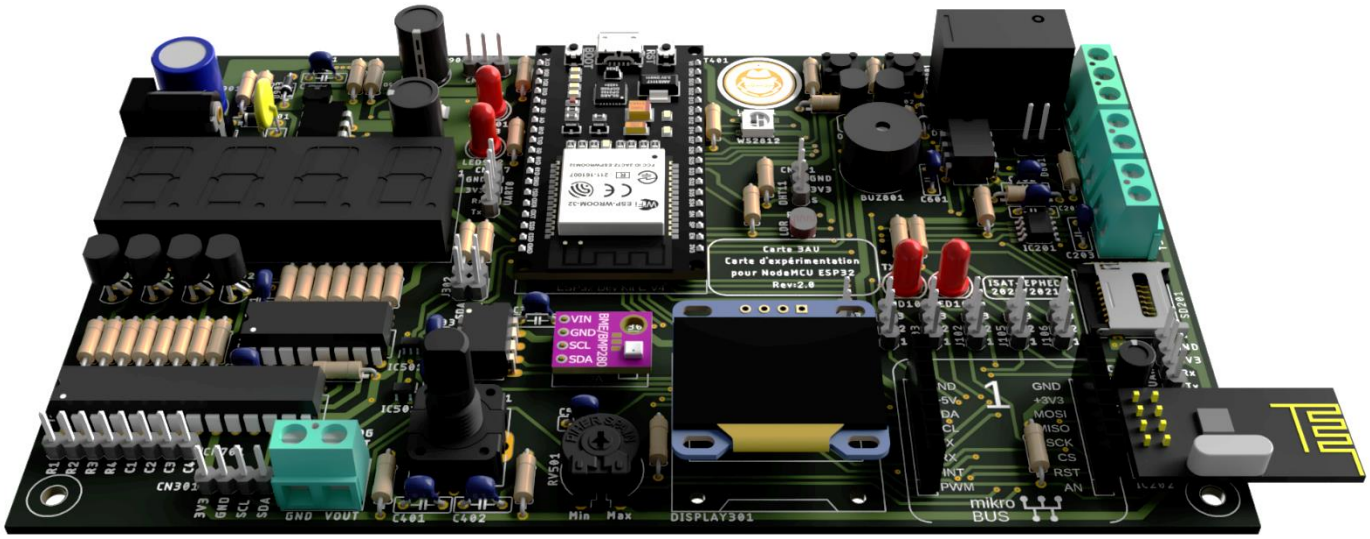
ECOLE PRATIQUE DES HAUTES ETUDES COMMERCIALES

CATEGORIE TECHNIQUE

Réalisation d'une carte d'expérimentation à Microcontrôleurs basée sur un ESP32

Travail de fin d'études présenté en vue de l'obtention du diplôme de bachelier en Automatisation

Juan-Felipe ALVAREZ LOPEZ



Responsable en entreprise : Émile Costa

Promoteur ISAT : Émile Costa

Année Académique 2020 – 2021

Abréviations

PLC	Programmable Logic Controller
CAN	Control Area Network
SPI	Serial Peripheral Interface
I²C	Inter-Integrated Circuit
PCB	Printed Circuit Board
MOSI	Master Out Slave In
MISO	Master In Slave Out
SS	Slave Select
DO	Data Out
SDA	Serial Data
CLK	Clock
OLED	Organic Light-Emitting Diode
PWM	Pulse-width modulation
µC	Microcontroller
EMI	Electromagnetic interference
GPIO	Global Peripheral Input Output
IO	INPUT/OUTPUT
PLC	Programmable Logic Controller
SPIFFS	Serial Peripheral Interface Flash File System
BCD	Binary Coded Decimal
DC	Direct Current
AC	Alternative Current

Avant-propos et remerciements

Le résultat de ce travail n'aurait pas été le même sans l'aide que j'ai pu recevoir. J'aimerais donc commencer en adressant un remerciement tout particulier à Mr. Emile Costa, qui a été mon promoteur et mon maitre de stage et qui a su se rendre disponible à chaque fois que je l'ai sollicité. Il ne m'a pas simplement aidé mais il a surtout pour moi été une source d'inspiration, me motivant à en apprendre toujours plus sur les différentes facettes de son métier.

J'aimerais également adresser un remerciement à ma famille (Lucero, Angel, Estelle, Andres, Monique, Christian et Luna) qui m'ont aidé et accompagné tout au long de mon stage et qui m'ont toujours permis d'être dans les meilleurs conditions pour la réalisation de ce travail.

Il également important je pense, d'adresser un remerciement à toutes les personnes qui produisent des travaux libres d'utilisation et qui les diffusent sur internet, ces derniers contribuent de manière non négligeable à la réalisation de nombreux travaux, sans eux cette carte n'aurait peut-être pas vu le jour. Que ce soit par la création de bibliothèques destinées à la programmation et à la conception du PCB, ou alors pour tout ce qui a été modèles 3D.

De manière plus générale, j'aimerais remercier l'ensemble du corps enseignant de l'ISAT EPHEC et notamment la direction qui rendu possible la réalisation de ce TFE que ce soit financièrement ou plus simplement en me donnant librement accès aux infrastructures de l'école. Mais aussi l'ensemble de mes professeurs qui ont tous su me communiquer leur passion et qui se sont également toujours rendu disponibles.
Parmi ceux-ci j'aimerais adresser un remerciement particulier à Mr Hirsoux et Mme Marsiat, puisque lorsque je suis arrivé à l'ISAT je me trouvais dans une situation délicate, ils ont malgré tout cru en moi et m'ont donné la possibilité de m'engager dans cette formation d'Automaticien où je me suis épanoui plus que je n'aurai pu l'imaginer et qui aura été une énorme réussite à titre personnel.

Préface

Adaptation du travail de fin d'études au contexte sanitaire

Ce travail a été réalisé alors qu'on était en pleine crise du COVID, il est donc certain qu'il s'est retrouvé impacté. Néanmoins, je pense avoir eu de la chance car le cadre de production du travail se prêtait assez bien au télétravail. Mais, je pense que le principe d'avoir son espace personnel au même endroit que son espace de travail, peut être un frein conséquent à la productivité. Ça l'a été pour moi, dans le sens où, personnellement, je suis beaucoup plus facilement déconcentré lorsque le cadre de travail est trop intime. Je trouve ça assez difficile de rester impliqué à 100% dans un projet quand on n'a pas la possibilité de prendre de la distance avec son lieu de travail. J'entends par là que, lorsque je travaille depuis chez moi je n'ai pas de réels horaires, ainsi il est arrivé que je commence à travailler seulement quelques minutes après mon réveil ou en dehors des heures de « bureau ». Cet aspect rend le travail beaucoup plus intense, et donc beaucoup plus fatiguant.
Je me rends bien compte que je suis bien moins impacté que d'autres. Cependant, je pense que négliger l'aspect psychologique lié au travail serait une erreur. Je n'ai réalisé qu'un stage de 16 semaines sur un sujet qui me plait beaucoup et même comme ça, j'ai senti l'usure. Personnellement, rien que les déplacements entre mon lieu de travail et mon domicile étaient un moment de détente.

J'ai également eu la chance que ni moi ni mes proches avons contracté le COVID, mais je pense que pour beaucoup, l'idée de potentiellement rater parce qu'on peut tomber malade à tout moment, et ainsi perdre un temps précieux, a été une source de stress permanente. Je suis assez content d'avoir eu un excellent contact avec mon maitre de stage, ce qui a permis que ce genre de préoccupations n'aient pas été un problème, tant je sentais que je pouvais facilement lui en parler. Malgré tout j'ai conscience mais que ce n'est pas systématiquement le cas, surtout lorsqu'on travaille dans une entreprise où le client apporte une exigence supplémentaire.
Et même si je sais qu'il y a toujours eu des pressions et du stress dans le monde professionnel, j'ai le sentiment que le COVID a accentué ces deniers.

Une autre source de stress liée à la crise du COVID a été que cette dernière a impacté un certain nombre de secteurs, et notamment le secteur de l'électronique où des retards de livraisons et de production assez important sont survenus. J'ai la chance d'avoir le choix sur les composants à utiliser, je n'ai donc pas eu de problème à m'adapter quand des pièces n'étaient plus disponible. Mais il est évident que pour des applications commerciales, on ne peut pas changer le contenu d'un projet aussi facilement.
Enfin terminons par évoquer que malgré le contexte compliqué, le domaine de l'automatique est un des secteurs qui se sont retrouvés propulsés en tête des besoins. Puisque sans la présence des gens sur les sites de travail, automatiser un maximum les tâches qui nécessitaient une présence humaine a été une priorité pour beaucoup d'entreprises. C'est donc important de relativiser et de prendre en compte que nous ne sommes clairement pas ceux qui avons le plus souffert de la crise.

Table des matières

Présentation de l'entreprise

Pour mon travail de fin d'études (TFE), j'ai fait le choix d'un stage/TFE en interne pour un de mes professeurs, Mr Emile Costa, qui m'a proposé un projet convainquant et qui correspondait assez bien à ce que je recherchais.
Au cours de ce stage j'ai donc été amené à travailler avec mon professeur et à découvrir les différentes facettes de son travail. Puisqu'en parallèle de son travail de professeur, Mr Costa travaille comme indépendant en réalisant différents produits intégrant informatique, électronique et modélisation 3D.

Le travail qui m'a été demandé est directement en rapport avec son cours de systèmes embarqués II, qu'il donne au premier quadrimestre de la troisième année du bachelier en automatique dispensé à l'ISAT. Un cours basé sur l'utilisation de différents microcontrôleurs, dont notamment l'ESP32.

Dans le cadre de mon travail j'ai pu accéder à tout le matériel disponible à l'ISAT, mais compte tenu de la situation exceptionnelle relative au COVID, j'ai réalisé une partie de ce travail en télétravail. Ainsi je n'étais sur place que 2 jours par semaine.
Sur le temps où j'ai pu travailler sur place, j'ai essentiellement été amené à utiliser les outils informatiques disponible sur les machines pour tout ce qui est modélisation 3D et conception de schémas électriques.

D'autres compétences plus techniques ont été sollicitées comme la soudure des différents composants sur la plaque mais par chance je disposais des outils nécessaires à mon domicile j'ai donc pu réaliser ces dernières depuis chez moi.

1. Introduction

1.1 Cahier des charges

Comme précisé dans le cahier des charges, mon travail consistait à réaliser une carte basée sur l'utilisation d'un ESP32 dont l'utilité serait de servir comme support didactique au cours de systèmes embarqués II, cette carte doit interfacer tous les composants nécessaires et utiles pour permettre aux étudiants de réaliser les travaux demandés.
Pour s'en assurer un certain nombre de fonctions sont prévues :

- Un bus I²C qui interface différents composants tels que :
  - Une mémoire EEPROM
  - Un afficheur alphanumérique (OLED 128x64)
  - Un capteur de température et d'humidité (BME280)
  - Un extendeur d'entrées/sorties (MCP23017) sur lequel seront connecté un clavier à matrice 4x4 et un afficheur 7Segments à 4 digits.
- Un bus SPI qui lui aussi interface différents composants :
  - Un module de gestion d'un thermocouple de type K (MAX31855)
  - Un module d'émission/transmission radio (NRF24l01)
- Des entrées/sorties digitales et analogiques pour interagir avec des composants :
  - Des boutons poussoirs et tactiles
  - Un potentiomètre
  - Un encodeur rotatif
  - Une LED RGB
  - Une photorésistance
  - Un capteur de température (DHT11)
  - Le pilotage d'un relais
- Un émetteur/récepteur CAN et son bornier qui doit permettre en combinaison du contrôleur CAN intégré à l'ESP32 d'exploiter le protocole CAN très repandu dans l'industrie automobile.
- Enfin la carte doit être équipée d'un emplacement mikrobus destiné aux clicboards de mikroelectronika.

En plus des composants déjà intégrés à la carte il est prévu de placer des pin header qui doivent permettre aux étudiants d'interfacer d'autres composants.

Pour permettre le fonctionnement des différents composants présents sur la carte il a fallu intégrer à celle-ci un système de gestion de l'alimentation et des protections contre les surtensions et inversions de polarité. Ainsi qu'un système automatique permettant de choisir la source d'alimentation (USB ou JACK).
Le système d'alimentation est conçu en 2 parties, la première assure l'alimentation +5V DC (« Direct Current » ou « courant continu » en français) à partir d'une entrée +12V DC grâce à un convertisseur Buck DC-DC (AP3211), la deuxième assure l'alimentation +3V3 DC très utilisé pour ce genre de composants, il est généré à partir du grâce à un régulateur linéaire (AZ1117). L'analyse en détail des différents composants de cette alimentation fera l'objet d'un chapitre.

Enfin la carte sera alimentée via USB notamment lors de la programmation, dans ce cas le 3V3 sera généré depuis le 5V fournis par l'USB.

Pour compléter ce travail il a été aussi demandé de produire un dossier technique contenant différents documents dont le but est de faciliter l'utilisation de la carte.

- Liste des assignations des différentes broches du microcontrôleur
- Liste du matériel (BOM) ainsi que leurs datasheets
- Les différents schémas électriques, PCB et modèles 3D
- Ainsi que produire différents codes nécessaires pour vérifier et valider le fonctionnement de la carte.

La carte répond à certaines exigences en termes de taille (format euro 100*160mm) et de disposition.

1.2 Objectifs

L'objectif de ce travail comme déjà évoqué est de servir de support didactique tout en me familiarisant avec les techniques de conception et de développement de circuits imprimés. Ce qui apporte une vraie plus-value à ce projet est la liberté que j'ai pu avoir, en effet il est fort appréciable de pouvoir explorer le monde professionnel à travers un projet dont je suis responsable. De ce fait j'ai pu durant mon stage découvrir et apprendre tout en avançant dans la production de mon travail.

1.3 Intérêt

Enfin, terminons cette introduction en évoquant l'intérêt que présente un tel projet.
Dans le cadre du cours de systèmes embarqués II auquel j'ai moi-même pris part, j'ai été amené à réaliser des montages sur « breadboard », cependant d'un point de vue pratique cela présente quelques défauts. D'une part il est peu évident de réaliser des montages complexes de cette manière et cela pouvait limiter les exigences, d'autre part tous les étudiants ne possédant pas exactement le même matériel, il peut arriver que certains ne puissent pas réaliser leurs travaux dans les meilleures conditions.

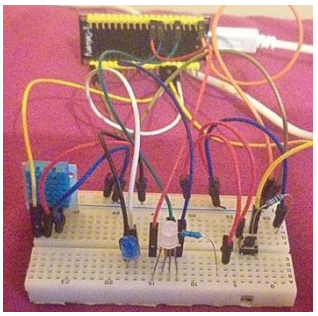


Figure 1 : Montage sur Breadboard

En plus de ces deux raisons, il y a un argument un peu plus subjectif, en effet je pense que pourvoir disposer d'un support de cours avec un aspect plus « pro » ne peut que stimuler les étudiants.
Pour ces raisons je pense que le travail que j'ai été amené à réaliser présente un réel intérêt non seulement pour mon professeur qui pourra disposer d'un support développé spécifiquement pour son cours mais également pour les étudiants qui à travers cet outil et la documentation technique qui l'accompagnera pourront prendre en main le cours plus facilement.
Enfin évoquons simplement que de telles cartes existent sur le marché, comme celle utilisée pour la première partie du cours (EasyPic7) mais leur prix peut vite devenir un obstacle (+/-150 euros). Précisons que l'optimisation du prix bien que non imposé par le cahier des charges a été un sujet de travail.

Méthodologie

2.1 Mind mapping

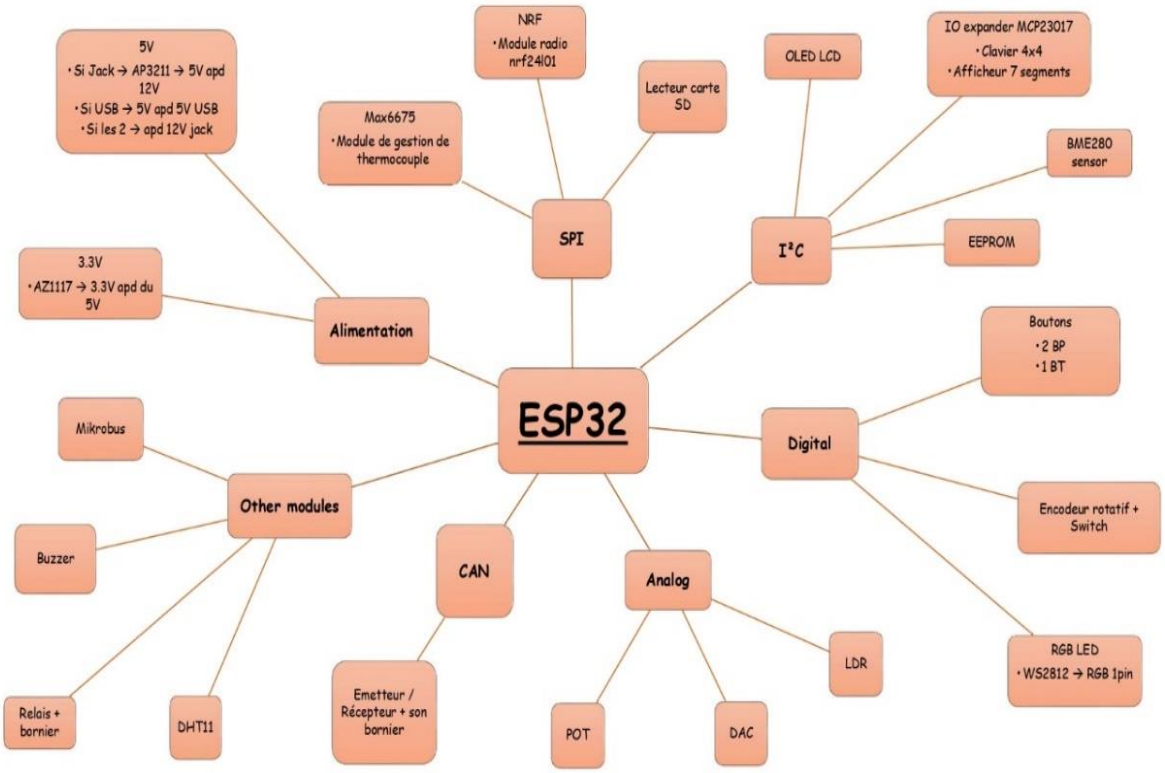


Figure 2 : Mind map2.0

Avant d'entamer la conception, j'ai réalisé une carte mentale comprenant l'ensemble des caractéristiques que la carte devait adopter. (ANNEXE H) J'ai aussi segmenté le travail en différentes « zones » et je me suis imposé de réaliser les schémas en suivant autant que possible ces mêmes zones.

2.2 Conception

Une fois une structure claire établie et validée, j'ai commencé en réalisant une première version des différents schémas tout en me familiarisant avec le vocabulaire spécifique au composants électroniques et à la manipulation des différents logiciels de conception.
Ensuite place à la réalisation du PCB à partir des schémas établis et enfin afin de visualiser et présenter ce qui a été produit j'ai généré un maquette 3D de la conception.
Ces différentes étapes seront reprises en détail dans ce qui va suivre.

2.3 Révision

Une fois la première version terminée et testée, nous avons étudié le résultat, il en est ressorti plusieurs améliorations possibles et quelques changements d'optimisation.
S'en est suivi la production d'une nouvelle « mindmap » et la modification des différents schémas en suivant les nouveaux plans. Et enfin nous avons clôturé par la production de la carte version 2.0, Version finale du projet.
Ces différents documents sont tous repris dans la documentation technique qui accompagne le projet et font l'objet de différentes annexes. (ANNEXE A, B, C et D)

3 Informations générales

Dans cette rubrique je vais rapidement passer en revue les composants principaux utilisés et, se faisant, développer les quelques notions qui seront utiles pour comprendre ce qui va suivre.
Je vais aborder les composants par groupes, suivant ainsi la structure de ma carte mentale et l'ordre des pages de mes schémas qui seront introduit au fur et à mesure.
Le détail complet de tous les composants et de leur datasheet est fourni en annexe (ANNEXE G) et dans la documentation technique.

3.1 Fusion360

La suite Autodesk propose de nombreux logiciels, parmi eux Fusion360, un logiciel de modélisation 3D, qui récemment a fusionné avec Eagle, un autre logiciel de la suite Autodesk qui permettait la création de schémas électriques et de PCB.
Aujourd'hui, il est donc possible de produire des schémas et générer un modèle 3D à partir de ceux-ci avec un seul logiciel.
Fusion360 permet de créer ses propres modèles 3D mais il est également possible grâce à une communauté très active de récupérer des modèles 3D libres d'utilisation afin de faciliter le développement quand cela est possible.
Enfin précisons que le fonctionnement de Fusion360 tout comme Eagle avant lui, se base sur l'utilisation de bibliothèques de composants, chaque composant possédant une emprunte (Footprint), qui définit l'encombrement et une schématisation (Symbol) destinée elle aux schémas de principes.

3.2 ESP32

Abordons à présent le microcontrôleur sur lequel est basé la carte.
Le NodeMCU ESP32 embarque donc un ESP32 et quelques composants de gestion d'alimentation, c'est à partir de cette carte, que les étudiants devront se procurer, que s'articule le cours. Un des sujets principaux du cours étant l'IoT (internet of things – internet des objets) le choix de l'ESP32 est dû à son intégration du WiFi.

