
3.14	Table de classification Eurocircuits	38
3.15	Positionnement du module STM32 sur la carte	39
3.16	Disposition des composants sur la carte	41
3.17	Configuration du <i>Via stitching</i> sur le plan de masse	42
3.18	Couche <i>Top</i> de la carte sans plan de masse	43
3.19	Couche <i>Bottom</i> de la carte	43
3.20	Vue 3D de la carte	44
3.21	Modification du <i>Solder Mask</i> du footprint	45
3.22	Aperçu du <i>Pour_Assemblage.OutJob</i>	46
4.1	Organigramme du programme sur la carte Rainette	53
4.2	Tableaux des différentes versions de FW du coprocesseur M0	57
4.3	Tableaux de spécifications des versions FW	57
4.4	FUS sur STM32CubeProgrammer	58
4.5	Diagramme du programme de la carte Disco	64
4.6	Organigramme du programme de la carte Disco	65
4.7	Diagramme du programme de la carte Rainette	68
4.8	Raspberry Pi Compute Module 4 IO Board	69
4.9	Organigramme des codes Raspberry Pi	70

TABLE DES FIGURES

4.10	GUI Disco	72
4.11	GUI Rainette	72
5.1	Schéma de mesure - Pile	74
5.2	Schéma de mesure - USB	75
5.3	Schéma de mesure - High DC	76
5.4	Schéma de mesure - 3V3	76
5.5	Adaptateur TC2050-ARM2010 ARM 20-pin to TC2050	77
5.6	Régression polynomiale pour correction de mesure	79
5.7	Défaut de communication I2C avec le VEML6030	80
5.8	Communication I2C correcte avec le VEML6030	81
5.9	Power Profiler Kit II	82
5.10	Schéma de mesure - consommation de courant	83
5.11	Consommation sur pile	84

Liste des tableaux

3.1	Tableau 6 pour pinout du ST-LINK-V3SET	30
3.2	Estimation de la consommation sur pile	35
3.3	Estimation de la consommation filaire	36
3.4	Devis <i>PCBWay</i>	46
4.1	Services et caractéristiques BLE exposés par la carte	61
4.2	Services et caractéristiques BLE exposés par la carte	67
5.1	Résultats du premier contrôle visuel	73
5.2	Résultats du deuxième contrôle visuel	74
5.3	Mesure tension pile	74
5.4	Mesure tension USB	75
5.5	Mesure tension High DC	76
5.6	Mesure tension 3V3	76
5.7	Résumé des fonctionnalités	82
5.8	Revue des objectifs	85

Liste des codes sources

4.1	Exemple de thread avec deux leds clignotant à des fréquences différentes	50
4.2	Exemple de eventQueue avec appel différé depuis une ISR	51
4.3	Exemple de eventFlags avec synchronisation entre deux threads	52
4.4	Fichier mbed_app.json	56
4.5	Log d'erreur sur code d'exemple BLE	57
4.6	Modification de la taille ROM pour la carte Disco	58
4.7	Fonctions pour capteur STTS22H	60
4.8	Classe GeneralService de la carte Disco	62
4.9	Classe GattServerEventHandler	63
4.10	Classe DiscoApp	63
4.11	Fichier custom_targets.json	66
4.12	Modification de la taille ROM pour la carte Rainette	66
4.13	Allocation dynamique de l'I2C pour les mesures	67
4.14	Conversion des valeurs analogiques	71

Chapitre 1

Introduction

L'aquaponie est une méthode de culture qui combine l'aquaculture (élevage de poissons) et l'hydroponie (culture de plantes dans l'eau). Cette technique permet de créer un écosystème durable où les déchets des poissons fournissent des nutriments aux plantes, tandis que les plantes filtrent l'eau pour les poissons. Bien que récente en Europe, cette technique remonte en vérité aux Aztèques qui cultivaient déjà des plantes sur des radeaux flottants dans les lacs du Mexique. Le développement de cette méthode de culture permet de répondre à plusieurs enjeux contemporains, tels que la sécurité alimentaire ou l'empreinte écologique des aliments, car ses avantages sont nombreux.

La mise en place d'un tel système nécessite néanmoins de bonnes connaissances car il est question de créer un écosystème fermé capable de s'auto-réguler. À ce titre, la présence d'un système de contrôle est un réel avantage car il permet d'anticiper les écarts de fonctionnement et de réguler certaines grandeurs physiques comme la température de l'environnement. Si d'imposants systèmes de contrôle existent déjà pour les grandes serres industrialisées, les possibilités sont bien réduites pour les particuliers ou les maraîchers possédant une petite infrastructure.

Ce travail de Bachelor a pour but de concevoir de petits modules électroniques capables de remplir cette mission tout en restant accessibles au grand public. Conçus pour être déployés en tout point du système aquaponique, ils permettraient d'interagir avec des capteurs et des actionneurs tout en communiquant sans fil avec une unité centrale qui orchestrerait l'ensemble. Les données pourraient ensuite être facilement consultées depuis un moniteur ou un smartphone.

1.1 *Contexte*

Ce projet s'inscrit dans le cadre du Bachelor en Électronique Embarquée et Mécatronique de la HEIG-VD (Haute École d'Ingénierie et de Gestion du canton de Vaud). Proposé par l'étudiant, il repose sur un cahier des charges conçu pour mobiliser un large éventail de compétences acquises durant la formation. Il implique la conception d'un système électronique embarqué, incluant donc la réalisation du schéma électrique et du PCB, la programmation sur microcontrôleur mais également l'établissement d'un cahier des charges, la planification du développement, ainsi que la rédaction d'un rapport technique structuré.

Ce projet constitue ainsi une mise en pratique concrète des connaissances théoriques et pratiques acquises au cours de la formation. Il vise à reproduire les conditions d'un développement industriel, en intégrant des contraintes de fiabilité, d'optimisation énergétique, de communication sans fil et de modularité.

1.2 Problématique

Comme mentionné précédemment, un système aquaponique repose sur l'interaction entre trois éléments fondamentaux : les poissons, les plantes et les bactéries. Les poissons produisent des déchets riches en ammoniaque, qui sont transformés par les bactéries en nutriments assimilables par les plantes. Ces dernières, en absorbant ces nutriments, contribuent à purifier l'eau qui retourne ensuite aux poissons, créant ainsi une boucle fermée.

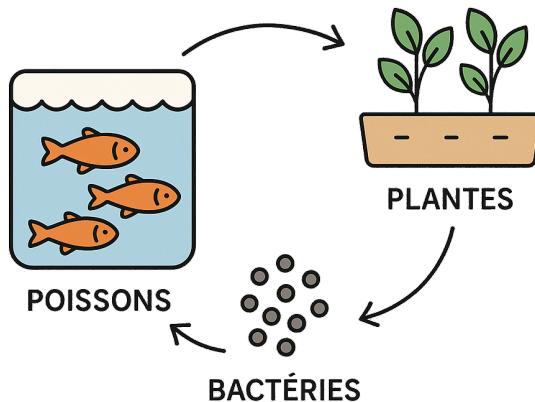


Figure 1.1 – Principe de fonctionnement de l'aquaponie

L'aquaponie offre de nombreux avantages. Elle optimise l'utilisation de l'eau, avec une consommation réduite jusqu'à 90 % par rapport à l'agriculture traditionnelle. En combinant l'élevage de poissons et la culture de plantes, elle permet d'obtenir à la fois des protéines animales et des végétaux à partir d'un apport unique. Cultivées hors sol, les plantes peuvent être disposées en bacs superposés, ce qui permet l'installation de fermes verticales dans des zones densément peuplées. Le système peut être implanté aussi bien sur un toit que dans un sous-sol, à condition d'y assurer un éclairage artificiel adapté.

Cependant, un système aquaponique constitue un environnement complexe faisant intervenir de nombreux paramètres physiques, chimiques et biologiques, dont l'équilibre est souvent délicat à maintenir. La température de l'eau, le pH, la concentration en oxygène dissous, la qualité des nutriments ou encore la densité de population des poissons sont autant de facteurs qui doivent être surveillés et régulés en permanence. La moindre variation de l'un de ces paramètres peut entraîner des déséquilibres

1.3. OBJECTIFS

susceptibles de compromettre la santé des organismes vivants et la productivité du système. C'est pourquoi la gestion et le contrôle précis de ces différentes variables sont essentiels pour garantir la pérennité et l'efficacité d'une installation aquaponique.

Il existe aujourd'hui des systèmes automatisés de gestion pour les serres agricoles et aquaponiques, intégrant des capteurs, des contrôleurs et des interfaces logicielles permettant de réguler les différents paramètres de l'écosystème. Toutefois, ces solutions sont principalement conçues pour des structures professionnelles ou industrielles, avec des coûts et une complexité adaptés à de grandes exploitations. Pour les particuliers ou les petits producteurs, ces outils restent souvent inaccessibles, tant financièrement que techniquement. De petits projets open source existent sur Internet mais ils sont souvent incomplets et nécessitent des compétences techniques pour les adapter à sa propre installation.

La solution imaginée pour ce travail de Bachelor répond à cette problématique. Elle consiste en de petits modules électroniques disposables dans le système aquaponique. Capables de réaliser des mesures ou d'activer les éléments d'une serre, comme des ventilateurs ou une pompe hydraulique, ils peuvent prendre en charge la régulation de la serre pour alléger les tâches humaines. En proposant un fonctionnement sur pile, il est possible de réaliser des mesures là où aucun câble n'est présent pour alimenter le système. Les modules communiquent avec un système sans fil, ce qui permet de couvrir plus facilement de petites distances, évitant ainsi de devoir acheminer des câbles à travers toute la serre. Une unité centrale permet finalement d'orchestrer le tout, en lançant des commandes et en récupérant les mesures, ce qui centralise la gestion.

1.3 Objectifs

Une première version du cahier des charges a été établie avant le début du projet. Elle établissait les grandes lignes du travail ainsi que les principales étapes et objectifs imaginés. Cette version a par la suite été retravaillée après avoir étudié le sujet pour proposer une version plus complète et détaillée. Cette partie est décrite dans l'*Analyse Préliminaire*. Les objectifs listés ci-dessous sont repris de la version 0 rendue le 27 mars 2025 qui a été acceptée par le professeur encadrant. Ils décrivent les points aisément vérifiables qui doivent être atteints à la fin du travail.

- Mener le projet en documentant l'avancement. Identifier et expliquer les raisons et solutions à apporter en cas d'incapacité à remplir un objectif fixé.
- Concevoir une carte électronique sous forme de prototype selon les critères suivants.
 1. Doit pouvoir fonctionner sur pile et/ou par alimentation filaire.
 2. Doit pouvoir collecter des données provenant d'un capteur.
 3. Doit pouvoir piloter un élément (ex : GPIO, relais, LED).
 4. Doit pouvoir lire une tension analogique.

- Programmer la carte électronique selon les critères suivants.
 - 5. Doit pouvoir communiquer sans fil.
 - 6. Doit pouvoir transmettre une donnée.
 - 7. Doit pouvoir recevoir une donnée.
- Programmer l'unité centrale selon les critères suivants.
 - 8. Doit pouvoir communiquer sans fil.
 - 9. Doit pouvoir transmettre une donnée.
 - 10. Doit pouvoir recevoir une donnée.

1.4 Méthodologie

Le projet a débuté par une phase d'analyse du cahier des charges initial, afin d'étudier la problématique à résoudre. Une version retravaillée du cahier des charges, comprenant des objectifs redéfinis et réalistes, a été rendue une fois cette première étape accomplie. Le travail a été planifié sur l'ensemble du temps imparti, les différentes étapes étant clairement découpées et ordonnées sur un diagramme de Gantt.

L'ensemble des réflexions, étapes et incidents ont été consignées dans un journal de bord, permettant de retracer le déroulement du travail au fil des semaines. Une communication fluide a été mise en place entre l'étudiant et le professeur encadrant, afin de valider des choix techniques et d'obtenir une expertise avisée sur des domaines clés. Plusieurs ingénieurs de la HEIG-VD ont également été consultés pour obtenir des éclairages sur des points précis. Les sources Internet utilisées pour ce projet sont renseignées dans la Bibliographie du rapport. Enfin, l'ensemble des ressources de ce projet peuvent être consultées sur le dépôt GitHub suivant : <https://github.com/dpmlli/Rainette>.

Chapitre 2

Analyse préliminaire

2.1 Mesure des grandeurs physiques

L'aquaponie repose sur la création d'un écosystème fonctionnant en circuit d'eau fermé. Plusieurs grandeurs physiques mesurables interviennent dans la régulation de cet équilibre. Bien qu'il ne soit pas nécessaire de toutes les surveiller en permanence, le suivi des paramètres les plus significatifs permet généralement d'évaluer l'état de santé du système. La combinaison de différentes mesures et l'utilisation de dispositifs correctifs facilitent le maintien de l'équilibre et permettent de limiter les baisses de production. Les principales grandeurs physiques à prendre en compte sont présentées ci-dessous.

Température de l'eau

Influence le métabolisme des poissons et la croissance des bactéries nitrifiantes.

pH de l'eau

Doit être stable pour éviter le stress des poissons et optimiser la nitrification.

Dissolved Oxygen (Oxygène dissous - DO)

Essentiel pour la respiration des poissons et des bactéries.

Conductivité électrique (EC) & Nutriments

Indique la concentration en sels minéraux et nutriments pour les plantes.

Niveau d'ammoniaque, nitrites et nitrates

L'ammoniaque et les nitrites doivent être quasi nuls (toxiques). Les nitrates doivent être présents pour nourrir les plantes.

Température et humidité de l'air

Influence l'évaporation de l'eau et la croissance des plantes.

Lumière (intensité et durée)

Essentielle pour la photosynthèse des plantes.

Débit d'eau / Niveau d'eau

Pour éviter les débordements et assurer la bonne circulation de l'eau.

L'enjeu est de permettre à l'utilisateur de connecter le plus grand nombre de sondes et capteurs afin qu'il puisse se concentrer sur les paramètres qu'il juge pertinents.

2.2 *Produits existants*

Il existe déjà des solutions, plus ou moins complexes, dédiées à la gestion des serres aquaponiques ou agricoles. Elles peuvent être regroupées en deux grandes catégories : les systèmes industriels et les systèmes grand public. Les premiers sont conçus pour répondre aux besoins des exploitations de très grandes envergures, ils ne seront pas traités ici car le projet se focalise sur de petites installations.

La seconde catégorie concerne donc les produits destinés aux particuliers ou aux petites et moyennes exploitations. Certains produits sont des solutions relativement complètes, proposées par des fabricants, d'autres sont de petits projets open source développés et partagés par des passionnés. Ces produits traitent principalement de la gestion et de la régulation de la serre, les différents éléments techniques essentiels d'un système aquaponique tel que la pompe principale ou le filtre mécanique ne sont donc pas évoqués. Quatre produits trouvés sur Internet sont listés et analysés ci-dessous.

ponics32

[ponics32] Projet open source pour la gestion de systèmes aquaponiques et hydroponiques basé sur un microcontrôleur ESP32. Il propose une petite interface web et permet de communiquer avec des capteurs et actionneurs à l'aide de l'ADC et des GPIO du MCU. Il possède une fonctionnalité capable d'émettre des alertes en cas d'urgence.

Limitations : Le projet comporte peu d'éléments hardware et est principalement axé sur l'interface web permettant de relever les valeurs récupérées sur les capteurs. Il s'agit d'une solution trop peu aboutie pour un usage réel et manque de fonctionnalités et de matériel adapté.

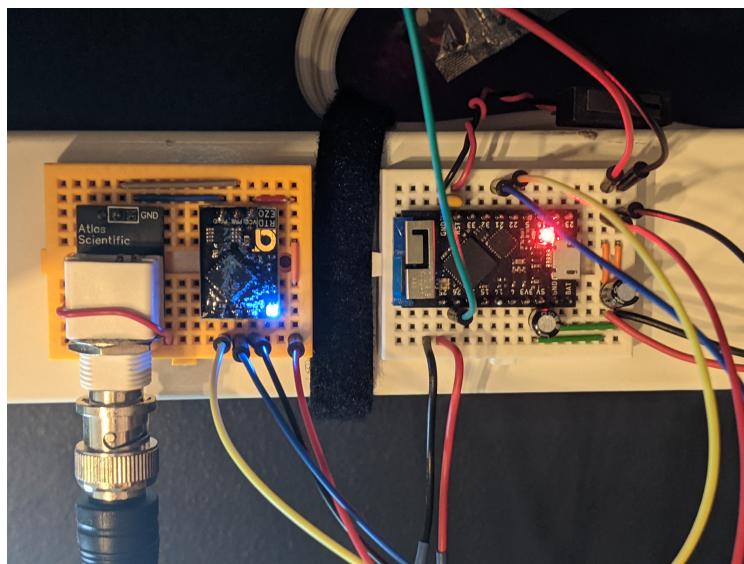


Figure 2.1 – Réalisation hardware du projet ponics32

2.2. PRODUITS EXISTANTS

PiPonics

[PiPonics] PiPonics est un projet open source de supervision et de contrôle pour systèmes aquaponiques et hydroponiques basé sur un Raspberry Pi 4B et un microcontrôleur STM32L432KC. Il offre une interface web via le Raspberry Pi et permet de communiquer avec de petits capteurs à l'aide de l'ADC et des GPIO du MCU. Les deux éléments sont en communication grâce à une interface série (UART, USB ou I2C). Le projet est inspiré de ponics32 et semble développé pour de très petites installations.

Limitations : Tout comme le ponics32, ce projet semble davantage focalisé sur l'interface web. Il n'y a pas de connecteurs prévus et le STM32 doit se trouver à proximité du Raspberry Pi à cause de l'interface filaire. Il ne semble pas suffisamment complet en l'état pour être porté sur une application réelle.

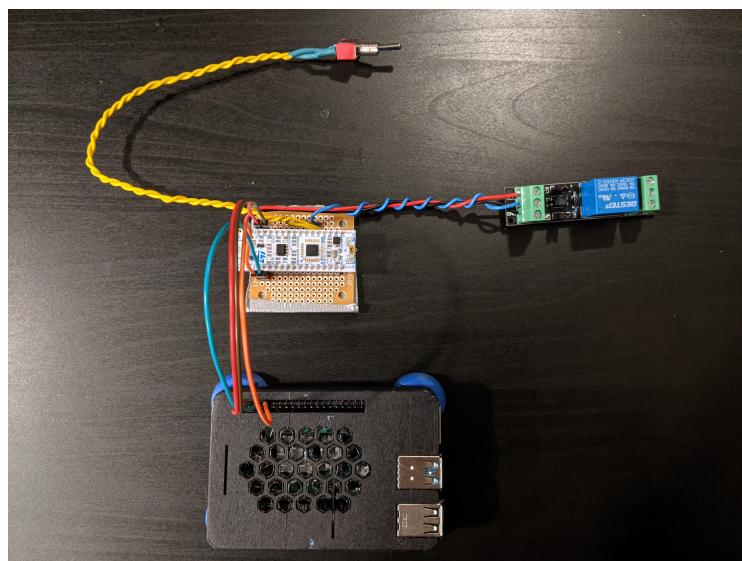


Figure 2.2 – Réalisation hardware du projet PiPonics

Atlas Scientific

[ATLAS] Atlas Scientific est une entreprise américaine spécialisée dans la conception de capteurs pour la mesure des paramètres physiques et chimiques des liquides. Leurs produits sont conçus pour être facilement intégrés à des systèmes embarqués, notamment via des microcontrôleurs comme Arduino ou Raspberry Pi. Atlas Scientific propose également des kits complets qui permettent une surveillance à distance des paramètres de l'eau avec des données automatiquement collectées.

Limitations : Leurs produits et en particulier leurs kits sont proposés à des prix très élevés (supérieurs à 1000 \$ par module). Cela peut freiner les particuliers qui souhaiteraient équiper leur serre à des coûts raisonnables ainsi que les petits maraîchers qui auraient besoin de placer plusieurs dispositifs à des endroits différents dans leur installation. De plus, les kits ne semblent pas être prévus pour une utilisation sur pile, ce qui nécessiterait de devoir tirer des câbles d'alimentation là où l'on souhaiterait effectuer une mesure. Il faut toutefois noter qu'ils disposent d'un grand nombre de sondes ainsi que d'interfaces permettant de lire les valeurs reçues. Cela peut s'avérer intéressant pour la suite car toutes leurs interfaces s'emploient de la même manière.

CHAPITRE 2. ANALYSE PRÉLIMINAIRE



Figure 2.3 – Kit pour aquaponie de Atlas Scientific

Novaspex Inc.

[NOVASPEX] Basée au Canada, Novaspex Inc. propose des solutions d'automatisation pour la gestion des environnements contrôlés, notamment dans les secteurs de l'agriculture et de l'aquaponie. Ils travaillent avec Home Assistant, une plateforme open source de domotique qui permet d'automatiser et visualiser en temps réel les données issues de capteurs et d'appareils connectés.

Limitations : Il semble s'agir d'un grand groupe travaillant sur de grandes infrastructures, cela ne convient donc pas aux particuliers et aux petits producteurs. Leur utilisation de Home Assistant peut néanmoins constituer une source d'inspiration pour ce projet.

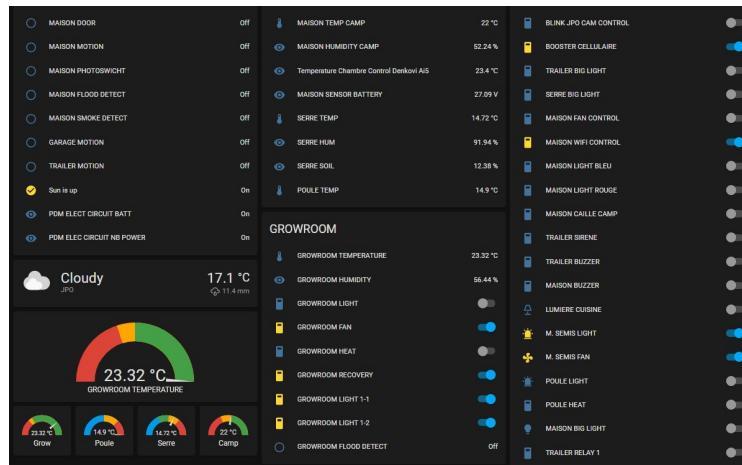


Figure 2.4 – Exemple de l'interface utilisateur Home Assistant

2.3. TECHNOLOGIES ET MÉTHODES EXISTANTES

2.3 Technologies et méthodes existantes

Après avoir exploré les grandeurs physiques et les produits déjà disponibles sur le marché, on obtient une vision d'ensemble plus claire de la problématique. Il devient dès lors possible de se focaliser sur les technologies utilisables ainsi que la façon de répondre aux différents objectifs du projet.

2.3.1 Fonctionnement global du système

La version initiale du cahier des charges décrit une première idée du système dans son ensemble. Plusieurs cartes électroniques sont déployées dans la serre aquaponique afin de relever des mesures ou d'actionner des éléments capables de réguler l'écosystème. Toutes ces cartes sont orchestrées par une unité centrale qui envoie des consignes et récupère des valeurs pour les traiter. Un Raspberry Pi ferait parfaitement l'affaire pour ce rôle d'unité centrale. Il s'agit d'un petit ordinateur, tournant avec Linux, et qui se prête très bien à ce genre d'application. Plus puissant qu'un microcontrôleur, il parvient facilement à communiquer via Internet et à utiliser une interface utilisateur graphique avec un petit écran.

Pour permettre à l'utilisateur de relever des mesures en tout point de la serre sans devoir tirer des câbles pour l'alimentation et la communication, il est prévu de travailler sur la consommation d'énergie des modules à développer afin de permettre un fonctionnement sur pile. Un grand nombre de systèmes embarqués utilisent aujourd'hui des batteries rechargeables. Notre cas se rapproche davantage d'un système IoT où la durée de vie de la batterie s'étend sur plusieurs semaines ou mois. Il n'est donc pas prévu de pouvoir recharger quotidiennement la batterie, ce qui oriente le choix vers de simples piles.

Il faut également prévoir la possibilité d'alimenter les modules par câble lorsqu'ils se trouvent à proximité. En cas de fonctionnement sur batterie, le microcontrôleur ne devrait être actif qu'une faible partie du temps afin de ne pas consommer trop d'énergie de la pile. Deux cas de fonctionnement apparaissent donc :

Alimentation sur pile

Consommation limitée, pas de câbles d'alimentation à proximité, donc pas d'actionneurs à piloter. Le module ne sert ainsi qu'à relever des valeurs sur les sondes et à les envoyer à l'unité centrale avant de repasser en mode veille.

Alimentation filaire

La présence de câble d'alimentation peut signifier celle d'actionneurs. Le sommeil n'est plus nécessaire, le système doit pouvoir recevoir en permanence des commandes de l'unité centrale pour piloter des actionneurs. La lecture des sondes reste disponible.

2.3.2 Communication sans fil

La communication entre les différentes cartes constitue un élément central du système. Les cartes pouvant être espacées de plusieurs mètres, il convient d'opter pour une communication sans fil. Il existe beaucoup de protocoles envisageables, ils présentent chacun des avantages et des inconvénients. Ceux étant les plus susceptibles de convenir sont listés si dessous.

CHAPITRE 2. ANALYSE PRÉLIMINAIRE

Bluetooth Low Energy (BLE)

Protocole sans fil à courte portée optimisé pour la faible consommation d'énergie.
Idéal pour les objets connectés personnels (wearables, capteurs).

Limitations : Portée limitée (environ 10-30 mètres) et faible débit de données comparé au Wi-Fi.

Wi-Fi

Protocole de communication sans fil à haut débit utilisé pour les réseaux locaux.
Permet la transmission rapide de grandes quantités de données.

Limitations : Consommation énergétique élevée et portée réduite en environnement encombré.

Thread

Protocole maillé basé sur IPv6 conçu pour la domotique, assurant une communication fiable entre appareils. Offre une bonne interopérabilité avec Matter.

Limitations : Nécessite un routeur de bordure (border router) pour la connexion à Internet et n'est pas encore largement adopté.

Zigbee

Protocole maillé à faible consommation utilisé pour les applications domotiques.
Permet la communication entre de nombreux noeuds avec une faible latence.

Limitations : Sensible aux interférences sur la bande 2,4 GHz et nécessite une passerelle pour l'accès à Internet.

LoRaWAN

Protocole longue portée et basse consommation adapté aux réseaux étendus (LPWAN) pour la collecte de données sur de grandes distances.

Limitations : Débit de données très faible et forte latence, peu adapté aux applications en temps réel.

Au vu des conditions et des critères imposés par le cahier des charges, le Bluetooth Low Energy semble être le mieux adapté pour une communication entre les cartes et l'unité centrale. Il consomme en effet assez peu d'énergie, son débit moyen (100-300 kbps) convient à ce type d'application où le volume d'informations est faible et il est très répandu dans l'industrie. Sa portée peut en revanche beaucoup dépendre de l'environnement dans lequel il se trouve et peut chuter à moins de 10 mètres dans de mauvaises conditions. Il existe en revanche une version maillée du Bluetooth (Bluetooth MESH) qui permet d'augmenter la portée en utilisant les cartes comme répéteurs pour créer un réseau de noeuds. Cette extension du BLE nécessite cependant certaines fonctionnalités logicielles spécifiques qui ne sont pas incluses par défaut dans toutes les implémentations BLE.

Si la portée est trop limitée, une autre alternative au BLE est le Zigbee qui possède nativement un système de maillage. Moins répandu que le Bluetooth, il est néanmoins basé sur la norme IEEE 802.15.4 et propose des communications fiables.

Voici donc les deux options à privilégier, ils vont guider le choix du microcontrôleur dans la partie *Conception Hardware*. Il serait envisageable de commencer par développer une interface BLE car plus rapide à mettre en place avec les nombreux exemples déjà existants, et de basculer sur Zigbee si les tests ne sont pas concluants.

2.4. NOUVELLE VERSION DU CAHIER DES CHARGES

2.4 Nouvelle version du cahier des charges

Il est demandé de rendre une version définitive du cahier des charges après les premières semaines de travail. Cette dernière peut comporter des différences avec la version initiale car elle doit permettre de fixer des objectifs réalistes avec les ressources et le temps impartis. Dans notre cas, peu de modifications ont été apportées car la version initiale avait été conçue de manière cohérente dès l'origine. Le cahier des charges définitif inclut des objectifs mesurables pour chaque phase du projet, cités précédemment dans la section *Objectifs*. La plupart de ces objectifs nécessitent que le projet soit mené à terme pour être validés (hardware et software fonctionnels).

Le projet doit respecter les règles fixées par la HEIG-VD pour les travaux de Bachelor, notamment en ce qui concerne :

- Le budget alloué.
- Le temps imparti.
- Les ressources disponibles.

Le budget n'est pas précisément fixé, néanmoins les dépenses doivent être justifiées au-delà de plusieurs centaines de francs. Le temps prévu pour la première partie du TB est fixé à 180 heures. Un rapport intermédiaire doit être retourné à ce moment avant d'entamer la deuxième phase. 240 heures sont ensuite prévues pour un travail à plein temps avant le rendu final du projet. Ainsi, ce sont 420 heures qui sont allouées au travail de Bachelor. Un planning sous forme de diagramme de Gantt a été réalisé pour organiser correctement les différentes étapes et tâches.

Il est possible d'utiliser l'ensemble du matériel mis à disposition par l'école, de commander des composants en ligne et de s'adresser aux différents professeurs et ingénieurs de l'école. Si d'autres ressources, externes à la HEIG-VD, devaient être utilisées, il incombe à l'étudiant de mettre en place l'organisation nécessaire pour se les procurer.

Les livrables attendus pour ce projet sont :

- **le rapport du projet** : version *intermédiaire* au milieu du projet et version *finale* en fin de projet.
- **la carte électronique** : prototype fonctionnel selon les critères fixés.
- **le code source** de la carte électronique : selon les critères fixés.
- **le code source** de l'unité centrale : selon les critères fixés.
- **la documentation** détaillant le fonctionnement (si non incluse dans le rapport).

Chapitre 3

Conception Hardware

3.1 Connexions externes

La conception de la carte électronique débute en représentant le système vu de l'extérieur, afin de mettre en avant les connexions qu'il possède avec son environnement. Cette première étape permet de déceler les passages par lesquels pourraient se présenter des décharges électrostatiques. En les identifiant dès le début de la conception, il est plus facile de mettre en place des mesures pour les contrer. Conformément à ce qui a été enseigné en cours de Projet Électronique, les éventuels boutons et leds ne sont pas renseignés à ce stade.

Les différentes connexions sont listées :

Power

Apport d'énergie au système. Le cahier des charges impose de prévoir une utilisation sur pile ainsi qu'une utilisation filaire. La pile étant considérée comme externe au système, cette connexion contient donc au moins deux entrées dans le système.

GPIO

Communications numériques avec des éléments externes au système. Cela inclut des entrées et sorties en 3,3 V, supportant de faibles courants.

ADC

La carte doit être capable de lire des valeurs analogiques, c'est par cette connexion que les signaux atteignent le convertisseur analogique-numérique du microcontrôleur (ADC).

ST-Link

Cette connexion est nécessaire pour programmer et déboguer le microcontrôleur. Le ST-Link est un outil de communication entre l'ordinateur et le microcontrôleur, permettant le transfert du code et l'analyse en temps réel du fonctionnement du programme.

Bus

Connexion entre les différents protocoles disponibles sur la carte (I2C, UART, etc.) et l'extérieur du système.

Relays

Permet de piloter des courants plus importants en agissant comme un interrupteur.

Le schéma correspondant est illustré ci-dessous.



Figure 3.1 – Schéma pour analyse connexions externes

3.2 Analyse fonctionnelle

L'étape suivante consiste à faire un pas supplémentaire dans l'analyse du système en réalisant une analyse fonctionnelle. On doit y retrouver les premiers découpages au sein de la carte mais sans faire référence aux composants. Seules les fonctions principales qui composent le système doivent être mises en évidence à ce niveau de la conception.

Le schéma de la rubrique précédente peut naturellement servir de base pour cette partie, les entrées et sorties identifiées entrant directement en contact avec les différentes fonctions du système.

Power Supply

Gestion de l'alimentation du système. Doit être en contact avec les différentes sources d'alimentation, sélectionner celle utilisée et fournir la ou les tension(s) du système à l'aide de régulateurs de tension. Doit aussi contenir un système permettant de mesurer la tension de la pile.

Data Processing

Traitement des données et gestion du système.

Analog - Sensors

Contient les connecteurs et l'éventuel prétraitement des différentes sondes analogiques ainsi que les capteurs directement présents sur la carte.

Communications

Contient les composants nécessaires aux communications du ST-Link et des différents bus du système. Si la communication sans fil se fait à l'aide d'une antenne externe au microcontrôleur, cette dernière ainsi que les composants nécessaires à son fonctionnement se trouveront également ici.

Digital - GPIO

Partie numérique contenant les GPIO, les interrupteurs pour le pilotage de la haute-tension ainsi que les leds et boutons pour l'interface utilisateur.

3.3. ANALYSE MATÉRIELLE

Le schéma correspondant est illustré ci-dessous.

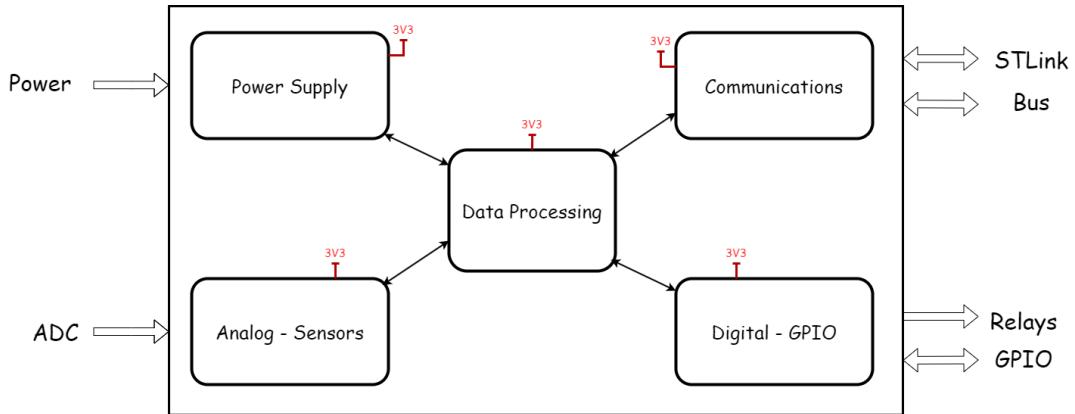


Figure 3.2 – Schéma pour analyse fonctionnelle

3.3 Analyse matérielle

Une fois l'analyse fonctionnelle complétée, il devient possible d'approfondir l'analyse du système en commençant à détailler chaque fonction. À ce niveau d'analyse, les composants ou ensembles de composants apparaissent, tout comme les connexions qui les relient. La référence des composants n'est pas déterminée ici, certaines connexions pourront être modifiées dans le schéma électrique selon les solutions choisies (protocole différent par exemple).

Certaines décisions sur la composition du système sont détaillées à cette étape, bien que toutes n'aient pas été prises à ce stade du développement. Cela concerne majoritairement la partie *Power Supply*.

Sources d'alimentation

Le cahier des charges prévoit deux types d'alimentation en énergie du système : sur pile et filaire. Si le fonctionnement sur pile n'est prévu qu'avec une seule entrée d'alimentation, il a été décidé de diversifier les sources d'alimentation filaire. Lors de l'imagination initiale du projet, seule une source filaire de 5V était envisagée. En réfléchissant cependant à un cas concret d'utilisation, il n'est pas garanti de disposer une telle source de tension à disposition dans une serre agricole. Ainsi, une seconde source de tension filaire a été envisagée pour permettre l'usage de tensions DC plus importantes. En employant un convertisseur de tension *Step-down*, il va être possible d'employer les tensions DC prévues pour alimenter les systèmes à contrôler directement par cette carte.

Sélection de la source d'alimentation

Le système a besoin d'un moyen de sélectionner sa source d'alimentation. Pour ce faire, il a d'abord été envisagé d'employer un IC dédié à cette tâche, comme le LTC4412. Cette catégorie de composants équipe beaucoup d'appareils embarqués et permet de basculer de la batterie à une source filaire lorsque cette dernière est connectée à l'appareil. Il est souvent prévu de gérer la recharge de la batterie avec ces IC. Plusieurs raisons ont fait renoncer à ce composant :

- La majorité de ces gestionnaires d'alimentation sont prévus pour un usage avec batterie (donc rechargeable). Or il n'est pas prévu ici de recharger une batterie, le système utilise donc une pile.
- Le LTC4412 convenait bien en termes de caractéristiques mais il n'était que peu disponible chez les fournisseurs et coûtait donc bien trop cher pour son utilité ($> 5.-$ CHF pour de petites quantités).
- En réfléchissant à nouveau au fonctionnement du système, il est apparu qu'un usage avec deux sources branchées simultanément n'était pas amené à se produire, ce qui évite l'emploi d'un tel composant.

Cela a finalement conduit à choisir une solution beaucoup plus simple et adaptée à l'application : un header 3x2 avec jumper pour sélectionner manuellement la source d'alimentation. C'est donc à l'utilisateur de faire ce choix (comme sur les Discovery Kits de chez STMicroelectronics), ce qui n'est pas un problème car la source d'alimentation n'est pas amenée à changer régulièrement.



