MYVACUUM, L'ASPIRATEUR INTELLIGENT

Conception Mécanique, Électronique, Traitement De Signal, Programmation, …

Par DI VENTI Davide
60456@etu.he2b.be

RÉSUMÉ

Le projet en question consiste en un aspirateur doté de fonctionnalités numériques supplémentaires. Comme tout aspirateur domestique, il a pour fonction principale d'aspirer les miettes, la poussière, le sable, etc. Toutefois, contrairement à ses homologues, myVacuum est capable de fournir à son utilisateur un état de santé en temps réel grâce à un écran. Cet état inclut des informations telles que la température interne de l'appareil, le niveau de remplissage du réservoir, la qualité de l'air aspiré, et bien d'autres encore.

L'utilisateur peut contrôler l'aspirateur à partir d'une interface utilisateur. Cette dernière prend la forme d'une télécommande filaire équipée d'un petit écran et de 4 boutons poussoirs. L'interface comprend plusieurs menus préprogrammés, notamment une zone permettant de visualiser et de commander l'état de l'appareil, ainsi qu'une autre zone permettant d'acquérir des données et de créer un système de logs[1] à partir d'une carte SD.

Tout comme les aspirateurs domestiques classiques, myVacuum est équipé d'une fonction permettant de régler la puissance d'aspiration. Toutefois, contrairement aux autres modèles, cette fonction est entièrement numérique, ce qui la rend plus précise. De plus, il dispose d'un tachymètre intégré qui fournit un retour sur la vitesse du ventilateur.

1. INTRODUCTION

Ce rapport présente la conception et la réalisation d'un aspirateur digital, en mettant l'accent sur les différentes parties mécaniques, électroniques et de programmation du système. La partie mécanique est abordée en premier lieu, en décrivant la conception de pièces spécifiques telles que des adaptateurs en plastique et en métal. Ensuite, la partie électronique est examinée, mettant en lumière les choix de composants ainsi que le traitement du signal du capteur, un point central du projet. Le traitement du signal est effectué en utilisant des principes tels que le filtrage, l'amplification et le registre à décalage. Enfin, la partie programmation est présentée en détail, expliquant le fonctionnement final du code et incluant un lien vers le dépôt GitHub où celui-ci est disponible. Des schémas électroniques et mécaniques détaillés du système sont également fournis dans les annexes du rapport.

L'objectif premier de ce rapport est de présenter un produit sortant d'un bureau d'études, qui intègre des systèmes embarqués de manière technique.

2. DESCRIPTION DU PROJET

Le projet est subdivisé en quatre grandes parties distinctes. Chacune de ces parties a nécessité des prérequis spécifiques ainsi qu'une durée de conception relativement similaire à celle des autres parties.

2.1. Mécanique

La mécanique est un aspect essentiel de ce projet. Chaque élément a été préalablement conçu en 3D, allant des différents adaptateurs jusqu'aux joints d'étanchéité. Les adaptateurs, qui sont imprimés en 3D en PLA, ont été créés pour servir de supports pour les capteurs, pour les systèmes d'emboîtement (entre filtre, ventilateur, réservoir, tube d'aspiration, etc.), pour les connecteurs électroniques avec détrompeurs, etc. Le filtre, pour sa part, est également imprimé en 3D. Sa structure est conçue pour accueillir un tissu carbonisé servant de filtre à air. De même, l'interface utilisateur possède un socle imprimé en 3D. Les joints d'étanchéité sont également imprimés, mais cette fois-ci en TPU, un caoutchouc adapté pour l'extrusion des imprimantes 3D.

En outre, j'ai utilisé une CNC pour découper une pièce : il s'agit d'une plaque en acrylique positionnée à la base du filtre. Cette plaque permettra à un capteur de pouvoir émettre une lumière et recevoir sa réflexion tout en étant protégé des déchets dans le réservoir (capteur de niveau de remplissage du réservoir). Le réservoir de l'aspirateur est également fabriqué en acrylique. Il s'agit d'un tube cylindrique de 100 mm de diamètre.

Enfin, une fois toutes ces pièces assemblées, j'ai remarqué que l'aspirateur avait tendance à tomber latéralement à plusieurs reprises. J'ai donc été contraint de trouver une solution, et j'ai décidé de concevoir un socle en aluminium à l'aide d'un tour à métaux. Cette pièce s'emboîte dans le bas du réservoir, permettant ainsi de fermer le réservoir et de descendre le centre de gravité de l'aspirateur. Cette solution a été la plus adaptée et satisfaisante pour moi, étant passionné depuis mon enfance par l'usinage de pièces industrielles.

[1]Logs : Sous forme d'un tableau, il s'agit d'une suite séquentielles de valeurs acquises par des capteurs ou autre en fonction du temps. Cela peut se référer à un historique. Chaque données est stockée tous les laps de temps.

En prenant en compte le temps de conception de chaque pièce mécanique (temps d'impression 3D non inclus), cela représente plus de 22 heures de travail.

2.2. Électronique

Dans ce projet, l'électronique utilise une architecture basée sur une unité de contrôle. Cette carte électronique centralise les données provenant de différents périphériques tels que les capteurs, le ventilateur et l'écran, pour les traiter, les afficher, les stocker, etc.

Un microcontrôleur ESP32 a été choisi pour cette unité de contrôle. Toutes les broches du microcontrôleur ont été utilisées afin de tirer pleinement parti de ses capacités. L'annexe 7 présente les périphériques connectés (ou transistorisés) à ce microcontrôleur, je vous invite à la prendre.

Chaque périphérique est justifié et joue un rôle crucial dans le cadre de ce projet. Le buzzer permet d'alerter l'opérateur en cas de détection d'un code d'erreur. Par exemple, si l'aspirateur aspire de la matière brûlée (détectée par le DHT11 et le capteur de gaz MQ-2), de la matière humide (par le DHT11), ou un gaz nocif pour la santé humaine (par le capteur de gaz MQ-2), le buzzer émet une alerte. D'autres situations peuvent également survenir, telles que le remplissage du réservoir (capteur de distance VL6180X), la nécessité de nettoyer le filtre (vitesse du ventilateur trop faible), ou encore la saturation de la carte SD.

Le relais fonctionne selon le même principe que celui intégré dans les aspirateurs industriels, à savoir une prise 230VAC destinée à connecter une machine-outil telle qu'une scie sauteuse, une foreuse ou une défonceuse. En d'autres termes, si le périphérique 230VAC est activé, l'aspirateur s'allume. Dans le cas de myVacuum, ce principe est repris, mais sous une forme miniaturisée avec un relais.

Le ventilateur utilisé (bg0903-b049-p0s) provient d'un PC Dell fixe où il occupait la place du ventirad. L'ensemble du projet est dimensionné en fonction de ce ventilateur, qui est un ventilateur 12VDC comportant 4 broches dont deux pour l'alimentation 12V, une pour le tachymètre, et une pour la commande PWM. Le « Fan MOSFET » affiché dans l'annexe 7 du ventilateur, est un simple transistor MOSFET que j'ai ajouté permettant de couper ou de fournir l'alimentation 12V externe au ventilateur. En effet, même en forçant un état logique bas (0V) à la commande PWM du ventilateur, sa vitesse (tr/min) ne descend pas à zéro, mais atteint plutôt un seuil minimum de XX tr/min.

En prenant en compte le temps nécessaire à la réflexion (choix des composants) et à la conception électronique (schémas électroniques et conception du PCB), le temps total de travail estimé s'élève à plus de 16 heures.

2.3. Traitement de signal

Le traitement du signal réalisé dans ce projet est la partie la plus électronique de la machine concernée. Il est effectué à la sortie du tachymètre du ventilateur, de manière à ce que l'ESP32 puisse lire la vitesse sans qu'il y ait de nombreuses erreurs ni de latences. Les étapes suivantes décrivent le dimensionnement des étages de traitement du signal tachymètre.

Avant toute chose, il convient de noter que ce ventilateur envoie 2 signaux logiques par tour.