3.2.0 Espressif

Espressif est une multinationale chinoise qui produit ce qu'on appelle des « SoC », pour « System on Chip », ce sont les ESP8266 et récemment leur ESP32 décliné dans tout un tas de modèles, mais ils produisent également des cartes de développement basées sur leurs microcontrôleurs [1]. C'est sur ces dernières qu'est basé notre projet, la « Carte 3AU ».

3.2.1 Qu'entend-on par microcontrôleur

Dans ce qui va suivre le terme « microcontrôleur » va être employé à plusieurs reprises puisque c'est le composant principal de la carte.
Définissons en premier lieu le terme microprocesseur : « c'est un moteur de calcul programmable constitué d'une unité arithmétique et logique, d'un processeur et de registres, capable d'effectuer des calculs et de prendre des décisions » [2]. Ainsi un microcontrôleur est un microprocesseur auquel on a ajouté des composants (mémoire, entrées/sorties, DAC, …). On peut considérer ça comme un « ordinateur sur une puce » d'où le nom de SoC.

On comprend bien maintenant ce qu'est un microcontrôleur mais le fonctionnement de ce dernier reste assez flou. Pour mieux comprendre il faut aller plus loin et malheureusement il est fort complexe d'aborder le concept de processeur, c'est pourquoi nous allons ici introduire ce dernier brièvement omettant volontairement les aspects théoriques trop pointilleux.
Le terme CPU, pour « central processing unit », trouve son origine dans les premiers ordinateurs en 1955, à cette époque les ordinateurs étaient d'une taille démesurée et par conséquent indéplaçables, ils étaient alors caractérisés de « fixed-program computer » où l'unité de calcul principale était donc appelée CPU [3]. Aujourd'hui les avancées dans le domaine de la miniaturisation des composants électroniques ont mené à produire des unités de calcul bien plus compactes. Elles n'intègrent pas moins de composants pour autant.
Différentes architectures existent, mais l'idée derrière leur fonctionnement reste identique, ainsi ce principe consiste en l'exécution d'une série d'instructions, qu'on appelle programme et qui sont enregistrées dans une mémoire dédiée.
Pour exécuter ce programme le processeur fonctionne comme suit : [3] [4]

1) Il extrait l'instruction à exécuter de la mémoire, l'adresse de cette instruction est fixée lors de la programmation, sans rentrer dans les détails, précisons simplement que le compilateur, transforme le programme codé en une suite d'instructions exploitables par la CPU, ce sont ces instructions qui sont enregistrées dans la mémoire.
2) Ensuite la CPU décode l'instruction, cette partie dépend de l'architecture du processeur. Puisque le constructeur construit son processeur pour que, lorsqu'on applique l'état de ce qui se trouve dans la mémoire aux entrées de sa logique, cela corresponde à des instructions qui permettent de réaliser les différents calculs.
   Illustrons ce principe pour éclaircir ce propos, imaginons un programme qui consiste à additionner deux nombres et à donner leur résultat.
   Il faut donc réserver des emplacements en mémoire pour les deux nombres et un troisième pour le résultat. Il faut ensuite, si celle-ci existe, exploiter l'instruction d'addition du processeur. Le tout est codé et programmé, lorsque le processeur fonctionne il exécutera en boucle les instructions mémorisées. Il extraira donc le type d'instruction puis la décodera, ira ensuite chercher les nombres mémorisés pour enfin les additionner et stocker le résultat de cette addition dans sa mémoire également.
3) Ce qui nous amène à la 3^{ième} étape de son fonctionnement l'exécution. En effet une fois les données extraites de la mémoire et décodées, le processeur exécute l'instruction puis passe à la suivante.

Enfin un dernier aspect important à aborder est la notion de vitesse, quand on entend parler de processeur, on parle souvent de leur fréquence. C'est en réalité le nombre d'instruction que ce dernier est capable de traiter par seconde qui est évoquée. Ainsi pour fonctionner il faut que les cycles d'exécution se suivent avec des timing toujours plus serrés, c'est ce qui cause entre autres, l'échauffement de ces derniers, on retrouve ainsi sur nos ordinateurs des systèmes de refroidissement toujours plus efficaces.
Les microcontrôleurs sont souvent accompagnés d'un oscillateur qui permet de cadencer le rythme d'exécution [5]. Pour notre ESP32 par exemple on parle d'une vitesse de 4MHz.

3.2.2 C/C++

La programmation est une des compétences principales que requiert le cours, c'est donc assez naturellement qu'elle se retrouvera dans ce qui va suivre et plus précisément lorsque le dossier technique sera abordé.

Le C, tout comme le C++, est un langage de programmation de bas niveau, c'est à dire qu'il est proche du jeu d'instruction de la machine, et cette dernière caractéristique permet une gestion de l'utilisation de la mémoire suffisamment performante pour en faire un des langages les plus utilisés quand on parle d'optimisation, il est aussi basé sur un standard ouvert qui le rend donc libre d'utilisation. Il trouve son origine en 1972 dans le très connu « Bell Labs » [6].

Aujourd'hui tous nos ordinateurs, téléphones et autres objets high-tech intègrent des processeurs toujours plus puissants et où la mémoire n'est plus un problème, le langage C présente donc peu d'intérêt au premier regard.
Non-seulement, la conception de tels microprocesseurs est lourde et coûteuse mais ils sont également difficiles à mettre en place. En effet, ceux-ci imposent certaines contraintes afin d'être utilisés, le refroidissement en est un exemple.
Il est aussi important de réaliser que, si ces objets high-techs sont les plus visibles, beaucoup d'appareils reposent sur des fonctionnements bien plus simples, à titre d'exemple en voici quelques-uns que l'on retrouve fréquemment : radioréveil, micro-ondes, thermostat, … À ça s'ajoutent la plupart des électroménagers et ce ne sont là que quelques exemples d'une liste interminable.
Tous ces appareils ne nécessitent pas de posséder des microprocesseurs puissants, l'utilisation de microcontrôleurs comme l'ESP32 est donc courante pour la conception de ces machines mais leurs capacités, et surtout leur mémoire, étant limitées, l'optimisation se trouve bien souvent au centre du développement. C'est donc le langage C et C++ qui aujourd'hui, sont les plus utilisés quand il est question de programmation de microcontrôleurs.
Lorsqu'on travaille avec des microcontrôleurs on retrouvera toujours le même schéma de programmation, à savoir des déclarations de variables donc l'allocation d'espace mémoire et ensuite un programme contenu dans une boucle infinie appelée « main » [7].
Si on décide de travailler avec l'environnement Arduino, on retrouvera une structure un peu différente qui cache en réalité ce même fonctionnement, plutôt qu'une fonction main où la boucle est créée manuellement, Arduino propose de travailler avec deux fonction une première « setup » où sont configurés les différents modules, c'est une fonction qui s'exécute une seule fois au démarrage, et une deuxième fonction « loop », qui tourne en boucle et qui contient le programme [8].

4 Analyse des schémas

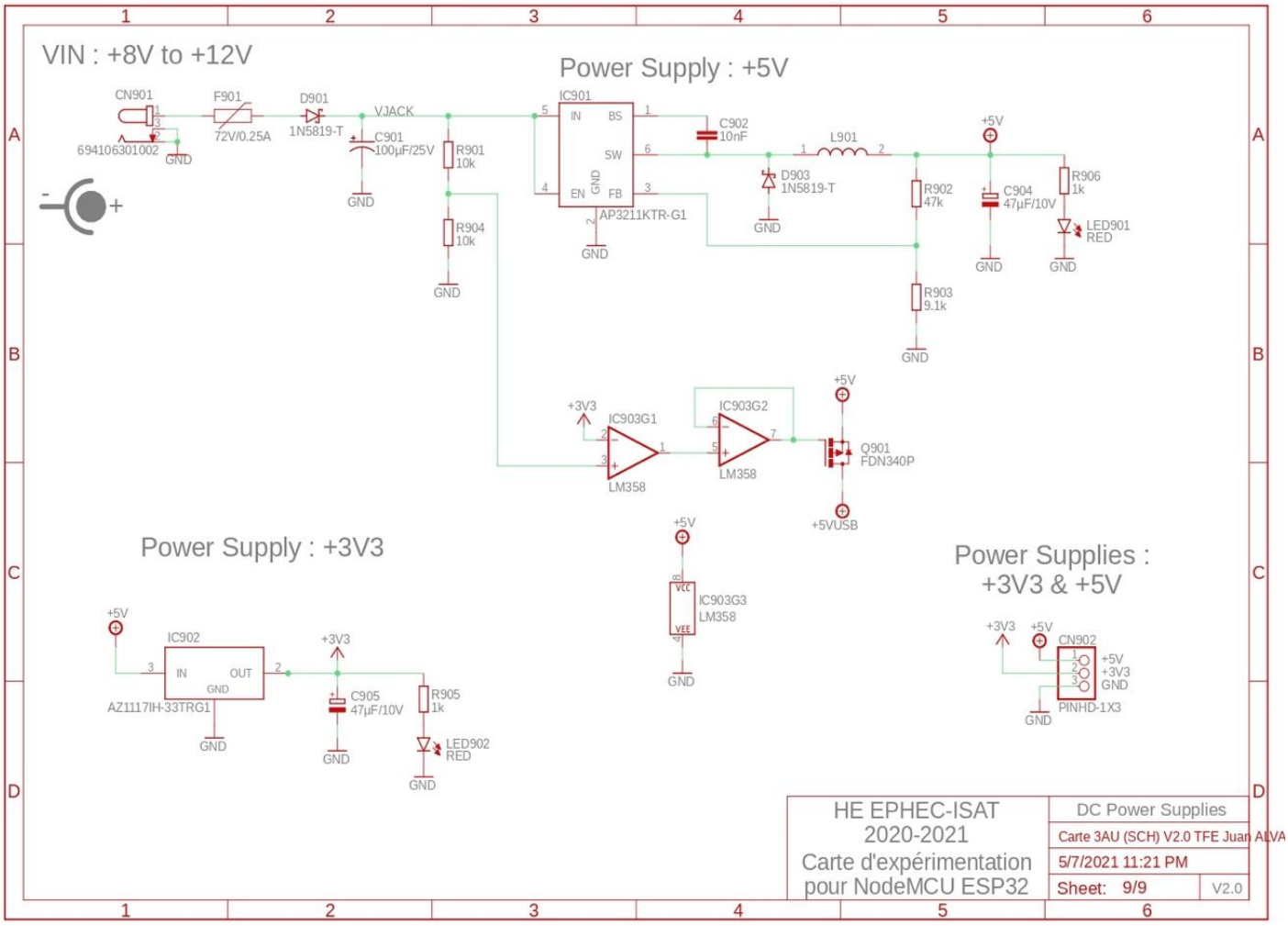


Figure 3 : Schéma d'alimentation

4.1 Alimentation

4.1.0 Convertisseur BUCK et régulateur de tension linéaire

Le Convertisseur BUCK également connu comme « hacheur série », est un composant qui « hache », découpe une tension continue pour en abaisser la valeur. Ce découpage est le résultat de la commutation rapide de transistors pendant une durée définie. En effet, la tension effective disponible en sortie d'une alimentation est rigoureusement égale à la tension moyenne, il en résulte que si pour une durée déterminée on décide de couper l'alimentation sur 50% de la durée de celle-ci, la tension effective disponible en sortie sera égale à la moitié de la tension d'entrée. Dans le cas de notre convertisseur, ce hachage a lieu périodiquement avec une fréquence très élevée. En effet, si les temps de commutation sont trop longs le courant tomberait trop bas. Ainsi, l'idée est de couper l'alimentation assez pour voir la tension diminuer mais pas trop pour conserver le courant au-dessus d'un certain seuil. On parle d'ondulation pour décrire cette variation due aux temps de commutation, la fréquence détermine donc directement la qualité du hacheur puisque ces temps sont d'autant plus courts que la fréquence est élevée.

Le régulateur de tension linéaire tout comme le « hacheur » sert à produire une tension de sortie à partir d'une autre tension en entrée. Mais son fonctionnement est très différent, en effet le régulateur utilise un composant passif (diode Zener entre autres) ou actif(transistor) pour garantir la tension de sortie. Notons toutefois que son rendement est moins bon, qu'il dissipe plus d'énergie et surtout qu'il limite le courant en sortie [9].

4.1.1 AP3211(IC901) et AZ1117(IC902)

Le AP3211 est un convertisseur BUCK, il est utilisé pour abaisser la tension en entrée du connecteur JACK (9-18V) à 5V DC, il est tout indiqué car il a un excellent rendement (95%+) et de ce fait ne dissipe que très peu de chaleur.

V_{FB}	Feedback Voltage	–	0.785	0.810	0.835	V
V_{FBOV}	Feedback Over Voltage Threshold	–	–	0.972	–	V
I_{FB}	Feedback Bias Current	V_{FB} = 0.85V	-0.1	–	0.1	μA

Figure 4 : exigences d'alimentation entrée "FB"

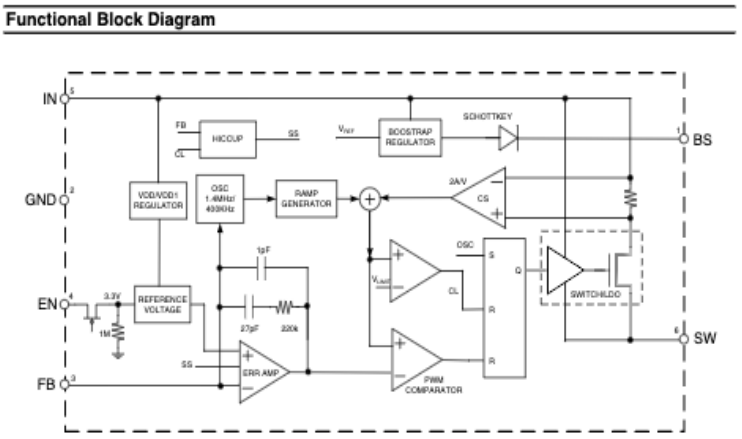


Figure 5 : Schéma Bloc AP3211

A partir du schéma de la figure 4, qui provient de la datasheet du composant. Il est possible de comprendre dans les grandes lignes le fonctionnement de ce dernier. Ainsi, l'entrée « EN » sert à activer le composant, sur la carte nous n'avons pas placé d'interrupteur, cette dernière est donc reliée à l'alimentation pour que le composant fonctionne en permanence lorsque le bloc d'alimentation est branché.
A partir d'un pont diviseur on fixe les valeurs des résistances (R902 et R903) pour placer l'entrée FB à 0.8V, c'est à partir de ce résultat que sont fixés les temps de commutation.« BS » est l'entrée dite « bootstrap », ce terme désigne en électronique un circuit où la sortie d'un amplificateur est appliqué à son entrée à travers un condensateur, dans l'idée de modifier l'impédance d'entrée de ce dernier.
Enfin « SW » est la sortie du convertisseur et « IN » est évidement l'entrée du module [10].

Le AZ1117 est lui un régulateur de tension linéaire, qui assure la conversion du 5V DC en 3V3 DC. Il est plus simple à mettre en place, avec simplement 2 entrées d'alimentation (IN et GND) et une sortie (OUT) qui délivre une tension fixe et un courant limité. Il possède cependant un plus faible rendement toutefois compensé par un travail réduit car ne devant pas gérer de grandes puissances, l'abaissement vers le 5V DC étant fait par le hacheur. Son choix est donc raisonnable. Ce genre de régaleurs sont fréquemment utilisés sur des modules pour fixer l'alimentation, ainsi on retrouve un régulateur de ce type sur le nodeMCU pour alimenter l'ESP à partir de l'USB, ou sur le module radio qui sera étudié plus tard [11].

4.1.2 LM358 (IC903)

La carte doit pouvoir être alimentée via son connecteur JACK en 12V mais lors du testing et de sa programmation elle sera également connectée via l'entrée micro USB du NodeMCU. Cette particularité exige la mise en place d'un système automatique de sélection de la méthode d'alimentation.
Le principe étant de détecter si le connecteur JACK est connecté en comparant l'entrée JACK à la tension 3V3. A partir du résultat de cette comparaison réalisée par le LM358, on alimente ou non la grille d'un transistor MOSFET canal P (Q901). Et c'est ce transistor qui en fonction de son état détermine quelle source alimente le 5V de la carte.

4.1.3 Autre

Cette alimentation intègre un certain nombre de composants passifs et des composants de protection. En ce qui concerne les protections de circuit, un fusible protège contre les surtensions et des diodes Schottky protègent contre les inversions de polarité.
Le fusible est un picoFuse, ce sont des fusibles réamorçables, prévu pour tenir des courants pouvant aller jusque 1.5 ampères avant de céder.
Une diode Schottky du nom du physicien Walter H. Schottky, est un type de diode ayant un très faible seuil de tension directe. On entend par là, que la chute de potentiel qu'elle apporte sur la ligne (0.15 à 0.4V) est bien plus faible que celle d'une diode classique (0.7V). Elle présente donc, dans le cas d'une alimentation devant fournir une tension la proche possible d'une valeur fixée, un intérêt certain. Elle se différencie d'une diode classique par la nature de sa jonction, en effet, là où une diode classique exploite une jonction P-N, la diode Schottky utilise une jonction métal semi-conducteur.
Le cas des condensateurs (sauf C902) sera abordé un peu plus loin, gardez cependant en tête qu'il s'agit de condensateurs de découplage [12].
Enfin on peut remarquer la présence de l'inductance (L901), son rôle est ce qu'on appelle une inductance de lissage. L'idée est de diminuer l'ondulation, principe physique introduit précédemment. Pour ce faire on exploite les propriétés de l'inductance, cette dernière a la particularité de « stocker » le courant ainsi lorsque des fluctuations de courant apparaissent cette dernière les compense, on dit que la bobine possède une inertie car elle est sensible aux variation de courant.
On peut assez facilement déterminer la valeur de cette dernière à partir de la formule régissant le comportement d'une inductance et des spécifications électriques du AP3211 qui fixe les fluctuations de courant. [9] [13]
UL = L*di/dt où di = Imax-Imin [13]

4.1.4 Remarques

Il faut toutefois savoir que des valeurs « standard » existent et sont largement utilisées, ensuite certains constructeurs dont notamment Texas Instrument ont développé des outils d'assistance pour le dimensionnement de composants destinés à leurs convertisseurs. Le travail effectué ici était donc destiné à fournir les outils nécessaires à l'utilisateur dans le cas où le régulateur utilisé venait à devoir être changé et qu'apparaissait alors le besoin de refaire les calculs de dimensionnement.

Enfin notons qu'un bornier est prévu afin d'exposer les pins d'alimentation pour éventuellement alimenter d'autres composants qui seraient ajoutés par les étudiants.

4.2 SPI

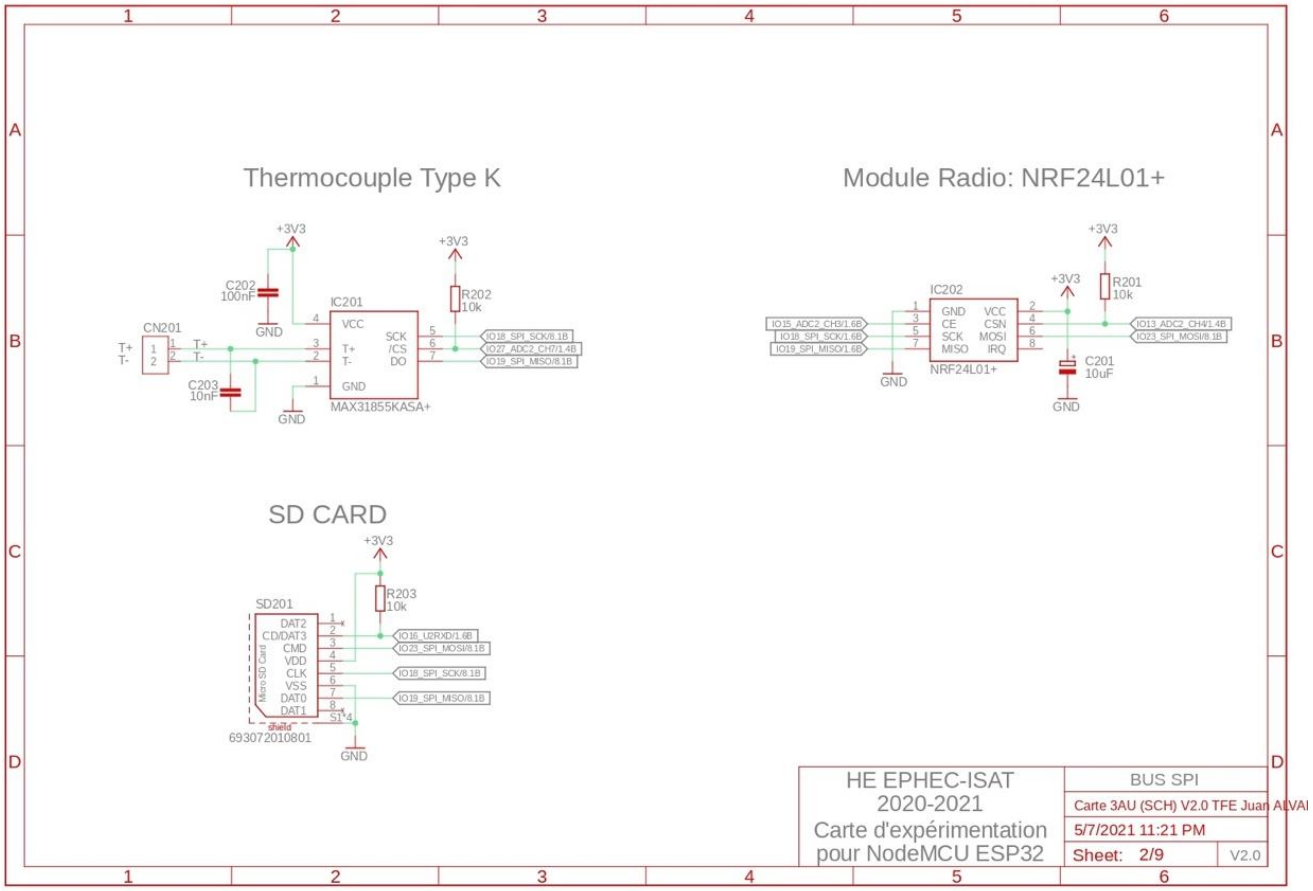


Figure 6 : Schéma bus SPI

4.2.0 Introduction au bus SPI

Le « SPI » pour « Serial Peripheral Interface », est un bus de données série synchrone qui opère en full duplex. On entend par là que c'est un bus physique qui fonctionne en série, donc où les données sont envoyées bit par bit et où l'envoi et la réception se font sur des lignes séparées.
On parle de « MISO » pour « master in slave out » et « MOSI » pour « master out slave in ». Le SPI fonctionne suivant le principe « maître-esclave », c'est toujours le maitre qui prend l'initiative de la communication par l'intermédiaire d'une liaison directe entre le maitre et chacun de ses esclaves. On appelle cette dernière « SS » pour slave select. Notons que ces appellations peuvent parfois changer mais le principe reste identique. On trouve parfois l'appellation « DO » pour « data out » à la place de MISO ou « CS » pour « chip select » à la place de SS.
Enfin précisons que les PINs SS fonctionnent à l'état bas c'est-à-dire qu'il faut leur appliquer un état logique bas ou autrement dit, placer la valeur physique de la PIN à 0V. Elles seront donc toutes « pulled up », c'est-à-dire reliées au +5V à travers une résistance tout en étant connectées directement à l'ESP, d'où leur nom qui signifie « tiré vers le haut » et qui fait référence au fait de tirer la ligne à son état logique dit « haut ». Ce système permet de s'assurer que toutes les entrées sont bien désactivées sauf quand il est explicitement programmé de passer la ligne à l'état bas [14] [15].