Figure 3.3 – Header 3x2

3.3.1 Mesure du niveau de charge de la pile

Dans un système embarqué, l'état de la batterie se mesure à l'aide d'un *Fuel gauge*. Ce type de composants se basent souvent sur la mesure du courant au travers d'une résistance de shunt (parfois en combinaison avec une mesure de tension), c'est en intégrant ce courant qu'il parvient à estimer la capacité restante. Il n'a pas été jugé bon d'employer ce type de mesure sur une pile car il n'y a pas de recalibration possible avec une seule décharge et le profil de décharge d'une pile est relativement bien connu (donc mesure de tension suffisante). Il a donc été choisi de réaliser un petit montage permettant de mesurer la tension de la pile grâce à l'ADC du microcontrôleur. Sa réalisation est détaillée dans la section 3.4.4 *Power Supply*.

3.3.2 Régulateurs de tension

Comme beaucoup de petits systèmes embarqués, la carte est prévue pour fonctionner avec une tension de 3.3V. Il a cependant fallu déterminé si d'autres tensions n'étaient pas requises pour le fonctionnement de certaines parties. C'est notamment le cas des communications numériques (GPIO, I2C, SPI, USART) où la tension de fonctionnement du périphérique avec lequel on communique n'est pas connue. Il aurait été nécessaire d'ajouter des convertisseurs de niveaux logiques pour permettre la communication entre du 3V3 et du 5V par exemple. Après discussion avec le professeur encadrant, il apparaît que l'emploi de 3V3 convient dans la majorité des cas et qu'il n'est donc pas nécessaire d'inclure ces convertisseurs. Ainsi, un régulateur de tension 5V n'est pas nécessaire sur la carte. Une autre question se posait pour la tension de fonctionnement du module STM32WB5MMG (choix principal pour

3.3. ANALYSE MATÉRIELLE

le microcontrôleur, détaillé dans la section suivante). En effet, son fonctionnement interne se fait sous une tension de 1V8, il fallait donc clairement définir quelle tension d'appliquer lui appliquer. Après vérification, ce module doit être alimenté en 3V3, ce qui supprime l'utilité d'un régulateur de tension 1V8. Seul le régulateur de tension 3V3 est donc indispensable.

3.3.3 Module STM32

Le choix du microcontrôleur est une étape importante de la conception du système. Il est important qu'il puisse correctement répondre aux différentes contraintes du cahier des charges. Dans notre cas, plusieurs critères ont aiguillé la sélection :

- Doit pouvoir prendre en charge la communication sans fil, en particulier les protocoles mentionnés lors de l'analyse préliminaire (Bluetooth Low Energy et Zigbee).
- Doit être axé basse consommation car monté dans un système embarqué.
- Doit contenir suffisamment de broches pour prendre en charge l'ensemble du système, pour ne pas avoir à déléguer des tâches, ce qui pourrait prolonger le temps de développement.
- Doit pouvoir fonctionner avec un OS comme Mbed OS ou FreeRTOS afin de faciliter sa programmation.

Le système incluant une communication sans fil, une antenne adaptée est indispensable. Or, avec un microcontrôleur classique, la connexion de cette dernière au microcontrôleur demande un certain travail, notamment pour l'adaptation d'impédance. Cette étape aurait été faisable car il s'agit d'un domaine étudié en classe mais le temps alloué au TB étant restreint, il était préférable de pouvoir s'épargner ce travail.

L'autre moyen est de passer par un module. Il s'agit d'un composant intégré regroupant le microcontrôleur et la partie radiofréquence avec son antenne, souvent déjà optimisée et certifiée, ce qui permet de simplifier considérablement l'intégration matérielle et logicielle du système sans fil.

L'un des critères importants est la compatibilité avec la stratégie adoptée pour programmer le MCU. Dans certains cas particuliers, il est utile de travailler en *bare metal*, c'est-à-dire programmer directement sur le matériel sans système d'exploitation intermédiaire. Cette approche permet un contrôle total sur les ressources du microcontrôleur, une exécution plus rapide et une consommation énergétique minimale mais nécessite toutefois une grande connaissance des registres du MCU et implique souvent plus de temps de développement, car il faut gérer manuellement l'ensemble des services (timers, interruptions, communication, etc.).

Néanmoins, l'adoption d'un OS embarqué est préférable dans un grand nombre de cas, car elle simplifie la gestion des ressources, permet un développement plus modulaire, et favorise la réutilisabilité du code. Ce projet s'est déroulé en parallèle de cours portant sur l'OS mentionné plus tôt : Mbed OS [MBED]. Développé pour les processeurs ARM Cortex-M, Mbed OS est un système d'exploitation open source spécialement conçu pour les systèmes embarqués à faible consommation. Il fournit une couche d'abstraction matérielle, une gestion simplifiée des périphériques, ainsi que des services avancés comme la gestion des threads et l'intégration de bibliothèques

CHAPITRE 3. CONCEPTION HARDWARE

pour la connectivité (Bluetooth, Wi-Fi, LoRa, etc.). De plus, cet OS a été utilisé en cours de *Conception de Systèmes Embarqués* en parallèle de ce travail, ce qui facilite sa prise en main. Il s'agit donc du système d'exploitation privilégié pour ce projet.

Une famille de microcontrôleurs souvent utilisée par les hobbyistes est la série des ESP32, conçue par Espressif Systems. Ces microcontrôleurs sont appréciés pour leur excellent rapport qualité-prix, leur connectivité intégrée (Wi-Fi, Bluetooth), leur puissance de calcul et leur large communauté de développeurs. Ils sont très utilisés dans les projets amateurs, les objets connectés, et les prototypes nécessitant une connectivité sans fil.

Cependant, cette option a été écartée dans le cadre de ce projet pour plusieurs raisons. Tout d'abord, les ESP32 sont connus pour avoir une consommation énergétique relativement élevée, même en mode veille, ce qui peut poser problème dans des applications embarquées nécessitant une autonomie prolongée ou une alimentation sur batterie. Ensuite, et surtout, les ESP32 ne bénéficient pas d'une compatibilité officielle avec Mbed OS. Bien que certains projets communautaires aient tenté de porter Mbed OS sur certaines variantes d'ESP32, ces tentatives restent limitées, peu maintenues et ne garantissent pas une stabilité suffisante pour un projet structuré. Par conséquent, il a été jugé préférable de s'orienter vers des microcontrôleurs pleinement compatibles avec Mbed OS.

Le projet ayant été proposé par l'étudiant, un critère secondaire était de pouvoir travailler, dans la limite du possible, avec un microcontrôleur du fabricant STMicroelectronics. Leur gamme 32 bits est couramment utilisée dans l'industrie mais n'avait jamais été proposée pour des projets au cours de la formation. Ces microcontrôleurs sont, contrairement aux ESP32, basés sur l'architecture ARM, ce qui les rend compatibles avec Mbed OS. Comme expliqué précédemment, il est préférable de rechercher un module intégrant déjà la partie radiofréquence. Le travail de sélection débute donc sur la page dédiée aux MCU du site de STMicroelectronics [ST MCU]. Le choix se porte sur la gamme 32 bits, puis sur celle des MCU wireless (sans fil). On identifie alors quatre séries disponibles.

STM32WBA

Microcontrôleurs multiprotocoles en 40 nm, optimisés pour les performances et la sécurité avec Bluetooth LE, Zigbee, Thread et Matter.

STM32WB

Microcontrôleurs multiprotocoles en 90 nm, compatibles avec Bluetooth LE, Zigbee, Thread et Matter pour la maison connectée.

STM32WB0

Microcontrôleurs Bluetooth LE ultra-basse consommation, conçus pour maximiser l'autonomie des objets connectés.

STM32WL

Premiers microcontrôleurs intégrant LoRa, destinés aux réseaux longue portée à faible consommation (LPWAN).

3.3. ANALYSE MATÉRIELLE

Il est possible d'exclure la série STM32WL car elle se focalise sur la communication longue portée, ce qui n'est pas le domaine recherché. La série STM32WB0 convient pour la basse consommation mais n'inclut pas les protocoles 802.15.4. Pour les séries restantes, seule la STM32WB propose des modules : le STM32WB5MMG et le STM32WB1MMC [ST32WBXM]. Il s'avère que le STM32WB5MMG est le seul à proposer les protocoles 802.15.4, c'est donc le choix à retenir.

Il s'agit d'un module ultra-basse consommation intégrant un double cœur Cortex-M4/M0+, 1 Mo de Flash, compatible Bluetooth LE 5.4, Zigbee, Thread, Matter, avec alimentation à découpage et antenne intégrée. Il possède entre autres 68 GPIO et un ADC 12 bits et est compatible avec Mbed OS. Ce module remplit donc tous les critères, il a donc été retenu pour la conception du système. Ce module est par ailleurs monté sur une carte *Discovery Kit*, le STM32WB5MM-DK. Cela permet de mener des tests sur une carte déjà assemblée et prête à l'emploi, notamment pour la partie *Communication sans fil*. Deux de ces cartes ont ainsi été commandées dès le début du projet.

3.3.4 Sondes analogiques

Au cours de l'*analyse préliminaire*, il est apparu que les différentes sondes qu'il serait bon de lire possèdent toutes des caractéristiques différentes et qu'il n'est pas possible de développer un système "universel" capable de toutes les traiter. Ainsi, il a été choisi de ne pas développer l'interface entre les sondes et l'ADC directement sur la carte mais de mettre uniquement en place la lecture de ces différentes tensions analogiques. Pour recentrer les objectifs et être capable de terminer le projet dans les temps, il a été décidé de ne travailler qu'avec les produits du fabricant Atlas Scientific. Ils proposent différentes interfaces pour leurs sondes, toutes fonctionnent avec du 3V3 et fournissent une tension analogique comprise entre 0V et 3.0V. Ainsi, il ne faut prévoir que l'alimentation de cette interface et la lecture de cette tension analogique (3 broches).

Au terme de ces réflexions, il est possible de dresser un schéma plus complet du système. Ce dernier est illustré ci-dessous.

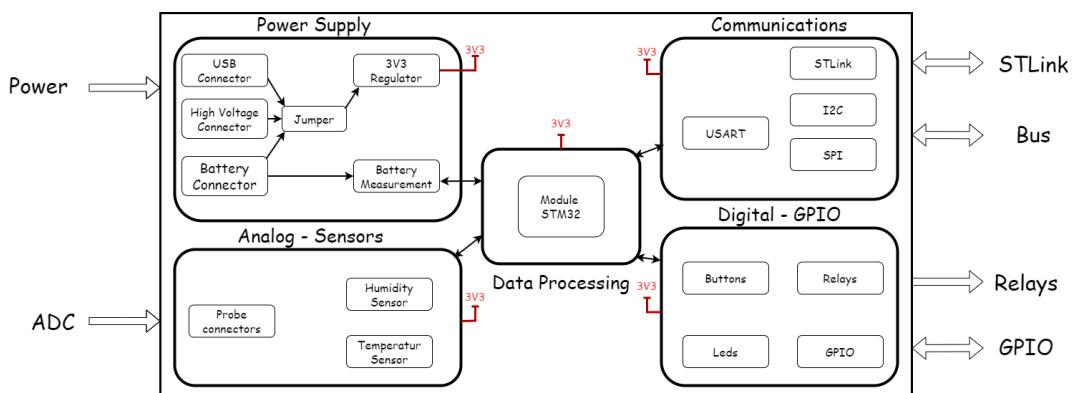


Figure 3.4 – Schéma pour analyse matérielle

3.4 Schéma électrique

Avec le niveau de détails de l'analyse matérielle, le schéma électrique de la carte peut commencer à être rempli. C'est également à cette étape que le choix des différents composants se fait. Il va dépendre de

- la disponibilité des composants chez les différents fournisseurs
- le prix des composants
- la qualité de la documentation du composant et la renommée du fabricant
- la disponibilité des ressources CAD du composant (symbole, footprint, modèle 3D)

Le logiciel utilisé pour cela est **Altium**. Il permet de concevoir des circuits imprimés (PCB), en intégrant à la fois la capture schématique, le routage, la gestion des composants et la simulation électronique. Cette solution est beaucoup utilisée dans le milieu professionnel et également pour l'enseignement au sein de la HEIG-VD. Si le projet devait se réaliser dans un autre établissement possédant moins de budget, sa complexité modérée permettrait de travailler avec d'autres logiciels moins onéreux, comme KiCad par exemple.

3.4.1 Ressources composants

Pour compléter le schéma électrique, il est essentiel d'avoir les ressources CAD des composants qui le composent. Ces ressources se décomposent en trois parties :

- Le symbole schématique : représentation graphique du composant utilisée dans le schéma électrique, avec ses broches et ses désignations.
- L'empreinte (footprint) : représentation physique du composant destinée au routage du PCB, incluant les dimensions exactes, les pastilles de soudure et les distances entre les broches.
- Les modèles 3D : permettent de visualiser l'intégration mécanique du composant sur le circuit imprimé et de vérifier les contraintes d'encombrement.

Il est possible de réaliser ces différentes parties à la main mais cela représente un investissement de temps important et n'est pas représentatif des pratiques industrielles. Il est préférable en effet de récupérer ces ressources dans de grandes bibliothèques disponibles sur Internet (SamacSys, Ultra Librarian, etc.).

Une licence académique fournie par l'école est utilisée pour ce projet, elle permet d'avoir accès à une grande bibliothèque de composants alimentée par les membres de la HEIG-VD. Peu après le début du travail sur le schéma électrique, l'extension *Altium Library Loader V2.2* de SamacSys a été installée sur Altium [Samacsy Guide]. Cette extension permet de rechercher les ressources CAD d'un composant depuis Altium dans la riche bibliothèque de SamacSys. Cela permet un grand gain de temps et un accès à des ressources de bonne qualité, la vérification de ces dernières étant toujours de mise.

3.4. SCHÉMA ÉLECTRIQUE

3.4.2 Décomposition du schéma

Sur Altium, il est possible de diviser son schéma électrique sur plusieurs pages afin d'améliorer la visibilité et le découpage du système. Pour ce faire, les *Sheet Symbols* sont utilisés sur la page principale du schéma. Elles représentent des sous-parties du projet et permettent de structurer le schéma de manière hiérarchique. Chaque Sheet Symbol agit comme un lien vers une autre page du schéma, contenant une partie spécifique du circuit.

Le découpage réalisé lors de l'analyse fonctionnelle peut être récupéré ici pour séparer le circuit selon ses fonctions principales. Ainsi, le schéma est découpé en 5 parties :

- Power Supply
- STM32
- Analog - Sensors
- Digital - GPIO
- Communications

3.4.3 Choix généraux

Certains choix affectent plusieurs parties du système, ils sont expliqués ci-dessous.

Protections ESD

Les points d'accès au système mentionnés lors de l'analyse des connexions externes doivent être protégés contre les décharges électrostatiques. Pour ce faire, il est commun d'employer des TVS (Transient Voltage Suppressors), qui protègent les circuits en limitant rapidement les surtensions transitoires. Les TVS peuvent être unidirectionnelles ou bidirectionnelles, selon qu'elles protègent contre des surtensions dans une seule direction ou dans les deux directions, et doivent être dimensionnées en fonction de la tension à laquelle elles conduisent (breakdown).

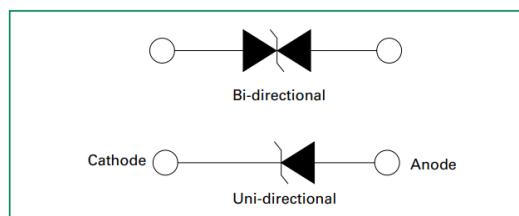


Figure 3.5 – TVS bidirectionnelle et unidirectionnelle

Les entrées de la partie *Power Supply* sont à protéger avec des TVS différentes car elles n'accueillent pas les mêmes tensions. La bibliothèque de composants de l'école contient déjà un grand nombre de TVS, il est pertinent de les utiliser lorsque c'est possible. Le connecteur USB-C fournit une tension 5V à la carte, la TVS *CDSOD323-T08S* lui est assignée. Il s'agit d'un composant unidirectionnel possédant une tension de breakdown de 8.5V. Il est également assigné au connecteur de la pile dont la tension maximale se trouvera entre 4.5V et 6V. Pour le connecteur d'alimentation à haute tension, la tension maximale de fonctionnement est de 36V. La TVS unidirectionnelle *824500331* aussi trouvée dans la bibliothèque de composants est bien adaptée, avec une tension de clamping de 38.65V.

CHAPITRE 3. CONCEPTION HARDWARE

Les entrées et sorties *ADC*, *GPIO* et *Bus* des premiers schémas fonctionnent avec une tension de 3V3. La bibliothèque de composants ne comporte pas de TVS adaptées, une référence trouvée chez le fournisseur Mouser a été utilisée. Il s'agit de la TVS *PESD3V3L1BSLZ* [TVS], elle est bidirectionnelle et possède une tension de breakdown de 5.2V.

Pour le ST-Link, amené à n'être que peu employé car utilisé pour la programmation de la carte, les signaux ne sont pas protégés à l'aide de TVS. En effet, les pistes contiennent à la place des résistances de $22\ \Omega$ placées en série, ce dispositif a été repris du schéma électrique du Discovery Kit STM32WB5MM-DK.

Les relais MOSFET employés pour contrôler les grands courants (détails dans la section 3.4.6) contiennent un optocoupleur. Après discussion avec le professeur encadrant à ce sujet, il n'a pas été jugé utile de placer des protections ESD supplémentaires.

Connecteurs filaires

Dans les parties *Power Supply* et *Digital GPIO*, il est nécessaire de prévoir des connecteurs capables d'accueillir une paire de fils nus. Ainsi, une référence a été trouvée pour ces cas, il s'agit du connecteur 1729128 du fabricant Phoenix Contact [CONN]. Couramment utilisé dans l'industrie, il permet le serrage de fils nus à l'aide de petites vis. Ses ressources CAD ont été récupérées sur SamacSys puis vérifiées avant d'être intégrées au circuit.

3.4.4 Power Supply

USB

Le protocole USB a été choisi pour l'alimentation du système car il fonctionne avec une tension de 5V et est très répandu. Le transport de données n'est cependant pas prévu via ce bus, seul l'apport d'énergie est exploité pour ce circuit. Plusieurs types de connecteurs USB sont apparus au cours de l'évolution de ce protocole, la version la plus récente est utilisée pour ce projet, à savoir le connecteur USB Type-C. La première version du connecteur ajouté au schéma fut trouvée dans la bibliothèque de composants de l'école. Il s'agissait d'un connecteur THT, traversant le PCB avec ses fixations, et possédant les pattes requises pour la communication.

Après discussion avec le professeur encadrant, il fut décidé de remplacer le connecteur traversant (THT) par une version en montage en surface (SMD) afin de limiter l'interruption du plan de masse sur la couche intérieure. Cette version ne comporte pas de pattes de communication, elle n'est utile que pour l'alimentation de la carte. 2 pattes sont prévues pour le GND, directement reliées à la masse du circuit, et 2 pattes pour l'alimentation 5V. Ces dernières sont directement raccordées au sélecteur de source de tension. La structure métallique du connecteur est également relié au GND via ses 4 pattes de fixations. Ses ressources CAD ont été récupérées sur SamacSys puis vérifiées avant d'être intégrées au circuit.

Pile

Le connecteur de la pile est relié à une diode Schottky sur la broche positive dans le but de protéger le circuit d'une inversion de polarité de la pile. La tension de la pile se situant dans la même plage de tension que la tension de sortie du régulateur de tension 3V3, il est important que la chute de tension présente sur la diode de protection soit la plus faible possible (chute de tension d'environ 350 mV ici), raison pour laquelle une diode Schottky a été préférée à une diode classique. Un fusible

3.4. SCHÉMA ÉLECTRIQUE

PTC (Positive Temperature Coefficient) a également été ajouté pour prévenir des surcourants. Ces deux composants se trouvaient dans la bibliothèque de composants de l'école.

Mesure de la tension de la pile

Comme indiqué dans la section 3.3.1, un petit montage a été mis en place avec l'aide du professeur pour mesurer la tension de la pile. Il est principalement composé d'une paire complémentaire de transistors MOSFET et d'un pont diviseur.

La paire de MOSFET est constituée d'un MOSFET canal N et d'un MOSFET canal P, les deux MOSFET sont montés drain-à-drain (back-to-back). Cela permet d'obtenir un interrupteur bidirectionnel capable de bloquer le courant dans les deux sens quand il est éteint, minimisant donc les pertes. Ainsi, lorsque la mesure est inactive, le pont diviseur et l'ADC n'ont pas accès à la tension de la pile, ce qui évite de la décharger inutilement. Le schéma ci-dessous illustre la configuration employée, résistance incluse, mais avec une autre version du circuit intégré utilisé.

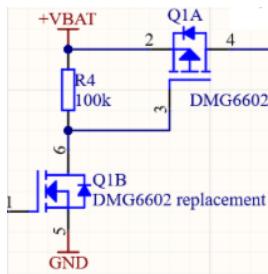


Figure 3.6 – Configuration des MOSFET pour mesure de la tension de pile

Les deux transistors sont contenus dans le circuit intégré DMC3071LVT [DMC3071], il s'agit du composant remplaçant le DMG6602 (selon sa fiche technique), présent sur l'illustration ci-dessus. Le rapport des résistances du diviseur de tension a été choisi de manière à ramener la tension maximale en entrée de l'ADC à sa valeur maximale admissible. L'hypothèse retenue établit que la tension maximale de la pile à 4,5V (3 piles AAA en série). En prenant en compte la chute de tension de la diode Schottky ($\approx 350\text{mV}$), le rapport des résistances a été déterminé pour atteindre une tension légèrement inférieure ou égale à 3,3V en sortie du pont.

$$\frac{R_2}{R_1 + R_2} \approx \frac{3,3}{4,5 - 0,35} \approx \frac{82\text{k}\Omega}{22\text{k}\Omega + 82\text{k}\Omega} \quad (3.1)$$

Ce choix est confirmé par le calcul de la tension en sortie du diviseur pour une tension d'entrée de 4,15 V (4,5 - 0,35) :

$$V_{\text{out}} = 4,15 \times \frac{82\text{k}\Omega}{22\text{k}\Omega + 82\text{k}\Omega} = 3,27\text{ V} \quad (3.2)$$

Cette valeur reste inférieure à la limite de 3,3 V, assurant ainsi la sécurité de l'ADC.

Une diode Zener a été placée pour protéger le MCU en cas de tension de pile supérieure à 4,5 V. La tension Zener est fixée à 3,6 V, limitant ainsi la tension

maximale appliquée à l'entrée analogique. Selon sa fiche technique, la diode présente une capacité parasite de 250 pF.

Pour atténuer les éventuels bruits haute fréquence sur la ligne de mesure, un condensateur de 1 nF a été ajouté en parallèle à l'entrée de l'ADC, formant ainsi un filtre passe-bas RC avec les résistances du pont diviseur. La constante de temps τ du filtre est donnée par la formule suivante :

$$\tau = R_{\text{eq}} \cdot C_{\text{tot}} = \left(\frac{R_1 \cdot R_2}{R_1 + R_2} \right) \cdot (C + C_{\text{Zener}}) \quad (3.3)$$

avec :

$$R_1 = 22 \text{ k}\Omega, \quad R_2 = 82 \text{ k}\Omega, \quad C = 1 \text{ nF}, \quad C_{\text{Zener}} = 250 \text{ pF}$$

Le calcul donne :

$$\tau = \left(\frac{22 \text{ k}\Omega \cdot 82 \text{ k}\Omega}{22 \text{ k}\Omega + 82 \text{ k}\Omega} \right) \cdot (1 \text{ nF} + 250 \text{ pF}) \approx 21 \mu\text{s} \quad (3.4)$$

Cette constante de temps est suffisante pour filtrer les perturbations haute fréquence tout en maintenant une réponse rapide du signal mesuré. La figure suivante présente une simulation effectuée à l'aide de *NumPy*, illustrant la tension en sortie du montage, au niveau de l'ADC, en fonction de la tension de la pile, sur une plage allant de 3,5 V à 6 V.

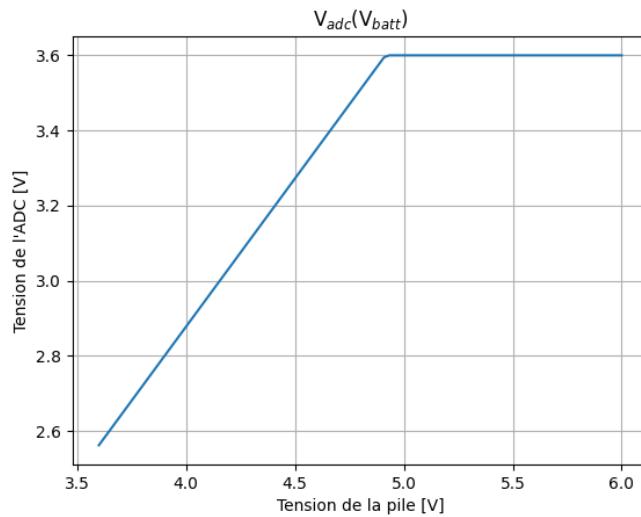


Figure 3.7 – Simulation V_{ADC} en fonction de V_{batt}

Convertisseur de tension Step-down

Le circuit intégrant la possibilité d'être alimenté à l'aide de hautes tensions DC, il a besoin d'un convertisseur de tension permettant d'abaisser ces tensions. Le régulateur de tension du système fournissant du 3V3, le convertisseur doit pouvoir fournir du 5 V (légèrement supérieur à 3V3 pour pallier les différentes chutes de tension, y compris au sein du régulateur). Les hautes tensions à convertir peuvent

3.4. SCHÉMA ÉLECTRIQUE

typiquement se trouver dans une gamme allant de 6 V à 24 V (petits moteurs DC). Avec une telle différence entre l'entrée et la sortie, les régulateurs linéaires ne sont plus une bonne option, notamment en raison de leur faible rendement énergétique, car toute la différence de tension est dissipée sous forme de chaleur. Il est donc préférable d'opter pour des convertisseurs à découpage (SMPS), qui permettent de réguler la tension avec un bien meilleur rendement, même en présence d'une grande différence entre entrée et sortie.

Il aurait été possible de concevoir un tel convertisseur DC/DC directement sur la carte car son fonctionnement interne a été étudié en cours, mais les fabricants de composants proposent des modules clé en main et à de bons prix. Après des recherches chez les différents revendeurs en ligne, le TSR 1-2450E de Traco Power [TSR] a été retenu pour ce projet. Ce convertisseur *Step-down* fournit du 5 V à partir d'une tension d'entrée comprise entre 7 V et 36 V et peut atteindre 1 Ampère en sortie. Ses ressources CAD ont été récupérées sur SamacSys puis vérifiées avant d'être intégrées au circuit.

Pour protéger ce convertisseur contre les inversions de polarité sur le connecteur, une diode trouvée sur la bibliothèque de composants de l'école a été placée entre les deux.

Régulateur de tension 3,3V

Élément indispensable pour l'alimentation du système, le régulateur de tension fournit une tension définie et stable à l'ensemble du circuit. Il reçoit ainsi en entrée la tension sélectionnée à l'aide du header 3x2 et limitée à 6 V par une diode Zener qui agit comme une protection. Le choix du régulateur de tension se fait plutôt sur la fin de la conception du schéma car il faut être en mesure d'estimer la quantité de courant que le système requiert. Cette estimation se trouve dans la section *3.4.9 Estimation du courant*.

Il existe deux grandes catégories de régulateurs de tension : les régulateurs linéaires et les régulateurs à découpage. Le schéma initial du système avait été conçu avec un régulateur linéaire de type LDO (*Low Dropout Regulator*), un composant qui permet de fournir une tension de sortie régulée avec une chute de tension minimale entre l'entrée et la sortie. Les LDO ont pour avantages leur simplicité d'implémentation, leur faible coût et leur faible niveau de bruit, ce qui les rend adaptés à des applications où la précision de régulation est plus critique que le rendement énergétique. Cependant, après discussion avec le professeur encadrant, il a été décidé que cette solution n'était pas la plus adaptée au contexte du projet. En effet, bien que les régulateurs LDO soient efficaces pour des alimentations simples, leur rendement reste relativement faible. Dans notre cas, le système étant destiné à une application embarquée nécessitant une autonomie de plusieurs semaines, le rendement énergétique est un critère crucial. Par ailleurs, un LDO impose que la tension d'entrée soit strictement supérieure à la tension de sortie. Or, avec une alimentation initialement prévue par piles (4,5 V), la chute de tension provoquée par la diode Schottky (0,35 V) réduit rapidement cette marge. À mesure que la tension des piles diminue au cours du temps, il devient difficile de maintenir une tension de sortie stable de 3,3 V avec un régulateur linéaire.

Pour pallier ces limitations, il a été décidé d'utiliser un régulateur à découpage de type *buck-boost* (ou abaisseur-élévateur). Ce type de régulateur est capable, comme

son nom l'indique, d'abaisser ou d'élever la tension d'entrée pour maintenir une tension de sortie constante. Contrairement aux LDO, les convertisseurs buck-boost fonctionnent par découpage rapide de la tension d'entrée. Ce principe leur permet d'atteindre des rendements élevés, souvent supérieurs à 85 %, tout en s'adaptant à une large plage de tensions d'entrée. La référence retenue pour ce projet est le *TPS63001* de Texas Instruments [TPS63001]. Il est capable d'assurer une tension de sortie de 3.3V avec une plage d'entrée allant de 1.8V à 5.5V tout en étant capable de fournir au minimum un courant de 800mA. La figure ci-dessous illustre son fonctionnement typique, repris dans notre circuit, avec une tension fixe de 3.3V en sortie.

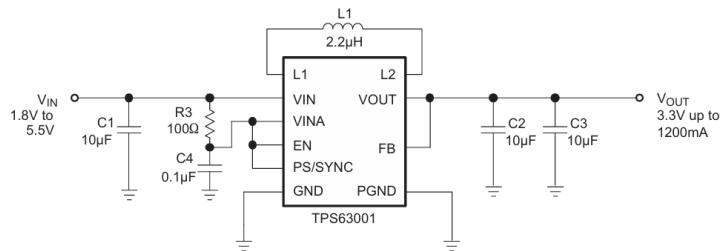


Figure 3.8 – Application typique du TPS63001 selon son datasheet

Ce type de convertisseur implique d'y associer une inductance, la fiche technique fournit une liste de références recommandées pour nous aiguiller. La *LPS3015-222MLB* du fabricant Coilcraft [INDUCTOR] a été choisie car disponible en grande quantité. Toutes les ressources CAD ont été importées et vérifiées depuis SamacSys.

3.4.5 Analog - Sensors

Connecteurs sondes analogiques

Comme indiqué précédemment, il est nécessaire d'avoir 3 pattes pour utiliser les interfaces Atlas Scientific, 2 pour l'alimentation et 1 pour le signal analogique. Côté interface, le connecteur est un header mâle coudé 3x1. Il a donc été décidé de choisir le même type de connecteur sur la carte afin de pouvoir les relier avec des fils possédant des headers femelles aux extrémités. La référence trouvée chez les revendeurs de composants est le PH1RB-03-UA de Adam Tech. Ses ressources CAD ont été récupérées sur SamacSys puis vérifiées avant d'être intégrées au circuit.

Le signal analogique va directement jusqu'à l'ADC du microcontrôleur, sans être redimensionné. En parcourant les différentes interfaces de Atlas Scientific, il apparaît que ces signaux analogiques ne se situent pas tous sur une plage de 0 V à 3 V. Pour l'oxygène dissous et la température, les tensions vont respectivement de 0 V à 420 mV et de 1.058 V à 1.3 V. On pourrait à première vue penser qu'il serait bon de retravailler ces tensions pour augmenter la dynamique mais cela impactera également celle allant de 0 V à 3 V.

En analysant le datasheet du STM32WB5MMG, on apprend qu'il possède un ADC 12 bits, sa résolution est donc calculée de la manière suivante.

$$\text{Résolution} = \frac{V_{\text{ref}}}{2^{12}} = \frac{3.3}{4096} \approx 0.805 \text{ mV} \quad (3.5)$$

3.4. SCHÉMA ÉLECTRIQUE

Cette résolution, inférieure à 1 % de la plus petite plage de tension à mesurer, permet largement d'assurer une mesure de qualité suffisante, surtout pour une telle application où un niveau extrême de précision n'est pas requis.

Load Switchs

Dans un système embarqué, la consommation de courant est un élément crucial qu'il faut parvenir à minimiser, c'est pourquoi il est courant d'utiliser des interrupteurs pour couper l'alimentation d'éléments lorsque ces derniers ne sont pas actifs. Ces interrupteurs, appelés *load switchs*, permettent de contrôler l'alimentation de différents blocs du système de manière fine et automatisée, tout en garantissant une consommation réduite. Pour ce projet, la référence du composant a été fournie par le professeur car elle était déjà connue et employée, il s'agit du XC8102AA01MR-G de Torex [XC8102]. Ses ressources CAD ont été récupérées sur SamacSys puis vérifiées avant d'être intégrées au circuit.

Capteur de température et d'humidité

Plusieurs solutions sont disponibles pour relever la température. Il est possible d'utiliser une cellule sensible à la chaleur et d'élaborer un petit montage pour collecter une tension analogique avec l'ADC mais cela s'avère moins fiable et plus complexe que la seconde option. La solution retenue est un IC contenant une cellule sensible à la température mais également à l'humidité. La conversion analogique-numérique se fait à l'intérieur et il est possible de communiquer avec ce composant via le bus I2C. C'est une solution fiable, compacte et bon marché, la référence du composant est SHT41-AD1B-R2 de Sensirion [SHT41]. Ses ressources CAD ont été récupérées sur SamacSys puis vérifiées avant d'être intégrées au circuit. La figure suivante est tirée de la datasheet du composant et représente l'application typique de ce dernier.

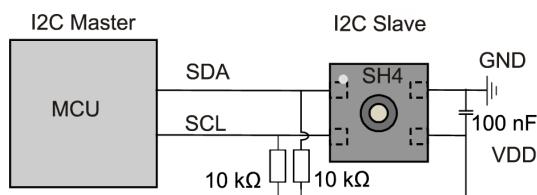


Figure 3.9 – Application typique du SHT41-AD1B-R2 selon son datasheet

Capteur de luminosité

Le choix de l'intégration d'un capteur de luminosité dans le système s'est fait au cours de la conception de la carte. Le raisonnement sur la solution retenue est identique à celui du capteur de température et d'humidité. Ainsi, un capteur sous forme de circuit intégré et communiquant à l'aide du bus I2C a été sélectionné, il s'agit du VEML6030 de Vishay [VEML6030]. Ses ressources CAD ont été récupérées sur SamacSys puis vérifiées avant d'être intégrées au circuit. Son adresse I2C peut être configurée à l'aide de la patte *ADDR*, elle est fixée à 0x10 si la patte est reliée au GND et à 0x48 si reliée au VCC. La première option a été retenue, la patte a été reliée au GND.

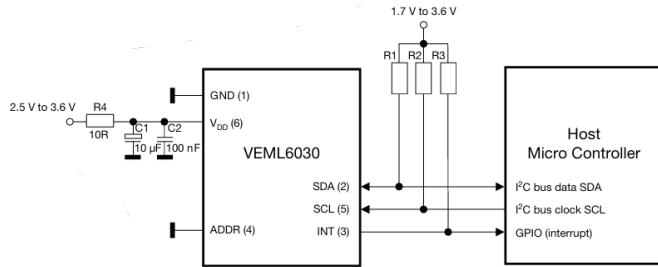


Figure 3.10 – Application typique du VEML6030 selon son datasheet

Ce composant est capable de lancer des alertes via sa patte *INT*, cette dernière nécessite une résistance de pull-up qui sera ajoutée dans la configuration de la GPIO.

3.4.6 Digital - GPIO

GPIO

Deux boutons ont été placés sur le circuit, un pour le *reset* du microcontrôleur et l'autre pour un usage général. Cinq connecteurs ont été reliés aux GPIO du MCU pour un usage général et un connecteur 1729128 a également été utilisé pour utiliser un flotteur. Ce flotteur peut permettre de mesurer le niveau d'eau d'un bassin en agissant comme un switch.

Leds

Deux leds ont été placées sur le schéma, elles sont toutes les deux rouges pour limiter le nombre de références différentes et parce que les leds de cette couleur ont une tension de fonctionnement inférieure aux autres couleurs (2 V). Les résistances qui les accompagnent ont été dimensionnées de sorte à obtenir un courant de fonctionnement de 10 mA selon la formule suivante.

$$R_r = \frac{3.3 - V_f}{I_f} = \frac{3.3 - 2}{10 \times 10^{-3}} = 130 \Omega \rightarrow 120 \Omega \text{ (E12)} \quad (3.6)$$

avec :

$$V_f = 2 \text{ V}, \quad I_f = 10 \text{ mA}$$

Relais MOSFET

Des relais MOSFET sont intégrés dans le circuit pour contrôler des actionneurs fonctionnant avec une tension supérieure à celle de la carte. Ces composants sont présents pour enclencher ou déclencher l'alimentation de l'appareil à piloter. La référence retenue pour ce projet est le *G3VM-61VY3TR05* de Omron Electronics [MOSFET]. Ce composant peut supporter une tension de charge de 60V max avec un courant continu de 700 mA. Il agit comme un optocoupleur et possède ainsi une isolation galvanique qui protège le circuit. La led qu'il contient est caractérisée par une tension de fonctionnement de 1.27 V et un courant direct de 10 mA, ce qui permet de dimensionner une résistance de limitation selon la formule suivante.

3.4. SCHÉMA ÉLECTRIQUE

$$R_r = \frac{3.3 - V_f}{I_f} = \frac{3.3 - 1.27}{10 \times 10^{-3}} = 200 \Omega \rightarrow 220 \Omega \text{ (E12)} \quad (3.7)$$

avec :

$$V_f = 1.27 \text{ V}, \quad I_f = 10 \text{ mA}$$

3.4.7 Communications

Cette partie du système contient l'ensemble des interfaces permettant de communiquer avec un système externe au moyen d'un câble. Le bus *SPI* (Serial Peripheral Interface) n'est pas utilisé au sein de la carte mais il était tout de même prévu au départ de placer des broches pour permettre son usage avec un appareil externe. Pour des raisons d'encombrement lors du routage, il a été décidé de le supprimer.