Le premier étage de traitement du signal vise à transmettre ces signaux avec une logique de 0V à 3,3V (ce qui n'est pas le cas par défaut). Étant donné que le capteur intégré est à effet Hall, l'intensité du signal tachymètre varie en fonction de la vitesse. Plus la vitesse est élevée, plus l'intensité du signal augmente en tension (de 0 à 1,8V), ce qui peut perturber la lecture de la vitesse. Il est donc nécessaire de procéder à un étage d'amplification. Toutefois, avant cela, il est préférable de filtrer les parasites potentiels (pics de tension à très haute fréquence) susceptibles de "brouiller" l'amplification. Pour ce faire, un simple filtre passe-bas RC est suffisant. 'ai dimensionné le filtre de telle sorte que la fréquence de coupure soit d'environ 5 kHz, qui est la fréquence maximale du tachymètre du ventilateur. Ensuite, par expérimentation, j'ai essayé différentes valeurs de R et C pour obtenir une enveloppe de signal filtré adaptée à mes besoins, finalement en utilisant C = 50 nF et R = 2 kΩ, j'ai obtenu un filtrage opérationnel.

Une fois le filtrage terminé, j'ai procédé à l'amplification en utilisant un amplificateur opérationnel (AOP) que j'ai utilisé en tant que comparateur, tel un trigger de Schmitz. Étant donné que le signal filtré possédait un offset de 400 mV, il était important d'éliminer cet offset avant l'amplification. Pour ce faire, j'ai utilisé une tension de référence provenant d'un pont diviseur de tension pour éliminer l'offset par le biais du V- de l'AOP, évitant ainsi que l'offset ne s'amplifie. La tension de référence a été fixée à 800 mV pour garantir que l'offset ne serait pas pris en compte à la sortie du comparateur. Le signal filtré du tachymètre (V+) et la tension de référence (V-) ont été comparés, et seule la tension supérieure à 800 mV a été copiée en sortie. Le signal de sortie du comparateur a atteint un niveau logique équivalent à l'alimentation de l'AOP (0V-3.3V).

Cependant, une problématique persiste en dépit de l'aspect impeccable du signal, carré et net. En effet, la vitesse du signal demeure trop rapide pour être traitée de manière satisfaisante par l'ESP32. Bien que le microcontrôleur puisse traiter et convertir la vitesse du signal en tr/min, un facteur important doit être pris en compte : la latence du programme. À mesure que le programme gère davantage de périphériques, des latences se produisent, entraînant un délai de quelques millisecondes à la fin de la boucle. Une latence de quelques millisecondes (1ms à 20ms) est considérable,

surtout si le microcontrôleur doit lire un signal à haute fréquence. L'oscilloscope a mesuré une fréquence de 1250Hz pour le signal filtré et amplifié, ce qui correspond à une période de signal de 800µs. Cela équivaut à une vitesse de rotation du ventilateur de 5500 tr/min.

Un deuxième étage de traitement de signal s'est donc imposé, celui permettant de diminuer la fréquence du signal, tout en restant beau, carré et propre. Pour ce faire, il n'y a rien de plus efficace que de l'électronique ancienne : le registre à décalage. J'ai disposé d'un registre à 8 bits, ce qui m'a permis de diviser la fréquence du signal par 8, résultant en une période de 6,4 ms au lieu de 800 µs. Cette solution est largement suffisante pour mon application. Afin d'assurer le bon fonctionnement de cette idée subite, j'ai simulé ce principe et procédé à plusieurs essais et erreurs. Heureusement, le résultat a été couronné de succès.

En outre, compte tenu du temps nécessaire à l'étape de dimensionnement et aux tests requis pour ces traitements de signaux, le temps total de travail s'élève à plus de 11 heures.

2.4. Programmation

Dans le but d'améliorer la compréhension et la simplicité du code, j'ai procédé à l'importation de la bibliothèque Arduino. Cette démarche m'évite de concevoir mes propres méthodes de classe pour effectuer des opérations élémentaires, telles que l'attribution d'une broche en tant que sortie ou entrée, l'affichage de données dans la console et la gestion des interruptions, etc.

2.4.1. Affichage dans l'écran TFT

En complément de cette bibliothèque, le code est doté d'une classe que j'ai élaborée. Cette dernière me permet de créer des icônes et des animations personnalisées sur l'écran TFT. L'objectif est de pouvoir générer plusieurs objets (widgets : icône de ventilateur, icône de thermomètre, ...) et les afficher avec la latence la plus minimale possible.

Afin d'atteindre cet objectif, je n'enregistre pas les images au format PNG, JPG ou autre sur la carte SD, mais plutôt dans un format personnalisé et léger : une chaîne de caractères. À partir de cette chaîne, ma classe décode les données et affiche l'image correspondante pixel par pixel en couleur.

La déclaration de la chaîne de caractères est précédée par les dimensions (en pixels) de l'image à afficher :

```
int myIconeWidth, myIconeHeight = 16, 16;
String myIcone = "7C00820101014901490FF96
805DFE41407D3F8510851079080206EC02A80444";
```

Ensuite, ces trois variables sont insérées pour la création de l'objet de ma classe. La chaîne hexadécimale est ensuite convertie en binaire, ce qui entraîne une augmentation de sa longueur quatre fois supérieure. Ligne par ligne et colonne par colonne, la chaîne binaire est parcourue, et la classe affiche des pixels aux coordonnées correspondantes là où des "1" sont présents, tandis qu'elle n'affiche rien là où des "0" sont présents (comme illustré ci-dessous dans la figure 1) :

```
0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 1 0 1 0 0 1 0 0 1 0 0 0 0
0 0 0 1 0 1 0 0 1 0 0 1 0 0 0 0
1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 0
1 0 0 0 0 0 0 0 0 1 0 1 1 1 0 1
1 1 1 1 1 1 1 1 0 0 1 0 0 0 0 0 1
0 1 0 0 0 0 0 0 0 0 1 1 1 1 1 0 1
0 0 1 1 1 1 1 1 1 1 0 0 0 0 1 0 1
0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 1
0 0 0 1 0 0 0 0 0 0 1 1 1 1 0 0 1
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0
0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 0 0
0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0
```

Figure 1. Processus d'affichage dans l'écran TFT

Cependant, il n'est pas possible pour un être humain d'écrire intuitivement cette chaîne de caractères. Il est donc nécessaire d'effectuer l'inverse du processus. À cet effet, j'ai développé un logiciel en JavaScript qui me permet de générer la chaîne de caractères hexadécimale/binaire à partir d'un espace de travail de dessin en PixelArt. Les détails de fonctionnement de ce logiciel ne seront pas abordés dans ce rapport, car cela s'éloignerait du sujet principal. J'ai hébergé ce logiciel

sur GitHub, et il est directement accessible via le lien suivant :

https://davidediventi.github.io/PixelToHexConverter/

Le plus grand défi de ce projet a été de surmonter les latences du microcontrôleur, et cette classe combinée à ce logiciel ont joué un rôle clé pour résoudre ce problème.

2.4.2. Acquisitions de données des capteurs

En parallèle de l'affichage, il y a également l'acquisition de données. Quatre des cinq données à acquérir sont actualisées à intervalles réguliers de 300 ms. Ces données comprennent la température (°C), l'humidité (%), la qualité de l'air (%) et le niveau de remplissage du réservoir (mm). Quant à la cinquième donnée, qui correspond à la vitesse du ventilateur, elle est acquise d'une manière différente.

En effet, sur une période de 1800 ms, l'ESP32 compte le nombre de tours effectués par le ventilateur à l'aide d'interruptions externes. Ensuite, une conversion est effectuée pour obtenir les tours par minute (rpm).

Ces données sont affichées dans le menu DASHBOARD, précédées des icônes correspondantes telles que le thermomètre, la goutte d'eau, le compteur, etc. Elles sont également affichées dans le menu LOGS de l'écran, mais cette fois-ci sous forme de graphique en temps réel.

Les graphiques présentés dans le menu LOGS conservent un historique de données sur une période de 30 secondes, correspondant à une largeur de courbe de 100 pixels. En d'autres termes, toutes les 300 ms, une acquisition est effectuée, et un pixel est affiché à une échelle commune pour tous les capteurs. Donnée par donnée, pixel par pixel, une courbe prend forme. Toutes les courbes de tous les capteurs sont regroupées dans un même graphique en temps réel.

2.4.3. Carte SD

Toutes les 3 secondes, il y a une écriture sur la carte SD. Les données sont enregistrées sous forme d'un fichier CSV contenant cinq colonnes correspondant aux cinq données pouvant être acquises : la température (°C), l'humidité (%), la qualité de l'air (%), le niveau de remplissage du réservoir (mm) et la vitesse de rotation du ventilateur (rpm).