4.2.1 Module radio NRF24l01(IC202)

Le NRF24l01 est un module émetteur/récepteur radio qui opère sur la bande 2.4GHz.

Figure 7 : NRF24l01 [1]

Le détail complet de ses caractéristiques se trouve sur la datasheet qui est disponible dans la documentation technique mais la description du module y étant assez bien faite en voici la retranscription : « The nRF24L01 is a single chip 2.4GHz transceiver with an embedded baseband protocol engine (Enhanced ShockBurst™), designed for ultra low power wireless applications. The nRF24L01 is designed for operation in the worldwide ISM frequency band at 2.400 - 2.4835GHz. An MCU (microcontroller) and very few external passive components are needed to design a radio system with the nRF24L01». [16]
Le module est opéré et configuré à travers le SPI et embarque un régulateur de tension qui lui permet d'être alimenté en 5V.
Enfin, on remarque qu'un condensateur accompagne le module, en effet ce genre de modules fonctionnent sur des fréquences très énergétiques il arrive donc d'avoir des pics de courant assez élevés lors de leur utilisation, il est donc important pour le bon fonctionnement et la longévité du module, qu'il soit toujours accompagné d'un condensateur de découplage.

Puisque cette notion de condensateur de découplage reviendra encore, analysons-la rapidement. Dans des applications comme celle-ci, les pics de courant dont j'ai parlé entrainent toujours une chute de tension assez brève mais qui perturbe le fonctionnement sain des composants. A la mise sous tension, le condensateur se charge de sorte que lorsque ces pics de courants ont lieux, la chute de tension qui y est associée est compensée par la décharge très brève du condensateur.
Le mot « découplage » fait référence au principe de couplage électronique qui consiste en un transfert de l'énergie d'un élément à un autre. Il est donc question ici d'empêcher ce transfert entre signal et alimentation [17].

Terminons par préciser que le module possède un emplacement vide à sa surface prévue pour ajouter manuellement un composant SMD sur le PCB, il est entouré sur la figure 7 et est nommé C4 sur le PCB. Ce condensateur n'est autre que le condensateur de découplage intégré à la carte. Le choix existe donc sur la manière d'intégrer ce condensateur à la carte. Par facilité, le choix a été fait pour ce travail d'utiliser un maximum de composants traversant car plus simples à souder.

4.2.2 Lecteur micro SD (SD201)

La carte embarque un lecteur de carte micro SD qui communique en SPI. Bien que le nom des différentes connexions qui apparaissent sur le schéma diffèrent de celle classiques liées au SPI, le fonctionnement reste néanmoins rigoureusement identique et on retrouve bien les 4 connexions classiques du SPI.
Les carte SD et leur format compacte micro SD sont des mémoire dites « flash ».
On appelle ainsi toute mémoire à semi-conducteur réinscriptible, ainsi ce sont des mémoires qui conservent leurs données lorsque l'alimentation électrique est coupée.
A proprement parler ce sont donc des mémoires dites « EEPROM », mais ce terme sera défini lorsque nous étudierons le cas de la mémoire EEPROM intégrée directement sur la carte. Notons simplement que dans le cas de la mémoire flash, la spécificité de cette dernière est de pouvoir y écrire à plusieurs endroits en une seule opération.
L'apparition des premières mémoire flash remonte à 1988, initialement commercialisées par Intel, ce sont les mémoires NOR, peu de temps après ce sont les mémoires NAND qui sont commercialisées par Toshiba. Ces termes font référence à la logique interne de ces mémoires. Si on veut vraiment comprendre leur fonctionnement, il faut étudier le concept « d'effet tunnel », qui est un phénomène du domaine de la physique quantique et qui est hors de mon champ de compétence. Pour faire simple, l'idée qui se cache derrière ces mémoires à fonctionnement électrique, est de piéger des électrons, à l'aide de cet effet, dans des semi-conducteurs pour fixer l'état de cellules à « à 0 ou 1 », représentant simplement l'état logique de chaque bit de donnée par une de ces cellules [18].

4.2.3 Module de gestion de thermocouple type K (IC201)

Dans l'idée de la carte 3AU, l'objectif était d'y intégrer un certain nombre de capteurs. C'est dans cet optique que le MAX31855, module de gestion de thermocouple de type K, y a été ajouté.
Pour rappel, on appelle thermocouple un couple de deux matériaux qui lorsqu'il subissent l'effet Seebeck, sont utilisés pour mesurer une température. Cet effet connu depuis très longtemps, veut que lorsque la jonction de deux matériaux soudés en 1 point est soumise à une température, il apparaisse une différence de potentiel entre les deux matériaux [19].
Le terme « type K » lui indique simplement la composition des deux matériaux, en l'occurrence il s'agit d'une jonction entre « Chromel » et « Alumel »[20].
La mesure visée par le module concerne la jonction entre les deux matériaux, appelée « jonction chaude », et pour y parvenir on fait ce qu'on appelle une compensation par jonction froide, c'est-à-dire calculer, à partir de la différence de potentiel qui est apparue et de la température individuelle de chacun des matériaux, la température à laquelle se trouve la « jonction chaude ».
Il est donc également possible de récupérer la mesure de température interne du module, réalisée sur la « jonction froide » [21].
Enfin, notons que seul le module MAX31855 est intégré à la carte, et que donc le thermocouple doit être ajouté manuellement par l'intérimaire du bornier placé à cet effet.

4.3 I²C

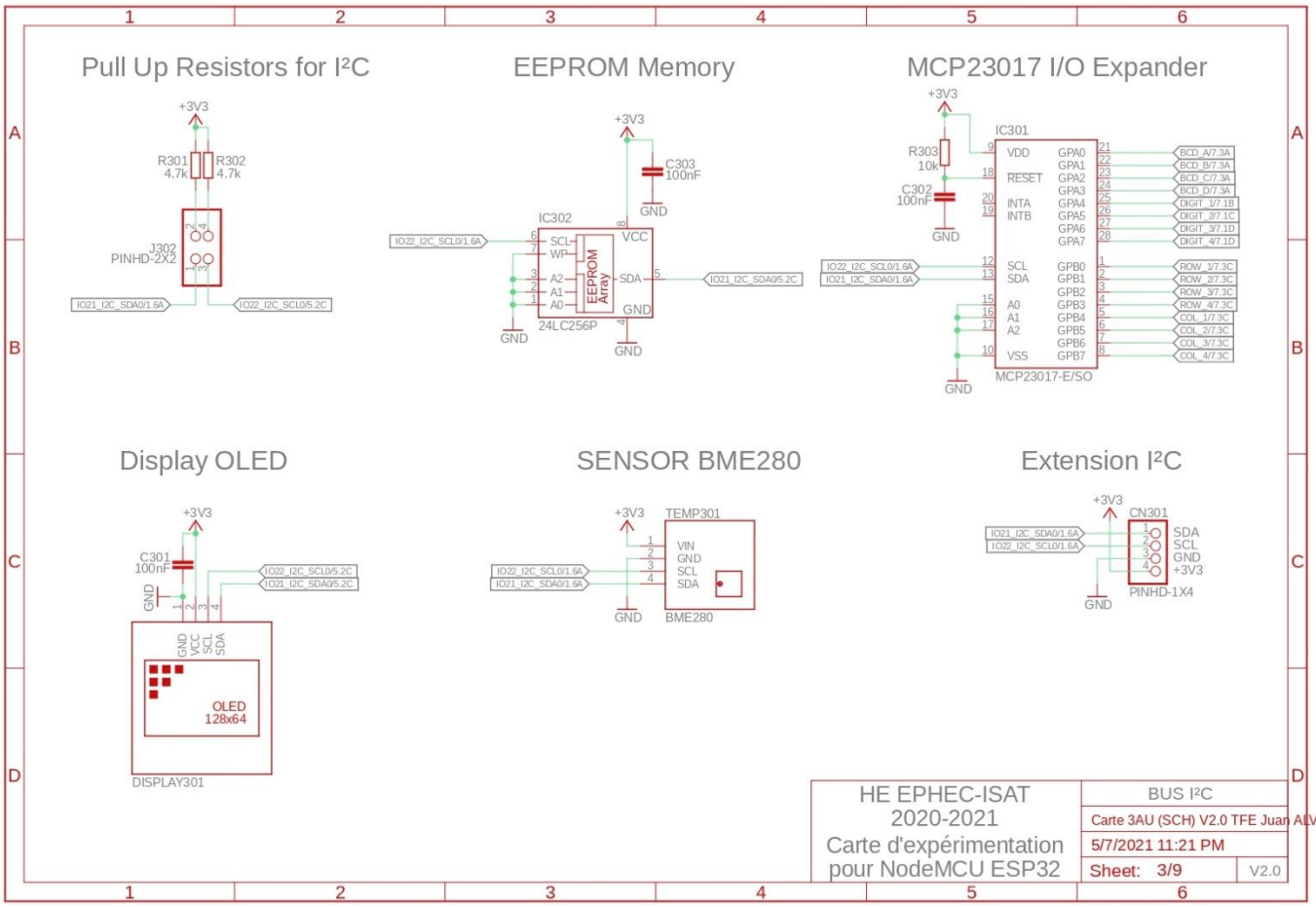


Figure 8 : BUS I2C

4.3.0 Introduction au bus I²C

Également connu sous l'appellation « Two Wire Interface » ou « Two Wire Serial Interface », la norme I2C pour « Inter Integrated Circuit », développée par Phillips en 1982, est un bus série synchrone bidirectionnel half-duplex [22].

En pratique tout participant au bus possède deux connexions, SDA (Serial Data Line) ET SCL (Serial Clock Line). On parle alors de bus synchrone car la communication est cadencée par le signal d'horloge communiqué par le maitre à travers la ligne SCL. Les échanges ont toujours lieu entre un maitre et un ou des esclaves, mais tous les participants du bus peuvent jouer le rôle d'esclave ou de maitre.

Le terme half-duplex désigne un échange où l'envoi et la réception se font sur la même ligne et ne sont pas possible simultanément. Alors que bidirectionnel fait simplement référence au fait que on peut communiquer dans les deux sens bien que la communication soit toujours à l'initiative du maitre [22].

Les lignes sont actives à l'état bas c'est pourquoi elles sont « pulled up » (J302), mais puisque parfois certains composants peuvent intégrer leur propre système de pullup, la mise à l'état haut est dépendante de la mise ou non du jumper(J302).

Enfin soulignons que le protocole I2C peut être implémenté de manière logicielle, ce qui lui permet d'être utilisés sur la plupart des microcontrôleurs, et fait du protocole une référence dans le monde de l'électronique. Même si le nombre de participants est limité

par le nombre d'adresses prévue à savoir 7bits d'adresses + un bit R/W qui détermine le sens de la communication. Ce sont donc 128 adresses qui sont possibles. Notons tout même que, certains constructeurs produisent des modules avec des adresses prédéfinies ce qui peut également poser problème si plusieurs composants possèdent la même adresse.

4.3.1 EEPROM Memory 24LC256 (IC302)

Une mémoire « EEPROM » pour « Electrically Erasable Programmable Read Only Memory », Le terme « ROM » désigne une forme de mémoire qu'on appelle communément « mémoire morte » en français, ce terme désigne une mémoire dont le contenu ne s'efface pas même lorsque cette dernière n'est plus sous tension.
Une PROM est donc une mémoire morte programmable, on entend par là, qu'on peut fixer son contenu lors de la programmation. Alors que « Electrically Erasable » indique que son contenu est effaçable électriquement. A noter que le nombre de réécritures (effacement puis nouvelle écriture) est limité et cette information est donnée par le constructeur et disponible sur la datasheet [23].
Très utiles quand il s'agit d'enregistrer des informations qui ne doivent pas être perdues, elles sont aujourd'hui très utilisées dans pas mal d'applications, à titre d'exemple, citons le cas du secteur automobile, où sur les voitures les données comme le kilométrage ou le numéro de châssis sont enregistrés dans une mémoire EEPROM, puisque ces données ne peuvent pas disparaitre lorsqu'on coupe le contact.

La 24LC256 est donc une mémoire EEPROM de 256Kbit, qui exploite le protocole I2C pour sa programmation et sa lecture. Elle dispose de 3 entrées pour fixer physiquement les bits d'adresse au cas où plusieurs mémoires seraient connectées en même temps. Enfin, on notera la présence d'un condensateur de découplage sur son entrée d'alimentation.

4.3.2 OLED 128x64 (Display301)

Dans l'idée de la conception de la carte 3AU, la présence d'afficheurs était importante. En effet, étant donné l'utilisation qui en sera faite, à savoir la réalisation de différents travaux pour le cours, l'intérêt d'avoir des afficheurs est donc d'avoir une manière de visualiser le résultat de ces derniers.
En effet, lorsqu'on pense à l'écriture ou la lecture d'une mémoire EEPROM par exemple, le résultat de ces opérations est totalement invisible, le passage par l'affichage de cette lecture sur un écran rend donc visible ce dernier.
OLED est l'acronyme de « Organic Light-Emitting Diode », à la différence donc d'un écran LCD, sur un écran OLED chaque pixel est source de lumière, ces derniers ne nécessitent donc pas de rétro-éclairage, ce qui les rend plus fin. Mais revenons rapidement sur ce qu'est un écran OLED, c'est un film organique composé de molécules semi-conductrices, placé entre deux autres films qui font office d'électrodes. Lorsqu'un courant circule à travers le film organique, il émet de la lumière dans le spectre visible [24].
L'écran qui est intégré à la carte 3AU est donc un OLED avec une résolution de 128 pixels en largeur et 64 en hauteur.

4.3.3 BME280/BMP180 (Temp301)

Le BME280 est un capteur de pression, d'humidité et de température développé par BOSH. Le module destiné à être présent sur la carte 3AU, est monté sur un PCB de taille réduite avec son régulateur de tension et quelques résistances.
Sur la carte 3AU se trouve un connecteur pour ce dernier.
Ce capteur se révèle être très intéressant par sa relative facilité d'implémentation et d'utilisation, tout en étant un reflet réaliste de ce qui se fait dans le domaine.

4.3.4 MCP23017(IC301)

Le MCP23017 est un module qui permet l'ajout d'entrées/sorties par l'intermédiaire du I2C. Il possède deux ports de 8 bits (PORTA et PORTB), ce qui ajoute 16 entrées/sorties. Il possède également 2 sorties d'interruptions qui peuvent fonctionner ensemble ou séparément. Tout comme pour la mémoire EEPROM, le composant possède 3 entrées qui permettent de fixer l'adresse physique du composant, ce qui autorise jusqu'à 7 de ces composants sur le même bus.
L'étude des composants qui sont sur le MCP23017 se fera séparément. Mais précisons déjà que le PORTA sera dédié à un clavier à matrice 4X4 alors que le PORTB sera lui dédié à la gestion d'un afficheur 7 segments de 4 digits.

4.3.6 Remarques

Les lignes SDA et SCL sont exposées pour être utilisées (CN301), ainsi les étudiants pourront ajouter des composants sur le bus s'ils en ont besoin.
Remarquons que plusieurs composants possèdent leur propre condensateur de découplage.
Enfin, revenons sur le cas des lignes de données et d'horloge qui ont été « pulled up ». En effet la raison de leur mise à « l'état haut » a été expliquée, mais la valeur des résistances a été passée sous silence. Sans rentrer dans les calculs, qui seront disponibles pour être analysés dans un document Excel, précisons simplement que leur présence doit permettre de maintenir l'état de la ligne quand aucune donnée ne transite mais sans que la chute de potentiel qu'elles apportent ne perturbe le signal. Il faut savoir qu'en fonction du nombre de participants connectés sur le bus, ce dernier peut parfois chuter. Dans ce cas, il faut alors faire attention car de trop grandes résistances de pull up pourraient bloquer le bus [25].

4.4 Entrées/sorties digitales

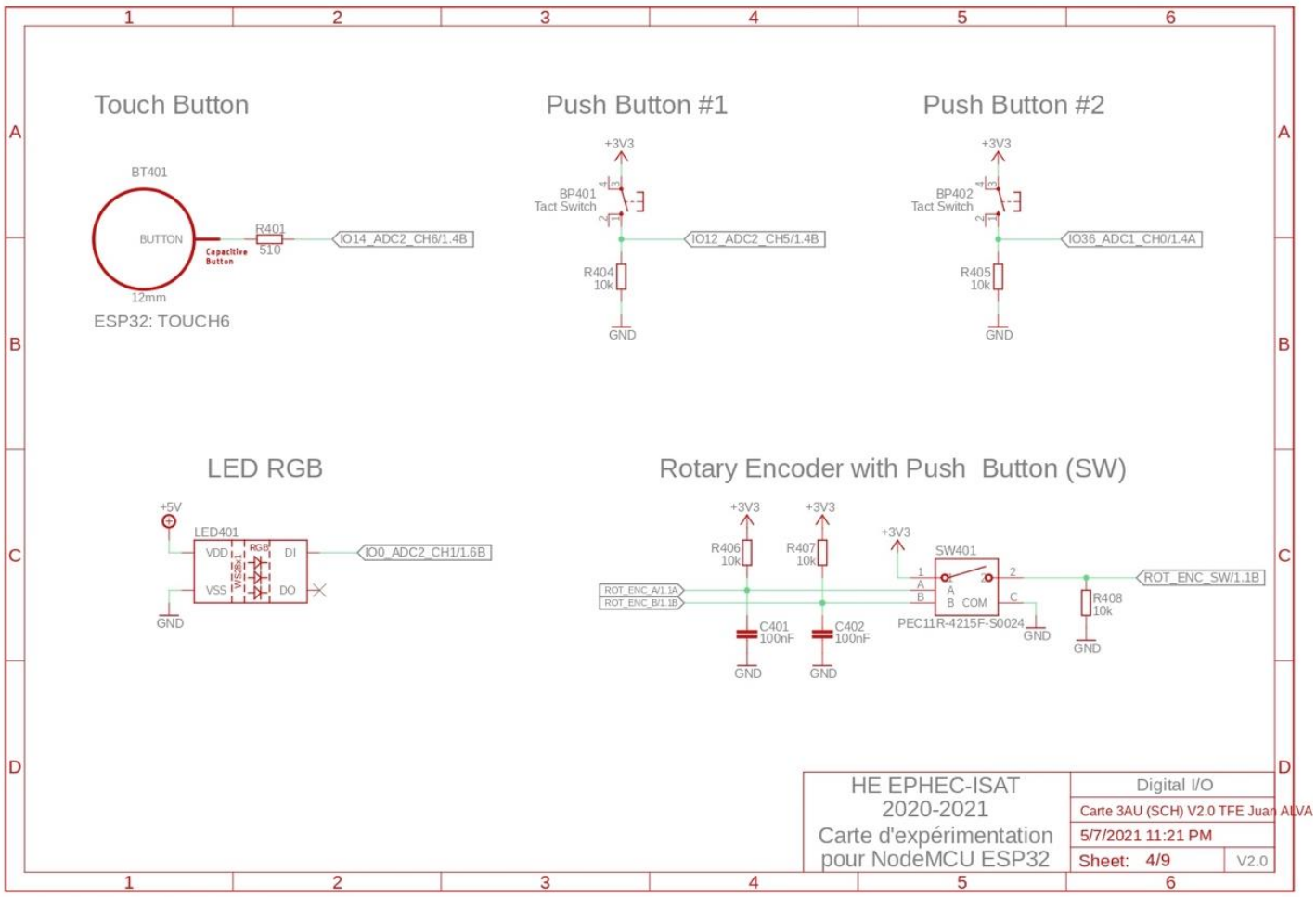


Figure 9 : Entrées/sorties digitales

4.1.0 Principe de l'entrée/sortie digitale

Les entrées/sorties sont aujourd'hui à la base de tous les microcontrôleurs, c'est à partir de ces dernières qu'ils interagissent. On entendra souvent parler de « GPIO », acronyme anglais de « General Purpose Input/Output ». Ensuite le terme « digitale » par opposition à « analogique », renseigne sur le genre d'utilisation qu'on en fait.

