Projet 44 : Clone amélioré d'Arduino Mega

Encadrants :

Alexandre Boé

Xavier Redon

Théau Moinat

IMA4 – 2018/2019

Remerciements

Je tiens tout d'abord à remercier Matthieu Delobelle, étudiant en IMA5, qui a pris de son temps pour m'expliquer en détail les tenants et les aboutissants du projet. Je remercie par ailleurs également l'équipe de Robotech qui m'a permis d'accéder plusieurs fois aux matériels à leur disposition, pour me permettre d'établir les caractéristiques nécessaires à la réalisation de ce projet.

Je remercie également mes tuteurs, Xavier Redon et Alexandre Boé, qui ont pris le temps de répondre à mes questions et me conseiller dans certains choix techniques.

J'adresse un remerciement particulier à Thierry Flamen, qui nous a apporté à mes camarades et moi-même nombreux conseils quant à la réalisation de carte électronique et qui nous a supportés durant tout ce semestre en C201 chaque mercredi.

Je remercie également mes camarades qui m'ont apporté un soutien indéfectible tout au long du projet, sans vous la vie serait moins lumineuse. Mention spéciale à Hugo Velly et Rémi Foucault qui m'ont aidé à l'élaboration de ma bibliothèque de composants Altium.

Introduction

Comme chaque année, nous sommes menés à réaliser un projet durant notre deuxième semestre de quatrième année en IMA. J'ai choisi d'effectuer un projet intitulé « Clône amélioré d'Arduino ».

Le but de ce projet était de réaliser une carte Arduino à destination de l'association de robotique de Polytech « Robotech ». D'après le cahier des charges fourni, je devais réaliser une carte basée sur un Arduino Mega possédant un contrôleur moteur, une connexion sans-fil permettant dialogue et reprogrammation à distance ainsi que le nécessaire pour alimenter la carte sur batterie. Aucune restriction de taille n'a été donnée, cependant l'utilisation exclusive de composants de surface CMS était obligatoire, choix accommodant car permettant de réduire (drastiquement) la taille, au prix d'une réalisation manuelle plus complexe. La principale difficulté de ce projet était l'association d'une partie d'électronique numérique et d'une partie d'électronique de puissance donc potentiellement dangereuse pour la première et l'utilisateur (quiconque a déjà vu un court-circuit sur une batterie Li-Po saura de quoi il est question). Nous verrons tout au long des schémas comment nous avons fait pour « protéger » chacune des parties.

Pour la partie microcontrôleur Mega, je me suis entièrement inspiré de ce que propose la société Arduino avec son Mega. Les fichiers de conception étant libre d'accès, je me suis basé sur la version 3E, la dernière en date pour la réalisation de cette partie. J'ai donc utilisé un Atmega16U2 pour la communication avec l'ordinateur et un Atmega2560 pour la partie

Pour la partie sans-fil, j'ai décidé d'utiliser un microcontrôleur connu des bidouilleurs pour sa flexibilité et son prix accessible : l'ESP8266 de chez Espressif System. Il dispose d'une connexion WiFi et d'un ensemble de foncions tels que l'Arduino Mega.

Pour tout ce qui concerne la gestion des moteurs et des tensions d'alimentations je me suis orienté vers le matériel Texas Instrument. Ce sont des puces qui sont certes plus onéreuses que l moyenne mais il s'agit d'une marque que j'affectionne pour leur fiabilité à toute épreuve et qui est largement utilisée dans l'industrie électronique.

Bien que la carte soit conçue et prête à être imprimée, je n'ai malheureusement pas eu le temps de la concevoir physiquement et de la tester. Cependant, ce rapport permettra de mettre en lumière les différentes fonctions de la carte et de vérifier par la même occasion qu'aucune erreur ne s'est glissée lors de la conception.

Nous verrons d'abord les différents schémas de la carte et les fonctions de chacun d'entre eux. Puis nous verrons finalement comment la carte est réalisée « mécaniquement ».

Sommaire

Remerciements 2
Introduction 3
1. Schémas de la carte 5
   A. L'atmega2560 5
   B. L'atmega16U2 6
   C. L'ESP8266 8
   D. Alimentation et gestion des moteurs 10
   E. Alimentation de la partie numérique 13
2. Design du PCB 14
   A. L'Arduino Mega 15
   B. L'ESP8266 16
   C. Gestion de la batterie 16
   D. Alimentation et gestion des moteurs 17
   E. Alimentation des modules numériques 17
Conclusion 18
Annexes 19

1. Schémas de la carte

Dans cette partie, nous aborderons les différents schémas de la carte électronique et leurs conceptions. Nous nous attarderons sur certaines fonctions et sur la communication des cartes. Sera également justifié le choix des composants tout au long de cette partie à chaque fois que cela sera nécessaire. Nous commencerons tout d'abord par l'élément le plus important : l'atmega2560. Nous verrons par la suite l'atmega16U2 permettant l'interface ordinateur – microcontrôleur. Ensuite nous verrons l'ESP8266, et sa « simplicité » et nous finirons par les modules d'alimentation et de gestion des moteurs et enfin l'alimentation de la partie numérique de la carte. Les schémas seront accessibles sur page entière en annexe.

A. L'atmega2560

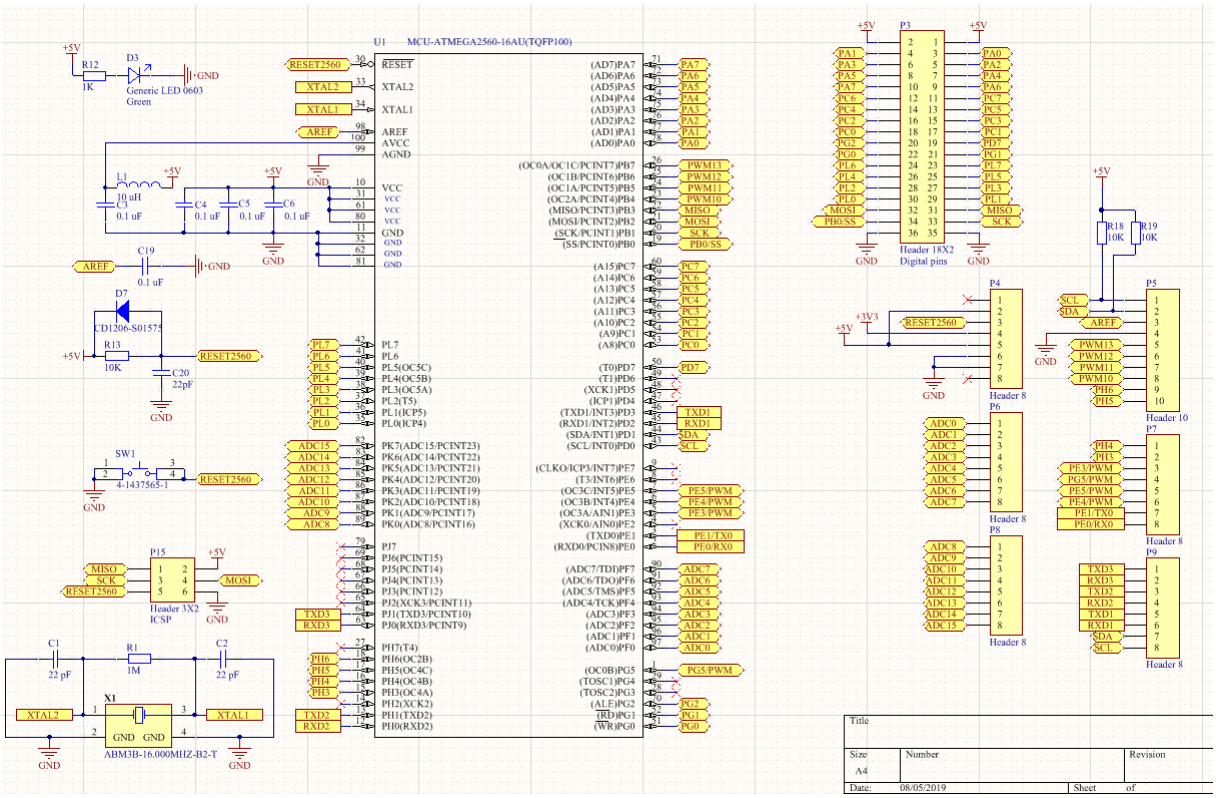


Figure 1 : Schéma de l'atmega2560

Il n'y a rien d'extravagant dans cette partie qui suit quasi à la lettre le schéma fourni par Arduino. Nous pouvons relever les quelques broches qui ne sont pas connectées et pouvons voir tout de suite qu'il s'agit de certaines broches d'interruptions du microcontrôleur ainsi qu'absolument toutes les voies clock (sauf celle destinée à la programmation In Situ). L'atmega2560 tel qu'il est conçu ici ne pourra être utilisé que pour des communications asynchrones, à l'exception du protocole I2C.

Le respect le plus strict de la documentation fournit par Arduino a pour but de rendre la carte compatible avec les boucliers déjà disponibles sur le marché. Ils seront ainsi compatibles « Out of the box ».