À chaque démarrage de l'aspirateur, un nouveau fichier CSV est créé. Cela permet ensuite de lire les données sur un ordinateur à l'aide d'un logiciel tel qu'Excel. En créant des graphiques à partir de ces valeurs, il devient possible de diagnostiquer d'éventuels défauts de la machine.

Bref, en prenant en compte le temps consacré au développement du code, le travail total s'élève à plus de 60 heures (le logiciel pixelart n'étant pas inclus).

3. MATÉRIEL & MÉTHODES

Ce projet n'aurait pu être mené à bien sans l'accès au matériel adéquat et aux prérequis nécessaires. Plusieurs outils sont mis à disposition des étudiants et j'en ai profité.

3.1. Outils utilisés

Parmi les nombreux outils indispensables pour mener à bien ce projet, je possède certaines machines à mon domicile, ce qui m'a grandement facilité la tâche pour poursuivre le travail à distance.

3.1.1. Outils mécaniques

Le Fablab de l'école met à disposition des étudiants des imprimantes 3D Prusa. Pour imprimer en 3D, il est nécessaire de générer les fichiers STL de la pièce en 3D depuis un logiciel de modélisation 3D, puis de les donner à un préparateur afin qu'il génère le GCode à partir du slicer approprié. J'ai personnellement utilisé Fusion 360 comme logiciel de modélisation 3D. Par ailleurs, des machines à commande numérique (CNC) sont également disponibles, notamment la machine Mekanika qui est particulièrement appréciée. En ce qui concerne l'usinage industriel, j'ai eu recours à un tour à métaux Celtic 14 mis à disposition par l'école, qui est un modèle de grande qualité.

3.1.2. Outils électroniques

Pour visualiser les signaux électriques, un oscilloscope est indispensable. J'ai utilisé le modèle Tektronix TDS 2014B tout au long de l'étude de dimensionnement des étages de traitement de signal de mon projet à l'école.

Une fois que j'ai validé l'électronique sur une platine d'expérimentation, j'ai conçu le PCB à l'aide du logiciel Fusion360. Cette étape m'a permis de générer les fichiers Gerber, nécessaires pour fabriquer le PCB chez JLCPCB, un fabricant de circuits électroniques.

3.1.3. Outils informatiques

Pour la programmation de l'ESP32, j'utilise l'IDE Arduino. Cette interface de développement logiciel propose une arborescence simple et efficace, adaptée à mes besoins. Pour stocker les données collectées, l'aspirateur édite des fichiers CSV sur une carte SD. Cette méthode facilite la visualisation et l'analyse des données sur un ordinateur, grâce notamment à la création de graphiques. Cet outil se révèle particulièrement utile lorsqu'il s'agit de diagnostiquer

d'éventuels dysfonctionnements rencontrés par l'aspirateur lors de sa mise en service.

3.2. Méthodes d'avancement de l'étude

Il n'existe pas de méthode ou de technique universelle pour la conception de projets. Il est nécessaire de se connaître soi-même et de déterminer la durée estimée de chaque tâche avant de s'engager dans une étape dont la fin ne nous est pas clairement définie.

En tant que maker né, j'ai acquis une expérience qui me permet de mener à bien chacun de mes projets dans les délais impartis. Cependant, chaque projet est unique et nécessite une méthode d'avancement différente en fonction de ses spécificités. La méthode à utiliser doit s'adapter au contenu du projet.

La réussite de cette méthode est conditionnée par la nécessité de connaître au préalable le rendu final du premier prototype. Pour atteindre ce stade, il est impératif que le projet nous motive et nous fasse rêver. Dans mon cas, la motivation est atteinte lorsqu'il y a une intégration de la mécanique. Mais ma condition première est que le projet doit avoir une utilité en dehors du monde académique (s'il s'agit d'un projet scolaire).

L'idée de concevoir un aspirateur m'est lointaine. Mais, je n'avais pas encore discerné l'utilité d'un aspirateur personnel. En effet, mes projets personnels se multiplient au fil du temps. À présent, j'en suis arrivé au stade où je conçois une CNC semi-professionnelle pour un usage domestique. Le but de cette machine est de découper de l'aluminium pour des projets à venir. Et qui dit CNC, dit aspirateur. Le tube d'aspiration sera relié à la broche du routeur de la machine pour aspirer les copeaux. Lorsque la fraise se met à tourner, l'aspirateur se déclenche. Pour ce faire, la CNC communiquera en UART avec l'aspirateur (des headers mâles sont prévus dans le PCB : UART2). Telle est donc l'idée qui m'a inspiré pour mener à bien ce projet.

Pour ce projet, j'ai estimé qu'il serait plus judicieux de travailler en parallèle sur toutes les parties du projet (mécanique, électronique et programmation) tout en gardant à l'esprit les objectifs finaux. Telle a été ma méthode.

J'ai donc commencé par dessiner un croquis sur papier, qui incluait l'idée finale du projet selon mes réflexions. J'ai transmis ce croquis ainsi que le cahier des charges aux professeurs en charge du projet concerné, trois mois avant le début du premier cours. Une fois qu'ils ont validé l'idée du projet, je me suis lancé directement dans sa conception.

3.3. Étapes de la réalisation du projet

Le premier cours du projet a débuté le 1er février 2023. Quant au dernier, il a eu lieu le 22 mai 2023. Dans la table 1 se trouve l'avancement du projet en fonction des semaines académiques, en commençant par la semaine -1, qui est celle précédée du premier cours :

Semaine	Avancement
-1	Commande PWM du ventilateur avec un potentiomètre. Affichage de la vitesse sur l'écran TFT.
1	Assemblage de quelques capteurs/ actionneurs dans un même programme. Modélisation 3D du ventilateur.
2	Ajout du capteur de distance dans le programme. Premières étapes du traitement de signal du tachymètre.
3	Réalisation de l'interface utilisateur (UI)(impression 3D + soudage PCB + usinage des boutons poussoirs).
4	Impressions 3D des adaptateurs et du filtre. Premiers tests avec tube d'aspiration.
5	Connectiques des capteurs dans le filtre et premiers tests sur leur support.
6	Usinage du socle en aluminium.
7	Finalisation des étages du traitement du signal du tachymètre.
8	Changements de plan sur le type de tube d'aspiration. Débogage et correction d'une mauvaise compatibilité entre périphériques SPI (section 4.).
9	Conception du PCB principale.
10	Corrections des problèmes du PCB.
11	Développement du programme et de tous les systèmes embarqués de l'UI.
12	Finalisation et rédaction du rapport.

Table 1. Étapes de la réalisation du projet

4. PROBLÈMES RENCONTRÉS

Dans cette section, seuls les problèmes les plus importants rencontrés lors de la réalisation du projet seront abordés. Bien que d'autres problèmes aient été identifiés, ils ont eu moins d'impact sur l'avancement global du projet et ne seront pas traités ici.

4.1. Plusieurs périphériques SPI dans un même bus

Comme évoqué dans la section précédente, lors de la huitième semaine de travail sur le projet, j'ai identifié et corrigé un problème important qui avait été négligé jusqu'alors. Ce problème affectait mon module ST7735, qui est responsable de l'affichage sur l'écran TFT ainsi que de la gestion de la carte SD. Bien que la fonctionnalité combinée de ces deux éléments soit très pratique, j'ai rapidement découvert que les deux ne pouvaient pas fonctionner sur le même bus SPI.

Bien que j'ai pu faire fonctionner l'écran indépendamment, j'ai rencontré des conflits de

bibliothèques lorsque j'ai essayé de combiner l'écran et la carte SD, ce qui m'a empêché de lire ou écrire sur cette dernière. Après des recherches approfondies, je n'ai trouvé aucun tutoriel ou forum expliquant comment faire fonctionner les deux éléments ensemble. Je tournais en boucle, cela ne servait à rien.

J'ai pris l'initiative de remédier à ce problème par mes propres moyens. Après avoir étudié les datasheets de l'écran et de la carte SD, j'ai découvert que les deux périphériques ne fonctionnaient pas sous la même fréquence d'horloge SPI. En effet, l'écran fonctionne à une fréquence de 20 kHz, ce qui est extrêmement faible pour un bus SPI, tandis que la carte SD fonctionne sous une fréquence de 4 MHz. Par conséquent, il n'était pas possible de les faire fonctionner sur le même bus SPI, ce qui m'a contraint à créer deux bus SPI indépendants dans le circuit.