ST-Link

Il est essentiel de prévoir un moyen de programmer et déboguer le microcontrôleur, chez STMicroelectronics cela se fait par un ST-Link. Le ST-Link est une sonde de programmation et de débogage qui permet une interface directe entre l'ordinateur et le microcontrôleur via les protocoles SWD ou JTAG. L'institut *IICT* dans lequel s'est déroulée la conception de la carte électronique possède déjà des ST-Link, il n'était donc pas nécessaire d'en acheter un nouveau. Celui proposé par le professeur pour ce projet est un ST-Link-V3SET [ST-LINK]. Ce modèle, complet et bien adapté au STM32WB5MMG, possède entre autres les interfaces JTAG et SWD, principalement utilisées pour la programmation et le débogage des microcontrôleurs 32 bits du fabricant. JTAG offre une solution complète multi-broches, tandis que SWD, plus léger, n'utilise que deux fils pour un fonctionnement équivalent sur les coeurs ARM Cortex-M. Les deux interfaces partagent les mêmes pattes sur le microcontrôleur, ce qui simplifie le routage.

Le ST-Link est un boîtier reliant l'ordinateur et le microcontrôleur, mais il faut encore définir le type de connecteur utilisé sur la carte. Pour cela, il a été conseillé par le professeur de se rendre sur le site de Tag-connect, un fabricant spécialisé dans ce genre de connecteurs. Sur leur site [Tag-Connect], on peut trouver leurs offres pour le ST-Link-V3SET, et notamment le TC2050-IDC-050-STDC14. Il s'agit d'un câble doté, côté PCB, d'un connecteur TC2050 "plug-and-play" conçu pour s'insérer directement dans des trous métallisés, sans nécessiter de connecteur soudé sur la carte, grâce à des crochets en plastique assurant un bon maintien mécanique. Côté ST-Link, on retrouve un connecteur STDC14 femelle, la version mâle étant présente sur le boîtier. Ce câble, principale offre du fabricant pour ce ST-Link, a donc été retenu pour le projet.

CHAPITRE 3. CONCEPTION HARDWARE

Le nombre de pattes diffère d'un bout à l'autre du câble, 14 pattes pour le ST-Link contre 10 pour le PCB. Aucune information n'est donnée sur le datasheet du câble, il faut lire la description du produit sur le site Tag-connect pour obtenir des informations. Les broches 3 à 12 du connecteur STDC14 correspondent aux broches 1 à 10 du footprint TC2050, selon un décalage fixe : chaque broche n du TC2050 est reliée à la broche $n+2$ du STDC14, tandis que les broches 1, 2, 13 et 14 du STDC14 ne sont pas connectées. Les ressources CAD de ce câble ont été récupérées sur SamacSys puis vérifiées avant d'être intégrées au circuit.

Une fois le symbole ajouté au schéma électrique, il convient d'étudier la façon dont il faut le relier au microcontrôleur. Le manuel d'utilisation du ST-Link-V3SET contient une première partie de cette information dans le tableau 6, on y retrouve les différents signaux en provenance du ST-Link. Il est reporté ci-dessous.

Pin number	Description	Pin number	Description
1	Reserved ⁽¹⁾	2	Reserved ⁽¹⁾
3	T_VCC ⁽²⁾	4	T_JTMS/T_SWDIO
5	GND	6	T_JCLK/T_SWCLK
7	GND	8	T_JTDO/T_SWO ⁽³⁾
9	T_JRCLK ⁽⁴⁾ /NC ⁽⁵⁾	10	T_JTDI/NC ⁽⁵⁾
11	GNDetect ⁽⁶⁾	12	T_NRST
13	T_VCP_RX ⁽⁷⁾	14	T_VCP_TX ⁽²⁾

Table 3.1 – Tableau 6 pour pinout du ST-LINK-V3SET

⁽¹⁾ Ne pas connecter à la cible.

⁽²⁾ Entrée pour le STLINK-V3SET.

⁽³⁾ SWO est optionnel, requis uniquement pour la trace SWV (Serial Wire Viewer).

⁽⁴⁾ Bouclage optionnel de T_JCLK sur la carte cible, requis si le bouclage est retiré du côté du STLINK-V3SET.

⁽⁵⁾ NC signifie « Non Connecté », donc non requis pour la connexion SWD.

⁽⁶⁾ Relié à la masse (GND) par le firmware du STLINK-V3SET ; peut être utilisé par la cible pour détecter l'outil.

⁽⁷⁾ Sortie pour le STLINK-V3SET.

La seconde partie se découvre soit en consultant la documentation du STM32WB5MMG, qui n'est pas très claire sur le mapping des interfaces JTAG/SWD, soit en utilisant le configurateur STM32CubeMX, option retenue ici. STM32CubeMX est un outil graphique de STMicroelectronics permettant de configurer facilement les périphériques et les broches d'un microcontrôleur.

Pour configurer, et dans notre cas définir les pattes utilisées pour la programmation du microcontrôleur, il faut se rendre dans le MCU Selector, puis rechercher celui que l'on utilise. En recherchant "STM32WB5MMG", on tombe sur 2 références dont l'une semble être inactive. On choisit logiquement celle active : STM32WB5MMGH6TR.

3.4. SCHÉMA ÉLECTRIQUE

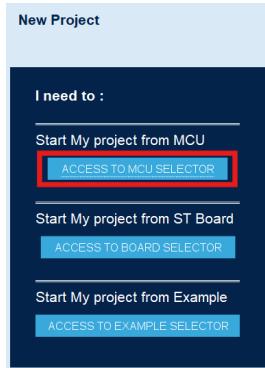


Figure 3.11 – Accès au sélecteur de MCU

The screenshot displays the STM32WB Series product page for the STM32WB5MMGH6TR. It includes a summary table with details like Unit Price for 10kU (US\$: 6.0), Boards (STEVAL-ASTR1B - STEVAL-PROTEUS1 - STM32WB5MM-DK), and Package (SIP LGA 86 7.3x11x1.342 mm). Below the table, a detailed description of the module's features and performance is provided, mentioning its ultra-low-power nature, dual-core Cortex-M4/M0+ architecture, and support for various wireless protocols.

Commercial ...	Part No	Reference	Marketing ...	Unit Price for 1 ...	Board	Package	Flash	RAM	I/O	Frequ...
★ STM32WB5MM...	STM32WB5...	STM32WB5M...	Preview	NA	SIP LGA 86 7.3x11x1.342 ... 1024 kB... 256 kB... 73	1024 kB... 256 kB... 73	64 MHz			
★ STM32WB5MM...	STM32WB5...	STM32WB5M...	Active	6.0	STEVAL-AS STEVAL-PR STM32WB5...	SIP LGA 86 7.3x11x1.342 ... 1024 kB... 256 kB... 73	64 MHz			

Figure 3.12 – MCU sélectionné

Il faut ensuite créer le projet pour attérir sur la page de configuration. La configuration des interfaces de programmation se fait généralement depuis l'onglet *Debug*, or il n'est pas présent pour ce MCU. Il faut en vérité se rendre dans l'onglet *System Core* puis dans *SYS* pour accéder à la configuration du *Debug*. Un menu déroulant liste les options pour les différentes interfaces. En les comparant, il apparaît que toutes utilisent les mêmes pattes. La configuration où le plus grand nombre de pattes est activé est "JTAG (5 pins)", les 5 pattes sont listées ci-dessous.

- PA13 : SYS_JTMS-SWDIO
- PA14 : SYS_JTCK-SWCLK
- PA15 : SYS_JTDI
- PB3 : SYS_JTDO-SWO
- PB4 : SYS_JTRST

Les pattes du SWD peuvent être identifiées en sélectionnant "Serial Wire" dans ce même menu déroulant, il s'agit des pattes PA13 et PA14 (déjà contenues dans la liste). Cela valide donc le choix des 5 pattes qui seront reliées sur la carte jusqu'au connecteur du ST-Link.

CHAPITRE 3. CONCEPTION HARDWARE

Le tableau 6 est donc repris pour établir les liaisons sur le schéma électrique. Après avoir consulté la documentation, la broche 7 *T_JRCLK/NC* est une broche optionnelle de "loopback", utile pour valider le bon fonctionnement de loopback mais qui n'est pas utilisée car ce mécanisme est déjà présent côté ST-Link-V3SET. Elle n'est donc pas connectée. La broche 9 *GNDetect* est tirée au GND par le ST-Link et peut servir pour un test de présence depuis le microcontrôleur. Elle a donc été initialement reliée à une GPIO pour permettre la détection du programmeur, mais a finalement été retirée lors du routage car il était nécessaire d'économiser de la place et cette connexion n'était pas prioritaire. Les broches d'alimentation sont reliées au VCC et GND de la carte, le reste des signaux sont raccordés aux pattes du microcontrôleur selon la liste suivante.

- PA13 : T_JTMS/T_SWDIO
- PA14 : T_JCLK/T_SWCLK
- PA15 : T_JTDI/NC
- PB3 : T_JTDO/T_SWO
- PB4 : T_NRST

Des résistances de $22\ \Omega$ ont été rajoutées en série pour les signaux à destination du microcontrôleur. C'est une reproduction de ce qui a été fait sur le Discovery Kit du STM32WB5MMG pour le Tag connect, permettant une limitation du courant.

I2C

Le bus I2C (Inter-Integrated Circuit) est l'unique protocole utilisé sur la carte pour assurer la communication entre les deux capteurs et le microcontrôleur. Sa présence est donc indispensable. Un connecteur de type header 2×1 a été ajouté pour permettre le raccordement de périphériques supplémentaires à la carte en vue d'un usage futur. Le protocole I2C requiert l'ajout de résistances de pull-up sur ses lignes de communication : SCL (clock) et SDA (data). Ces résistances ont été dimensionnées après le placement des composants sur la carte, afin de connaître la longueur de chaque piste. Leur valeur a été fixée à $10\ k\Omega$. Le raisonnement ayant conduit à ce dimensionnement est détaillé ci-dessous.

Le dimensionnement se base sur la note d'application *I2C Bus Pull-up Resistor Calculation* de Texas Instruments [I2C Calc], ses informations sont cohérentes avec d'autres guides et les conseils du professeur encadrant. À des fréquences de fonctionnement supérieures à $100\ kHz$, l'effet des capacités parasites du bus devient significatif. Ces capacités proviennent des pistes de circuit imprimé ainsi que des entrées des composants maîtres et esclaves. Elles introduisent un comportement de type RC qui peut ralentir les temps de montée et de descente des signaux, compromettant ainsi la validité de la communication.

Les résistances de pull-up doivent donc être :

- suffisamment grandes pour respecter les temps de transition imposés par la norme I2C,
- mais pas trop grandes, afin de garantir que les périphériques puissent tirer le courant nécessaire pour générer des niveaux logiques bas.

3.4. SCHÉMA ÉLECTRIQUE

La valeur maximale que doit respecter la résistance est donnée par la relation suivante :

$$R_p(\max) = \frac{t_r}{0.8473 \times C_b} \quad (3.8)$$

où :

- $R_p(\max)$ est la résistance maximale de pull-up (en Ohms),
- t_r est le temps de montée maximal autorisé par la norme (en secondes),
- C_b est la capacité totale du bus (en farads),
- 0,8473 est un facteur empirique dérivé de l'analyse du circuit RC à 30%–70% du temps de montée.

Cette formule permet d'assurer que le signal SDA/SCL atteindra un niveau haut dans un délai compatible avec la fréquence du bus I2C. Le temps de montée maximal dépend du mode de fonctionnement du bus. Dans notre cas, il ne sera pas beaucoup sollicité et peut donc être configuré en mode *Standard*. Dans ce mode, t_r est fixé à $1\mu s$.

La capacité de bus C_b est la somme des différentes capacités parasites présentes. Les circuits intégrés possèdent des capacités parasites qui, faute de données concrètes sur les datasheets, sont estimées à 100pF . La carte en compte trois (MCU + 2 capteurs), soit un total de 300pF . Il faut ajouter à cela les capacités de lignes qui dépendent de la longueur des pistes. Il serait possible de les calculer avec une formule plus aboutie mais le logiciel Saturn PCB [SATURN] et divers guides techniques fournissent une règle empirique de 0.6 à 0.8 pF/cm. Les longueurs des lignes SCL et SDA font respectivement 43mm et 44mm, ce qui donne une capacité de ligne de 3pF .

Ainsi, la capacité totale du bus est estimée à :

$$C_b = C_{\text{IC}} + C_{\text{lignes}} = 300 \times 10^{-12} + 6 \times 10^{-12} = 306 \text{ pF} \quad (3.9)$$

La résistance de pull-up maximale est donc :

$$R_p(\max) = \frac{1 \times 10^{-6}}{0.8473 \times 306 \times 10^{-12}} \approx 3.9 \text{ k}\Omega \quad (3.10)$$

La valeur de $10\text{k}\Omega$ a été choisie pour les résistances de pull-up, car elle est supérieure à la valeur maximale calculée et couramment utilisée dans les applications I2C. Ce raisonnement aura permis de s'assurer que les résistances de pull-up choisies respectent les spécifications du bus I2C.

USART

Le bus USART (Universal Synchronous Asynchronous Receiver Transmitter) est un protocole de communication série synchrone ou asynchrone. Bien qu'il ne soit pas utilisé dans le circuit, il a été décidé de le conserver pour permettre une communication série avec un appareil externe. De plus, il va être utile pour la programmation du microcontrôleur car il permet de transmettre des données entre le microcontrôleur et l'ordinateur.

3.4.8 STM32

Les ressources CAD du microcontrôleur ont été récupérées sur la page officielle du fabricant [MODULE]. Il est nécessaire d'utiliser Ultra Librarian, un logiciel permettant de télécharger des ressources CAD, pour transformer les fichiers téléchargés en composants Altium. La procédure est fournie dans le dossier téléchargé et sur le site internet [ULTRA LIB]. Le modèle 3D du microcontrôleur n'était pas présent dans le dossier, cependant il était disponible sur le loader SamacSys (uniquement le modèle 3D). Ce modèle a donc été téléchargé et ajouté au footprint correspondant (fichier .PcbLib) sur Altium en sélectionnant Place → 3D Body, puis en sélectionnant le modèle 3D.

Le câblage du microcontrôleur au reste du circuit s'est notamment fait en s'inspirant du schéma électrique du STM32WB5MM Discovery Kit [DISCO] et de la datasheet du STM32WB5MMG. Une fois que les signaux dont l'emplacement est défini ont été routés, il ne reste plus que les GPIO à placer. Celles-ci peuvent être placées sur n'importe quelle patte disponible, il a donc été choisi d'éviter au maximum les pattes présentes au centre du module, soit les ports D et E majoritairement. Cette étape s'est effectuée avec STM32CubeMX ouvert afin de pouvoir consulter leur disposition.

Le microcontrôleur, comme tout circuit intégré, doit recevoir une tension d'alimentation stable. Pour cela, il est impératif de placer des condensateurs au plus proche des différentes pattes d'alimentation du système. Cinq condensateurs ont donc été placés sur le schéma, quatre de 100nF pour assurer le découplage haute fréquence, et un de 10 μ F pour stabiliser la tension en agissant comme réservoir d'énergie à plus basse fréquence, notamment lors de transitoires de courant plus importants.

3.4.9 Estimation de la consommation

Il est crucial, lors de la conception d'un système embarqué, d'être capable d'estimer la consommation de courant. Cela permet de dimensionner correctement le système fournissant le courant et surtout d'estimer le temps de fonctionnement sur batterie. Cette étape ne peut cependant s'effectuer qu'après avoir imaginé le système dans sa globalité et idéalement lorsque les composants sont choisis afin de connaître leur propre consommation.

L'analyse se découpe en deux parties, la première concerne le fonctionnement sur pile (consommation minimale) et la seconde le fonctionnement lorsque le système est alimenté par fil. L'hypothèse pour le premier cas comprend un réveil du MCU de 10 secondes toutes les 10 minutes. Lors de cette courte phase active, le système effectue ses mesures analogiques, interroge ses capteurs puis communique ces résultats à l'unité centrale avant de se mettre en veille. Dans le second cas, le MCU ne s'endort pas complètement afin de pouvoir recevoir des commandes depuis l'unité centrale par communication sans fil. Un intervalle de 10 minutes est maintenu pour les mesures et on suppose que le flotteur, les GPIO, les leds et les relais MOSFET conduisent. Les valeurs de courant consommé proviennent des datasheets des différents composants.

3.4. SCHÉMA ÉLECTRIQUE

Alimentation sur pile

Composants	Courant maximal [μA]	Courant moyen sur 10 minutes [μA]
Mesure tension pile	40	< 1
Régulateur 3V3	40	< 1
MCU : BLE	9000	145
MCU : Core	5000	145
MCU : ADC 3 canaux	200	< 1
2 Interfaces sondes	3000	50
4 Load switchs	14.4	< 1
Capt température	320	4
Capt luminosité	45	< 1
Total	17'659.4	345

Table 3.2 – Estimation de la consommation sur pile

Plusieurs commentaires peuvent être émis sur ces premières valeurs.

- Les courants concernant le microcontrôleur dépendent énormément de la configuration sélectionnée, il s'agit là d'estimations probables mais qui peuvent varier selon les paramètres sélectionnés.
- Le courant maximal du MCU employant le BLE correspond au pic maximal, le courant nominal se situe davantage autour de 3 mA.
- Les courants moyens inférieurs à 1 μA sont négligeables par rapport à la consommation moyenne du microcontrôleur.
- La somme des courants maximaux permet de confirmer que n'importe quel régulateur de tension standard est capable de fournir suffisamment de courant au système lorsqu'il fonctionne sur pile, les LDO sortent généralement 100 mA au minimum et le convertisseur DC/DC sélectionné peut monter jusqu'à 800 mA.

La durée de fonctionnement du système sur pile peut dès lors être estimée. En supposant que la pile se compose de 3 piles AAA montées en série, dont la capacité moyenne est d'environ 1200 mAh, le nombre de jours d'autonomie D se calcule de la manière suivante :

$$D = \frac{C_{\text{pile}}}{I_{\text{moyen}}} = \frac{1200 \text{ mAh}}{345 \mu\text{A}} = \frac{1200 \text{ mAh}}{0,345 \text{ mA}} \approx 3478 \text{ heures} \approx 145 \text{ jours} \quad (3.11)$$

Ainsi, l'autonomie théorique est d'environ **145 jours**, soit un peu moins de **5 mois**. Cette valeur peut varier selon le choix des piles, il s'agit ici d'une valeur moyenne.

Alimentation filaire

Lorsque la carte est alimentée par câble, les éléments précédemment cités sont à conserver. Il n'est pas nécessaire de calculer ici le courant moyen, seul le courant maximal est intéressant afin de dimensionner correctement le régulateur de tension.

Composants	Courant maximal [mA]
Régulateur 3V3	0.04
MCU : BLE	9
MCU : Core	5
MCU : ADC 3 canaux	0.2
2 Interfaces sondes	3
4 Load switchs	0.014
Capt température	0.32
Capt luminosité	0.045
2 leds	20
2 relais MOSFET	20
Flotteur	0.066
5 GPIO	0.33
Total	58.02

Table 3.3 – Estimation de la consommation filaire

Le courant maximal estimé se trouve ainsi autour des **58mA**. Cette valeur est largement acceptable par n’importe quel régulateur de tension utilisé dans un système embarqué. Dans notre cas, ce dernier est un convertisseur *buck-boost* capable de fournir 800mA. Il n’y aura donc pas de problème de saturation de courant. Dans la configuration utilisée pour cette estimation, les différentes GPIO sont configurées avec des résistances de pull-up ou pull-down, dont la valeur est de $50\text{k}\Omega$, pour limiter la consommation de courant.

3.5 Conception du PCB

La conception débute lorsque le schéma électrique est en grande partie terminé et qu’il ne reste que certains points à ajuster en fonction du PCB. Parmi les éléments déterminés au cours de cette étape, on peut citer le dimensionnement des résistances de pull-up de l’I2C et les condensateurs de découplage des circuits intégrés. Le schéma électrique n’est donc pas fixé dès lors que l’on passe à l’étape suivante, il est au contraire amené à s’adapter aux contraintes qui s’ajoutent à ce stade.

3.5.1 Premières considérations

Le PCB doit, dans la mesure du possible, conserver le découpage du circuit établi sur le schéma électrique. Cela permet de simplifier le routage des pistes en réduisant la distance entre chaque composant et de rendre la carte plus lisible. Aucune contrainte de dimensions n’est établie par le cahier des charges, la logique va être de produire le design le plus compact possible, sans sacrifier les bonnes pratiques de conception. Les réflexions à propos de l’utilisation de l’appareil mènent en effet à penser qu’une carte peu encombrante facilitera le placement du système dans une serre. Cela permet également de limiter les perturbations possibles sur certaines lignes, comme les pistes analogiques.

3.5. CONCEPTION DU PCB

La carte comporte beaucoup de connecteurs, ces derniers doivent être placés sur les bords de la carte pour des raisons d'accessibilité. Certains composants, comme le module STM32, nécessitent un positionnement particulier sur la carte. Les fabricants recommandent parfois une certaine configuration des composants pour des performances améliorées, ces consignes doivent être suivies dans la limite du possible.

Le circuit comporte une partie analogique. Cette dernière est plus sensible aux perturbations que les autres, il faut donc veiller à l'éloigner le plus possible des gros éléments perturbateurs. Parmi les causes principales de perturbations, on retrouve les convertisseurs DC/DC chargés de fournir l'alimentation au circuit. Contrairement aux régulateurs linéaires, les convertisseurs DC/DC fonctionnent en commutant rapidement des transistors à haute fréquence, générant ainsi des signaux de commutation riches en harmoniques. Ces signaux peuvent provoquer des émissions électromagnétiques et des perturbations de conduction qui se propagent à travers le plan de masse et les pistes d'alimentation. Parmi les trois lignes analogiques, une partie de la partie *Power Supply*, il s'agit de la mesure de tension de la batterie. Cette piste est ainsi forcément plus exposée aux perturbations mais en consultant la datasheet du convertisseur DC/DC il apparaît que la fréquence d'oscillation de l'IC (1.25 MHz) est largement supérieure à celle de l'ADC, ce qui évite les désagrément.

Règles de conception et classe de PCB

La conception du PCB est supervisée par de nombreuses règles de conception qui assurent la fabrabilité, la fiabilité électrique et l'intégrité physique du circuit. Altium dispose d'un outil appelé *Design Rule Checker* (DRC), qui signale en temps réel tout dépassement de ces règles. Celles-ci peuvent être ajustées en fonction de la classe de précision du PCB que l'on souhaite atteindre. Naturellement, plus la classe est élevée, plus le coût de fabrication augmente. Il peut également analyser l'ensemble du design et rendre un rapport contenant toutes les violations qu'il faut corriger.

Pour déterminer la classe du PCB, une première approche, certes simplifiée mais rapide, consiste à consulter le site d'un fabricant de PCB proposant un calculateur de devis instantané. Dans notre cas, le fabricant PCBWay a été retenu. L'objectif est alors d'étudier l'impact de certains paramètres sur le prix, notamment le *Min track/spacing* (largeur minimale des pistes / espacement minimal entre elles) et le *Min hole size* (diamètre minimal des trous de perçage).

Bien que la classe de PCB ne soit pas explicitement indiquée par cet outil, il apparaît que certains paramètres permettent de rester dans une gamme de prix tout à fait raisonnable. Les valeurs suivantes ont ainsi été retenues.

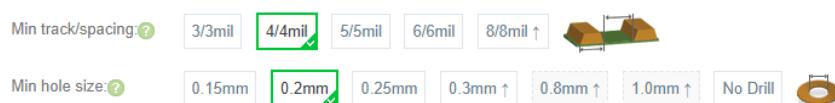


Figure 3.13 – Paramètres mimimum sur l'outil de devis de PCBWay

CHAPITRE 3. CONCEPTION HARDWARE

On retrouve ainsi un espacement et une largeur minimale de piste de 4/4 mil, soit environ **0,102 mm**, ainsi qu'un diamètre de perçage minimal de **0,2 mm**. Pour respecter ces paramètres, la largeur de piste minimale utilisée sur cette carte est fixée à **0,15 mm**, tout comme l'espacement minimal entre pistes. Une largeur plus fine pourrait être préférée si le routage l'imposait, mais il n'est pas bon de chercher les valeurs les plus petites car cela augmente entre autres l'impédance des pistes.

Le fabricant de PCB Eurocircuits fournit quant à lui une ressource pour la classification de PCB en fonction de leurs paramètres. Au regard des paramètres définis, la classe de PCB visée est la **7D**.

Eurocircuits - PCB Design Classification Overview												
Pattern Class	class 3		class 4		class 5		class 6		S + DZ + RF + SF		class 8	
	Service	mm	mil	mm	mil	mm	mil	mm	mil	mm	mil	mm
OTW	0.250	10	0.200	8	0.175	7	0.150	6	0.125	5	0.100	4
OTT-OTP-OPP	0.250	10	0.200	8	0.175	7	0.150	6	0.125	5	0.100	4
OAR	0.200	8	0.150	6	0.150	6	0.125	5	0.125	5	0.100	4
ITW	0.250	10	0.200	8	0.175	7	0.150	6	0.125	5	0.100	4
ITT-ITP-IPP	0.250	10	0.200	8	0.175	7	0.150	6	0.125	5	0.100	4
IAR	0.200	8	0.150	6	0.150	6	0.125	5	0.125	5	0.100	4
IPI	0.275	11	0.225	9	0.225	9	0.200	8	0.200	8	0.200	8

The smallest value (OTW, OTT-OTP-OPP, OAR, ITW, ITT-ITP-IPP, IAR, IPI) determines the Pattern Class of the board

Base Cu	min Pattern Values			
	OTT-OTP-OPP		OTW	
Base Cu OL	mm	mil	mm	mil
12µm	0.090	3.5	0.090	3.5
18µm	0.100	4	0.090	3.5
35µm	0.125	5	0.125	5
70µm	0.200	8	0.200	8
105µm	0.250	10	0.250	10

Base Cu IL	ITT-ITP-IPP		ITW	
	mm	mil	mm	mil
12µm	0.090	3.5	0.090	3.5
18µm	0.100	4	0.090	3.5
35µm	0.125	5	0.125	5
70µm	0.200	8	0.200	8
105µm	0.250	10	0.250	10

Diagram illustrating the relationship between hole size, outer annular ring (OAR), inner annular ring (IAR), and tool size. It shows a cross-section of a plated through hole with dimensions labeled: Finished Hole, OAR, TOOL SIZE, and IPI.

Preceding letters O and I stand for Outer- and Inner layer
Example: OTW = Outer layer Track Width

OAR : smallest OAR (Outer layer Annular Ring = 1/2 (Outer layer pad diameter - TOOLSIZE))
IAR : smallest IAR (Inner layer Annular Ring = 1/2 (Inner layer pad diameter - TOOLSIZE))

IPI (Inner layer Pad Insulation) : Clearance between edge TOOLSIZE of any unconnected hole(PTH/NPTH) and any nearest copper

Drill Class	class A		class B		class C		class D		class E		class F	
	Service	All Services	P + S + DZ + RF + SF + B	S + DZ + RF + SF	-	-	-	-	-	-	-	-
	mm	Inch	mm	Inch	mm	Inch	mm	Inch	mm	Inch	mm	Inch
PTH	0.50	0.020	0.35	0.014	0.25	0.010	0.15	0.006	0.10	0.004	<0.10	<0.004
NPTH	0.60	0.024	0.45	0.018	0.35	0.014	0.25	0.010	0.20	0.008	<0.20	<0.008
min TOOLSIZE	0.60	0.024	0.45	0.018	0.35	0.014	0.25	0.010	0.20	0.008	<0.20	<0.008

NOTE: The smallest value (TOOLSIZE) determines the Drill Class of the PCB

Max. PCB Thickness to Drill Class	mm	Inch	mm	Inch	mm	Inch	mm	Inch	mm	Inch
	3.20	0.125	3.20	0.125	2.40	0.093	2.00	0.079	1.60	0.062
	Aspect Ratio is 1:8 (Based on the TOOLSIZE)									

Note A: VIA holes are Plated Through Holes, default defined as <0.45mm (18mil) for all services or <= as defined by the customer in the order details.
VIA holes have a maximum negative tolerance of 0.30mm (12mil)
Note B: This classification table can only be put into praxis on PCB designs that have a Plating Index of 0.40 or higher. This is calculated in the PCB Visualizer analysis and displayed in the PCB Visualizer order details.

Services Index : P = PCB proto S = STANDARD pool DZ = DEFINED IMPEDANCE RF = RF pool SF = SEMI-FLEX pool I = IMS pool B = BINDI pool

eC-classification Table - 07/05/2024

Figure 3.14 – Table de classification Eurocircuits

Fixation mécanique

Des pads de fixation mécanique ont été prévus au nombre de trois pour la carte. Ayant été placés après les composants électroniques, il ne restait que peu de place, ce qui a mené à choisir la taille **M2.5**.

3.5.2 Placement des composants

Pour placer les composants, il faut se trouver dans le fichier *.PcbDoc* et importer les changements depuis le schéma. Pour ce faire, rendez-vous dans l'onglet *Design → Import Changes From Rainette.PjrPcb*. La fenêtre de l'*Engineering Change Order* s'ouvre alors et il est possible de voir les différences existantes entre le schéma et le PCB (composants, numérotations, etc.). Si aucune erreur n'est présente, il est ensuite possible de valider les changements proposés (ou de les modifier) en

3.5. CONCEPTION DU PCB

cliquant sur *Validate Changes*. Les changements validés peuvent finalement être exécutés en cliquant sur *Execute Changes*. Le projet Altium du Discovery Kit STM32WB5MMG-DK a été récupéré pour s'inspirer des techniques mises en place pour le placement et le routage. Plus complexe que ce projet car utilisant la quasi totalité des pattes du STM32WB5MMG, il utilise des solutions plus chères qui peuvent être évitées dans ce projet (comme les vias *Blind*, ne traversant pas toute la carte).

Recommandation de layout

Les composants se retrouvent donc sur le fichier *.PcbDoc*, il est à présent temps de les disposer sur le PCB. Le fichier contient par défaut une carte de 150x100 mm, elle est amenée à être modifiée mais sert de point de départ pour le placement. Le module STM32, de par son importance et son nombre de pattes élevé, est un élément dont le positionnement doit être rapidement planifié. Or, son antenne embarquée implique de le disposer selon la figure suivante, extraite du datasheet.

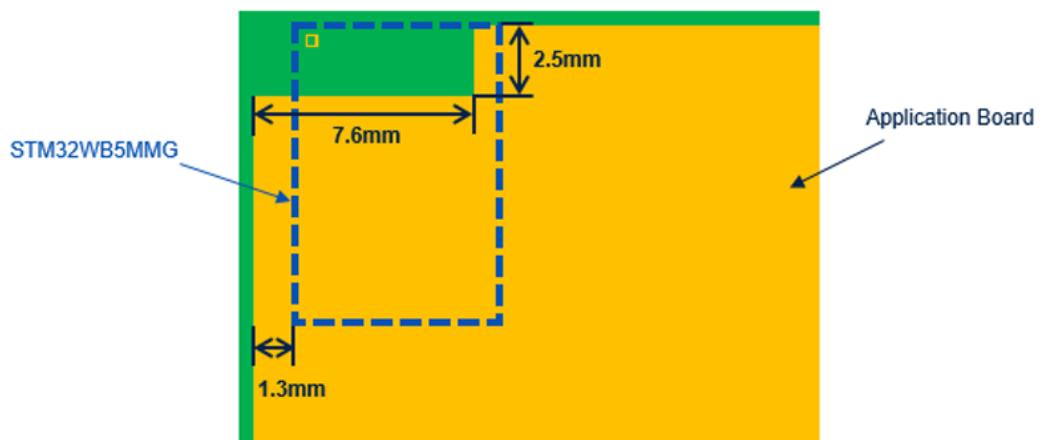


Figure 3.15 – Positionnement du module STM32 sur la carte

Il est donc recommandé de le placer dans le bord supérieur gauche du PCB (surface verte) et également de ne placer ni pistes ni plan de masse sous la surface de l'antenne. La fiche technique précise que le non-respect de cette recommandation n'empêche pas le fonctionnement des communications sans fil, mais entraîne une dégradation des performances. Ce placement dans un coin complexifie le routage car le module ne peut donc pas être placé au centre de la carte et il devient nécessaire de superposer des signaux sur plusieurs couches.

Cette contrainte importante force à débuter le placement des composants avec le module STM32. Il est avantageux d'éviter au maximum l'emploi de pattes se trouvant au centre du composant, cela permet de ne pas employer de vias placés directement sur le pad (*Via in Pad*), ce qui évite une tarification supérieure pour la confection des PCB. La disposition des différents groupes de composants est influencée par l'emplacement des pattes du module STM32 auquel ils sont reliés. Les signaux analogiques ainsi que les éléments comportant beaucoup de connexions avec le module STM32 sont à placer proches de ce dernier.

CHAPITRE 3. CONCEPTION HARDWARE

Comme précédemment indiqué, l'interface SPI était initialement prévue sur le schéma, des headers devaient être placés pour permettre la communication avec un appareil externe. N'étant pas fondamentale et nécessitant le routage de 4 pistes, elle a été écartée du schéma pour simplifier le placement des composants. L'interface USART a été conservée pour permettre d'afficher du texte sur une console.

Une forte densité de pistes au niveau du module STM32 implique l'utilisation de vias. Il est possible sur Altium de concevoir des vias dans une bibliothèque dédiée nommée *PuLib1.Lib*. En se rendant sous l'onglet *Panels* puis en cliquant sur *Pad Via Library*, on peut ajouter un nouveau via et définir ses caractéristiques comme le diamètre extérieur et la taille du trou. Après discussion avec le professeur encadrant, l'emploi de vias de diamètre 0.45 mm et avec un trou de 0.2 mm (v45h20) a été validé. Ces dimensions sont conformes à la classe de PCB désirée et très proches des dimensions du layout STM32WB5MMG-DK (v40h20).

Le datasheet du STM32WB5MMG fournit certaines recommandations pour le layout du composant. Il indique notamment qu'un plan de masse doit être présent sur la couche supérieure de la carte autour du microcontrôleur. Il doit par ailleurs s'étendre sur un grand espace à droite du MCU, en incluant des vias de stitching. Les signaux présents à droite du microcontrôleur doivent donc rapidement être déplacés sur une couche inférieure à l'aide de vias.

Disposition sur la carte

Le placement des différentes parties du système est donc influencé par l'importante contrainte qu'est l'emplacement du microcontrôleur. Plusieurs décisions se sont alors imposées pour parvenir à router le circuit de la manière la plus optimisée possible.

- Les parties du schéma contenant beaucoup de liaisons avec le microcontrôleur sont placées au plus proche de ce dernier.
- Les composants de la partie *Power Supply* étant nombreux et prenant donc beaucoup de place, ils sont placés dans le coin inférieur droit. Cela simplifie le routage et éloigne tant bien que mal cette source de perturbations des parties sensibles, comme la zone analogique.
- Les grands connecteurs ont été placés au maximum sur les bords de la carte. La surface de cette dernière a été réduite au mieux pour obtenir un faible encombrement.
- Les pattes du microcontrôleur reliées aux connecteurs GPIO et au flotteur se trouvaient proches de l'antenne, cela a influencé le placement des éléments mentionnés sur le haut de la carte.

3.5. CONCEPTION DU PCB

Il en résulte une disposition décrite sur la figure ci-dessous.

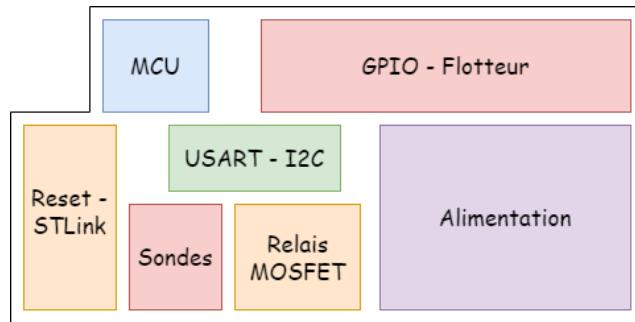


Figure 3.16 – Disposition des composants sur la carte

Les dimensions de la carte tombent ainsi à **91.9 x 44.9mm**. Compte tenu de l'encombrement de la carte, il est difficile d'obtenir une surface plus petite. La décision a été prise de placer le maximum de composants sur la face supérieure de la carte. Cela permet de la fixer plus facilement dans un boîtier par la suite. La modification de la forme du PCB se fait en passant dans la vue *Board Planning Mode*, puis en utilisant les outils de modification dans l'onglet *Design*.

3.5.3 Routage et plans de masse

Le routage autour du microcontrôleur a nécessité l'emploi de pistes avec largeur minimale, soit 0.15 mm. Un plan de masse autour du module STM32 étant recommandé, il a été décidé de l'étendre sur toute la couche supérieure de la carte. Cela comporte plusieurs avantages, notamment une meilleure réduction des interférences électromagnétiques, un retour de courant optimisé et une structure plus simple pour le routage du GND. Le circuit comporte ainsi un plan de masse sur deux couches, il est indispensable d'égaliser au mieux leur potentiels en réalisant du *Via stitching*. Cette étape consiste à placer des vias sur l'ensemble de la carte pour relier chaque partie des plans de masse. Les vias ne doivent pas contenir de freins thermiques, ce qui ajouterait de l'inductance. Leur espacement doit se situer à 2 mm sur la droite du module selon le datasheet, ils ont été placés manuellement à cet endroit. Pour le reste de la carte, l'outil de placement automatique de vias a été utilisé (*Tools → Via Stitching/Shielding → Add Stitching to Net*) et un espacement de 4 mm a été fixé. L'ensemble des vias de stitching ont été recouverts de masque épargne soudure pour ne pas laisser le cuivre apparent.