B. L'atmega16U2

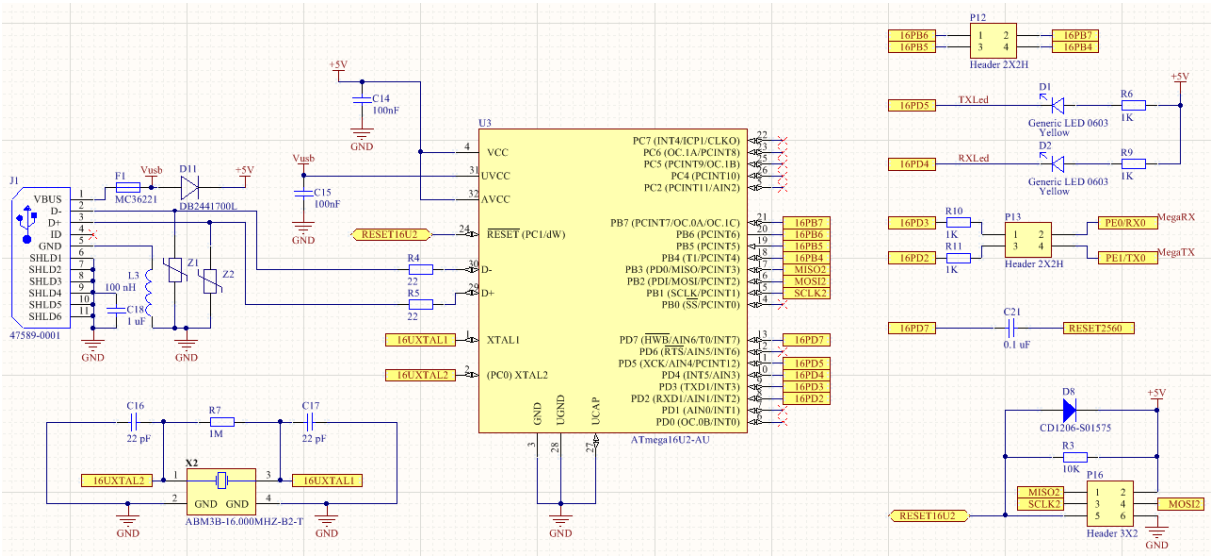


Figure 2: Schéma de l'atmega16u2

Je me suis interrogé sur l'utilisation d'un FT232 qui permet également la communication USB <-> UART, mais après discussion avec Matthieu, il était préférable de garder le 16U2. La raison est plutôt simple, lors de notre premier semestre de quatrième année, nous avons appris à programmer le 16U2 pour le faire reconnaître en tant que périphérique USB.

Cette option restant fort intéressante (par exemple, en le couplant à un ordinateur de bord tel qu'un Raspberry pi pour ne plus se limiter à son port GPIO), il nous est paru important de la garder. Ne nous fermons aucune porte.

Comme pour l'atmega260, il n'y a ici également que peu de nouveautés, hormis quelques-unes.

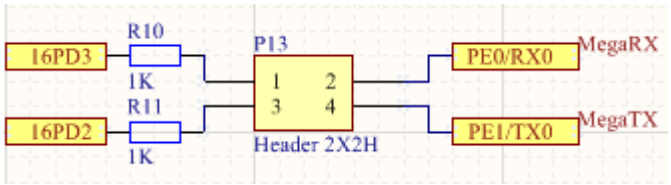


Figure 3:Header Communication 16u2

Nous noterons l'ajout d'un connecteur entre l'atmega16u2 et l'atmega2560. En effet comme le 2560 est susceptible d'être reprogrammé à distance, il peut être nécessaire de déconnecter l'interface entre ces deux chipsets pour ne pas interférer entre la communication du 2560 et de l'ESP8266. Plusieurs « témoignages » vont dans ce sens sur la toile et Monsieur Boé le confirma également. Mieux vaut prévenir que guérir.

Dans cette optique également, il existe une connexion qui n'est pas utilisée sur les Arduino Mega grands publics, servant à priori de déverminage ou de signal d'interruption :

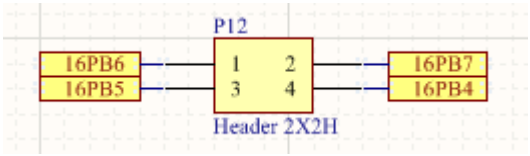


Figure 4: Header d'interruptions

Il n'agit pas ici d'une nouveauté mais notons son existence.

Une autre nouveauté est l'apparition d'une diode entre l'alimentation USB et la voie 5V.

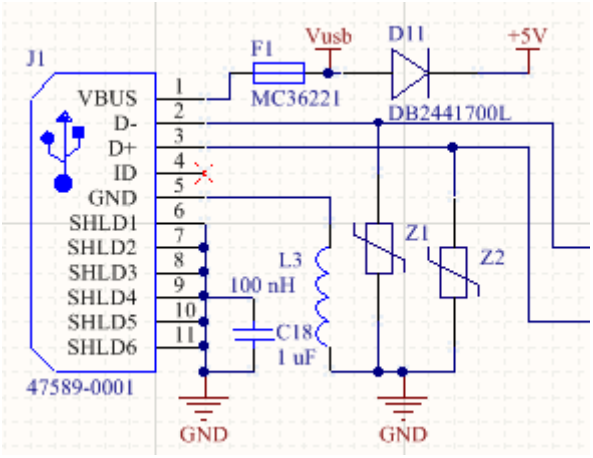


Figure 5: Port USB du 16U2

Cette diode (D11) servira à éviter tout retour de courant si jamais la carte est à la fois branchée sur un PC et sur batterie ou sur alimentation externe. Cela permettra en effet de griller le port USB du PC. Notons la présence de diode de décharge électrostatique pour empêcher également une surtension sur les ports DATA de l'USB.

A la différence de l'Arduino Mega qui utilise un USB type B, j'ai fait le choix d'utiliser un micro-USB de type A/B. Ce n'est pas à cause de la taille, bien que cela joue, mais parce qu'il s'agit d'une connexion que nous avons souvent tous à portée de main du fait de sa généralisation, à l'inverse de l'USB B qui reste toujours une connexion plutôt spécialisée (pas besoin de fournir un câble USB B vers USB A).

C. L'ESP8266

Le choix de ce composant est évident : c'est un des plus simples à mettre en œuvre, il possède un excellent rapport qualité/prix et demande peu de composants additionnels, 11 d'après la documentation technique. De plus, l'utilisation d'une mémoire est fortement recommandée, cela peut être très utile lors de la collecte de données volumineuses telles que des photos ou sur l'acquisition de données de capteurs continue.

Alors s'il est si simple à mettre en œuvre, autant le faire soi-même, cela permettra d'optimiser la taille de la carte tout en étant plus instructif !

Cependant, je rencontre rapidement deux difficultés : il existe qu'un seul guide de design matériel fourni par Espressif et il s'agit de la réalisation sur une plaque « 4 couches » faite par une société externe qui a laissé des inconnues (certaines valeurs de capacités et de bobines) et la seconde difficulté est l'antenne, sa réalisation et son dimensionnement.

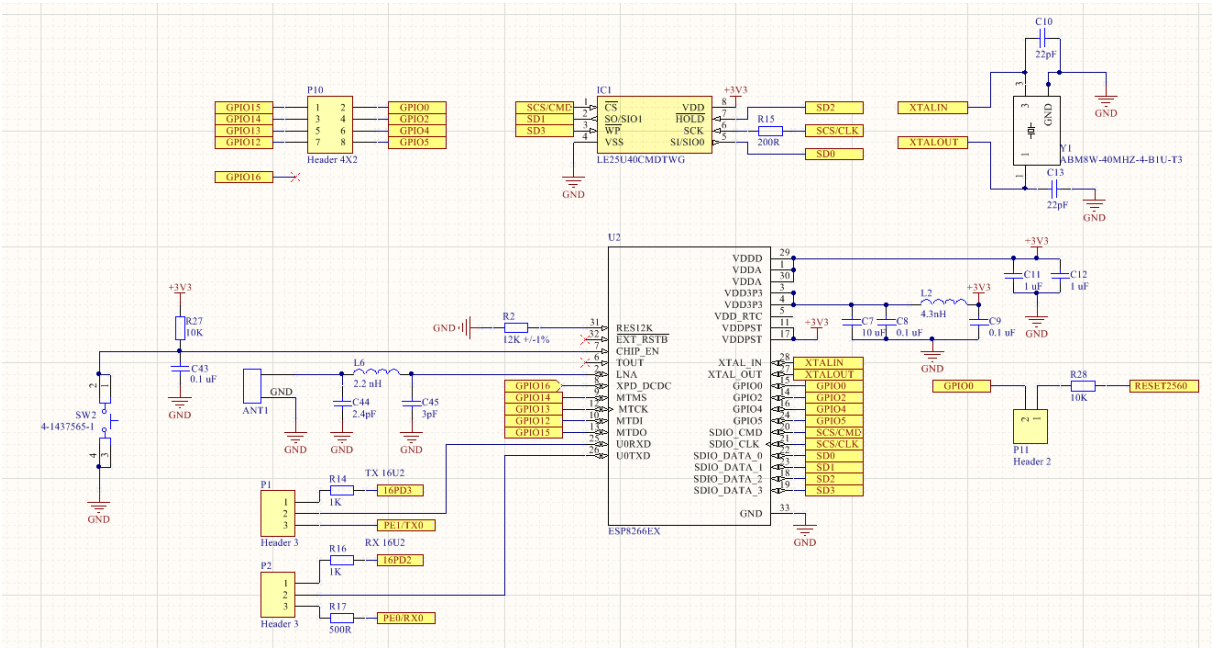


Figure 6: Schéma de l'esp8266

Concentrons-nous d'abord sur le Reset de l'esp8266. Il est recommandé par la documentation technique de ne pas utiliser la fonction censée être prévue pour. En effet, cela peut créer des instabilités si le chipset est actif. Par conséquent, j'ai choisi d'utiliser le port 7 Chip_enabler pour effectuer cette opération, comme recommandée. L'inconvénient est qu'il est impossible de le mettre en veille matériellement. Ceci dit, ce n'est pas un périphérique qui consomme énormément (environ 200mA au max) et comme il n'est pas prévu de l'utiliser sur un temps d'utilisation prolongé (de plusieurs jours), ce n'est pas un problème.