L'idée derrière le concept « digital » est de faire varier l'était logique (0 ou 1) en faisant passer la ligne à 0 ou 5 Volts. On différencie entrée et sortie, puisque comme sortie c'est le microcontrôleur qui fait varier l'état, à titre d'exemple citons l'allumage d'une LED, alors que en tant qu'entrée la variation provient du monde extérieur et le microcontrôleur ne fait que surveiller et réagir aux changements d'état, comme entrée digitale nous avons, entre autres, les boutons poussoirs [15].

Pour bien comprendre ce concept, attardons-nous sur le fonctionnement d'un microcontrôleur. Si on se réfère à la datasheet de notre ESP32 par exemple, on peut voir le détail des différents registres exploités par celui-ci [26]. Les registres sont à la base du fonctionnement d'un microcontrôleur, ils ont une fonction de mémoire, c'est en fait un assemblage de bascules qui les composent [27].

La bascule étant un circuit logique qui en fonction des signaux appliqués à ses entrées fait varier sa sortie, on parle ici donc toujours d'états logiques soit 0 soit 1 (tout ou rien).

Les GPIO sont donc liées à un registre (on retrouve souvent INTCON comme nom) et c'est à travers ce derniers qu'on peut accéder à l'état d'une GPIO. En fonction donc du rôle configuré, on viendra tantôt agir sur l'entrée de la bascule tantôt lire sa sortie.

4.4.0 Boutons (BT401-BP401-BP402)

Les boutons sont toujours un moyen privilégié d'interagir avec un appareil électronique, c'est pourquoi il était quasi obligatoire d'en retrouver sur la carte. On retrouve au total 3 boutons sur celle-ci : deux boutons poussoirs classiques et un bouton tactile (capteur capacitif).
Puisque l'intérêt de la carte 3AU est d'offrir aux étudiants un large échantillon de ce qui se fait dans le monde des systèmes embarqués, c'était donc l'occasion d'aborder l'utilisation des capteurs capacitifs, en effet, l'ESP32 permet l'utilisation de certaines GPIO comme entrées tactiles.

Analysons donc ce concept :

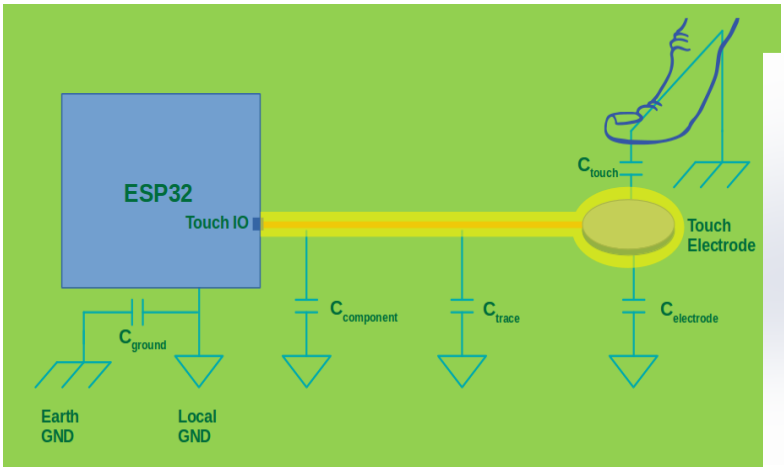


Figure 10 : Schéma montage capteur

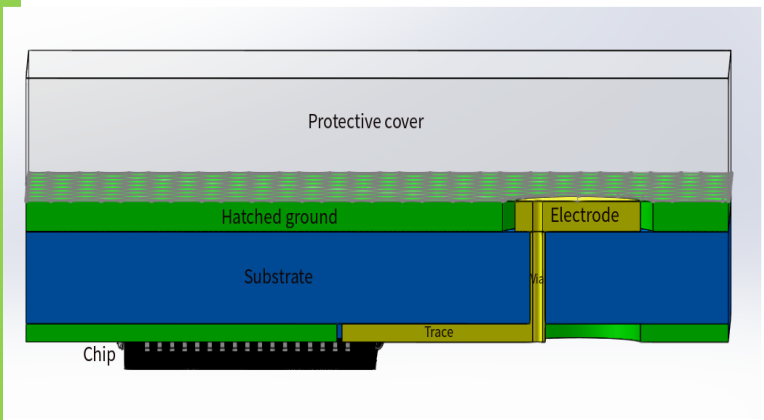


Figure 11 : Schéma capteur capacitif classique

Ces schémas proviennent directement de la documentation d'Espressif à propos de leur capteur capacitif.
En effet, Espressif tient un Github avec des notes à propos des différentes fonctions de leur ESP32.
Les liens vers leur Github se trouve dans la documentation technique et la bibliographie.[28]

Le principe de fonctionnement d'un capteur capacitif se base sur la variation de la capacité qu'on apporte en approchant notre doigt.
Sur notre carte 3AU la « protective cover » est la couche de vernis qui recouvre les pistes, l'électrode est simplement la piste de cuivre. Le substrat est en fait le support sur lequel repose la piste de cuivre, dans notre cas c'est le plastique du PCB. Ensuite la trace est intégrée au NodeMCU.

A noter que sur la figure 7, Espressif intègre un certain nombre de condensateurs pour réduire au maximum les interférences, mais sur notre carte seule la partie finale est présente et nous avons simplement lié la piste de cuivre au NodeMCU. Cela rend notre capteur moins fiable car facilement déclenchable mais cela est largement suffisant pour l'utilisation qui en sera faite.

4.4.1 LED RGB WS2812 (LED401)

Figure 12 : WS2812 [2]

La LED RGB WS2812 fait partie d'un type de technologie novatrice qui aujourd'hui est de plus en plus répandue.
En effet, pour la plupart d'entre nous quand on pense à une LED RGB, on pense à la LED blanche à 4 pins, où on joue sur la luminosité de chacune des 3 LEDS avec un signal analogique pour en fixer la couleur.
Alors que dans le cas de notre WS2812 c'est un signal carré qui est utilisé et interprété pour fixer la couleur [29]. Ce genre de LEDs sont pensées pour fonctionner sur des rubans, l'idée est alors de pouvoir fixer la couleur de tout le ruban à travers une seule ligne.
Le principe est donc d'utiliser ce signal carré comme signal d'horloge et venir moduler les paramètres des données relatives aux couleurs sur celui-ci. Ce signal modulé consiste en un signal carré où les longueurs d'impulsions varient. Sans l'utilisation de bibliothèques dédiées, l'emploi de ce genre de LED peut vite devenir complexe tant la modulation en fréquence de signaux n'est pas un concept simple à mobiliser [29].

4.4.2 Encodeur rotatif (SW401)

Figure 13 : PEC11 [3]

Enfin le dernier module de cette catégorie, est l'encodeur rotatif.
Son fonctionnement est basé sur un codeur incrémental, ayant l'avantage de pouvoir tourner sans limite, cependant le calcul de la position est plus complexe qu'avec un simple potentiomètre. En effet, le principe du codeur incrémental est de détecter le passage ou non d'une source lumineuse à travers une bande trouée à intervalles réguliers. [30]

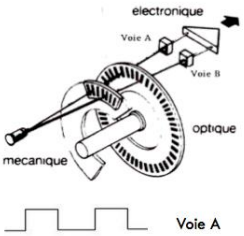


Figure 14 : Principe du codeur [4]

Voici l'illustration du principe de ce dernier.
Le fait d'avoir deux voies permet de déterminer le sens de rotation.

Enfin soulignons que notre encodeur rotatif possède en plus un bouton poussoir, il suffit d'enfoncer la molette pour l'utiliser.

4.4.3 Remarques

Terminons par souligner que quand on utilise un microcontrôleur il est de bonne pratique d'utiliser des résistances pour limiter le courant qui circule sur les lignes, à différencier avec les résistances dites de pull up/down, qui servent-elles à fixer l'état de la ligne quand rien ne se passe.
Il est aujourd'hui courant dans la plupart des microcontrôleurs d'avoir des résistances internes qu'on peut activer pour faire office de pull up, mais à contrario il n'y a pas beaucoup de protections quand il s'agit de limiter le courant. Il faut donc veiller, lors de l'utilisation, à ne pas connecter de composants directement en sortie du microcontrôleur, sous peine, dans le meilleur des cas, d'endommager le composant et dans le pire des cas, d'endommager le microcontrôleur lui-même.
C'est très souvent au niveau des circuits d'alimentation que se placent des composants de protection mais il est souvent compliqué de les remplacer.

4.5 Entrées/sorties analogiques

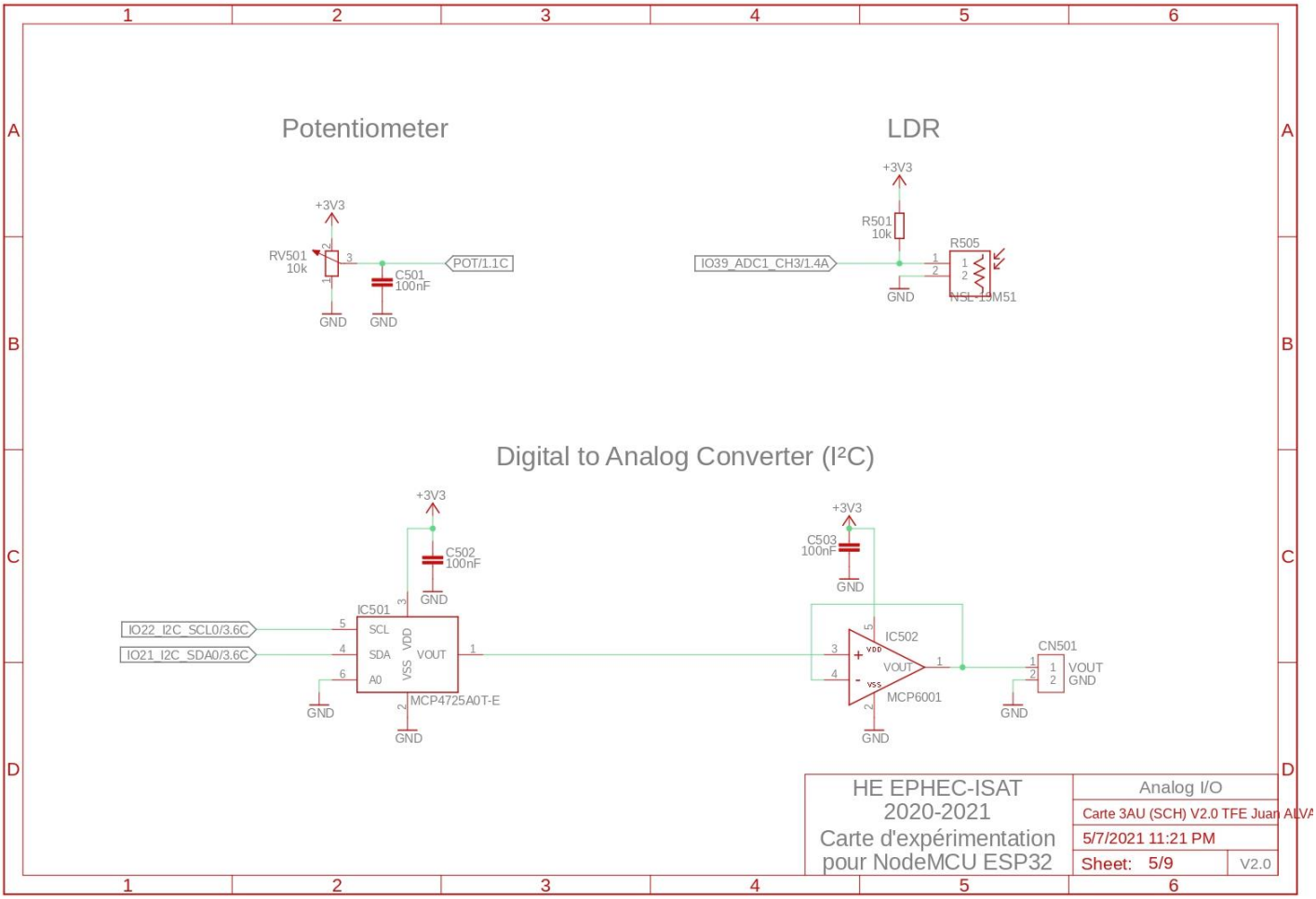


Figure 15 : Entrées / sorties analogiques

4.5.0 LDR (R505) et Potentiomètre (RV501)

Dans la lignée de ce qui a été expliqué pour les GPIO digitales à la rubrique précédente, attardons-nous cette fois sur les GPIO analogiques. Avant la généralisation de l'informatique et de la digitalisation telle qu'on la connait aujourd'hui, on concevait déjà des appareils électroniques complexes basés sur des systèmes dit analogiques.
En effet, pour le cas du potentiomètre, il s'agit de faire varier la résistance pour agir sur le courant. Nous ne sommes plus ici en présence de systèmes tout ou rien, nous ne pouvons donc plus représenter l'état d'une GPIO par une seule bascule. Ce sont des registres dédiés aux fonctions analogiques qui sont utilisés. Dans le cas de notre ESP32 c'est un registre de 12 bits, les valeurs pouvant aller de 0 à 4095 [31].
A noter que sur d'autres microcontrôleurs la taille de ces registres varie, ainsi sur un Arduino UNO ceux-ci sont de 10bits soit des valeurs comprises entre 0 et 1023.
La LDR pour « Light Dependent Resistor », est un type de résistance dont la valeur est dépendante de la luminosité, ainsi lorsqu'on expose une LDR à de la lumière celle-ci voit sa résistance chuter. Ce qui a pour conséquence de faire augmenter le courant qui la traverse, et c'est ce dernier que l'on exploite pour représenter la luminosité.[32]

4.5.1 DAC (IC501)

DAC est l'acronyme anglais de « Digital to Analog Converter » traduit et connu en français comme « Convertisseur numérique analogique » [27].
L'idée de son intégration à la carte 3AU vient de la volonté d'illustrer aux étudiants le fonctionnement des sorties analogiques. En effet, l'utilisation de DAC étant plus que courante, leur implémentation au sein même des microcontrôleurs est aujourd'hui quasi systématique.
Mais l'ESP32 a comme particularité de n'intégrer qu'un ADC (CAN en français) et aucun DAC [31]. Il reste néanmoins possible de piloter des sorties analogiques, pour ce faire, l'utilisation des fonctions de génération de PWM sont mobilisées. Cependant, leur utilisation demeure un peu plus complexe.
Il nous est donc venu l'idée d'intégrer un DAC et ainsi non seulement faciliter l'utilisation d'une sortie analogique mais en plus permettre l'illustration de ce principe aux étudiants. Le MCP6001 (IC502) est une AOP montée en suiveur, son intérêt est d'abaisser l'impédance de sortie du circuit [9].

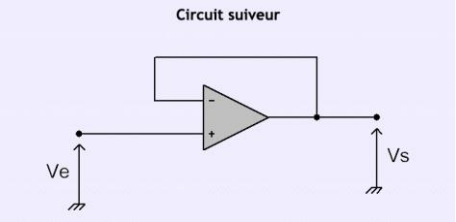


Figure 16 : Montage AOP suiveur [5]

Ci-contre, on voit l'illustration de ce genre de montage.

A son entrée, son impédance est en théorie infinie alors qu'à sa sortie elle est nulle.
Cette propriété permet de relier deux circuits sans que ce que l'on connecte d'un côté ne modifie l'état de ce qui se trouve de l'autre. L'idée ici, est de faire en sorte que l'impédance d'entrée du montage en aval n'ait aucun impact sur le signal produit par le DAC [9][27].

Le MCP4725 est donc un DAC, il reçoit ses signaux numériques de l'ESP32 par l'intermédiaire du protocole I2C. Bien qu'une rubrique entière ait été consacrée aux composants du bus I2C, il semblait plus pertinent de placer le composant dans cette rubrique puisqu'il sert à produire une sortie analogique.
Enfin, on remarque la présence de condensateurs de découplage sur l'alimentation du DAC et sur celle de l'AOP. Tout comme pour les précédents, ils visent à maintenir le niveau de la ligne et à supprimer les interférences, ce rôle est encore plus important dans le cas d'un DAC.
Si on voulait commander la luminosité d'une LED avec ce dernier par exemple, sans la présence des condensateurs, la lumière changerait d'intensité à la moindre interférence, c'est quelque chose qu'on cherche à éviter.

4.6 CAN

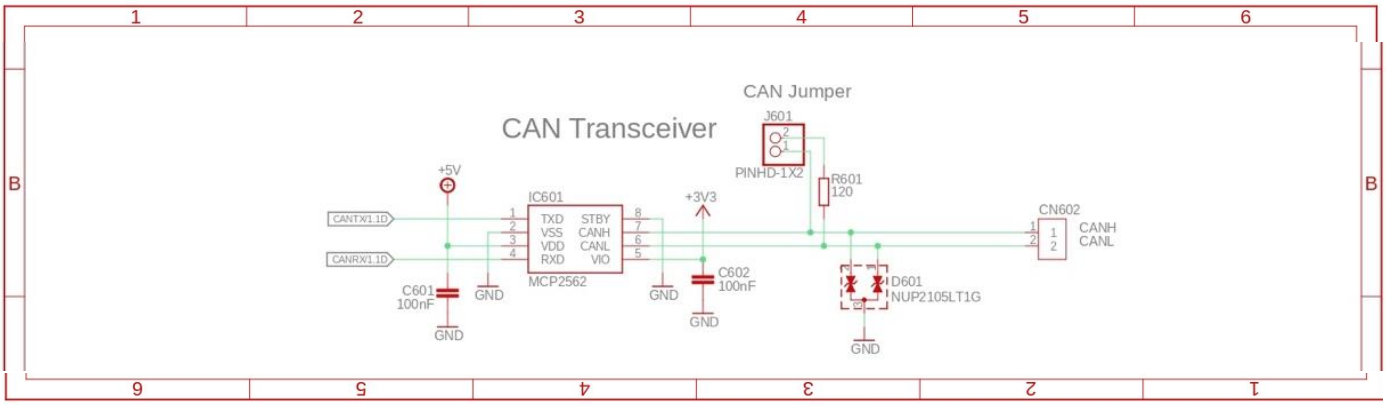


Figure 17 : CAN

4.6.0 Introduction au CAN

Si on se réfère à la datasheet de l'ESP32, on verra le nom « TWAI » pour « Two Wire Automotive Interface », c'est le nom que Espressif donne à son contrôleur CAN puisque ce dernier est sous brevet [31]. CAN pour « Control Area Network », est un bus de données développé par Bosch en 1985, aujourd'hui largement répandu dans l'industrie automobile, son utilisation s'est même étendue à d'autres domaines comme l'aéronautique par exemple [33].
Afin d'exploiter le bus il faut d'une part, un contrôleur CAN qui lui est intégré à l'ESP32 et d'autre part, un « transceiver », contraction anglaise de deux mots receiver et transmitter, traduit en français par émetteur/récepteur.

4.6.1 MCP2562 (IC601)

Le MCP2562 est donc le module qui permet l'envoi et la réception des trames, étant donné que les GPIO de l'ESP32 fonctionnent en 3V3 et que les lignes du CAN elles fonctionnent en 5V, l'utilisation d'un composant comme le MCP2562 qui permet l'utilisation de tensions différentes pour ces deux fonctions était indispensable.
D'autres composants accompagnent le module, déjà le NUP2105 (D601) qui est un suppresseur d'interférences, ensuite il y a un bornier (CN602), enfin un jumper (J601) permet d'ajouter une résistance de 120ohm entre les deux lignes qu'exploite le CAN.
Le bus CAN est ce qu'on appelle un bus différentiel, c'est une méthode qui permet le transport d'informations sur de longues distances (jusque 25 mètres). On réalise cela en codant l'information sur une paire de fils par différence réduisant ainsi les perturbations électromagnétiques, on entend par là que les deux fils véhiculent des tensions opposées [33]. Cette particularité du bus impose que la résistance de terminaison doit correspondre à l'impédance caractéristique de la ligne de transmission, cette impédance est de 120ohm, c'est pourquoi on utilise une résistance de cette valeur à chaque extrémité du bus. L'utilité de cette résistance est d'éviter de voir les signaux qui circulent sur la ligne se réfléchir aux extrémités [34].
Finalement, on notera encore la présence de condensateurs de découplage sur les lignes d'alimentation pour les mêmes raisons déjà évoquées.