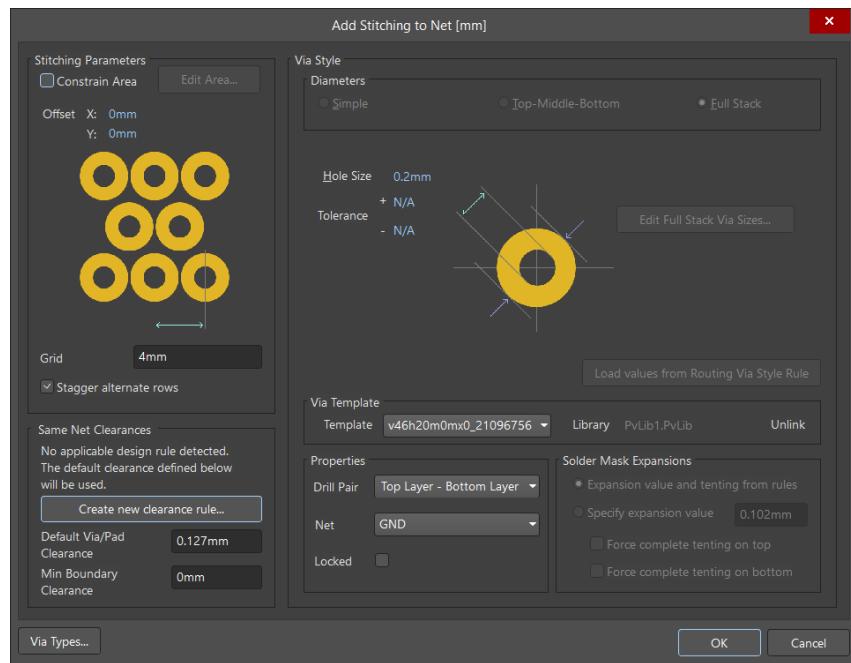


Figure 3.17 – Configuration du *Via stitching* sur le plan de masse

Les pattes d'alimentation 3V3 sont elles plus épaisses, elles vont d'une largeur de 0.3 mm lorsque l'espace est resserré à 0.6 mm dans la zone *Power Supply* car des courants plus importants peuvent y circuler.

3.5. CONCEPTION DU PCB

L'ensemble des signaux a été routé sur les couches externes de la carte, afin de perturber au minimum les plans de masse (*GND*) et d'alimentation (3,3 V) présents sur les couches internes. Plus généralement, une attention particulière a été portée à la gestion des retours de courant, en veillant à ce que les signaux disposent d'un chemin de retour direct et continu. Cette approche permet de limiter les boucles de courant et d'assurer un bon comportement en compatibilité électromagnétique. Les couches supérieure et inférieure du circuit, où circulent les signaux sont illustrées sur les figures ci-dessous, le plan de masse est retiré pour plus de lisibilité.

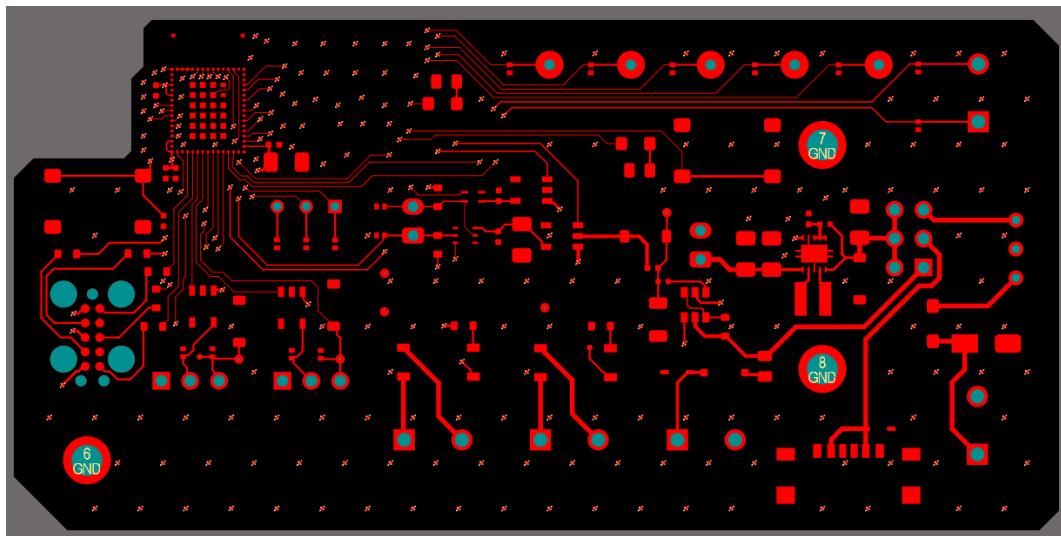


Figure 3.18 – Couche Top de la carte sans plan de masse

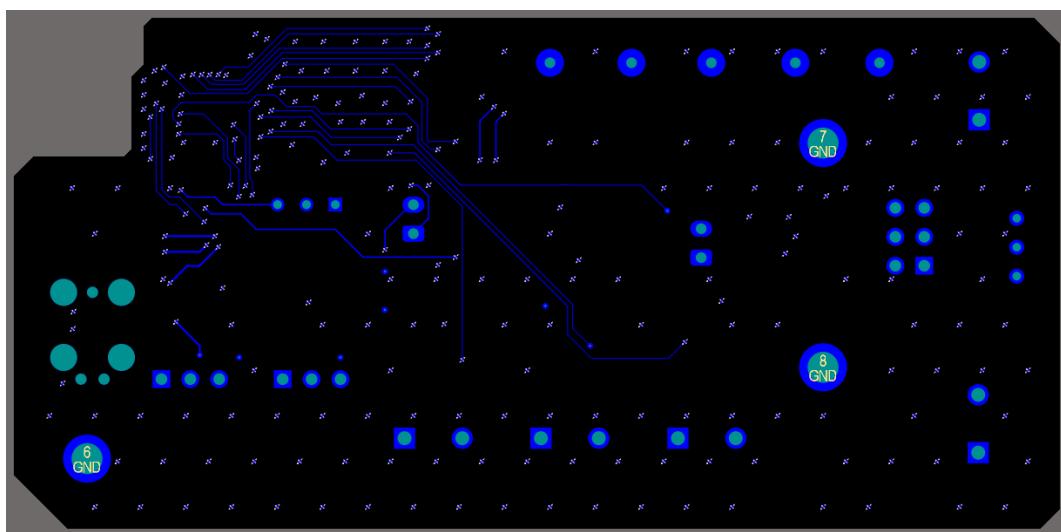


Figure 3.19 – Couche Bottom de la carte

Il est aisément de reconnaître la zone du microcontrôleur dans le coin supérieur gauche, avec cette multitude de pistes 0.15 mm. La partie *Power Supply* se devine elle sur dans le coin inférieur droit car les pistes y sont plus épaisses. Les composants étant placés sur la surface supérieure, le routage des pistes y a également été privilégié. La

CHAPITRE 3. CONCEPTION HARDWARE

couche inférieure est donc surtout présente pour les zones denses où la seule couche *Top* ne suffirait pas. L'ensemble des pads de composants, présents en rouge sur la couche *Top* et n'étant pas rattaché à une piste, est connecté au plan de masse avec un frein thermique.

La figure suivante illustre la vue 3D du circuit.

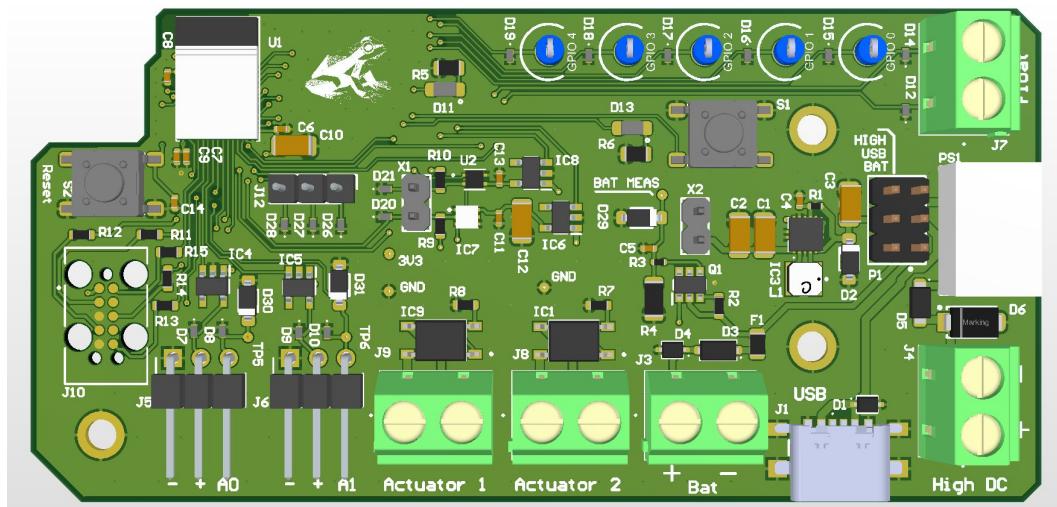


Figure 3.20 – Vue 3D de la carte

3.5.4 Vérification et exportation des fichiers Gerber

Certaines règles ont été changées manuellement pour correspondre à la classe de PCB *7D*, il manquait cependant certaines spécifications, c'est pourquoi il a été décidé d'importer un fichier de règles *.RUL* depuis Eurocircuits. Il est en effet possible de leur soumettre un PCB pour recevoir les règles à respecter selon la classe correspondante. Ces règles peuvent également être fournies en sélectionnant la classe désirée. Ce fichier de règles est ensuite fourni au DRC afin de pouvoir procéder à une analyse complète du layout.

En exécutant le *Design Rules Check*, un rapport contenant toutes les violations de règles est généré. Si la majeure partie de ces violations doit être corrigées, certaines n'ont aucune influence sur la réalisation de la carte et sur sa fiabilité. Ainsi, 3 erreurs relevées par le DRC ont délibérément été laissées.

3.5. CONCEPTION DU PCB

Minimum Solder Mask Sliver

102 occurrences Largeur minimum d'une fine couche de masque épargne. Concerne le module STM32 dont l'espacement entre les pattes (0.15mm) est plus fin que la valeur du fichier de règles (0.254mm). Un échange avec le fabricant de PCB PCBWay a permis de déterminer qu'ils sont en mesure de placer une couche de masque épargne d'une largeur de 0.19mm. Ainsi, le footprint du composant a été légèrement modifié en étant le masque épargne de 0.02mm de plus par côté sur les pads afin de coller à cette contrainte. Cette décision n'a pas lieu d'affecter la brasure du composant au vu de la modification minime apportée.

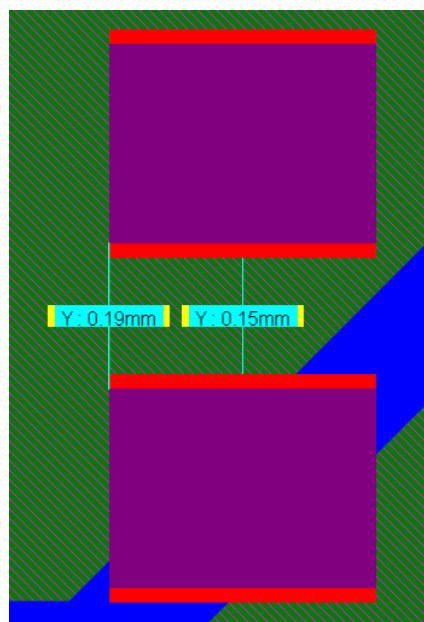


Figure 3.21 – Modification du Solder Mask du footprint

Silk To Solder Mask

126 occurrences Sérigraphie sur le masque épargne. Ce chevauchement n'entraîne pas de complications et peut être laissé comme tel.

Silk To Silk

22 occurrences Sérigraphies chevauchées, cette erreur est purement esthétique et peut également être ignorée.

Lorsque les erreurs importantes ont été corrigées, le PCB peut être transformé en fichiers Gerber afin de le faire vérifier par le fabricant. Pour cela, le fichier *Pour_Assemblage.OutJob* a été récupéré de la part des membres du laboratoire, il s'agit d'un fichier automatisant l'exportation d'éléments depuis le projet. Le fichier en question contient les bons réglages pour générer les fichiers Gerber.

CHAPITRE 3. CONCEPTION HARDWARE

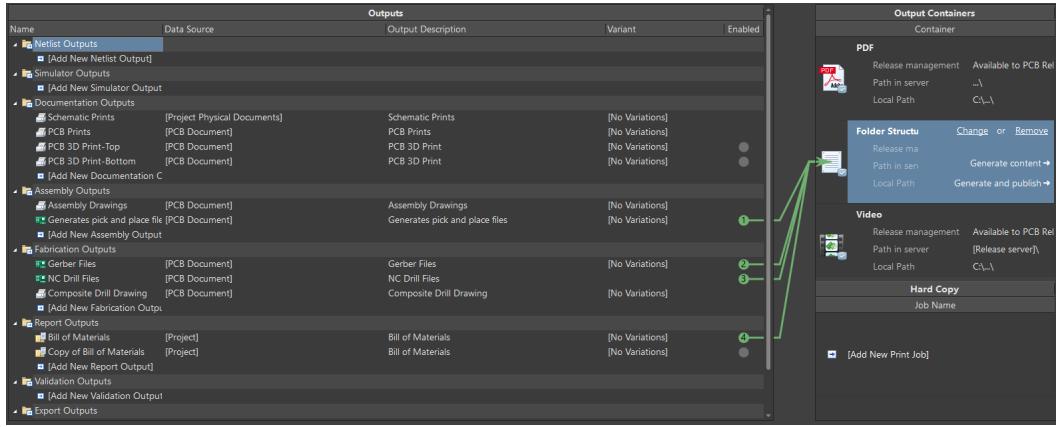


Figure 3.22 – Aperçu du Pour_Assemblage.OutJob

Parmi les fichiers générés se trouvent la liste des composants (*BOM*), cette dernière est notamment utile pour renseigner le nom des composants donné par le fabricant (*Manufacturer Name*) ainsi qu'au moins un nom de fournisseur l'ayant en stock. Les composants récupérés sur la bibliothèque de composants de l'école ne contenant pas ces informations selon le bon format, la *BOM* a été éditée manuellement par la suite afin de remplir les informations manquantes.

3.5.5 Commande et coûts totaux

Après avoir soumis la demande d'assemblage, il apparaît que certains composants ont un délai d'approvisionnement compris entre 7 et 10 jours. Le temps étant très restreint pour ce projet, il a été décidé de ne pas faire assembler ces quelques composants et de les monter sur place lorsque la carte sera arrivée. Cinq PCB ont été commandés (quantité minimale), dont trois assemblés directement chez le fabricant. Le devis est renseigné ci-dessous.

Component Cost	\$115.66
Assembly Cost	\$29.00
PCB Cost	\$25.97
All Total 3 Sets	\$170.63

Table 3.4 – Devis PCBWay

Les coûts de PCB et d'assemblage sont tout à fait raisonnables mais ceux des composants sont relativement élevés. Cela est principalement dû au prix des modules STM32WB5MMG trouvés par PCBWay qui s'élèvent à \$15.90 contre 10.10 CHF si commandé chez Mouser au moment de la commande. S'agissant d'un élément critique, il a été choisi de le laisser être monté chez PCBWay au détriment du prix. Si ce projet venait à être produit en plus grandes séries, il serait nécessaire de revoir la façon dont sont assemblées les cartes.

3.5. CONCEPTION DU PCB

Pour compléter le coût du projet, il faut également prendre en compte le coût des composants à assembler sur place qui s'élève à 15.25 CHF. Un kit Atlas Scientific pour la mesure de température ainsi que le câble de programmation du ST-Link ont également été commandés pour un coût de 62.95 CHF. Enfin, on peut également compter les deux kits de développement STM32WB5MM-DK commandés en début de projet, pour un prix de 94.20 CHF.

Ainsi, le coût total du projet (aucun frais n'étant prévu pour la suite du projet) est calculé à **312.80 CHF** hors frais de livraison.

Chapitre 4

Développement logiciel

Cette étape, composante principale de la seconde phase du projet, s'est en grande partie déroulée lors de la confection et de l'assemblage de la carte électronique. Ainsi, c'est principalement sur le kit de développement STM32WB5MM-DK qu'ont été testées les différentes fonctionnalités de Mbed OS. Différents codes ont ainsi été développés, le programme de la carte STM32WB5MM-DK (renommée *Disco*), celui de la carte *Rainette* et les codes associés à ces cartes pour le Raspberry Pi (même structure mais adaptés pour leurs caractéristiques propres).

4.1 Système d'exploitation

Pour débuter le développement du code, il est important de comprendre le fonctionnement du support sur lequel on travaille. Pour ce projet, la programmation en *bare metal*, c'est-à-dire sans système d'exploitation, n'est pas retenue car la carte comporte beaucoup de périphériques. L'usage d'un système d'exploitation (Operating System ou OS) est ainsi préféré. Il s'agit d'une couche logicielle qui permet d'abstraire le matériel sous-jacent et de gérer de manière efficace les ressources du microcontrôleur (MCU).

Dans le cas de ce projet, un OS temps réel est utilisé pour simplifier le développement : **Mbed OS**. Il offre des services comme la gestion de tâches concurrentes, la synchronisation entre ces tâches, la gestion de la mémoire dynamique, ainsi que l'accès aux périphériques matériels via des API bien définies. Il s'agit d'une solution pertinente pour le développement IoT où la consommation est cruciale, avec une intégration tout-en-un (RTOS + middleware + HAL + sécurité + réseau). Son API, conçue en C++, est axée Programmation Orientée Objet (POO), ce qui facilite le développement et le prototypage rapide. Développé par Arm Ltd., Mbed OS est compatible avec une large gamme de microcontrôleurs et le portage du code vers un autre support est simplifié par sa configuration matérielle standardisée.

La qualité première de cet OS temps réel est de pouvoir ordonner des tâches concurrentes, c'est-à-dire gérer leur exécution en assurant le partage équitable et déterministe des ressources processeur, selon des priorités définies, des contraintes de temps et des mécanismes de synchronisation. Les éléments principaux qui permettent ce comportement et qui sont employés dans ce projet sont décrits ci-dessous. Il s'agit d'une liste non exhaustive.

Scheduler

Le scheduler de Mbed OS gère l'ordonnancement des threads. Il détermine quel thread doit s'exécuter en fonction de sa priorité et de son état, assurant ainsi une utilisation optimale du processeur. Ce comportement résulte d'un mécanisme algorithmique intégré au noyau, qui ne se manifeste pas sous la forme d'une entité distincte dans le code utilisateur, mais agit comme une logique centrale du système d'exécution.

Thread

Un thread (tâche) est une unité d'exécution au sein d'un processus. Dans Mbed OS, les threads partagent l'espace mémoire du processus parent et peuvent être créés, synchronisés et terminés dynamiquement pour exécuter des tâches concurrentes. Une fois démarré, le thread exécute une fonction qui possède sa propre boucle infinie, permettant ainsi à plusieurs processus de s'exécuter en parallèle, la boucle principale du code étant considérée comme un thread.

```
// Déclaration d'un thread et de deux leds
Thread thread1;
DigitalOut led1(LED1);
DigitalOut led2(LED2);

// Fonction exécutée par le thread
void clignoter_led1() {
    while (true) {
        led1 = !led1; // Inversion de l'état de la led
        ThisThread::sleep_for(500ms); // Attente de 500ms
    }
}

int main() {
    // Démarrage du thread avec la fonction clignoter_led1
    thread1.start(clignoter_led1);

    // Boucle principale : autre tâche concurrente
    while (true) {
        led2 = !led2;
        ThisThread::sleep_for(1000ms); // Attente de 1s
    }
}
```

Listing 4.1 – Exemple de thread avec deux leds clignotant à des fréquences différentes

4.1. SYSTÈME D'EXPLOITATION

EventQueue

Une eventQueue est une structure de gestion d'événements différés dans Mbed OS. Elle permet de planifier l'exécution de fonctions ou de callbacks en les plaçant dans une file d'attente traitée séquentiellement. Les événements peuvent être déclenchés immédiatement ou différés dans le temps. L'eventQueue est généralement utilisée pour déléguer des traitements depuis des ISR vers un contexte thread, évitant ainsi d'exécuter des opérations longues dans une interruption.

```
InterruptIn button(BUTTON1);
EventQueue queue;
Thread queue_thread;

void on_button_pressed() {
    printf("Bouton pressé (traitement dans le thread)\n");
}

void isr() {
    // Ne pas exécuter printf directement ici (ISR) !
    queue.call(on_button_pressed); // Planifie dans EventQueue
}

int main() {
    // Démarrage de la file d'événements
    queue_thread.start(callback(&queue, &EventQueue::dispatch_forever));

    // Attache l'interruption sur le bouton
    button.fall(&isr);

    while (true) {
        ThisThread::sleep_for(1s); // Boucle principale inactive
    }
}
```

Listing 4.2 – Exemple de eventQueue avec appel différé depuis une ISR

EventFlags

Un eventFlags est un mécanisme de synchronisation basé sur des drapeaux binaires, permettant à plusieurs threads de communiquer ou de se synchroniser autour d'un ensemble d'événements. Dans Mbed OS, un thread peut attendre qu'un ou plusieurs bits soient définis, de manière bloquante ou avec un timeout.

```
EventFlags flags;
Thread worker;

#define FLAG_READY 0x01

void thread_function() {
    // Attente du drapeau READY
    flags.wait_any(FLAG_READY);
    printf("Drapeau reçu, traitement lancé\n");
}

int main() {
    worker.start(thread_function);

    // Simulation d'un traitement
    ThisThread::sleep_for(500ms);

    // Signale l'événement au thread
    flags.set(FLAG_READY);

    while (true) {
        ThisThread::sleep_for(1s);
    }
}
```

Listing 4.3 – Exemple de eventFlags avec synchronisation entre deux threads

Il existe encore de nombreux mécanismes non listés. Mbed OS est une bonne solution, très bien documentée, mais dont le développement actif par Arm a pris fin. La version finale devrait être publiée en juillet 2026, suite à quoi les mises à jour ne seront plus effectuées. Une version communautaire a vu le jour mais, il sera plus pertinent à l'avenir de se diriger vers d'autres OS semblables, comme FreeRTOS ou Zephyr. Mbed OS a été choisi car il a enseigné en cours de *Conception de Systèmes Embarqués* et était employé dans l'institut *IICT*.

4.2 Fonctionnement global du programme

4.2.1 Organigramme

Le programme de la carte, bien que peu complexe dans son ensemble, nécessite tout de même d'être réfléchi sur papier avant son développement. Cela permet d'avoir un fil conducteur lors de la rédaction du code, car il est facile de ne pas prévoir une situation sans plan initial.

4.2. FONCTIONNEMENT GLOBAL DU PROGRAMME

Après son initialisation, il est impératif de déterminer dans quel cas de figure se trouve la carte : alimentée par pile ou par une source filaire. La présence de la pile sert ainsi à répondre à cette interrogation. Si la pile est présente, la carte doit consommer le moins possible. Elle ne reçoit donc pas de commandes de la part du Raspberry Pi, se contentant uniquement de publier ses données à intervalles réguliers. La communication Bluetooth est activée uniquement lors de la publication.

Si aucune tension n'est mesurée au niveau de la pile, la consommation n'est plus jugée critique et une communication permanente est mise en place avec l'unité centrale. La carte conserve sa publication régulière, mais elle peut également recevoir des commandes. Pour s'assurer de la connexion BLE, un test est prévu chaque minute.

L'organigramme qui découle de ce raisonnement est illustré ci-dessous.

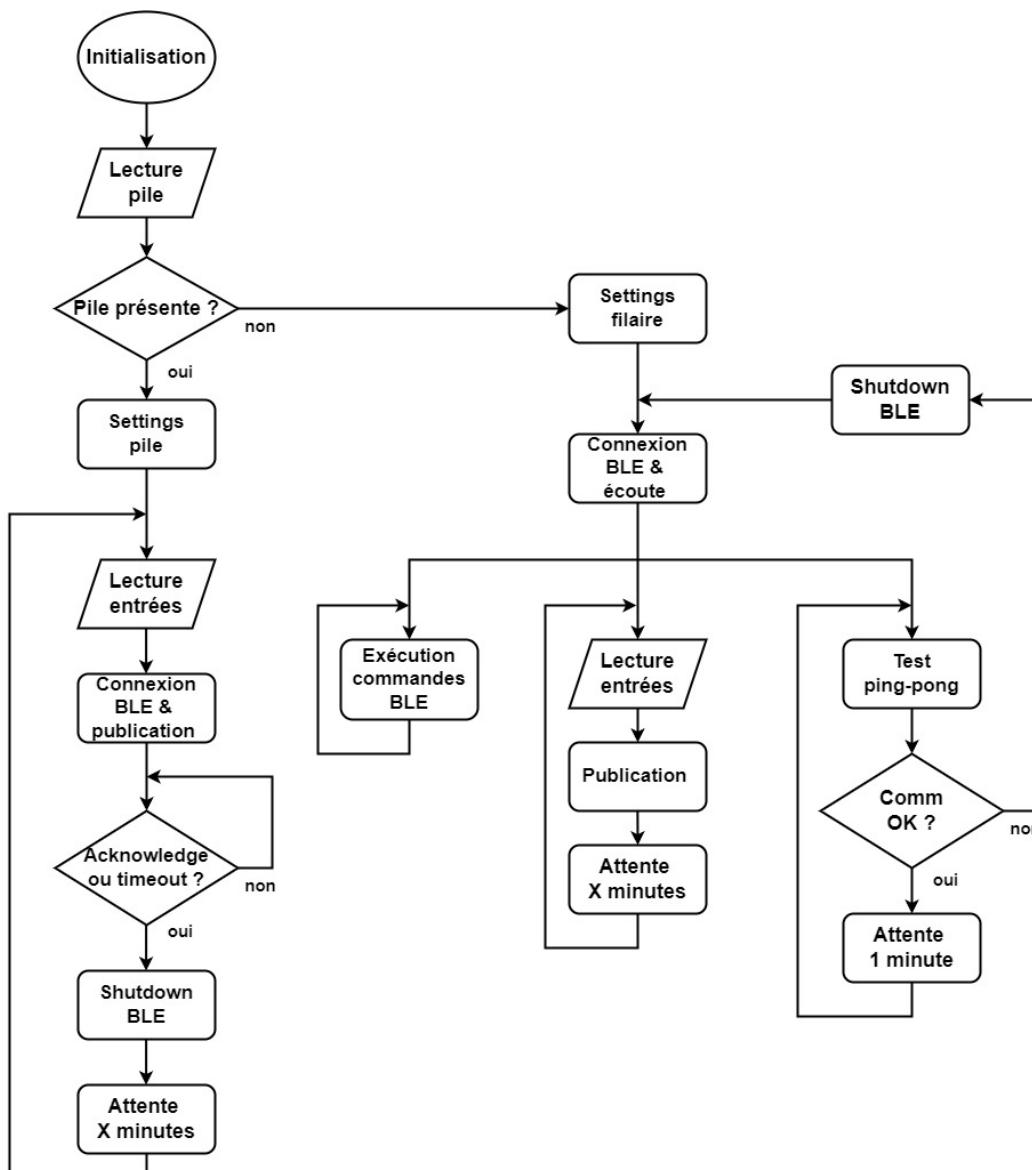


Figure 4.1 – Organigramme du programme sur la carte Rainette

CHAPITRE 4. DÉVELOPPEMENT LOGICIEL

Avec le fonctionnement filaire, trois boucles apparaissent en parallèle. Il s'agit d'un cas rendu possible grâce aux outils proposés par Mbed OS, comme les threads ou les eventQueues. En cas de *reset* de la pile BLE, les trois processus sont interrompus pour être relancés ensemble.

Cet organigramme ne comprend pas l'ensemble des éléments mis en place dans le code (comme les spécificités propres au Bluetooth) mais sert de fil conducteur pour l'implémentation finale. La rubrique suivante détaille le fonctionnement et les caractéristiques du BLE.

4.2.2 *Bluetooth Low Energy*

Le Bluetooth Low Energy (BLE) est un protocole de communication radio caractérisé par une architecture orientée événements. Plusieurs concepts permettent de décrire son fonctionnement.

Architecture Serveur/Client

BLE fonctionne selon une architecture client/serveur :

- Le **serveur** BLE (souvent l'objet connecté) expose des données via une interface normalisée.
- Le **client** BLE (typiquement un Raspberry Pi ou un smartphone) interroge ou souscrit à ces données.

Dans le cadre de ce projet, c'est la carte Rainette qui fait office de serveur, publiant différents services que le Raspberry Pi peut ensuite venir consulter et modifier.

Spécification GATT

Le BLE repose sur la couche GATT (Generic Attribute Profile) pour organiser les données échangées selon une structure hiérarchique. L'arborescence d'attributs est organisée comme suit.

A. Services

Un service est un regroupement de fonctionnalités ou de données relatives à une tâche, pouvant contenir une ou plusieurs caractéristiques. Il existe des services standardisés par la norme Bluetooth mais il est évidemment possible de créer ses propres services personnalisés.

B. Caractéristiques

Une caractéristique est une valeur de donnée exposée par le serveur. Elle contient une valeur principale (ex. : température = 27.3°C) et peut contenir un ou plusieurs descripteurs.

C. Descripteurs

Les descripteurs sont des métadonnées associées à une caractéristique, elles peuvent, par exemple, permettre au client de s'abonner aux notifications ou indications ou contenir une description de la caractéristique. Mbed OS simplifie cependant l'utilisation du BLE et ne fournit pas d'interface directe pour déclarer des descripteurs.

4.3. PARAMÉTRAGE

Spécification GAP

Le GAP (Generic Access Profile) est l'un des profils fondamentaux du protocole Bluetooth. Il définit comment les appareils BLE découvrent, se connectent et interagissent. Ses principaux mécanismes sont les suivants :

Advertising

Messages courts envoyés par les périphériques pour signaler leur présence.

Scanning

Processus d'écoute des paquets d'advertising.

Initiating

Initialisation de la connexion après la découverte d'un périphérique.

UUID et propriétés

Les UUID sont des identifiants uniques exprimés sur 128 bits, utilisés pour distinguer les services, caractéristiques et descripteurs. Le Bluetooth SIG (Bluetooth Special Interest Group) propose une série de UUID standards composés de 16 bits (ex. : 0x180A). Pour obtenir le format 128 bits de ces UUID, il faut les insérer dans la chaîne hexadécimale *0000xxxx-0000-1000-8000-00805f9b34fb*, à la place des *x*.

Ces UUID permettent d'identifier des services interprétables par n'importe quel appareil. Pour un usage personnalisé, il est possible de composer son propre UUID 128 bits.

Les propriétés sont quant à elles rattachées aux caractéristiques et définissent les droits d'accès du client sur ces dernières.

read

Permet au client d'obtenir la valeur courante.

write

Permet au client de modifier la valeur.

notify

Le serveur envoie automatiquement une mise à jour sans accusé de réception.

indicate

Le serveur envoie automatiquement une mise à jour avec accusé de réception.

Ces quelques explications permettent de mieux cerner la structure du code ainsi que les différents procédés mis en place pour atteindre un résultat fonctionnel.

4.3 Paramétrage

4.3.1 Paramètres du projet Mbed OS

L'IDE choisi est *VSCCode*, éditeur multiplateforme, léger et extensible, conçu par Microsoft. L'environnement ajouté à cet IDE pour faciliter le développement sur Mbed OS a été développé et proposé par un ingénieur de l'institut. Le projet Mbed OS vierge contient plusieurs fichiers et dossiers importants.

Fichier mbed_app.json

Le fichier *mbed_app.json* est un fichier de configuration JSON utilisé par Mbed OS pour définir des paramètres de compilation, tels que les macros, les configurations de modules ou les options spécifiques aux cibles matérielles. Il permet de personnaliser

le comportement d'une application sans modifier le code source, en injectant des valeurs dans l'arbre de configuration de Mbed lors de la compilation.

```
{
  "macros": ["MY_CUSTOM_MACRO=1",
             "MBED_TICKLESS"],
  "target_overrides": {
    "*": {
      "mbed-trace.enable": true,
      "mbed-trace.max-level": "TRACE_LEVEL_DEBUG",
      "platform.stdio-convert-newlines": false,
      "platform.stdio-baud-rate": 115200,
      "platform.stdio-buffered-serial": false,
      "cordio.trace-hci-packets": false,
      "cordio.trace-cordio-wsf-traces": false,
      "ble.trace-human-readable Enums": false,

      "target.printf_lib": "minimal printf",
      "platform.minimal printf-enable-floating-point": true,
      "platform.minimal printf-set-floating-point-max-decimals": 6,
      "platform.minimal printf-enable-64-bit": false
    },
    "DISCO_WB5MMG": {
      "target.mbed_rom_size": "0xE1000"
    },
    "RAINETTE": {
      "target.mbed_rom_size": "0xE1000"
    }
  }
}
```

Listing 4.4 – Fichier mbed_app.json

Dossier targets

Le dossier *targets*, lui-même contenu dans le dossier *mbed-os*, contient les définitions spécifiques aux plateformes matérielles (cibles), incluant les fichiers de configuration, les couches d'abstraction matérielle (HAL), les descriptions des broches (PinNames.h), ainsi que les fichiers source propres au microcontrôleur ou à la carte. Il permet d'adapter le système aux différentes architectures et variantes matérielles supportées, en structurant le support des cibles de façon modulaire.

4.3.2 Paramètres du BLE

Les concepts fondamentaux du BLE ayant été détaillés précédemment, il est à présent possible de débuter l'implémentation sur la carte. Pour ce faire, on trouve un guide sur le dépôt GitHub de Mbed OS des targets STM32WB [TARGETS]. Ce guide attire l'attention sur le coprocesseur M0 et la version de son firmware.

Comme le précise la fiche technique du STM32WB5MMG, il contient deux coeurs afin d'augmenter sa capacité à gérer plusieurs mécanismes. Le coeur secondaire Cortex-M0+ prend en charge la gestion de la pile BLE (ensemble des couches logicielles). Ce coprocesseur possède son propre firmware, dont les différentes versions sont renseignées ci-après.

4.3. PARAMÉTRAGE

Link between wireless coprocessor binaries and Bluetooth® LE stack variant		
Wireless coprocessor binary	Variant of the Bluetooth® LE stack	Binary size v1.16.0
stm32wb5x_BLE_Stack_full_extended_fw.bin	-	184 KB
stm32wb5x_BLE_Stack_full_fw.bin	BF = Basic Features	148 KB
stm32wb5x_BLE_Stack_light_fw.bin	PO = Peripheral Only	112 KB
stm32wb5x_BLE_HCILayer_extended_fw.bin	LO = Link Layer Only	100 KB
stm32wb5x_BLE_HCILayer_fw.bin	LB = Link Layer Only Basic	78 KB
stm32wb5x_BLE_HCI_AdvScan_fw.bin	BO = Beacon Only	35 KB

Figure 4.2 – Tableaux des différentes versions de FW du coprocesseur M0

Le guide indique que la version installée par défaut est la *stm32wb5x_BLE_Stack_full_fw.bin* et indique que pour des raisons d'optimisation de la mémoire, la version *stm32wb5x_BLE_HCILayer_fw.bin* est conseillée. Un test effectué motive ce changement de version. En testant un code d'exemple BLE [BLE EX], le log d'erreur suivant apparaît.

```
[INFO] [BLWB]: Command Status event, status:1, opcode=0x2002
```

Listing 4.5 – Log d'erreur sur code d'exemple BLE

En consultant la note d'application sur l'interface sans fil des STM32WB [AN BLE], il apparaît que la commande HCI appelée (0x2002 : HCI_LE_READ_BUFFER_SIZE) n'est pas disponible avec la version de firmware par défaut. Le tableau suivant est également consulté.

BLE Stack 5.4 Certified	GAP Peripheral	GAP Central	GATT Server	GATT Client	PHY 2M	Data Length Extension	Legacy Pairing, LE secure connection	Privacy	Filter Accept List	HCI Interface	Direct Test Mode	L2CAP Connection Oriented channels support	Channel Selection #2
Full Extended	•	•	•	•	•	•	•	•	•	•	•	•	•
Full	•	•	•	•	•	•	•	•	•	Reduced	•		
Light	•		•			•	•	•	•	Reduced	•		
HCI Extended					•	•				•	•		
HCI					•	•				•	•		
AdvScan									Reduced				

Figure 4.3 – Tableaux de spécifications des versions FW

Il indique clairement la limitation à l'interface HCI de la version par défaut (Full). Les différentes fonctionnalités GAP et GATT ont d'abord été comprises comme étant indispensables pour le bon fonctionnement du BLE, mais il s'agit en vérité de la prise en charge du coprocesseur dont il est question, alors que le GAP et le GATT sont gérés par Mbed OS. Le firmware *stm32wb5x_BLE_HCILayer_fw.bin* version 1.12.0 a donc été installé à l'aide de STM32CubeProgrammer, dans l'onglet FUS (Firmware Upgrade Service).