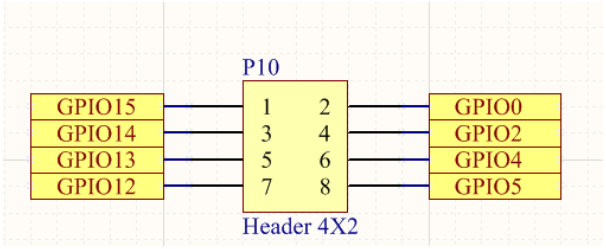


Figure 7 : Header ESP8266

Plusieurs pins sont reliés à un header facilement accessible (nous verrons dans la partie PCB où il se situe) afin de permettre de s'en servir indépendamment si besoin.

Il est possible de reprogrammer l'esp8266 à l'aide de l'atmega16u2 (même s'il est nécessaire d'appuyer sur le bouton pour reset l'esp8266 manuellement). Comme il s'agit des mêmes ports pour la programmation et la communication série, ces ports fonctionnent sous système de jumpers : il faut les déplacer pour soit communiquer avec le 16u2 soit communiquer avec l'atmega2560.

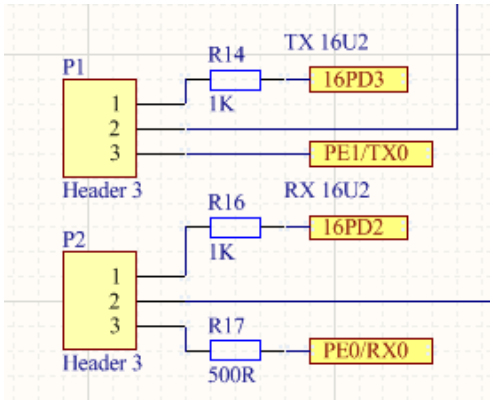


Figure 8: Jumpers ESP8266

Bien qu'à priori ce choix ne soit pas pratique, si le 16u2 est utilisé pour la reprogrammation de l'esp8266 cela implique qu'il est branché à un PC et qu'en conséquence la carte est proche de l'utilisateur. Il n'y a donc aucune raison que cela soit gênant.

Dans le même principe, un pin relie le GPIO0 de l'esp8266 au Reset de l'atmega2560. Un jumper est également présent si jamais le port GPIO0 est utilisé à d'autres fins (il faudra alors retirer ce jumper.)

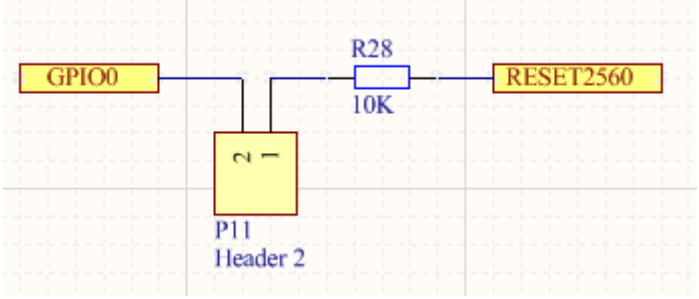


Figure 9: Reset de l'atmega2560 via le GPIO0

D. Alimentation et gestion des moteurs

Il s'agit sûrement de la partie la plus conséquente du projet et celle qui mérite le plus d'explication. Beaucoup de temps a été nécessaire notamment pour trouver les composants adéquats, dimensionner les besoins des moteurs (qui consomment jusqu'à 3A) et par conséquent choisir les modules adaptés.

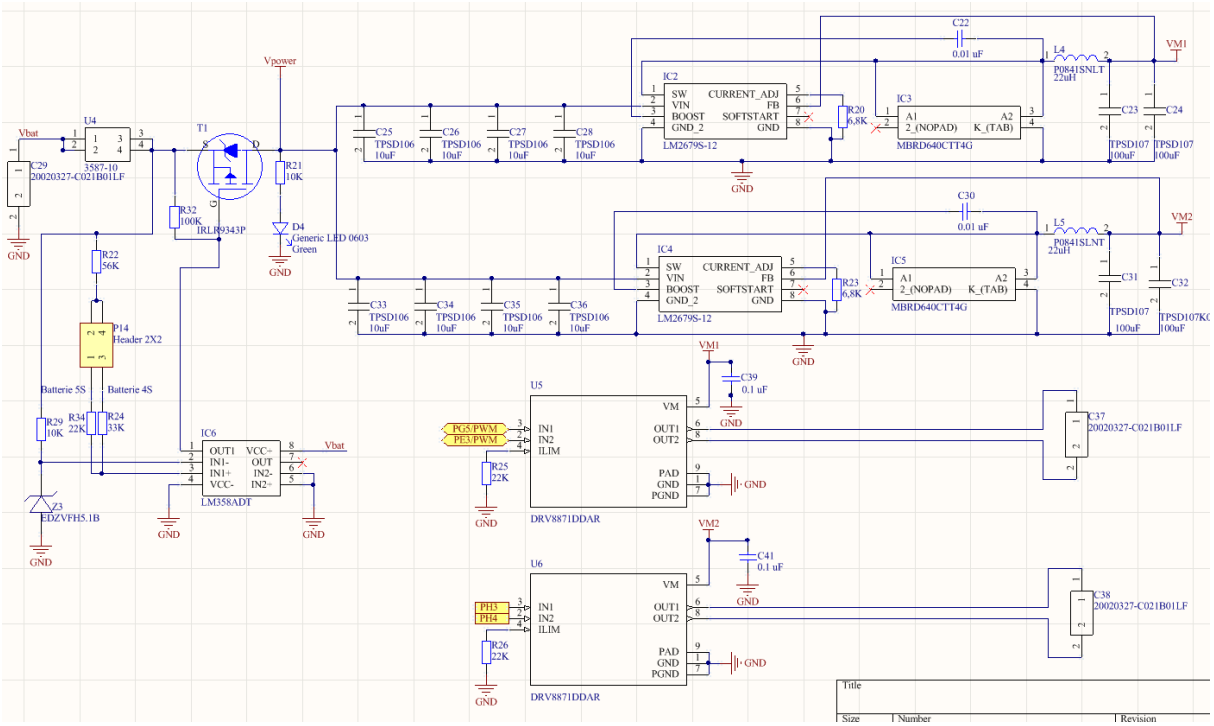


Figure 10 : Schéma de l'alimentation et des gestions des moteurs

Commençons par étudier la partie gauche de la carte. Nous pouvons y voir un MOSFET P, un fusible et un AOP monté en comparateur.

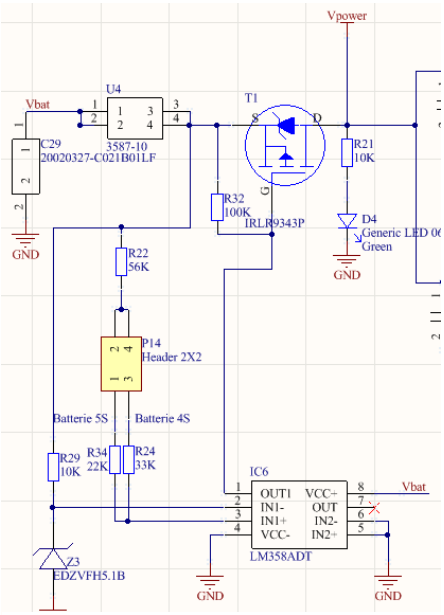


Figure 11:Gestion de la batterie

Nous voyons tout d'abord l'entrée de la batterie qui débouche sur un fusible automobile de 15A si jamais il y a un problème de court-circuit quelque part (pour que la batterie ne s'emballe pas).

Un comparateur avec une référence à 5V (donnée par une Zener) permet de commander le FET, ainsi quand la tension des cellules de la batterie passe sous 3.3V, le FET est coupé. Un indicateur lumineux signale l'alimentation ou non de la carte. Il faut bien sûr choisir convenablement le jumper qui utilise un pont diviseur pour la comparaison, sous peine de décharge profonde de la batterie (surtout en 4S). Ce montage fut testé sur une plaque de test et s'avère fonctionnel. Le FET coupe l'alimentation générale de la carte et supporte jusqu'à 20A.

Afin d'alimenter les moteurs, j'ai choisi d'utiliser de doubler deux modules de chez TI, le LM2679. Il s'agit de modules fixes 12V capable de débiter 5A chacun.

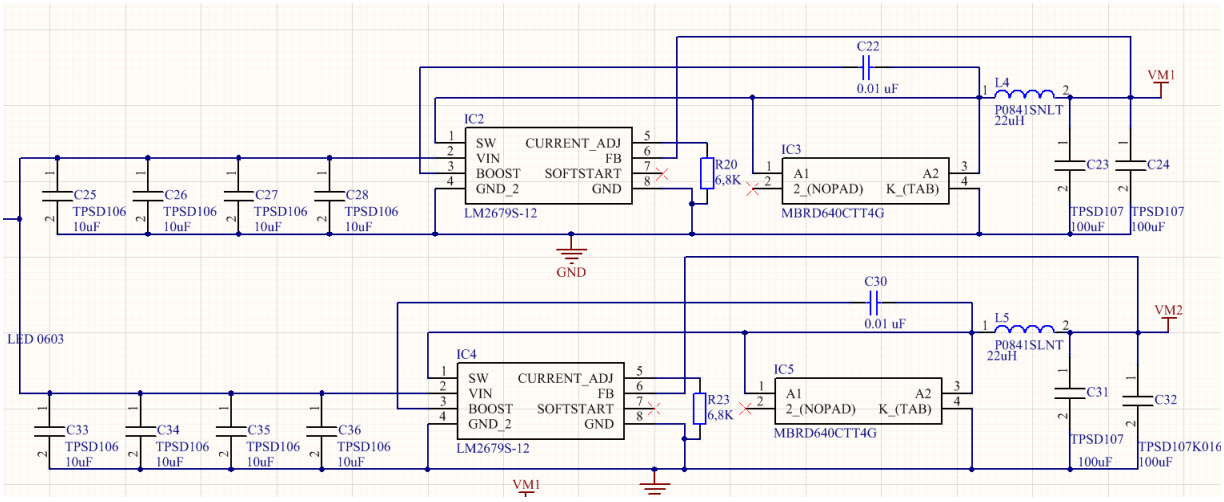


Figure 12: Alimentation des moteurs

J'avais tout d'abord imaginé la possibilité de choisir la tension de sortie en utilisant sa version ajustable, permettant d'alimenter soit à 6V soit à 12V. Cependant, lors de la conception selon la documentation technique de TI, il s'avère qu'il y avait trop de disparité entre les valeurs des composants à choisir (Bobine de 33uH ou de 22uH, des capacités de valeurs supérieures ou inférieures).

