

Diplôme de technicien ES en électronique - 2019

Astrophotography Assistant



Candidat : Dietrich Tanguy

Professeurs encadrants : Christian Humpert

Table des matières

1.	Sujet.....	4
2.	Introduction	5
2.1.	Quelque notion d'astronomie	5
2.1.1.	<i>Le mouvement du ciel (Mouvement diurne)</i>	5
2.1.2.	<i>Les montures</i>	6
2.1.2.1.	<i>Les montures azimutales.....</i>	6
2.1.2.2.	<i>Les montures équatoriales</i>	6
2.1.3.	<i>La mise en station :.....</i>	6
2.2.	Description du projet.....	8
2.3.	Photos du produit fini.....	8
3.	Analyse globale	9
3.1.	Analyse fonctionnelle	9
3.1.1.	<i>Diagramme FAST.....</i>	10
3.2.	Schéma bloc général.....	11
3.2.1.	<i>Télescope avant.....</i>	11
3.2.2.	<i>Télescope après</i>	12
3.3.	Schéma Bloc détaillé.....	13
3.4.	Brainstorming.....	14
4.	Conception et développement	15
4.1.	Conception et développement matériel	15
4.1.1.	<i>Interface ST4.....</i>	18
4.1.2.	<i>Alimentation et Raspberry Pi.....</i>	21
4.1.3.	<i>Carte de température</i>	22
4.1.4.	<i>Consommation</i>	27
4.2.	Conception et développement logiciel	28
4.2.1	<i>Développement du code de gestion de moteur.....</i>	28
4.2.1.1.	<i>Calcul de la vitesse des moteurs.....</i>	28
4.2.1.2.	<i>Analyse du code de gestion de moteur</i>	36
4.2.2.	<i>Programmation carte température.....</i>	40
4.2.2.1.	<i>Analyse de la carte de gestion de température</i>	43
4.2.3.	<i>Protocole de communication.....</i>	47
4.3.	PHD2 Guiding	49
4.4.	Télescope Goto.....	50
4.5.	Interface Web.....	52
4.5.1.	<i>HTML – JavaScript – Ajax - Python</i>	54
4.5.2.	<i>Installation de la Raspberry Pi.....</i>	56
4.5.2.1.	<i>VNC</i>	56
4.5.2.2.	<i>Écran</i>	56

4.5.2.3. <i>Port série</i>	56
4.5.2.4. <i>Clavier virtuelle</i>	57
4.5.2.5. <i>Librairies Python</i>	57
4.5.2.6. <i>Script Python au démarrage</i>	57
4.5.2.7. <i>PHD2</i>	58
4.6. Tests fonctionnels	59
5. Conclusion	61
5.1. Aspects techniques sur le projet.....	61
5.2. Aspects gestions sur le projet	62
5.3. Aspects personnels.....	62
6. Remerciements	62
7. Outils de développement utilisés	63
8. Bibliographique et sites internet	63
9. Annexes	64
9.1. Mode d'emploi et d'installation	64
9.1.1. <i>Interface Physique</i>	66
9.1.2. <i>Démarrage des applications</i>	71
9.1.3. <i>Page Web</i>	72
9.1.4. <i>PHD2</i>	73
9.2. Liste des source et pourcentage d'apport	75
9.3. Planning.....	76
9.4. Schéma.....	81
9.5. Dossier de fabrication des circuits imprimés	88
9.6. Listing des codes sources.....	118-175

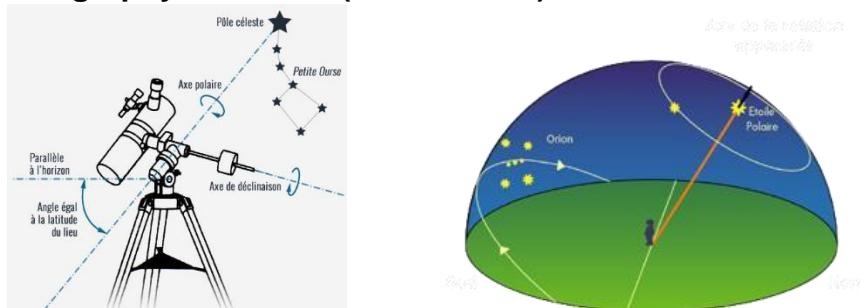
1. Sujet

Diplôme de technicien ES en électronique - 2019

Candidat : **Tanguy Dietrich**

Sujet

AstroPhotography Assistant (Star Tracker)



Description

Réalisation d'un dispositif permettant la photographie des astres au travers de l'objectif d'un télescope.

Le dispositif est composé :

- D'une mini station météo et d'un corps de chauffe pour lutter contre la rosée sur l'objectif.
- D'un asservissement motorisé permettant le suivi des astres.
- D'une interface web permettant de piloter l'équipement.
-

Travail

Gestion du projet (planning, ressources)

Pré-étude du système (analyse du besoin, analyse fonctionnelle, FAST)

Étude et réalisation

Développement des logiciels

Validation du dispositif complet

Documentation

À remettre un mémoire sous forme dactylographiée (couverture en blanc) ainsi qu'une version PDF, comprenant :

- Une description technique complète, avec la motivation des choix effectués, les schémas et les calculs des composants.
- Une description des structures de données et du protocole de communication
- Les descriptions, organigrammes et "listings" des programmes.
- Les résultats des essais et mesures effectuées. - Les caractéristiques des composants spéciaux.

Le montage du dispositif, mécanique et électronique.

Les logiciels, schémas, organigrammes, etc., sur support informatique.

Humpert Christian

2. Introduction

2.1. Quelque notion d'astronomie

2.1.1. Le mouvement du ciel (Mouvement diurne)

Le mouvement diurne est le résultat de la rotation de la Terre sur elle-même. Ainsi, le soleil se lève à l'Est et se couche à l'Ouest. L'axe de rotation de la Terre correspond à la ligne que l'on peut tirer entre le pôle Sud et le pôle Nord. Comme on peut le voir (cf. figure 1), l'étoile Polaire se trouve presque sur la prolongation de l'axe de rotation. C'est pour cela qu'elle paraît immobile tout au long de la nuit.

Il est possible d'observer le mouvement diurne en faisant plusieurs images avec de longues poses comme un timelapse des étoiles du ciel. Ces

images sont appelées des circumpolaires.

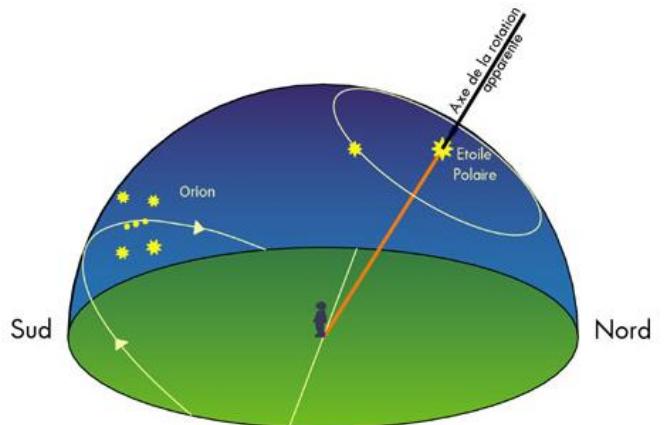


Figure 1 Représentation des constellations vue de la terre



Figure 2 TimeLapse qui représente le mouvement diurne

2.1.2. Les montures

Pour observer le ciel, on utilise des télescopes qui sont posés sur des montures. Il y a différentes montures : chacune avec leurs avantages et leurs inconvénients.

2.1.2.1. Les montures azimutales

La monture azimutale est l'un des modèles les plus simples. Elle permet de déplacer le télescope en azimut (de gauche à droite) et en élévation (de haut en bas) et de l'aligner sur n'importe quel objet désiré.

Ces modèles ont l'avantage d'être très faciles à transporter et ne nécessitent pas de connaissances particulières concernant leur utilisation. Par contre, pour observer les objets célestes, il faut constamment corriger la position du télescope en agissant sur les deux axes.

2.1.2.2. Les montures équatoriales

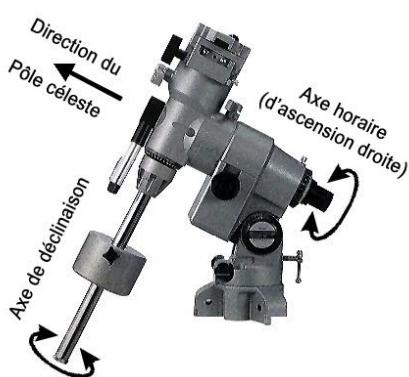


Figure 3 Mécanisme d'une monture équatoriale

Les montures équatoriales sont un peu plus complexes et要求 une mise en station.

La monture équatoriale utilise l'axe de rotation de la Terre. L'étoile Polaire est souvent utilisée comme point de repère grâce au fait qu'elle se trouve quasiment sur l'axe de rotation de la Terre. L'avantage de cette monture est que si la mise en station est bien faite, un seul axe suffit pour suivre l'objet visé.

Le fait qu'un seul axe suffise pour suivre le ciel est très important pour la suite. Car c'est ce type de monture que j'utilise dans le projet.

2.1.3. La mise en station :

La mise en station consiste à aligner l'axe d'ascension (Polaire) de la monture sur l'axe de rotation de la Terre.

Si les deux axes sont parfaitement alignés et bougent à la même vitesse, alors l'objet visé par le télescope bougera en même temps que la monture. Ils seront donc immobiles l'un par rapport à l'autre. (cf. figure 5)

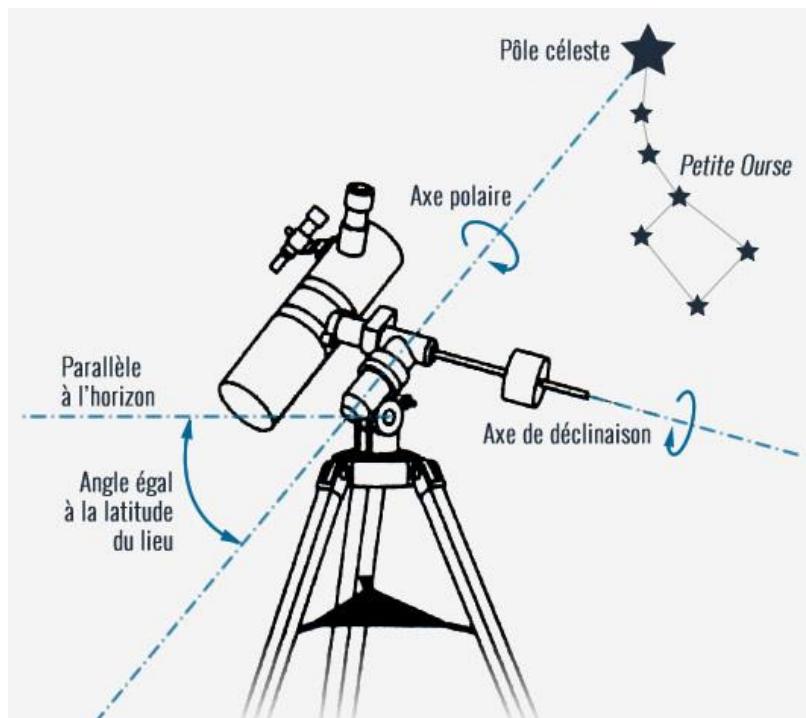


Figure 4 Mise en station d'un télescope newton avec une monture équatoriale

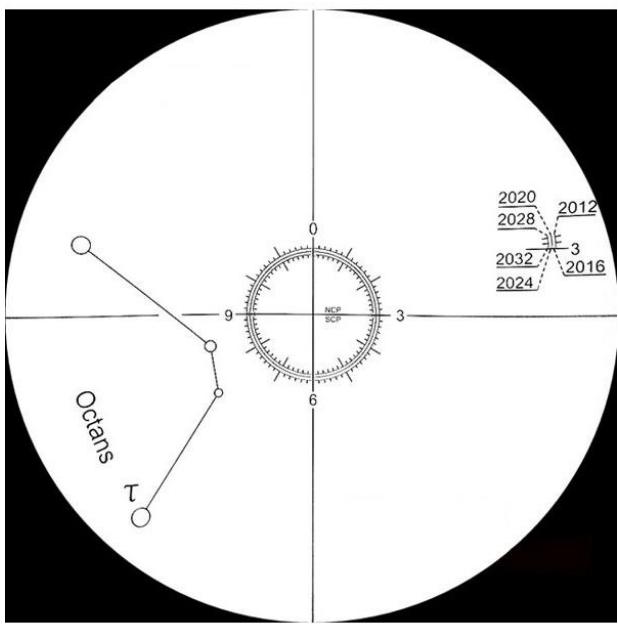


Figure 5 Vue à travers un viseur polaire

Certaines montures équatoriales sont équipées de viseurs polaires. Ces viseurs sont là pour aider à faire la mise en station. C'est une sorte de mini loupe incrustée dans la monture.

Comme mentionné précédemment, l'étoile Polaire est très proche de l'axe de rotation de la Terre ce qui permet de la repérer.

La figure 6 représente ce que l'on observe en regardant dans un viseur polaire.

Lors d'une mise en station, l'étoile Polaire devrait être placée quelque part sur les cercles au centre suivant l'heure d'observation et de l'année.

Ensuite, durant la nuit, si la mise en station est bien faite, elle devrait suivre les cercles et ne jamais les quitter.

Le temps de pose en photographie :

Le temps de pose définit le temps durant laquelle l'appareil photo va capter de la lumière. Ce paramètre est important en photographie, car un temps de pose trop long peut générer une image floue si l'objet photographié bouge.



Bus flou derrière une cabine téléphonique nette

Comme dans cette image où la cabine téléphonique est nette, mais que le bus s'est déplacé pendant le temps de pose. Le même phénomène se produit sur la photo circumpolaire, à la page 5.

Une mise station n'est jamais parfaite, donc il est impossible de poser très longtemps sans qu'un flou de mouvement apparaissent. C'est pour cela qu'une simple mise en station ne suffit pas pour faire de l'astrophotographie.

Donc il est possible d'utiliser une caméra de guidage pour réguler la position de l'étoile dans le champ, et augmenter le temps de pose.

Mais il est nécessaire d'avoir un ordinateur pour contrôler la caméra de guidage.

2.2. Description du projet

Le but de projet est de refaire toute l'électronique de gestion des moteurs d'un télescope de type NEQ5 (monture équatoriale). Et d'ajouter un système de régulation de température afin d'éviter l'apparition de condensation sur le miroir du télescope en hiver.

Mais aussi d'y incorporer une Raspberry pi, car actuellement pour avoir un suivi précis des étoiles pour faire de la photographie, il est nécessaire d'avoir un PC qui contrôle une caméra de suivi.

Le télescope ne sera pas constamment utilisé pour faire de la photographie, donc la Raspberry pi n'est pas toujours nécessaire. Par exemple si je décide de faire de l'observation visuelle, la Raspberry pi sera éteinte et le suivi du ciel se fera à l'aide du microcontrôleur afin d'économiser de l'énergie.

Donc tout le système doit pouvoir fonctionner sans la Raspberry PI.

Pour résumer, il y aura deux types de suivi du ciel :

- Mode observation, économique en énergie, mais moins précis.
- Mode photographique, gourmand en énergie, mais très précis.

Il y aura aussi plusieurs modes pour la régulation de température :

- Manuel, en utilisant des potentiomètres (Hardware)
- Manuel, en utilisant l'interface Web ou l'écran
- Automatique, par régulation PID en calculant le point de rose avec des capteurs sur les lentilles, et un capteur d'humidité

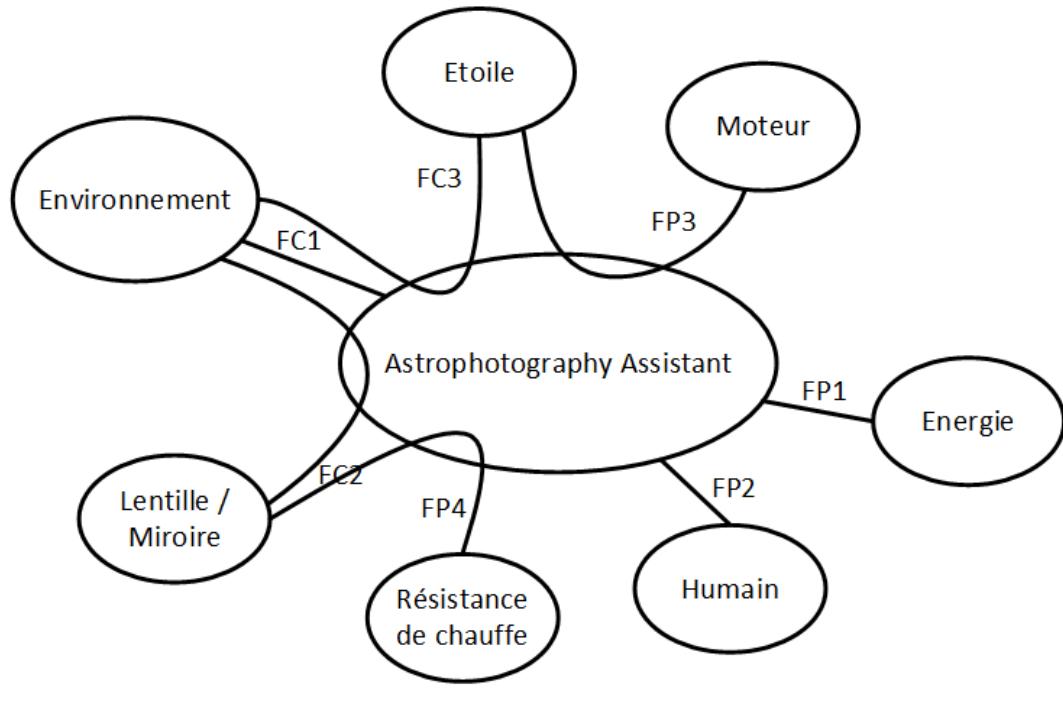
2.3. Photos du produit fini



3. Analyse globale

3.1. Analyse fonctionnelle

Représentation sous forme de diagramme *FAST (Functional Analysis System Technique)*.