CHAPITRE 4. DÉVELOPPEMENT LOGICIEL

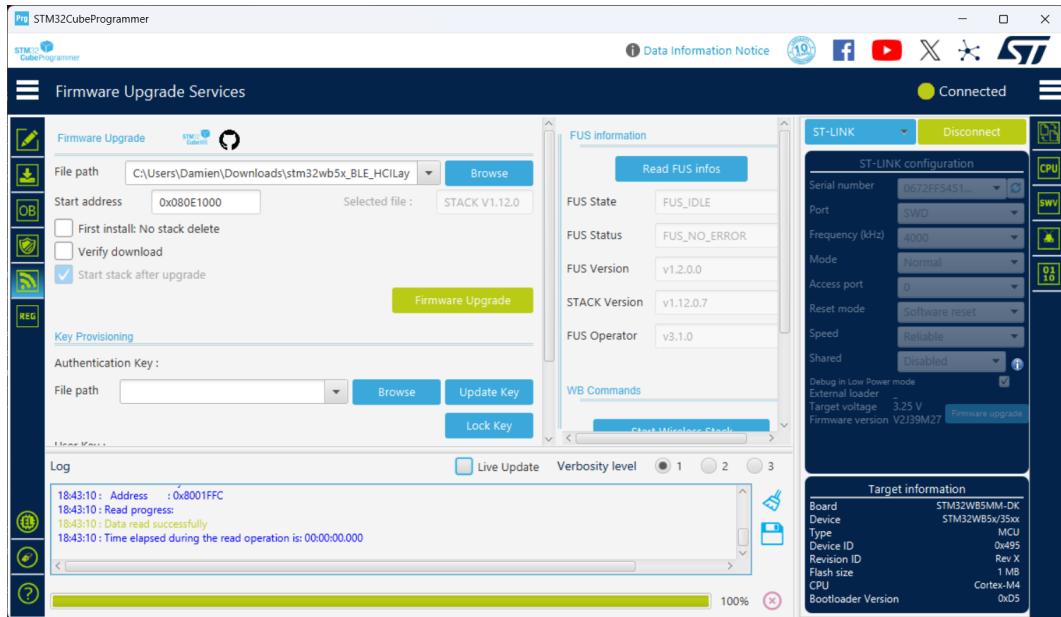


Figure 4.4 – FUS sur STM32CubeProgrammer

L'indication suivante a également été ajoutée au fichier mbed_app.json pour indiquer une modification sur la mémoire ROM conformément au guide.

```
"DISCO_WB5MMG": {
    "target.mbed_rom_size": "0xE1000"
}
```

Listing 4.6 – Modification de la taille ROM pour la carte Disco

4.4 Développement des fonctionnalités

Pour parvenir à composer le programme principal, il est nécessaire d'implémenter et de tester au préalable les différentes fonctionnalités qui le composent. Il est en effet très courant, lorsque l'on travaille avec une API, de débuter le développement logiciel par le test de petits codes d'exemple afin d'identifier le comportement de la fonction à mettre en place.

Les différents protocoles, comme l'I2C ou le BLE, ainsi que l'interface avec les différents capteurs peuvent donc être développés séparément afin que leurs difficultés propres ne s'additionnent pas. Les cartes *Rainette*, conçues préalablement, étant en cours de production au moment du développement logiciel, c'est sur la carte de développement *Disco* (*STM32WB5MM-DK*) que les premiers tests ont été menés.

4.4. DÉVELOPPEMENT DES FONCTIONNALITÉS

4.4.1 Carte Disco

Paramètres Mbed OS de la carte

La target est définie en configurant différentes tâches dans le fichier *tasks.json*. Ces tâches comportent notamment la compilation (*build*) et la programmation du code dans le microcontrôleur (*flash*).

En sélectionnant la target, les fichiers suivants, contenus dans le dossier *targets*, sont associés au projet.

PeripheralPins.c

Définit la correspondance entre les fonctions des périphériques (UART, SPI, ADC, etc.) et les broches physiques du microcontrôleur, utilisé pour configurer dynamiquement les interfaces matérielles.

PinNames.h

Lie les noms symboliques des broches disponibles sur la carte, mappés aux broches physiques du microcontrôleur et sert d'abstraction pour rendre le code portable entre plateformes.

Rgb_led.cpp

Fichier d'implémentation, spécifique à la carte, gérant les opérations liées à la led RGB intégrée.

DiscoIO & DiscoInfo

DiscoIO est une classe singleton centralisant les entrées/sorties du système, y compris l'interface I2C. Cette architecture permet à l'ensemble des fichiers d'accéder aux IO sans recourir à une déclaration *extern*.

DiscoInfo est une structure regroupant l'état des leds *R*, *G*, *B*, ainsi que celui du bouton. Elle sert à conserver les valeurs destinées au Raspberry Pi, notamment dans le *DigitalService*.

Capteur de température STTS22H

La carte STM32WB5MM-DK contient un capteur de température semblable au SHT41-AD1B-R2 de la carte *Rainette*. Il est donc possible de communiquer avec le bus I2C pour le configurer et lire ses mesures.

```

#define WRITE_ADDR 0x70
#define READ_ADDR 0x71

#define WHO_AM_I_REG      0x01
#define CTRL_REG1         0x04
#define TEMP_OUT_L        0x06
#define TEMP_OUT_H        0x07

void write_sensor_register(uint8_t reg, uint8_t value) {
    char data[2] = { (char)reg, (char)value };
    DiscoI0::getInstance().i2c.write(WRITE_ADDR, data, 2);
}

uint8_t read_sensor_register(uint8_t reg) {
    char data = reg;
    char result;
    DiscoI0::getInstance().i2c.write(WRITE_ADDR, &data, 1);
    DiscoI0::getInstance().i2c.read(READ_ADDR, &result, 1);
    return result;
}

float read_temperature() {
    char reg = TEMP_OUT_L;
    char data[2];

    if (DiscoI0::getInstance().i2c.write(WRITE_ADDR, &reg, 1) != 0) {
        printf("Erreur d'écriture I2C\n");
        return -50.0f;
    }

    if (DiscoI0::getInstance().i2c.read(READ_ADDR, data, 2) != 0) {
        printf("Erreur de lecture I2C\n");
        return -50.0f;
    }
    int16_t raw_temp = (int16_t)((data[1] << 8) | (uint8_t)data[0]);
    return raw_temp / 100.0f; // Selon configuration par défaut
}

```

Listing 4.7 – Fonctions pour capteur STTS22H

Le code ci-dessus permet ainsi de communiquer avec le capteur. Sa fiche technique contient ses adresses de lecture et d'écriture, les différents registres internes qu'il est possible d'interroger, ainsi que la formule avec laquelle on transforme la valeur en degrés Celsius.

La fonction *write_register()* sert principalement à configurer le capteur. En écrivant la donnée **0x0C** dans le registre **CTRL_REG1**, on active le mode *FREERUN* qui effectue automatiquement des mesures et gère l'incrémentation de l'adresse interne du registre, mécanisme prévu par ce capteur.

La fonction *read_temperature()* envoie l'ordre de relever une mesure avant de lire le résultat. Ce dernier est retourné sur 2 octets au format *little endian* (octet de poids faible en premier) et doit être divisé par 100 selon la formule par défaut.

4.4. DÉVELOPPEMENT DES FONCTIONNALITÉS

BLE : services et caractéristiques

Les éléments publiés par la carte vers l'unité centrale sont découpés en trois catégories (services) :

- **General** : Données booléennes pour les commandes données par l'unité centrale et les informations générales de la carte.
- **Analog** : Données entières pour les mesures prises par la carte, en lecture uniquement.
- **Digital** : Données booléennes pour les différents éléments numériques de la carte (bouton, leds,...).

Les caractéristiques rattachées aux services sont détaillées dans le tableau suivant (R : *Read*, W : *Write*, N : *Notify*).

Service : General	
START_MEAS	(R,W,N) Commande pour lancer des mesures
END_OF_COMM	(R,W,N) Fin de communication
PING_PONG	(R,W,N) Test ping-pong
Service : Analog	
TEMP	(R,N) Mesure de la température
ADC_TEST	(R,N) Mesure de la tension analogique externe
Service : Digital	
LED_R	(R,W,N) État de la led R
LED_G	(R,W,N) État de la led G
LED_B	(R,W,N) État de la led B
BUTTON	(R,N) État du button 2

Table 4.1 – Services et caractéristiques BLE exposés par la carte

Une classe C++ est créée pour chaque service. Elle permet de regrouper les différentes caractéristiques et propriétés du tableau précédent, de lire ou d'écrire les caractéristiques et fournir le handle (référence d'une ressource) de ces dernières. Le prototype de la classe *GeneralService* est illustré sur la page suivante à titre d'exemple.

```

class GeneralService {
public :
    GeneralService(BLE &arg_ble);

    void resetAndAddService();

    bool readStartMeas();
    void writeStartMeas(bool new_start_meas);
    bool readEndOfComm();
    void writeEndOfComm(bool new_end_of_comm);
    bool readPingPong();
    void writePingPong(bool new_ping_pong);

    GattAttribute::Handle_t getStartMeasHandle();
    GattAttribute::Handle_t getEndOfCommHandle();
    GattAttribute::Handle_t getPingPongHandle();

private :
    BLE &ble;

    bool start_meas_val;
    bool end_of_comm_val;
    bool ping_pong_val;
    ReadWriteGattCharacteristic<bool> start_meas_char;
    ReadWriteGattCharacteristic<bool> end_of_comm_char;
    ReadWriteGattCharacteristic<bool> ping_pong_char;

    GattCharacteristic* charTable[3];
    GattService service;
};


```

Listing 4.8 – Classe GeneralService de la carte Disco

EventHandler GAP/GATT

Deux classes sont développées pour gérer l'ensemble des interactions avec la pile Bluetooth et les données fournies par le client : *GapEventHandler* et *GattServerEventHandler*. Cette architecture est reprise des exemples fournis par Arm Ltd. sur le fonctionnement du BLE [BLE EX]. Elles héritent de classes appartenant à Mbed OS, surchargeant certaines fonctions de *callback* pour personnaliser leur comportement.

La classe *GapEventHandler* gère l'initialisation de la pile BLE, les connexions et déconnexions du client ainsi que l'advertising. Elle permet d'indiquer l'état la connexion et de gérer des timeouts liés à l'advertising et à la communication.

La classe *GattServerEventHandler* est plus légère et prend en charge certaines fonctionnalités. Elle surcharge deux *callbacks* pour définir des actions lorsque des caractéristiques sont lues ou écrites par un client.

Le prototype de la classe *GattServerEventHandler* est présenté ci-après.

4.4. DÉVELOPPEMENT DES FONCTIONNALITÉS

```
class GattServerEventHandler : public ble::GattServer::EventHandler {
public:
    GattServerEventHandler();

private:
    void onDataRead(const GattReadCallbackParams &params) override;
    void onDataWritten(const GattWriteCallbackParams &params) override;
};
```

Listing 4.9 – Classe GattServerEventHandler

Classe DiscoApp

La classe *DiscoApp* est l’élément principal du programme. Elle contient l’ensemble des fonctionnalités au travers de ses deux méthodes publiques : *init()* et *run()*, ainsi que par diverses méthodes privées. Elle contient les classes *GapEventHandler* et *GattServerEventHandler* pour les intégrer à ses fonctionnalités, tout en découplant distinctement les fonctions. La personnalisation de l’application est principalement réalisée à cet endroit, les classes *BLE* pouvant être presque intégralement reprises à l’identique pour le code de la carte *Rainette*. Le prototype de la classe est renseigné ci-dessous.

```
class DiscoApp {
public:
    DiscoApp();
    void init();
    void run();

private:
    void clearAndFillServices();
    void handleDataWritten();
    void fillHandlesVector();
    void startMeasurement();
    void pingPongTest();
    void pingPongTimeout();
    void btn1Irq();
    void btn2Irq();
    void toggleButtonChar();

    BLE &ble;
    GapEventHandler gap_handler;
    GattServerEventHandler gatt_handler;

    bool ping_pong_in_progress;

    GeneralService general_service;
    AnalogService analog_service;
    DigitalService digital_service;
};
```

Listing 4.10 – Classe DiscoApp

Structure du programme

La figure suivante fournit un aperçu des classes et éléments du programme, ainsi que de leurs interactions.

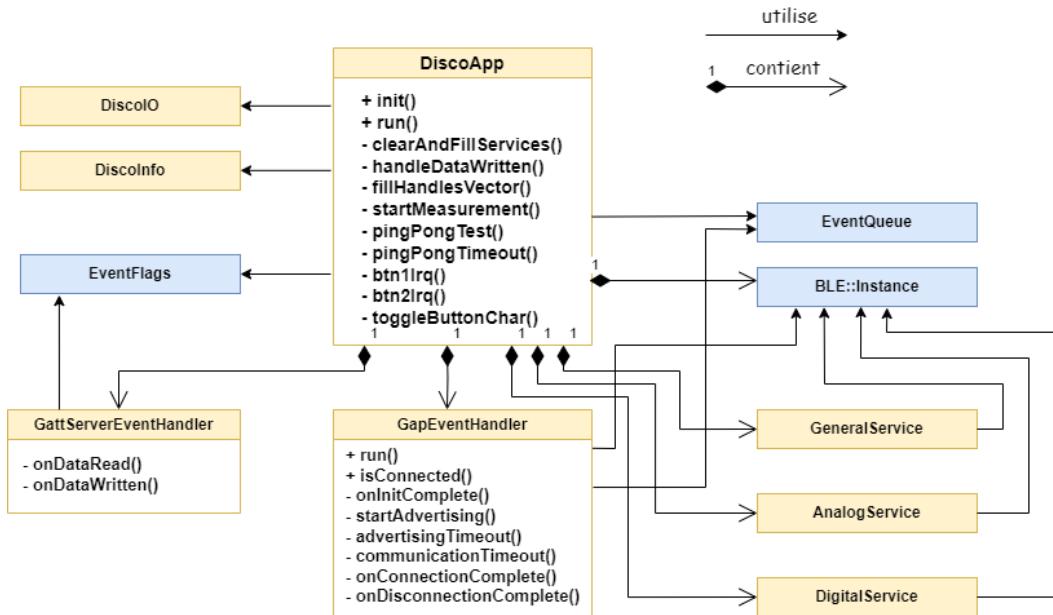


Figure 4.5 – Diagramme du programme de la carte Disco

L'organigramme développé pour ce premier programme est illustré ci-après. Après la phase d'initialisation, l'utilisateur doit pressé le bouton B1 pour activer la communication BLE. Si la connexion a lieu avant le timeout, l'utilisateur peut transmettre des instructions à la carte *Disco* depuis l'interface graphique. Il peut ainsi lui demander d'effectuer des mesures et de retourner le résultat. Un système de test de communication "ping-pong" est également présent, chaque minute, le serveur active ce test et l'utilisateur doit y répondre dans le temps imparti pour confirmer sa présence.

4.4. DÉVELOPPEMENT DES FONCTIONNALITÉS

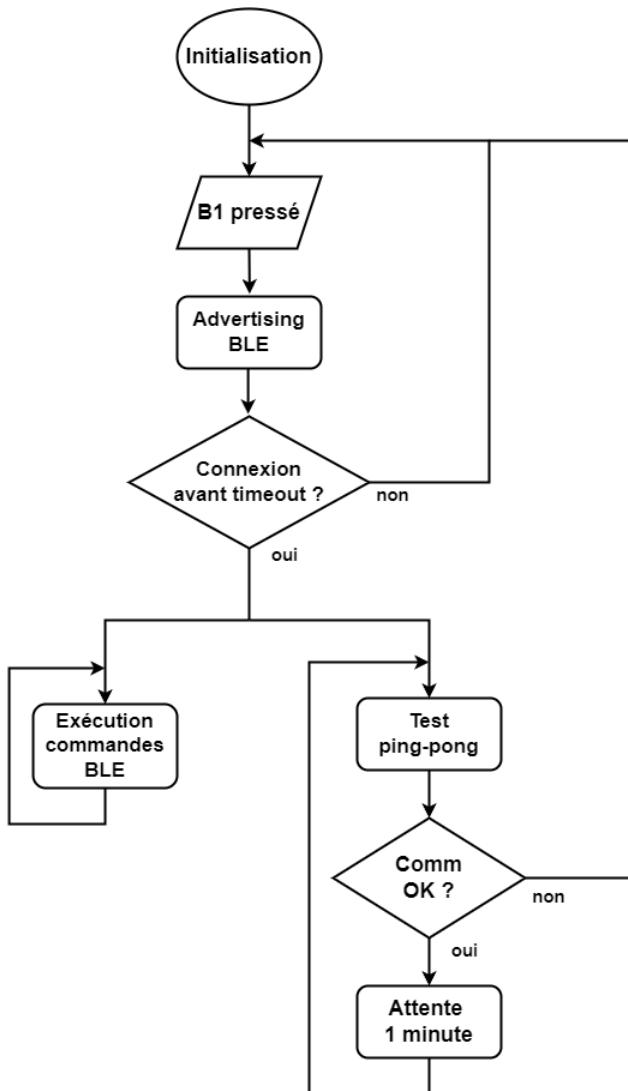


Figure 4.6 – Organigramme du programme de la carte Disco

4.4.2 Carte Rainette

Une fois la structure et les fonctionnalités du code *Disco* validées, la conception du programme *Rainette* débute en réutilisant les éléments développés précédemment.

Paramètres Mbed OS de la carte

Nouvelle carte, nouveaux paramètres. Le script Python *STM32_gen_PeripheralPins.py*, présent dans le dossier *TARGET_STM32WB/tools* permet de générer les fichiers *PeripheralPins.c* et *PinNames.h* propres au module STM32WB5MMG employé. Un dossier *TARGET_RAINETTE* est créé à la racine du projet *Mbed OS* pour contenir ces fichiers. Un fichier *custom_targets.json* est également ajouté, son contenu est détaillé sur la page suivante.

```
{
  "RAINETTE": {
    "inherits": [
      "MCU_STM32WB5MxG"
    ],
    "detect_code": [
      "0884"
    ],
    "device_name": "STM32WB55VGYx"
  }
}
```

Listing 4.11 – Fichier *custom_targets.json*

Enfin, la ligne suivante est ajoutée au fichier *mbed_app.json*, elle découle du raisonnement qui avait conduit à inclure la ligne équivalente pour la carte *Disco*.

```
{
  "RAINETTE": {
    "target.mbed_rom_size": "0xE1000"
  }
}
```

Listing 4.12 – Modification de la taille ROM pour la carte Rainette

BLE : services et caractéristiques

Les services et caractéristiques suivent le même principe que pour la carte *STM32WB5MM-DK*. Le nombre de caractéristiques augmente mais elles conservent le même découpage de services. Les caractéristiques rattachées aux services sont détaillées dans le tableau suivant (R : *Read*, W : *Write*, N : *Notify*).

4.4. DÉVELOPPEMENT DES FONCTIONNALITÉS

Service : General	
BATT_DETECT	(R,N) Indique la présence de la pile
START_MEAS	(R,W,N) Commande pour lancer des mesures
END_OF_COMM	(R,W,N) Fin de communication
PING_PONG	(R,W,N) Test ping-pong
Service : Analog	
PROBE0	(R,N) Mesure de la sonde 0
PROBE1	(R,N) Mesure de la sonde 1
LUMI	(R,N) Mesure de la luminosité
TEMP	(R,N) Mesure de la température
HUMI	(R,N) Mesure de l'humidité
BATT	(R,N) Mesure de la tension de la pile
Service : Digital	
FLOAT	(R,N) État du flotteur
MOS_REL0	(R,W,N) État du MOSFET 0
MOS_REL1	(R,W,N) État du MOSFET 1
GPIO0	(R,W,N) État du GPIO 0
GPIO1	(R,W,N) État du GPIO 1
GPIO2	(R,W,N) État du GPIO 2
GPIO3	(R,W,N) État du GPIO 3
GPIO4	(R,W,N) État du GPIO 4
BUTTON	(R,N) État du bouton

Table 4.2 – Services et caractéristiques BLE exposés par la carte

RainetteIO, RainetteInfo & I2C

Les classes *DiscoIO* et *DiscoInfo* sont reprises et modifiées pour s'adapter à cette application. *RainetteInfo* contient la variable *batt_detect*, indiquant le mode de fonctionnement du programme, ainsi que l'état du bouton et celui du flotteur.

L'I2C bloquant l'entrée du système en mode *deepsleep*, il a été retiré de la classe *RainetteIO*. N'étant utilisé que durant les mesures, il est désormais alloué dynamiquement juste avant leur déclenchement, puis détruit une fois les résultats obtenus.

```
I2C* i2c = new I2C(SDA_PIN, SCL_PIN);
i2c_init(i2c);

// mesures i2c

i2c_deinit();
delete i2c;
```

Listing 4.13 – Allocation dynamique de l'I2C pour les mesures

EventHandler GAP/GATT

Les classes *GapEventHandler* et *GattServerEventHandler* sont reprises du code *Disco*. Seule la méthode *onDataWritten()* est adaptée afin ajouter les caractéristiques supplémentaires, telles que les GPIO et les relais MOSFET.

Classe RainetteApp

Cette classe reprend une grande partie de la classe *DiscoApp*, en adaptant ses différentes méthodes liées à la carte. Ainsi, les fonctions en rapport avec la détection de la pile, les *IRQ* du bouton et du flotteur ou encore le clignotement de la led0 sont prises en compte.

Structure du programme

La figure suivante fournit un aperçu des classes et éléments du programme, ainsi que de leurs interactions.

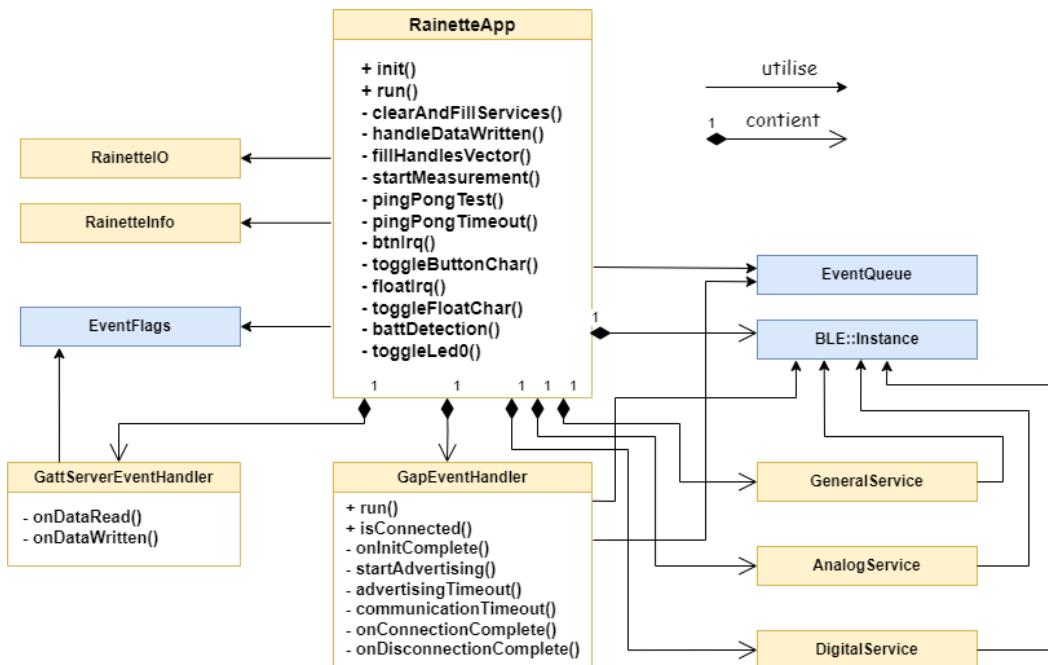


Figure 4.7 – Diagramme du programme de la carte Rainette

L'organigramme de ce programme est celui présenté dans la rubrique *Fonctionnement global du programme*.

4.5. PROGRAMMATION DU RASPBERRY PI

4.5 Programming du Raspberry Pi

4.5.1 Carte utilisée

Un Raspberry Pi Compute Module 4 IO Board est utilisé dans le rôle du client BLE. Cette version, comportant davantage de périphériques, n'est pas nécessaire, elle était simplement en stock dans le laboratoire. Le système d'exploitation *Raspberry Pi OS* a été installé, il s'agit de la distribution Linux officielle de la marque Raspberry.

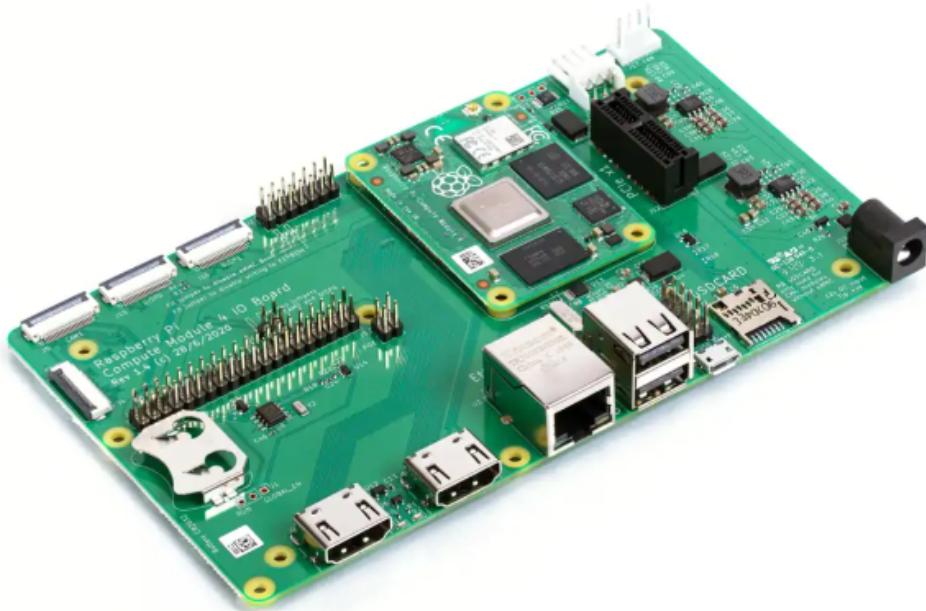


Figure 4.8 – *Raspberry Pi Compute Module 4 IO Board*

4.5.2 Informations générales sur le code

L'application développée est une version de test et d'illustration, elle contient une interface graphique (GUI) afin de pouvoir communiquer manuellement avec le serveur BLE. Elle est codée en Python v3.11 car il s'agit d'un langage simple et riche, possédant beaucoup d'API. Cette version Python était installée par défaut sur *Raspberry Pi OS*, c'est l'une des dernières versions et convient très bien pour l'application.

Comme indiqué précédemment, les codes pour la *STM32WB5MM-DK* et pour la *Rainette* sont similaires dans leur structure. Les seules différences apparaissent dans la liste des caractéristiques proposées par le serveur BLE.

Le code repose sur 3 bibliothèques Python : *bleak*, *asyncio* et *PySide6*.

bleak est une bibliothèque Python libre qui implémente un client GATT Bluetooth Low Energy multiplateforme, elle fournit une API asynchrone unifiée pour découvrir, se connecter et échanger des caractéristiques avec des périphériques BLE via les piles natives de l'OS.

asyncio est le cadre officiel d'I/O asynchrones de la bibliothèque standard Python : il orchestre une boucle d'événements et des tâches afin d'exécuter du code concurrent sans threads explicites.

CHAPITRE 4. DÉVELOPPEMENT LOGICIEL

PySide6, enfin, constitue la composante principale de « Qt for Python », il s'agit du lien officiel entre Python et Qt 6, exposant en Python l'ensemble des API C++ de Qt (widgets, Qt Quick, multimédia, etc.) pour créer des interfaces graphiques natives, portables et maintenues par le projet Qt lui-même.

L'organigramme du programme est détaillé ci-dessous.

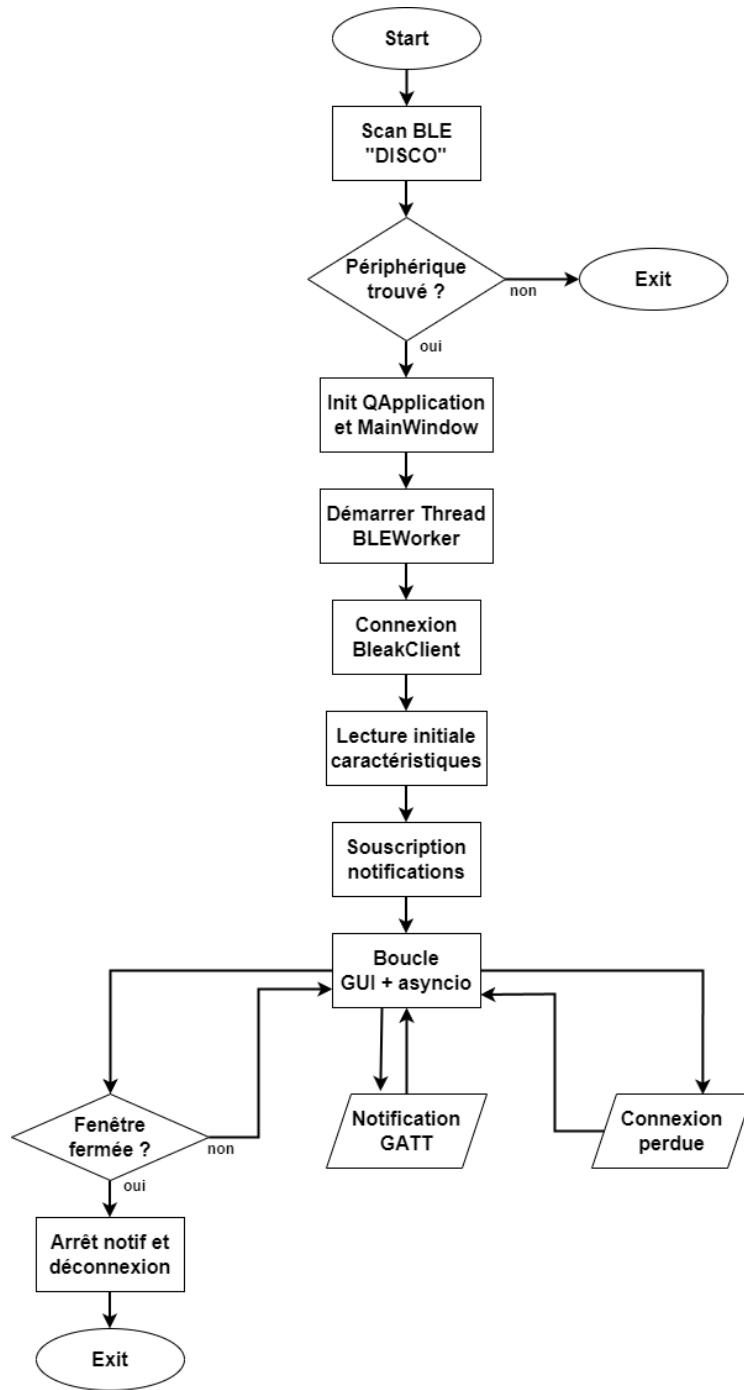


Figure 4.9 – Organigramme des codes Raspberry Pi

4.5. PROGRAMMATION DU RASPBERRY PI

4.5.3 Spécificités du programme

Modèle de concurrence hybride

Le code combine la programmation asynchrone (asyncio) avec un thread dédié (QThread) pour exécuter la boucle d'événements asyncio en parallèle de l'interface Qt. Ce choix garantit que les notifications BLE sont reçues et traitées sans bloquer l'interface graphique et évite les conflits entre la boucle Qt (synchrone) et celle d'asyncio, lesquelles sont mutuellement exclusives dans le thread principal.

Conversion des valeurs analogiques

L'API Mbed OS de la couche GATT permet de transmettre des tableaux d'octets sous forme de *uint8_t*. Pour transmettre les valeurs analogiques, le serveur multiplie par 100 ces dernières avant de les caster. Cela signifie que la conversion de ces données analogiques doit être pris en compte dynamiquement. Lorsqu'une valeur est mise à jour, son UUID est analysé. S'il s'agit du *AnalogService*, la division par 100 est automatiquement effectuée.

```
try:
    numeric_val = int(val)
    if uuid in analog_uuids:
        read_field.setText(f"{numeric_val / 100:.2f}")
    else:
        read_field.setText(str(numeric_val))
except ValueError:
    read_field.setText(val)
```

Listing 4.14 – Conversion des valeurs analogiques

4.5.4 Interface utilisateur

Les interfaces utilisateurs développées sont illustrées ci-dessous. Elles partagent la même structure mais se distinguent par les caractéristiques proposées. Lorsque le script détecte le serveur, la fenêtre apparaît à l'utilisateur. Si aucun serveur n'est détectée après 10 secondes, le programme se termine. Les valeurs sont lues une première fois pour être affichées, puis uniquement mises à jour lorsqu'une modification apparaît.



Figure 4.10 – GUI Disco



Figure 4.11 – GUI Rainette

Chapitre 5

Mise en service et tests

Les cartes *Rainette* ont été livrées 27 jours après leur commande. Ce délai est principalement dû au montage des composants sur les PCB par le fabricant *PCBWay*. Si ce grand intervalle de temps n'a pas grandement impacté le bon déroulement du projet, le code étant développé à ce moment-là, il serait avantageux de faire assembler les cartes sur place lors de la prochaine commande.

Les composants ayant dû être commandés à part en raison leur temps d'acheminement chez *PCBWay* jugé trop long ont été réceptionnés le même jour que les cartes. Si une inspection visuelle a été menée sur l'ensemble des cartes telles qu'elles ont été reçues, l'assemblage des composants restants n'a d'abord été réalisé que sur une seule carte. Cette précaution permettait d'éviter un retour en arrière si toutes les cartes avaient été assemblées et qu'un défaut matériel se présentait.

5.1 Contrôle qualité

Le contrôle qualité se décline en trois phases, la première consiste en un contrôle visuel de la carte reçue, la seconde en un nouveau contrôle visuel après l'assemblage des derniers composants et la dernière en un test de l'alimentation avant son raccordement au reste du circuit.

5.1.1 Premier contrôle visuel

Cette inspection a pour objectif de s'assurer de la qualité des brasures et de la polarité des composants sur les cartes envoyées par *PCBWay*. Un stéréomicroscope numérique a été employé pour vérifier les brasures et un multimètre a servi à contrôler la polarité des différentes diodes ainsi que les valeurs des résistances.

Résultats	
Brasures	OK
Polarité des composants	OK

Table 5.1 – Résultats du premier contrôle visuel

Cette première phase ne révèle aucun défaut sur les cartes assemblées par le fabricant. Cependant, il n'est pas possible d'inspecter visuellement le module STM32WB5MMG car ses pattes se trouvent sous son boîtier. Un contrôle de conductivité entre chaque patte aurait été trop fastidieux, il a été choisi de valider cette première partie.

5.1.2 Deuxième contrôle visuel

Ce second test, plus rapide, intervient après le montage manuel des derniers composants sur la carte. Les mêmes appareils que lors de l'étape précédente ont été utilisés.

Résultats	
Brasures	OK
Polarité des composants	OK

Table 5.2 – Résultats du deuxième contrôle visuel

Cette seconde partie ayant été validée, un test de l'alimentation peut être effectué.

5.1.3 Contrôle des sources d'alimentation

La carte n'a pas été conçue avec une série de header/jumper pour contrôler l'acheminement de la tension vers les différentes parties de la carte, cependant il est possible de vérifier les tensions présentées au convertisseur DC/DC. Pour cela, le jumper servant à la sélection de la source d'alimentation est retiré et il devient possible de mesurer les tensions sur les broches du header 3x2.

Test pile

La tension de la pile est simulée avec une alimentation de laboratoire. La valeur 4.5V correspond aux tensions superposées de 3 piles AAA.

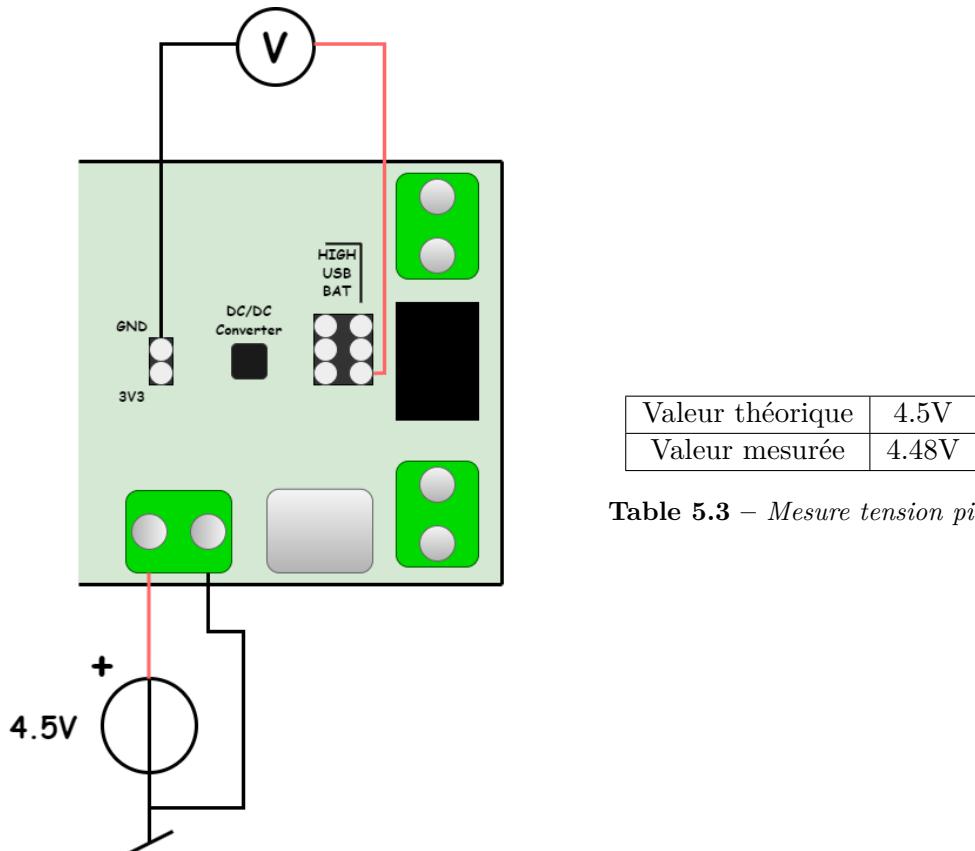


Table 5.3 – Mesure tension pile

Figure 5.1 – Schéma de mesure - Pile

5.1. CONTRÔLE QUALITÉ

La tension mesurée correspond à la tension appliquée par l'alimentation de laboratoire, sans chute de tension sur la diode Schottky. Cela s'explique par la mesure à vide, ce qui n'entraîne pas de chute de tension due au seuil de la diode.