Après avoir connecté et reprogrammé tous les composants, j'ai constaté que l'écran fonctionnait correctement, mais pas la carte SD. Toutefois, je savais pourquoi cette fois-ci. En effet, avec le même code et un lecteur de carte SD externe, les deux périphériques fonctionnaient mutuellement. J'en ai donc déduit qu'il y avait une mauvaise compatibilité de tensions logiques entre l'ESP32 et le lecteur de carte SD du modèle ST7735. Après une analyse minutieuse, j'ai remarqué que sur le PCB de ce module, des résistances abaisseuses de tension étaient placées pour les broches SCK, MISO et MOSI. Par conséquent, j'ai déduit que ce module était conçu pour une tension logique de 5V plutôt que de 3V3.

J'ai donc court-circuité les résistances du module, mais malgré cela, la carte SD ne fonctionnait toujours pas. J'ai immédiatement compris pourquoi. Juste à côté des résistances se trouvait un régulateur de tension de 3V3 qui alimentait la carte SD. Hors, en entrée de ce régulateur, il convient de fournir une tension de 5V, et non de 3V3. J'en ai conclu que ce module ST7735 était dimensionné pour les Arduino (travaillant sous du 5V), et non les ESP32 (travaillant sous du 3V3). C'est pourquoi j'ai procédé à la suppression de ce régulateur, en réalisant un court-circuit entre son entrée et sa sortie, permettant ainsi de fournir directement une tension de 3V3 à la carte SD. Cette modification a permis de résoudre le problème de compatibilité entre l'ESP32 et le lecteur de carte SD du ST7735.

Le dépannage de cette problématique a nécessité une durée de plus de quatre heures, impactant ainsi négativement ma soirée. Toutefois, l'issue positive de la situation m'a permis de passer une nuit paisible.

4.2. Le choix du microcontrôleur

À ce stade de la lecture de ce dossier, vous avez certainement une idée précise sur le sujet et avez déjà formé votre opinion. Cependant, il est important pour moi de mentionner un détail qui a peut-être traversé votre esprit, mais que je n'ai pas encore abordé :

« Pourquoi utiliser l'ESP32 sans exploiter sa connexion WLAN ? »

En effet, l'ESP32 est capable d'exécuter des tâches WLAN telles que le Wifi 2,4 GHz et le Bluetooth 2,4 GHz. Vous pourriez alors objecter :

« C'est comme utiliser une GoPro[2] pour prendre des photos banales. »

En effet, je reconnais que l'utilisation de la GoPro dans ce contexte serait peu judicieuse. Toutefois, ma situation présente des particularités qui ne sauraient être comparées à ce cas de figure.

En partant du principe que vous avez raison et que le microcontrôleur actuel doit être remplacé par un autre plus adapté, j'ai mené une petite étude, qui a conduit au choix du microcontrôleur STM32 comme solution idéale pour le présent projet, étant donné l'absence de connexion WLAN. En effet, il s'avère être moins surdimensionné qu'un ESP32, tout en offrant des performances surprenantes. Avec une fréquence CPU de 72 MHz, presque équivalente à celle de l'ESP32 (160 MHz), un nombre de broches d'entrée/sortie de 37 (contre 36 pour l'ESP32), et surtout une capacité à gérer les interruptions de manière beaucoup plus performante que l'ESP32, il présente toutes les caractéristiques nécessaires pour ce projet [2].

Je suis conscient que le choix du microcontrôleur est un sujet délicat, qui nécessite des justifications argumentées. Cependant, il est important de rappeler que ce projet n'est pas un produit fini, mais un prototype. Les améliorations les plus fréquentes d'un prototype sont d'abord de nature matérielle, mais par la suite, c'est le logiciel qui ne cesse de se mettre à jour.

Dans l'hypothèse où le matériel resterait tel quel, c'est-à-dire que le PCB ne serait pas modifié, il serait toujours possible d'améliorer le produit en termes de logiciel. En effet, à ce stade, je pourrais débloquer la connexion WLAN et concevoir des applications typiques de l'IoT[3]. Si j'avais choisi le STM32 pour ce projet, pour procéder à ce type d'amélioration, j'aurais dû modifier le PCB pour y intégrer un module WLAN supplémentaire, ce qui n'est pas rentable. En revanche, l'ESP32 peut tout faire en un seul module.

[2]GoPro : caméra dotée d'une protection la rendant résistante aux chocs et étanche. Ultra polyvalente et tellement petite qu'elle tient dans la poche (prix compris entre 300 & 400€) [1].

[3]IoT : Internet of things, ou Internet des Objet en français. L'IoT reflète les objet connectés équipés de capteurs et actionneurs accessibles à distance depuis des logiciels et autres [3].

5. RÉSULTATS FINAUX

Les résultats finaux de ce projet ont été un vrai succès. Le produit a pu respecter l'entièreté des critères du cahier des charges.

5.1. GitHub

J'ai mis à disposition ce projet sur un dépôt GitHub afin de le rendre accessible aux personnes les plus curieuses d'entre vous. Si vous désirez obtenir des informations techniques supplémentaires, ce dépôt est tout indiqué :

https://github.com/DavideDiVenti/myVacuum

5.2. Résultats finaux mécaniques

En ce qui concerne la partie mécanique de ce projet, des images sont plus explicites que des mots. Je vous invite donc à consulter l'annexe 4 pour avoir une meilleure compréhension. Dans le dépôt GitHub cité dans la section 5.1 de ce rapport se trouve également des informations supplémentaires, notamment un site web (propre à Fusion 360) où il nous est possible visualiser en 3D le projet en ligne.

Je tiens à souligner que chaque partie du projet a été conçue intégralement par mes soins, en utilisant mon expérience et mon imagination dans le domaine. Les seules pièces qui n'ont pas été conçues par moi-même sont le réservoir (ou plutôt le tube en acrylique transparent de diamètre 100 mm, que j'ai acheté), le ventilateur (que j'ai récupéré d'un vieux PC) et le tissu carbonisé servant de filtre à air (que j'ai acheté). La visserie a également été achetée. Pour le reste, j'ai utilisé des matières premières et j'ai fabriqué les pièces nécessaires moi-même.

Un élément clé qui n'a pas encore été discuté est l'architecture de l'aspirateur. Chaque composant est minutieusement placé, avec une justification précise pour chaque emplacement. Il est conseillé d'avoir l'annexe 4 à portée de main avant de décrire chaque emplacement des composants.

L'aspirateur est soutenu par quatre pieds en caoutchouc, fixés sur une base en aluminium. Le réservoir, un tube en acrylique transparent de 100 mm de diamètre, est emboîté dans cette base. Pour assurer une étanchéité adéquate, un joint en TPU est intercalé entre les deux éléments. À côté du tube transparent se trouve un orifice destiné au tube d'aspiration. Pour fixer le tube, un adaptateur est fixé à l'orifice, également équipé d'un joint. Le tube d'aspiration vient se visser dans cet adaptateur. Dans l'opposé de cet orifice, un autocollant en vinyle portant l'inscription myVacuum est apposé sur la circonférence du réservoir.

Au-dessus du réservoir, un adaptateur permet de fixer le filtre et le ventilateur. Le filtre contient trois supports pour les capteurs (DHT11, MQ-2 et VL6180X), le dernier étant dirigé vers le bas pour mesurer le niveau de remplissage du réservoir. Une plaque en acrylique transparente est placée entre le fond du réservoir et la base du filtre pour permettre la transmission de la lumière des capteurs tout en protégeant le capteur des déchets. Cette plaque est fixée à la base du réservoir.

Le support du PCB se trouve sur le ventilateur, qui est conçu pour épouser la même forme visuelle que le ventilateur. Le connecteur de l'interface utilisateur est branché sur le PCB. Cette interface portable, mais avec fil.

L'interface utilisateur est dotée de quatre pieds en caoutchouc et fixée sur une base rectangulaire en aluminium. Quatre entretoises sont vissées sur cette base pour accueillir le PCB de la manette. Un couvercle en PLA est ensuite clipsé sur le PCB pour renfermer l'électronique, n'exposant que l'écran, l'accès à la carte SD et les boutons-poussoirs. Des extensions en aluminium ont été usinées pour les boutons-poussoirs, leur permettant ainsi d'être accessibles depuis l'extérieur du couvercle.