FP1 : Batterie 12V / 5V

FP2 : Interface

FP3 : Suivie des astres

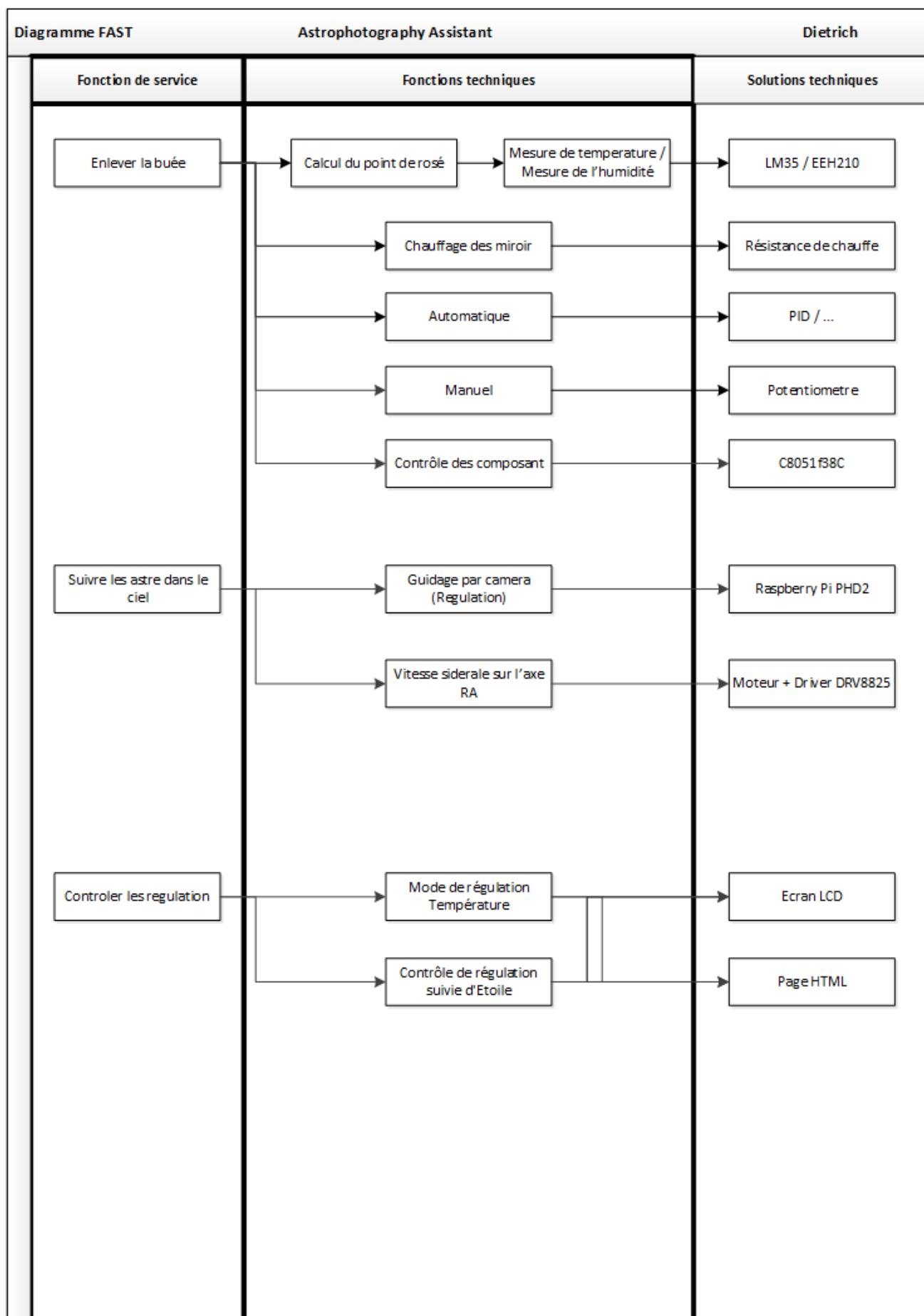
FP4 : régulation Anti-condensation

FC1 : Météo

FC2 : Température / Humidités

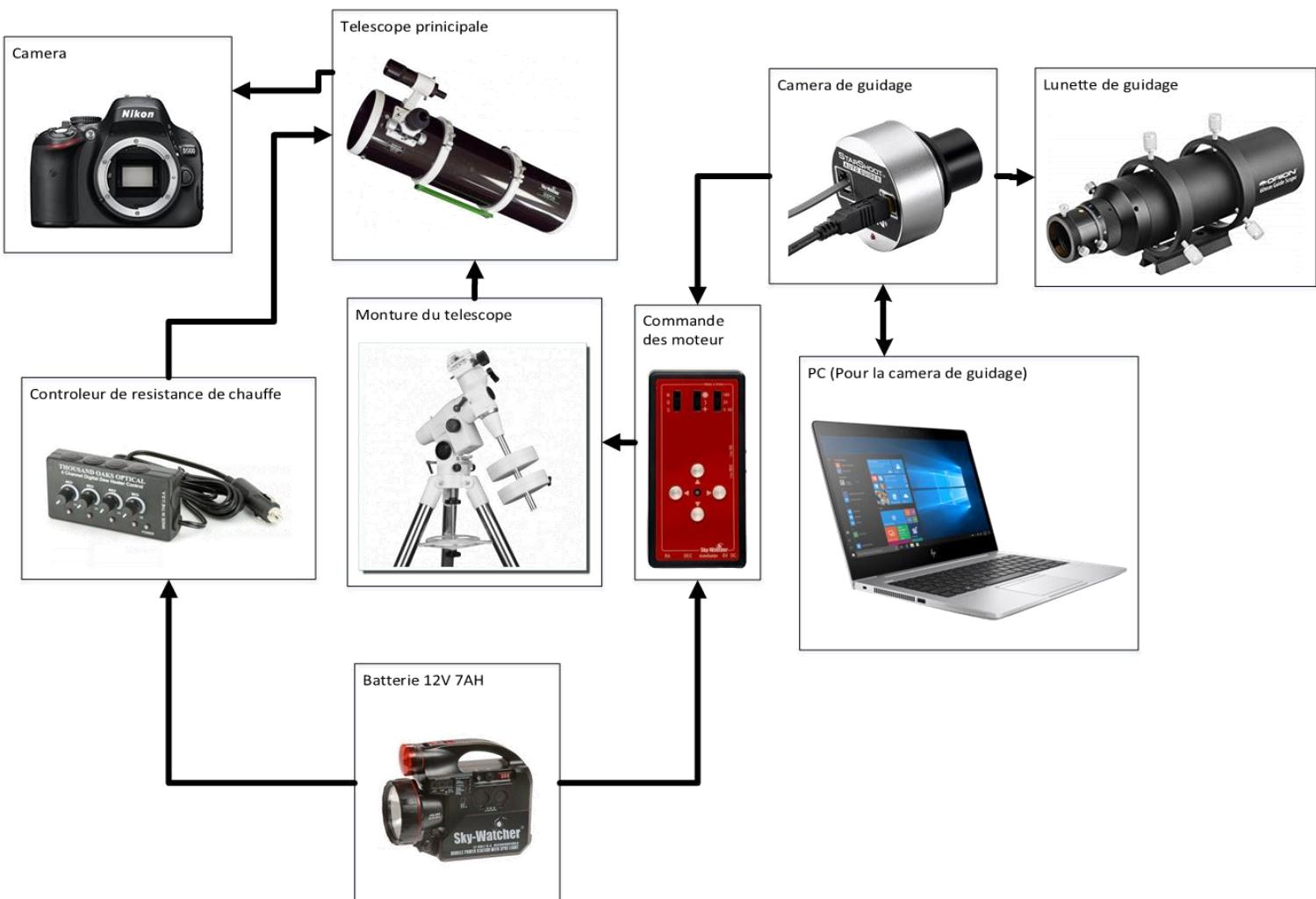
FC3 : Mouvement du ciel

3.1.1. Diagramme FAST



3.2. Schéma bloc général

3.2.1. Télescope avant

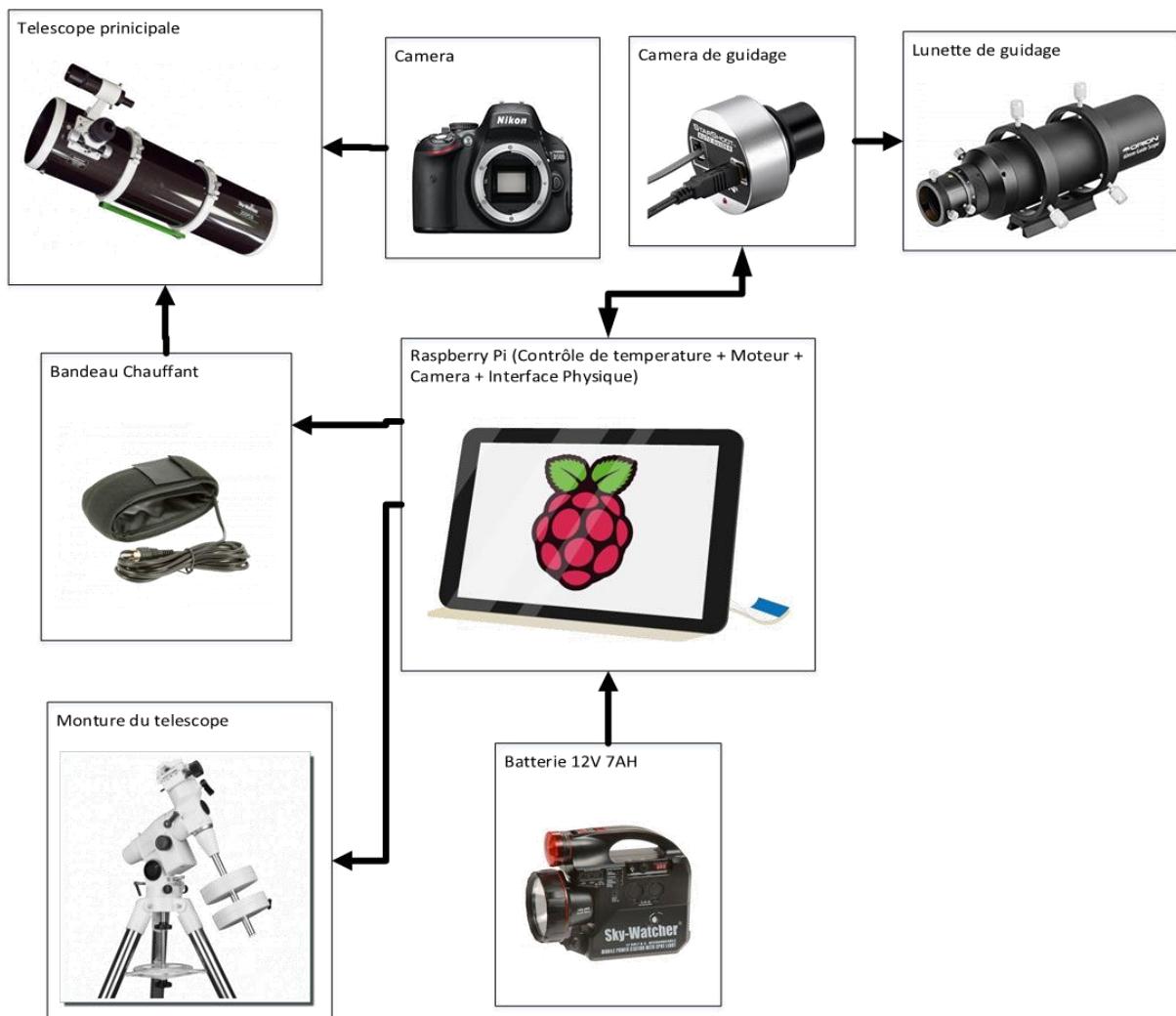


Ce schéma bloc représente mon télescope avant le projet.

Remarque : le suivi de ciel peut fonctionner sans le PC et la caméra de guidage, il nécessite seulement la raquette de commande (Boîte rouge).

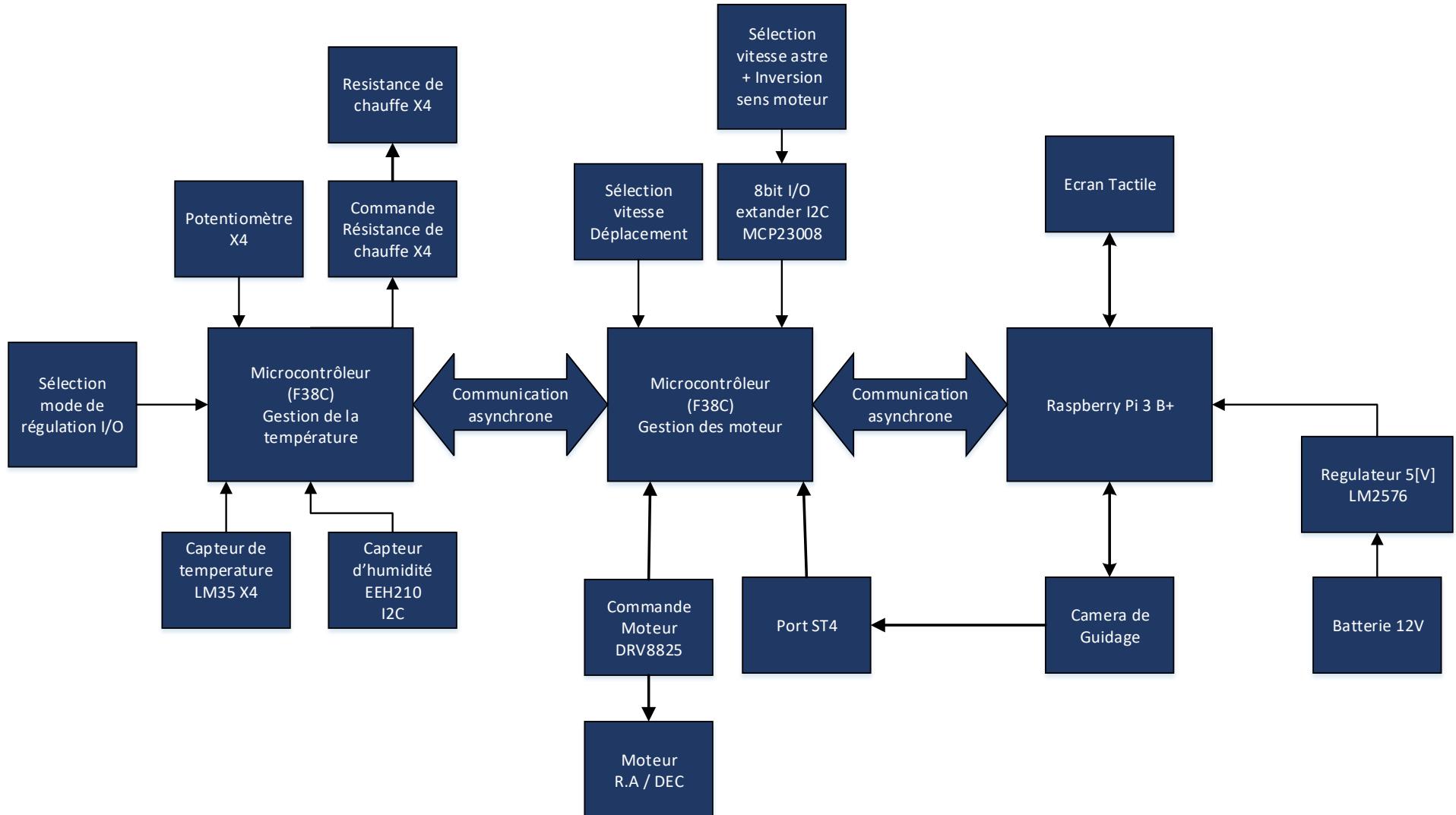
Le désavantage de cette configuration est qu'il nécessite beaucoup de matériel différent (Ex : Contrôleur de résistance de chauffe, commande moteur, PC...)

3.2.2. Télescope après

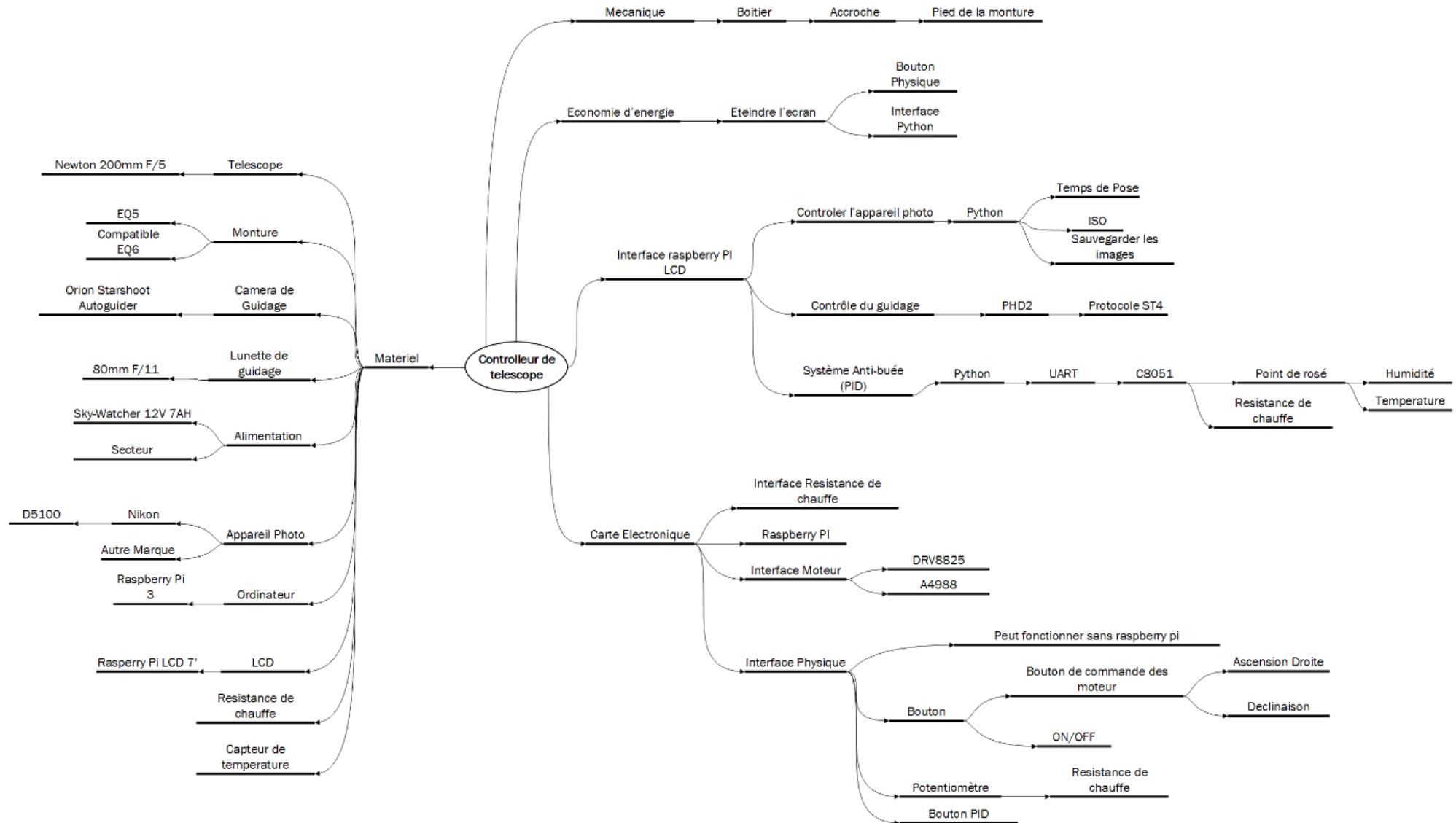


Comme expliquer plus haut le but de projet est de centraliser toutes les fonctionnalités de mon télescope actuelles en un seul boîtier, et d'y ajouter de fonction supplémentaire (contrôle par une interface web)

3.3. Schéma Bloc détaillé



3.4. Brainstorming

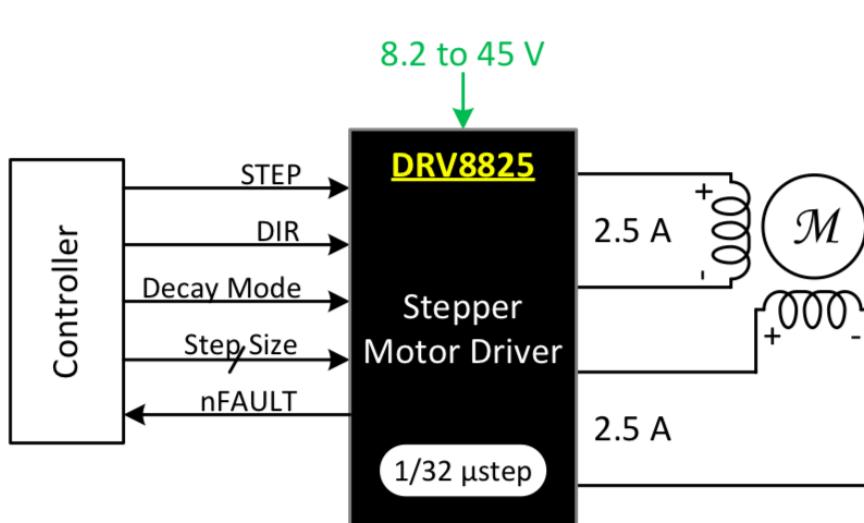


4. Conception et développement

4.1. Conception et développement matériel

Gestion des Moteurs

Afin de contrôler les moteurs du télescope, j'ai choisi d'utiliser le DRV8825 c'est un contrôleur de moteur pas à pas bipolaire, qui permet des faîtes des micropas.



Le contrôle est simple, il suffit de choisir la direction avec la pin DIR, puis il suffit de mettre un flanc montant sur la pin STEP afin de faire un pas sur le moteur.

Il contient un circuit de protection contre les courts-circuits, et les températures trop élevées.

Lorsque le DRV8825 se met en mode protection, il désactive le moteur, et met la pin nFault au niveau logique 0. Il faut ensuite utiliser la pin nReset afin de le sortir du mode de protection.

Il contient un régulateur de courant, qui permet de limiter le courant passant dans les bobines du moteur. Cette fonction a été très utile, car la carte de base (Sky-Watcher) fonctionnait en 6V, et le moteur consommait 125mA, alors lorsque le moteur était alimenté en 12V, il chauffait énormément.

J'ai calculé la valeur des résistances à connecter sur les pins ISEN A et ISEN B afin de réguler le courant et d'empêcher le moteur de chauffer en utilisant cette formule donnée dans la datasheet :

$$I_{FS} (A) = \frac{xVREF(V)}{A_v \times R_{SENSE} (\Omega)} = \frac{xVREF(V)}{5 \times R_{SENSE} (\Omega)}$$

Lors de mes mesures sur la raquette de commande Sky-Watcher, j'ai mesuré un courant max de 0,25[A] à 6[V] (1.5[W]).

Donc pour garder la même puissance à 12[V] il me suffit de réguler le courant à 0,125[mA].

Après une transformation de formule, j'obtiens le résultat suivant :

$$RSense = \frac{Vref}{Av * iFS} = \frac{3.3[V]}{5 * 0,125[A]} = 5,28[Ohm]$$

La résistance que j'ai utilisée est une résistance de 5.6[Ohm]

Précision théorique de l'axe d'ascension droite :

Les caractéristiques du moteur sont les suivantes :

Reference	Nombre de pas	deg/pas	(Réducteur)	Phase	Tension	Résistance
42PM48L	48	7.5	(120)	4	19	17 Ohm

Le réducteur dans le tableau au-dessus est intégré au moteur, mais il y a un deuxième réducteur intégrer dans la monture d'une valeur de 144, et j'utilise le DRV8825 en mode demi-pas (réduction de 2).

Donc la précision est la suivante :

$$Step = \frac{degPas}{Reducteur1 * Reducteur2 * 2} = \frac{7.5}{120 * 144 * 2} = 0.000217014[^{\circ}]$$

$$StepSecArc = Step * 3600 = 0.000217014 * 3600 = 0,78["]$$

Vitesse de déplacement maximal :

La vitesse maximale que je peux atteindre avec mes moteurs est de 48x la vitesse sidérale.

La terre fait un tour sur elle-même en 23 heures 56 minutes et 4 secondes, convertie en seconde cela donne 86164 secondes pour parcourir 360deg, donc la vitesse sidérale est de 15,04"/s sur l'axe d'ascension droite.

La vitesse en seconde d'arc par seconde :

$$VitesseMax = 48 * Vsideral = 48 * 15.04 = 721.92["/sec]$$

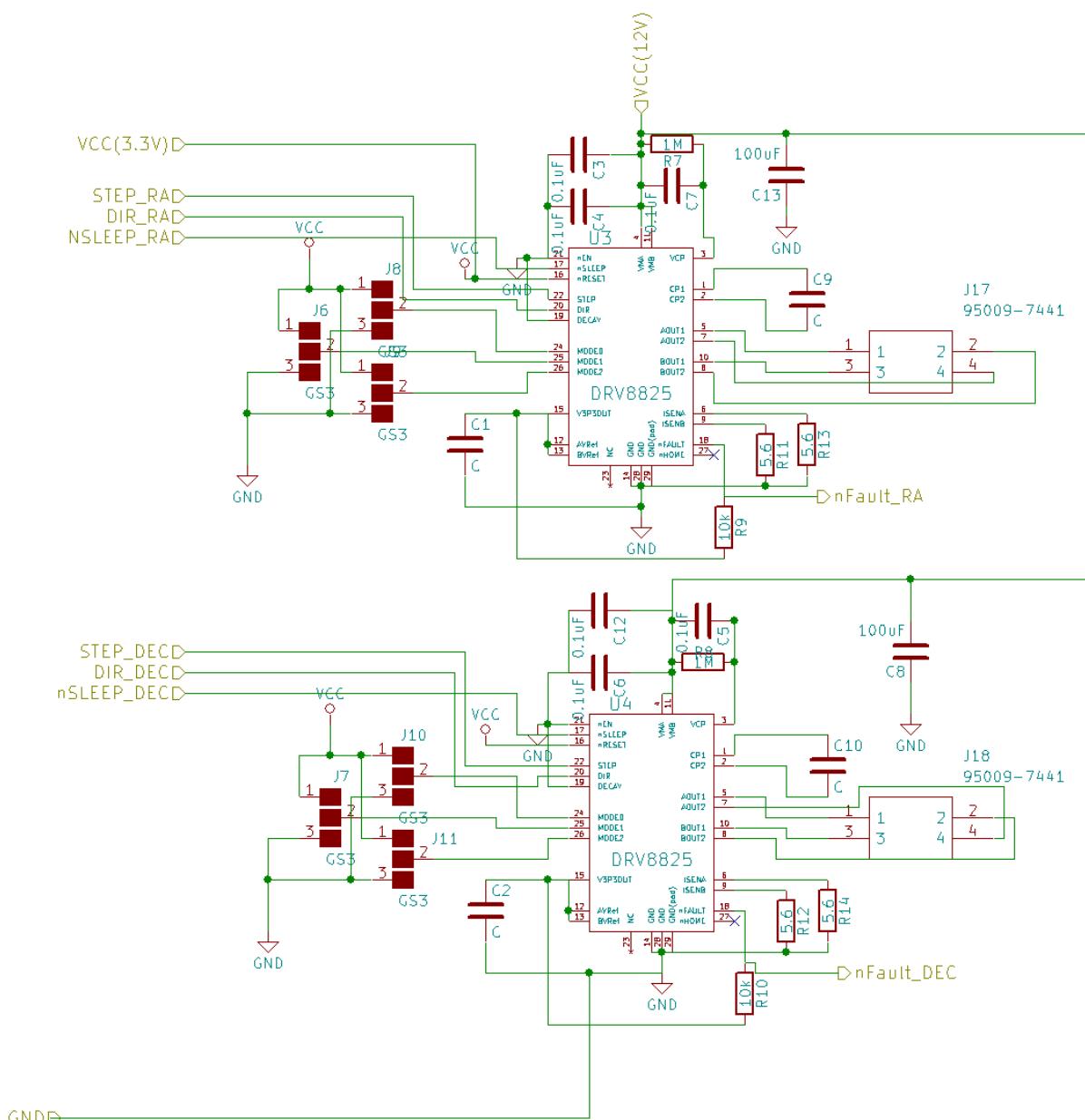
Convertie en degré :

$$VitesseMax = \frac{48 * Vsideral}{3600} = \frac{48 * Vsideral}{3600} = 0.2[^{\circ}/sec]$$

Donc avec la vitesse maximale, il faut 30 minutes pour faire un tour complet sur l'axe d'ascension droite :

$$Ttour = \frac{360}{0.2 * 60} = 30 [minute]$$

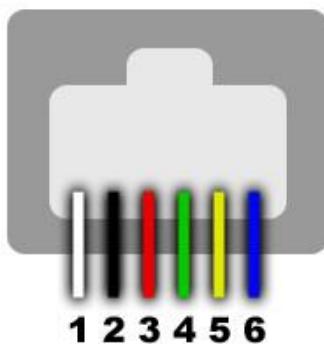
Schéma de câblage pour les deux drivers moteurs DRV8825 :



Les pads J6 à J11, me permettent de choisir le mode de fonctionnement du moteur, actuellement ils sont configurés pour que les moteurs effectuent des demi-pas.

4.1.1. Interface ST4

C'est un connecteur RJ12 dont la fonction de base a été détournée afin de permettre de contrôler un télescope, il contient 6 fils :



**RJ11 connector
for ST4 cable**

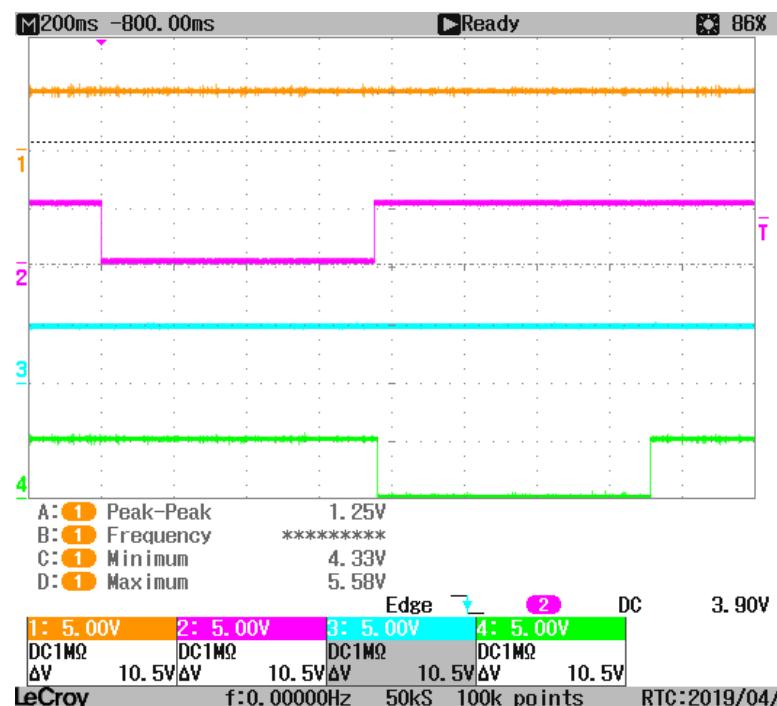
- 1 not used**
- 2 GND**
- 3 -RA (-X), N.O.**
- 4 -DEC (-Y), N.O.**
- 5 +DEC (+Y), N.O.**
- 6 +RA (+X) , N.O.**



Par défaut les pins sont à 1, lorsque l'un des pins est connecté à la masse, cela indique au microcontrôleur qu'il faut se déplacer sur l'axe sélectionné.