Test USB

Pour ce deuxième test, un adaptateur AC/DC 5V USB est utilisé pour alimenter la carte.

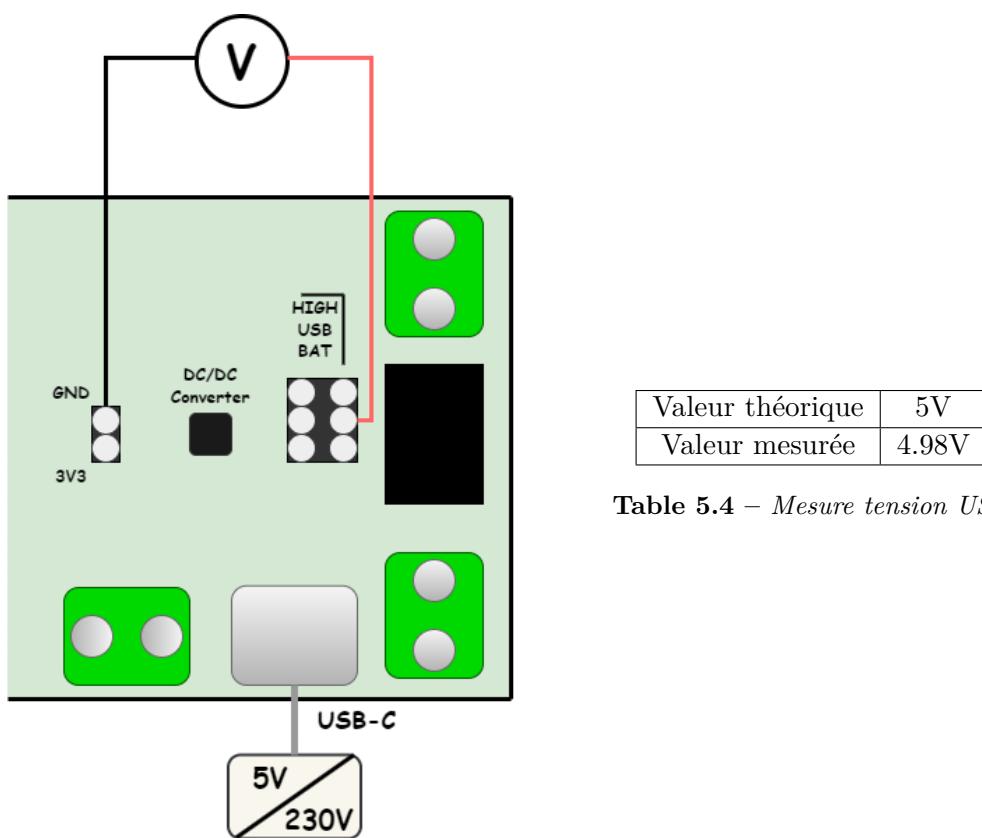


Table 5.4 – Mesure tension USB

Figure 5.2 – Schéma de mesure - USB

La mesure est conforme au résultat attendu.

Test high DC

Afin de vérifier le bon fonctionnement du TSR 1-2450E, l'alimentation de laboratoire fournit une tension continue comprise entre 8V et 20V (valeurs de test arbitraires). Ici aussi, le résultat correspond à la valeur attendue, ce qui permet de valider le bon fonctionnement des trois sources d'alimentation.

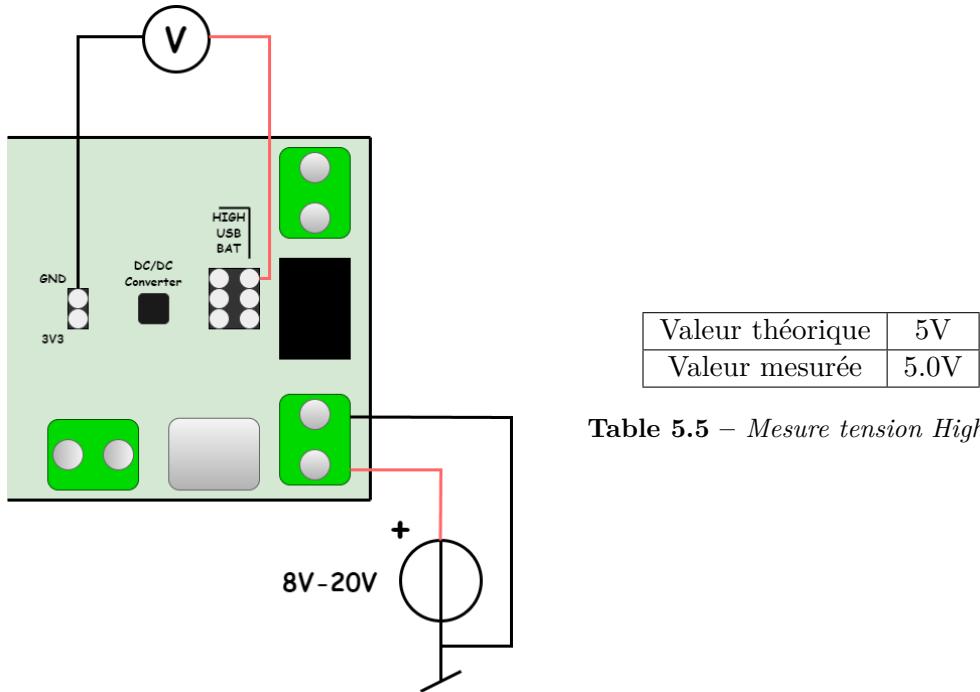


Table 5.5 – Mesure tension High DC

Figure 5.3 – Schéma de mesure - High DC

Test du buck-boost 3V3

Finalement, après s'être assuré que les différentes sources fournissent bien la tension attendue, il est possible de tester la tension de sortie du convertisseur DC/DC en charge de fournir le VCC au circuit. Le schéma de mesure ci-dessous illustre une mesure réalisée avec les piles comme source d'alimentation mais le résultat est identique avec les autres sources.

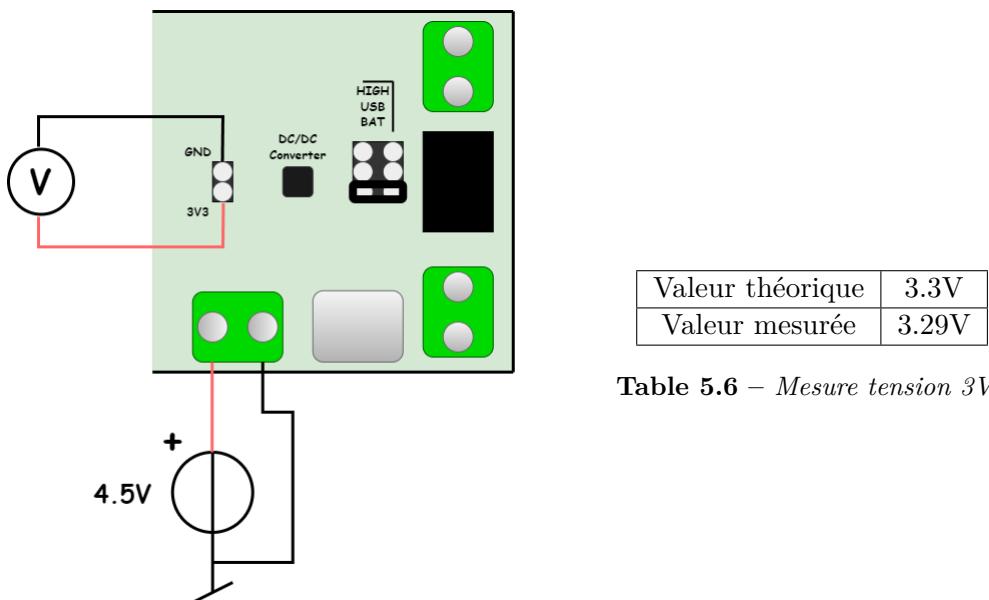


Table 5.6 – Mesure tension 3V3

Figure 5.4 – Schéma de mesure - 3V3

Cela conclut les tests préliminaires menés avant de flasher le code sur la carte, les résultats correspondent aux attentes.

5.2. TESTS FONCTIONNELS DE LA CARTE

5.2 Tests fonctionnels de la carte

5.2.1 Connecteur ST-Link

Un oubli a été constaté lorsque le connecteur TC2050-IDC a dû être utilisé. Ce connecteur est en effet compatible avec le ST-Link V3SET mais uniquement au moyen d'un adaptateur *TC2050-ARM2010 ARM 20-pin to TC2050 [ADAPTER]*. Ce connecteur onéreux aurait nécessité un délai de livraison important, c'est pourquoi il a été remplacé par des câbles plats mâle-femelle, en se basant sur le tableau de correspondance fourni dans la fiche technique.

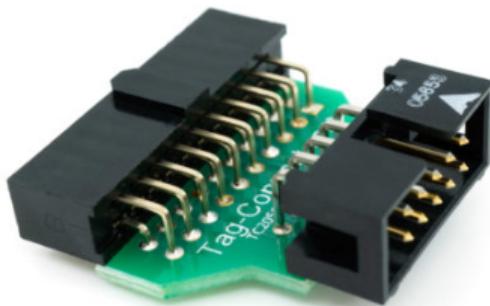


Figure 5.5 – Adaptateur TC2050-ARM2010 ARM 20-pin to TC2050

5.2.2 Programmation de la carte

La programmation a débuté par la mise à jour du firmware du coprocesseur M0 (comme détaillé précédemment). La connexion entre le logiciel *STM32CubeProgrammer* et le module STM32WB5MMG a été établie avec succès et le firmware *stm32wb5x_BLE_HCILayer_fw.bin* version 1.12.0 a pu être correctement installé. Le code *Rainette* a ensuite pu être téléchargé depuis l'onglet *Erasing & Programming* car cela ne fonctionnait pas avec la commande *Openocd flash (GCC debug)* utilisée sur la carte *Disco*.

Si la phase d'initialisation se déroule sans problème, le programme s'arrête subitement lors de l'appel de la méthode *ble.init()* qui initialise la couche BLE. Une erreur apparaît ensuite, sans indiquer de piste de correction. Ce problème n'est mentionné nulle part sur les différents forums et documentation. En utilisant le mode *Run & Debug*, la cause de l'erreur est un timeout consécutif à une attente trop importante d'un sémaaphore dans le driver HCI. Une cause possible à ce changement par rapport au code *Disco* (qui contient la même structure) est la recompilation complète du programme, mais cela n'a pu être prouvé. Cette difficulté a, après de longs essais, pu être surmontée en suivant systématiquement la procédure suivante lors de la programmation de la carte depuis *STM32CubeProgrammer*.

1. Reconnexion à la carte
2. Onglet FUS : *Upgrade firmware*
3. Onglet FUS : *Start Wireless Stack*
4. Onglet Erasing & Programming : *Start Programming*

Cette procédure doit être suivie à chaque reprogrammation de la carte. L'étape *Start Wireless Stack* n'était pas nécessaire sur la carte *Disco* auparavant, elle peut expliquer le timeout qui déclenche l'erreur mais pas la raison pour laquelle il ne faut l'exécuter que pour le code *Rainette*.

Afin de recevoir sur le PC les messages envoyés par le programme *Rainette* sur la sortie standard, un câble USB modifié avec des câbles plats femelles à une extrémité est employé. Raccordé aux RX et TX de l'UART ainsi qu'au GND de la carte, il permet d'établir une communication série, visible en ouvrant un port série depuis une console *VSCCode* par exemple.

5.2.3 Validation des fonctionnalités

Les tests suivants sont réalisés sur la carte équipée du programme principal, connectée au *Raspberry Pi* via Bluetooth. Cette configuration permet d'activer certaines actions depuis l'interface graphique et de s'assurer du bon fonctionnement du programme.

Connexion BLE

La connexion s'établit correctement entre la carte *Rainette* et le *Raspberry Pi*. Le transfert d'informations y est fluide et correspond aux attentes. Il a toutefois été constaté certains essais où la connexion s'établit entre les cartes, l'interface graphique apparaissant, mais où les données ensuite ne transitent pas, comme si la communication était bloquée. Lorsque ce phénomène apparaît, il est nécessaire de relancer complètement l'advertising côté serveur. À ce stade de prototype, un *reset* manuel est l'option la plus rapide mais un timeout pour l'advertising a été implémenté au cas où le système devait être entièrement autonome et gérer cette situation.

Tension de la pile

Le premier test mené sur la tension de la pile ne correspondait pas aux attentes. Les mesures étaient en effet bien inférieures aux valeurs attendues et avaient un comportement non linéaire (erreur relative qui varie selon la tension). Des vérifications ont été effectuées, aucun défaut physique n'est à relever sur les composants de la partie *Battery tester* et leurs valeurs sont correctes.

En menant à nouveau le test sur une autre carte *Rainette* où la diode Zener D29 n'était pas montée, ce phénomène disparaît, incriminant ainsi la diode MMSZ5227BT1G. Un courant de fuite semble provoquer une baisse d'impédance qui abaisse la tension mesurée, et ce de manière non-linéaire. Pour ne pas avoir à supprimer cette protection, ce qui exposerait le microcontrôleur à des tensions supérieures à ses limites, une autre solution a été envisagée.

En effectuant une série de mesures (valeurs attendues et valeurs biaisées), il est possible de réaliser une régression polynomiale inversée de degré 2. Cette étape permet, à partir des valeurs biaisées, de calculer la véritable tension de pile.

5.2. TESTS FONCTIONNELS DE LA CARTE

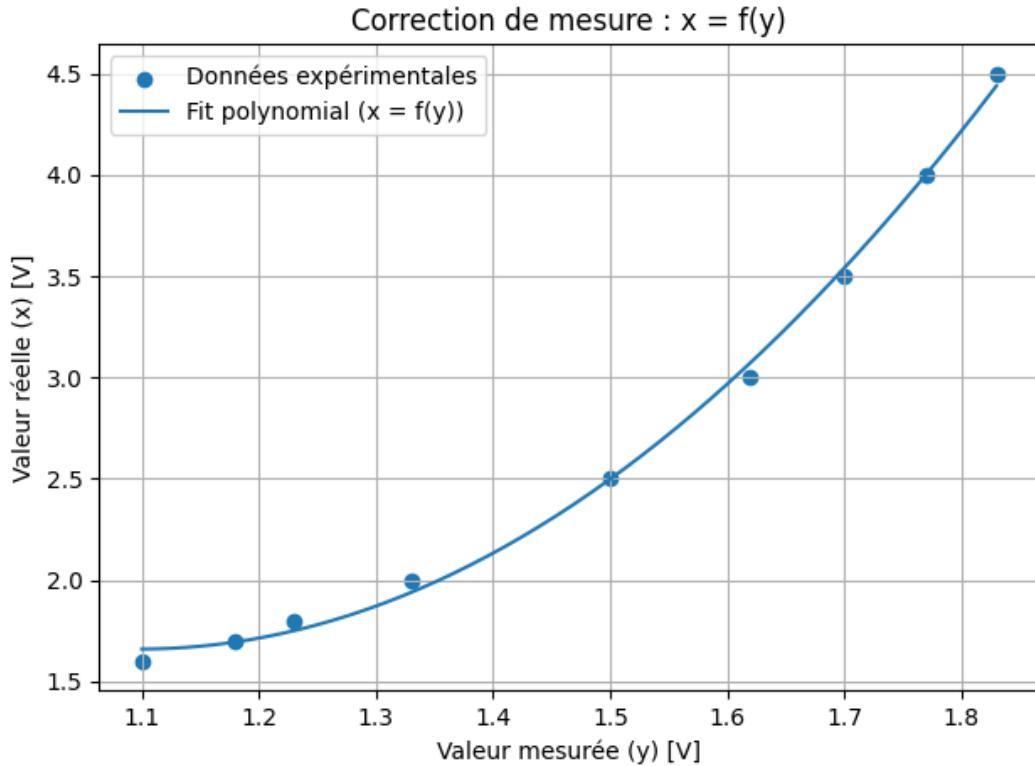


Figure 5.6 – Régression polynomiale pour correction de mesure

La régression polynomiale permet de calculer les coefficients a , b et c de l'équation suivante.

$$x = a \times y^2 + b \times y + c \quad (5.1)$$

où :

$$a = 5.187502,$$

$$b = -11.3899,$$

$$c = 7.911509$$

Cette correction est ainsi appliquée aux valeurs mesurées, et permet de retrouver les bonnes valeurs, validant ainsi la lecture de la tension de la pile. Cette lecture permet de définir correctement le mode de fonctionnement du programme lors de son initialisation.

Sondes analogiques

La lecture des sondes analogiques fonctionne comme attendu. Si le header 3x1 n'est pas brasé sur la carte, une tension de l'ordre de quelques centaines de millivolts peut être mesurée lors de la lecture. En connectant l'interface *Atlas Scientific*, la tension analogique mesurée sur le point de test est identique à celle relevée par l'ADC.

CHAPITRE 5. MISE EN SERVICE ET TESTS

Capteur SHT41

L'adresse du capteur de température et d'humidité *SHT41-AD1B-R2* est correctement détectée (*0x44*) et la communication au travers des fonctions développées fournit des valeurs cohérentes. Ces dernières varient de manière attendue lorsque l'on cherche à modifier les grandeurs mesurées (en approchant le doigt par exemple). La température relevée par ce capteur est légèrement supérieure à celle mesurée à l'aide de la sonde analogique et la PT1000. Ce comportement, déjà observé sur la carte *Disco*, peut s'expliquer par un léger échauffement de la carte et par un possible manque d'étalonnage.

Capteur VEML6030

Les fonctions propres au capteur de luminosité *VEML6030* ont été développées sur la base de la fiche technique du composant et de codes d'exemple trouvés sur Internet. Cependant et après de nombreuses tentatives, il n'a pas été possible de communiquer correctement avec le capteur.

Lorsqu'une commande I2C était initiée pour communiquer avec ce composant, une erreur de communication était toujours signalée, indiquant un dysfonctionnement avec le protocole I2C. En observant les lignes SDA et SCL à l'oscilloscope, la cause de ces erreurs a été identifiée. Il s'agissait du capteur de température et d'humidité qui n'était pas alimenté au moment de la communication avec le *VEML6030*, ce qui entraînait une déviation du courant sur ses pattes I2C comme le montre l'illustration suivante.



Figure 5.7 – Défaut de communication I2C avec le *VEML6030*

5.2. TESTS FONCTIONNELS DE LA CARTE

En activant le capteur *SHT41-AD1B-R2*, ce problème disparaît et les communications redeviennent lisibles avec l'oscilloscope.



Figure 5.8 – Communication I2C correcte avec le VEML6030

Si la partie concernant le protocole I2C fonctionne désormais, le capteur répondant correctement avec les acknowledges, les données qu'il renvoie sont systématiquement nulles (valeurs hexadécimales 0x00), peu importe la requête ou le registre interrogé. Le registre de configuration du capteur a été modifié pour activer les fonctionnalités, de la même manière que les exemples trouvés sur Internet, mais cela n'a pas impacté le résultat. Le test a été mené sur une carte encore inutilisée pour vérifier qu'il ne s'agissait pas d'un problème entraîné par un défaut matériel mais les valeurs restent nulles. La polarité ainsi que la présence de courts-circuits ont été contrôlés, sans pouvoir trouver de cause à ce phénomène.

Arrivé au terme du temps imparti pour ce travail de diplôme, ce problème n'a pas pu être corrigé, laissant en suspens l'emploi de ce capteur de luminosité. Si ce projet devait être repris par la suite et que la communication ne pouvait pas être améliorée, il serait préférable d'essayer un autre capteur de luminosité. Trouver un composant compatible avec le footprint du *VEML6030*, comme le *VEML6035* du même fabricant Vishay, éviterait d'avoir à modifier le design du PCB.

Bouton et flotteur

Un *toggle switch* a été utilisé pour simuler le flotteur. Lorsque le bouton et le flotteur sont activés, l'indication est correctement transmise sur l'interface graphique.

GPIO et relais MOSFET

Dans la première version du programme *Rainette*, toutes les GPIO sont configurées en sortie. Il est ainsi possible de les contrôler, tout comme les relais MOSFET, depuis l'interface graphique. Les résultats correspondent aux attentes, les relais MOSFET permettent bien d'ouvrir et de fermer un circuit.

Résumé des tests

Le tableau suivant résume l'état de fonctionnement des différentes fonctionnalités présentées dans cette rubrique.

Fonctionnalités	État
Connexion BLE	OK mais peut bloquer parfois
Tension de la pile	OK après correction de l'influence de la diode Zener
Sondes analogiques	OK
Capteur SHT41	OK
Capteur VEML6030	Ne fonctionne pas
Bouton et flotteur	OK
GPIO et relais MOSFET	OK

Table 5.7 – Résumé des fonctionnalités

5.3 Mesure de consommation

Un aspect important du projet est la consommation du système embarqué. Elle n'est déterminante que pour le mode de fonctionnement sur pile. C'est donc cette configuration qui est mise en place pour mener les tests.

Les mesures de courant sont réalisées avec le *Power Profiler Kit II (PPK2)*. Il s'agit d'un petit appareil développé par le fabricant *Nordic Semiconductor* facile d'utilisation et conçu pour la mesure de courant sur systèmes embarqués. Relié à un PC par câble USB, il permet de visualiser de très faibles courants et d'enregistrer des mesures.



Figure 5.9 – Power Profiler Kit II

Le *PPK2* est relié au PC à l'aide d'un câble USB, le logiciel dédié à cet appareil doit être installé [PPK2 SW]. Il est ensuite branché comme un ampèremètre pour mesurer le courant entrant dans le système. Le ST-Link ne doit pas être relié à la carte, sans quoi il consomme plusieurs milliampères et fausse la mesure. Il en va de même pour le port série qui consomme même lorsqu'aucune donnée n'est transmise.

5.3. MESURE DE CONSOMMATION

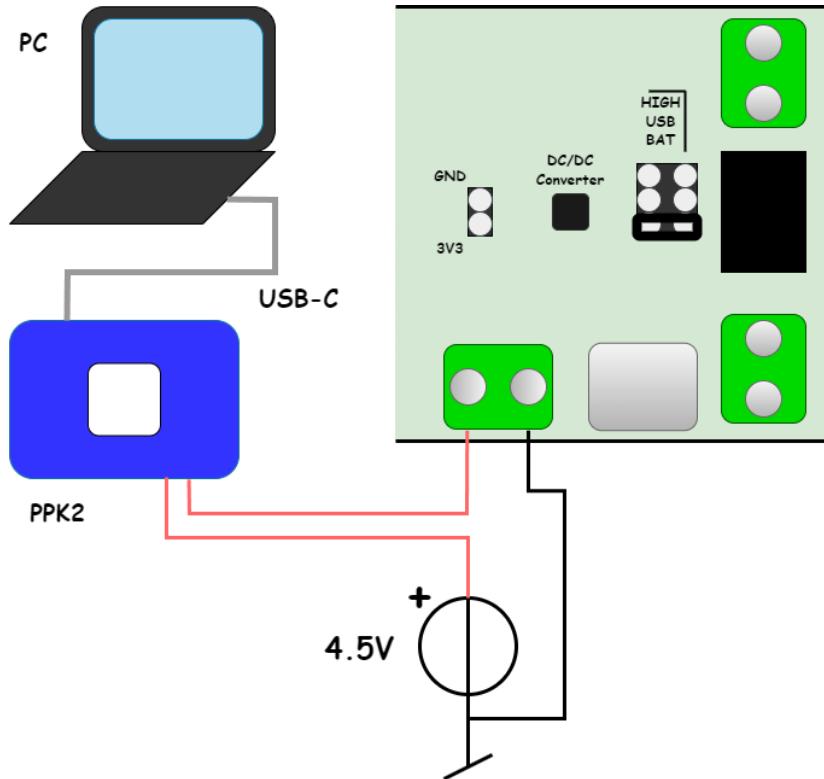


Figure 5.10 – Schéma de mesure - consommation de courant

La première mesure réalisée ne correspondait pas aux résultats attendus. En plus des mesures faussées par la présence du ST-Link et du port série, un important pic, atteignant 370 mA, apparaissait lors de l'étape des mesures. Après analyse, il s'agit de la charge du condensateur C12 10 μ F lorsque le *load switch* IC6 était activé. Ce condensateur, initialement recommandé par la fiche technique n'est finalement pas pertinent dans ce cas d'application où le capteur de température ne doit être actif que sur une très courte période. En prenant la décision de la retirer, ce pic disparaît, rendant la consommation plus cohérente, sans impacter négativement l'alimentation du capteur.

À l'activation de la source d'alimentation, un pic de 870mA est atteint, premier appel de courant dû aux multiples condensateurs et au réveil du module STM32WB5MMG. S'ensuit une courte période de 2.5 secondes où la consommation oscille autour d'une valeur moyenne de 5mA, avec des pics à 15mA. Elle correspond aux premières instructions réalisées au démarrage par le module ainsi qu'à la phase d'initialisation du programme.

Après cette phase de relative haute consommation, cette dernière chute en passant en mode *deepsleep* et se stabilise autour des 700 μ A pour une durée de 10 secondes, suite à quoi la consommation baisse encore pour atteindre environ 170 μ A. La durée du mode *deepsleep* est identique à celle programmée pour le sommeil du STM32WB5MMG.

Lorsque le module se réveille, la couche *BLE* s'initialise puis active l'*advertising*. S'en suit une phase où le Bluetooth est activé, caractérisée par des périodes comprises entre 7 et 8 secondes (valeurs ne correspondant à aucune constante de temps écrite ou visible dans le code) où une courte phase d'activité précède un retour à une

consommation inférieure à 1mA. Chaque début d'activité est marqué par un pic avoisinant les 30mA.

La transmission de données est invisible au regard de la consommation, aucune variation n'apparaît sur les mesures, seule l'acquisition des mesures est perceptible. Lorsque la communication prend fin, le cycle de consommation reste présent, ce qui indique que le coprocesseur M0+ ne s'éteint pas avec la commande `ble.shutdown()`. La figure ci-dessous illustre ces différentes phases de consommation.

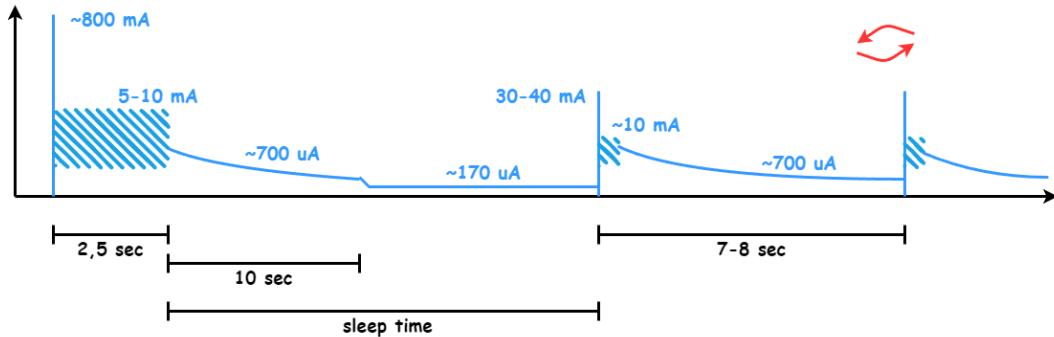


Figure 5.11 – Consommation sur pile

Dans l'état actuel, si l'on omet l'initialisation et le premier sommeil, la consommation moyenne sur un cycle de 7-8 secondes est de 1.18 mA. Le nombre de jours d'autonomie se calcule de la manière suivante :

$$D = \frac{C_{\text{pile}}}{I_{\text{moyen}}} = \frac{1200 \text{ mAh}}{1.18 \text{ mA}} \approx 1017 \text{ heures} \approx 42 \text{ jours} \quad (5.2)$$

Si le module parvenait à repasser en mode *deepsleep* avec deux cycles de communication et un *sleeptime* de 10 minutes, le courant serait calculé comme suit.

$$I_{\text{moyen}} = \frac{\sum(I_i \cdot t_i)}{\sum t_i} = \frac{I_1 t_1 + I_2 t_2 + I_3 t_3}{t_1 + t_2 + t_3} \quad (5.3)$$

Avec :

$$\begin{array}{ll} I_1 = 1.18 \text{ mA}, & t_1 = 10 \text{ s} \\ I_2 = 0.17 \text{ mA}, & t_2 = 590 \text{ s} \\ I_3 = 1.18 \text{ mA}, & t_3 = 15 \text{ s} \end{array}$$

On obtient :

$$I_{\text{moyen}} = \frac{1.18 \cdot 10 + 0.17 \cdot 590 + 1.18 \cdot 15}{10 + 590 + 15} = \frac{129.8}{615} \approx 0.211 \text{ mA} \quad (5.4)$$

Le nouveau nombre de jours d'autonomie est le suivant :

$$D = \frac{C_{\text{pile}}}{I_{\text{moyen}}} = \frac{1200 \text{ mAh}}{211 \mu\text{A}} \approx 5687 \text{ heures} \approx 237 \text{ jours} \quad (5.5)$$

L'autonomie est ainsi de **42 jours** actuellement mais pourrait passer à **237 jours** en résolvant le problème du *deepsleep*.

5.4. REVUE DES OBJECTIFS

5.4 Revue des objectifs

À la lumière des tests réalisés, il est possible d'affirmer si les objectifs fixés par le cahier des charges ont été atteints.

Objectifs	État de réalisation
Concevoir une carte électronique selon les critères suivants :	OK
1. Doit pouvoir fonctionner sur pile et/ou par alimentation filaire.	OK
2. Doit pouvoir collecter des données provenant d'un capteur.	OK
3. Doit pouvoir piloter un élément (ex : GPIO, relais, LED).	OK
4. Doit pouvoir lire une tension analogique.	OK
Programmer la carte électronique selon les critères suivants :	OK
5. Doit pouvoir communiquer sans fil.	OK
6. Doit pouvoir transmettre une donnée.	OK
7. Doit pouvoir recevoir une donnée.	OK
Programmer l'unité centrale selon les critères suivants :	OK
8. Doit pouvoir communiquer sans fil.	OK
9. Doit pouvoir transmettre une donnée.	OK
10. Doit pouvoir recevoir une donnée.	OK

Table 5.8 – Revue des objectifs

Ainsi, le cahier des charges a pu être correctement respecté. Les rubriques *Validation des fonctionnalités* et *Mesure de consommation* pointent certains éléments pouvant encore être améliorés, mais le système *Rainette* est, dans sa phase de prototype, fonctionnel.

5.5 Perspectives

Si le projet venait à être poursuivi, une série d'améliorations pourrait être envisagée.

Améliorations matérielles

- Trouver la cause empêchant de communiquer avec le capteur de luminosité ou essayer de le remplacer par une autre référence.
- Réduire les résistances du pont diviseur pour mesurer la tension de la pile afin de limiter l'influence de la diode Zener.
- Retirer le condensateur C12 du PCB pour éviter l'important pic de consommation lors des mesures.
- Développer un boîtier étanche et impactant le moins possible la communication Bluetooth afin de pouvoir mener des tests dans des conditions réelles.

Améliorations logicielles

- Fiabiliser les connexions/déconnexions BLE.
- Développer un programme d'automatisation pour le Raspberry Pi afin de mener des tests en conditions réelles.

Chapitre 6

Conclusion

Voilà qui conclut le compte rendu de ce travail de Bachelor. Il a débuté avec une première phase d'analyse ayant pour but d'identifier les besoins et les solutions techniques à disposition de ce projet. S'en est suivie la conception matérielle où le schéma électrique et le PCB auront été développés avant d'atteindre la phase de conception logicielle. La fin du développement *software* a marqué le début de la dernière partie, la mise de service et les tests, où il a pu être montré que **l'ensemble des objectifs ont été atteints.**

La carte n'est cependant pas exempte de tout défaut, le capteur de luminosité n'est pour le moment pas fonctionnel et il n'est pas encore possible d'atteindre de manière répétée une consommation acceptable avec le mode de fonctionnement sur pile. L'autonomie actuelle dans ce mode est de **42 jours** mais pourrait passer à **237 jours** en résolvant le problème lié au passage en sommeil du module. La solution développée montre néanmoins des signes de bonne conception et pourrait être reprise pour dépasser le stade de prototype sans avoir à apporter de modifications majeures. Le coût total du projet s'élève à **312.80 CHF** hors frais de livraison.

Le projet a ainsi été mené sur un total de **451 heures** sur les 420 heures planifiées, soit un dépassement de 7%. L'avancement a correctement été consigné dans un journal de bord qu'il est possible de consulter pour comprendre les différents choix et raisonnements pris, et ce en plus du présent rapport. L'ensemble des ressources de ce projet peut être consulté sur le dépôt GitHub suivant : <https://github.com/dpmlli/Rainette>.

Damien Pomelli



Bibliographie

Références générales

- [Samacsys Guide] *Altium Library Loader - Guide.* URL : <https://www.samacsys.com/altium-designer-library-instructions/>.
- [AEE] *Applied Embedded Electronics.* URL : <https://learning.oreilly.com/library/view/applied-embedded-electronics/9781098144784/>.
- [Zener Diodes] *Are Zener Diodes Unidirectional or Bidirectional.* URL : <https://forum.digikey.com/t/are-zener-diodes-unidirectional-or-bidirectional/965>.
- [ATLAS] *Atlas Scientific.* URL : <https://atlas-scientific.com/?srsltid=AfmB0op3LyVFWyACmSXTexSvUbNJLr-7BNpAVwX6ErTdmLDOjT3eoj-x>.
- [OPENAI] *Chat GPT.* URL : <https://chatgpt.com>.
- [RAINETTE] *Dépôt GitHub du projet.* URL : <https://github.com/dpmlli/Rainette>.
- [PCB TVS] *Directives de conception de PCB pour l'utilisation de diodes TVS pour la protection contre les transitoires.* URL : <https://resources.altium.com/fr/p/pcb-design-guidelines-using-tvs-diode-transient-protection>.
- [MBED DISCO] *DISCO-WB5MMG / Mbed.* URL : <https://os.mbed.com/platforms/DISCO-WB5MMG/>.
- [EURO] *Eurocircuits.* URL : <https://www.eurocircuits.com/>.
- [FIRM M0] *Firm Copro.* URL : https://github.com/STMicroelectronics/STM32CubeWB/tree/master/Projects/STM32WB_Copro_Wireless_Binaries/STM32WB5x.
- [I2C Calc] *I2C Bus Pull-Up Resistor Calculation.* URL : https://www.ti.com/lit/an/slva689/slva689.pdf?ts=1748334312212&ref_url=https%253A%252F%252Fwww.google.com%252F.
- [GEX] *La Ferme Aquaponique du Pays de Gex / Versonnex.* URL : <https://www.lafermeaquaponique.com>.
- [PCB ESD] *Le guide de la protection ESD dans la conception de PCB.* URL : <https://resources.altium.com/fr/p/beginners-guide-esd-protection-circuit-design-pcbs>.
- [MBED] *MBed OS.* URL : <https://os.mbed.com/mbed-os/>.

BIBLIOGRAPHIE
RÉFÉRENCES GÉNÉRALES

- [BLE EX] *Mbed OS BLE examples.* URL : <https://github.com/ARMmbed/mbed-os-example-ble>.
- [MBED UTILS] *Mbed OS BLE utils.* URL : <https://github.com/ARMmbed/mbed-os-ble-utils/tree/master>.
- [NOVASPEX] *Novaspex Inc.* URL : <https://www.novaspex.com/culture-en-serre/>.
- [PCBWAY] *PCBWay.* URL : <https://www.pcbway.com/>.
- [PiPonics] *PiPonics.* URL : <https://github.com/karaulj/PiPonics>.
- [ponics32] *ponics32.* URL : <https://github.com/karaulj/ponics32>.
- [PPK2 SW] *PPK2 Software.* URL : <https://www.nordicsemi.com/Products/Development-hardware/Power-Profiler-Kit-2/Download>.
- [SATURN] *Saturn PCB.* URL : <https://saturnpcb.com/>.
- [ST MCU] *ST MCU.* URL : <https://www.st.com/en/microcontrollers-microprocessors.html>.
- [Tag-Connect] *ST-Link V3 SET Cable Selection & Installation.* URL : <https://www.tag-connect.com/debugger-cable-selection-installation-instructions/stlink-v3set>.
- [ST32WBXM] *ST32WBXM.* URL : <https://www.st.com/en/microcontrollers-microprocessors/stm32wbxm-modules.html>.
- [DISCO] *STM32WB5MMG - Disco Kit.* URL : <https://www.st.com/en/evaluation-tools/stm32wb5mm-dk.html>.
- [MODULE] *STM32WB5MMG - Ultra-low-power Module.* URL : <https://www.st.com/en/microcontrollers-microprocessors/stm32wb5mmg.html>.
- [TARGETS] *Targets STM32WB.* URL : https://github.com/ARMmbed/mbed-os/tree/master/targets/TARGET_STM/TARGET_STM32WB.
- [ULTRA LIB] *Ultra Librarian - Altium.* URL : https://app.ultralibrarian.com/content/help/?altium_designer.htm.
- [USB Wiki] *USB Type-C.* In : URL : https://fr.wikipedia.org/w/index.php?title=USB_Type-C&oldid=224388627.
- [BLE STACK] *Wireless Stack.* URL : https://wiki.st.com/stm32mcu/wiki/Connectivity:STM32WB_BLE_Wireless_Stack.
- [AN BLE] *Wireless Stack.* URL : https://www.st.com/resource/en/application_note/an5270-stm32wb-bluetooth-low-energy-wireless-interface-stmicroelectronics.pdf.