5.3. Résultats finaux électroniques

Dans la partie électronique se trouve l'étape de traitement du signal du tachymètre. La figure 2 ci-dessous illustre l'ensemble des étapes du traitement du signal, depuis sa capture initiale jusqu'à son traitement final :

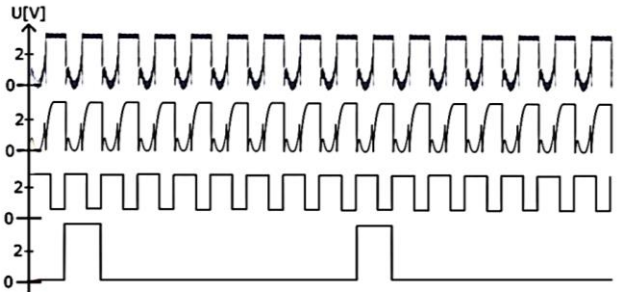


Figure 2. Signaux filtrés d'étage en étage

Il s'agit ici du signal correspondant à la vitesse maximale du ventilateur. Plus la vitesse est réduite, moins l'intensité des interférences est élevée. Le signal n°1 est celui issu du ventilateur. À chaque impulsion carrée (pulled up par une résistance que j'ai ajoutée), le signal en question est précédé. Ce dernier est généré par le capteur à effet Hall et sa forme est évolutive et croissante dans le temps, jusqu'à ce que le capteur ne soit plus dans la portée magnétique du pôle encastré dans l'axe rotatif du ventilateur. Dans ce signal se trouvent donc deux signaux indépendants, l'un étant celui généré par le capteur à effet Hall, et l'autre, qui le suit, étant celui de la résistance tirée vers 3V3. Ce signal carré de 3V3 est tiré vers le haut dès l'instant où le capteur à effet Hall sort de la portée magnétique du pôle

de l'axe du ventilateur. Et c'est ce signal carré qui sera exploité tout au long du traitement du signal.

Le signal n°2 est celui filtré par le filtre passif passe-bas RC. Ce signal se différencie du premier par sa forme plus douce et atténuée.

Le signal n°3 est celui de la sortie de l'amplificateur opérationnel utilisé comme comparateur, tel un trigger de Schmitz. Le signal « offset » généré par le capteur à effet Hall est ainsi neutralisé, de sorte à avoir uniquement le signal carré 0V-3V3.

Quant au signal n°4, il s'agit du signal final. Une fois que le signal n°3 est passé par le registre à décalage 8 bits, sa fréquence est divisée par 8. Ce sera ce signal qui sera lu et interprété par l'ESP32 afin qu'il puisse le convertir en tr/min.

En ce qui concerne le reste de l'électronique, des schémas permettent une compréhension plus précise. Je vous invite à visualiser les ANNEXES 1,2 & 3.

5.4. Résultats finaux de la programmation

La programmation est un domaine à part, différent de l'électronique et de la mécanique. Pour cette raison, je ne vais pas expliquer en détail le code utilisé, car il a déjà été brièvement présenté dans la section 2.4 de ce rapport. Cependant, pour une meilleure compréhension, j'ai mis le code à disposition dans le dépôt GitHub que j'ai mentionné dans la section 5.1 de ce rapport.

6. CONCLUSION

En conclusion, la conception et la réalisation de cet aspirateur numérique ont été une expérience enrichissante pour moi en tant qu'étudiant en ingénierie. Malgré un budget limité, j'ai réussi à concevoir et à construire un système efficace en faisant des choix judicieux de composants et en utilisant des principes de traitement du signal appropriés, permettant ainsi d'obtenir des résultats satisfaisants en termes de précision et de rapidité.

Des possibilités d'amélioration sont envisageables pour ce système. En particulier, l'intégration de l'IoT permettrait d'améliorer la surveillance et le contrôle à distance de l'aspirateur. Par ailleurs, l'amélioration du PCB peut également être envisagée, car celui-ci n'intègre actuellement que des composants DIP et des modules éducatifs ou de prototypage tels que le DHT11, le ST7735, le VL6180X, etc. Cette configuration entraîne une consommation d'énergie inutile et est gourmande en espace. Pour remédier à cela, il serait judicieux de concevoir un PCB plus compact avec des composants SMD, sans modules éducatif ou de prototypage, consommant ainsi moins d'énergie.

En somme, je considère que ce projet a été une réussite. Cette étude m'a permis de concevoir un produit fonctionnel et utile. Il représente également une base solide pour d'autres projets similaires à l'avenir.

7. RÉFÉRENCES

[1] AntheDesign, Phénomène GoPro & YouTube, *https://www.anthedesign.fr/autour-du-web/phenomene-gopro-you-tube/*, consulté le 03 mai 2023.

[2] ICRFQ, 2022, ESP32 vs STM32, *https://www.icrfq.net/esp32vs-stm32/*, consulté le 03 mai 2023.

[3] SAP, Qu’est-ce que l’IoT (Internet des objets)?, *https://www.sap.com/france/products/artificial-intelligence/what-is-iot-internet-of-things.html*, consulté le 03 mai 2023.

ANNEXES

Annexe 1 : Schéma électronique 1/3 - Microcontrôleur & alimentations

Annexe 2 : Schéma électronique 2/3 - Ventilateur & Traitement du signal

Annexe 3 : Schéma électronique 3/3 - Connecteurs & autres périphériques

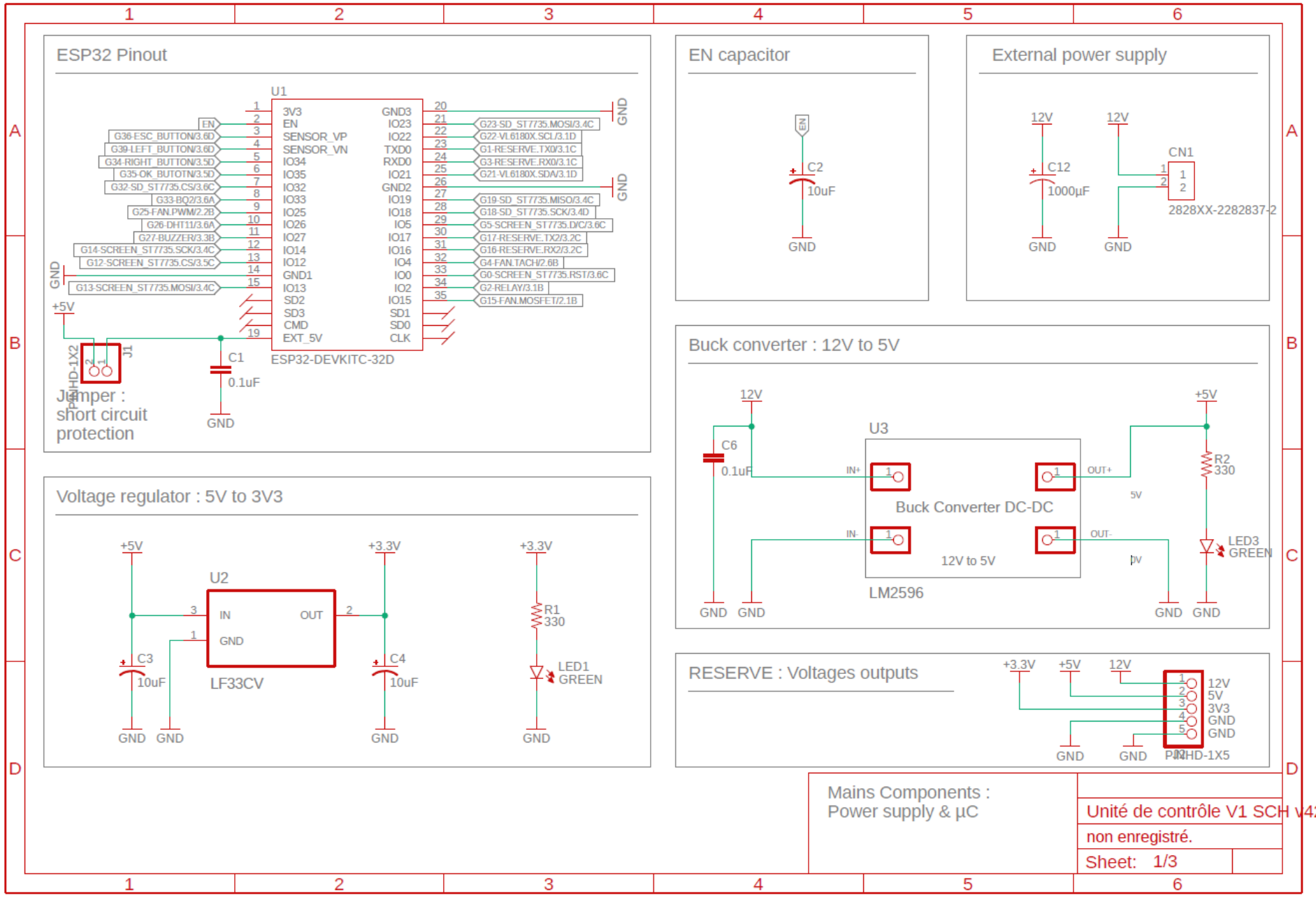
Annexe 4 : Rendu 3D - Vue de gauche & de droite