Plutôt que de fournir une alimentation qui soit anorexique ou à contrario sous stéroïde, j'ai préféré jouer la carte de la sécurité et n'utiliser qu'une seule tension. Finalement ce choix n'est pas gênant et nous allons voir par la suite pourquoi.

Le nombre (peut-être affolant) de capacité est justifié par la documentation technique du LM2679. Nous remarquerons l'immensité des bobines plus tard.

Il s'agit du second module le plus onéreux de ce projet (5.51€/pièce), après l'atmega2560. Il arrive en tête vu qu'il est nécessaire de le doubler pour obtenir un courant suffisant sur chacune des voies moteur.

Les moteurs de Robotech sont des moteurs à courant continu nécessitant au max 2.8A.

Pour gérer ces moteurs, j'ai fait le choix de deux modules DRV8871 qui acceptent jusqu'à 3.6A.

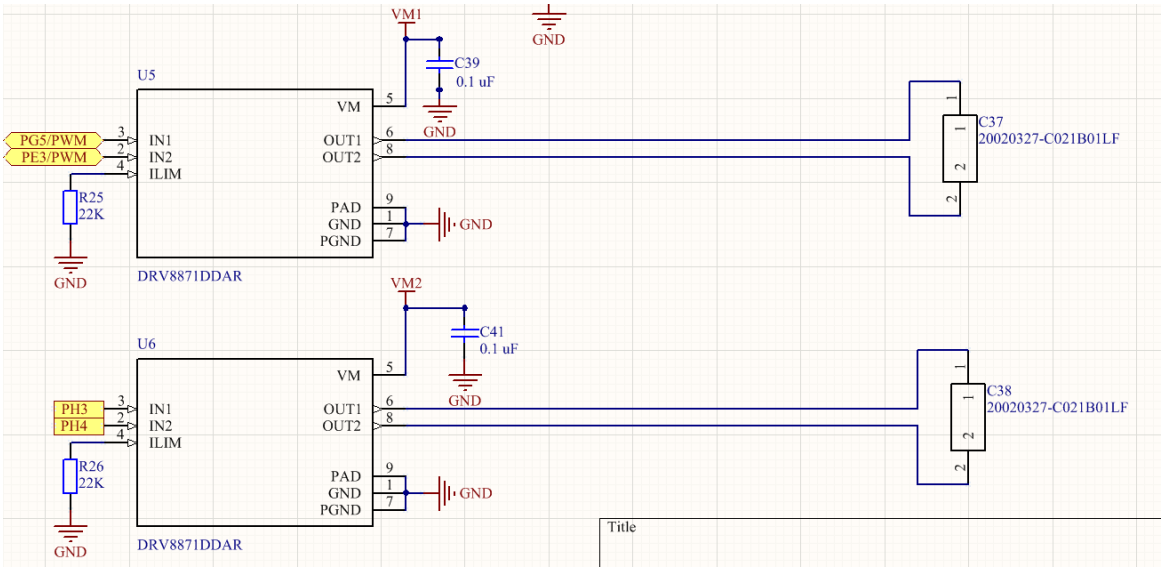


Figure 13 : Gestion des moteurs

Ces deux modules arrivent sur un bornier à vis comme pour la batterie. L'avantage de ces deux modules est leur mode de fonctionnement : il suffit d'envoyer un signal PWM pour générer les mouvements. En plus de permettre ainsi de gérer la tension et donc la vitesse des moteurs, cela permet de générer 4 types de mouvements :

Table 1. H-Bridge Control

IN1	IN2	OUT1	OUT2	DESCRIPTION
0	0	High-Z	High-Z	Coast; H-bridge disabled to High-Z (sleep entered after 1 ms)
0	1	L	H	Reverse (Current OUT2 → OUT1)
1	0	H	L	Forward (Current OUT1 → OUT2)
1	1	L	L	Brake; low-side slow decay

Figure 14 : Modes de fonctionnement des moteurs

Il est normalement nécessaire de rajouter une capacité supplémentaire afin de prévenir les pics de courant. Cependant, de par l'architecture d'alimentation choisie (plusieurs capacités de sorties) et le surdimensionnement des modules d'alimentation, elles ne m'ont pas semblé nécessaire.

E. Alimentation de la partie numérique

Pour alimenter les parties numériques, je me suis encore une fois orienté sur un module de chez TI, le LM2575 en version 5V et en version 3.3V, capable de débiter jusqu'à 1A. Comme l'ESP8266 a besoin d'être alimenté exclusivement en 3.3V, il était nécessaire de lui dédier un module supplémentaire lors de l'alimentation USB. J'ai décidé de partir sur un régulateur linéaire à faible perte capable de fournir jusqu'à 500mA.

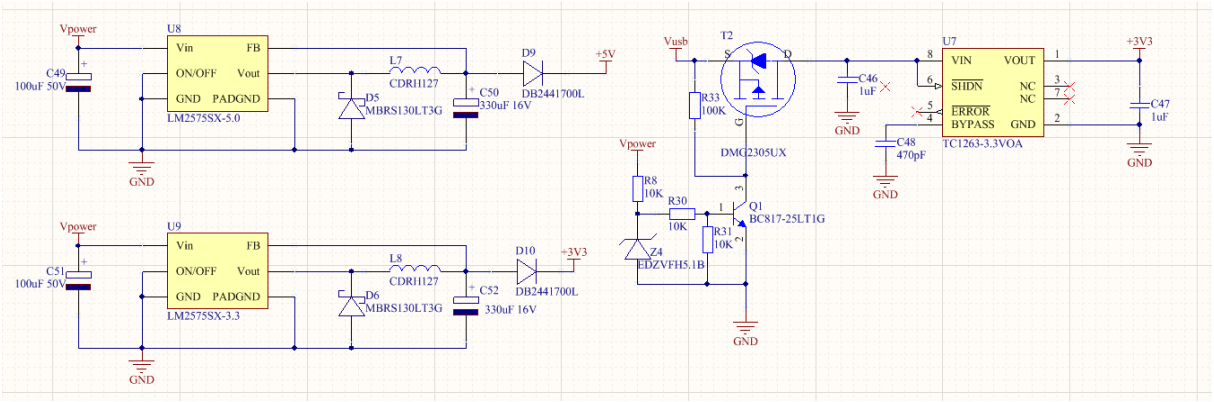


Figure 15: Alimentation de la partie numérique

On peut encore une fois remarqué la présence de diode pour empêcher un retour de courant dans les LM2575. Le circuit d'alimentation en 3.3V nécessite également un moyen d'être coupé lorsque la batterie est branchée. Pour cela j'utilise le même système que pour le comparateur, c'est-à-dire un MOSFET P commandé par un comparateur. J'ai fait cette fois le choix d'utiliser un comparateur à base de transistor bipolaire du fait de l'éloignement entre le premier AOP et cette fonction sur la carte.

Nous allons maintenant voir comment ces fonctions ont été placées sur le PCB.

2. Design du PCB

Cette partie sera bien plus rapide que la précédente, je ne ferai que pointer les différentes fonctions évoquées précédemment afin de pouvoir se repérer facilement sur la carte. Elle est d'environ

Tout d'abord, voilà la carte finalisée face par face :