4.7 MCP23017

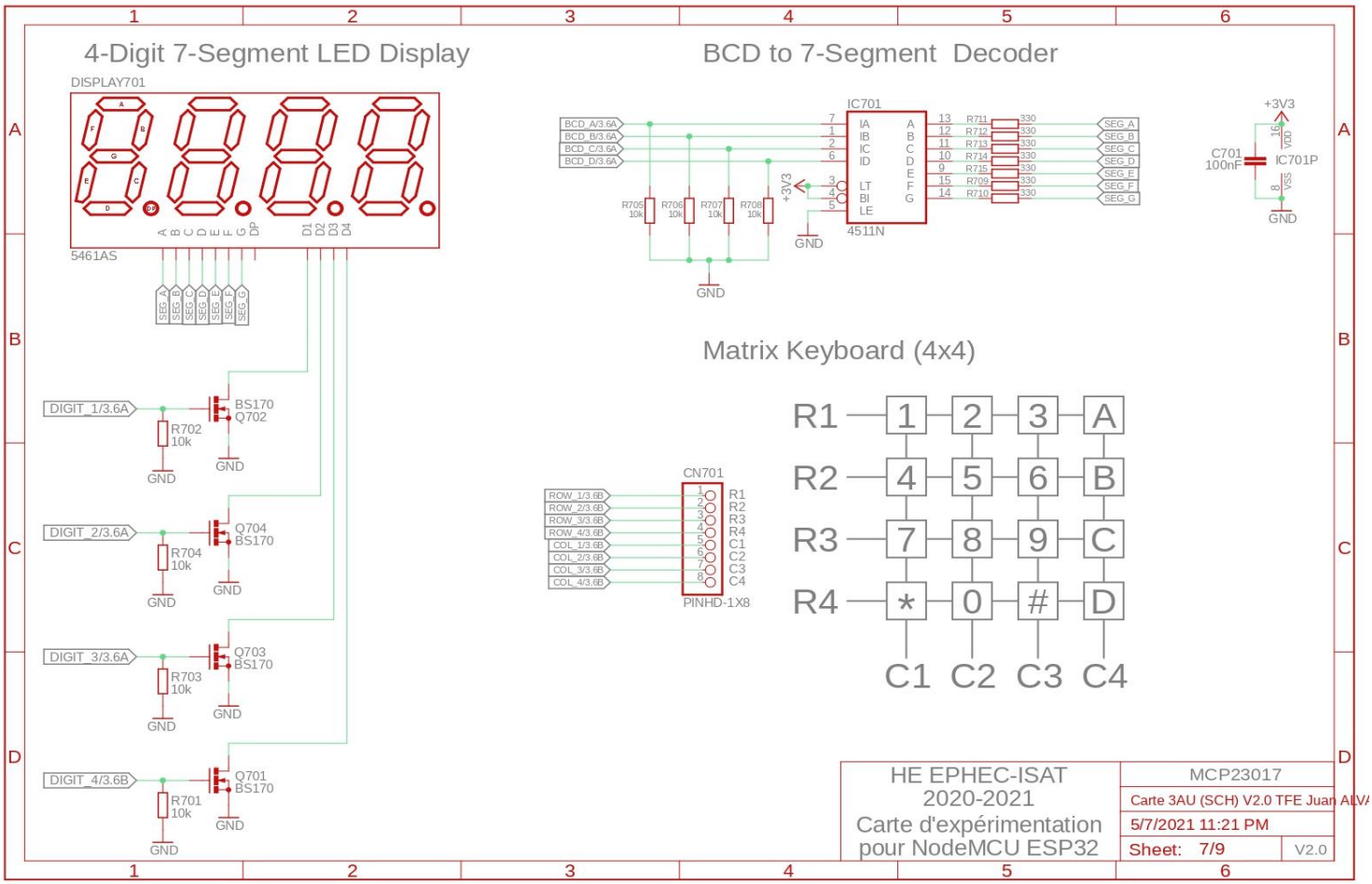


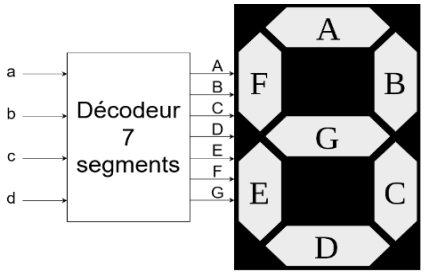
Figure 18 : MCP23017

La présentation du MCP23017 a été faite dans la rubrique 4.3.4, cette rubrique-ci comme annoncé, détaille les composants connectés au module.
Le choix de placer ces composants sur ce module trouve son origine dans la volonté de proposer aux étudiants, non seulement la possibilité d'apprendre à travailler avec un module d'extension d'entrées sorties mais aussi car il me permettait d'ajouter d'autres composants à la carte sans monopoliser des GPIO pour autant.

L'idée d'ajouter un deuxième MCP23017 a été envisagée mais fut malheureusement abandonnée faute de place. Cette envie d'ajouter un autre module vient de la volonté de permettre à l'utilisateur de la carte d'interfacer ses propres modules.
En l'état sauf si ce sont des composants prévus pour fonctionner via I2C, il est assez difficile d'ajouter des composants car les GPIO de l'ESP sont toutes utilisées.
C'est ici un axe d'amélioration potentielle de la carte.

4.7.0 Afficheur 7 segments à 4 digits (DISPLAY701)

L'afficheur 7 segments est un incontournable, il est aujourd'hui présent dans un grand nombre d'appareils présents dans notre quotidien comme les horloges numériques ou sur nos calculatrices, c'était tout naturel de le retrouver sur la carte 3AU. Le fonctionnement de ce genre d'afficheur est relativement simple en apparence, il suffit d'allumer les LEDs correspondant aux segments dont on a besoin pour dessiner les chiffres sur l'afficheur. C'est donc 7 connecteurs qui sont nécessaires, à quoi il faut ajouter 4 autres connexions pour sélectionner le digit sur lequel on souhaite écrire [27].
Pour le cas de la carte 3AU nous avons fait le choix de piloter la sélection du digit à travers des transistors MOSFET BS170 (Q701, Q702, Q702, Q704).
Pour piloter l'afficheur (d'une part pour s'éviter de consacrer 7 sorties à cette fonction et d'autre part car d'un point de vue pratique cela facilite son utilisation), le choix a été fait d'utiliser un décodeur « BCD to 7 segments » (IC701) où BCD signifie « Binary Coded Decimal » [27]. C'est un module qui permet la conversion de chiffres depuis une forme binaire en données exploitables par l'afficheur 7 segments. Notons qu'un 8ieme « segment » est disponible pour ajouter des points entre les digits mais nous ne l'avons pas connecté car nous n'avions pas assez de sorties sur le MCP23017.



L'image ci-contre permet d'illustrer le principe de fonctionnement. On voit bien ici que l'utilisation du décodeur n'est pas nécessaire mais qu'elle simplifie grandement le montage faisant passer les connexions de 7 à 4.

Figure 19 : 7 segment et décodeur BCD[6]

4.7.1 Clavier à matrice 4X4

Les 8 autres entrées du MCP23017 sont prévues pour l'utilisation d'un clavier à matrice 4X4, qui sera branché par l'étudiant sur le connecteur prévu à cet effet.

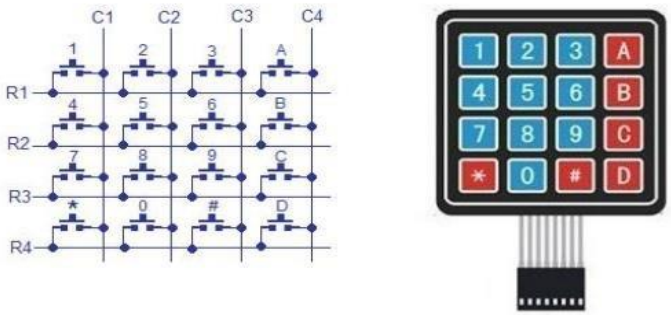


Figure 20 : Clavier à matrice [7]

Le fonctionnement du clavier consiste en une série de contacts, qui lient lignes et colonnes, ainsi l'appui sur la touche 1 lie la colonne 1 et la ligne 1. Les codes liés à son utilisation seront étudiés en détail par la suite. (Figure 17)
Ce genre de claviers font systématiquement partie des kits Arduino possédés par un grand nombre de personnes, mais il est surtout intéressant pour nous car le cours de robotique requiert que les étudiants s'en procurent un.

4.8 Autres

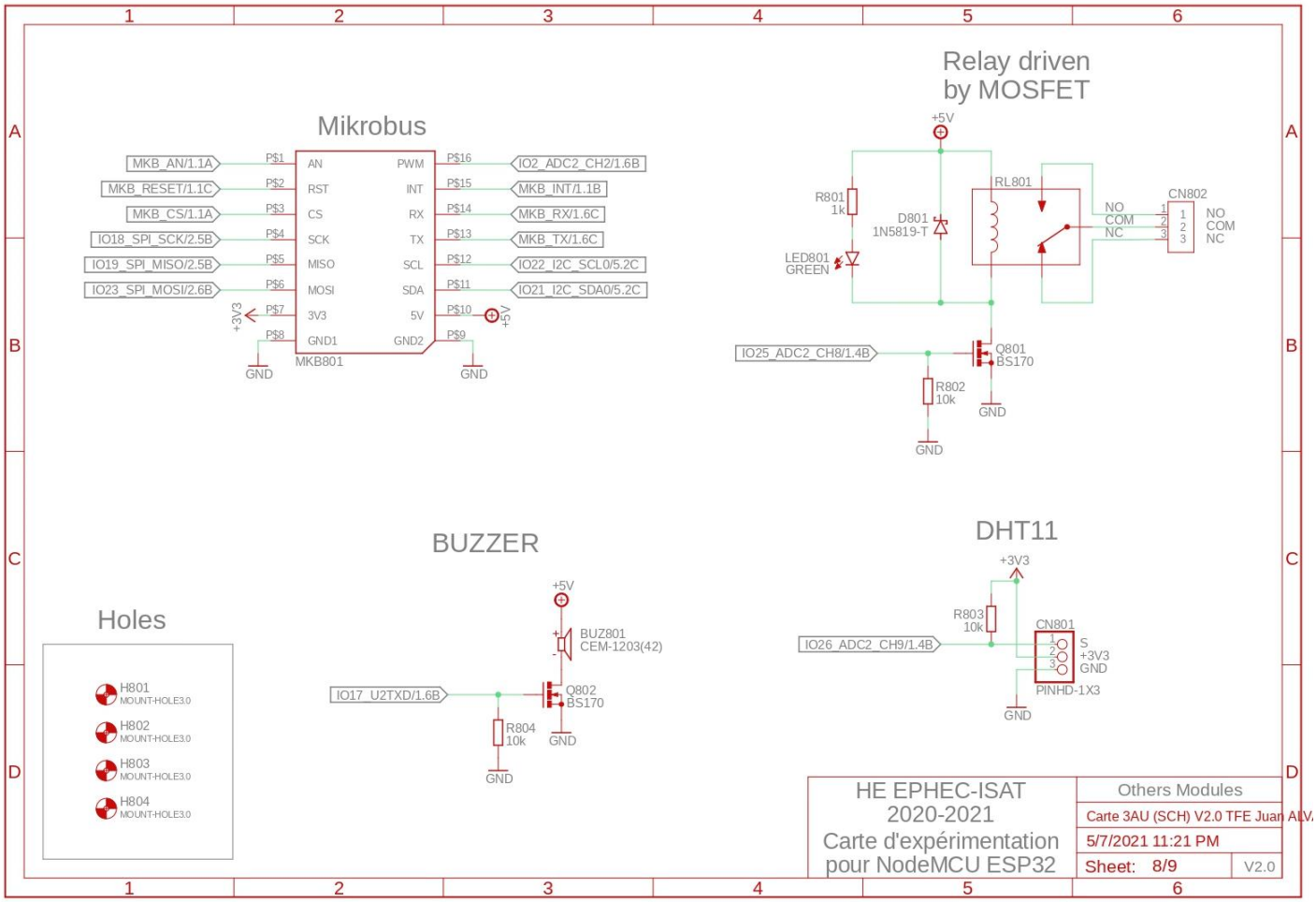


Figure 21 : Autres modules

4.8.0 Mikrobus (MKB801)

Le connecteur « Mirkobus » est l'un des composants les plus pratiques de la carte.
En effet, plutôt que d'ajouter de multiples modules, il a semblé particulièrement intéressant d'ajouter un connecteur basé sur un standard ouvert, développé par Mikroelectronika.
De nombreux modules sont compatibles avec le connecteur, et Mikroelectronika eux-mêmes développent ce qu'ils appellent des « clicboards », ce sont des cartes intégrant différents modules et compatibles avec le connecteur Mikrobus [35].
L'idée est donc de disposer d'un connecteur sur lequel on pourra venir ajouter différents types de modules en fonction des besoins.
Le connecteur prévoit toutes les connexions pouvant être utiles (SPI, I2C, UART, …).
Mais le plus intéressant, est que le standard fixe l'encombrement des clicboards, peu importe donc le module qu'on veut utiliser, on a la certitude qu'on peut l'interfacer facilement.

Cependant, le connecteur n'est pas destiné à être utilisé en permanence, en effet, les travaux porteront surtout sur l'utilisation des modules intégrés à la carte.

Dès lors vu la faible fréquence d'utilisation prévue, le choix a été fait de placer les différentes connexions du connecteur Mikrobus sur des jumpers.
Le but est de partager les connexions du Mikrobus avec d'autres fonctions (encodeur rotatif, potentiomètre et CAN). Permettant ainsi d'exploiter au maximum les GPIO que propose l'ESP32.

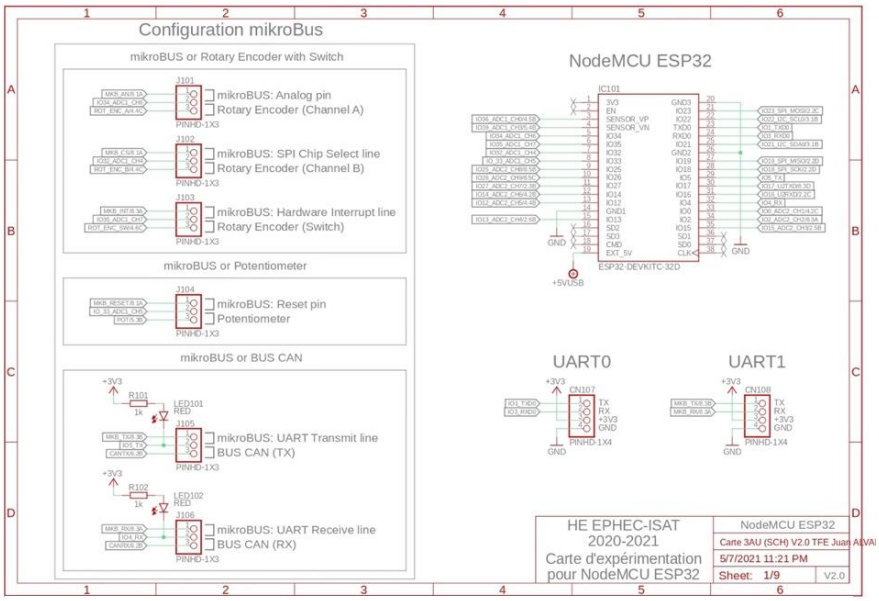


Figure 22 : Sheet1 - ESP32 et jumpers

Cette page ne contient que des informations structurelles, elle n'a donc pas été présentée en détail, cependant la compréhension de ce qui a été dit à propos des jumpers nécessite son introduction.
Outre les jumpers (sur la gauche), on remarquera les deux connecteurs UART0 et UART1, qui sont respectivement des connecteurs qui rendent disponible la liaison série de l'USB et la liaison série exploitée par le CAN et le connecteur Mikrobus.

4.8.1 Relais (RL801)

Si dans le cadre des travaux qu'ils réaliseront, les étudiants devaient commuter des charges puissantes, l'utilisation d'un relais serait adaptée. Il semble donc pertinent d'intégrer ce dernier à la carte. Il s'agit d'un relais 5V piloté à travers un transistor MOSFET BS170 (Q801). L'ajout de ce dernier est dû au fait que les GPIO de l'ESP32 fonctionnent en 3V3 et que ces relais nécessitent une tension de 5V, mais quand bien même ce serait un relais qu'on peut commuter avec moins de 5V, une pratique courante est de brancher le moins de modules possibles en sortie directe du microcontrôleur, l'idée étant d'essayer autant que possible de limiter le courant qu'on tire de l'ESP sans quoi, si un grand nombre de composants devaient fonctionner simultanément l'ESP ne pourrait pas tous les alimenter [15].

4.8.2 Buzzer (BUZ801)

La carte intègre également un buzzer, celui-ci est piloté à partir d'un signal carré où on fait varier la largeur d'impulsion. De la même manière que les autres composants, il est piloté par un MOSFET BS170 (Q802) [15].

4.8.3 DHT11 (CN901)

Un bornier pour brancher un autre capteur de température et d'humidité, le « DHT11 », est prévu. Ce capteur est le plus souvent monté sur un mini « PCB » (figure 23), qui intègre les résistances nécessaires à son fonctionnement. Néanmoins on remarque que la carte possède une résistance (R903) entre sa ligne de signal et d'alimentation, en effet d'autres modèles ne possédant aucune résistance existent, sans donc savoir quel modèle auront les étudiants, nous avons préféré assurer le coup. D'autant plus que surdimensionner cette dernière ne change en rien le fonctionnement du capteur.

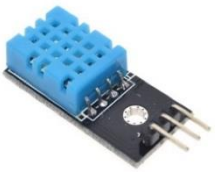


Figure 24 : DHT11 module sans résistance [8]

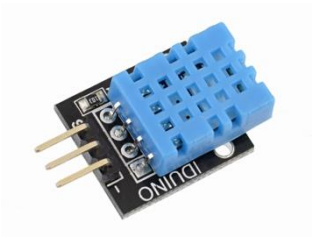


Figure 23 : DHT11 module avec résistance [9]

Soulignons que l'intérêt de ce capteur est qu'il est lui aussi systématiquement inclus dans les kits Arduino.

4.8.4 Remarques

Les composants qui se trouvent dans l'encadré « holes », sont les perçages prévus pour fixer les pieds de la carte. Ce ne sont pas des composants à proprement parler mais fusion360 impose de placer ces dernier dans le schéma.
L'ajout de pieds pour surélever la carte présente un double intérêt, le plus évident étant l'intérêt esthétique qu'ils apportent. L'autre est un peu moins évident mais autrement plus important. Rappelons que c'est sur le dessous de la carte que se trouvent toutes les soudures, dès lors si on pose par inadvertance la plaque lorsque cette dernière est sous tension sur une surface conductrice, on court-circuite tous les composants, risquant ainsi d'endommager voire de détruire la carte. C'est pourquoi l'ajout de supports sur ce genre de cartes est assez courant.
ENFIN il est important de noter que, dans la lignée de ce qui a déjà été dit au point 4.7 à propos des possibilités d'interfacer d'autres composants avec la carte, la GPIO 26 est pensée pour être utilisée avec un DHT11 mais cette dernière peut tout autant être exploitée par les étudiants pour interfacer d'autres modules. Puisqu'elle est exposée par le pin header destiné à accueillir le DHT. Cette remarque s'applique également au cas du connecteur du BMP280.

5 PCB et 3D

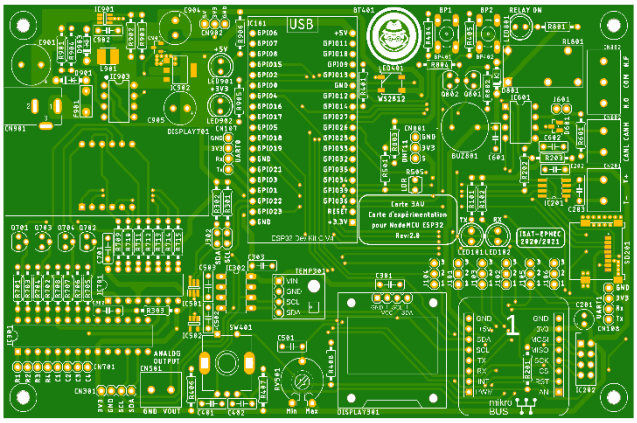


Figure 25 : Prévisualisation de production

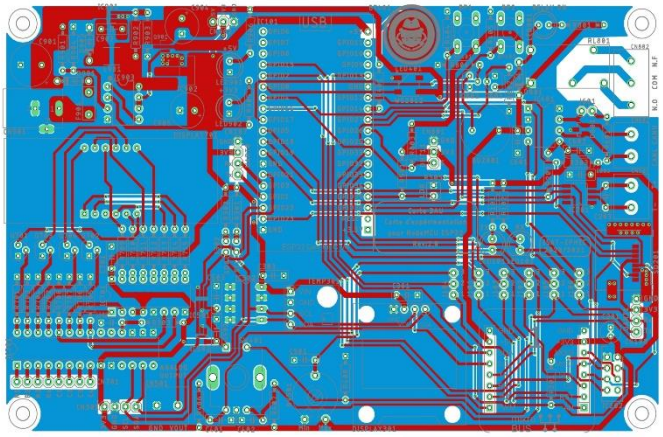


Figure 26 : PCB

Les versions originales de ces documents se trouvent en annexe (ANNEXE B et C).

5.1 Disposition

Le choix de placer l'ESP32 en plein milieu découle de plusieurs tentatives de placement, cette position centrale permet d'interconnecter plus facilement les différents modules au µC.
En haut à gauche, on retrouve tout ce qui est propre à l'alimentation, les pistes d'alimentation sont plus épaisses et à certains endroits particulièrement sensibles elles ont été agrandies au maximum.
En bas à gauche se situent tous les composants reliés au bus I2C alors qu'en bas à droite, il y a les composants reliés au bus SPI.
Finalement, sur la partie supérieure droite se trouvent les autres composants, on notera la présence des jumper évoqués en 4.8.1 juste au-dessus du connecteur Mikrobus, et les différents borniers disposés le long de la face droite.
A noter que la carte intègre quelques composants dit « SMD » acronyme anglais traduit en français par « monté en surface », on entend par là, qu'à la différence des composants traversant (Figure 27) qui se soudent du côté opposé où ils se trouvent, les composants SMD se soudent sur la même face où ils sont placés (Figure 28).