Les sorties du port ST4 sont en open collector.

Mesure sur le port ST4 en utilisant l'interface PHD2 guiding, en mode manuel :



Appui sur le bouton de déplacement Nord, puis Est, PHD2 est configuré pour envoyer un pulse de 750[ms].

Ch1 : bouton Sud

Ch2 : Bouton Nord

Ch3 : Bouton Ouest

Ch4 : Bouton Est

Remarque :

Deux signaux d'un même axe ne peuvent pas être à l'état bas en même temps, cela n'aurait pas de sens, car cela indiquerait au microcontrôleur d'aller dans deux directions opposées en même temps. Ex : Nord et Sud en même temps.

Pour la programmation je voulais mettre chacune de ces pins sur interruption, mais le microcontrôleur que j'utilise n'a que 2 interruptions, donc seulement 2 pins disponibles.

Afin de remédier à ce problème, j'ai décidé d'utiliser une porte ET à deux entrées (CD4081), et de connecter les axes dessus (Ex : Nord et Sud) ce qui me permet d'avoir une pin d'interruption pour un axe. Il me suffit d'aller regarder lors de l'interruption quel pin est à l'état bas pour connaître la direction.

INPUT		OUTPUT
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

L'état de repos lorsque rien ne bouge serait l'état haut, et l'état bas indiquerait une demande de mouvement.

Il me suffit de capter un flanc descendant pour commencer un mouvement et de capter le flanc montant pour arrêter le mouvement

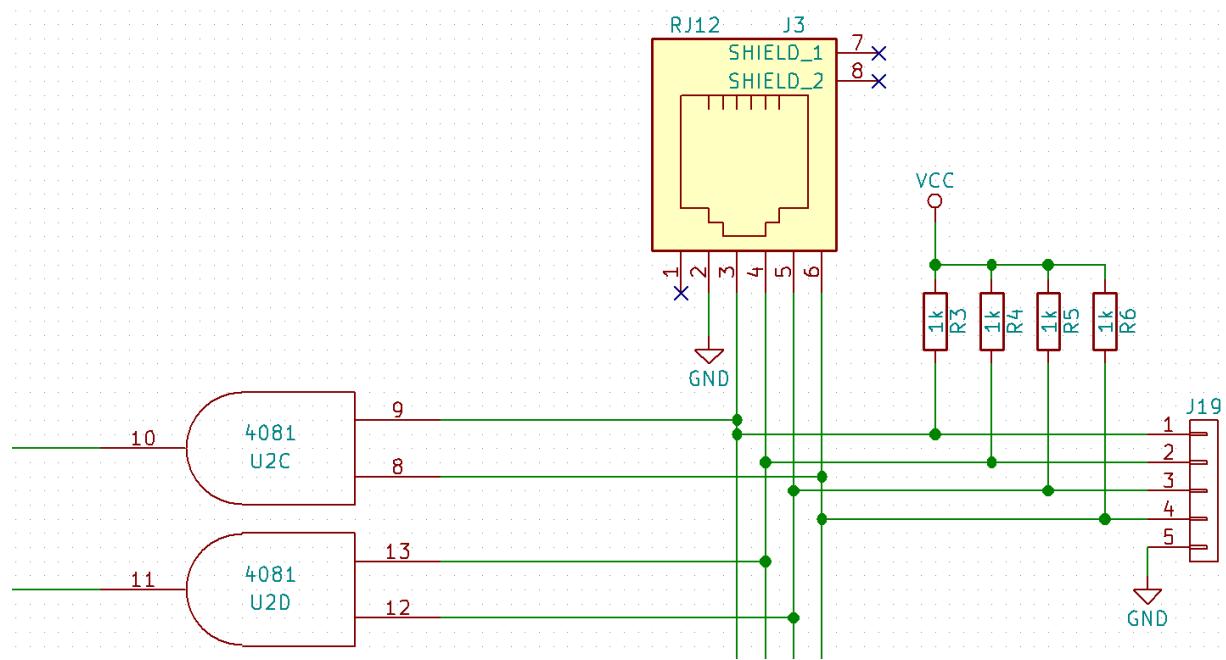
Le cas où les entrées A et B sont à 0 ne devrait pas se produire.

Je me suis rendu compte que je pouvais remplacer la porte ET avec une porte OU-Exclusif, car son avantage serait que je pourrais détecter sur un flanc le cas où les deux entrées sont à 0.

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Dans ce cas le flanc montant indique un mouvement, alors que le flanc descendant indique un arrêt des moteurs.

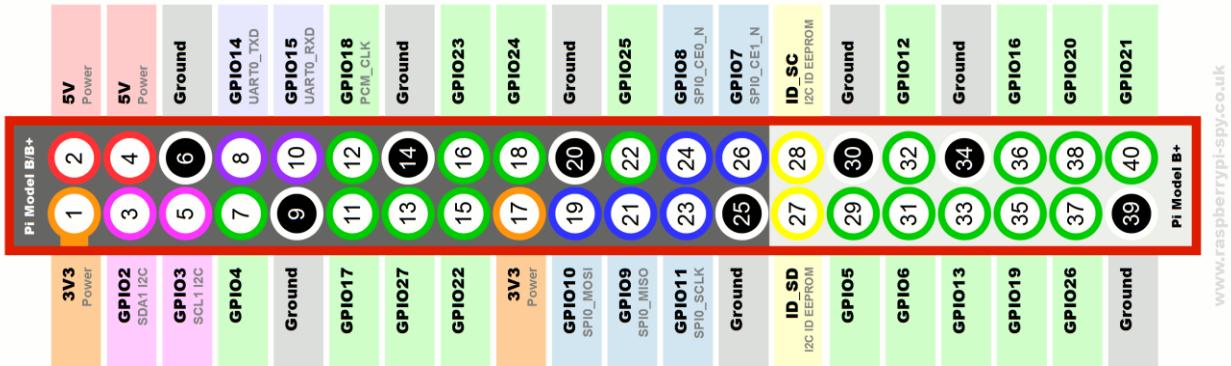
Schéma de câblage du port ST4 :



Lors de test je me suis rendu compte que lorsque le microcontrôleur sautait dans l'interruption, il voyait toujours les deux pins de direction à 1, le problème venait des résistances de pull-up que j'avais choisi, j'avais mis des résistances de 10[kOhm]. Pour résoudre le problème, j'ai remplacé les résistances de 10[KOhm] par des résistances de 1 [kOhm], afin que le flanc descendant soit plus rapide.

4.1.2. Alimentation et Raspberry Pi

Lors de mes tests pour alimenter la Raspberry Pi en utilisant une pin 5V disponible sur le connecteur 40 pin de la Raspberry Pi, j'ai remarqué qu'elle ne démarrait pas, alors qu'en utilisant l'alimentation fournit avec la raspberry pi, il n'y avait aucun problème.

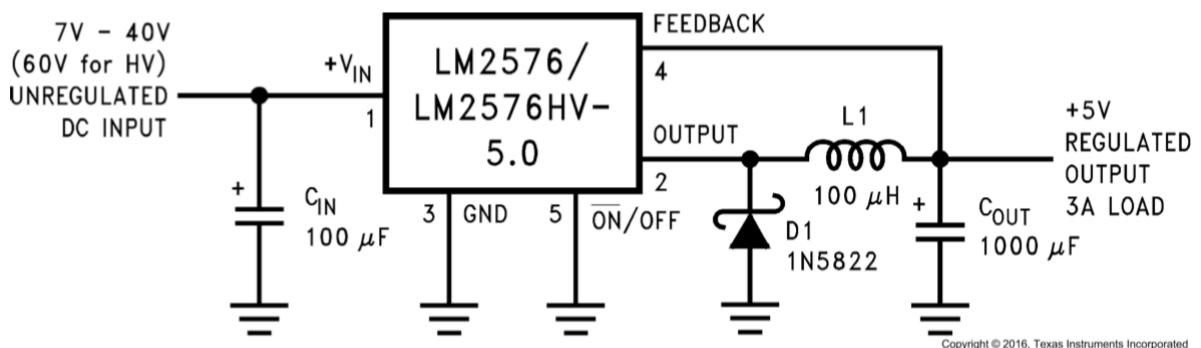


Ma première idée a été de tester différentes alimentations, mais aucune alimentation que j'ai testée ne permettait de faire démarrer la Raspberry pi, suite à une discussion avec M. Kenzi, il m'a conseiller d'utiliser des câbles avec une section plus grosse, ce qui a résolu le problème. Donc lors de la conception de la carte mère sur kicad j'ai mis de très grandes pistes afin d'assurer la bonne conduction entre la Raspberry pi et l'alimentation.

Afin de générer le 5[V] sur la carte pour alimenter la Raspberry Pi et les deux microcontrôleurs, je me suis tourné vers une alimentation à découpage.

Pendant la deuxième année de technicien, j'ai étudié le LM2575, un régulateur à découpage 5V – 1[A], mais j'avais besoin d'un régulateur capable de débiter au minimum 2[A].

M. Cayuso m'a dit qu'il utiliserait un LM2576 pour alimenter son projet, ce régulateur à découpage est parfait pour moi, car son courant de sortie maximum est de 3[A]. De plus le schéma est le même que celui du LM2575 donc je n'avais qu'as réutilisé les cartes développer durant l'année.



4.1.3. Carte de température

La carte de température me permet de contrôler des résistances de chauffe, afin d'éviter que de la condensation apparaisse sur les lentilles ou les miroirs.

Pour empêcher la condensation d'apparaître sur les miroirs, il faut calculer le point de rosée et réguler la température de la lentille au-dessus de cette température.

Le point de rosée détermine la température à partir de laquelle la condensation va se former.

Le point de rosée varie en fonction de l'humidité ambiante et de la température ambiante.

$$Tr = \sqrt[8]{\frac{H}{100}} * (112 + (0.9 * T)) + (0.1 * T) - 112$$

T=Température ambiante (C)

H=Humidité ambiante (%)

Mais il ne faut pas trop chauffer le miroir ou la lentille, car cela créerait des déformations qui engendrerait une perte de qualité. Mais trop chauffer peut aussi générer des flux d'air dans le tube du télescope ce qui entraînerait aussi une perte de qualité visuelle.



Afin de lutter contre la condensation, j'ai décidé d'utiliser des résistances chauffantes sous forme de bandeau.

Pour faire varier la puissance dégagée par la résistance de chauffe j'ai décidé d'utiliser un PWM et de le générer avec les modules PCA du microcontrôleur.

Il y a quatre sorties RCA sur la carte température : une pour le miroir secondaire, une pour le miroir primaire, une pour le porte-oculaire, et une de réserve.

Dans mon cas j'ai très rarement eu besoin de chauffer le miroir primaire ni le porte-oculaire.

Capteur de température et d'humidité :

Afin de connaître le point de rosé, il est nécessaire d'avoir un capteur de température ambiante, ainsi qu'un capteur d'humidité

Afin de connaître le point de rose, j'ai décidé d'utiliser un capteur qui mesure la température et l'humidité et qui est accessible par une communication I2C.

J'ai utilisé le capteur EEH210, car je l'avais utilisé lors de mon deuxième travail de semestre, donc je savais déjà comment il fonctionnait.

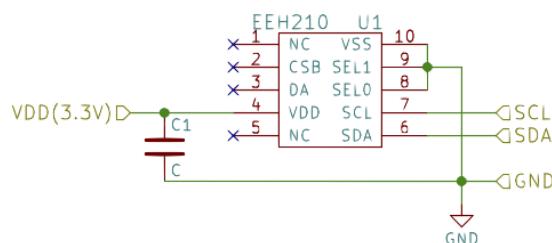
Ces caractéristiques sont les suivantes :



Température : -40[°C] à 120[°C] - 14 bits

Humidité relative : 0 à 100[%] – 12 bits

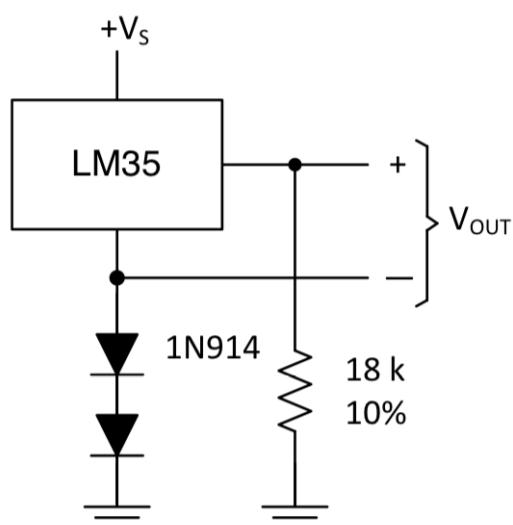
Alimentation : 3.3[V]



Ensuite il me reste à connaître la température de chaque élément que je veux réguler.

Au début je pensais utiliser un capteur en I2C ou en SPI, mais je me suis rendu compte que je ne pourrais pas utiliser l'I2C, car il me faudrait des capteurs avec une adresse différente pour chaque capteur, et je n'ai pas utilisé de capteur SPI, car le protocole SPI demande beaucoup de fil. Donc j'ai choisi des capteurs analogiques, j'avais des capteurs de température LM35 à disposition, donc j'ai décidé de les utiliser.

Les capteurs LM35 font varier leur tension de sortie de 10[mV/C], mais le 0[C] correspond à 0[V], ce qui signifie qu'il ne mesure pas de température négative. En lisant la datassheet, j'ai trouvé ce schéma qui me permet de lire les températures négatives, en ajoutant seulement deux diodes et une résistance.



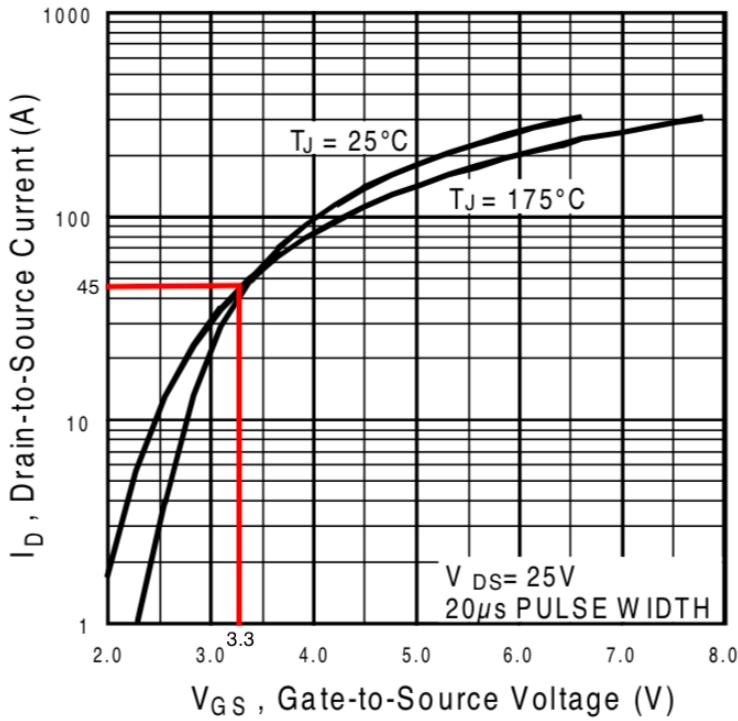
Dans mon schéma j'ai décidé de n'utiliser qu'une seule diode, car je n'ai pas besoin de mesurer jusqu'à -55[°C]

Figure 18. Temperature Sensor, Single Supply
(-55° to +150°C)

Interface de puissance pour résistance de chauffe :

J'ai utilisé un MOSFET canal N de puissance pour commander les résistances de chauffe. Après en avoir parlé avec M. Carbon, il m'a guidé vers le MOSFET qui a été utilisé dans le projet de charge active étudier durant cette année.

Caractéristique de transfert du MOSFET IRL3705 :



Le microcontrôleur F38C a une tension de sortie de 3.3[V].

Ce qui veut dire que le courant maximum que peut tirer le Irl3705 est de 45[A]

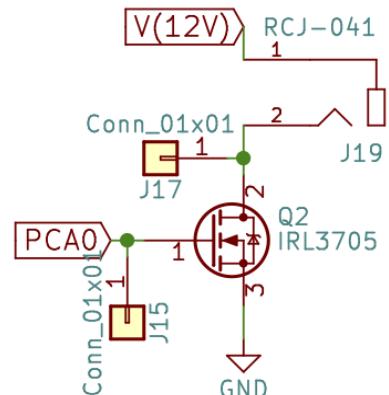


Fig 3. Typical Transfer Characteristics

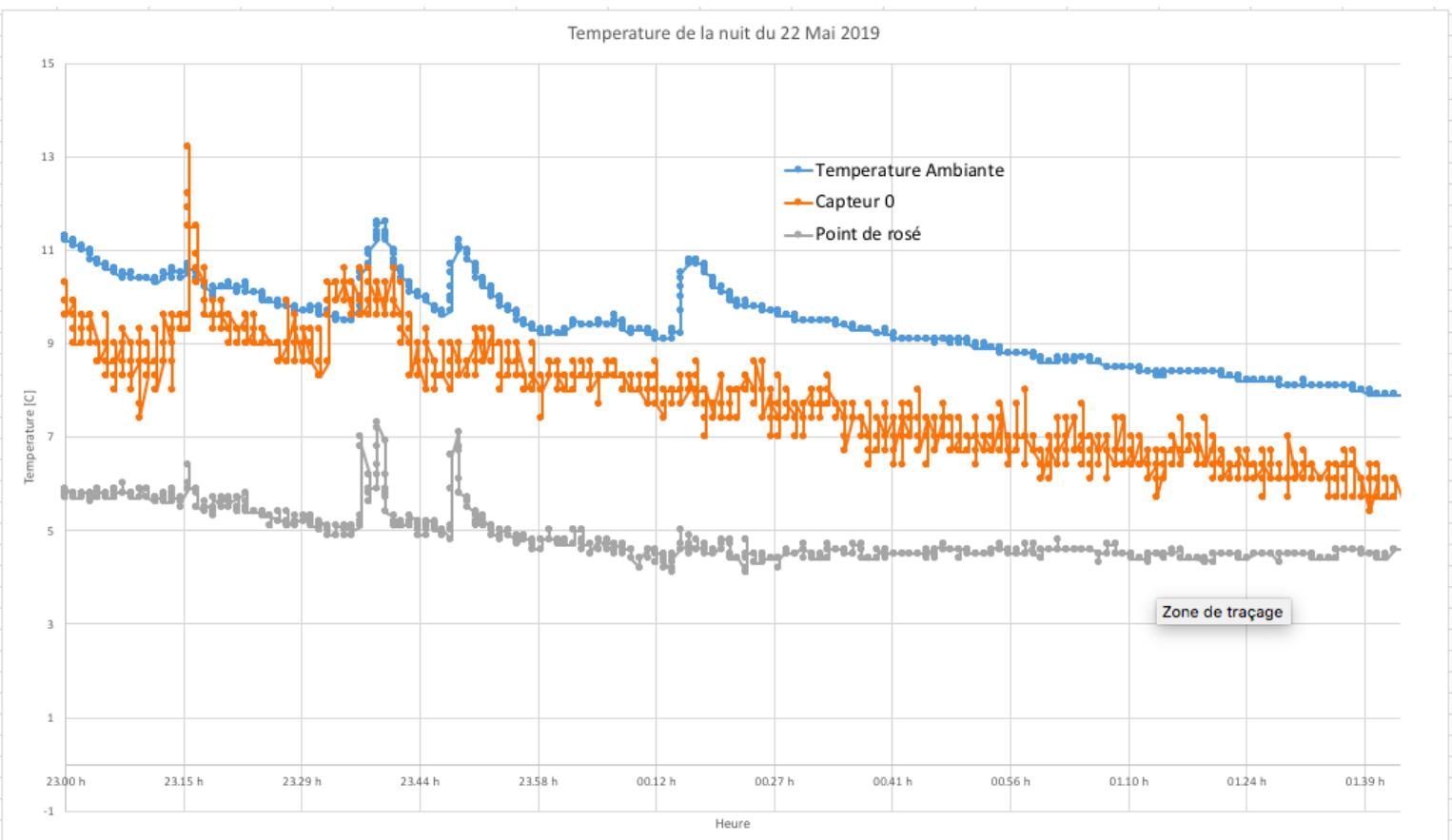


Mesure de VGS et de VDS avec une résistance de chauffe connecter sur le circuit :

Ch1 : VDS (J17)

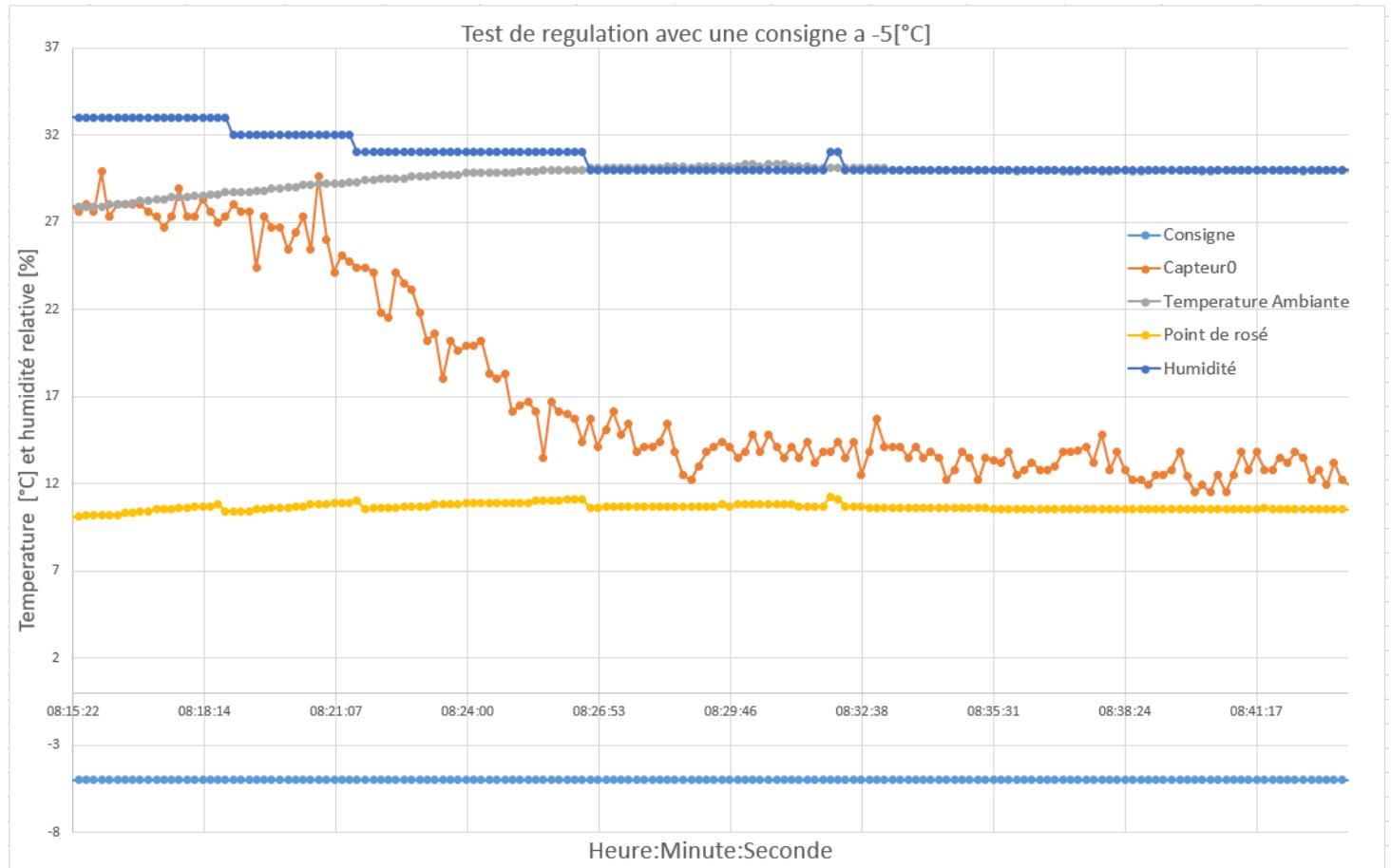
Ch2 : VGS (J15)

Test de régulation dehors pendant une nuit :



Lors de cette nuit, les résistances de chauffe n'ont pas eu besoin de s'activer, car la température du capteur 0 n'est jamais descendue en dessous du point de rosée.