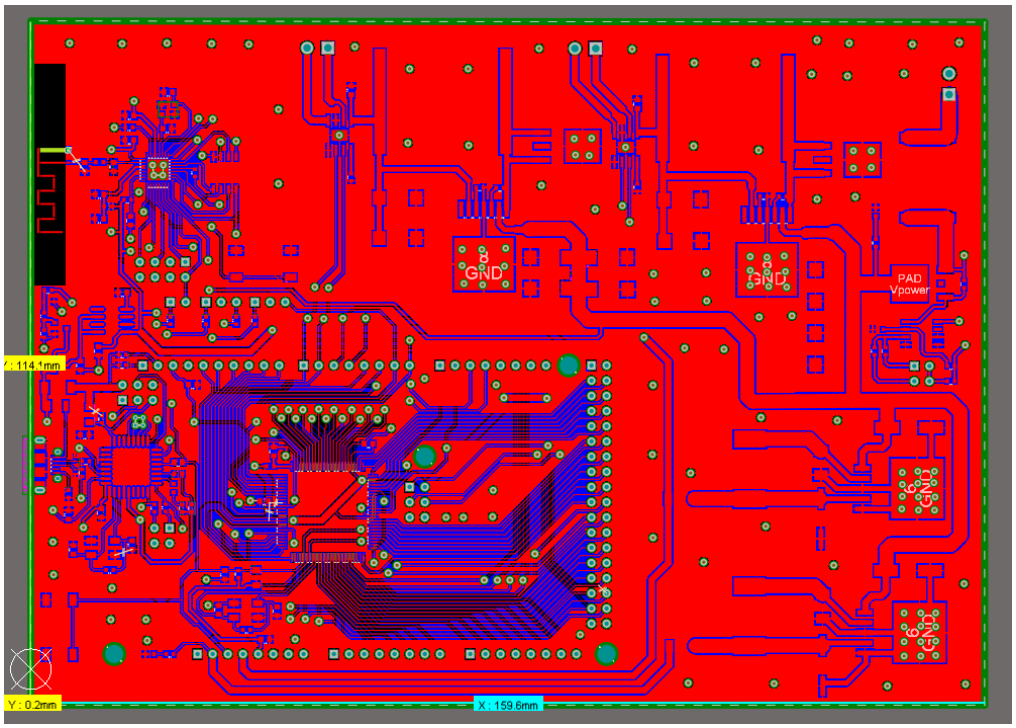


Figure 16:Face supérieure carte

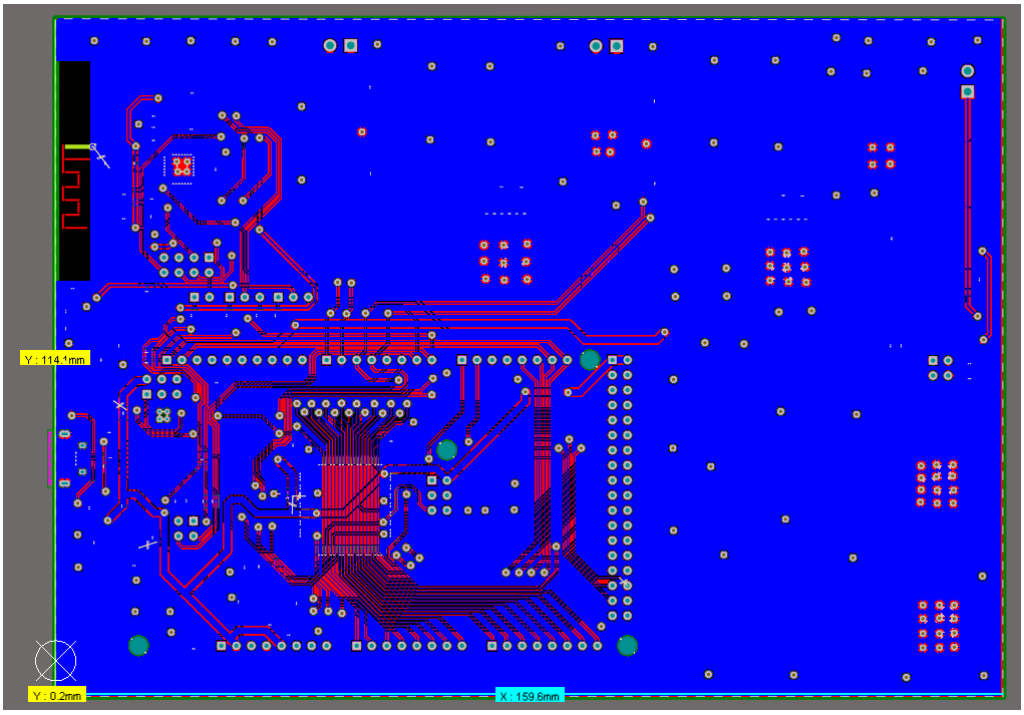


Figure 17: Face inférieure carte

On identifie tout de suite les parties numériques et les parties puissances de la carte par la présence de nombreux via et de nombreuses pistes dans la partie numérique.

Les dimensions sont plutôt importantes mais il est préférable que les composants de puissance puissent « respirer » afin d'éviter de surchauffer. Afin d'identifier plus facilement (si tenté que ce soit plus facile), j'ai retiré les plans de masses sur les deux faces. Nous noterons la présence de nombreux vias afin de mettre au même potentiel les deux plans de masses.

Reprenons l'ordre établi par la première partie :

A. L'Arduino Mega

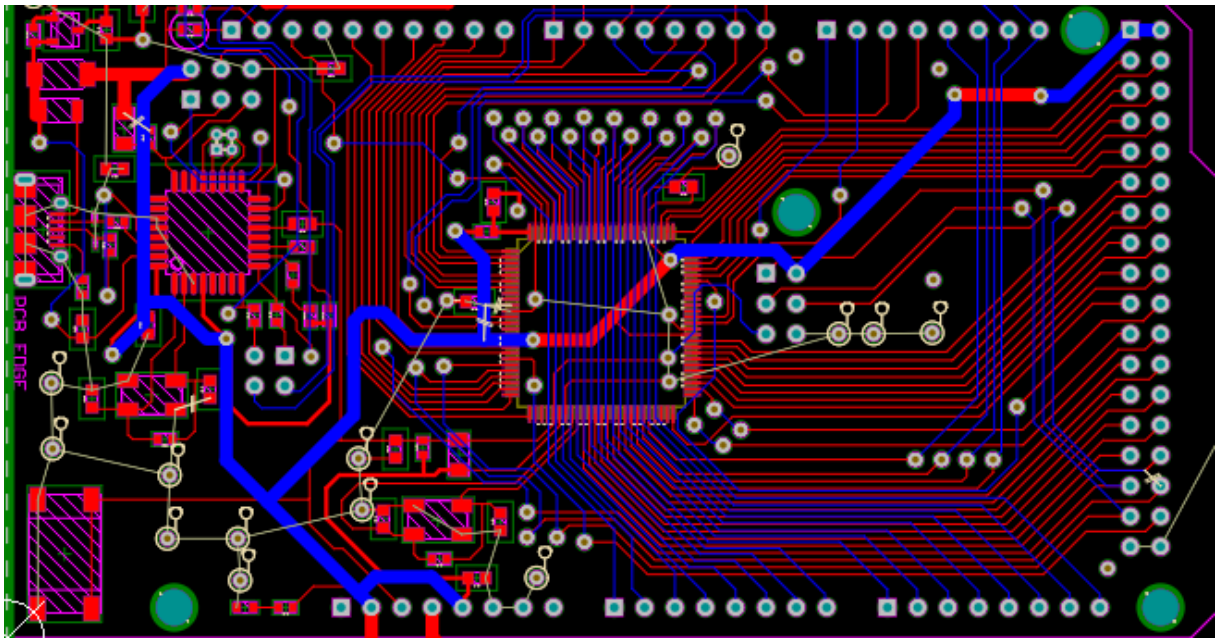


Figure 18: PCB Arduino Mega

Nous voyons ici la partie Arduino Mega. Rien d'exceptionnel, le bouton est mis en bas par faute de place. Si on compare à l'Arduino Mega rev 3, nous y verrons de nombreuses similitudes, comme annoncé.

B. L'ESP8266

J'ai choisi de le mettre côté USB afin de permettre à l'antenne de fonctionner correctement, nous remarquerons sur la partie basse le header 8-pins, le bouton et les différents jumpers permettant la gestion des communications.

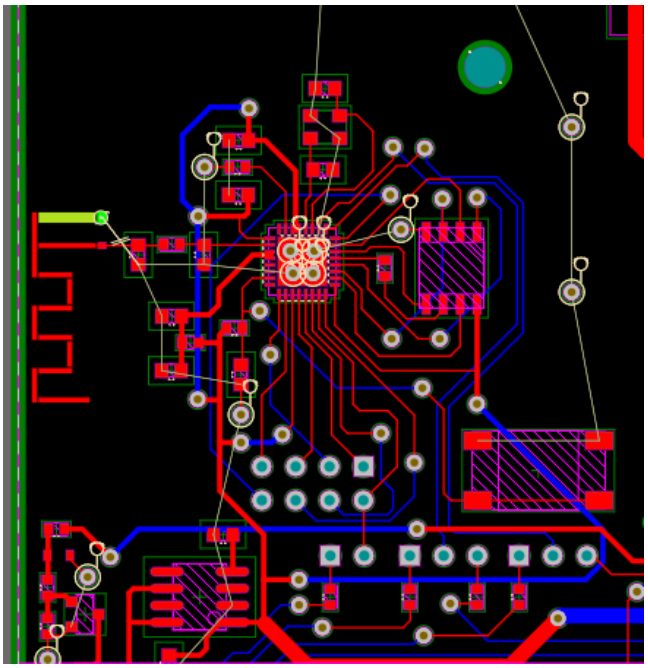


Figure 19: ESP8266

C. Gestion de la batterie

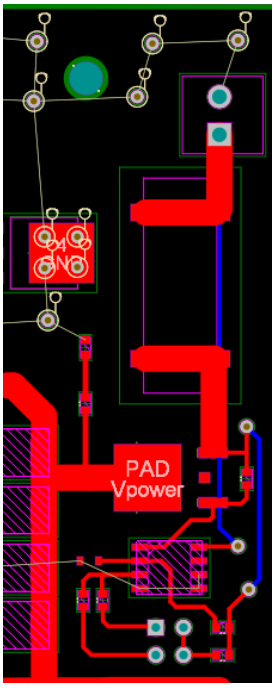


Figure 20: Gestion de la batterie

La gestion de la batterie se trouve sur la partie droite de la carte, nous y voyons bien entendu le bornier, le porte fusible, le MOSFET qui desservira toutes les alimentations de la carte ainsi que le comparateur. En bas nous voyons également le jumper qui permettra de sélectionner quelle batterie on utilise.

D. Alimentation et gestion des moteurs

Nous voyons sur la partie haute les modules d'alimentations des batteries avec les borniers permettant d'alimenter les moteurs. Comme dit précédemment, nous voyons la taille impressionnante des bobines, qui occupent la majeure partie de l'espace disponible. Nous remarquons également la faible taille des contrôleurs moteurs comparés aux modules d'alimentations (les chipsets qui sont reliés aux borniers).