Bibliographie

Datasheets

- [CONN] *1729128 / Phoenix Contact.* URL : <https://www.phoenixcontact.com/fr-ch/produits/borne-de-circuit-imprime-mkdsn-15-2-508-1729128>.
- [DMC3071] *DMC3071LVT-13 Transistor.* URL : https://eu.mouser.com/ProductDetail/Diodes-Incorporated/DMC3071LVT-13?qs=wUXugUrL1qxBKRufN84dSg%3D%3D&srltid=AfmB0orpyj9QzCxFE3L_3Qg4sPQM_EBSOMvnnQ_Gkl3i0JJbLKZtuFK0.
- [MOSFET] *G3VM-61VY3TR05 Omron Electronics.* URL : <https://www.mouser.ch/ProductDetail/Omron-Electronics/G3VM-61VY3TR05?qs=F5EMLAvA7IAiM3jJSgsHtA%3D%3D>.
- [INDUCTOR] *Inductors - LPS3015.* URL : <https://www.mouser.com/datasheet/2/597/lps3015-270734.pdf?srltid=AfmB0oqLRXhwj64BeGDlULQmEiLCmNBQeuPtMatKbFa-3mZ52RC220ry>.
- [TVS] *PESD3V3L1BSL (ESD protection device).* URL : <https://www.nexperia.com/product/PESD3V3L1BSL>.
- [PH] *pH Surveyor / Atlas Scientific.* URL : <https://files.atlas-scientific.com/Surveyor-pH-datasheet.pdf>.
- [PPK2] *Power Profiler Kit II.* URL : <https://nsscprodmedia.blob.core.windows.net/prod/software-and-other-downloads/product-briefs/power-profiler-kit-ii-pbv10.pdf>.
- [SHT41] *SHT41 - Digital humidity and temperature sensor.* URL : <https://sensirion.com/products/catalog/SHT41>.
- [ST-LINK] *STLINK-V3SET.* URL : <https://www.st.com/en/development-tools/stlink-v3set.html>.
- [55VG] *STM32WB55VG.* URL : <https://www.st.com/en/microcontrollers-microprocessors/stm32wb55vg.html>.
- [ADAPTER] *TC2050 Adapter.* URL : <https://www.tag-connect.com/wp-content/uploads/bst-pdf-manager/2021/02/TC2050-ARM2010-2021.pdf>.
- [TC2050] *TC2050-IDC / Tag-Connect.* URL : https://www.tag-connect.com/wp-content/uploads/bst-pdf-manager/TC2050-IDC_Datasheet_7.pdf.

BIBLIOGRAPHIE
DATASHEETS

- [TEMP] *Temp Surveyor / Atlas Scientific.* URL : <https://files.atlas-scientific.com/Surveyor-Temp-datasheet.pdf>.
- [TPS63001] *TPS63001.* URL : <https://www.ti.com/lit/ds/symlink/tps63001.pdf?ts=1749626693967>.
- [TSR] *TSR 1-2450E / Traco Power.* URL : <https://www.tracopower.com/ch/fra/model/tsr-1-2450e>.
- [VEML6030] *Veml6030 / Vishay.* URL : <https://www.vishay.com/docs/84366/veml6030.pdf>.
- [XC8102] *xc8102 Load Switch.* URL : <https://product.torexsemi.com/system/files/series/xc8102.pdf>.

Annexes

Annexe A

Général

A.1 Codes source

L'ensemble des codes développés au cours de ce projet peuvent être consultés sur le dépôt GitHub suivant : <https://github.com/dpmlli/Rainette>

Les annexes suivantes se succèdent sans délimitation, leurs intitulés sont listés ci-dessous. Les autres ressources du projet peuvent, elles aussi, être trouvées sur le dépôt GitHub mentionné.

A.2 Schéma électrique Rainette

A.3 Devis PCBWay

A.4 Planification

A.5 Schéma électrique STM32WB5MM-DK

1

2

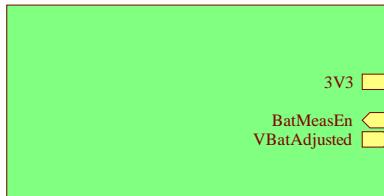
3

4

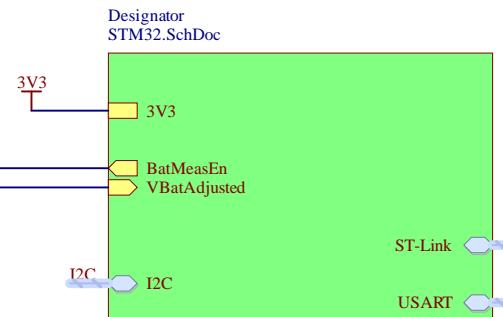
A

A

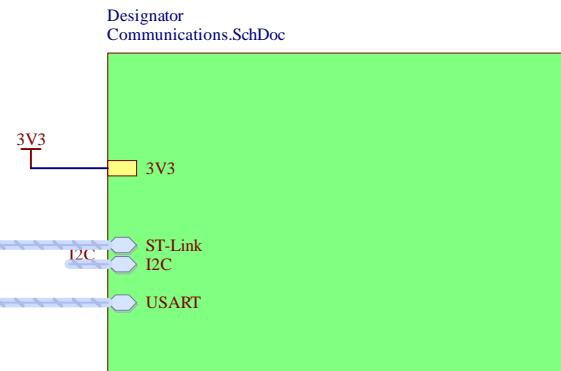
Designator
Power_Supply.SchDoc



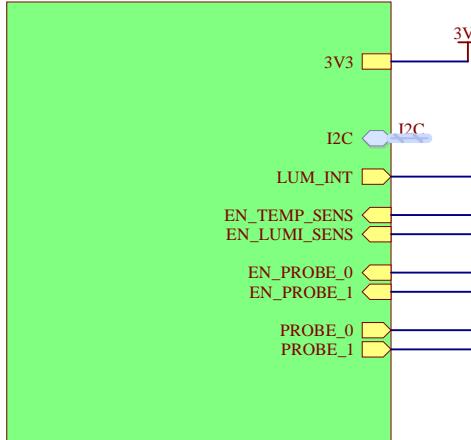
Designator
STM32.SchDoc



Designator
Communications.SchDoc



Designator
Analog_Sensors.SchDoc



UI
GPIO
GPIO_DEVICES

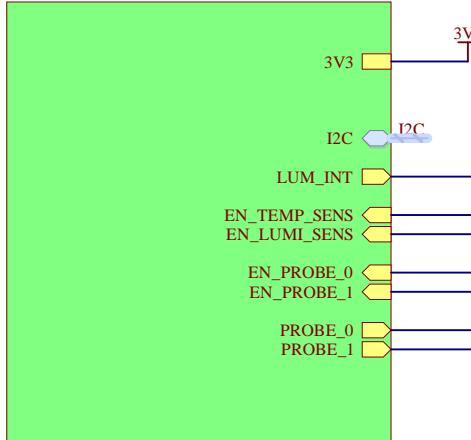
Designator
Digital_GPIO.SchDoc



B

B

Designator
Analog_Sensors.SchDoc



UI
GPIO
GPIO_DEVICES

Designator
Digital_GPIO.SchDoc



C

C

D

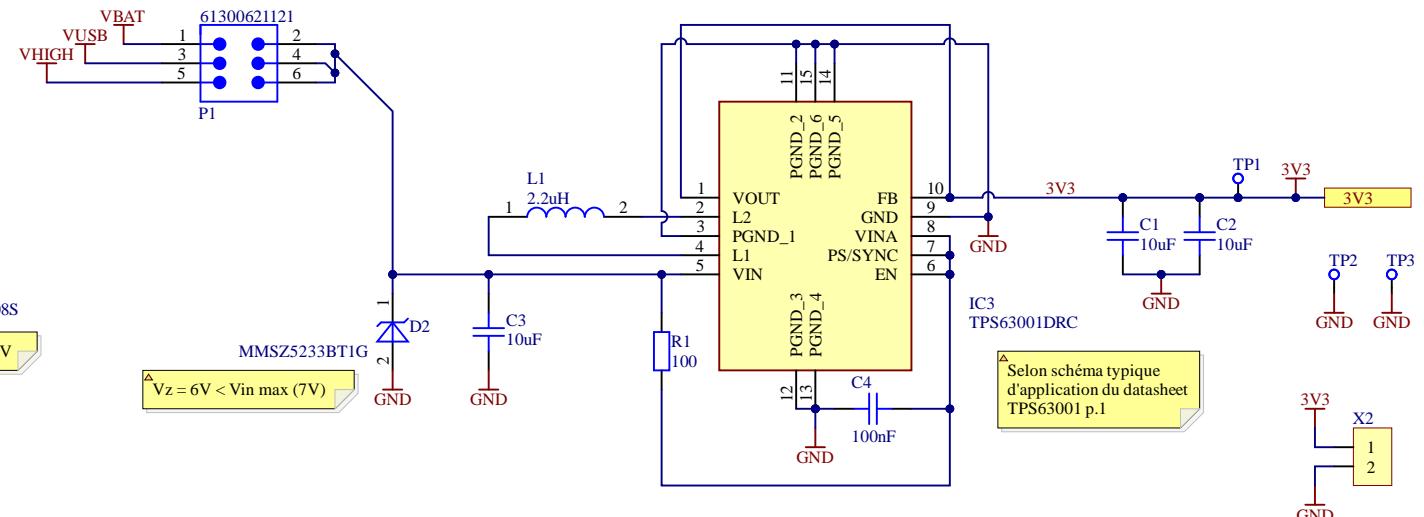
D

1

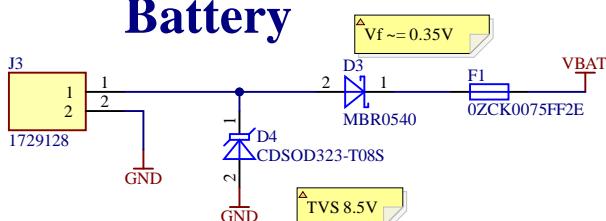
2

3

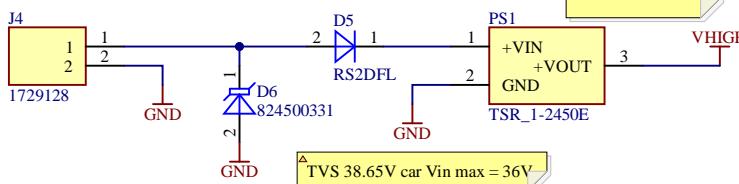
4



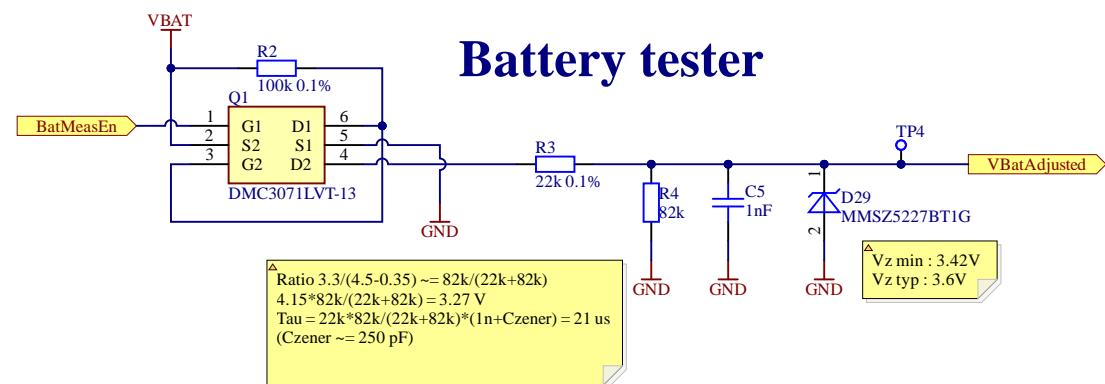
Battery



High DC Voltage



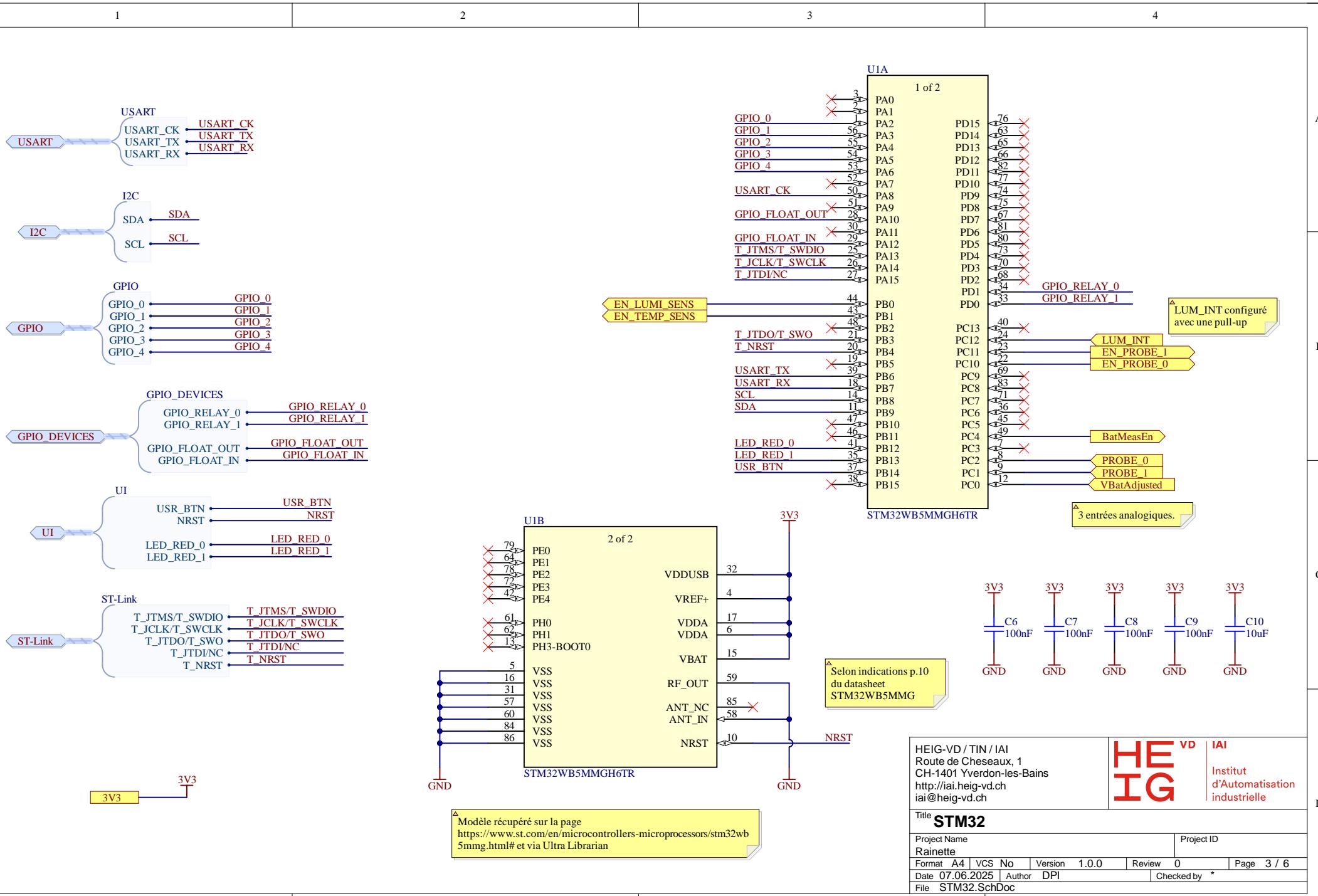
Battery tester

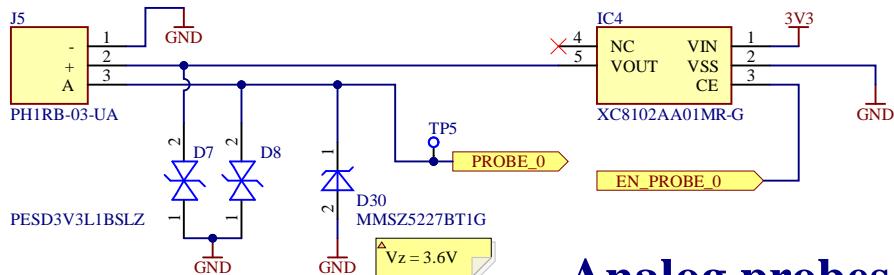


HEIG-VD / TIN / IAI
Route de Cheseaux, 1
CH-1401 Yverdon-les-Bains
<http://iai.heig-vd.ch>
iai@heig-vd.ch

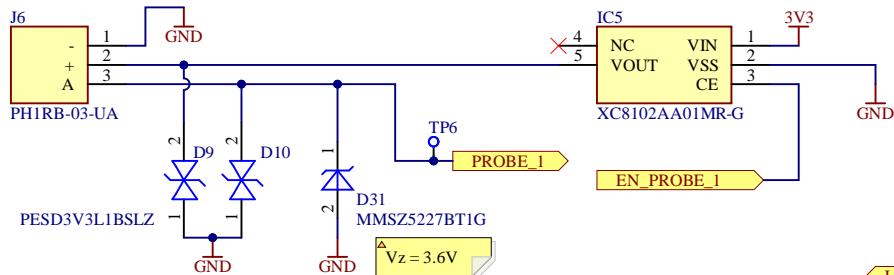
HEIG VD IAI
Institut
d'Automatisation
industrielle

Title		Power Supply					
Project Name				Project ID			
Rainette							
Format	A4	VCS	No	Version	1.0.0	Review	0
Date	07.06.2025	Author	DPI	Checked by *			
File	Power Supply.SchDoc						

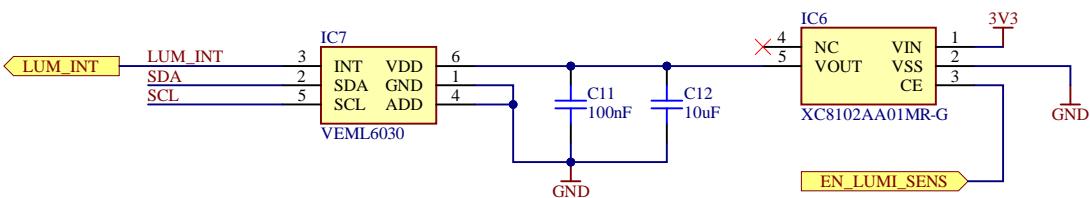




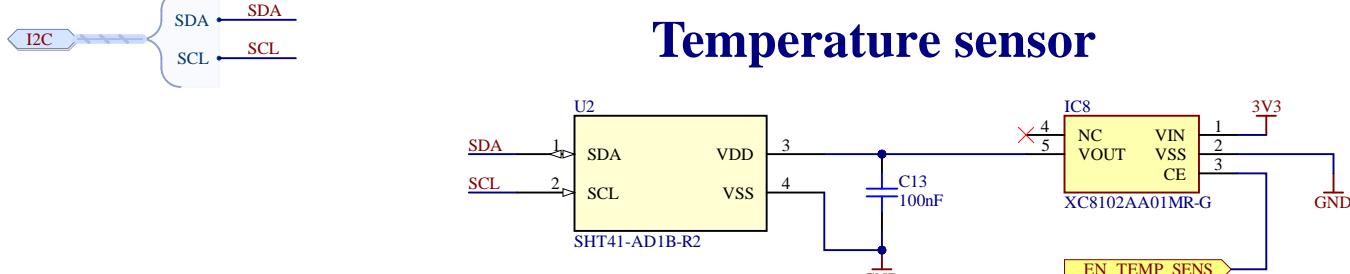
Analog probes



Luminosity sensor



Temperature sensor



▲ Récupéré sur Ultra Librarian.
Footprint modifié selon le datasheet, il fallait retiré le pad 5.

<p>HEIG-VD / TIN / IAI Route de Cheseaux, 1 CH-1401 Yverdon-les-Bains http://iai.heig-vd.ch iai@heig-vd.ch</p>	 VD IAI Institut d'Automatisation industrielle								
<p>Title Analog Sensors</p>									
Project Name Rainette	Project ID								
Format	A4	VCS	No	Version	1.0.0	Review	0	Page	4 / 6
Date 07.06.2025 Author DPI						Checked by *			
File Analog_Sensors.SchDoc									

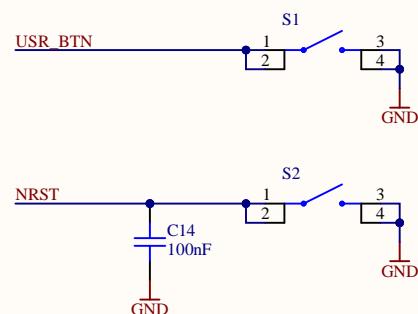
1

2

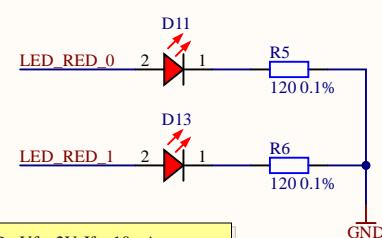
3

4

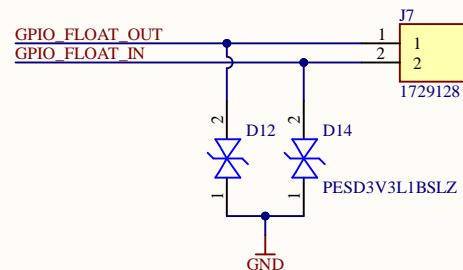
Buttons



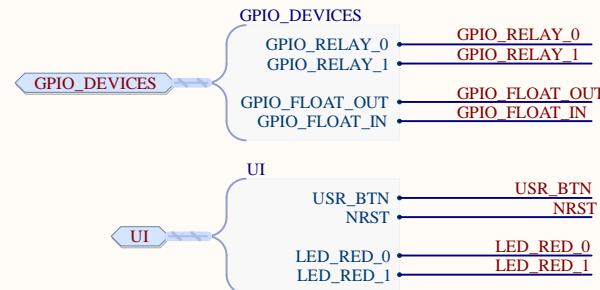
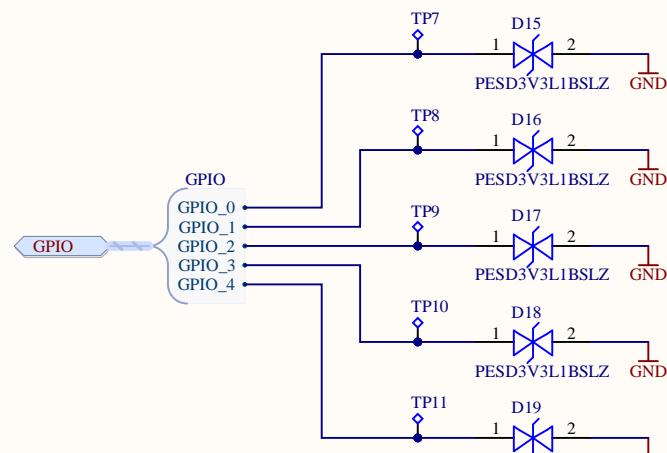
Leds



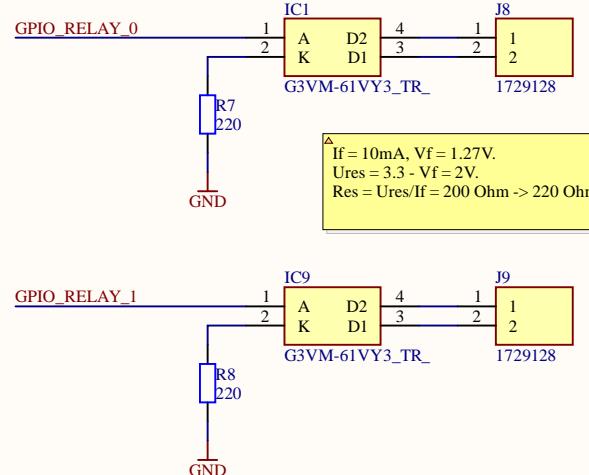
Float



GPIO



MOSFET Relays



Title		
Size	Number	Revision
A4		
Date: 6.07.2025	Sheet of	
File: C:\Users\.\Digital_GPIO.SchDoc		Drawn By:

1

2

3

4

A

B

C

D

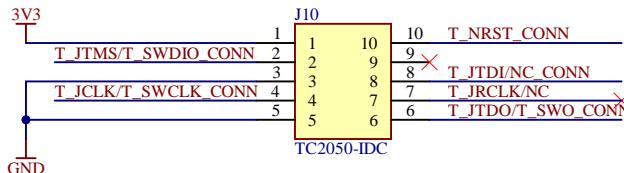
A

B

C

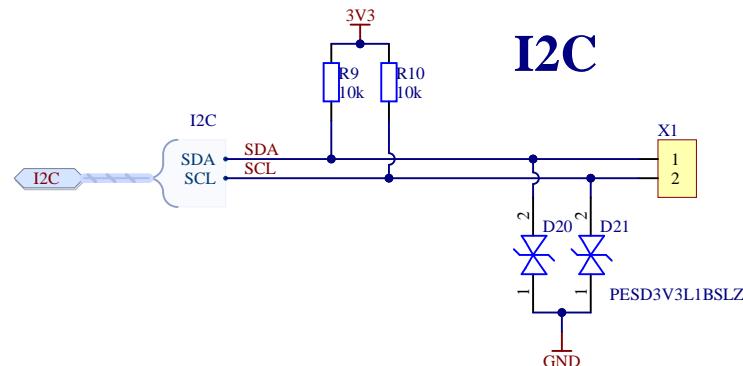
D

ST-Link

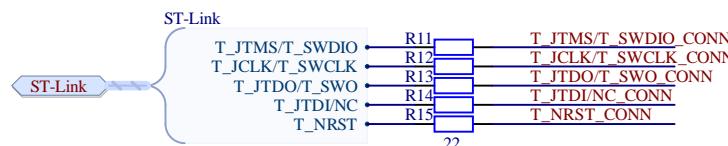


Optional loopback

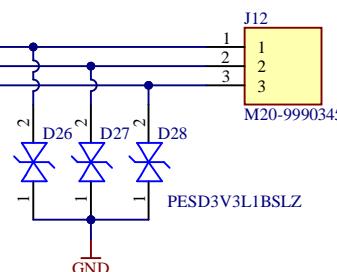
Pins 3-12 of the STDC14 connector map to TC2050 footprint pins 1-10. STDC14 pins 1,2,13&14 are not connected. TC2050 Pin n connects to STDC14 Pin n+2 (source : <https://www.tag-connect.com/product/tc2050-idc-050-stdc14>). Donc Virtual COM port (VCP) pas dispo



I2C



USART



HEIG-VD / TIN / IAI
Route de Cheseaux, 1
CH-1401 Yverdon-les-Bains
<http://iai.heig-vd.ch>
iai@heig-vd.ch

HEIG VD IAI
Institut
d'Automatisation
industrielle

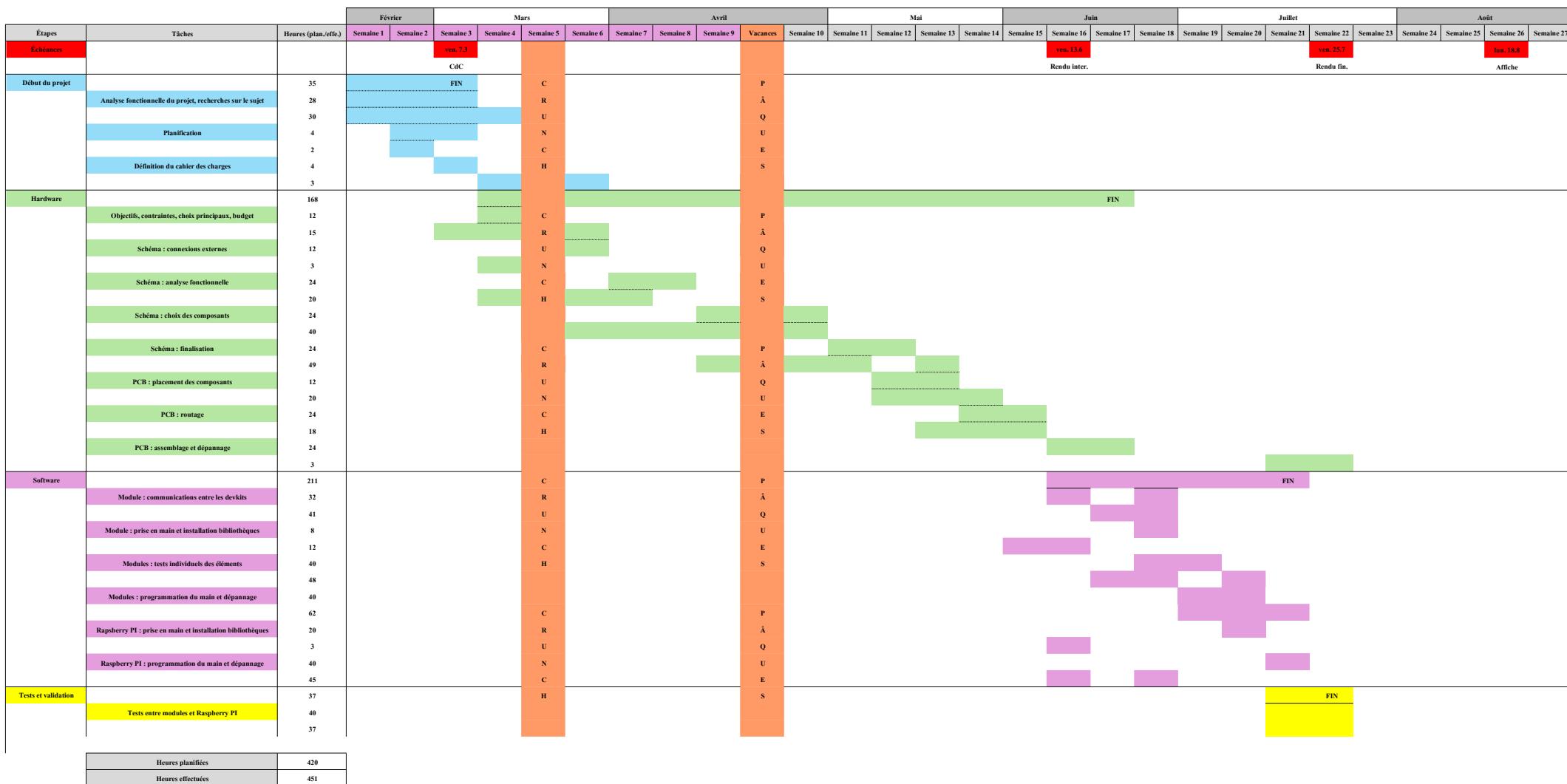
Title **Communications**

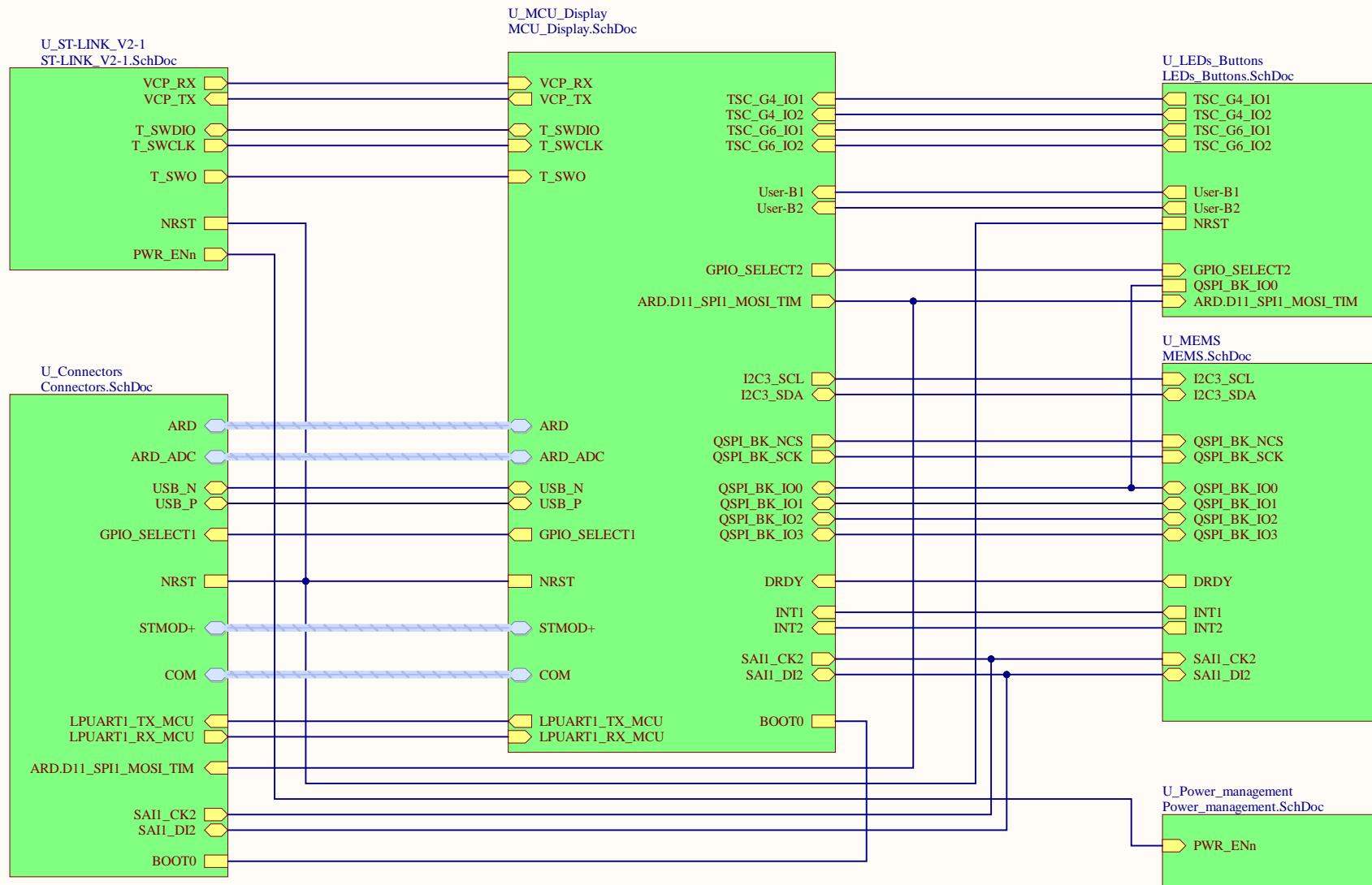
Project Name	Project ID
Rainette	
Format A4	VCS No
Date 07.06.2025	Version 1.0.0
Author DPI	Review 0
Checked by *	
File Communications.SchDoc	