Test de régulation en utilisant l'armoire thermique de l'atelier :



Au début du test, l'intérieur de l'armoire thermique était à la température ambiante (~28[°C]), puis la température à l'intérieur de l'armoire est descendu jusqu'à -5[°C].

Sur ces mesures, je remarque que la carte de gestion de la température a fonctionné, car la température du capteur 0 (Orange) n'est jamais descendue en dessous du point de rosée.

Le test a été effectué comme cela :



Le capteur0 (LM35) a été accroché à une plaque d'aluminium avec une vis et un écrou, puis la résistance de chauffe a été scotchée sur la plaque.

Lorsque j'ai sorti la plaque de l'armoire thermique, il n'y avait pas de condensation sur la plaque d'aluminium, mais il y en avait légèrement sur les parois de l'armoire thermique.

4.1.4. Consommation

Consommation maximale, en utilisant tous les composants :

Élément	Min[A]	Moyenne[A]	Max[A]
Raspberry Pi + LCD (100%) + Camera en fonctionnement + Site Web	0.5	0.6	0.75
Carte Moteur (en régulation)	0.056	0.082	0.086
Carte Température + 3 résistances thermiques à 100[%]	0.984	0.984	0.984
Total :	1.54	1.666	1.82

Remarque : le courant maximal au démarrage de la Raspberry pi peut atteindre 850[mA], ces mesures ont été faites en fonctionnement.

Donc le courant maximal est de 1.92[A], à 12[V], ce qui correspond à 23.04[W]

Ma batterie est une 12 [V] – 7[Ah].

Dans de telles conditions, le télescope ne pourrait fonctionner que pendant 3h et 38 minutes.

Exemple de consommation en été sans faire de la photographie :

Élément	Min[A]	Moyenne[A]	Max[A]
Raspberry Pi + LCD (100%) + Camera en fonctionnement + Site Web	0	0	0
Carte Moteur (suivi simple Axe RA)	0.05	0.05	0.06
Carte Température inutilisée	0	0	0
Total :	0.05	0.05	0.06

Dans ces conditions d'utilisation, le télescope pourrait fonctionner pendant 116h.

Exemple de consommation en été avec de la photographie :

Élément	Min[A]	Moyenne[A]	Max[A]
Raspberry Pi + LCD (100%) + Camera en fonctionnement + Site Web	0.5	0.6	0.75
Carte Moteur (suivi régulier)	0.056	0.082	0.086
Carte Température inutilisée	0	0	0
Total :	0.556	0.682	0.836

Ces conditions sont les plus intéressantes pour moi, car ce sera dans ces conditions que j'utiliserai mon télescope le plus souvent.

En utilisant le courant maximal, le télescope pourrait fonctionner pendant 8h et 22 minutes.

Mais en utilisant le courant consommer en moyenne, le télescope fonctionnerait durant 10h et 15 minutes.

4.2. Conception et développement logiciel

4.2.1 Développement du code de gestion de moteur

4.2.1.1. Calcul de la vitesse des moteurs

Pour rappel, les caractéristiques du moteur sont les suivantes :

Reference	Nombre de pas	deg/pas	(Réducteur)	Phase	Tension	Résistance
42PM48L	48	7.5	(120)	4	19	17 Ohm

Suivi du ciel :

Pour rappel : La terre fait un tour sur elle-même en 23 heures 56 minutes et 4 secondes, convertie en seconde cela donne 86164 secondes pour parcourir 360deg, donc un mouvement de 15,04"/s sur l'axe d'ascension droite.

$$Vitesse sideral = \frac{360}{86164} = 0,004178079 [deg/s]$$

$$Vitesse sideral = \frac{360}{86164} * 3600 = 15,04["/s]$$

La vis sans fin de l'axe d'ascension droite a 144 dents, cela implique une réduction de 144 en plus du réducteur monter sur le moteur.

En prenant en compte tous ces paramètres, la formule qui permet de calculer la vitesse des moteurs est la suivante :

$$VMot = \frac{Vsideral(deg) * 144 * 120 * 60}{360} = 12.03 [tr/min]$$

La datasheet du drv8825 donne cette formule afin de calculer la fréquence à mettre sur le pin STEP pour obtenir la vitesse de moteur choisi :

$$f_{step} (\mu steps / second) = \frac{\nu \left(\frac{\text{rotations}}{\text{minute}} \right) \times 360 \left(\frac{\circ}{\text{rotation}} \right) \times n_m \left(\frac{\mu steps}{\text{step}} \right)}{60 \left(\frac{\text{seconds}}{\text{minute}} \right) \times \theta_{step} \left(\frac{\circ}{\text{step}} \right)}$$

Dans mon cas nm=2, car le driver fonctionne en demi pas :

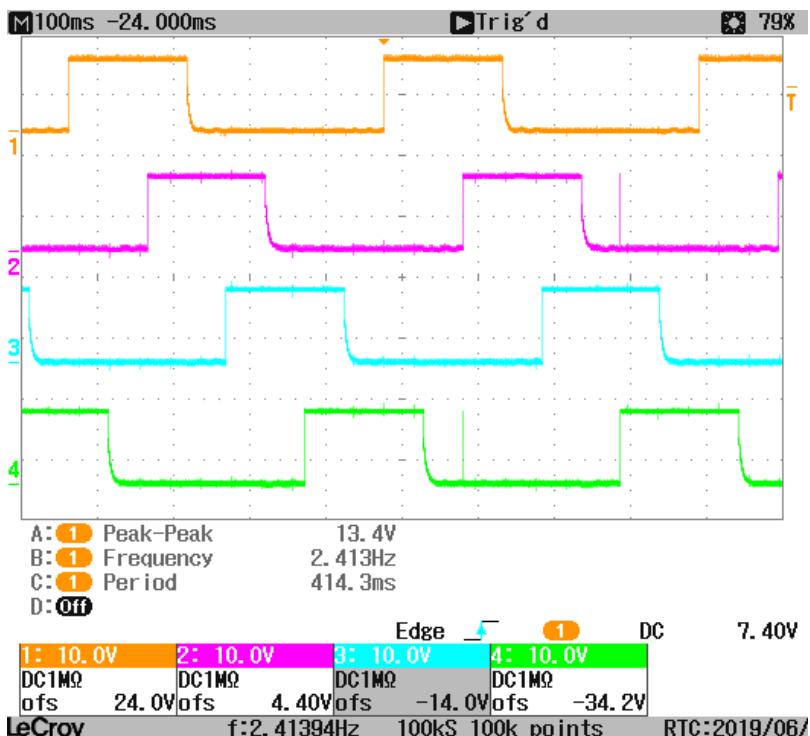
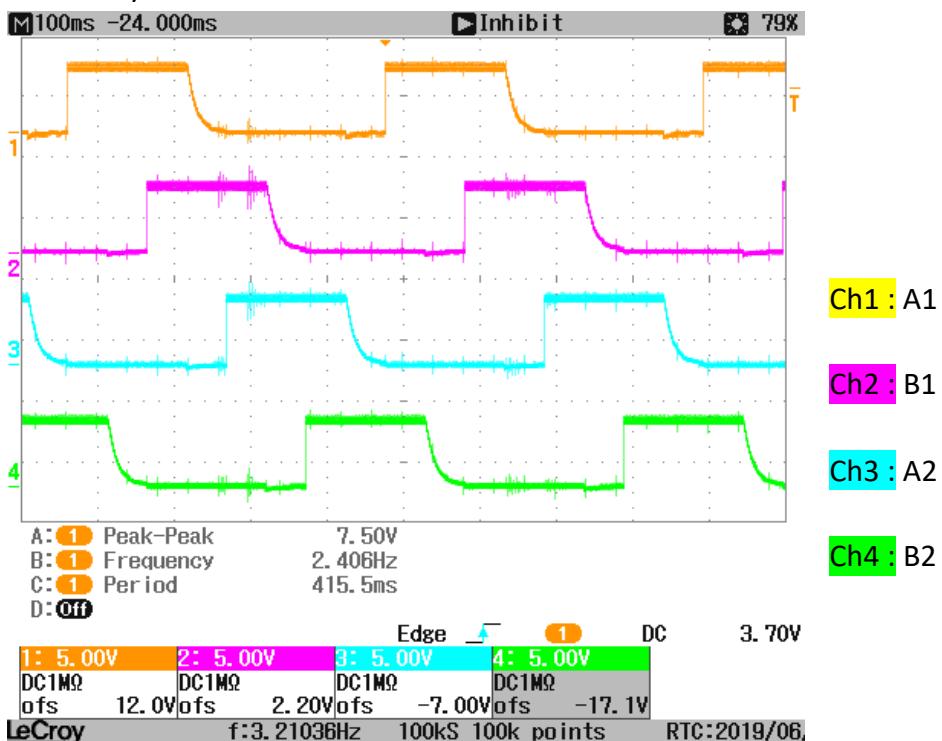
$$FStep = \frac{Vmot * 360 * 2}{60 * 7,5} = 19,25[Hz]$$

Afin de mettre cette fréquence sur le moteur, j'ai utilisé le timer 0 du f38C, j'ai calculé la valeur de préchargement avec cette formule :

$$LoadValue = 2^{16} - \frac{\frac{1}{FStep}}{\frac{2}{Preditiseur}} = 2^{16} - \frac{\frac{1}{19,25}}{\frac{2}{48}} = 39565,47 \rightarrow 39565$$

Comparaison entre la vitesse de la raquette de commande sky-watcher, et les pulsations générées par le driver DRV8825 pour la vitesse sidérale :

Sky-Watcher :



Signaux générés par le F38C :

On peut remarquer que les deux signaux sont presque identiques, la seule différence est le niveau des tensions. La régulation de courant n'est pas actif, car les mesures ont été faites sans brancher le moteur.

Suivi de la lune :

La vitesse de déplacement de la lune n'étant pas la même que la vitesse de déplacement du ciel, il est nécessaire de calculer une vitesse précise pour les moteurs, tout comme pour le ciel.

Les calculs sont les mêmes que pour le suivi du ciel :

Pour trouver le décalage de la lune chaque jour, j'ai utilisé le site wolframalpha, et j'ai cherché l'heure du lever de deux jours à la suite, puis j'ai calculé la différence. J'ai obtenu 82502 secondes.

$$Vitesse\ sideral = \frac{360}{82502} = 0.0043635 [deg/s]$$

$$Vitesse\ sideral = \frac{360}{82502} * 3600 = 15,7["/s]$$

La vis sans fin de l'axe d'ascension droite a 144 dents, cela implique une réduction de 144 en plus du réducteur monter sur le moteur.

En prenant en compte tous ces paramètres, la formule qui permet de calculer la vitesse des moteurs est la suivante :

$$VMot = \frac{Vsideral(deg) * 144 * 120 * 60}{360} = 12,56 [tr/min]$$

La datasheet du DRV8825 donne cette formule afin de calculer la fréquence à mettre sur la pin STEP pour obtenir la vitesse de moteur choisi :

$$f_{step} (\mu steps / second) = \frac{v \left(\frac{\text{rotations}}{\text{minute}} \right) \times 360 \left(\frac{\circ}{\text{rotation}} \right) \times n_m \left(\frac{\mu steps}{\text{step}} \right)}{60 \left(\frac{\text{seconds}}{\text{minute}} \right) \times \theta_{step} \left(\frac{\circ}{\text{step}} \right)}$$

dans mon cas nm=2, car le driver fonctionne en demi pas :

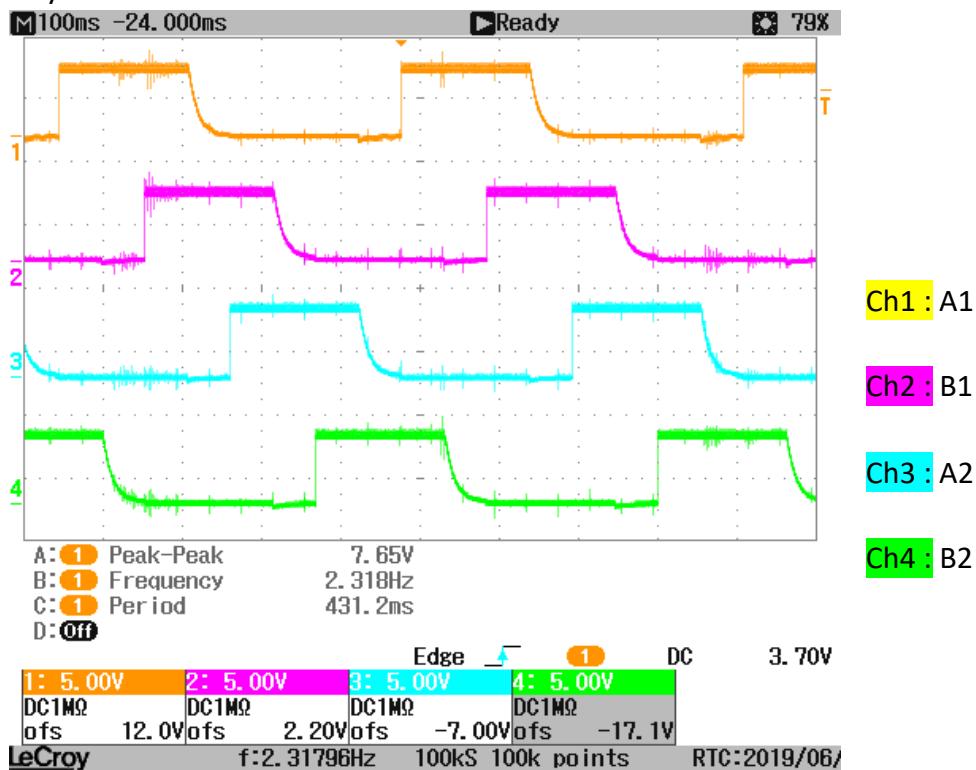
$$FStep = \frac{Vmot * 360 * 2}{60 * 7,5} = 20,1[Hz]$$

Afin de mettre cette fréquence sur le moteur, j'ai utilisé le timer 0 du f38C, j'ai calculé la valeur de préchargement avec cette formule :

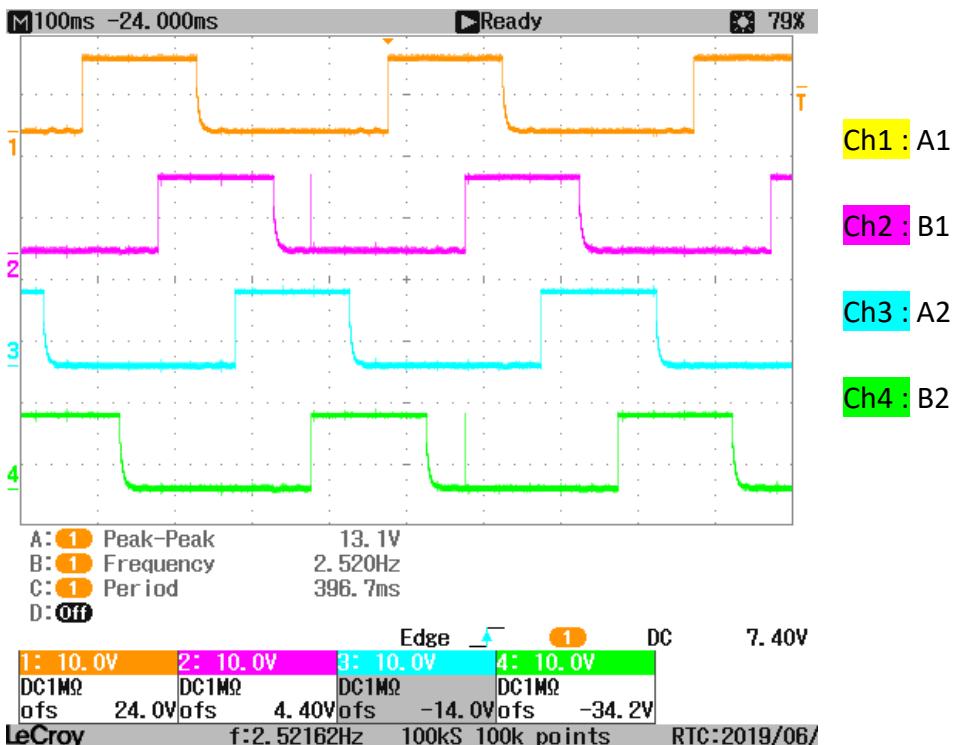
$$LoadValue = 2^{16} - \frac{\frac{1}{FStep}}{\frac{2}{Prediviseur}} = 2^{16} - \frac{\frac{1}{21,25}}{\frac{2}{48}} = 40669,22 \rightarrow 40669$$

Comparaison entre la vitesse de la raquette de commande sky-watcher, et les pulsations générées par le driver DRV8825 pour la vitesse lunaire :

Sky-Watcher :



Signaux générés par le F38C :



Contrairement à la vitesse de suivi sidéral, il y a une différence dans la fréquence du signal, la raquette de commande sky-Watcher génère un signal de 2.318[Hz], alors que mon programme génère une fréquence de 2.520[Hz].

Suivi du soleil :

Pour trouver le décalage du soleil chaque jour, j'ai encore utilisé le site wolframalpha, et j'ai cherché l'heure du lever de deux jours à la suite, puis j'ai calculé la différence. J'ai obtenu 85917 secondes.

$$Vitesse sideral = \frac{360}{85917} = 0.00419009 [deg/s]$$

$$Vitesse sideral = \frac{360}{85917} * 3600 = 15,08["/s]$$

La vis sans fin de l'axe d'ascension droite a 144 dents, cela implique une réduction de 144 en plus du réducteur monter sur le moteur.

En prenant en compte tous ces paramètres, la formule qui permet de calculer la vitesse des moteurs est la suivante :

$$VMot = \frac{Vsideral(deg) * 144 * 120 * 60}{360} = 12,06 [tr/min]$$

La datasheet du DRV8825 donne cette formule afin de calculer la fréquence à mettre sur la pin STEP pour obtenir la vitesse de moteur choisi :

$$f_{step} (\mu\text{steps / second}) = \frac{v \left(\frac{\text{rotations}}{\text{minute}} \right) \times 360 \left(\frac{\text{°}}{\text{rotation}} \right) \times n_m \left(\frac{\mu\text{steps}}{\text{step}} \right)}{60 \left(\frac{\text{seconds}}{\text{minute}} \right) \times \theta_{step} \left(\frac{\text{°}}{\text{step}} \right)}$$

dans mon cas nm=2, car le driver fonctionne en demi pas :

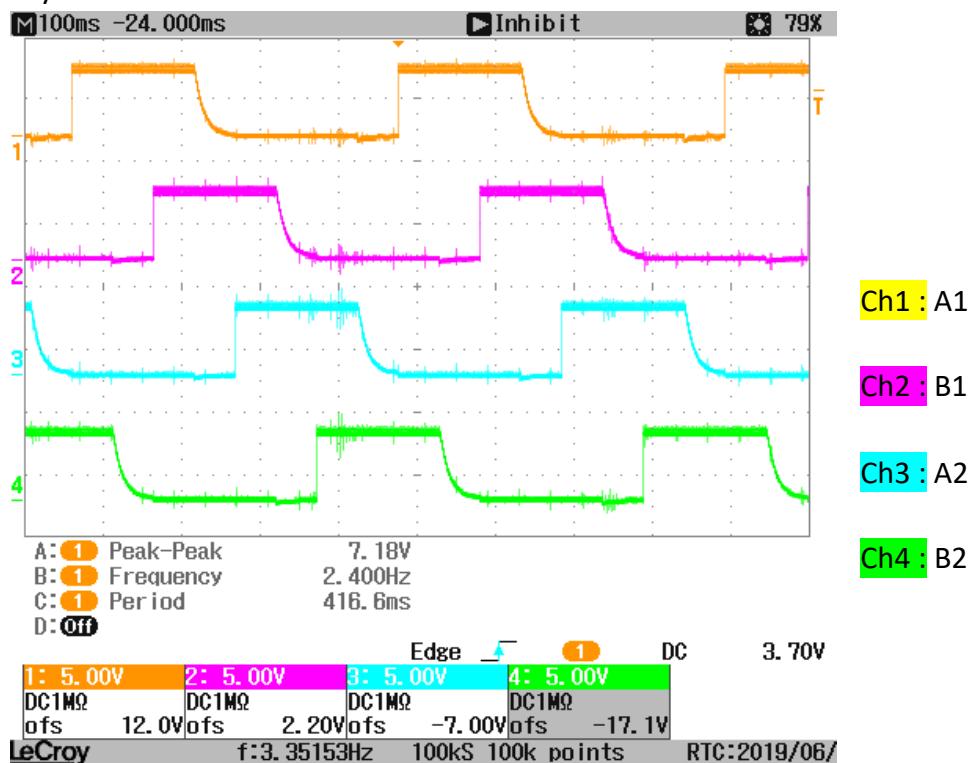
$$FStep = \frac{Vmot * 360 * 2}{60 * 7,5} = 19.3[Hz]$$

Afin de mettre cette fréquence sur le moteur, j'ai utilisé le timer 0 du f38C, j'ai calculé la valeur de préchargement avec cette formule :

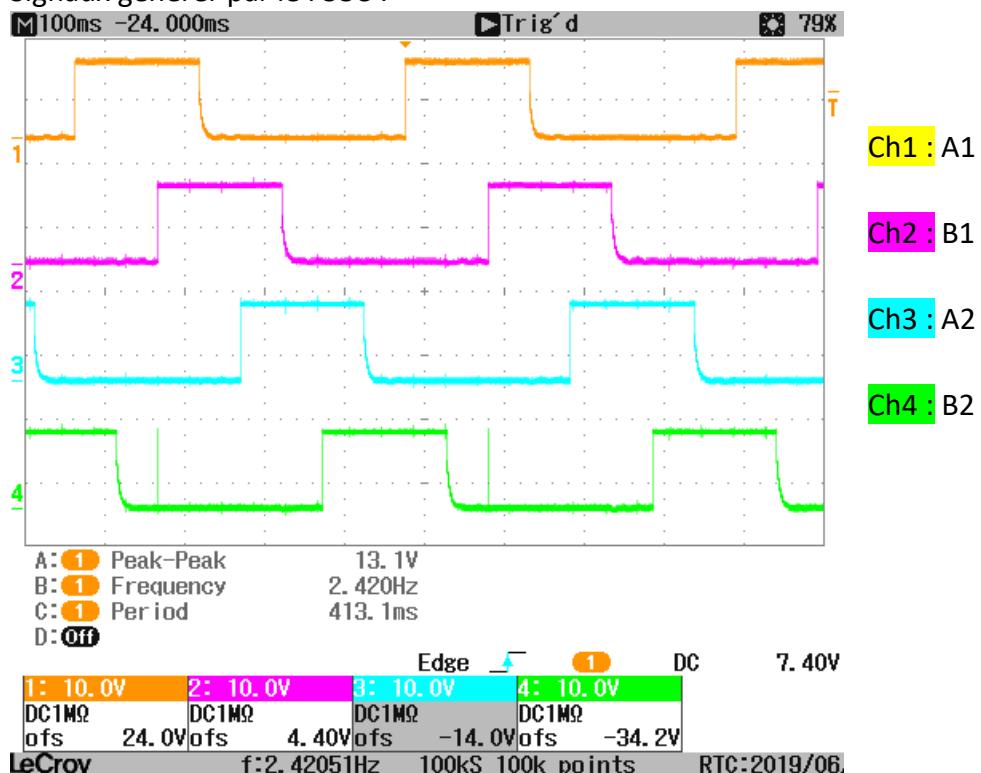
$$LoadValue = 2^{16} - \frac{\frac{1}{FStep}}{\frac{2}{Prediviseur}} = 2^{16} - \frac{\frac{1}{21,25}}{\frac{2}{48}} = 39639.91 \rightarrow 39640$$

Comparaison entre la vitesse de la raquette de commande sky-watcher, et les pulsations générées par le driver DRV8825 pour la vitesse lunaire :

Sky-Watcher :



Signaux générés par le F38C :

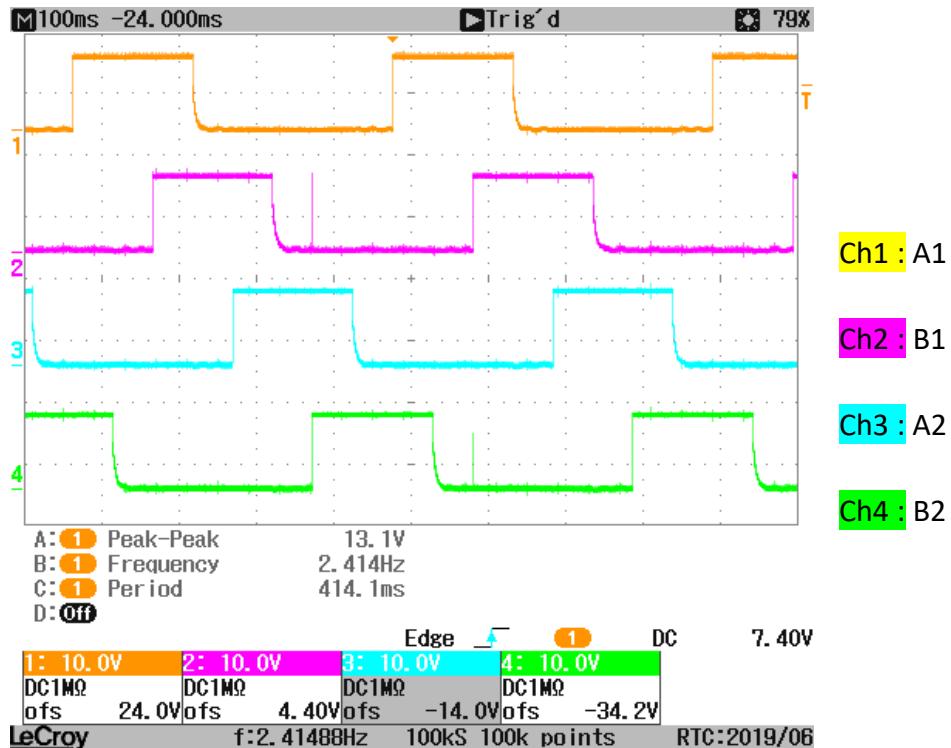


Dans ce cas, la vitesse obtenue pour suivre le soleil est aussi très proche de la vitesse de la raquette de commande sky-Watcher.