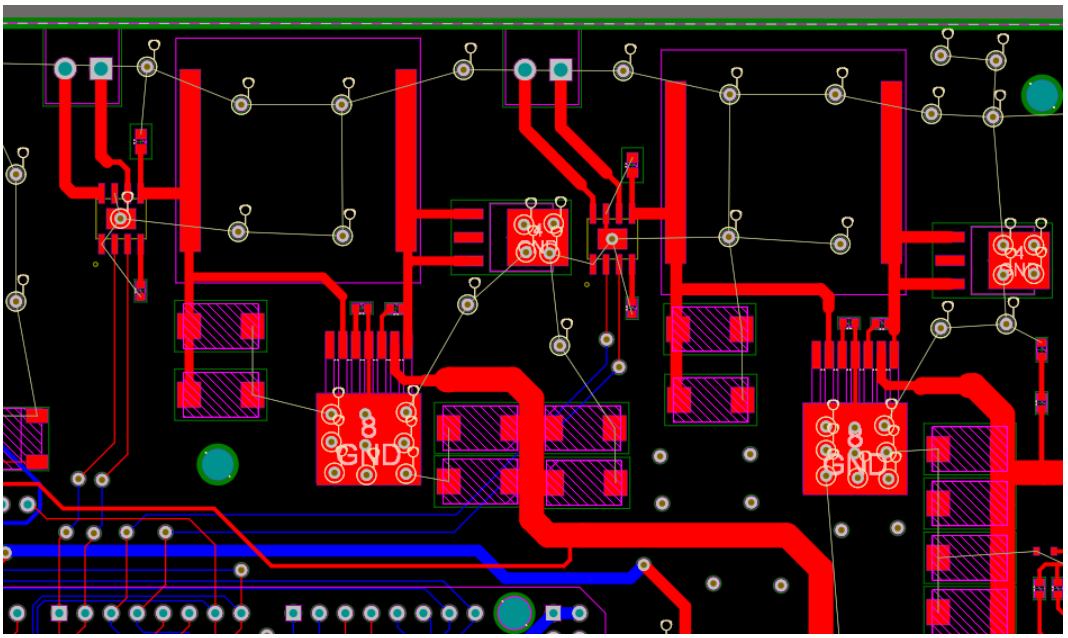
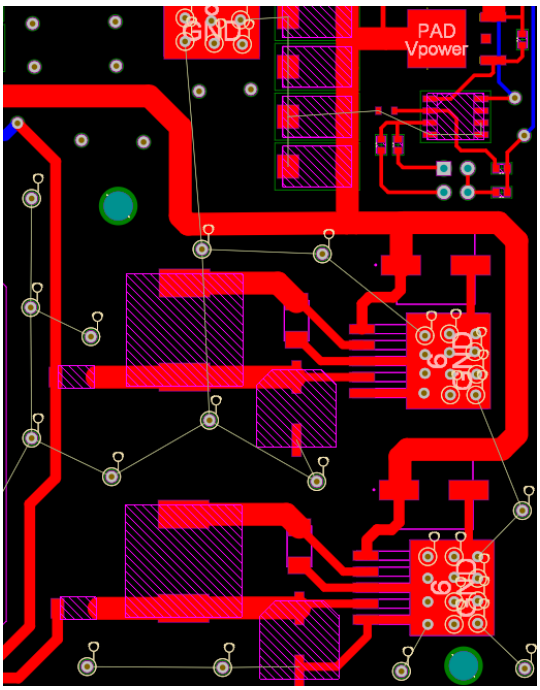


Figure 21: Alimentation des moteurs

E. Alimentation des modules numériques



Les modules 3.3V et 5V sont placés en haut et en bas respectivement. On remarque encore une fois que toutes les pistes d'alimentations proviennent du MOSFET en haut à droite.

Conclusion

La carte est entièrement réalisée, les composants sont disponibles et il est maintenant l'heure de chercher les éventuelles dernières erreurs de conception avant de produire la carte.

Nous avons vu les différentes fonctions ainsi que les valeurs de courant admissibles et les tensions utilisées sur la carte. Bien que très adaptées à contrôler des moteurs à courant continu, il sera impossible d'en contrôler d'autres types et sont très spécifiques aux moteurs utilisés par Robotech. Malgré tout, trouver des moteurs possédant des caractéristiques similaires est chose aisée.

Je n'ai malheureusement pas pu tester la carte in situ ni produire les différents rapports de tests que je comptais appliquer (notamment en terme de tensions acceptées et de courants maximums générés, ainsi que le fonctionnement correct du wifi, un des points les plus importants de la carte).

Il fut réellement difficile de réaliser une liste de matériel dans les temps, tant rechercher les composants et les adapter entre eux pour toute la partie puissance fut difficile. Comprendre la documentation technique de l'ESP8266 s'avéra plus compliquée que je ne l'imaginais et j'aurai peut-être dû utiliser une carte construite d'avance et l'adapter, mais je n'aurai pas eu la même flexibilité dans le placement et le choix des composants.

Cela fut d'autant plus compliqué car les séances chaque semaine me demandaient un peu de temps pour retrouver précisément la dernière chose dont je m'occupais.

Je pense par ailleurs que ce projet aurait été réalisé entièrement si nous avions été deux à travailler dessus.

Il est par contre possible de s'abstenir de la partie de contrôle des moteurs afin d'obtenir un Arduino Mega quasiment prêt à être créé d'après les contraintes de la graveuse PCB présente en C202.

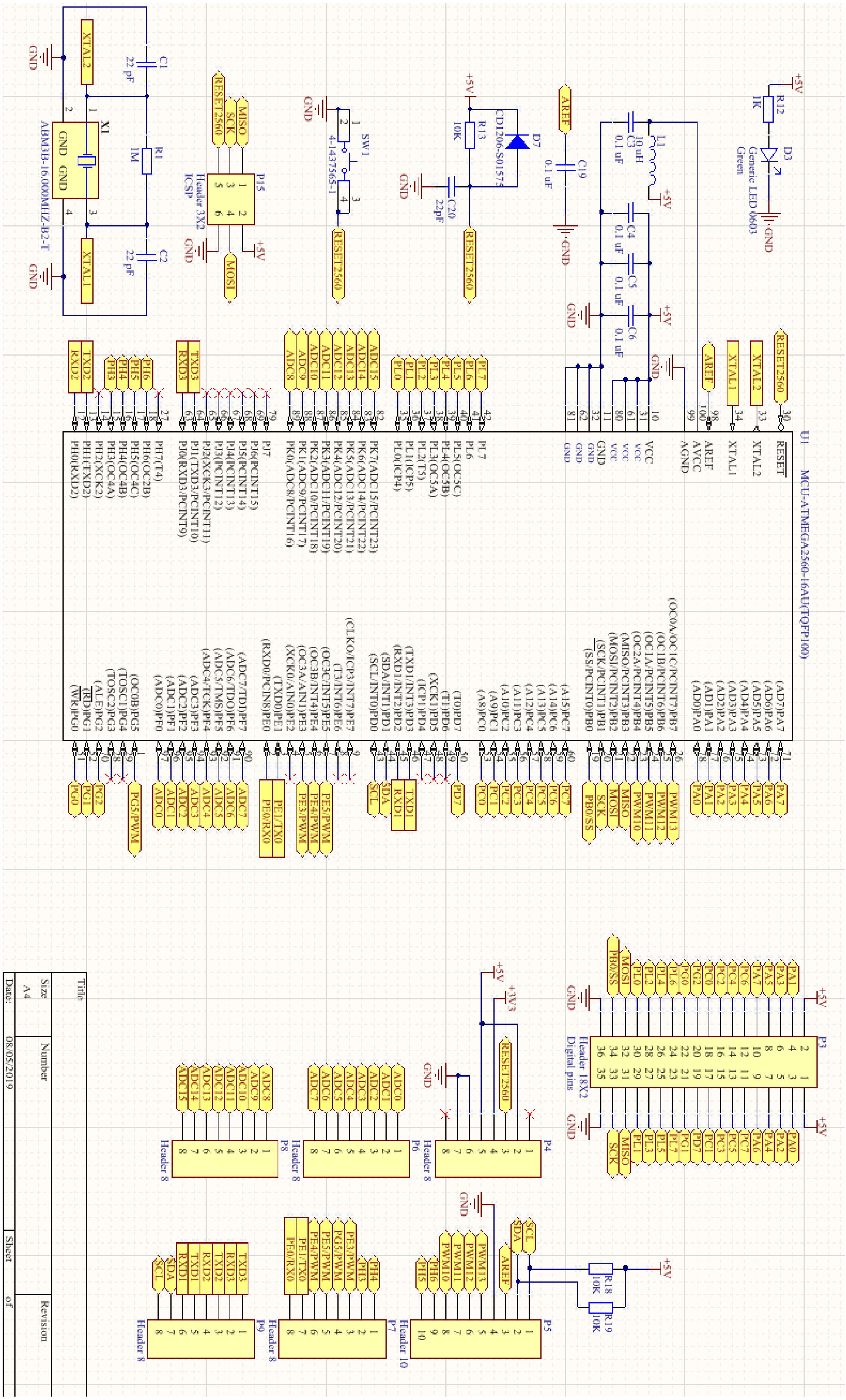
Finalement, le cahier des charges est respecté et nous avons ici une carte de qualité professionnelle par le choix des composants, qui seront robustes et accepteront de sortir légèrement de leur plage d'utilisation nominale, à l'exception faite des microcontrôleurs bien entendu.