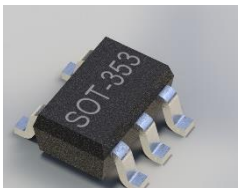


Figure 27 : composants SMD

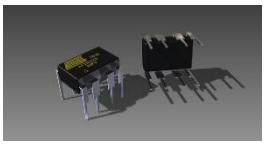


Figure 28 : composants traversants

L'idée poursuivie tout au long de la conception du PCB était de rapprocher au maximum les composants de zones spécifiques. On le voit clairement avec l'alimentation, tous les composants sont placés dans le coin supérieur gauche, en essayant de réduire au maximum la longueur des pistes et en collant les condensateurs le plus possible à l'endroit où ils se trouvent sur les schémas.

Enfin il peut être intéressant de remarquer que, dans la mesure du possible, les zones prévues pour connecter la carte avec d'autres cartes ou modules, ont été placées sur les bords. Il en est de même pour les composants destinés à interagir avec l'utilisateur.

5.2 Routage

Le routage de la carte a fait l'objet de longues réflexions en vue de l'optimiser, les différentes versions feront même l'objet d'une rubrique entière mais pour l'instant restons sur cette version 2.0. On voit en rouge toutes les pistes de cuivre sur la face dite « top », c'est-à-dire la face du dessus. (Figure 26)
Rappelons que le processus de production de ce genre de plaques, est de prendre une plaque entièrement recouverte d'une couche de cuivre et de venir retirer la matière superflue. Dès lors on comprend bien que laisser plus de cuivre, ne change en rien le processus [36].
La face dite « bottom », face du dessous, est quasi entièrement dédiée au plan de masse, on appelle ainsi la piste très élargie qui relie les composants à la masse, sur la Figure 26 on peut constater que quasi toute la carte apparait en bleu, c'est là notre plan de masse qui, comme évoqué, occupe la face du dessous.
De cette façon les composants sont liés entre eux par le dessus, et reviennent à la masse par le dessous. L'idée est d'avoir un plan de masse le plus grand possible et de préférence ininterrompu afin de « réduire le trajet que font les signaux électrique » [37].
Pour mieux comprendre introduisons le terme « EMI », pour « Electromagnetic interference », qui désigne donc les interférences électromagnétiques. Sans nécessairement rentrer dans les détails, soulignons simplement que notre PCB, comme tout circuit électrique, est sensible aux perturbations électromagnétiques. Il y a ici deux choses à surveiller, la longueur des boucles que font les signaux électriques, qui est un paramètre important notamment lorsqu'on regarde la résistance aux EMI, et le deuxième paramètre qui entre en jeu est la forme que possèdent les pistes [38]. Ainsi il faut éviter d'interrompre le plan de masse mais aussi éviter qu'il soit trop sinueux, c'est-à-dire le concevoir de sorte qu'il occupe au maximum la face du dessous.

Il est pertinent de souligner que pour limiter l'effet des EMI, un dégagement des pistes d'alimentation des composants a été mis en place, ce dégagement est d'autant plus important que les courants qui circuleront sur ces pistes seront élevés. C'est pourquoi lorsqu'on regarde le relais on remarque directement le dégagement.

Terminons par préciser que lors de la conception il est de bonne pratique d'éviter au maximum d'avoir des pistes possédant des angles droits. Fusion 360 intègre même une fonctionnalité permettant de vérifier les conditions d'angle.

5.3 Modèle 3D

Enfin avant d'aborder la modélisation 3D, revenons un peu plus en détail sur ce qui a déjà été dit sur Fusion360. Autodesk propose un grand nombre de bibliothèques mais d'une part tous les composants ne se trouvent pas systématiquement dedans et d'autre part même les composants déjà présents dans cette liste n'ont pas toujours de modèles 3D liés. Ainsi afin de pouvoir construire un modèle 3D, il a d'abord fallu créer une bibliothèque personnalisée.
Un outil précieux pour réaliser cette tâche a été « GRABCAD », un site où sont publiés de nombreux modèles développés par une large communauté de manière bénévole et mis à disposition [39].
Lors de la réalisation du PCB, on emploie des encombrements, « footprint » en anglais, ainsi on travaille avec l'espace que prennent les composants. Il suffit alors de lier cet encombrement au modèle 3D.

Tout le contenu du projet fusion360 est disponible dans le dossier technique.

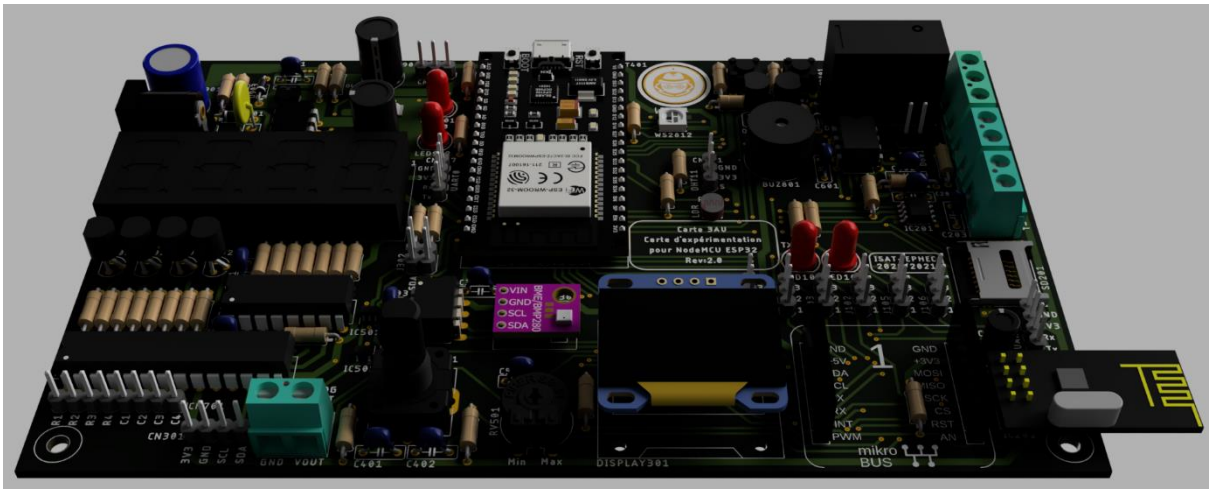


Figure 29 : Modèle 3D

Pour finaliser cette phase de conception, le passage par une étape de présentation est quasi systématique dans l'optique de présenter un produit le plus complet possible.
Voici un support destiné à l'écran OLED à titre d'exemple

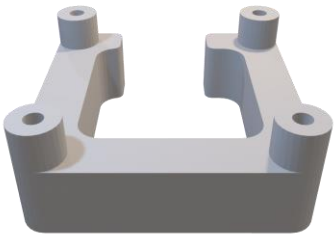


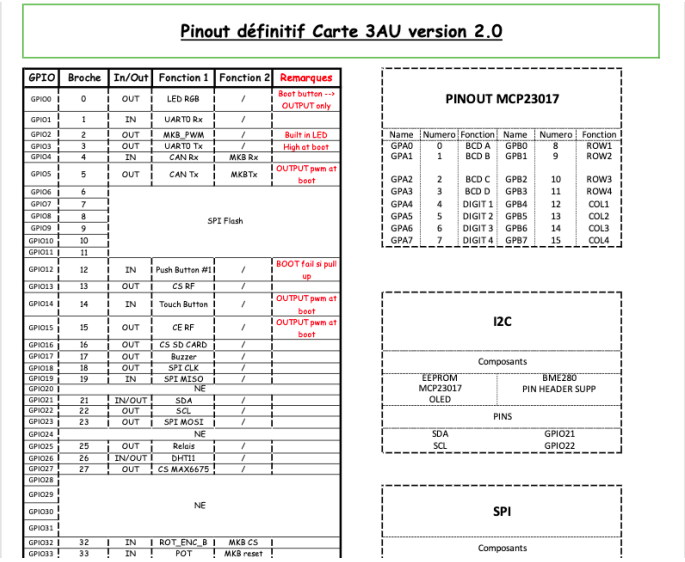
Figure 30: Support OLED

Destiné à être vissé sur la carte pour fixer l'écran OLED sans quoi ce dernier se retrouve suspendu et fragilisé.
Parmi les autres modèles développés, se trouve un boitier destiné à accueillir la carte.
Toutes ces conceptions sont présentes dans le projet fusion exporté qui se trouve dans le dossier technique.

6 Dossier technique

On aborde ici l'une des dernières rubriques de ce travail, elle n'en est pas moins importante. L'idée est de fournir une documentation complète aux étudiants pour qu'ils puissent prendre en mains la carte. Dans ce dossier se trouvent les informations détaillées de tous les composants, ainsi qu'un certain nombre de documents auxquels j'ai déjà fait référence.

6.1 Pinout



Pinout définitif Carte 3AU version 2.0

GPIO	Broche	In/Out	Fonction 1	Fonction 2	Remarques
GPIO0	0	OUT	LED RGB	/	Boot button --> OUTPUT only
GPIO1	1	IN	UART0 Rx	/	
GPIO2	2	OUT	MKB_PWM	/	Built in LED
GPIO3	3	OUT	UART0 Tx	/	High at boot
GPIO4	4	IN	CAN Rx	MKB Rx	
GPIO5	5	OUT	CAN Tx	MKBTx	OUTPUT pwm at boot
GPIO6	6				
GPIO7	7				
GPIO8	8		SPI Flash		
GPIO9	9				
GPIO10	10				
GPIO11	11				
GPIO12	12	IN	Push Button #1	/	BOOT fail si pull up
GPIO13	13	OUT	CS RF	/	
GPIO14	14	IN	Touch Button	/	OUTPUT pwm at boot
GPIO15	15	OUT	CE RF	/	OUTPUT pwm at boot
GPIO16	16	OUT	CS SD CARD	/	
GPIO17	17	OUT	Buzzer	/	
GPIO18	18	OUT	SPI CLK	/	
GPIO19	19	IN	SPI MISO	/	
GPIO20			NE		
GPIO21	21	IN/OUT	SDA	/	
GPIO22	22	OUT	SCL	/	
GPIO23	23	OUT	SPI MOSI	/	
GPIO24			NE		
GPIO25	25	OUT	Relais	/	
GPIO26	26	IN/OUT	DHT11	/	
GPIO27	27	OUT	CS MAX6675	/	
GPIO28					
GPIO29					
GPIO30			NE		
GPIO31					
GPIO32	32	IN	ROT_ENC_B	MKB CS	
GPIO33	33	IN	POT	MKB reset	

PINOUT MCP23017

Name	Numero	Fonction	Name	Numero	Fonction
GPA0	0	BCD A	GPB0	8	ROW1
GPA1	1	BCD B	GPB1	9	ROW2
GPA2	2	BCD C	GPB2	10	ROW3
GPA3	3	BCD D	GPB3	11	ROW4
GPA4	4	DIGIT 1	GPB4	12	COL1
GPA5	5	DIGIT 2	GPB5	13	COL2
GPA6	6	DIGIT 3	GPB6	14	COL3
GPA7	7	DIGIT 4	GPB7	15	COL4

I2C

Composants	
EEPROM MCP23017 OLED	BME280 PIN HEADER SUPP
PINS	
SDA SCL	GPIO21 GPIO22

SPI

Composants

Figure 31 : PINOUT

Un premier document qu'on peut évoquer est celui qui détaille l'utilisation des différentes PINs de l'ESP32. Ce document a connu plusieurs versions avant d'aboutir à celle présente en annexe (ANNEXE E). Ce sont des documents qui ont été produits avant d'entamer la conception et qui ont ensuite été modifiés à chaque avancée du projet. L'objectif suivi lors de la conception de la carte était d'exploiter au maximum l'espace disponible, ainsi l'utilisation de toutes les GPIO était souhaitée.
Il faut savoir que les GPIO n'ont pas toutes les mêmes propriétés, ces caractéristiques propres à chaque pin sont documentées par Espressif eux-mêmes et dans l'esprit de fournir des documents complets, le PINOUT détaille non seulement l'utilisation qui est faite de chaque pin mais aussi les caractéristiques de ces dernières.

6.2 BOM et datasheets

Une liste du matériel intégré à la carte est disponible, avec pour ceux qui voudraient essayer de produire leur propre carte un classeur Excel qui fournit les liens vers le catalogue de Farnell. En plus un dossier « datasheet » contient toutes les datasheets originales des différents composants intégrés à la carte.
Enfin ces différentes datasheets sont complétées par des documents qui expliquent certains principes, notamment en ce qui concerne l'alimentation.

6.3 Documents de conception

Le projet fusion est disponible en version exportée, mais afin de permettre d’accéder facilement au différents footprint et modèles 3D des composants utilisés. Un dossier est consacré au contenu du projet Fusion360. Il contient les différentes versions de la carte, la totalité des bibliothèques éditées et tous les modèles 3D.

6.4 Etude du prix

	MCP23017	mcp23017	farnell	MCP23017-E/SP - Microchip - Expander I/O, 16 bits, 1.7 MHz (farnell.com)	1,21	1		1,21	total i2c	5,21
AN	encodeur rotatif	PEC11R alternatif	Farnell	PEC11R-4120F-S0018 - Bourns - Codeur incrémental, Axe 20mm, 5 VDC (farnell.com)	1,97	1		1,97		
	encodeur rotatif différé	PEC11R		PEC11R-4215F-S0024 - Bourns - Codeur incrémental, Pro Audio, continu. 24 détentes (farnell.com)	1,48	0		0		
	pot	PT10	farnell	https://be.farnell.com/fr-BE/amphenol-piher-sensorscontrols/pt6mv-103a2020/trimmer-10k-0-1w-1-tour /dp/3128589?st=pt%2010	0,832	1		0,832		
	~~pot~~		~~Farnell~~	~~PTV09A-4020F-B104 - Bourns - Potentiomètre rotatif, 9 mm, Carbone (farnell.com)~~	~~0,823~~	~~0~~		0		
	LDR	GL5528	amazon par 100 à 10	Photorésistance,Xiuyer 100 Pièces 5mm Résistance Dépendant de la Lumière GL5516 GL5528 Capteur de Luminosité LDR: Amazon.fr: Commerce, Industrie & Science	0,1	1		0,1	total AN	2,90:
autre	Bouton	176432	farnell	B3F-1000 Omron, Commutateur tactile, Série B3F, A déclencheur supérieur | Farnell FR	0,23	1		0,23		
	TMB 3		Farnell	CTB55VG/3 - Camdenboss - Bornier standard, Série CTB, 3 Contact(s) (farnell.com)	0,345	1		0,345		
	Terminal Block 2		Farnell	CTB55VG/2 - Camdenboss - PCB TERMINAL BLOCK, LOW PROF, 5MM (farnell.com)	0,242	2		0,484	total autre	1,05!
relais mos	~~relais 5v~~	-	~~mouser~~	~~V23079G2001X071 TE Connectivity | Mouser Belgique~~	~~2,13~~	~~0~~	~~TROP~~	0		
	relais 5V		Farnell	ALQ105 Panasonic, Relais à usage général Série ALQ Puissance, Non verrouillable, SPDT | Farnell FR	1,02	1		1,02		
	mosfet	bs170	Farnell	BS170-D27Z - On Semiconductor - MOSFET de puissance, Canal N, 60 V (farnell.com)	0,421	5	vendu par 5	2,105		
	transistors bs170 7seg			-		4			total relais	3,12!
~~dip switch~~	~~switch pull up~~		~~farnell~~	~~ADE0204 Te Connectivity, Commutateur DIP / SIP, 2 Circuit(s), Glissière | Farnell FR~~	~~0,55~~	~~0~~	~~vendu par 10 50c~~	~~0~~		
~~switch~~	~~switch mkb~~	-	~~farnell~~	~~JS102011JAQN C & K Components, Commutateur à glissière, Subminiature, SPDT | Farnell FR~~	~~0,45~~	~~0~~	~~vendu par 10 50c~~	~~0~~	total switch	0
CAN	mcp2562		farnell	2-e-p/ic-can-tranceiver-high-speed-8pdip/dp/2362838?ost=MCP2562-E%2FP&exaMfpn=true&searchref=s	0,935	1		0,935		
	nup2105	supp int	farnell	NUP2105LT1G On Semiconductor, Diode TVS, Série NUP21, Bidirectionelle | Farnell FR	0,535	1	vendu par 5	0,535	total CAN	1,47
mcp23017	7seg 4digits		Amazon	https://www.amazon.fr/ARCELI-Segments-Cathode-dAffichage-Num%C3%A9rique/dp/B07J2PX5XM/ref=sr_1_14?__mk_fr_FR=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=1AXW1E7SNSMLJ&dchild=1&keywords=afficheur+7+segments&qid=1615462631&quartzVehicle=93-1184&replacementKeywords=afficheur+segments&sprefix=afficheur+7%2Caps%2C166&sr=8-14	1,398	1	vendu par 5 6,99	1,398		
	decodeur bcd		Farnell	CD74HCT4511E - Texas Instruments - Décodeur BCD vers 7 Segments, Driver, Verrou (farnell.com)	0,829	1		0,829		

Figure 32 : BOM

Comme évoqué dans l’introduction, l’optimisation du prix de la carte a été un sujet de réflexion lors de la conception du projet. C’est pourquoi le fichier Excel qui fournit les liens vers le site de Farnell, propose un calcul du prix de la carte. (ANNEXE F)
L’intérêt de ce document est de proposer une idée du prix par groupe de composants ainsi qu’un prix global. Ce n’est pas précisé mais les chiffres contenus dans les colonnes de droite sont les prix exprimés en euros.

6.5 Codes

La prise en main de la carte passe par la compréhension des différentes possibilités qu'offre cette dernière, c'est pourquoi proposer un dossier technique qui contient des exemples d'utilisation présente un intérêt tout particulier. Cette rubrique se divise en deux parties, en premier lieu on abordera les codes dit « de test », ce sont des codes produits pour chaque composant et qui visent à tester la fonctionnalité de ces derniers.
Ensuite viendra une deuxième partie dite « avancée », où ce sera un exemple de projet complexe réalisable qui sera présenté.

6.5.0 Présentation des outils de travail

Afin de bien comprendre la suite, il faut introduire les outils qui ont servi dans la réalisation des différents codes.

1) GitHub et Git

Git est un système de « versioning » gratuit et open source.

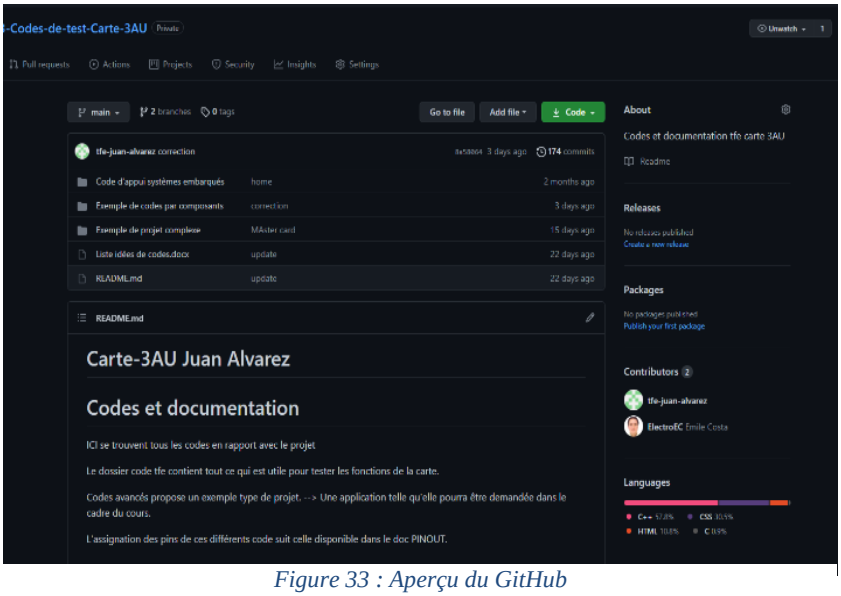


Figure 33 : Aperçu du GitHub

Ainsi combiné à GitHub, son système de stockage en cloud, ils permettent la gestion de projets en équipes. En effet, chaque membre peut travailler sur une version personnelle du projet grâce à Git, il suffit ensuite de fusionner les projets locaux à celui sur GitHub.
De cette manière plusieurs membres peuvent travailler sur un même projet simultanément et la gestion des conflits entre versions est gérée par Git, c'est ça qu'on appelle versioning [40] [41].
Dans le cadre de ce travail, un GitHub dédié au TFE a été créé, il contient un répertoire avec tous les codes et notes explicatives nécessaires pour faire fonctionner la carte.
Le lien vers le GitHub est disponible dans la bibliographie.

2) Visual Studio Code

Pour produire les différents codes, l'utilisation de ce qu'on appelle un « éditeur de code » est requise, allant du plus simple comme Notepad à d'autres plus complets comme Visual Studio. C'est un éditeur et un compilateur développé par Microsoft et pensé pour être optimisé, ainsi il intègre tout un tas de fonction particulièrement utiles, dont notamment la compatibilité avec Git [42].

3) PlatformIO

En parallèle, VScode permet aussi l'utilisation d'extensions qui améliorent l'expérience d'utilisation. Ainsi PlatformIO est une de ces extensions, c'est un environnement de programmation qui permet la programmation d'une large game de microcontrôleurs dont l'ESP32. PlatformIO présente un intérêt certain, déjà par les fonctionnalités qu'il intègre comme l'auto-complétion, qui est le fait de compléter automatiquement les mots qu'on écrit. Mais il est surtout intéressant car si quelqu'un dispose du projet, il peut directement compiler ce dernier sans besoin d'installer manuellement les bibliothèques. Toutes les informations nécessaires pour télécharger et installer les bibliothèques nécessaires sont dans « platformio.ini », qui est un fichier propre à l'environnement de PlatformIO [43].