Annexe 5 : Mode d'emploi 1/3 - Prérequis

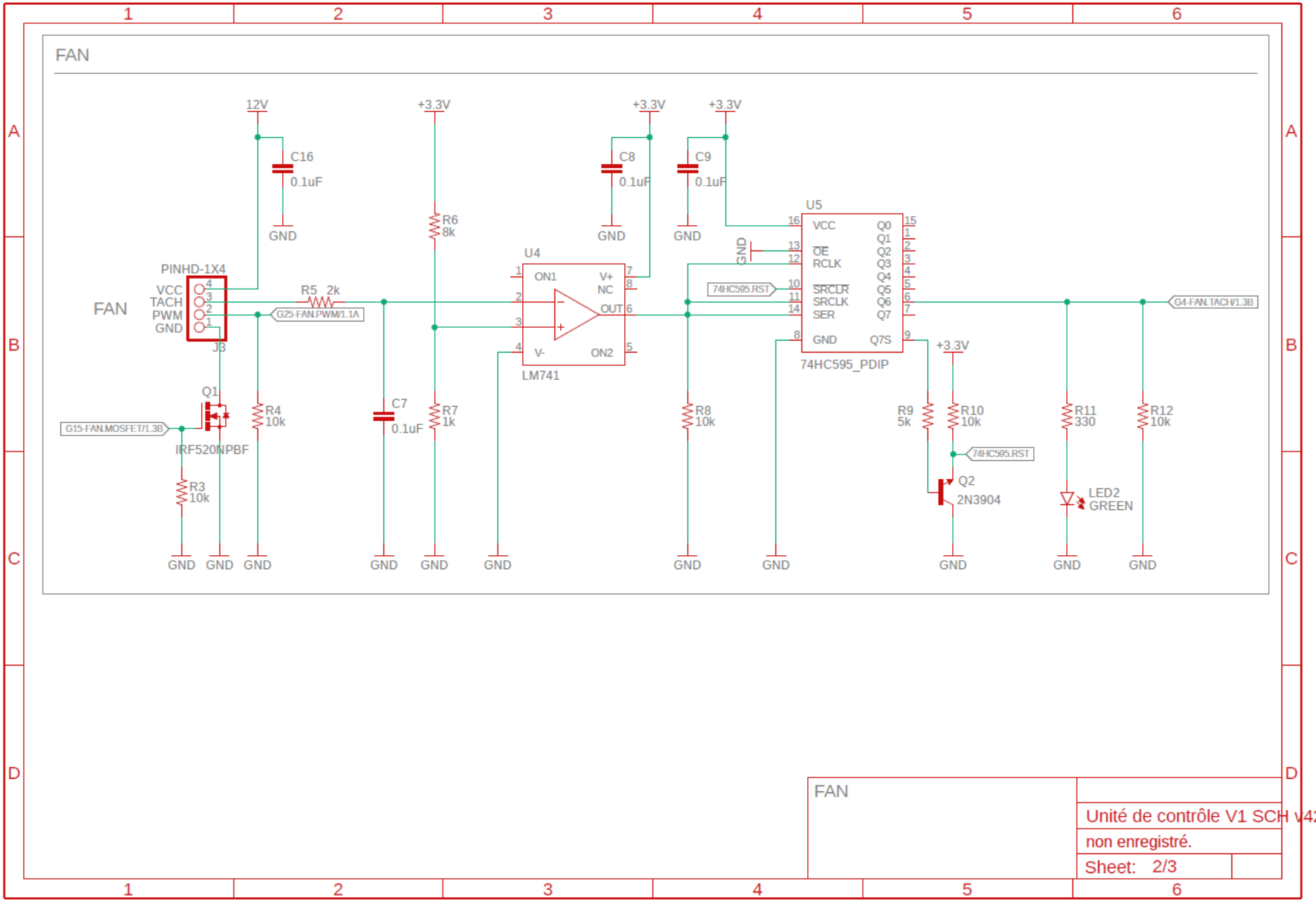
Annexe 6 : Mode d'emploi 2/3 – Schéma bloc du fonctionnement de l'aspirateur