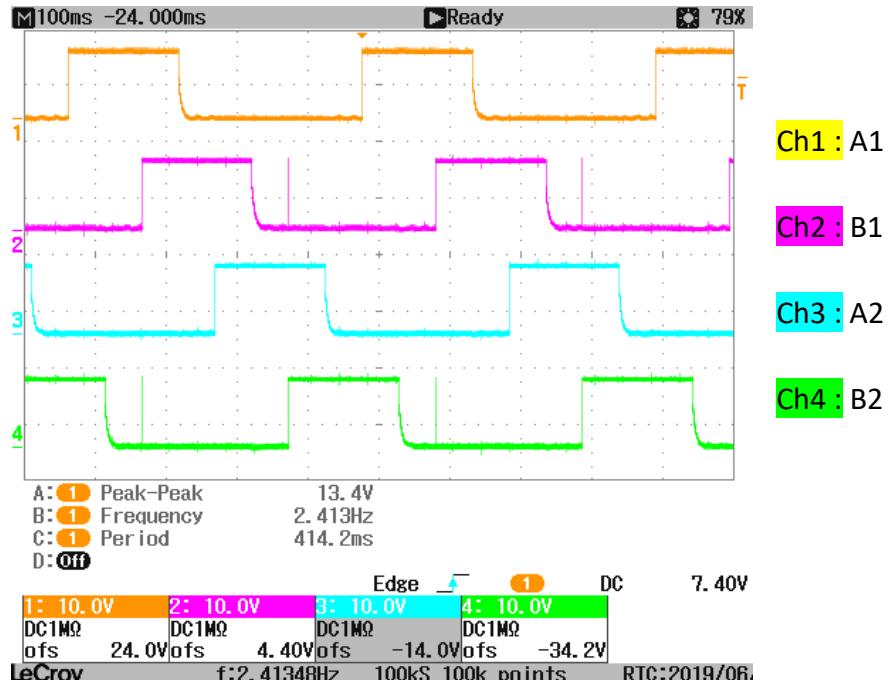
Les valeurs que j'ai trouvés pour Jupiter et Saturne sont les suivantes :

	Jupiter	Saturne
Temps entre chaque lever [seconde]	86131,2	86150,4
LoadValue	39575	39569

Mesure des pulses envoyées aux moteurs pour suivre Jupiter :



Mesure des pulses envoyées aux moteurs pour suivre Saturne :



J'ai aussi créé une feuille Excel qui calcule automatiquement les valeurs de précharge.

Toutes ces vitesses peuvent être choisi simplement en appelant la fonction setMotorRA() et setMotorDEC()

```
/*-----*
setMotorRA()

-----*
Descriptif: Applique une vitesse predefinit au moteur et une direction
Entree   :
    - unsigned char vitesse (0 ... 10)
    - bit direction          (0 ... 1)
Sortie   : --
-----*/
void setMotorRA(unsigned char vitesse,bit direction)

/*-----*
setMotorDEC()

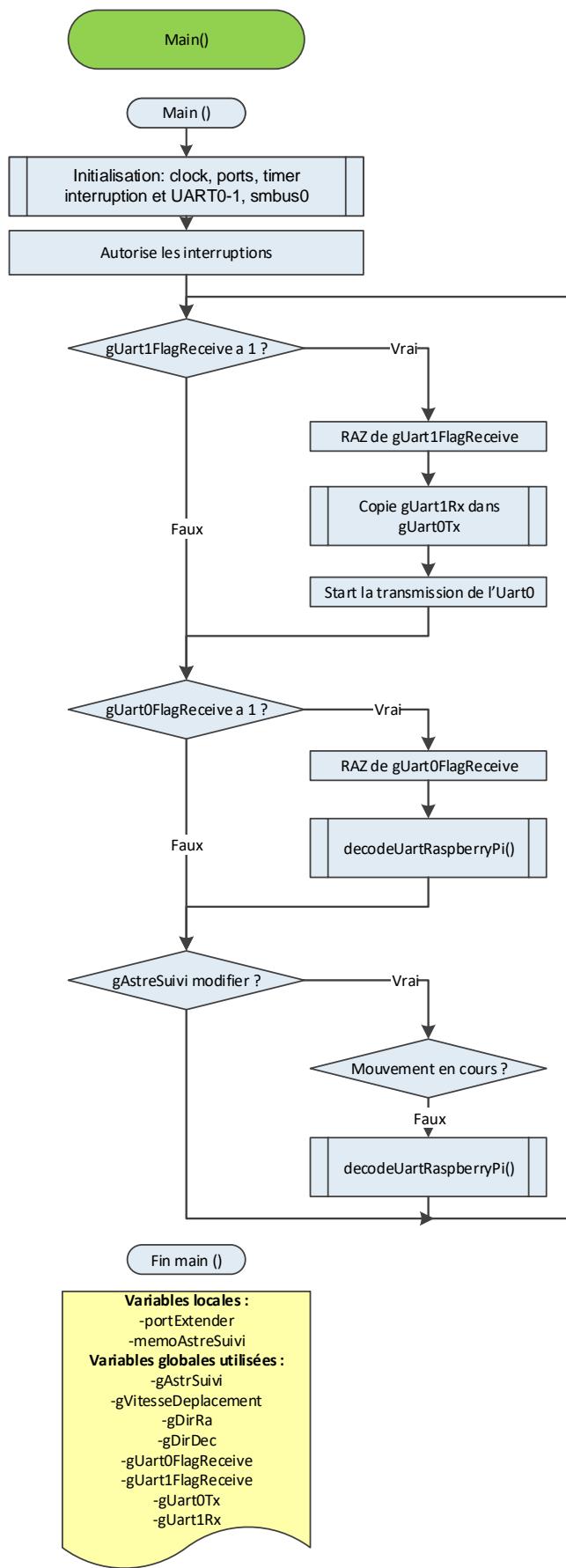
-----*
Descriptif: Applique une vitesse predefinit au moteur et une direction
Entree   :
    - unsigned char vitesse (6 ... 10)
    - bit direction          (0 ... 1)
Sortie   : --
-----*/
void setMotorDEC(unsigned char vitesse,bit direction)
```

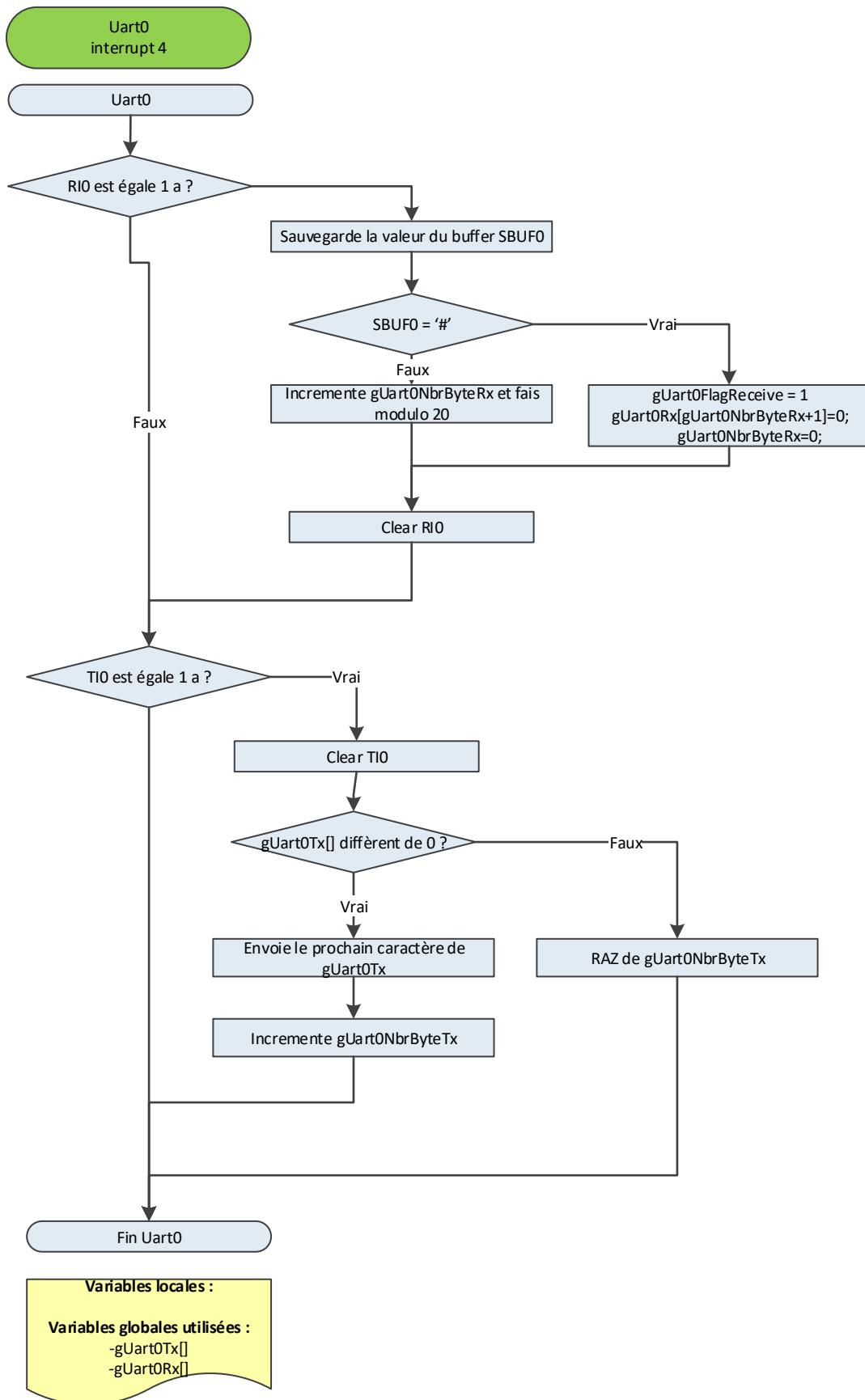
Les paramètres d'entrée pour choisir la vitesse sont définis par les définies suivants

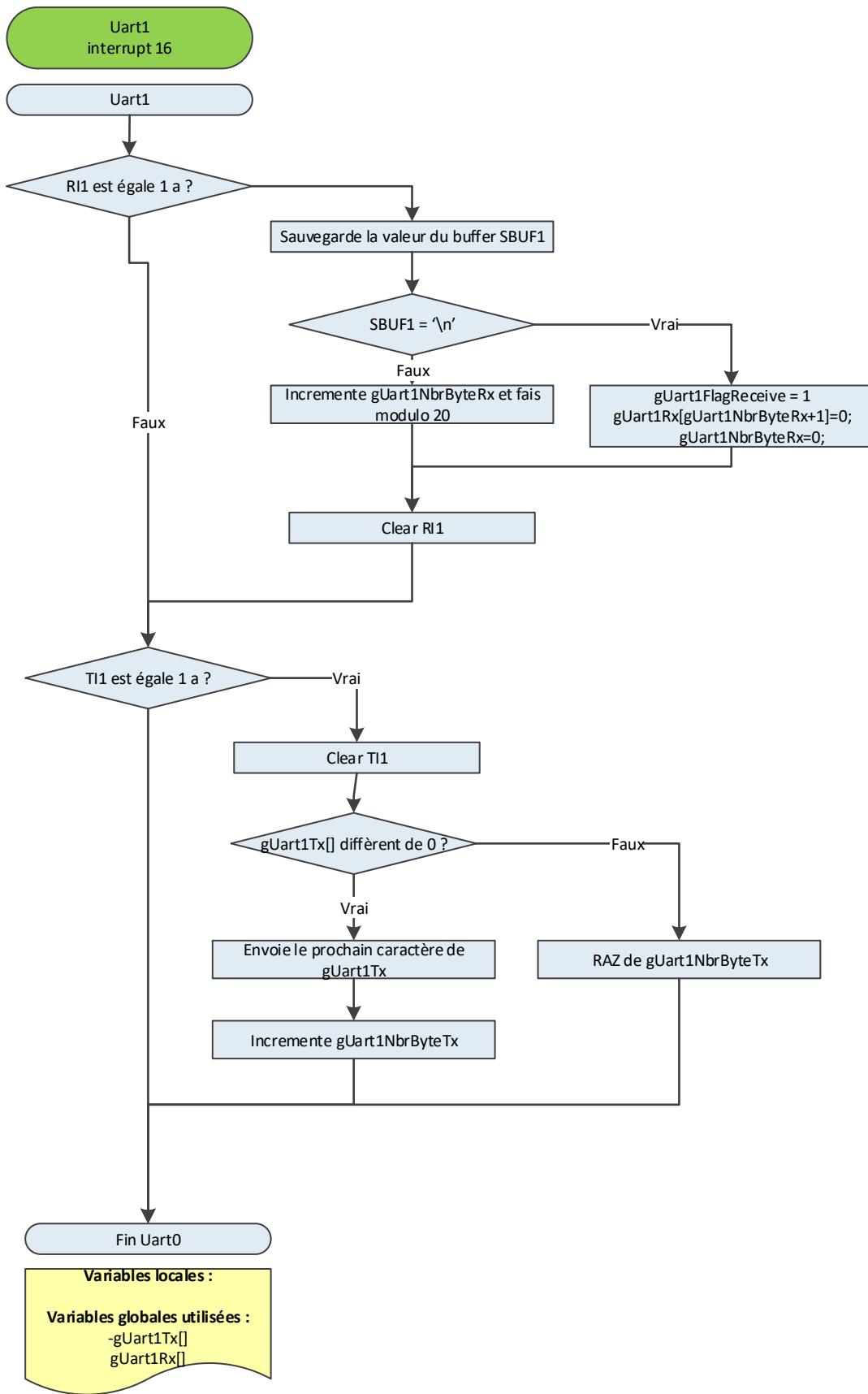
```
#define VITESSE_RA_SIDERAL 0
#define VITESSE_RA_LUNE 1
#define VITESSE_RASOLEIL 2
#define VITESSE_RA_SATURNE 3
#define VITESSE_RA_JUPITER 4
#define VITESSE_RA_ISS 5
#define VITESSE_X05 6
#define VITESSE_X2 7
#define VITESSE_X8 8
#define VITESSE_X16 9
#define VITESSE_MAX 10
```

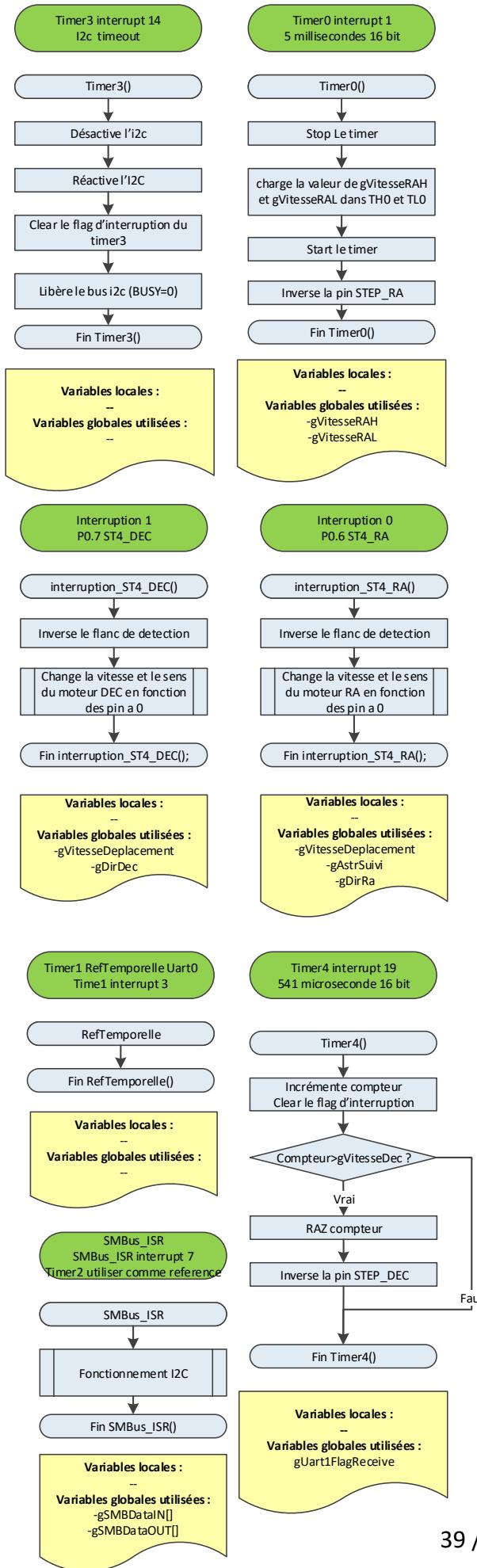
À noter que la vitesse x0,5 est fait pour la régulation de suivi avec PHD2.

4.2.1.2. Analyse du code de gestion de moteur









4.2.2. Programmation carte température

Afin de contrôler la puissance dissipée par les résistances de chauffe. J'utilise le module PCA intégré dans le F38C.

J'ai décidé d'utiliser le PCA en mode 8 bits, car je n'avais pas besoin d'une grande précision.

Pour faire varier le PWM, il suffit d'écrire dans le registre : **PCA0CPH0**



Par exemple, écrire 127 dans le registre aura comme effet de mettre un PWM de 50% en sortie.

Résultat obtenu :

Ch1 : VDS

Ch2 : VGS

Mais il y a deux petites subtilités avec le PCA :

-les valeurs de PCA0CPH0 correspondent à l'inverse du PWM, par exemple PCA0CPH0 = 220 correspond à un PWM de 13.7[%], et PCA0CPH0 = 50 correspond à 80.3[%].

-Pour obtenir un PWM de 0[%], écrire PCA0CPH0=255, ne suffis pas, car le rapport cyclique obtenu est de 0.36[%]. Il faut aussi mettre à zéro le bit ECOM0 du registre PCA0CPM0, en faisant un masquage : $PCA0CPM0 \&= \sim 0x40$;

Les 3 modes de régulation diffèrent :

Mode de régulation PID :

Le mode de régulation PID régule la température automatiquement au-dessus du point de rosée en utilisant le capteur EEH210 présent sur la carte.

```
/*-----*
 * actu_Pca0_Pid ()
 *
 * Descriptif: calcul le pwm a appliquer sur la resistance en fonction du PID
 * Entrée   :
 *           - int tempCapt
 *           - int consigne
 * Sortie   : --
 *-----*/
void actu_Pca0_Pid(int tempCapt,int consigne)
{
    static int pwml=0;
    static unsigned char old_error=0;
    int error=0;
    if(tempCapt<consigne)
    {
        error=consigne-tempCapt;
        pwml=255-((int)error*KP)+((old_error-error)*KD);
        //pwml=0;//mode tout ou rien
    }
    else
    {
        pwml=255;//0%
        error=0;
    }
    old_error=error;
    if(pwml<=0)
    {
        PCA0CPM0|=0x40;//reactive le pca0 si il a ete desactiver
        PCA0CPH0=0;
    }
    else if(pwml>=255)
    {
        PCA0CPH0=255;
        PCA0CPM0&=~0x40;//Clear ecom0 afin d'obtenir 0%
    }
    else
    {
        PCA0CPH0=pwml;
    }
}
```

Cette fonction doit être appelée de manière périodique afin que le facteur dérivé puisse fonctionner.

J'ai créé des define afin de choisir plus facilement les paramètres liés à la régulation PID :

```
#define NMESURE 10 //nombre de mesure a faire pour une moyenne
#define DELAI_MESURE 10//temps entre chaque mesure * 50[ms]

#define KP 30
#define KI 0
#define KD 15
```

Il existe une fonction pour chaque résistance de chauffe (4 aux totales).

Pour cadencer les mesures, le define DELAI_MESURE est utilisée par le timer0.

Mode de régulation avec potentiomètre :

Lorsque les mesures pour les potentiomètres sont terminées, le flag gFlagPot est mis à 1 afin de calculer les nouvelles valeurs à appliquer sur les résistances de chauffe.

```
if(gFlagPot)
{
    gFlagPot=0;
    if(gModeRegulation==MODE_POT)
    {
        PCAOPH0=ConvertToPca(gMesure[POT_0][0]);
        PCAOPH1=ConvertToPca(gMesure[POT_1][0]);
        PCAOPH2=ConvertToPca(gMesure[POT_2][0]);
        PCAOPH3=ConvertToPca(gMesure[POT_3][0]);
    }
}
```

L'ADC est configuré en mode 10 bits, donc les valeurs varient entre 0 et 1023, la fonction ConvertToPca effectue simplement une règle de trois, pour ramener la valeur entre 0 et 255.