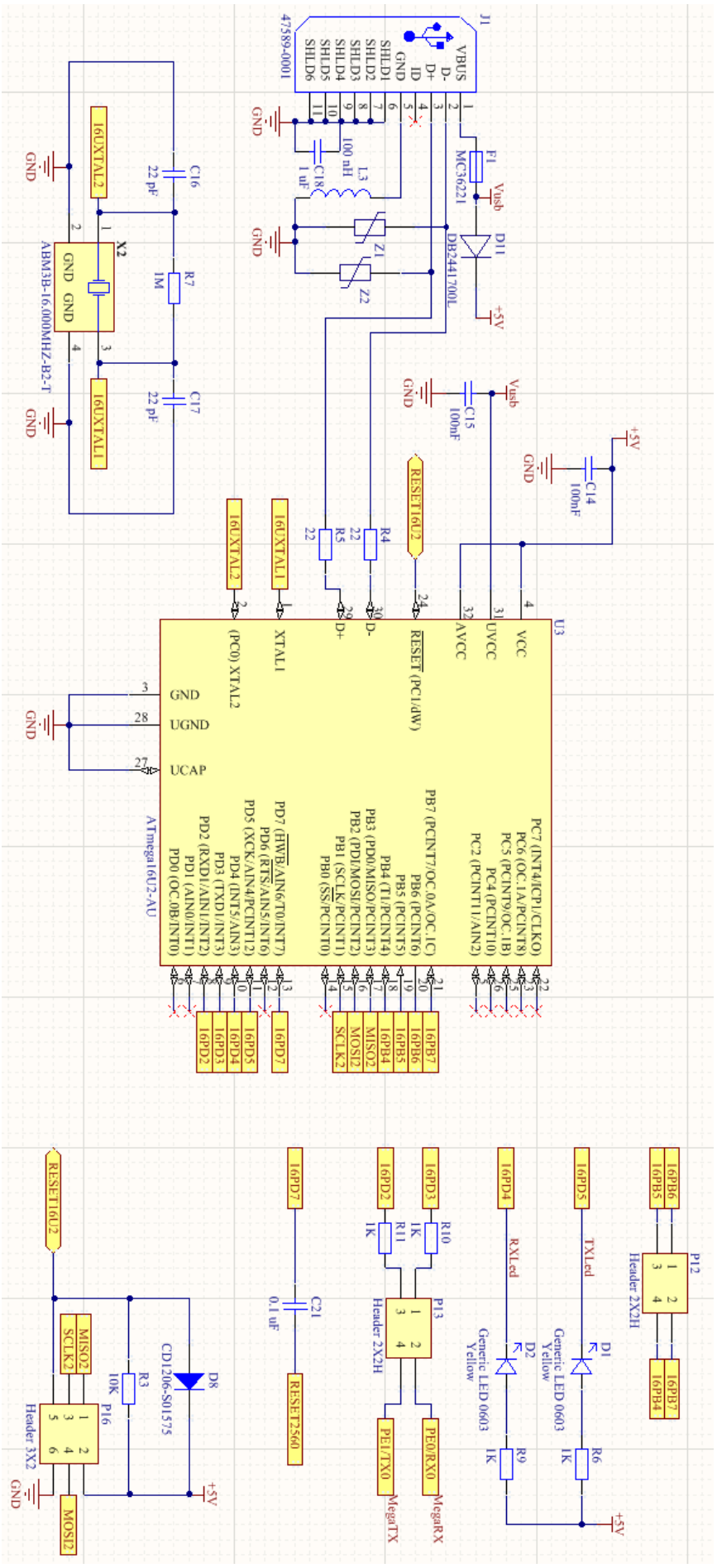
Annexes

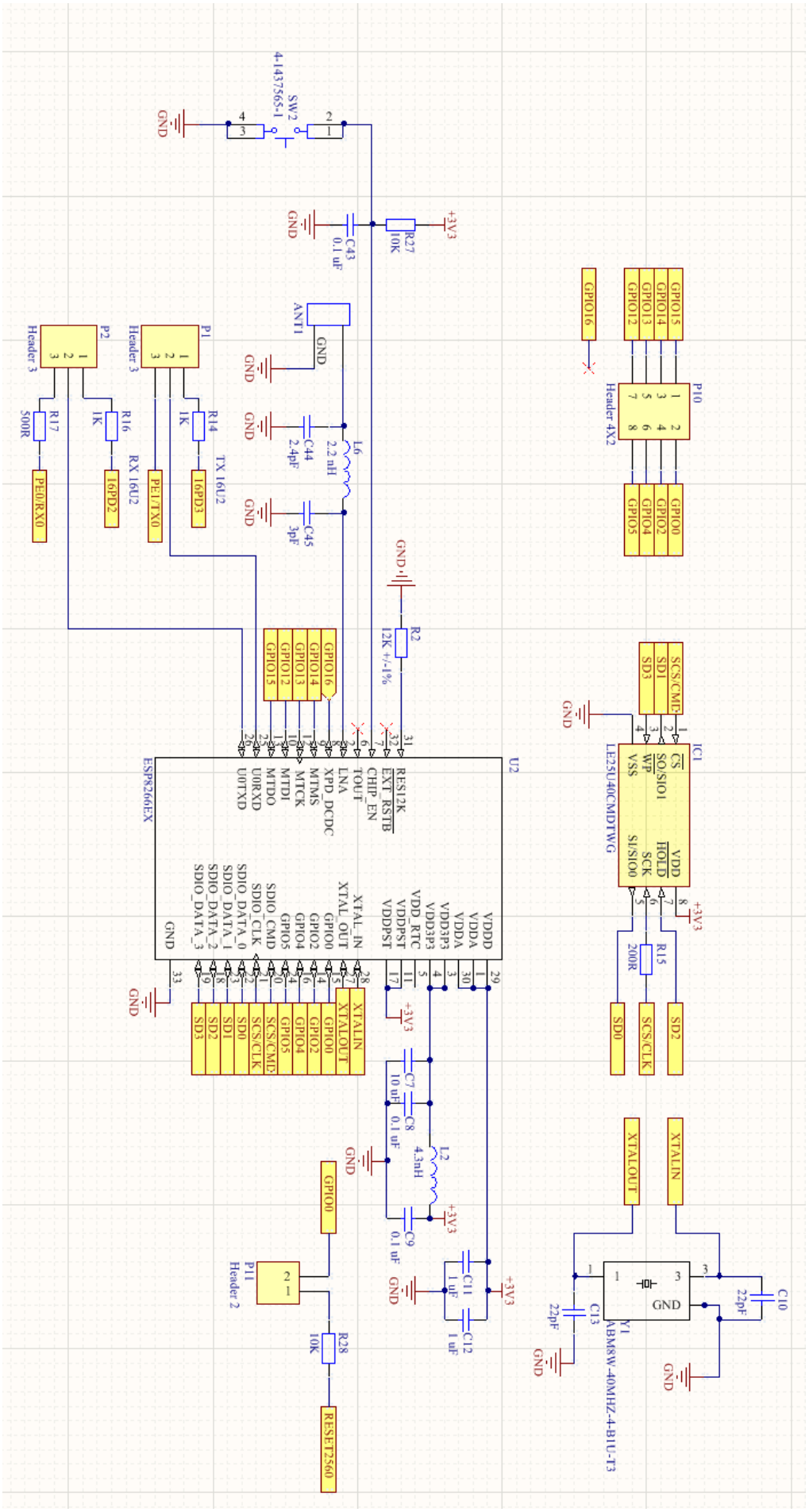
Nous trouverons dans cette partie les schémas agrandis.

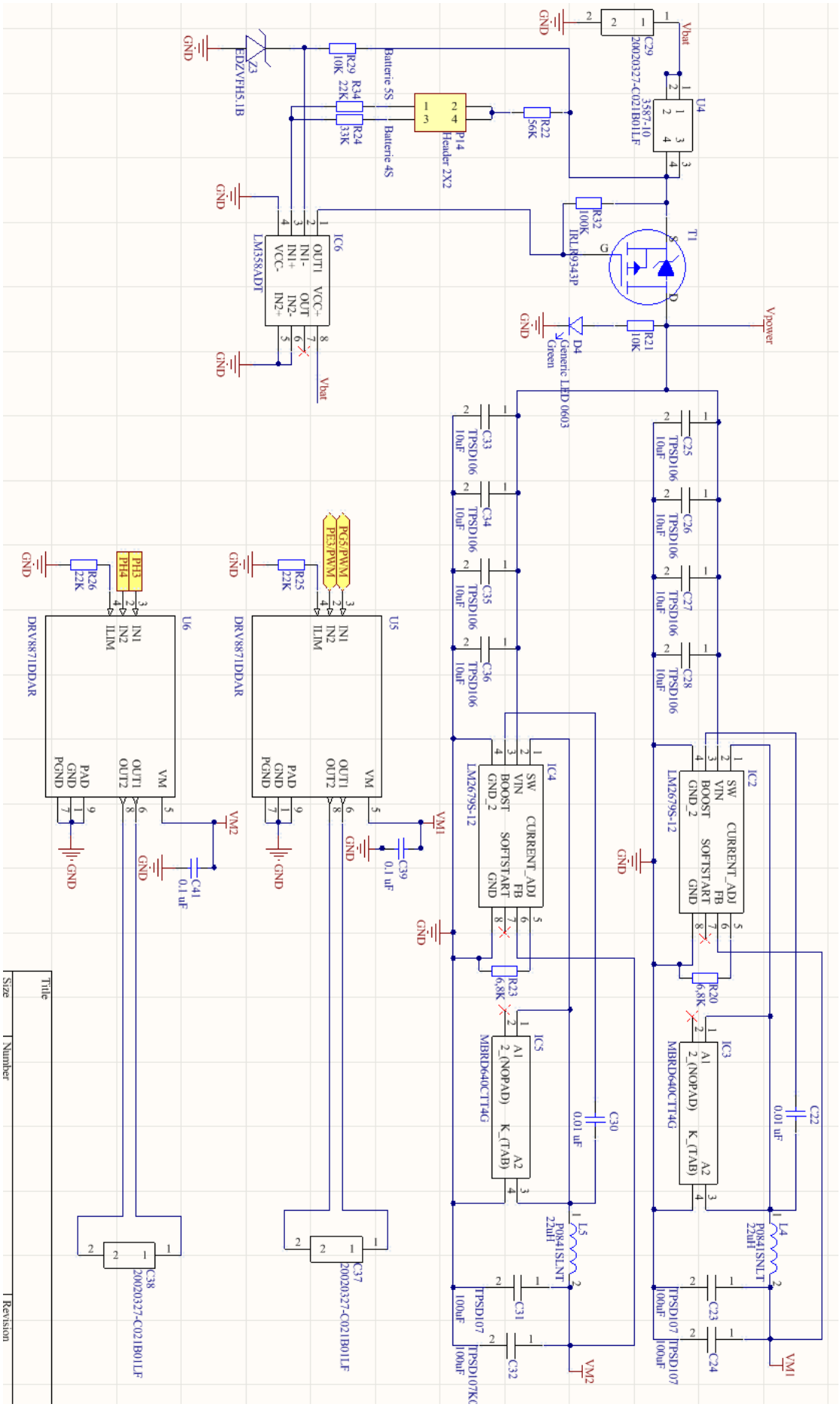
Dans l'ordre :