Annexe 7 : Mode d'emploi 3/3 – Pinouts de l'ESP32

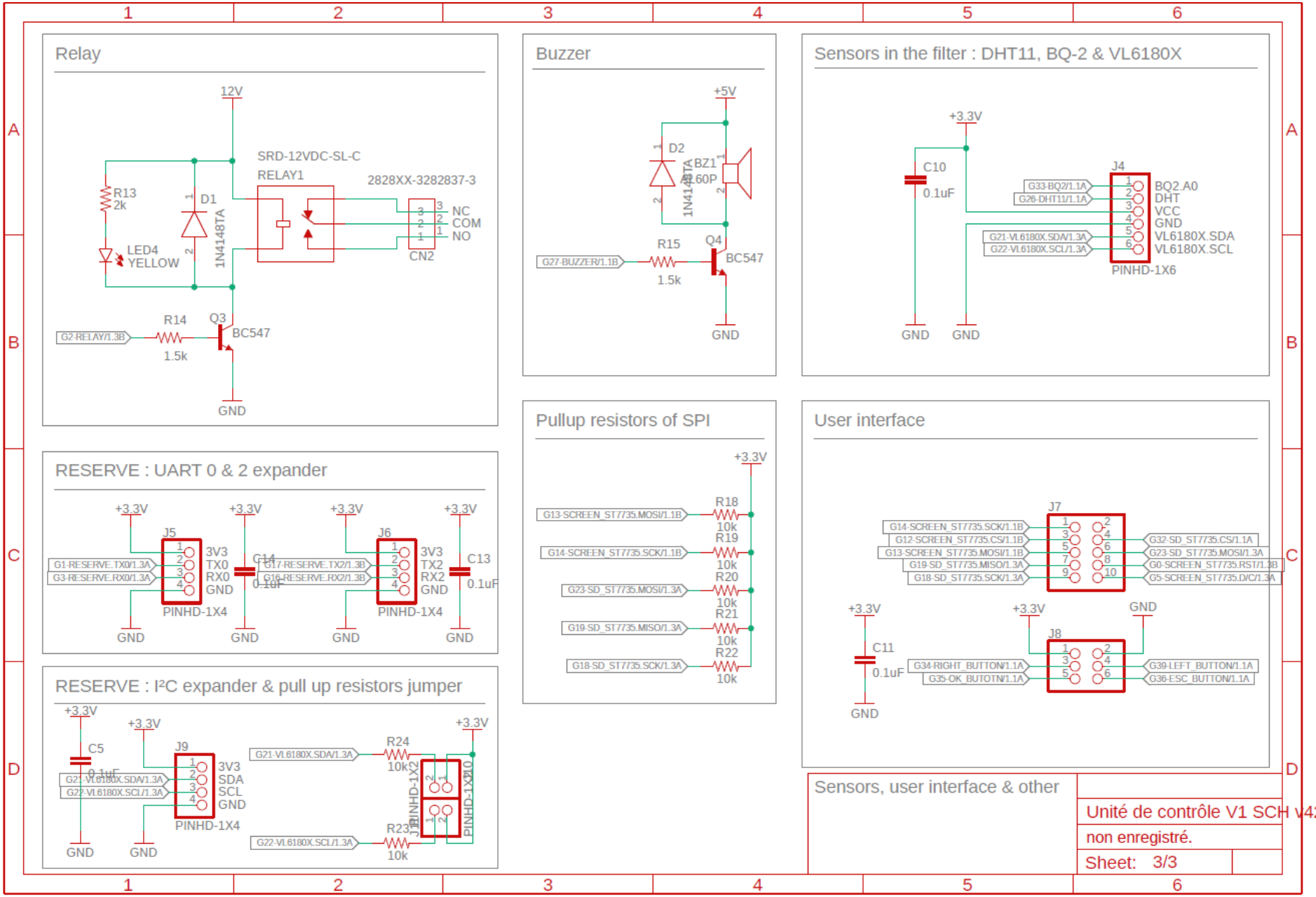
ANNEXE 1 : SCHEMA ELECTRONIQUE 1/3 - MICROCONTROLEUR & ALIMENTATIONS



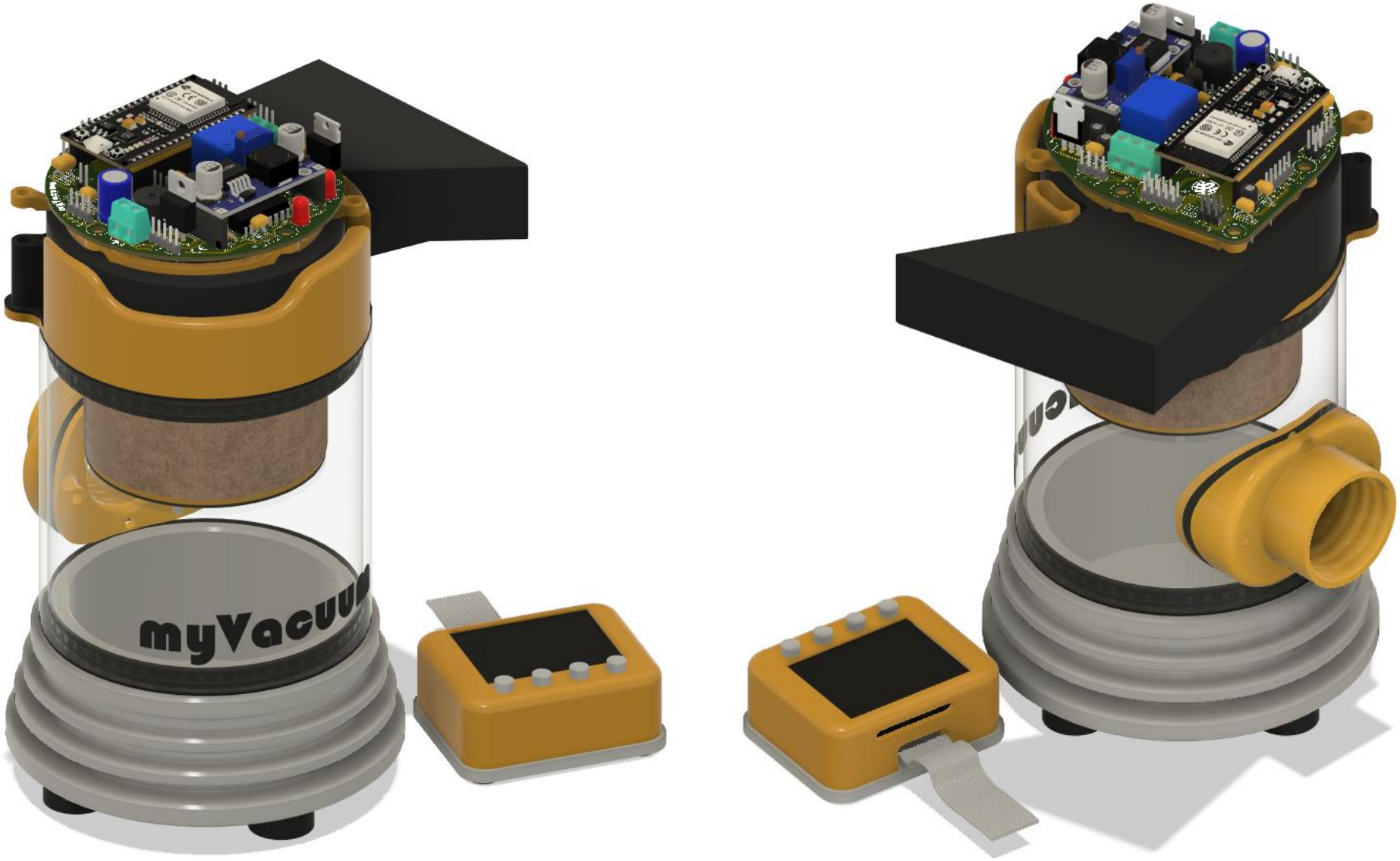
ANNEXE 82 : SCHEMA ELECTRONIQUE 2/3 - VENTILATEUR & TRAITEMENT DU SIGNAL



ANNEXE 163 : SCHEMA ELECTRONIQUE 3/3 - CONNECTEURS & AUTRES PERIPHERIQUES



ANNEXE 243 : RENDU 3D - VUE DE GAUCHE & DE DROITE



ANNEXE 315 : MODE D’EMPLOI 1/3 - PREREQUIS

Ce manuel d'utilisation a pour but de vous guider tout au long de vos premières manipulations de l'aspirateur, afin que vous puissiez le prendre en main en toute facilité.

1. MATERIEL REQUIS

L'alimentation externe n'est pas fournie avec l'aspirateur. Il est donc nécessaire de se procurer une alimentation adaptée à l'appareil, à savoir une alimentation de 12VDC puissant pomper au minimum 5A. La qualité du 12V de l'alimentation peut être tolérée dans une fourchette de 11V à 14V, ce qui signifie qu'une batterie peut également être utilisée.

Une fois que vous disposez d'une alimentation de cette capacité, il vous suffit de connecter le +12V et le 0V aux borniers correspondants de l'aspirateur (1), comme indiqué sur la figure 1. Des indications sont inscrites à côté des borniers pour vous guider et éviter toute confusion.

Par ailleurs, aucune configuration ni logiciel n'est requis pour utiliser l'aspirateur, qui est un produit "plug and play".

2. REGLES D’UTILISATION

Lors de la manipulation de l'aspirateur, il est recommandé de saisir l'appareil par sa base en aluminium. En effet, la structure de l'aspirateur est maintenue par un emboîtement par serrage et pourrait se déboîter en cas de manipulation inappropriée.

Il convient de veiller à utiliser l'aspirateur dans des conditions de température optimales. En effet, si la température ambiante dépasse un certain seuil, la dilatation du réservoir pourrait engendrer le détachement des pièces. Une température avoisinant les 20°C, avec une tolérance de +-10°, est recommandée.

Il est important de noter que l'aspirateur n'est pas conçu pour aspirer des liquides, ce qui pourrait entraîner une détérioration prématurée de l'équipement.

De même, l'utilisation d'une rallonge de tube d'aspiration réduira considérablement l'efficacité de l'aspirateur en entraînant une perte de charge trop importante. Cette situation mettra en sécurité le ventilateur pour éviter les surcharges.

Il est à noter que l'aspirateur ne possède pas de protection d'inversion de polarité. Il convient donc de ne pas inverser les bornes +12V et 0V en entrée du bornier (1), au risque de subir des conséquences regrettables.

Enfin, il est important de souligner que l'aspirateur n'est pas équipé d'un fusible. Il faudra donc se filler au projet.

3. INTERFACE UTILISATEUR

Dans l'annexe 6, vous trouverez un résumé des différentes options proposées par l'interface utilisateur. Pour naviguer entre ces options, vous disposez de 4 boutons : Escape, Left, Right & OK.

Il est important de ne retirer ou d'insérer la carte SD que lorsque l'aspirateur est éteint afin d'éviter tout dysfonctionnement.

4. PROGRAMMATION DE L’ESP32 PENDANT LA MISE EN SERVICE

Il est parfaitement envisageable de programmer l'ESP32 (3) durant la mise en service de l'aspirateur. Le PCB intègre une sécurité destinée à éviter les conflits de tensions externes avec celles internes, notamment avec l'USB d'un ordinateur branché, et ainsi éviter les risques de court-circuit.

Pour cela, il suffit de débrancher le cavalier (2) qui interfère avec le connecteur USB de l'ESP32 pour y connecter le câble de programmation. Si ce cavalier n'est pas débranché, il est impossible de connecter le câble de programmation à l’ESP32.