La totalité des ressources présentes sur GitHub sont reprises dans la documentation technique, cependant l'avantage de leur présence sur GitHub est d'une part de permettre l'ajout et la modification des codes en fonction des éventuelles mises à jour de la carte, et d'autre part, du fait de la nature collaborative de GitHub, de permettre à d'autres personnes d'accéder aux documents.

6.5.1 Codes de test

Le détail de tous les codes ne sera pas étudié dans cette rubrique, ce serait trop long et peu utile, Dès lors seul quelques codes qui mettent en valeur un maximum de fonctionnalités de la carte seront abordés.

1) OLED – DHT

Ce code permet de prouver le fonctionnement du capteur de température DHT11 et de l'écran OLED en affichant les valeurs enregistrées par le capteur sur l'écran.
A ce stade il faut noter que la totalité des codes présents dans cette rubrique sont assez simple, leur but étant avant tout de servir d'outil de diagnostic pour vérifier les fonctions de la carte.
Analysons donc sa structure :

Une première partie systématique est d'inclure les bibliothèques utiles au projet, en effet lors de la compilation, le compilateur va systématiquement compiler les dossiers spécifiés par la balise « #include ». On remarquera que certaines bibliothèques sont construites avec plusieurs fichiers source, c'est notamment le cas quand il s'agit de grandes bibliothèques où il n'y a pas de nécessité de posséder la totalité de la bibliothèque pour que le code fonctionne. Cela s'inscrit dans la lignée de ce qui a été expliqué précédemment, à savoir que le langage C est un langage prisé dans le monde de l'informatique embarquée grâce à l'optimisation qu'il permet, surtout en termes de mémoire.

```
/************************** OLED + DHT*******************************/
#include <Arduino.h>
#include <SPI.h>
#include <Wire.h>
/* Bibliothèques DHT --> adafruit propose la gestion de nombreux autres capteurs*/
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
/* Les deux bibliothèques indispensables pour exploiter l'écran OLED*/
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

Ensuite sont introduites les différentes variables utiles au programme, on appelle ça des déclarations. Dans celles-ci apparaissent des lignes un peu différentes puisque ce ne sont pas des variables qui sont déclarées mais plutôt des « objets », on appelle cela une « instance ». En effet en programmation il est courant de travailler avec des classes, ce sont des codes définissant les attributs (les propriétés) et les méthodes (le comportement) d'ensembles d'objets. L'intérêt d'avoir des objets étant de pouvoir à partir d'un simple code régissant leur comportement, en créer plusieurs sans pour autant alourdir le code. Ce genre de programmation est dite « orientée objet », elle est très répandue et particulièrement utile quand on veut produire des codes facilement réutilisables, en effet une fois le code relatif à la classe écrit, il suffit d'inclure ce dernier, comme pour les bibliothèques, et simplement instancier un objet.
Ainsi « display » est une instance de la classe Adafruit_SSD1306 par exemple.
Par la suite on pourra exploiter les méthodes de la classe Adafruit_SSD1306 pour utiliser l'objet display.

```
#define SCREEN_WIDTH 128 // Largeur du LCD OLED
```

```
#define SCREEN_HEIGHT 64 // Hauteur du LCD OLED
#define OLED_RESET     -1 // Pin de reset du LCD
#define SCREEN_ADDRESS 0x3C // Adresse du LCD souvent 0x3C
// Instance de SSD1306 display  I2C
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
// DHT11
#define DHTPIN 26
#define DHTTYPE    DHT11
DHT_Unified dht(DHTPIN, DHTTYPE); // Instance  DHT1
/* Variables programme*/
int send_time;  // timer envoi de données
char dht_buffer [16];
```

S'en suit la structure classique d'un programme Arduino avec une fonction setup et une fonction loop. Dans le setup sont configurés les différents modules et enfin dans la loop se trouve le code destiné à tourner en boucle dans notre cas : prise de donnée → envoi vers l'écran une fois par seconde.

```
void setup() {
  Serial.begin(9600);
  dht.begin();
  // SSD1306_SWITCHCAPVCC --> Génère le 3V3 en interne
  if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Boucle infinie si OLED n'a pas démarré
  }
  display.display();

  delay(2000); // Pause for 2 seconds
}
```

A ce stade le code ne fait encore rien, mais il est opportun d'illustrer ce qui é été introduit à propos des classes ainsi « begin » est une méthode de la classe Adafruit_SSD1306 appliquée à l'objet display, tout comme display en est également une. Pour appliquer une méthode à une classe il suffit donc d'écrire le nom de l'instance de l'objet suivi d'un point puis la méthode à appliquer avec les éventuels paramètres.

```
void loop() {
  if(send_time <= millis()){
/* Récupération des données du capteur*/
    sensors_event_t event;
    dht.temperature().getEvent(&event);
    float t = event.temperature;
    if (isnan(event.temperature)) {
      Serial.println(F("Error reading temperature!"));
    }
    else {
      Serial.print(F("Temperature: "));
      Serial.print(event.temperature);
      Serial.println(F("°C"));
    }

// Get humidity event and print its value.
    dht.humidity().getEvent(&event);
```

```
  float h = event.relative_humidity;
  if (isnan(event.relative_humidity)) {
   Serial.println(F("Error reading humidity!"));
  }
  else {
   Serial.print(F("Humidity: "));
   Serial.print(event.relative_humidity);
   Serial.println(F("%"));
  }
  delay(100);
 /*A présent on modifie l'affichage sur le LCD*/
  display.clearDisplay();

  display.setTextSize(1); // Un char vaut 6*8 pixels
  display.setTextColor(SSD1306_WHITE);
  display.cp437(true);    // Fixe la police
  display.setCursor(5, 0); // Place le curseur
  display.println("Valeur Humidite :"); // Fixe le texte à afficher
  display.setCursor(32,12);
  display.setTextSize(2); // Fixe la taille
  display.println(h);
  display.drawLine(0,30,128,30,WHITE);
  display.setCursor(5,36);
  display.setTextSize(1);
  display.println("Valeur Temperature :");
  display.setCursor(32,48);
  display.setTextSize(2);
  display.println(t);

/* finalement on actualise*/
  display.display();
  send_time = millis() + 1000; // On fixe la prochaine lecture --> ici 1s
 }
}
```

C'est ici la structure classique d'un code pour ESP32 développé avec le Framework Arduino.

2) IOAbstraction

Parmi les bibliothèques qui ont le plus servi, « IOabstraction » a été la plus utile.
Cette dernière se base sur TaskManagerIO, qui est un gestionnaire de tâches. C'est une bibliothèque qui propose des fonctions pour intégrer plusieurs objets présents sur la Carte 3AU, citons le cas du MCP23017.

Le code qui suit est une vérification de mot de passe : l'utilisateur appuie sur les touches du clavier à matrice et ça s'affiche sur le 7 segments. Ensuite si le code est correct, le relais commute.
On retrouve les inclusions systématiques en tête du code avec ensuite les déclarations et les différentes instances d'objets.

```
/************************** MCP23017*******************************/
#include <Arduino.h>
#include <Wire.h>
```

```
#include <IoAbstraction.h>
#include <IoAbstractionWire.h>
#include<TaskManagerIO.h>
#include <KeyboardManager.h>
#define relais 25
// Keypad I/O with MCP23017
#define ROW1 8  // GPB0
#define ROW2 9  // GPB1
#define ROW3 10 // GPB2
#define ROW4 11 // GPB3
#define COL1 12 // GPB4
#define COL2 13 // GPB5
#define COL3 14 // GPB6
#define COL4 15 // GPB7

MAKE_KEYBOARD_LAYOUT_4X4(keyLayout)
//keyboard manager class
MatrixKeyboardManager keyboard;
IoAbstractionRef ioExpander = ioFrom23017(0x20);

int cpt;
char code[4];
bool flag;
```

On a ici un exemple de création de classe. La création de myKeyboardListener à partir de l'instance d'une autre classe où nous avons redéfini deux méthodes. Le terme Public faisant référence à la portée des variables, on entend par là de déterminer depuis où on peut exploiter cette instance.

```
class MyKeyboardListener : public KeyboardListener {
public:
  void keyPressed(char key, bool held) override {
    Serial.print("Key ");
    Serial.print(key);
    Serial.print(" is pressed, held = ");
    Serial.println(held);
    code[cpt] = key;
    seg_write(cpt);
    cpt++;
  }
  void keyReleased(char key) override {
    Serial.print("Released ");
    Serial.println(key);
    if (cpt > 3){
      flag = 1;
      cpt = 0;
    }
  }
} myListener;
```

Ensuite on remarquera comment fonctionne le MCP2301, c'est très proche de la programmation des IO classiques. On définit le mode de fonctionnement (INPUT ou OUTPUT) puis on fixe l'état logique de la sortie ou on lit l'entrée.
L'utilisation du 7 segments à travers le décodeur BCD nécessite 8 sorties, c'est ce que font les lignes surlignées en vert.

On assigne ensuite les différents pins du MCP au différentes lignes et colonnes du clavier à travers « keyLayout », objet instancié précédemment.

```
void setup() {
   Wire.begin(); //Initialisation de l'I2C
   Serial.begin(9600);  //Initialisation de UART
   delay(1000);
   pinMode(relais,OUTPUT);                  //Fixe rôle relais
   for(int k = 0;k<8;k++){
      ioDevicePinMode(ioExpander, k, OUTPUT); //Fixe role IO MCP
   }
   keyLayout.setRowPin(0, ROW1);
   keyLayout.setRowPin(1, ROW2);
   keyLayout.setRowPin(2, ROW3);
   keyLayout.setRowPin(3, ROW4);
   keyLayout.setColPin(0, COL1);
   keyLayout.setColPin(1, COL2);
   keyLayout.setColPin(2, COL3);
   keyLayout.setColPin(3, COL4);
   keyboard.initialise(ioExpander, &keyLayout, &myListener);
   keyboard.setRepeatKeyMillis(850, 350);
   Serial.println("Keyboard is initialised!");
}
void loop() {
   taskManager.runLoop(); //Fonction qui actualise le gestionnaire de tâche
```

Pour bien comprendre le fonctionnement du code, il faut introduire le concept de fonction de « callback », qui se traduit en français par « rappel ». C'est une fonction qui est appelée automatiquement lorsque qu'un évènement se produit [44]. Ainsi c'est ici notre objet « myListener » qui définit ces fonctions de callback. La ligne surlignée en jaune initialise le clavier à matrice et définis sur quel module se trouve le clavier, comment sont assignées les lignes et les colonnes et enfin que faire lorsque qu'une touche est appuyée.
Précédemment j'ai introduit TaskManagerIO, en effet pour que le clavier puisse fonctionner il faut exécuter un certain nombre de taches dans un ordre précis.
La ligne surlignée en mauve permet donc l'exécution systématique de ces tâches.

Lorsque qu'on relâche une touche pour la quatrième fois, on active une variable « flag », c'est une pratique courante pour marquer l'exécution de morceaux de codes. C'est utilisé ici pour savoir quand vérifier si le code introduis est correct.
Enfin on remarquera, que tout comme pour le code précédent, l'utilisation de la fonction « millis », sert à programmer des fonctions sur une durée.

```
if (flag){
    cpt = 0;
    //Serial.println(code);
    cpt = millis() + 5000;
    while(cpt>millis()){
     for(int i = 0;i<=3;i++){
       seg_write(i);
       Serial.print(i);
       taskManager.runLoop();
       delay(100);
```

```
      }
     }
     Serial.print('\n');
     Serial.print("le code est : ");
     Serial.println(code);
     cpt = 0;
     flag = 0;
     if(code[0]=='3'&&code[1]=='2'&&code[2]=='1'&&code[3]=='0'){
      Serial.println("code ok");
      digitalWrite(relais,HIGH);
      delay(2000);
      digitalWrite(relais,LOW);
      ioDeviceDigitalWrite(ioExpander,4,LOW);
      ioDeviceDigitalWrite(ioExpander,5,LOW);
      ioDeviceDigitalWrite(ioExpander,6,LOW);
      ioDeviceDigitalWrite(ioExpander,7,LOW);
     }
    }
}
```

Dans le code présenté j'ai supprimé les deux fonctions qui permettent d'écrire sur le 7 segments, en effet ces dernières sont assez longues et répétitives, elles restent néanmoins utiles pour comprendre comment exploiter le MCP23017.

```
void seg_nb(int indice){
  if (code[indice]=='0'){
    ioDeviceDigitalWrite(ioExpander,0,LOW);
    ioDeviceDigitalWrite(ioExpander,1,LOW);
    ioDeviceDigitalWrite(ioExpander,2,LOW);
    ioDeviceDigitalWrite(ioExpander,3,LOW);
  }
  if (code[indice]=='1'){
    ioDeviceDigitalWrite(ioExpander,0,HIGH);
    ioDeviceDigitalWrite(ioExpander,1,LOW);
    ioDeviceDigitalWrite(ioExpander,2,LOW);
    ioDeviceDigitalWrite(ioExpander,3,LOW);
  }
  if (code[indice]=='2'){
    ioDeviceDigitalWrite(ioExpander,0,LOW);
    ioDeviceDigitalWrite(ioExpander,1,HIGH);
    ioDeviceDigitalWrite(ioExpander,2,LOW);
    ioDeviceDigitalWrite(ioExpander,3,LOW);
  }
```

Figure 35 : Fonction MCP23017 - 1

```
void seg_write(int digit){
  //Serial.print(digit);
  if (digit==0){
      ioDeviceDigitalWrite(ioExpander,4,LOW);
      ioDeviceDigitalWrite(ioExpander,5,LOW);
      ioDeviceDigitalWrite(ioExpander,6,LOW);
      ioDeviceDigitalWrite(ioExpander,7,HIGH);
  }
  if (digit==1){
      ioDeviceDigitalWrite(ioExpander,4,LOW);
      ioDeviceDigitalWrite(ioExpander,5,LOW);
      ioDeviceDigitalWrite(ioExpander,6,HIGH);
      ioDeviceDigitalWrite(ioExpander,7,LOW);
  }
  if (digit==2){
      ioDeviceDigitalWrite(ioExpander,4,LOW);
      ioDeviceDigitalWrite(ioExpander,5,HIGH);
      ioDeviceDigitalWrite(ioExpander,6,LOW);
      ioDeviceDigitalWrite(ioExpander,7,LOW);
  }
  if (digit==3){
      ioDeviceDigitalWrite(ioExpander,4,HIGH);
      ioDeviceDigitalWrite(ioExpander,5,LOW);
      ioDeviceDigitalWrite(ioExpander,6,LOW);
      ioDeviceDigitalWrite(ioExpander,7,LOW);
  }
  seg_nb(digit);
}
```

Figure 34: Fonction MCP23017 - 2

Ce sont ces deux fonction-ci, « seg_nb » permet de fixer l'état des 4 sorties qui codent le nombre à écrire sur le 7 segment en binaire, alors que « seg_write » permet de choisir le digit sur lequel on veut écrire.
On reconnait ici la fonction « digitalWrite » classique, mais appliquée au MCP23017.

3) Module radio NRF24l01

Finalement, un troisième programme va être étudié, c'est celui qui permet la communication du module radio. L'analyse de ce dernier servira pour la compréhension

de ce qui sera abordé dans l'exemple complexe qui suivra.
Le fonctionnement du programme est assez simple, c'est une communication entre 2 modules qui peuvent envoyer et recevoir des messages tous les deux.

```
/************************** NRF24l01 *****************************/
#include <arduino.h>
#include <SPI.h>
#include "RF24.h"

// radio (CE, CS) ==> introduire les pins utilisés
// Sur carte 3AU 2.0      CE 15 // CS 13
RF24 radio(15, 13);

/* Ici sont listées les adresses des modules
Dans cet exemple on travaille avec 2 modules*/
/*Tous les modules du réseau ont cette liste dans leur code*/
uint8_t address[][6] = {"1Node", "2Node"};

/*RadioNumber détermine quelle adresse donner à quel module*/
bool radioNumber = 0;
/*role détermine si le module envoie ou reçoit*/
bool role = false;

/*Variables programme*/
char payload;
char msg[8];

void setup() {
  Serial.begin(9600);
  while (!Serial) {}      //attend l'ouverte du moniteur série

  // initialisation du module sur le bus SPI
  if (!radio.begin()) {
    Serial.println(F("radio hardware is not responding!!"));
    while (1) {} // Si le module ne s'initialise pas reste bloqué
  }

  Serial.println(F("Module initialisé"));

/*****************************************************************/
  Serial.println(F("Introduire le numéro du module"));
  while (!Serial.available()) {
    // wait for user input
  }
  char input_number = Serial.parseInt();
  radioNumber = input_number;
  Serial.print(F("This is Node : "));
  Serial.println((int)radioNumber);


/*****************************************************************/
  Serial.println(F("Introduire le role du module"));
  while (!Serial.available()) {
    // wait for user input
```

```
  }
  char input_role = Serial.parseInt();
  role = input_role;
  Serial.print(F("With role : "));
  Serial.println((int)radioNumber);
/*********************************************************************/
// Set the PA Level low to prevent power supply related issues.
  radio.setPALevel(RF24_PA_LOW);

// Fixe la taille du payload
  radio.setPayloadSize(sizeof(msg));

// set the TX address of the RX node into the TX pipe
  radio.openWritingPipe(address[radioNumber]);

// set the RX address of the TX node into a RX pipe
  radio.openReadingPipe(1, address[!radioNumber]);
/*********************************************************************/
  if (role) {
    radio.stopListening();  // put radio in TX mode
     Serial.println(F("Enter message"));
  }
  else {
    radio.startListening(); // put radio in RX mode
  }
}
void loop() {
  if (role) {
    // This device is a TX node
    if (Serial.available()) {
       // wait for user input
      int i = 0;
      memset(msg,0,sizeof(msg));
      while(Serial.available()){
        char etr = Serial.read();
        msg[i] = etr;
        Serial.println(msg);
        i++;
      }
      unsigned long start_timer = micros();          // start the timer
      bool report = radio.write(&msg, sizeof(msg));  // transmit & save the report
      unsigned long end_timer = micros();            // end the timer
      i++;
      if (report) {
        Serial.print(F("Transmission successful! "));   // payload was delivered
        Serial.print(F("Time to transmit = "));
        Serial.print(end_timer - start_timer);        // print the timer result
        Serial.print(F(" us. Sent: "));
        Serial.println(msg);                          // print payload sent
      }
      else {
        Serial.println(F("Transmission failed")); // payload not delivered
      }
      delay(100);  // slow transmissions down by 1 second
```

```
104     }
105   }
106   else {
107   // This device is a RX node
108     memset(msg,0,sizeof(msg));
109     uint8_t pipe;
110     if(radio.available(&pipe)){
111       //uint8_t bytes = radio.getPayloadSize(); // get the size of the payload
112       //char rcp;
113       radio.read(&msg, sizeof(msg));           // fetch payload from FIFO
114       //msg[i] = rcp;
115       //i++;
116       //Serial.println(rcp);               // print the payload's value
117       //msg[i+1] = 0;
118       Serial.println(msg);
119     }
120   }
121 }
```

Rappelons que le module emploie le SPI pour sa communication avec le microcontrôleur, c'est pourquoi lorsqu'on instancie l'objet « radio » (ligne 7), on lui associe les GPIO nécessaires. C'est là une différence avec le I2C, qui a été le protocole de communication utilisé dans les deux codes présentés précédemment.

Ce dernier code m'a permis de proposer ici un aperçu concret du contenu de ce dossier qui contient tous les codes relatifs à la vérification du fonctionnement de la carte.
On constate que présenter ces derniers peut prendre beaucoup de place sans réellement apporter une plus-value, c'est pourquoi dans l'analyse faite ici je me suis limité à ces 3 programmes mais d'autres programmes présentent un intérêt réel à être analysés, citons : Celui portant sur l'utilisation du CAN, pour tout ce qui est utilisation de sorties analogiques il y a celui sur l'utilisation du DAC, ceux permettant de lire et d'écrire sur la mémoire EEPROM, et enfin celui permettant d'utiliser le connecteur Mikrobus.
Bien sûr ce ne sont là que quelques exemples, le dossier contenant une quinzaine des programmes différents.

6.6 Exemple de projet

Afin de démontrer l'utilité de la carte, il est pertinent de proposer un exemple complet d'un projet tel qu'il pourrait être demandé. L'idée est donc d'exploiter un maximum de fonctionnalités de la carte.
Pour contextualiser et donner un peu de sens à cet exemple posons un scénario :

Imaginons que chaque salle d'un complexe de bureaux possède un module qui permet de contrôler les différents paramètres de la salle : température, luminosité, humidité, pression, … Sur cette carte seront utilisés tous les capteurs.

Ainsi depuis une salle de contrôle on peut monitorer et agir pour modifier le comportement des systèmes installés dans les salles. Sur cette carte seront utilisés tous les modules qui permettent une interaction avec l'opérateur.

Résumons les fonctionnalités :

Master	Slave
- Introduction d'un code sur clavier à matrice pour accéder à la carte - Utilisation des boutons - Utilisation du potentiomètre et encodeur rotatif - Mise en ligne d'une page web locale pour visualiser les données	- Utilisation des différents capteurs - Utilisations de l'afficheur OLED - Utilisation de la LED RGB - Publication des données via MQTT

Enfin, afin de continuer dans l'idée d'exposer un maximum d'aspects différents du monde de l'informatique. Le choix a été fait pour la carte « master » de proposer un code multi fichier pour pouvoir illustrer comment on développe des projets complexes de ce genre.

6.3.1 Structure du code « master »

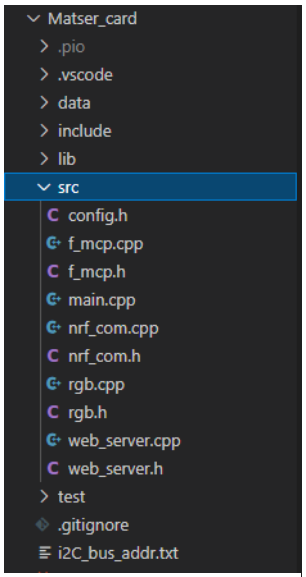


Figure 36 : Workspace master

Ci-contre on voit l'arborescence du programme « master ».
Il y a donc un certain nombre de fichier contenant du code dans ce projet. De la même façon qu'une bibliothèque, il suffit d'importer ces derniers pour pouvoir utiliser différents fichiers au sein du même code.
Avant d'aller plus loin, remarquons déjà la façon dont plateformIO structure ses projets. « src » contient les codes source, « lib » sont les bibliothèques ajoutées depuis un fichier local, alors que c'est dans « .pio » que se trouvent les bibliothèques ajoutées depuis le gestionnaire de bibliothèques.
Enfin « data » contient le code html et css de la page web déployée.

Ce principe de programmation multi fichiers s'appelle « programmation modulaire ».
L'intérêt est d'une part de faciliter la réutilisation de morceaux de codes, et d'autre part de faciliter la lecture de ce dernier. Les fichier « header », tous les fichiers ayant un « .h », sont les fichiers à importer. Ce sont ces derniers qui lient les fichier « .c » entres eux. Ainsi chaque fichier C possède un header, permettant d'importer le programme dans d'autres codes.
Le contenu de ces fichiers header est en général assez court, ils contiennent les prototypes des différentes fonctions. A titre d'exemple voici le contenu du fichier header « nrf_com.h ».