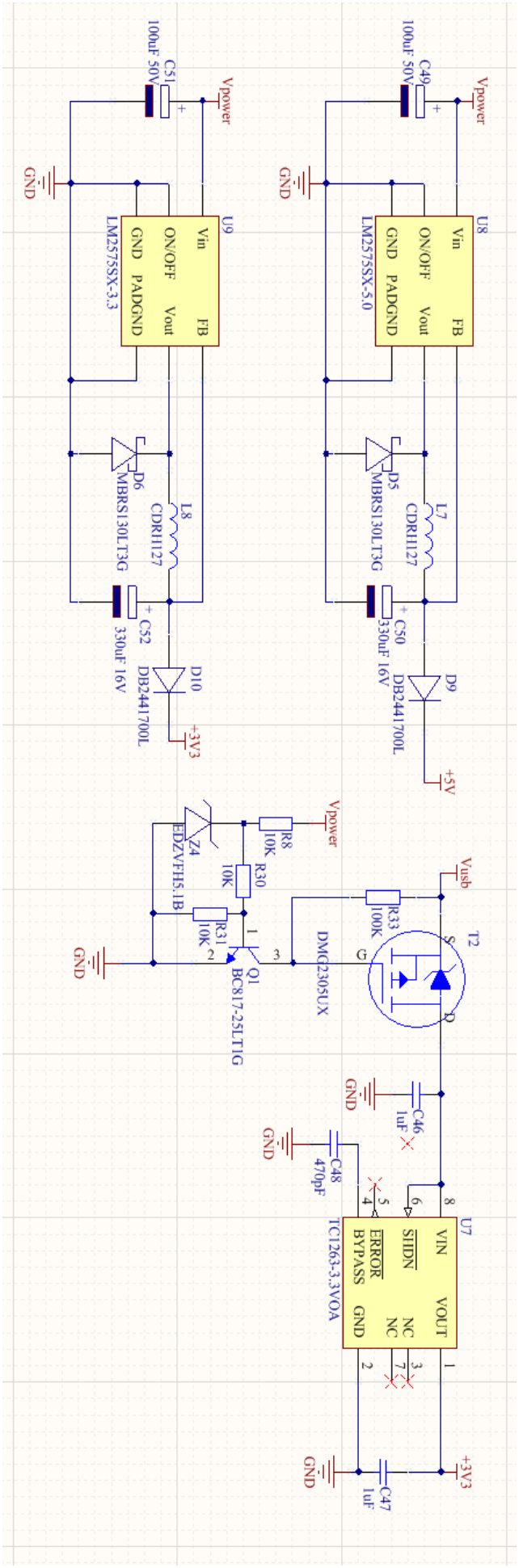
1. ATMEGA2560
2. ATMEGA16U2
3. ESP8266
4. Alimentation Moteurs + Gestion Batterie
5. Alimentation Circuit Numérique


MCU-ATMEGA2560-16AU(TQFP100)
U1
Header 18X2
Digital pins
ABM3B-16.000MHZ-B2-T
Generic LED 0603
Green
Header 3X2
ICSP
Title
Size
A4
Number
Revision
Date:
08/05/2019
Sheet
of


J1
47589-0001
VBUS
D-
D+
ID
GND
SHLD1
SHLD2
SHLD3
SHLD4
SHLD5
SHLD6
F1
MC36221
Vusb
D11
DB2441700L
+5V
Z1
Z2
L3
100 nH
C18
1 uF
C15
100nF
C14
100nF
RESET16U2
16UXTAL2
16UXTAL1
16UXTAL3
R5
22
R4
22
D-
D+
U3
ATmega16U2-AU
VCC
UVCC
AVCC
RESET (PC1/dW)
XTAL1
(PC0) XTAL2
GND
UGND
UCAP
PB7 (PCINT7/OC.0A/OC.1C)
PB6 (PCINT6)
PB5 (PCINT5)
PB4 (T1/PCINT4)
PB3 (PDO/MISO/PCINT3)
PB2 (PDI/MOSI/PCINT2)
PB1 (SCLK/PCINT1)
PB0 (SS/PCINT0)
PC7 (INT4/ICP1/CLKO)
PC6 (OC.1A/PCINT8)
PC5 (PCINT9/OC.1B)
PC4 (PCINT10)
PC2 (PCINT11/AIN2)
PD7 (HWB/AIN6/T0/INT7)
PD6 (RTS/AIN5/INT6)
PD5 (XCK/AIN4/PCINT12)
PD4 (INT5/AIN3)
PD3 (TXD1/INT3)
PD2 (RXD1/AIN1/INT2)
PD1 (AIN0/INT1)
PD0 (OC.0B/INT0)
16PD2
16PD3
16PD4
16PD5
16PD7
SCLK2
MISO2
MOSI2
16PB4
16PB5
16PB6
16PB7
X2
ABM3B-16.000MHZ-B2-T
C16
22 pF
C17
22 pF
R7
1M
C21
0.1 uF
R10
1K
R11
1K
P13
Header 2X2H
PE1/TXD
PE0/RXD
MegaTX
MegaRX
P12
Header 2X2H
D2
Generic LED 0603
Yellow
D1
Generic LED 0603
Yellow
RXLed
TXLed
R9
1K
R6
1K
P16
Header 3X2
R3
10K
D8
CD1206-S01575
RESET2560


SW2
4-1437565-1
GND
C43
0.1 uF
R27
10K
+3V3
ANT1
GND
C44
2.4pF
L6
2.2 nH
C45
3pF
P1
Header 3
P2
Header 3
R14
1K
TX 16U2
R16
1K
RX 16U2
R17
500R
16PD3
16PD2
PE1/TX0
PE0/RX0
R2
12K +/-1%
U2
ESP8266EX
RESL2K
EXT_RSTB
CHIP_EN
TOUT
LNA
XPD_DCDC
MTMS
MTCK
MTDI
MTDO
U0RXD
U0TXD
VDDD
VDDA
VDDA
VDD3P3
VDD3P3
VDD_RTC
VDDPST
VDDPST
XTAL_IN
XTAL_OUT
GPIO0
GPIO2
GPIO4
GPIO5
SDIO_CMD
SDIO_CLK
SDIO_DATA_0
SDIO_DATA_1
SDIO_DATA_2
SDIO_DATA_3
GND
XTALIN
XTALOUT
SD3
SD2
SD1
SD0
SCS/CLK
SCS/CMD
GPIO5
GPIO4
GPIO2
GPIO0
C7
10 uF
C8
0.1 uF
L2
4.3nH
C9
0.1 uF
C11
1 uF
C12
1 uF
P11
Header 2
R28
10K
RESET2560
GPIO16
GPIO15
GPIO14
GPIO13
GPIO12
GPIO5
GPIO4
GPIO2
GPIO0
P10
Header 4X2
IC1
LE25U40CMDTWG
CS
SO/SIO1
WP
VSS
VDD
HOLD
SCK
SI/SIO0
SCS/CMD
SD1
SD3
R15
200R
SD0
SCS/CLK
SD2
Y1
ABM8W-40MHZ-4-B1U-T3
C10
22pF
C13
22pF


Vbat
Vpower
GND
Batterie 5S
Batterie 4S
Header 2X2
Generic LED 0603
LM358ADT
IRLR9343P
LM2679S-12
MBRD640CTT4G
DRV8871DDAR
PG5/PWM
PE5/PWM
PH3
PH4
VM1
VM2
Title
Size
Number
Revision


C49
100uF 50V
Vpower
GND
U8
Vin
ON/OFF
Vout
FB
GND
PADGND
LM2575SX-5.0
D5
MBRS130LT3G
L7
CDRH127
C50
330uF 16V
D9
DB2441700L
+5V
C51
100uF 50V
Vpower
GND
U9
Vin
ON/OFF
Vout
FB
GND
PADGND
LM2575SX-3.3
D6
MBRS130LT3G
L8
CDRH127
C52
330uF 16V
D10
DB2441700L
+3V3
Vpower
R8
10K
R30
10K
R31
10K
Z4
EDZVFH5.1B
Q1
BC817-25LT1G
GND
Vusb
R33
100K
T2
DMG2305UX
S
G
D
GND
C46
1uF
GND
C48
470pF
U7
TC1263-3.3VOA
VIN
SHDN
ERROR
BYPASS
GND
NC
NC
VOUT
GND
C47
1uF
+3V3