Item #	Designator	*Qty	Manufacturer	*Mfg Part #	Vendor Part #	*Package/Footprint	Mounting Type	Customer Item No.	*Unit Price(3 sets)	*Total	*Delivery Time	Actual Purchase Mfg Part #	PCBWay Note	PCBWay Update
1	C1, C2, C3, C4	5	Samsung Electr	CL31B106KBHNNNE	187-CL31B106KBHNNNE	CAPC3216X18N 1206MLCC	Surface Mount, MLCC	CMP-0001778-4	\$0.172	\$2.580				
2	C4, C6, C7, C8	8	Murata Electron	GCM155L8EH104KE07	81-GCM155L8EH104KE7D	CAPC1005X05N 1204MLCC	Surface Mount, MLCC	CMP-0003371-1	\$0.063	\$1.512				
3	C5	1	Murata Electron	GCM155R71H102MA37	81-GCM155R71H102MA7D	CAPC1005X05N 1204MLCC	Surface Mount, MLCC	CMP-0003402-1	\$0.063	\$0.189				
4	D1, D4	2	Bourns	CDSOD323-T08SC	652-CDSOD323-T08SC	SODFL250X110-2N	Surface Mount	CMP-0002421-1	\$0.704	\$4.224				
5	D2	1	onsemi	MMSZ5233BT1G	MMSZ5233BT1GOSCT-ND	SOD3717X135	Surface Mount	CMP-0005477-4	\$0.170	\$0.510				
6	D3	1	Micro Commerc	MBR0540-TP	833-MBR0540-TP	SODFL371X135-2N	Surface Mount	CMP-0001774-4	\$0.279	\$0.837				
7	D5	1	Taiwan Semico	RS2DFL	1801-RS2DFLCT-ND	SODFL375X120-2N	Surface Mount	CMP-0002644-1					[DNP and do not provide];	
8	D6	1	Murata Electron	GCM155R71H102MA37	81-GCM155R71H102MA7D	DIOM4227X25N 1204MLCC	Surface Mount, MLCC	CMP-0005814-3					[DNP and do not provide];	
9	D7, D8, D9, D10	16	Nexperia	PESD3V3L1BSLZ	771-PESD3V3L1BSLZ	PESD3V3L1BSLZ	Surface Mount	PESD3V3L1BSLZ	\$0.034	\$1.632				
10	D11, D13	2	LITEON	LTST-C171KRKT	859-LTST-C171KRKT	LTST-C171 12085 (201)	Surface Mount	CMP-0002115-5	\$0.111	\$0.666				
11	D29, D30, D31	3	Wurth Elektronik	824500331	710-824500331	SOD3717X135		CMP-0005471-4					[DNP and do not provide];	
12	F1	1	Bel Fuse	0ZCK0075FF2E	530-0ZCK0075FF2E	RESC2012X05N		CMP-0002366-7					[DNP and do not provide];	
13	IC1, IC9	2	Omron Electron	G3VM-61VY3(TR)	653-G3VM-61VY3TR	G3VM61VY3TR		G3VM-61VY3_TR_	\$1.174	\$7.044				
14	IC3	1	Texas Instrume	TPS63001DRC		TPS63002DRCT		TPS63001DRC	\$1.509	\$4.527		TPS63001DRCR		
15	IC4, IC5, IC6	4	Torex	XC8102AA01MR-G	865-XC8102AA01MR-G	SOT95P280X130-5N	Surface Mount	XC8102AA01MR-G	\$0.436	\$5.232				
16	IC7	1	Vishay	VEML6030	78-VELM6030	VEML6030		VEML6030	\$0.877	\$2.631				
17	J1	1	Same Sky	UJC-HP-G-5-SMT-TR	179-UJCHPG5SMTR	UJCHPG5SMTR		UJC-HP-G-5-SMT-TR					[DNP and do not provide];	
18	P1	1	Wurth Elektronik	61300621121	732-5295-ND	61300621121	Through Hole	CMP-0002595-4	\$0.704	\$2.112				
19	J3, J4, J7, J8	5	Phoenix Contact	1729128	651-1729128	1729128	Through Hole	1729128	\$0.586	\$8.790				
20	J5, J6	2	Adam Tech	PH1RB-03-UA	737-PH1RB-03-UA	HDRRA3W64P0X254_1X3	Through Hole, Right Angle	PH1RB-03-UA					[DNP and do not provide];	
21	J10	1				TC2050-IDC		TC2050-IDC				TC2050-IDC	[DNP and do not provide];	
22	J12	1	Harwin	M20-9990345	855-M20-9990345	HDRV3W64P0X254_1X3	Through Hole	M20-9990345	\$0.403	\$1.209				
23	L1	1	COILCRAFT	LPS3015-222MLB	994-LPS3015-222MLB	LPS3015		LPS3015-222MLB	\$1.995	\$5.985				
24	PS1	1	Traco Power	TSR-1-2450E	495-TSR1-2450E	TSR12450E	Through Hole	TSR_1-2450E					[DNP and do not provide];	
25	Q1	1	Diodes Incorpor	DMC3071LVT-13	621-DMC3071LVT-13	SOT95P280X100-6N	Surface Mount	DMC3071LVT-13	\$0.455	\$1.365				
26	R1	1	YAGEO	RL0402FR-070R1L	603-RL0402FR070R1L	RES21005X04N 1204	Through Hole	CMP-0004701-2	\$0.139	\$0.417				
27	R2	1	YAGEO	RT0603BRD07100KL	RT0603BRD07100KL	RES21008X04N 1206	Through Hole	RES21008X04N 1206	\$0.147	\$0.441				
28	R3	1	YAGEO	RT0402BRD0722KL	603-RT0402BRD0722KL	RES21005X04N 1204	Through Hole	RES21005X04N 1204	\$0.147	\$0.441				
29	R4	1	YAGEO	RC1206FR-0782KL	603-RC1206FR-0782KL	RES21008X06N 1206	Through Hole	RES21008X06N 1206	\$0.050	\$0.150				
30	R5, R6	2	YAGEO	RT0805BRD07120RL	603-RT0805BRD07120RL	RES21005X04N 1204	Through Hole	RES21005X04N 1204	\$0.147	\$0.882				
31	R7, R8	2	YAGEO	RC0603FR-07220RL	603-RC0603FR-07220RL	RES21008X04N 1206	Through Hole	RES21008X04N 1206	\$0.021	\$0.126				
32	R9, R10	2	YAGEO	RT0603BRD0710KL	603-RT0603BRD0710KL	RES21008X04N 1206	Through Hole	RES21008X04N 1206	\$0.147	\$0.882				
33	R11, R12, R13	5	YAGEO	RC0603FR-0722RL	603-RC0603FR-0722RL	RES21008X04N 1206	Through Hole	RES21008X04N 1206	\$0.011	\$0.165				
34	S1, S2	2	C&K	PTS645SL43SMTR92L	CKN10880CT-ND	PTS645SL43SMTR92LFS	Surface Mount	CMP-00001-2	\$0.352	\$2.112				
35	TP7, TP8, TP9	5	Keystone Electron	5117	534-5117	5117	Through Hole	CMP-0002617-2	\$0.351	\$5.265				
36	U1	1	STMicroelectro	STM32WB5MMGH6TR	511-STM32WB5MMGH6TR	SIP-LGA86_STM	Surface Mount	STM32WB5MMGH6	\$15.899	\$47.697				
37	U2	1	Sensirion	SHT41-AD1B-R2	403-SHT41-AD1B-R2	DFN4_SHT41_SEN		SHT41-AD1B-R2	\$2.013	\$6.039				
38	X1, X2	2	Adam Tech	PH1-02-UA	737-PH1-02-UA	CONN HEADER VERT 2P	Through Hole	CMP-0002184-3					[DNP and do not provide];	

Component Cost	\$115.66
Assembly Cost	\$29.00
PCB Cost	\$25.97
V0 Member 0% off	\$0.00
All Total 3 Sets	\$170.63

Planification TB Pomelli

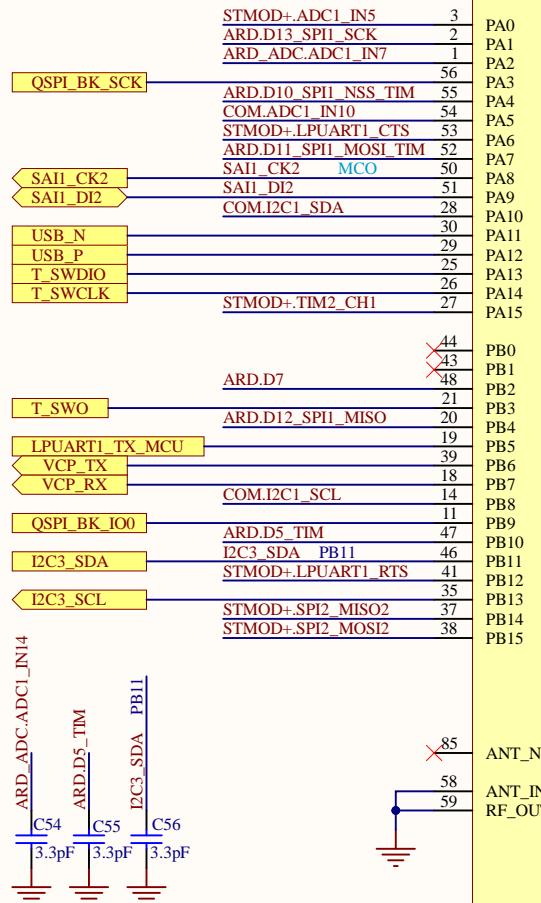




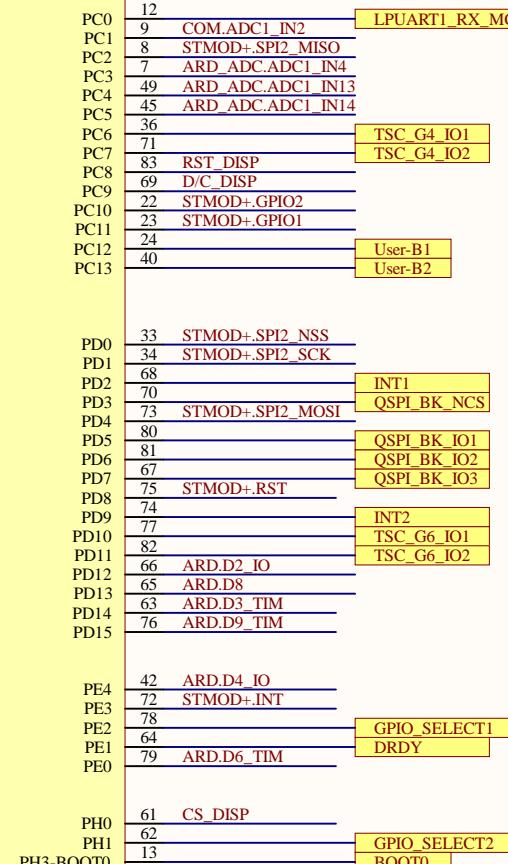
Title: Top	
Project: STM32WB5MM Discovery Kit	
Variant: WB5MM	
Revision: B-01	Reference: MB1292
Size: A4	Date: 27-Nov-20
Sheet: 2 of 8	



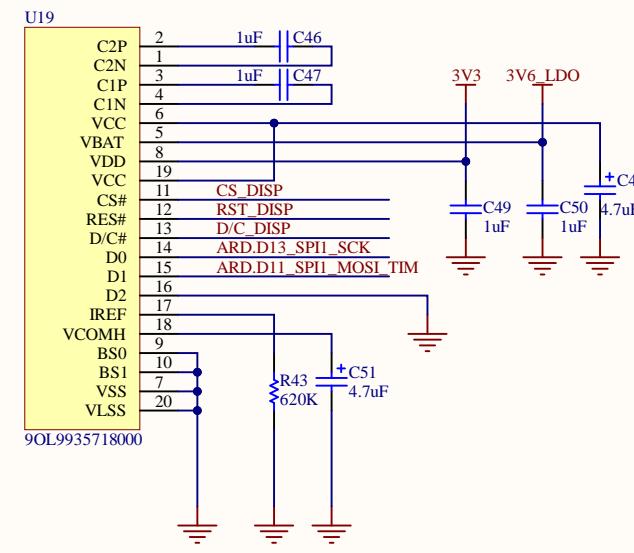
MCU



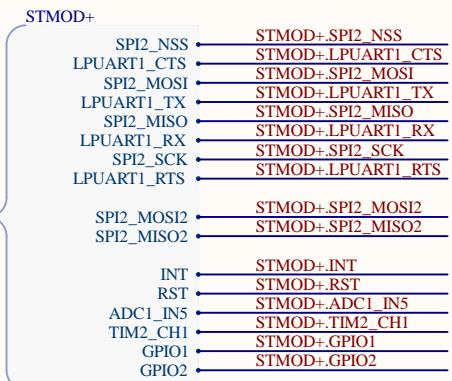
U1A



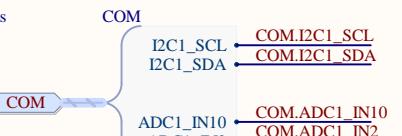
OLED DISPLAY



STMOD+



Common signals



Title: Microcontroller and OLED display

Project: STM32WB5MM Discovery Kit

Variant: WB5MM

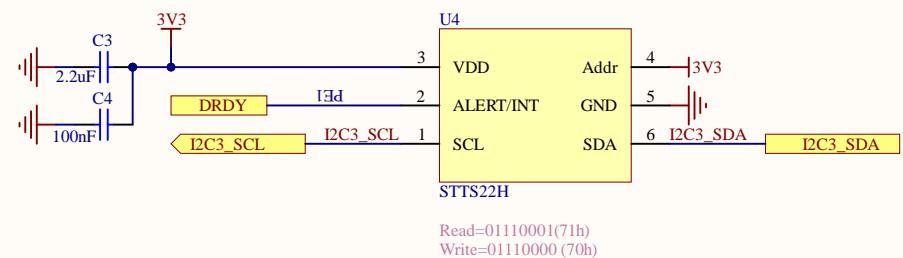
Revision: B-01 Reference: MB1292

Size: A4 Date: 27-Nov-20

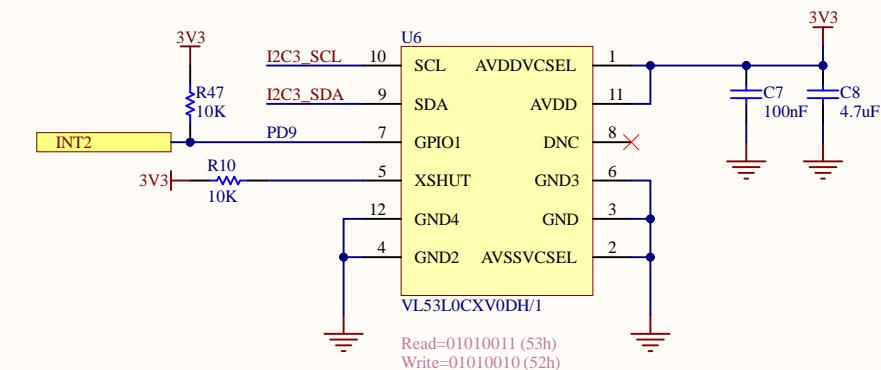


Sheet: 3 of 8

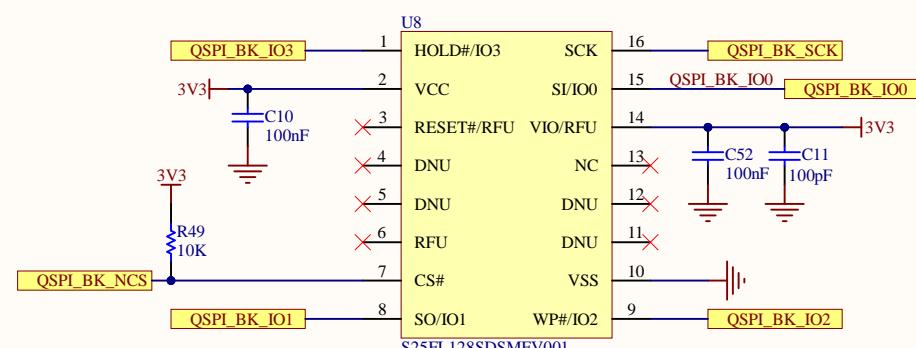
Temperature sensor



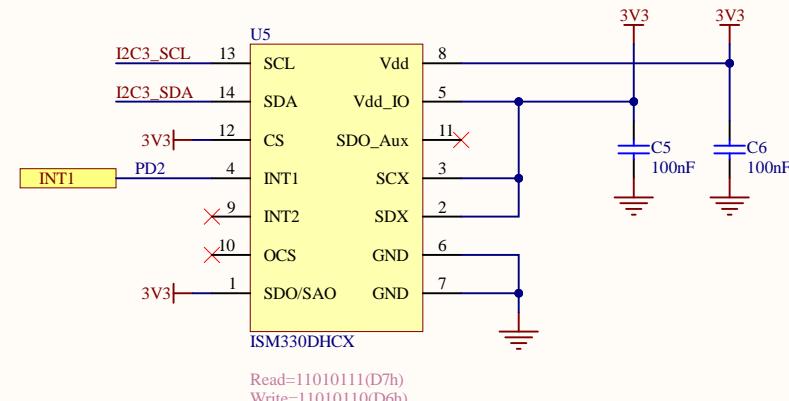
Time-of-Flight ranging and gesture detection sensor



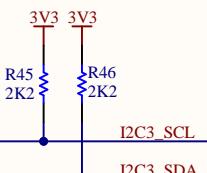
Quad-SPI



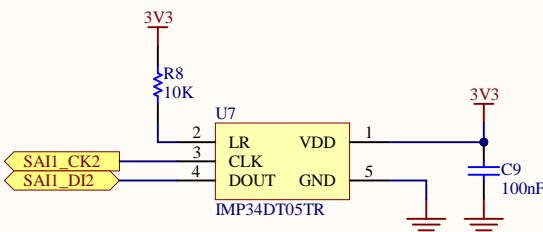
3D accelerometer and 3D gyroscope



I_{C3} Pull-up



MEMS micro



Title: MEMS sensors and Quad-SPI

Project: STM32WB5MM Discovery Kit

Variant: WB5MM

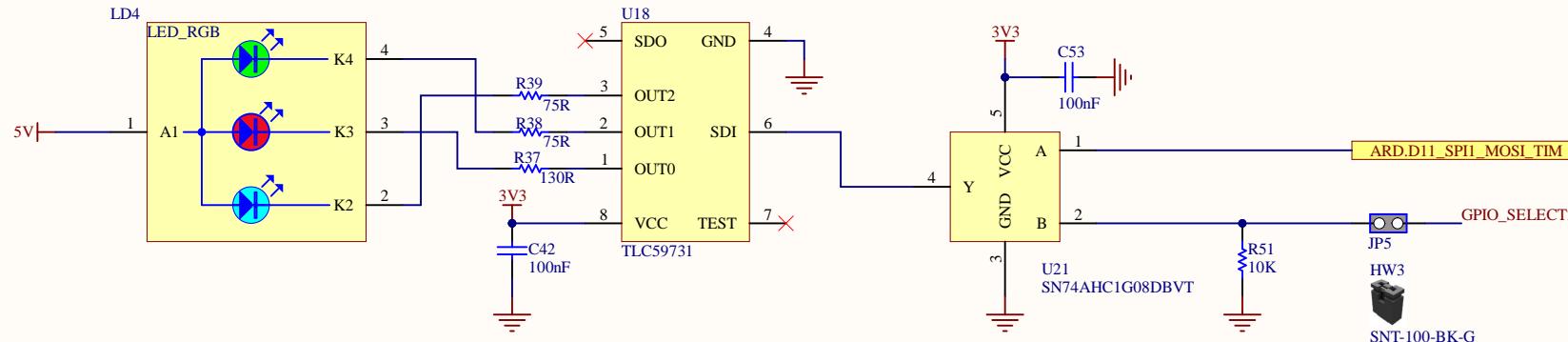
Revision: B-01 Reference: MB1292

Size: A4 Date: 27-Nov-20

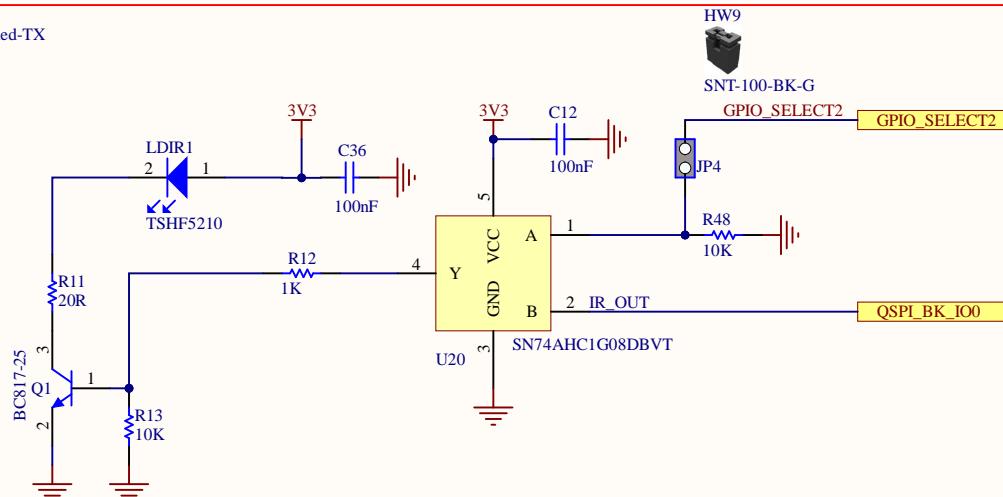
Sheet: 4 of 8



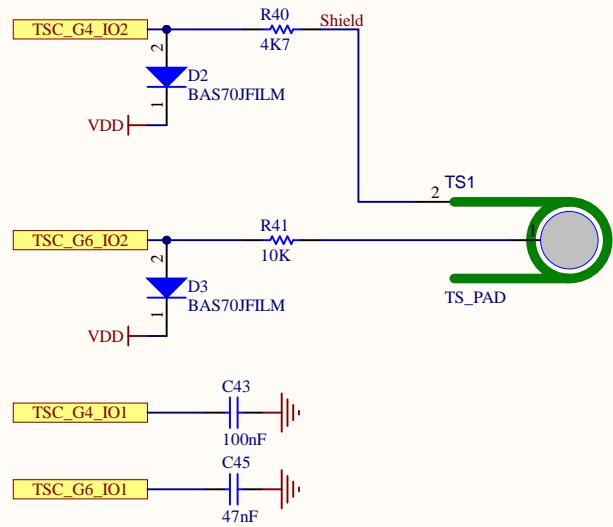
LED RGB



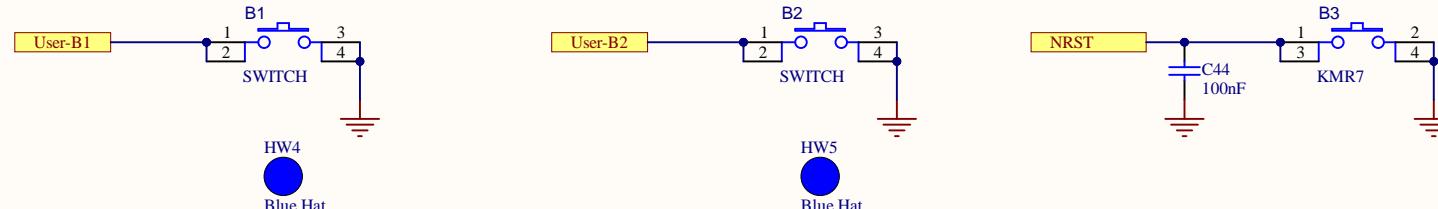
InfraRed-TX



Touchkey



Users buttons and reset button



Title: LEDs, touchkey and push-buttons

Project: STM32WB5MM Discovery Kit

Variant: WB5MM

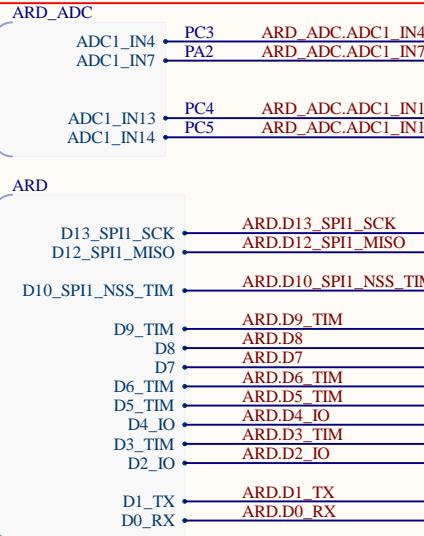
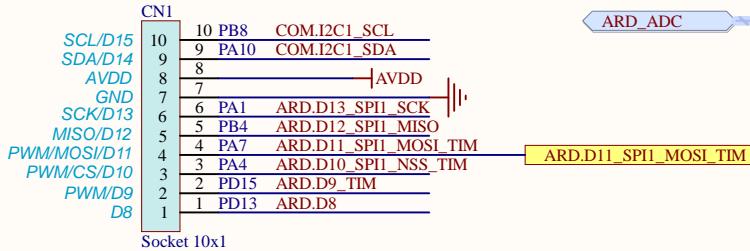
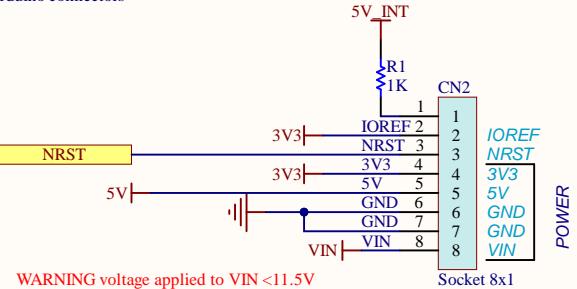
Revision: B-01 Reference: MB1292

Size: A4 Date: 27-Nov-20

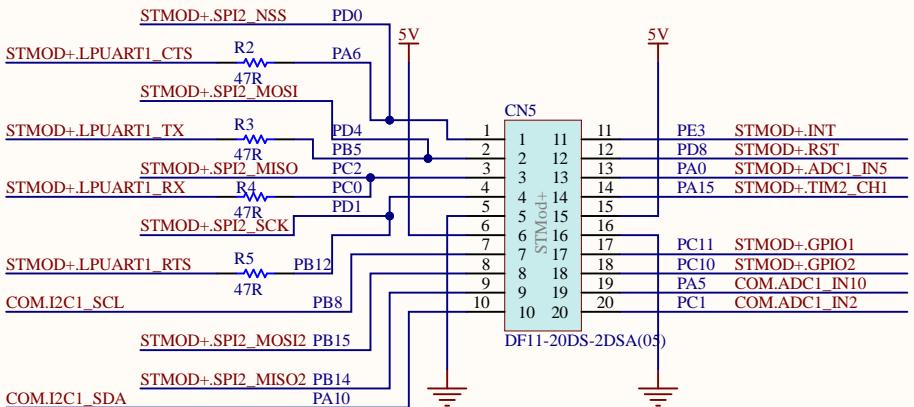
Sheet: 5 of 8



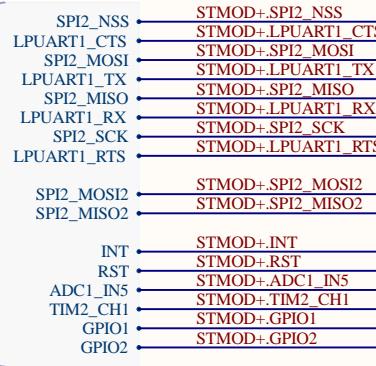
Arduino connectors



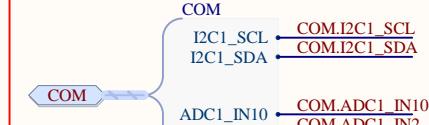
STMOD+



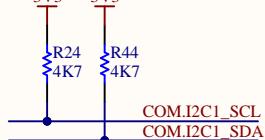
STMOD+



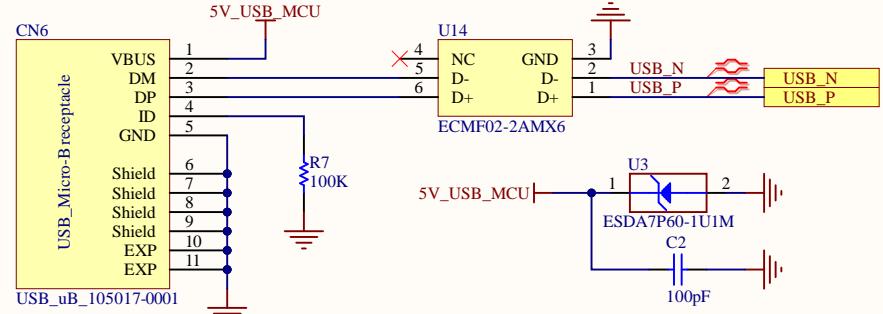
Common signals



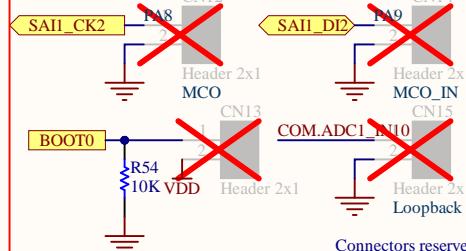
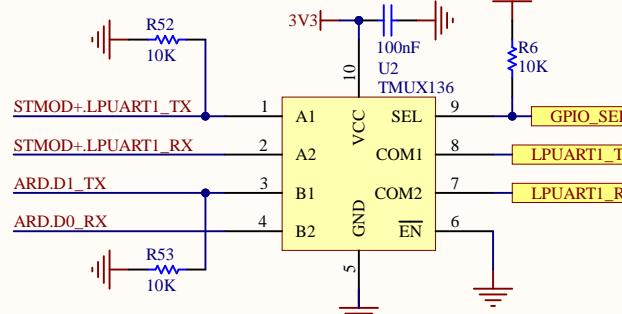
I2C1 Pull-up



USB User



LPUART switch



Title: Connectors

Project: STM32WB5MM Discovery Kit

Variant: WB5MM

Revision: B-01

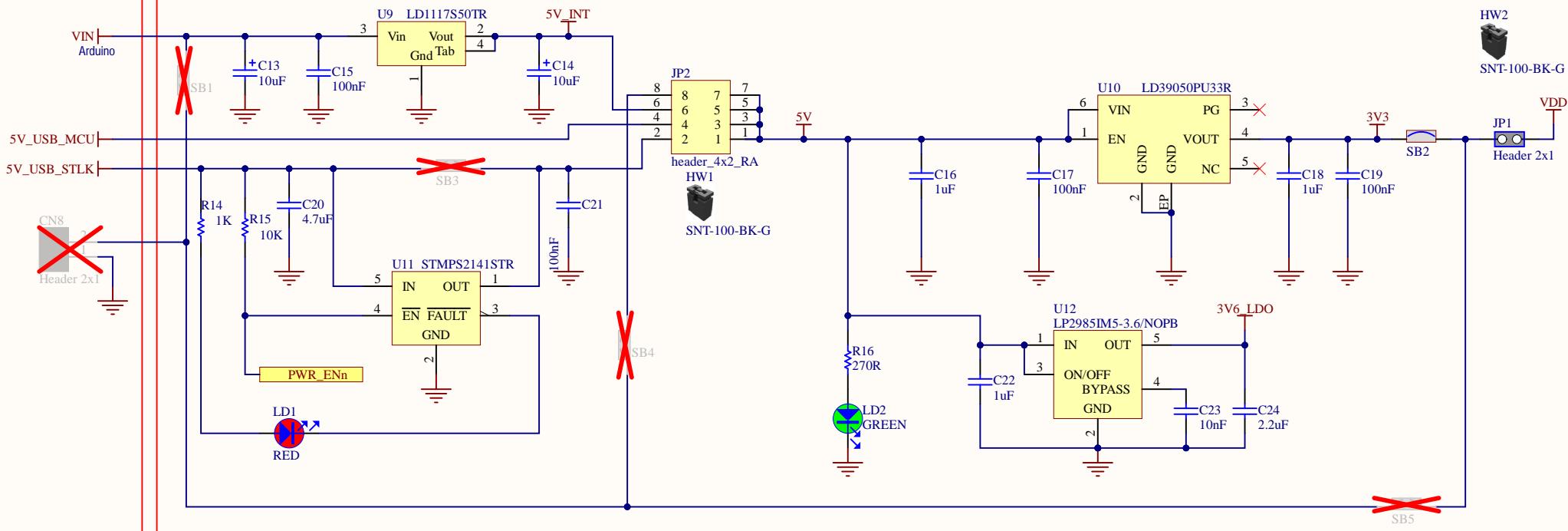
Reference: MB1292

Size: A4 Date: 27-Nov-20

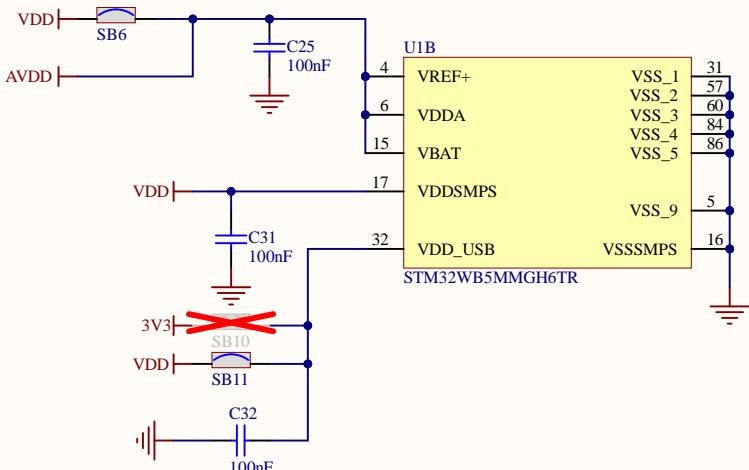
Sheet: 6 of 8



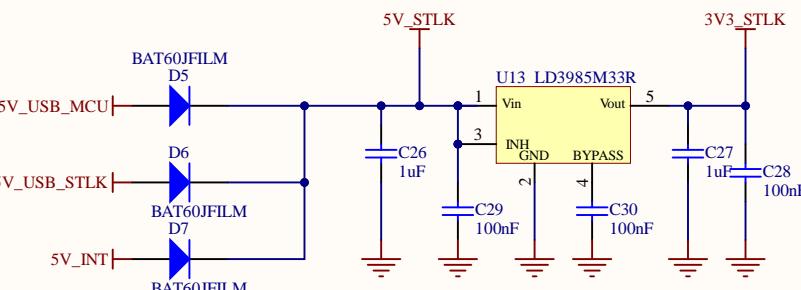
Supply sources



MCU and SMPS supply domain

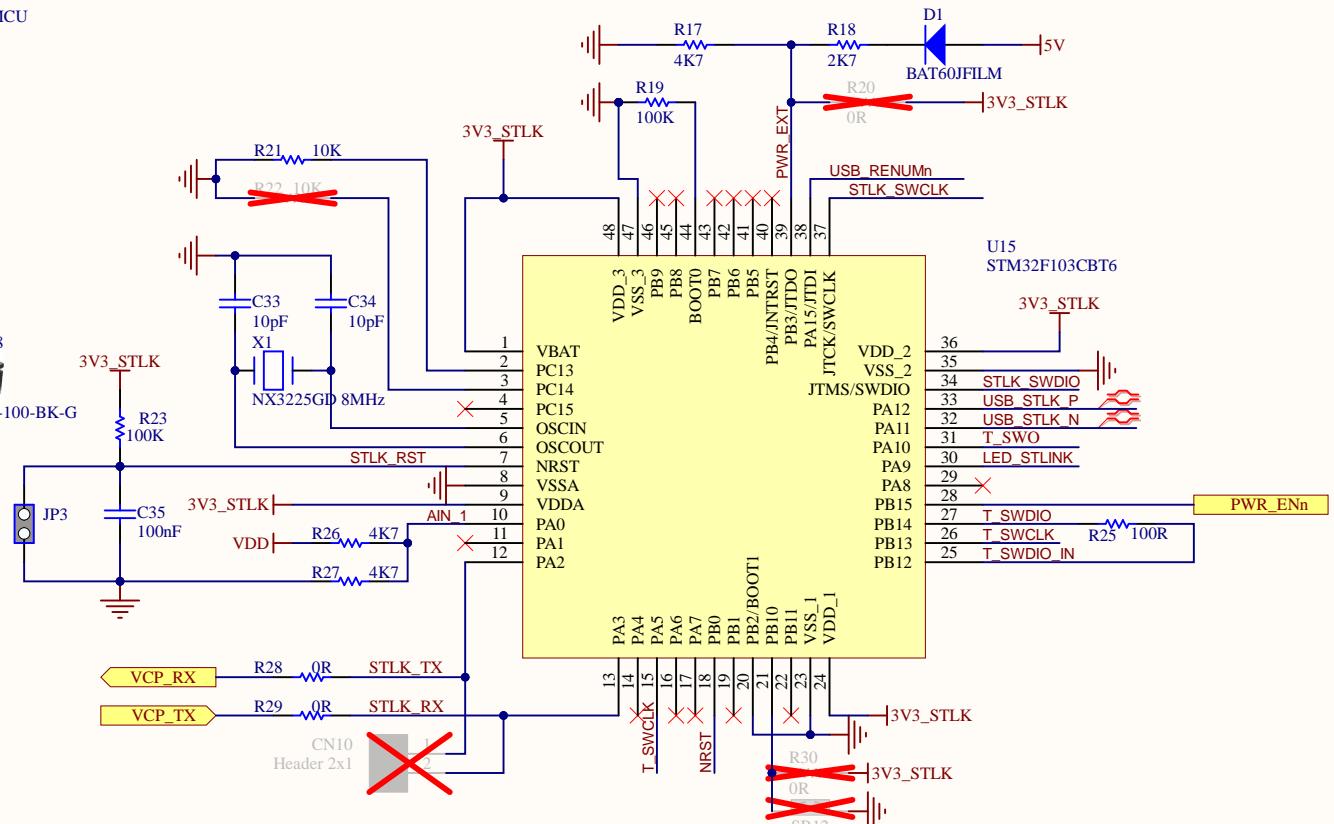


3V3 LDO dedicated to ST-Link

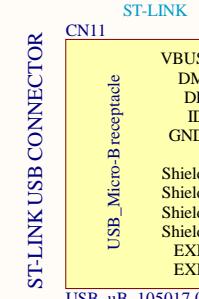




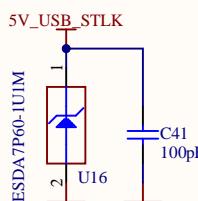
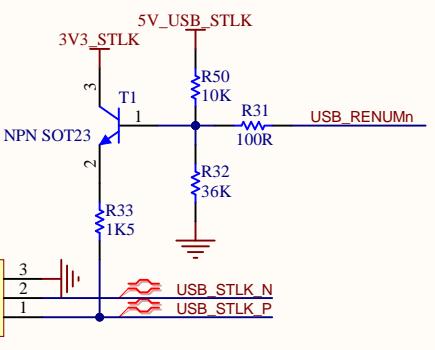
HW
SNT



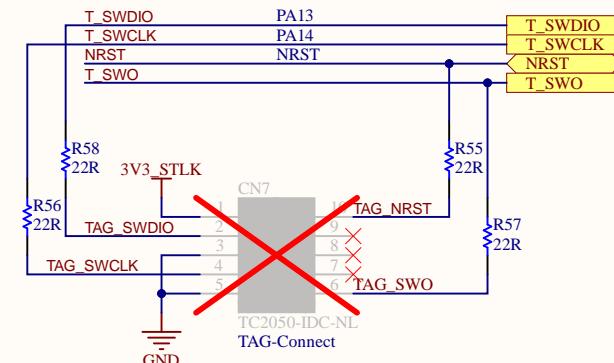
ST-Link USB



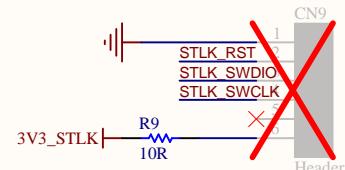
USB vB 105017.000



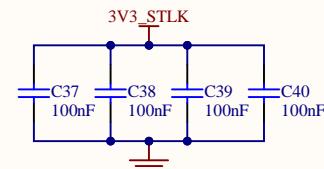
TAG connection



ST-Link SWD connector



ST-Link decoupling



ST-Link COM LED