```
1 #ifndef NRF_COM_H
2 #define NRF_COM_H
3
4 void setup_nrf(void);
5 void radio_send_data(char dtype ,float data);
6 float radio_receive_data(void);
7 #endif
```

« mainc.cpp » contient comme toujours le programme principal.
Il serait un peu long de détailler tout le projet ici, cependant vu que la carte vise à être utilisée dans le cadre d'un cours ayant un fort lien avec l'internet des objets. Il semble donc pertinent de regarder comment peut fonctionner la carte avec une application connectée. Voici le contenu du programme « web_server.cpp » :

```
1  #include <ESPAsyncWebServer.h>
2  #include <AsyncTCP.h>
3  #include <SPIFFS.h>
4
5  // Create AsyncWebServer object on port 80
6  AsyncWebServer server(80);
7
8  void setup_spiff(void){
9    //-------------------------------------------------SPIFFS
10   if(!SPIFFS.begin())
11   {
12     Serial.println("Erreur SPIFFS...");
13     return;
14   }
15
16   File root = SPIFFS.open("/");
17   File file = root.openNextFile();
18
19   while(file)
20   {
21     Serial.print("File: ");
22     Serial.println(file.name());
23     file.close();
24     file = root.openNextFile();
25   }
26
27   Serial.println("Successfully Ended");
28 }
29
```

```
30 void setup_server(void){
31    server.on("/", HTTP_GET, [](AsyncWebServerRequest *request)
32    {
33       // Route for root / web page
34       request->send(SPIFFS, "/index.html","text/html");
35       //request->send(200, "text/html",index_html);  --> ligne pour utiliser un css édité
36    });
37    // Start server
38    server.begin();
39 }
```

On définit ici deux fonctions : « setup_spiffs » et « setup_server ».
La première n'a pas un grand rôle, seul la ligne 10 est importante, elle permet d'initialiser le « SPIFFS ». Pour « Serial Peripheral Interface Flash File System » est un système de fichier adapté aux microcontrôleurs ayant une mémoire flash SPI comme l'ESP32 [45]. La bibliothèque « SPIFFS.h » permet d'accéder à cette mémoire. La seconde fonction est celle qui déploie le site, ainsi la ligne 31 indique que faire lorsque la racine du site est chargée, ensuite la ligne 34 et 35 envoient le code html et css vers le navigateur. On utilise le SPIFFS pour parcourir les fichiers enregistrés dans la mémoire flash contenant le code html.
Enfin la ligne 38, démarre le serveur.

Pour se connecter à internet il faut regarde du côté du « main » et du fichier « config.h » qui contiennent toutes les variables de configuration propres au WiFi. La totalité du projet étant disponible sur GitHub et dans le dossier technique, veuillez-vous y référer pour plus de détails. Des bibliothèques dédiées permettent l'exploitation du WiFi, ainsi pour exploiter ces modules connectés la connaissance détaillée des protocoles propres aux WiFi n'est pas nécessaire.
Afin d'illustrer aux mieux ces concepts, il y a également dans la documentation technique un dossier nommé « codes d'appui », qui contient un programme assez complet, qui est une production personnelle, et qui illustre assez bien les attentes du cours de systèmes embarqués. Sa présence vise à contextualiser l'utilisation de la carte dans le cadre du cours. Il permet de mettre en vis-à-vis ce qu'il était possible de faire avant et ce que l'introduction de la Carte 3AU permet à travers l'exemple de projet.

7 Version 2.0

7.1 Changements

Ça a déjà été évoqué, mais plusieurs versions de la carte ont vu le jour, l'idée de la version 2.0 était de corriger les éventuelles erreurs remarquées pendant le testing de la première version. Mais ça va un peu plus loin, puisque ce qui a été fait ici, c'est d'ajouter des modules là où c'était encore possible, ainsi entre les deux versions l'objectif était de vraiment réussir à exploiter au mieux ce que nous permet l'ESP32.
Concrètement voici les différents changements :

- Des changements d'assignation de pin, pour exploiter correctement les spécificités des différents pins.
- Suppression de la RGB 4 pins par une RGB WS2812.
- Ajout d'un lecteur micro SD
- Ajout d'un capteur BMP280
- Ajout d'une mémoire EEPROM
- Réassignation des entrées pour les boutons.
- Modification de la disposition des composants sur la carte
- Ajout d'un DAC
- Changements sur le fonctionnement de l'alimentation.

Concernant l'alimentation il faut savoir que sur la première itération, la carte utilisait deux hacheurs, mais en pratique cela présentait peu d'intérêt car le passage du 5V au 3V3 ne justifiait pas vraiment ce genre de montage. L'idée pour permettre la double connexion sur la première version était de placer une diode sur le chemin de l'ESP mais il n'y avait pas de production de 3V3 sans une connexion JACK. C'était là un défaut majeur puisqu'il fallait obligatoirement travailler avec la carte branchée. Sur la nouvelle version grâce à l'ajout du LM256, on génère le 3V3 à partir du 5V USB. Ainsi même lors des phases de test, il suffit de brancher la carte via son port USB pour disposer de la totalité des fonctionnalités de cette dernière.

L'ajout de nouveaux modules a été rendu possible par le remplacement de la LED RGB, puisque précédemment celle-ci utilisait 3 entrées de l'ESP. La nouvelle version n'en utilisant qu'une seule, les deux autres entrées ont servi pour le BMP280 et le lecteur de carte SD.

Ensuite soulignons que l'ajout de composant sur le bus I2C ne nécessite aucun changement, il faut cependant veiller à ne pas surcharger le bus sous peine de dégrader la communication. Ce principe de maintien du niveau de la ligne a été abordé précédemment.

Enfin, il y a eu un certain nombre de réassignations des différents pins, en effet lors des phases de test de la première version, certains défauts qui n'avaient pas été envisagés sont apparus, ainsi l'étape de révision est une étape cruciale dans le développement de tout système électronique. C'est avec ce prototype en main que nous avons pu évaluer non seulement le fonctionnement des composants mais également la disposition et l'espace disponible pour d'éventuels autres ajouts.

8 Conclusion

La version finale de cette carte offre la possibilité aux étudiants d'utiliser un certain nombre de composants, qui se veulent suffisamment divers pour donner un aperçu réaliste des possibilités qui existent dans le monde professionnel. Les différents schémas produits et codes rédigés doivent permettre de faciliter la prise en main du cours pour ainsi offrir la possibilité d'approfondir les concepts étudiés.
Terminons par évoquer l'importance que présente ce genre de projet, l'informatique occupant une place toujours plus importante dans le monde industriel, le développement d'outil dédiés à l'apprentissage est quelque chose de primordial. Et même si des sociétés comme « Mikroelectronika » le font de manière professionnelle, il ne faut pas se limiter à la simple utilisation d'outils payants mis en avant par de grandes marques.
La montée de l'informatique est accompagnée d'une complexification du milieu, ainsi certes ces nouveaux outils révolutionnent le monde professionnel mais ils rendent le milieu toujours plus difficile à apprivoiser.
L'enjeu ici est donc de garantir que l'apprentissage des domaines techniques reste accessible. Cet effort de vulgarisation passe par la production de documents et d'outils didactique complets et j'espère que c'est ce que sera cette carte pour de futurs étudiants.
De plus, le fait que ce soit réalisé par un étudiant en stage et non par une grande marque laisse entrevoir aux étudiants ce que le cours peut leur apporter et concrétise le contenu de ce dernier. J'espère donc que ce sera une source de motivation supplémentaire pour eux.
Enfin je vais conclure en précisant que le seul outil développé lors de ce stage n'a pas été la carte en elle-même, mais aussi la mise en place d'un compte GitHub qui contient toutes les informations relatives au projet ainsi on peut rendre ce projet disponible pour le plus grand nombre, et bien que mon stage soit terminé, j'ai des idées que j'aimerais mettre en place dans le futur pour améliorer encore la carte et les outils disponibles autour celle-ci, j'ai déjà évoqué la possibilité d'intégrer d'autres modules d'extension d'entrées sorties mais d'autres idées, comme la création de clicboards personnalisées suivant le standard du connecteur Mikrobus existent. Enfin une idée que je n'ai pas pu mener à terme est de proposer des codes simples à implémenter sur des Raspberry Pi pour mettre en place un réseau local destiné à la réalisation des différents TP.

Table des illustrations

Bibliographie

[1] Espressif, About-Espressif, en ligne.
https://www.espressif.com/en/company/about-espressif
Consulté le 16/04/2021
[2] WaytolearnX, Différence entre microprocesseur et microcontrôleur, 2018, en ligne
https://waytolearnx.com/2018/11/difference-entre-microprocesseur-et-microcontroleur.html
Consulté le 16/04/2021
[3] Wikipédia, CPU, en ligne
https://en.wikipedia.org/wiki/Central_processing_unit
Consulté le 18/04/2021
[4] Leduc, T., Systèmes d'exploitation, Ecole Nationale d'Architecture de Nantes, 2006, http://thomas.leduc.free.fr/SupportsFormations/os-cci-2006.pdf
Consulté le 18/04/2021
[5] First-TF, Oscillateur à quartz, en ligne
https://first-tf.fr/recherche-valorisation/science-technologie/oscillateurs/oscillateurs-a-quartz/
Consulté le 20/02/2021
[6] Bonaventure, O., Detal, G., Paasch, C., Le langage C, 2013, en ligne
https://sites.uclouvain.be/SyllabusC/notes/Theorie/C/intro-C.html#:~:text=Le%20langage%20C%20permet%20bien%20entendu%20la%20d%C3%A9finition,qu%27une%20fonction%20de%20type%20int%20retournera%20un%20entier.
Consulté le 27/02/2021
[7]Berthomier, E., Schang, D., Le C en 20 heures, 2017, 202p
[8]Plaisir Arduino, Les fonctions loop et setup, 2015, en ligne
https://plaisirarduino.fr/les-fonctions-loop-et-setup-arduino/
Consulté le 18/04/2021
[9] Naili, A., Cours d'électronique A204 ISAT, 2019-2020
[10] Diode Incorporated, AP3211, 2018, en ligne
https://www.diodes.com/assets/Datasheets/AP3211.pdf
Consulté le 02/03/2021
[11] Cours-exams.org, Régulateur linéaire de tension
https://www.cours-examens.org/images/Etudes_superieures/Ingeniorat_electricite/Mecatronique/Vaud/Systeme_Elecronique_I/Chapitre_3_Les_regulateurs_lineaires.pdf
Consulté le 30/03/2021
[12] Wikipedia, la diode Schottky, en ligne
https://fr.wikipedia.org/wiki/Diode_Schottky
Consulté le 15/05/2021
[13] Julien, Théories de lissage de courant par bobine (inductance), 2016, en ligne
http://electronique71.com/theories-lissage-de-courant-par-bobine-inductance/
Consulté le 15/03/2021
[14] SEMI, Le bus SPI, 2020, en ligne
https://semi.fpms.ac.be/simius/interfacage/spi.html
Consulté le 20/05/2021
[15] Costa, E., Robotique partie I A2071, ISAT 2019-2020

[16] Nordic Semiconductor, nRF24L01, 2007 en ligne
https://cdn.sparkfun.com/datasheets/Wireless/Nordic/nRF24L01_Product_Specification_v2_0.pdf
Consulté le 20/03/2021
[17] F4HOK, L'utilité des condensateurs de découplage, 2017, en ligne
https://f4hok.wordpress.com/2017/10/06/lutilite-des-condensateurs-de-decouplage/
Consulté le 25/03/2021
[18] https://boowiki.info/art/memoire-flash/memoire-flash-2.html
[19] Couleur science, Comment fonctionne un thermocouple ? d'où vient l'effet Seebeck ?, 2020, en ligne
https://couleur-science.eu/?d=154f2a--comment-fonctionne-un-thermocouple-dou-vient-leffet-seebeck
Consulté le 22/05/2021
[20] REOTEMP instruments, Type K thermocouple, en ligne
https://www.thermocoupleinfo.com/type-k-thermocouple.htm
Consulté le 15/05/2021
[21] Wikipedia, Thermocouple, en ligne
https://fr.wikipedia.org/wiki/Thermocouple
Consulté le 15/05/2021
[22]Wiki Monde, I2C, en ligne
http://wikimonde.com/article/I2C
Consulté le 19/05/2021
[23] CoursTechInfo, Les mémoires, en ligne
https://courstechinfo.be/Hard/Memoire.html
Consulté le 10/05/2021
[24]Futura Tech, OLED : qu'est ce que c'est ?, en ligne
https://www.futura-sciences.com/tech/definitions/technologie-oled-2900/
Consulté le 1/03/2021
[25] Texas Instrument, I2C Bus Pullup Resistor Calculation, 2015, en ligne
https://www.ti.com/lit/an/slva689/slva689.pdf
Consulté le 15/03/2021
[26]Espressif, ESP32 Series, 2021, en ligne
https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
Consulté le 1/03/2021
[27]Brah, F., Cours Architecture des systèmes et systèmes numériques A214 ISAT 2019-2020
[28] Espressif, Touch Sensor Application Note, 2019, en ligne
https://github.com/espressif/esp-iot-solution/blob/release/v1.0/documents/touch_pad_solution/touch_sensor_design_en.md
Consulté le 10/03/2021
[29] Altium, Comment utiliser et interfacer des LED WS2812B, 2017, en ligne
https://cutt.ly/0nlL8c8
Consulté le 08/04/2021
[30] Naili, A., Mesures et instrumentations A2131, ISAT 2019-2020

[31]Espressif, ESP-IDF Programming Guide, 2021, en ligne
https://docs.espressif.com/projects/esp-idf/en/latest/esp32/index.html
Consulté le 20/05/2021
[32]WarElectronics, What is a Light Dependent Resistor and Its Applications,2019, en ligne
https://www.watelectronics.com/light-dependent-resistor-ldr-with-applications/
Consulté le 14/05/2021
[33] Wikipedia, Bus de données CAN, en ligne
https://fr.wikipedia.org/wiki/Bus_de_donn%C3%A9es_CAN
Consulté le 22/03/2021
[34]Techno science, Adaptation d'impédances , en ligne
https://www.techno-science.net/glossaire-definition/Adaptation-d-impedances-page-3.html
Consulté le 29/05/2021
[35] Mikroelektronika, THE WORLD'S FASTEST GROWING ADD-ON BOARD STANDARD, en ligne
https://www.mikroe.com/mikrobus
Consulté le 13/05/2021
[36]PCBWAY, Fabrication de PCB - Fabrication de PCB étape par étape, en ligne
https://www.pcbway.fr/pcb-service.html
Consulté le 30/03/2021
[37]Altium, Comment définir des plans de masse dans un circuit imprimé à deux couches, 2018 – 2020, en ligne
https://resources.altium.com/fr/p/understanding-ground-planes-your-two-layer-pcb
Consulté le 28/03/2021
[38] Muthukrishnan, V., Electromagnetic Interference (EMI): What it is & How To Reduce it, 2020, en ligne
https://www.electrical4u.com/electromagnetic-interference/
30/03/2021
[39]GRABCAD, en ligne
https://grabcad.com/
[40] GitHub, Codes carte 3AU, en ligne
https://github.com/tfe-juan-alvarez/3-Codes-de-test-Carte-3AU
[41] git, About, en ligne
https://git-scm.com/
https://git-scm.com/about
[42]VScode, getting started, en ligne
https://code.visualstudio.com/docs
[43]PlatformIO, Professional collaborative platform for embedded development, en ligne
https://platformio.org/
[44]Costa, E., Systèmes embarqués II A304, ISAT 2020-2021
[45]Domotique et objets conectés, ESP32. Débuter avec la librairie SPIFFS.h pour lire, écrire, modifier des fichiers, 2020 , en ligne
https://cutt.ly/CnlL2Ja
Consulté le 1/06/2021

Figures

[1] Xukyo, Utilisation d'un module NRF24L01 avec Arduino, 2020, en ligne
https://www.aranacorp.com/wp-content/uploads/module-nrf24l01.jpg
consulté le 02/06/2021
[2] Electan, led smd rgb ws2812, en ligne
https://www.electan.com/led-smd-rgb-ws2812-p-4105-en.html
Consulté le 29/05/2021
[3]PEC11R, en ligne
https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcS-1ynEnqJ7uRuzj9QRkhFl31l4h5EJyAJNy1rb8J6wii6ezjNOWJNWoxmlmiDNfLGc4Ew&usqp=CAU
Consulté 01/06/2021
[4] Robot delta, fonctionnement de l'encodeur rotatif, en ligne
https://cutt.ly/AnlLBje
Consulté le 31/05/2021
[5] Electronique.AOP.free.fr, circuit suiveur, en ligne
http://electronique.aop.free.fr/AOP_lineaire_NF/1_suiveur.html
Consulté le 30/04/2021
[6] Sciences de l'ingénieur, afficheur 7 segments, 2020, en ligne
https://si.blaisepascal.fr/wp-content/uploads/2020/03/drawit-diagram-30.png
Consulté le 22/05/2021
[7] Hamouchi, H., Figure 31, en ligne
https://cutt.ly/inlZHvB
consulté le 22/05/2021
[8]Image DHT11, en ligne
https://sc04.alicdn.com/kf/HTB1QedxdAxz61VjSZFtq6yDSVXaJ.jpg
[9]Image DHT11, en ligne
https://www.gmelectronic.com/data/product/1024_1024/pctdetail.774-015.1.jpg

Résumé

Dans le monde d'aujourd'hui, l'informatique occupe une place toujours plus importante, en ce sens que la recherche de l'optimisation permanente passe par l'utilisation de nouveaux outils et aujourd'hui, l'informatique apporte une solution à ces problèmes.
Notre formation comporte plusieurs facettes, en effet l'automatisation est présente aujourd'hui dans la plupart des industries sous différentes formes.
Il y a d'une part l'automatisation à base de PLCs où des entreprises comme Siemens se sont fait un nom, ces mêmes automates sont utilisés pour faire de la régulation, c'est là un deuxième axe de notre formation et un domaine dans lequel à travers ma recherche de stage j'ai trouvé beaucoup d'offres.
D'autre part, il y a tout ce qui touche à l'informatique, l'électronique et à la programmation, de nos jours leur utilisation à des fin industrielles est plus que répandue. C'est dans cette optique que le cours de systèmes embarqués I et II est dispensé, avec pour but de nous fournir les bases de l'automatisation informatique.
En parallèle de ces compétences informatiques, nous sommes aussi formés à maitriser les outils nécessaires à la réalisation de tâches plus techniques toujours étroitement liées au métier d'automaticien.
Et c'est dans ce cadre que s'inscrit mon TFE, la programmation est la partie automatisée et le passage par la conception mobilise toutes les compétences techniques.

Le présent document propose dans un premier temps, un aperçu des différentes étapes de conception et le détail des schéma réalisés. Pour, enchainer dans un second temps, par un aperçu de son fonctionnement et de son utilisation.
L'objectif premier du projet est de produire un outil de travail adéquat pour les futurs étudiants. Mais à travers ce travail, l'idée est aussi de permettre aux étudiants et à d'autres personnes de pouvoir accéder à des plans et documents techniques, pour qu'ils puissent s'inspirer du travail réalisé et produire leur propre carte.
Enfin, un objectif secondaire poursuivi, est d'utiliser un maximum d'outils collaboratifs pour qu'à travers l'utilisation de la carte, on puisse se familiariser avec ces outils qui sont aujourd'hui omniprésent dans le monde professionnel.