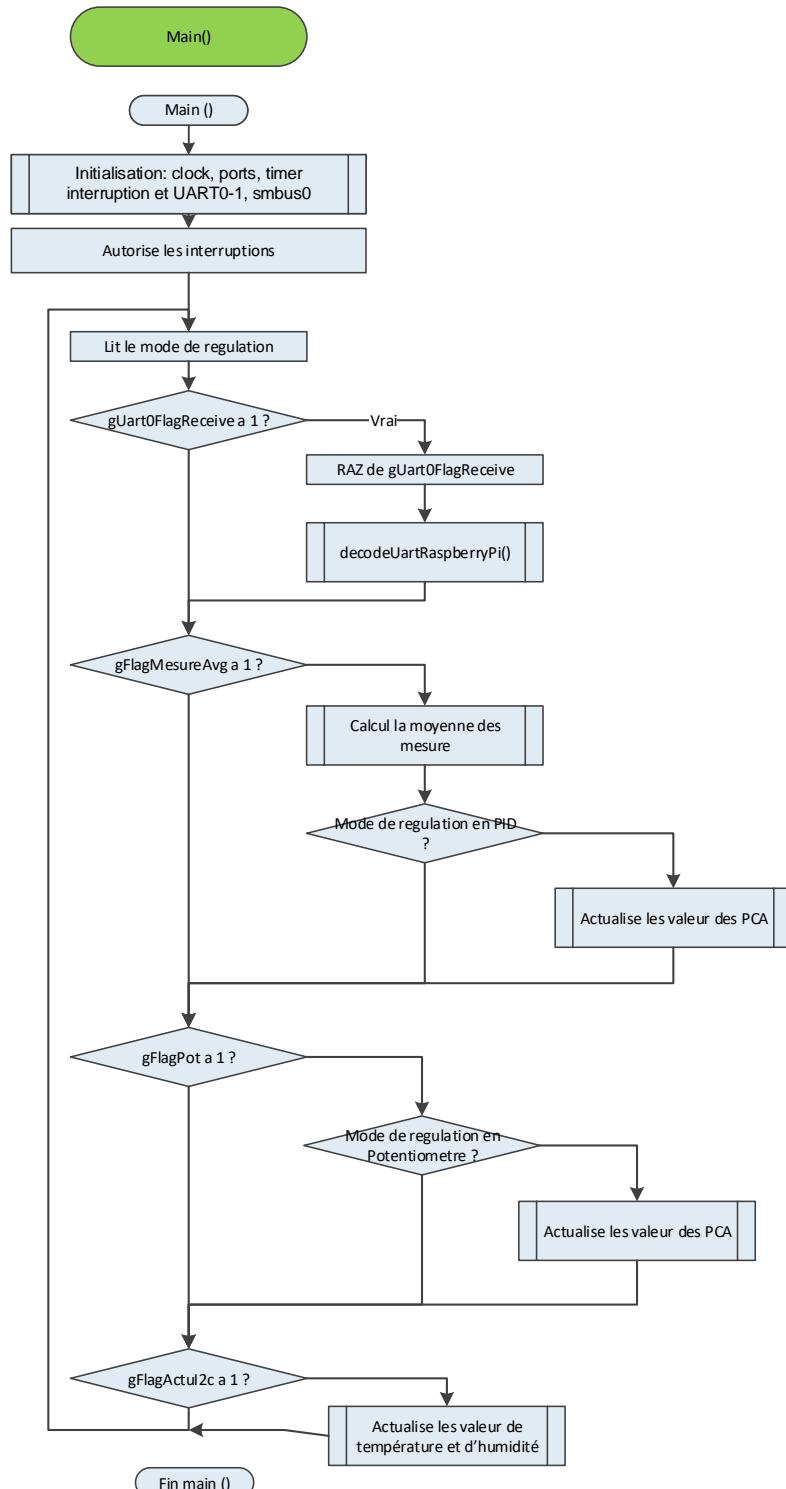
```
/*
 *-----*
 * ConvertToPca ()
 *-----*
 * Descriptif: Convertie une valeur entre 0 et 1023, de 0 a 255
 * Entrée   : unsigend int adc (0 ... 1023)
 * Sortie    : unsigend char (0 ... 255)
 *-----*/
unsigned char ConvertToPca(unsigned int adc)
{
    return 255-((unsigned long)adc*255)/1023;
}
```

Mode de régulation avec la page Web :

Lorsqu'une trame est reçue sur l'UART, elle est décodée par la fonction decodeUart0().

```
if( (gUart0Rx[0]=='R') && (gUart0Rx[1]=='0') && (gUart0Rx[2]=='_'))
{
    testPwm=(gUart0Rx[3]-'0')*100;
    testPwm+=(gUart0Rx[4]-'0')*10;
    testPwm+=gUart0Rx[5]-'0';
    if(testPwm>255)
    {
        //Erreur
        testPwm=255;
    }
    testPwm=255-testPwm;
    if(testPwm==255)//0%
    {
        PCAOPH0=0xFF;
        PCAOPM0&=~0x40;//RAZ de ECOMn pour forcer le pwm a 0%
    }
    else
    {
        PCAOPH0=testPwm;
        PCAOPM0|=0x40;//ECOMn a 1 pour autoriser le fonctionnement
    }
}
```

4.2.2.1. Analyse de la carte de gestion de température

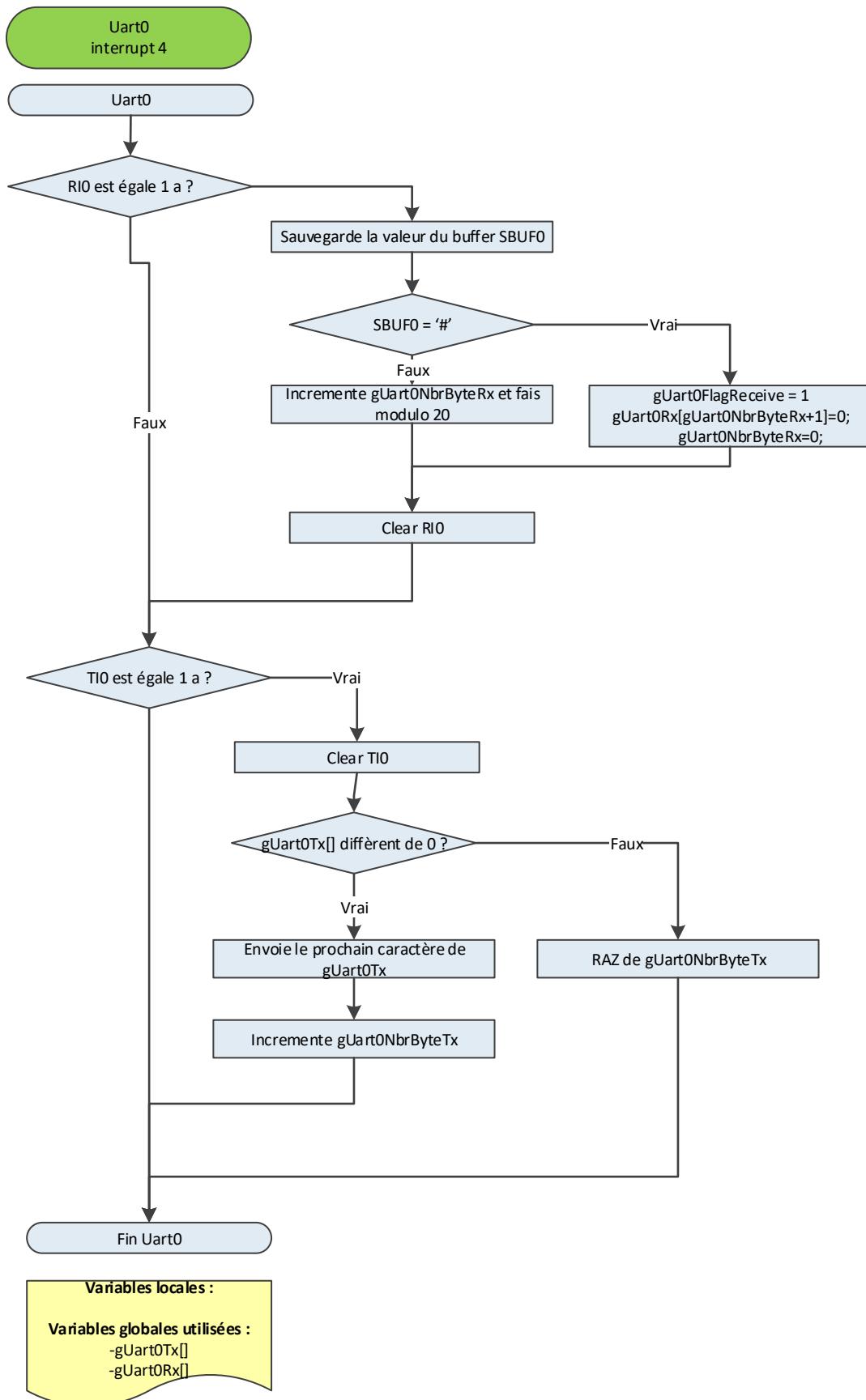


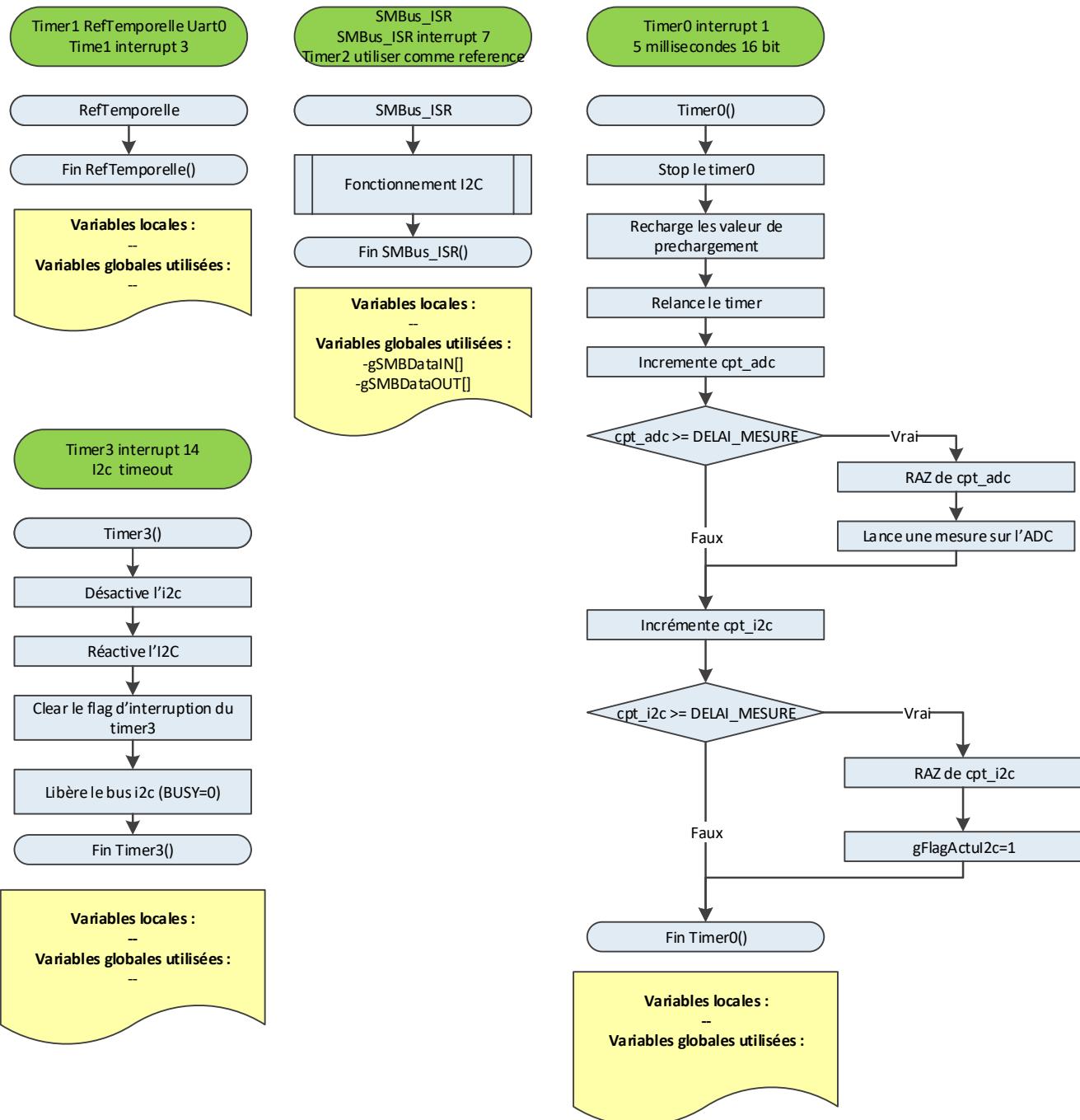
Variables locales :

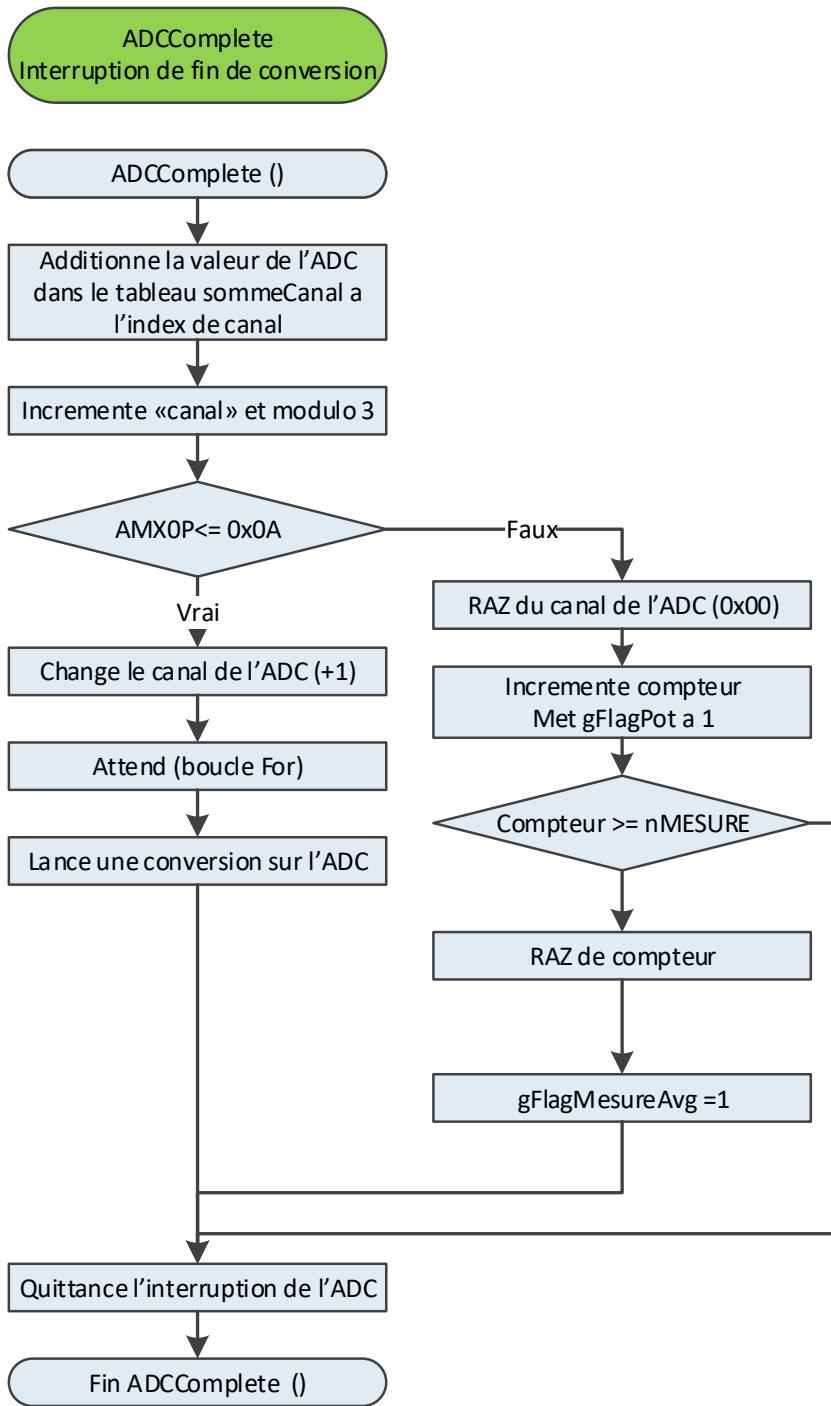
- tempSensor
- temp_ros

Variables globales utilisées :

- gUart0FlagReceive
- gModeRegulation
- gFlagMesureAvg
- gFlagPot
- gFlagActu12c







Variables locales :

Variables globales utilisées :

4.2.3. Protocole de communication

Afin de communiquer entre les différentes cartes, j'utilise l'UART intégrer à la Raspberry pi et aux microcontrôleurs. Comme écrit dans le schéma bloc au chapitre 3.3, le microcontrôleur qui gère les moteurs sert de passerelle entre la Raspberry pi et le microcontrôleur qui contrôle les moteurs.

Lorsque la carte de gestion de moteur reçoit une commande qu'il ne comprend pas, il fait suivre à la carte de gestion de température.

Les paramètres de la communication UART sont les suivantes : 57600 baud, 8 bit, 1 start bit et 1 stop bit, sans parité.

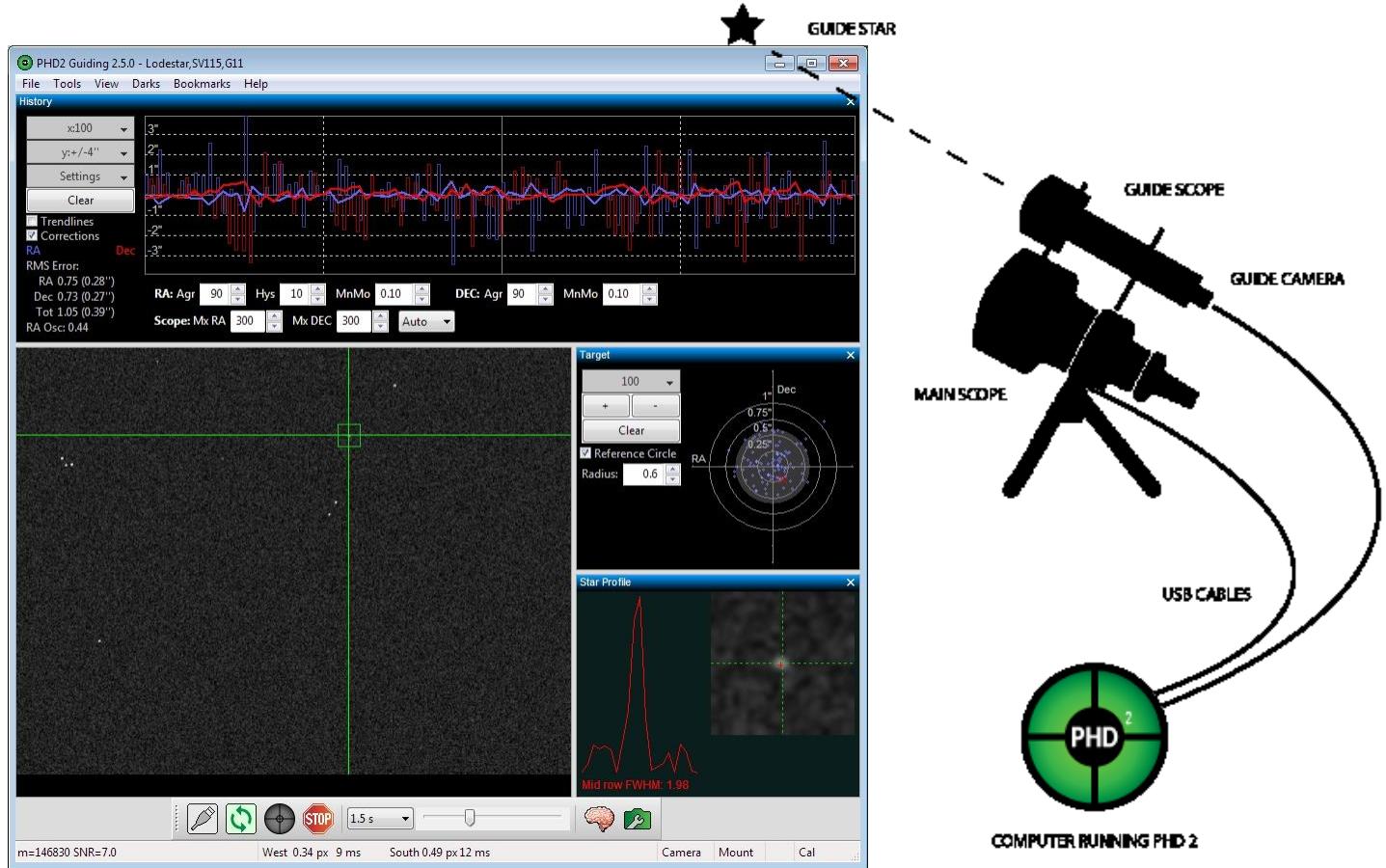
Les commandes envoyées par la Raspberry Pi sont les suivantes :

Commande	Cible	Réponse	Descriptif
RA+#	Carte Moteur	--	Démarre un déplacement du moteur RA dans le sens du ciel
RA0#	Carte Moteur	--	Stop le déplacement du moteur RA (mais laisse active la vitesse de suivi du ciel)
RA-#	Carte Moteur	--	Démarre un déplacement du moteur RA dans le sens inverse du ciel
DEC+#	Carte Moteur	--	Démarre un déplacement du moteur de déclinaison en positif
DEC0#	Carte Moteur	--	Stop le moteur de déclinaison
DEC-#	Carte Moteur	--	Démarre un déplacement du moteur de déclinaison en négatif
RO_XXX#	Carte Température	--	Choisis la valeur du PWM de la sortie 0 0 → 0 [%] 255 → 100[%]
R1_XXX#	Carte Température	--	Idem que RO_XXX
R2_XXX#	Carte Température	--	Idem que RO_XXX

R3_XXX#	Carte Température	--	Idem que R0_XXX
GET_T0#	Carte Température	+XX.X\n → +21.3\n → 21.3[°C] -XX.X\n → -05.4\n → -5.4[°C]	Demande la température ambiante
GET_T1#	Carte Température	Idem que GET_T0#	Demande la température du capteur 0
GET_T2#	Carte Température	Idem que GET_T0#	Demande la température du capteur 1
GET_T3#	Carte Température	Idem que GET_T0#	Demande la température du capteur 2
GET_T4#	Carte Température	Idem que GET_T0#	Demande la température du capteur 3
GET_TR#	Carte Température	Idem que GET_T0#	Demande la température du point de rose
GET_H#	Carte Température	XXX\n → 031\n → 31[%]	Demande l'humidité ambiante

Chaque trame envoyée par la Raspberry Pi se termine par un #, et chaque donnée reçue se termine par un \n.

4.3. PHD2 Guiding



C'est le programme dont je me sers pour avoir un suivi des Étoiles pour l'astrophotographie. Il me permet d'avoir un suivi beaucoup plus précis, car il fait une régulation de suivi.

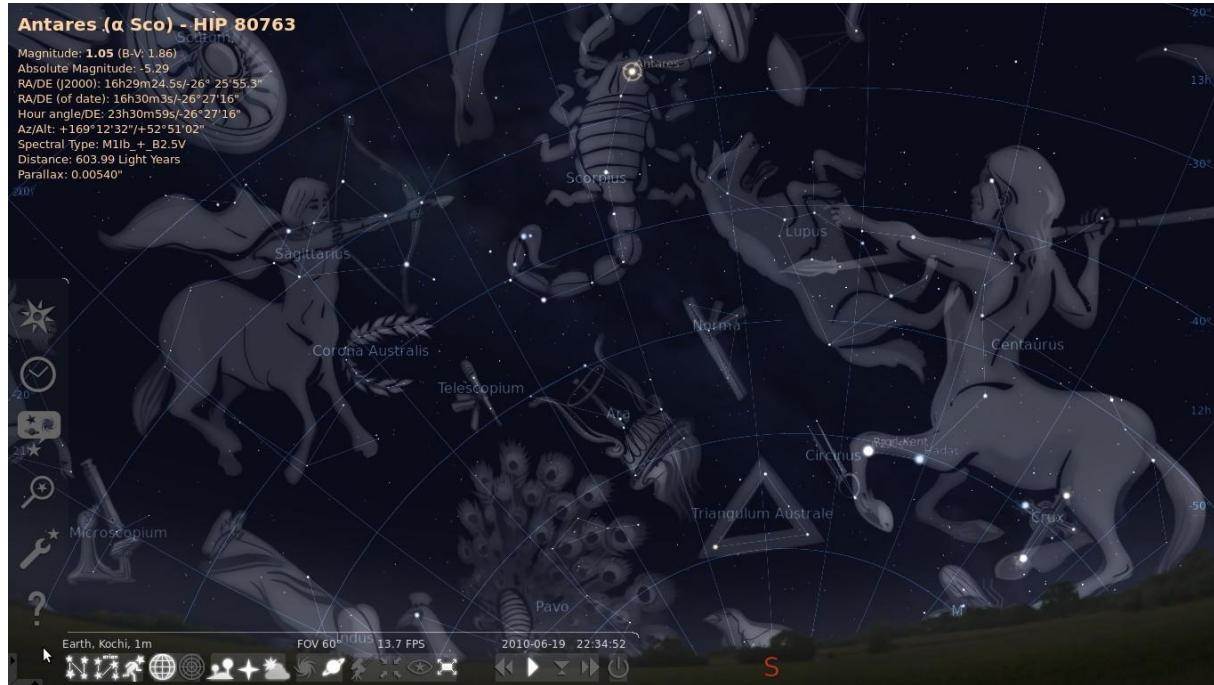
Il suffit d'avoir une étoile dans le champ de la caméra, ensuite il faut calibrer le système pour le suivi de l'Étoile.