Avant de programmer, je vous recommande de vous référer à l'annexe 7 pour identifier quel périphérique est connecté à quelle broche de l'ESP32.

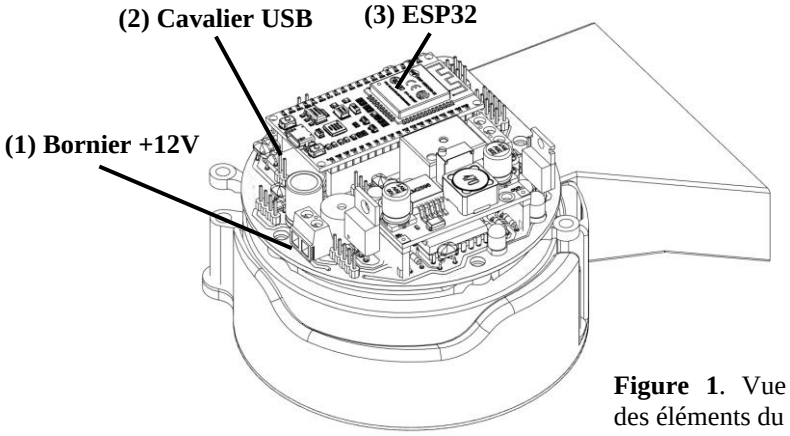
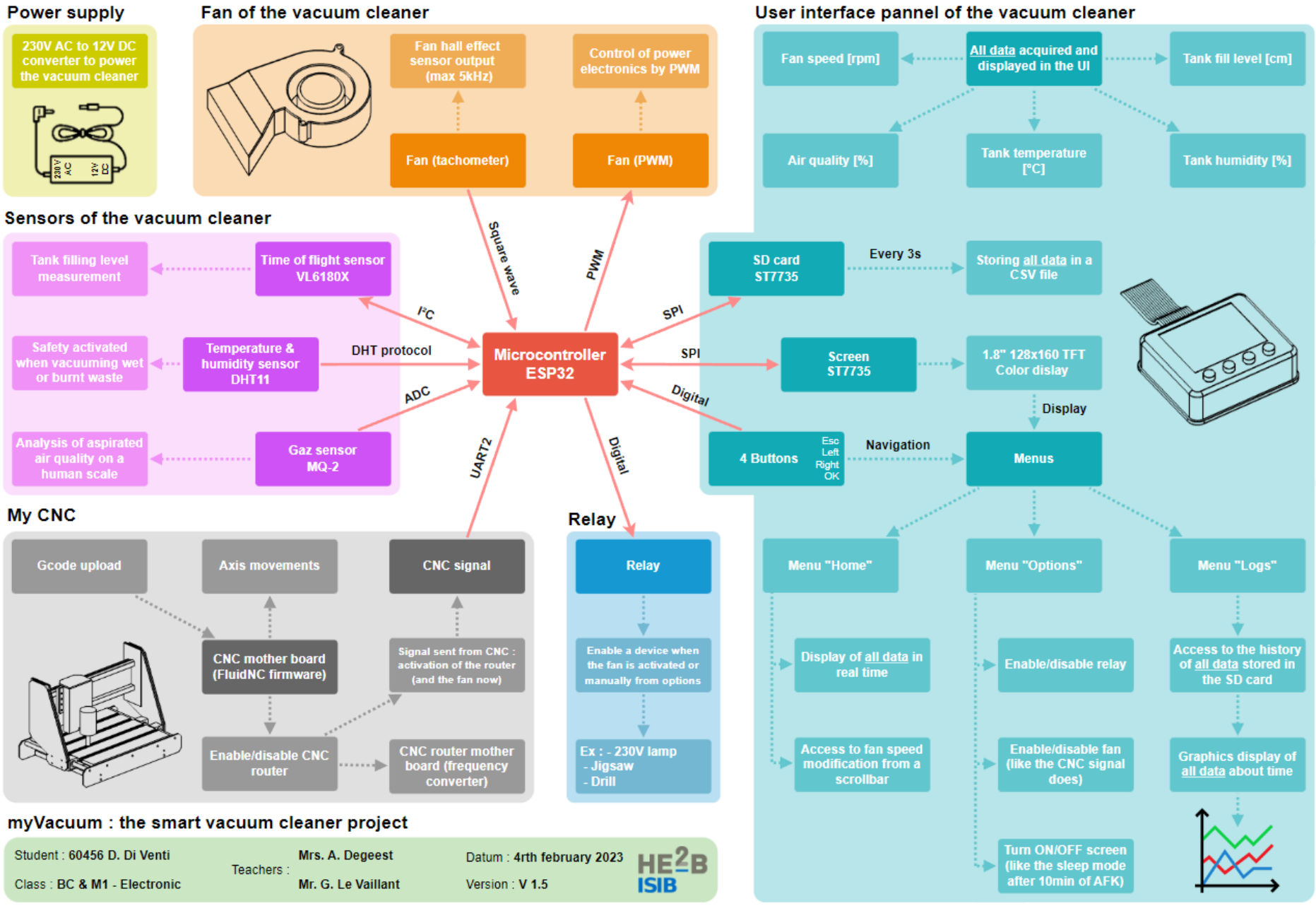
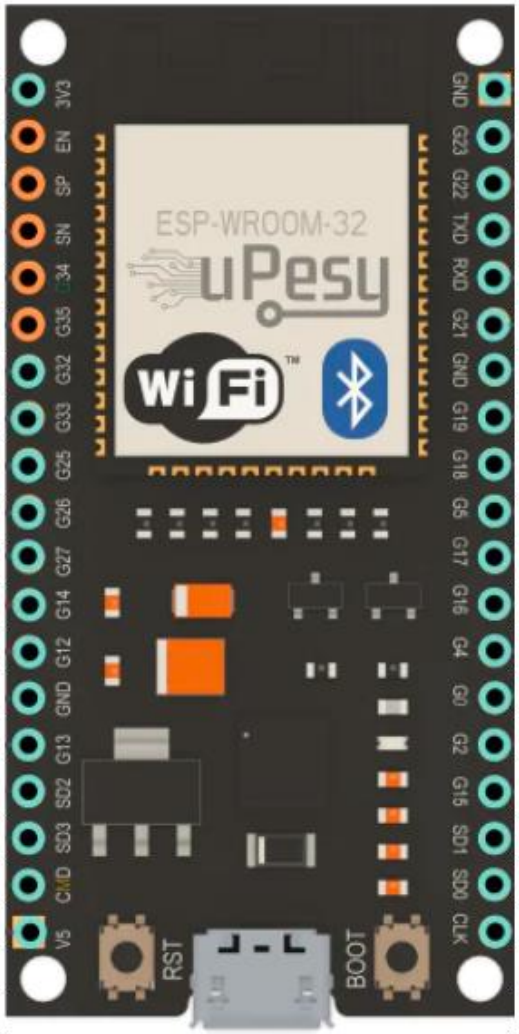


Figure 1. Vue détaillée des éléments du PCB.

ANNEXE 363 : MODE D'EMPLOI 2/3 – SCHEMA BLOC DU FONCTIONNEMENT DE L'ASPIRATEUR



ANNEXE 379 : MODE D'EMPLOI 3/3 – PINOUTS DE L'ESP32



Interface	Types	Devices			Devices	Types	Interface
			3V3	GND			
			EN	23	SD ST7735	→	MOSI
Digital	→	ESC button	36	22	VL6180X	→	SCL
Digital	→	LEFT button	39	1	/		TX0
Digital	→	RIGHT button	34	3	/		RX0
Digital	→	OK button	35	21	VL6180X	↔	SDA
CS	←	SD ST7735	32	GND			
ADC	→	MQ-2	33	19	SD ST7735	←	MISO
DAC	←	Fan PWM	25	18	SD ST7735	→	SCK
DHT protocol	↔	DHT11	26	5	Screen ST7735	→	D/C
PWM	←	Buzzer	27	17	/		TX2
SCK	←	Screen ST7735	14	16	/		RX2
CS	←	Screen ST7735	12	4	Fan tachometer	←	Digital
			GND	0	Screen ST7735	→	RST
MOSI	←	Screen ST7735	13	2	Relay	→	Digital
			9	15	Fan MOSFET	→	Digital
			10	8			
			11	7			
DC-DC Buck	→	LM2596	5V	6			