C'est à ce moment que la Raspberry pi prend de l'importance, car sans elle j'ai besoin d'un ordinateur pour utiliser PHD2 guiding.

Car le suivi du ciel en utilisant seulement le C8051F38C n'est pas assez précis pour faire de belle photo.

4.4. Télescope Goto

C'est un télescope qui se pointe tout seul vers un objet choisi (Étoile, galaxie, nébuleuse), en utilisant ces coordonnées, qui peuvent être fournis par un planétarium, comme stellarium :



Stellarium se connecte au télescope par le port 10001.

Afin de programmer le GOTO, je suis reparti de ce projet sur GitHub :

https://github.com/Svenito/Pyscope/blob/master/tcp_test.py

Ce projet ne fait que réceptionner et transmettre les données de position à stellarium.

Si je connais la distance entre les objets dans le ciel et leur position, il est possible de calculer le temps durant lequel le moteur doit fonctionner afin de se déplacer sur un nouvel objet.

Par exemple pour se déplacer entre Deneb et Vega, avec une vitesse de 48x la vitesse sidérale, il faut que le moteur d'ascension droite fonctionne pendant 124158 [ms], et le moteur de déclinaison doit fonctionner pendant 26349[ms] d'après les calculs de mon programme python.

Remarque :

Comme le ciel continue de bouger en même temps que le télescope se déplace, il faut le prendre en compte dans les calculs, c'est pour cela que si l'axe d'ascension droite doit se déplacer dans le sens du ciel, la vitesse sera de 49x, alors que dans le sens inverse, il aura une vitesse de 47x. Le moteur de l'axe de déclinaison n'est pas concerné.

```
M_PI = 3.1415926535897932385
MOVE_SPEED = 48
flagPosition=1
UNITE_DEPLACEMENT = 49843#correspond a 15.04"arc
DEPLACEMENT_SECONDS_DEC = UNITE_DEPLACEMENT * MOVE_SPEED
DEPLACEMENT_SECONDS_RA_POS = UNITE_DEPLACEMENT * (MOVE_SPEED+1)#le ciel continue de bouger en meme temps
DEPLACEMENT_SECONDS_RA_NEG = UNITE_DEPLACEMENT * (MOVE_SPEED-1)
serial = serial.Serial("/dev/ttyS0",57600,timeout=2)#

```

Calcul du système Goto :

Les coordonnées envoyées par Stellarium ne sont pas dans un format conventionnel,

Par exemple la position de Deneb est :

Ascension droite : 20h 41m 25.915s

Déclinaison : +45° 16' 49,22"

Mais les données reçues sont les suivantes :

Ascension droite : 3702715063

Déclinaison : 540215608

Les coordonnées pour Vega sont les suivantes :

Ascension droite : 18h 36m 56,3364s

Déclinaison : +38° 47' 01,291"

Mais les données reçues sont les suivantes :

Ascension droite : 3331407792

Déclinaison : 462729437

Pour déterminer le temps de déplacement, il me fallait savoir à quoi correspondait les valeurs reçues, alors j'ai demandé à stellarium de m'envoyer les coordonnées suivantes :

Ascension droite : 00h 00m 1s afin de connaître l'unité d'une seconde.

La valeur d'une seconde correspond à 49843.

Ensuite il faut calculer le déplacement effectué par le moteur pendant une seconde :

(Le calcul suivant sera fait pour l'axe de déclinaison, mais les calculs sont presque les mêmes pour l'axe d'ascension droite, il faut simplement prendre en compte le déplacement du ciel)

$$Deplacement_{sec} = 49843 * 48 = 2392464$$

Ensuite il faut calculer le déplacement à effectuer :

$$Deplacement = Pos1 - Pos2 = 540215608 - 462729437 = 77486171$$

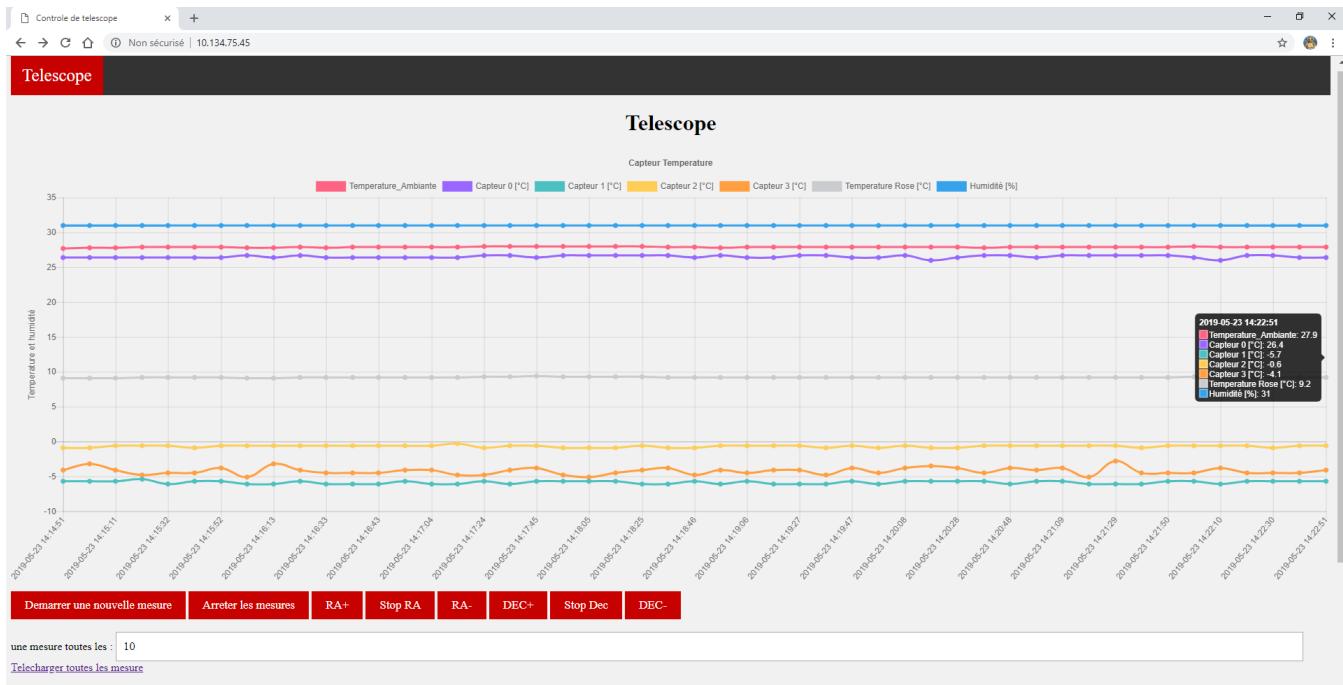
Il faut maintenant calculer le temps de déplacement :

$$TDeplacement = \frac{Deplacement * 1000}{DeplacementSec} = \frac{77486171 * 1000}{2392464} = 32387[ms]$$

Remarque : le *1000 permet d'obtenir le résultat en milliseconde.

Malheureusement lors de mes tests le télescope ne pointait pas au bon endroit, je n'ai pas réussi à trouver d'où venait le problème. Je n'ai pas essayé de faire fonctionner le système GOTO, car ce n'était pas demander dans le projet.

4.5. Interface Web



L'interface web permet de contrôler le télescope, à distance, mais elle peut aussi être affiché sur l'écran de la Raspberry pi.

L'interface web contient des boutons qui permettent de déplacer le télescope, ainsi que des boutons afin de contrôler les mesures de température et d'humidité en cours :



Les boutons RA+ et RA- font bouger le télescope sur l'axe d'ascension droite, et dans la direction choisie. Les bouton Stop RA et Stop DEC arrêtent le mouvement du télescope.



Les curseurs disponibles sur la page permettent de choisir les PWM à appliquer sur les résistances de chauffe. Les valeurs varient entre 0 et 255.

0 équivaut à éteindre la résistance de chauffe, et à l'inverse 255 l'active au maximum.

Programmation de la page :

Afin de ne pas recommencer de zéro, j'ai réutilisé la page HTML sur laquelle j'avais travaillé durant le troisième travail de semestre (armoire thermique).

Le code était déjà presque entièrement écrit, j'ai simplement dû ajouter des boutons, ainsi que le défilement des données sur le graphique (les dernières données ne disparaissaient pas, mais s'entassaient).

J'ai aussi amélioré le temps de réaction du site web, en modifiant le délai utiliser, car avant le délai utiliser, était celui contenu dans le fichier config, donc le délai était bloquant.

Le serveur sur le Raspberry pi utilise un fichier config afin de faire coexister les deux programmes python.

Le programme Mesure_Process.py gère les mesures sur l'armoire, et les écrit dans un fichier de mesure csv. Il réagit à ce qui est écrit dans le fichier confins.

Le fichier config.cfg est un fichier json, son contenu se présente comme cela :

```
{"Delai": "10", "ChangeRA": false, "ChangeDEC": false, "FileName": "2019-05-23  
10:34:35.csv", "CreateNewFile": false, "State": true, "ConsigneChange": false, "Consigne1":  
"R1_158#", "Consigne0": "R0_255#", "Consigne3": "R3_253#", "Consigne2": "R2_161#",  
"MoveRA": "RA0#", "MoveDEC": "DEC0#"}
```

Nom du Paramètre	Description	Valeur
Délai	Permet de choisir le délai entre chaque mesure	1...2...3...
ConsigneChange	Indique à Mesure_Process.py qu'une des valeur d'un des slider a été modifiée. Et envoie la valeur de consigne0,1,2,3	true, false
FileName	Le nom du fichier .csv dans le quelle les mesures doivent être écrite.	EX : 2019-05-23 10_34_35.csv
CreateNewFile	Indique à Mesure_Process.py qu'il faut créer un nouveau fichier de mesure	true, false
State	Indique si des mesures sont en cours	true, false
ChangeRA	Indique à Mesure_Process.py qu'il doit faire agir sur le moteur d'ascension droite	true, false
MoveRA	Contiens la commande à envoyer au moteur	« RA+# »« RA0# »« RA-# »
ChangeDEC	Indique à Mesure_Process.py qu'il doit faire agir sur le moteur de déclinaison	true, false
MoveDEC	Contiens la commande à envoyer au moteur	« DEC+#»« DEC0#»« DEC-#»
Consigne0	Contiens la valeur du pwm de la résistance 0	R0_XXX#
Consigne1	Contiens la valeur du pwm de la résistance 1	R1_XXX#
Consigne2	Contiens la valeur du pwm de la résistance 2	R2_XXX#
Consigne3	Contiens la valeur du pwm de la résistance 3	R3_XXX#

4.5.1. HTML – JavaScript – Ajax - Python

Le code HTML est vraiment très court, la plus grande partie du code sont les scripts de la page et dans le fichier css.

```
</head>
<body>
    <ul>
        <li><a class = "active" href="">Telescope</a></li>
    </ul>
    <h1>Telescope</h1>
    <canvas id="canvas" width="300" height="100"></canvas>
    <button id="CreateNewFile" >Demarrer une nouvelle mesure</button>
    <button id="StartMesure" >Demarrer les mesures</button>
    <button id="RA_Pos" >RA+</button>
    <button id="RA_Stop" >Stop RA</button>
    <button id="RA_Neg" >RA-</button>
    <button id="DEC_Pos" >DEC+</button>
    <button id="DEC_Stop" >Stop Dec</button>
    <button id="DEC_Neg" >DEC-</button><br>
    <br>
    une mesure toutes les :
    <input type="number" name="nMesure" id="nMesure" min="1" max="300" width="50"><br>
    <a href="/Mesure">Telecharger toutes les mesure</a><br>
    <h2>Pwm 0 :</h2>
    <input type="range" min="0" max="255" value = "255" class="slider" id="sliderPwm0"></input>
    <span id="valuePwm0">Pwm0 : </span>

    <h2>Pwm 1 :</h2>
    <input type="range" min="0" max="255" value = "255" class="slider" id="sliderPwm1"></input>
    <span id="valuePwm1">Pwm1 : </span>

    <h2>Pwm 2 :</h2>
    <input type="range" min="0" max="255" value = "255" class="slider" id="sliderPwm2"></input>
    <span id="valuePwm2">Pwm2 : </span>

    <h2>Pwm 3 :</h2>
    <input type="range" min="0" max="255" value = "255" class="slider" id="sliderPwm3"></input>
    <span id="valuePwm3">Pwm3 : </span>
</body>
</html>
```

Extrait du code JavaScript de la page :

Ce code est appelé lorsque le slider de choix du pwm de la résistance 0 est modifié.
Il envoie une requête ajax de type POST avec la valeur du PWM choisi au serveur flask sur python.

```
$( 'input#sliderPwm0' ).click(function()
{
    $.ajax
    ({
        url: "/Telescope/Set_Pwm0/",
        type: "POST", //send it through get method
        data:{"SliderValue": document.getElementById("sliderPwm0").value},
        success: function(response)
        {
            valueConsigne0 = Math.round(document.getElementById("sliderPwm0").value);
            document.getElementById("valuePwm0").innerHTML = valueConsigne0;
        }
    });
});
```

La valeur du curseur est récupérée par cette fonction dans WebServer.py et elle est écrite dans le fichier config.cfg

```
@app.route("/Telescope/Set_Pwm0/", methods = ['POST'])
def Set_Pwm0():
    value = int(request.form.get("SliderValue"))
    with open('Config.cfg') as f:
        config = json.load(f)
        f.close()
    config["Consigne0"] = "R0_{" + '{:03d}'.format(value) + "}"
    config["ConsigneChange"] = True
    with open('Config.cfg', 'w') as outfile:
        json.dump(config, outfile)
        outfile.close()
    return str(1)
```

Ensuite le programme Mesure_Process.py actualise la température de l'armoire

```
if (config["ConsigneChange"] == True):
    print "Changement de consigne"
    serial.write(config["Consigne0"].encode())
    time.sleep(0.01)
    serial.write(config["Consigne1"].encode())
    time.sleep(0.01)
    serial.write(config["Consigne2"].encode())
    time.sleep(0.01)
    serial.write(config["Consigne3"].encode())
    config["ConsigneChange"] = False
    with open('Config.cfg', 'w') as outfile:
        json.dump(config, outfile)
        outfile.close()
```

4.5.2. Installation de la Raspberry Pi

4.5.2.1. VNC

VNC permet de contrôler la Raspberry Pi à distance en utilisant une connexion internet. Afin de l'utiliser, il n'y a pas besoin de l'installer, car il est installé par défaut sur la Raspberry Pi.

Pour l'utiliser, il suffit de l'activer :

- Cliquer sur le Menu Préférence
- Configuration du Raspberry Pi
- Interface
- VNC Activer
- Valider

4.5.2.2. Écran

Inverser l'écran :

Lorsque la Raspberry Pi démarre pour la première fois, l'écran sera à l'envers par rapport à la carte mère, il est nécessaire de retourner l'écran afin de l'avoir dans le bon sens.

Afin de retourner l'écran il faut faire cette manipulation dans le terminal :

>sudo nano /boot/config.txt

Ajouter la ligne lcd_rotate=2 en haut du fichier ouvert.

Appuyer sur CTRL+X afin de sauvegarder et de quitter, faites « Y » afin de valider puis « ENTER » pour valider le nom du fichier à écrire.

Changer la luminosité :

La ligne de commande à entrer dans le terminal pour modifier la luminosité est la suivante :

> sudo bash -c "echo xxx > /sys/class/backlight/rpi_backlight/brightness"

Remarque : xxx dans la ligne de commande doit être remplacer par une valeur entre 0 et 255

Ex :

Luminosité à 100% :

> sudo bash -c "echo 255 > /sys/class/backlight/rpi_backlight/brightness"

Luminosité à 0% :

> sudo bash -c "echo 0 > /sys/class/backlight/rpi_backlight/brightness"

Mode Petit Ecran :

Menu → Preference → Appearance Settings → Default → For Small Screen → Set default

4.5.2.3. Port série

```
>sudo raspi-config  
>5 Interfacing Option  
>P6 Serial  
>Would you like a login shell to be accessible over serial?  
>No  
>Would you like the serial port hardware to be enabled?  
>Yes
```

The serial login shell is disabled

The serial interface is enabled

4.5.2.4. Clavier virtuelle

Par defaut il n'y a pas de clavier virtuelle d'installer, et comme ma raspberry pi a un écran tactile intégrer, je n'ai pas besoin de souris.

Afin d'installer le clavier virtuel, il suffit d'entrer cette ligne de commande dans le terminal :

> sudo apt-get install matchbox-keyboard

Puis redémarrer.

Le clavier peut être ouvert en allant dans le menu accessoire puis Keyboard.



4.5.2.5. Librairies Python

L'utilisation de la page web nécessite l'installation de pandas pour python.

Afin de l'installer, il suffit d'écrire cette ligne de commande dans le terminal :

> sudo apt-get install python-pandas

4.5.2.6. Script Python au démarrage

Entrer cette ligne de commande :

>sudo nano /etc/rc.local

Puis ajouter ces lignes en bas du fichier :

cd /home/pi/Desktop/Web_Telescope/

sudo python Mesure_Process.py&

sudo python WebServer.py&

4.5.2.7. PHD2

Installation de PHD2

```
> sudo apt-get install build-essential git cmake pkg-config libwxgtk3.0-dev wx-common  
wx3.0-i18n libbindi-dev libnova-dev gettext zlib1g-dev libx11-dev libcurl4-gnutls-dev
```

```
> apt-get install packaging-dev  
>sudo apt-get install cmake  
>sudo apt-get install curl  
>sudo apt-get install libcurl4-openssl-dev  
>sudo apt-get install libwxgtk2.8-dev  
> sudo apt-get install libwxgtk3.0  
>sudo apt-get install gcc-6 g++-6  
>git clone https://github.com/OpenPHDGuiding/phd2.git  
>cd phd2  
>mkdir -p tmp  
>cd tmp  
>cmake -DCMAKE_C_COMPILER=gcc-6 -DCMAKE_CXX_COMPILER=g++-6 ..  
>make
```

Une fois la compilation terminer, il ne reste plus qu'as lancé phd2 en utilisant ces lignes de commande :

```
>cd phd2  
>cd tmp  
>./phd2.bin
```

Installation du driver pour la camera StarShoot Autoguidier :

Télécharger le fichier libbindi_1.7.4_rpi.tar.gz sur le site :

<https://www.indilib.org/download/all.html>

Ensuite il faut entrer ces ligne de commande afin d'installer le driver.

```
> sudo apt-get install cdbs libcfitsio-dev libnova-dev libusb-1.0-0-dev libjpeg-dev libusb-dev  
libtiff5-dev libftdi1-dev fxload libkrb5-dev libcurl4-gnutls-dev libraw-dev libgphoto2-dev  
libgsl-dev dkms libboost-regex-dev libgps-dev libdc1394-22-dev
```

```
> cd Downloads/  
> tar -xzf libbindi_1.7.4_rpi.tar.gz  
> cd libbindi_1.7.4_rpi  
> sudo dpkg -i *.deb  
>reboot
```

Créer un exécutable qui lance phd2 depuis le bureau :

Cree un fichier .sh avec un éditeur de texte, et y coller ces lignes de commande :

```
cd ~/phd2/tmp/ && ./phd2.bin
```

Sauvegarder le fichier sur le bureau et nommez le PHD2.sh, puis ouvrez une fenêtre de terminal, ensuite naviguer jusqu'au bureau avec la commande *cd Desktop* puis utiliser la commande *sudo chmod 777 PHD2.sh* pour crée l'exécutable.

Maintenant il suffit de cliquer sur le fichier pour l'ouvrir.

4.6. Tests fonctionnels

Le mode de régulation de température fonctionne, mais n'a été testé en condition réelle, car les tests ont été effectués seulement dans l'armoire thermique de l'atelier. La température de l'armoire thermique varie beaucoup plus vite que la température en condition réelle.

Le suivi des étoiles et de Jupiter fonctionne, je n'ai pas eu le temps de tester le suivi du soleil, de la lune et de saturne.

Le suivi du ciel avec la caméra fonctionne, mais il reste encore certain paramètre à améliorer, car le suivi n'est pas encore parfait.

Exemple de photo prise avec le suivi par caméra sur M57 (La nébuleuse de la Lyre) :

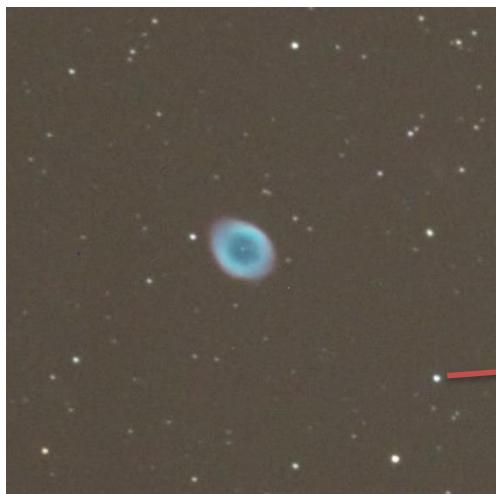
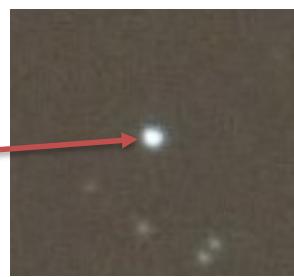


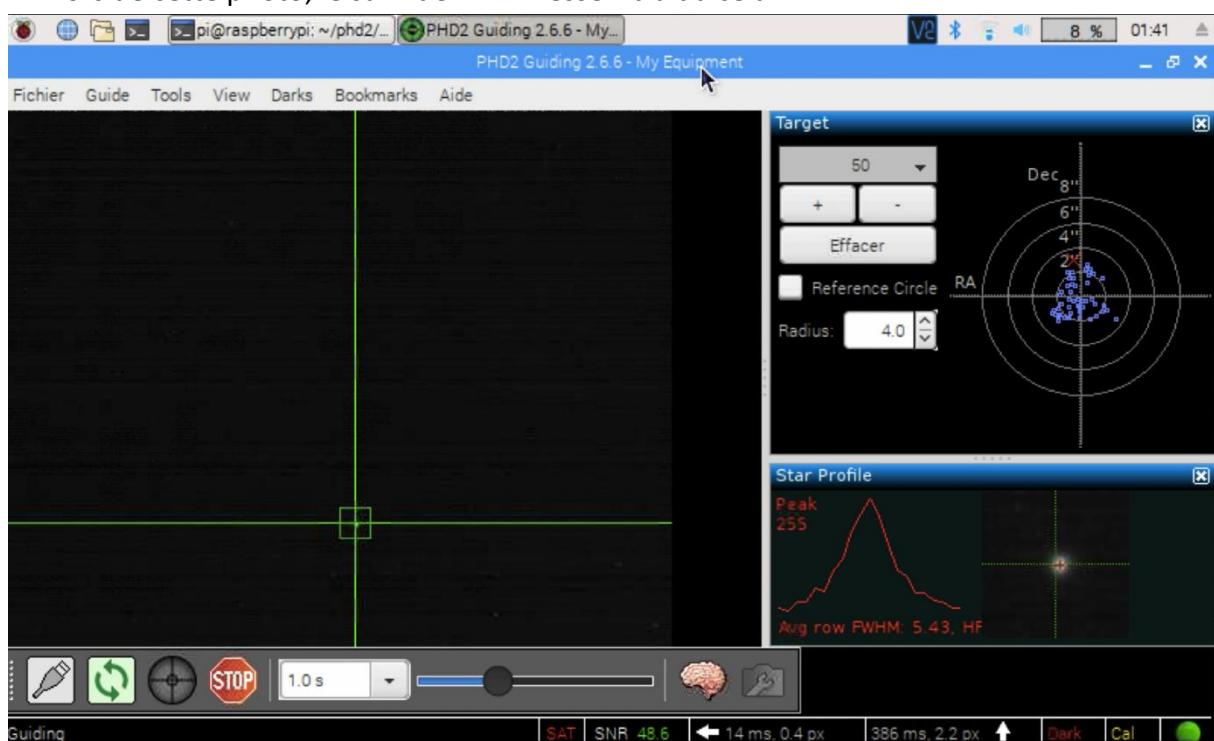
Photo brute prise le 3 juin 2019 à 1h41

1 minute de pose à 1600 ISO avec un télescope de 200mm d'ouverture et 1000mm de focale
(DSC_8399.NEF)

Sur cette image le suivi a bien fonctionné.



Lors de cette photo, le suivi de PHD2 ressemblait à cela :



Les points bleus dans la cible à droite représentent la position de l'étoile depuis les 50 dernières secondes.

En revanche sur cette image le suivi n'a pas très bien fonctionné :

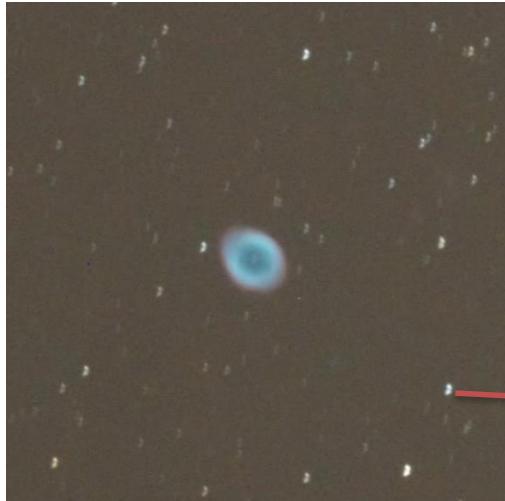
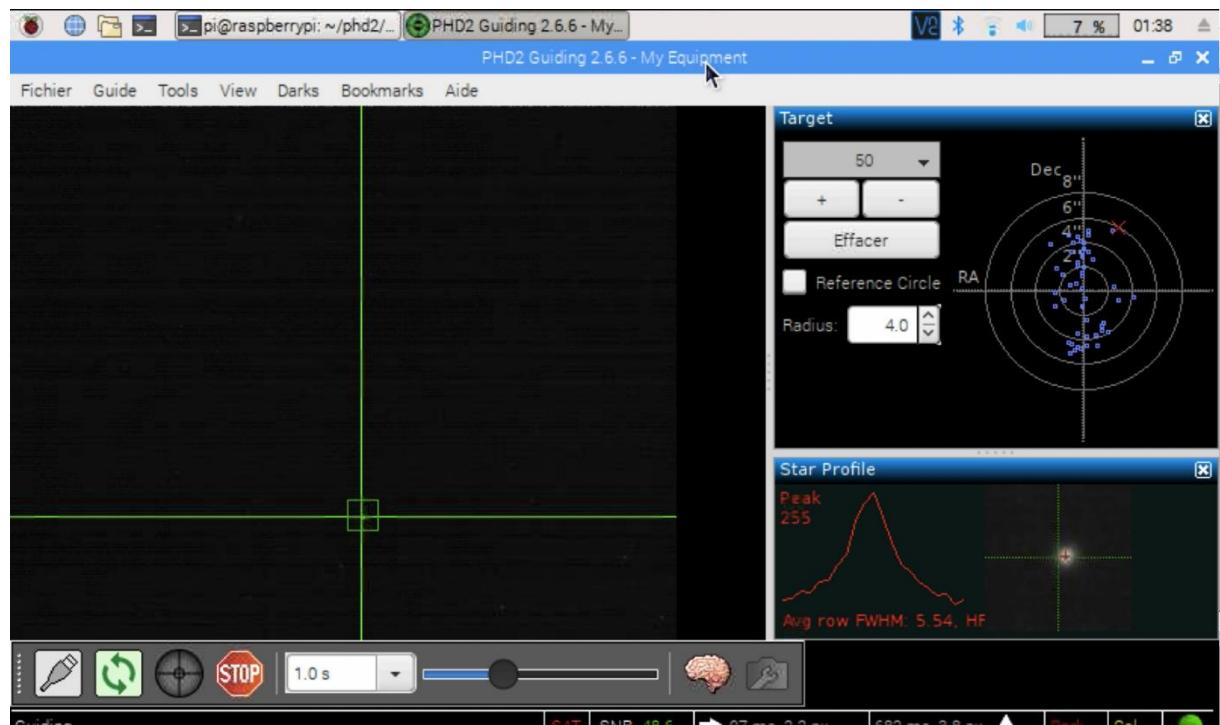


Photo brute prise le 3 juin 2019 à 1h38
1 minute de pose à 1600 ISO avec un télescope de 200mm d'ouverture et 1000mm de focale
(DSC_8396.NEF)

Les étoiles sur cette image ne sont pas rondes :



Lors de cette photo, le suivi de PHD2 ressemblait à cela :



Sur cette image les étoiles ont bougé, cela peut aussi être observé sur PHD2, car les points bleus ne sont pas au centre de la cible, mais bougent beaucoup autour de l'axe de déclinaison.

La différence entre ces deux images peut venir de beaucoup de chose, par exemple :

- un coup de vent qui a fait bouger le tube.
- Du jeu dans la mécanique.
- Des vibrations générées par les moteurs sur le tube.
- Un problème de code dans le microcontrôleur.

5. Conclusion

5.1. Aspects techniques sur le projet

Fonctions techniques	Pourcentage réalisé par catégorie			Remarques / commentaires
	HW	SW	Doc	
Contrôle des moteurs	0	0	0	
Gestion de la température	20	20	0	Récupérer le capteur EEH210 du 2e travail de semestre
Page Web	100	70	60	Récupérer la page web du 2e travail de semestre.
Système Goto	0	0	0	Le système n'était pas demandé dans l'énoncé.
Installation de PHD2	100	60	10	

Future Amélioration pour la carte moteur :

- ajouter plusieurs vitesses différentes pour la régulation, en plus de la vitesse x0,5. Par exemple une vitesse x0,4 sur l'axe de déclinaison, pour voir si les oscillations sont réduites. Ainsi que des vitesses pour les différentes planètes du système solaire.
- Modifier le mode de fonctionnement des moteurs pour passe du mode demi-pas, à $\frac{1}{4}$ ou $\frac{1}{8}$ ème de pas, afin de gagner en précision et de baisser les vibrations qui pourrait être induite au télescope par les moteurs.
- Faire fonctionner le système GOTO, et permettre de choisir des coordonnées par la page web.

Future amélioration pour la carte température :

- ajouter le paramètre intégral à la régulation PID, et tester d'autre coefficient en condition réelle.
- Trouver à partir de quelle différence de température entre le tube et l'air, des courants d'air trop importants se créent, et détériorent l'image du télescope.

Amélioration générale :

- ajouter des fusibles.
- Mettre la Raspberry Pi en accès-point afin de pouvoir y accéder lorsqu'elle n'est pas connectée à un réseau wifi.
- Ajouter la dernière image que le télescope a prise en photo, et la dernière image de la camera de guidage sur la page web afin de voir ou viser le télescope et si le suivi est bon depuis la page web, sans avoir besoin de passer par VNC pour accéder à PHD2.
- Ajouter un checksum pour éviter les erreurs de communication.
- Ajouter un petit ventilateur pour refroidir la Raspberry pi.

5.2. Aspects gestions sur le projet

Au début du projet, j'arrivais à tout écrire dans le cahier de laboratoire, et je suivais le planning, mais au bout de quelque semaine, j'ai eu de plus en plus de difficulté à remplir le cahier de laboratoire. Le planning a globalement été respecté, mais il m'arrivait d'oublier de le remplir pendant quelque jour. Donc je le remplissais que j'avais le temps.

5.3. Aspects personnels

Ce projet m'a permis de mieux comprendre le fonctionnement des mouvements du ciel, et de les étudier, mais j'ai eu beaucoup de difficulté à trouver et à calculer la vitesse des moteurs qui permettent un suivi du ciel. Ça faisait aussi longtemps que je n'avais plus travaillé avec de moteur pas à pas. Comme le mouvement de ciel est très lent, j'ai été obligé d'utiliser une nouvelle unité d'angle (la seconde d'arc).

Durant tout le projet je n'ai eu que quelque nuit pour tester le suivi du ciel, et je regrette de ne pas avoir eu plus de nuits pour tester, en plus lorsque je programmais la carte moteur, je n'étais jamais vraiment sûr d'avoir la bonne vitesse, et j'étais obligé d'attendre la nuit, afin de tester.

Si c'était à refaire, j'essaierais de noter plus de raisonnement dans le cahier de laboratoire, et je prendrais quelques minutes chaque jour afin de remplir le planning.

J'ai hâte que ce soit l'hiver, afin de tester la régulation de température en condition réelle, et de tester plus amplement mon projet.

De manière générale, je suis très content du travail que j'ai réalisé, et du résultat obtenu, j'ai vraiment hâte de l'utiliser, et de voir si j'arrive à améliorer mes photos avec mon projet.

6. Remerciements

Je remercie Christian Humpert de s'être intéressé à mon projet, et de m'avoir aidé à développer les applications sur la Raspberry Pi.

Je remercie Antonin Kenzi de m'avoir partagé sa page web lors du deuxième travail de semestre, ce qui m'a permis de l'ajouter à mon projet de diplôme. Et de m'avoir aidé à résoudre mon problème d'alimentation pour la Raspberry Pi.

Je remercie David Carballo de sa participation pour l'écriture du chapitre 2.1 lors de la 1ere année de technicien.

Je remercie Arthur Cayuso de m'avoir parler du LM2576 pour alimenter la Raspberry Pi.

Je remercie Denis Carbon de m'avoir guidé vers le IRL3705 pour utiliser contrôler les résistances de chauffe.

7. Outils de développement utilisés

Logiciel	Révision	Utilisation
KICAD	4	Réalisation PCB
KEIL uVision4 Compiler C51	V3.05x V9.05	Programmation microcontrôleur
VNC		
Python 2.7	2.7	Programmation Page web et GOTO
Geany		Page HTML

8. Bibliographique et sites internet

<https://www.astroeq.co.uk/tutorials.php>

http://www.astrosurf.com/quasar95/exposes/les_solutions_contre_la_buee.pdf

<https://www.indilib.org/download/all.html>

<https://indilib.org/download/raspberry-pi>

<https://www.cloudynights.com/topic/636559-building-a-diy-motorized-control-for-a-skywatcher-eq5-mount/>

http://www.motionking.com/Products/PM_Stepper_Motors/42PM_Stepper_Motor.htm

<https://www.webastro.net/forums/topic/10962-vitesse-sidérale/>

<https://www.astroeq.co.uk/forum/index.php/topic,135.0.html>

9. Annexes

9.1. Mode d'emploi et d'installation

Diplôme de technicien ES en électronique - 2019

Astrophotography Assistant

Mode d'emploi



Table des matières

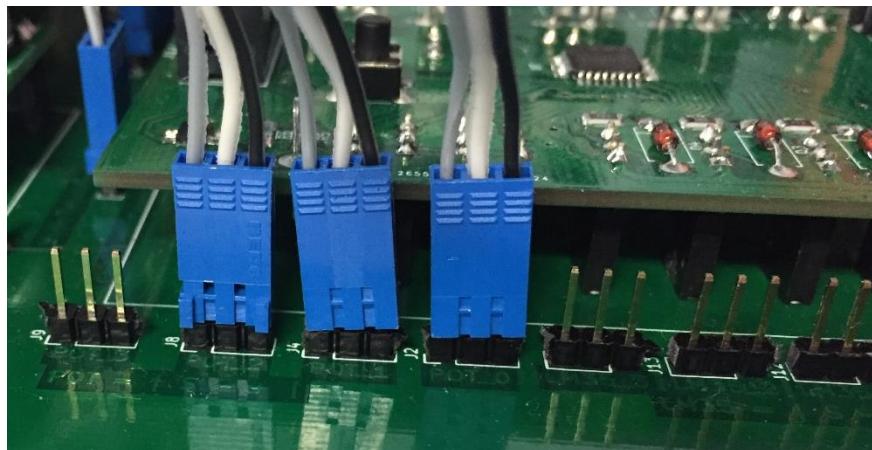
1. Interface Physique	66
2. Démarrage des applications	71
3. Page Web.....	72
4. PHD2	73

9.1.1. Interface Physique

La carte a besoin de potentiomètre et de roue encodeuse afin de fonctionner correctement et de connaître les paramètres de fonctionnement.

Potentiomètres :

Mais il faut les connecter en respectant un sens de connexion :



Les potentiomètres doivent avoir le câble gris en direction du haut de la carte.

GRIS = Signal

BLANC = GND

NOIR = VCC

Lorsque le sélecteur est en mode potentiomètre, les potentiomètres sont utilisés pour choisir la puissance des résistances de chauffe.



Sélecteur de modes de régulation :



Les différents modes de régulation sont les suivants :

GRIS = Mode potentiomètre

Blanc-Bleu = Mode PID

NOIR = Mode UART

Le câble gris doit être connecté sur le côté gauche de la carte.



Sélecteur de vitesse :



VERT = VCC

JAUNE = MSB (bit3)

ROUGE = bit2

BRUN = bit1

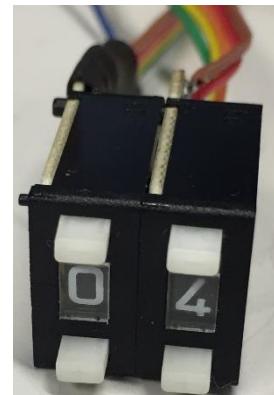
ORANGE = (LSB) bit0

Attention les deux connecteurs sont inversés. Un petit caractère « + » est noté sur le PCB pour indiquer le sens (VERT → +).

Les deux roues sont les mêmes donc il n'y a pas d'importance entre les deux.

La roue encodeuse connectée aux pins « vitesse » permet de choisir la vitesse de déplacement

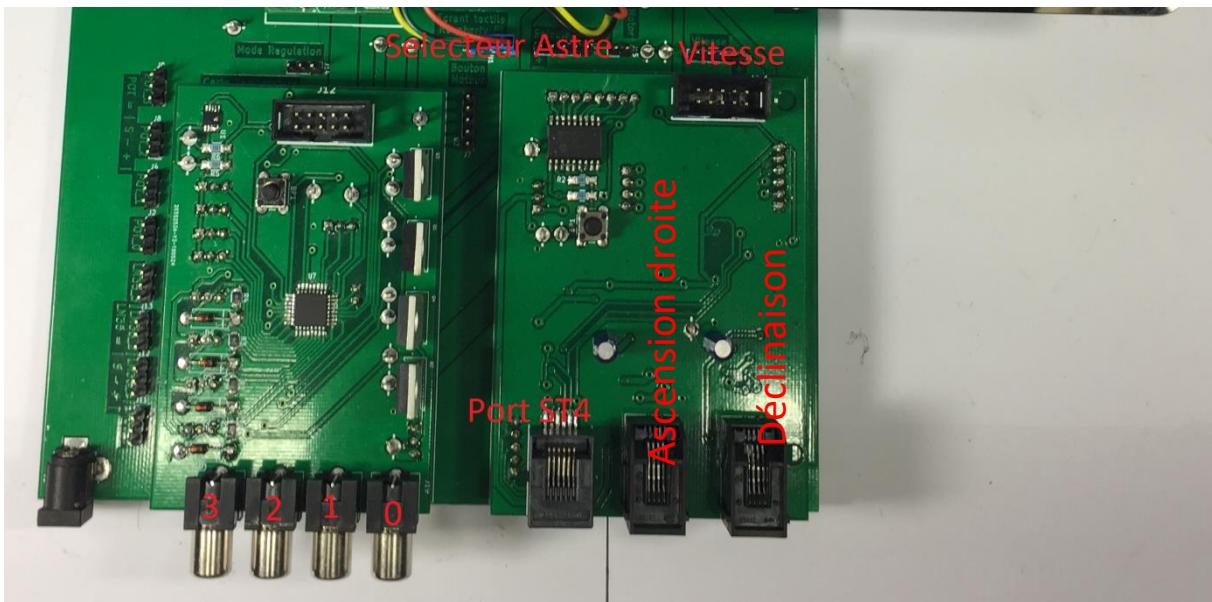
La roue encodeuse connectée à « Sélecteur Astre » permet de choisir la vitesse à appliquer sur le moteur d'ascension droite afin de suivre un astre.



Les différentes vitesses qui peuvent être choisies avec les roues encodeuses sont les suivantes :

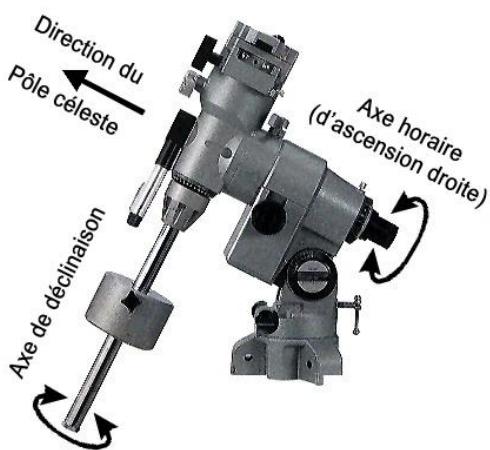
Sélecteur D'astre		Sélecteur de déplacement
Mode	Astre suivi	Vitesse de déplacement
0	Étoile	X0,5
1	Lune	X2
2	Soleil	X8
3	Saturne	X16
4	Jupiter	Vitesse maximale x48
5 ... 9	À définir (ne pas utiliser ces modes)	À définir (ne pas utiliser ces modes)

Câblage des prises RCA et des moteurs :



Connectez les moteurs, et le câble ST4 (le câble ST4 est plus gros que ceux des moteurs).

Petit rappel des axes de la monture pour le câblage :



Ascension droite :



Déclinaison :



Connecter la caméra :



Connecter le câble ST4 précédemment brancher a la carte sur la camera.

Puis brancher le port USB de la caméra sur le port USB de la Raspberry Pi à droite de l'écran.
Ainsi qu'un câble RJ45 pour que la Raspberry pi soit accessible sur le réseau.



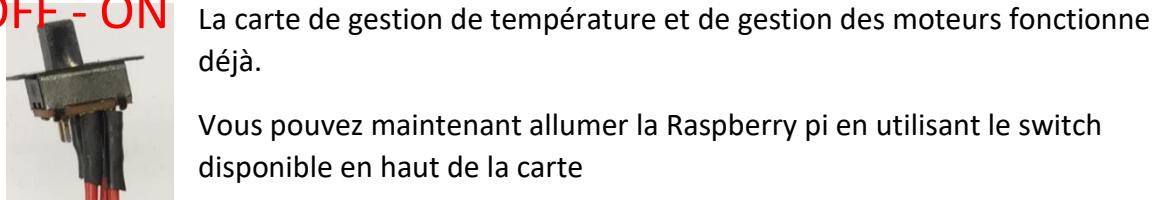
Connecteur Jack DC



Brancher et allumer la Raspberry Pi :

Brancher l'alimentation de la carte avec le connecteur Jack, ou avec les cosses disponible à gauche du PCB, attention à la polarité.

OFF - ON

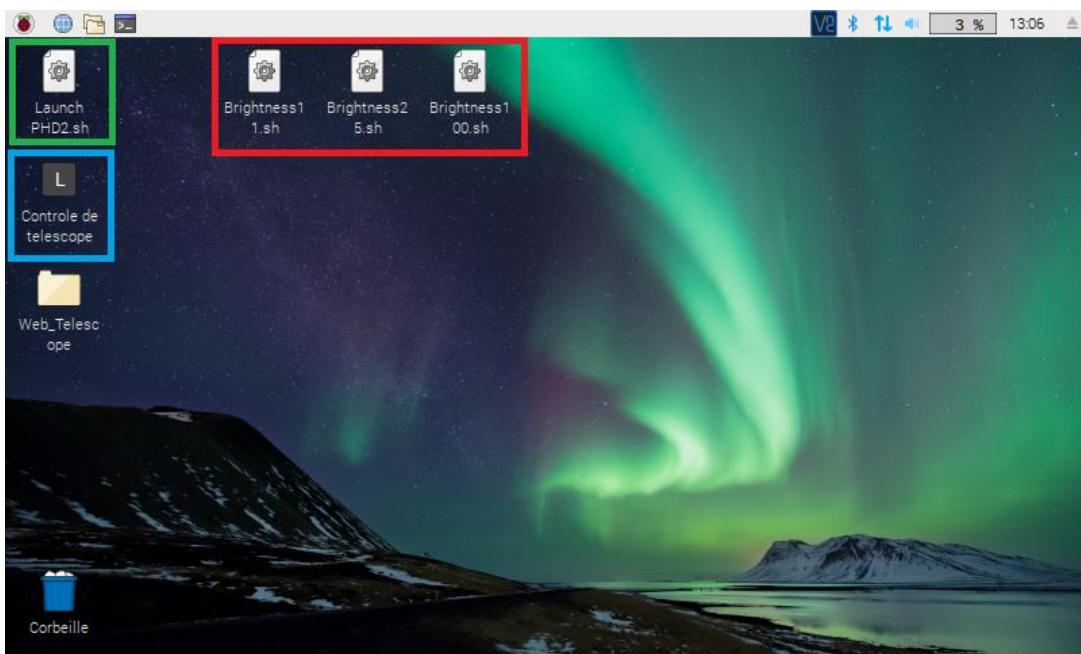


La carte de gestion de température et de gestion des moteurs fonctionne déjà.

Vous pouvez maintenant allumer la Raspberry pi en utilisant le switch disponible en haut de la carte

Avant d'éteindre la Raspberry Pi faites menu puis éteindre et attendez 30 secondes pour éteindre complètement la Raspberry pi.

9.1.2. Démarrage des applications

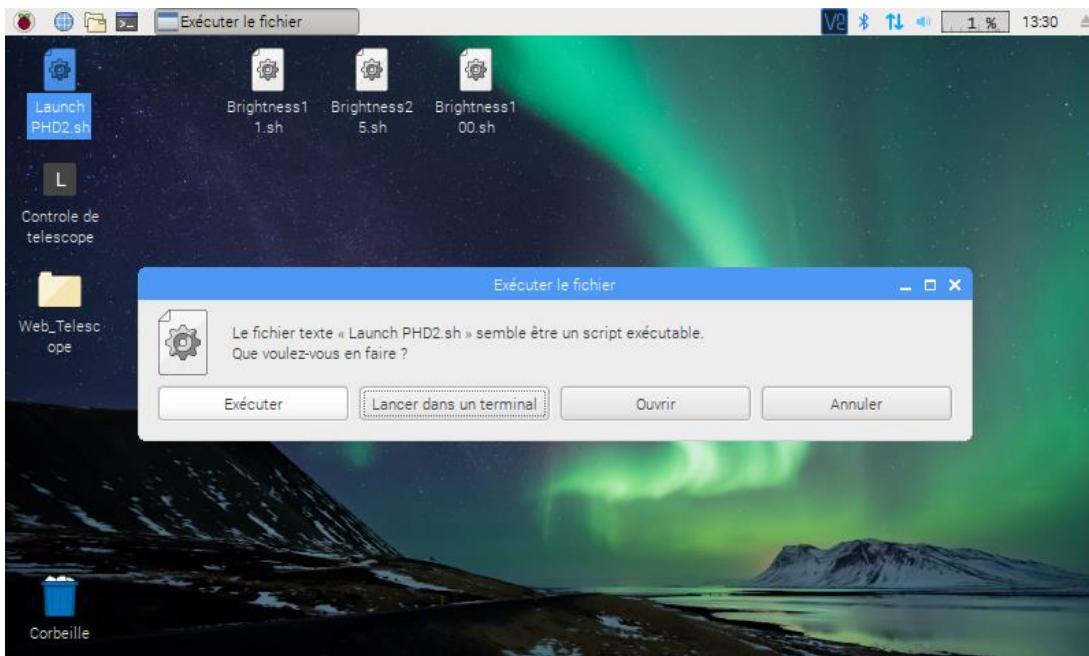


Lorsque la Raspberry pi démarre, le serveur web est lancé automatiquement en tâche de fond.

Pour ouvrir l'interface web sur la Raspberry pi pour contrôler le télescope, il faut cliquer sur l'icône qui s'appelle « Contrôle de télescope » encadré en bleu.

Pour ouvrir PHD2, afin de faire une régulation de suivi d'étoile, il faut cliquer sur l'icône qui s'appelle « Launch PHD2.sh » qui est encadré en vert.

Pour changer la luminosité de l'écran, il faut utiliser les exécutables mis à disposition dans le cadre rouge.

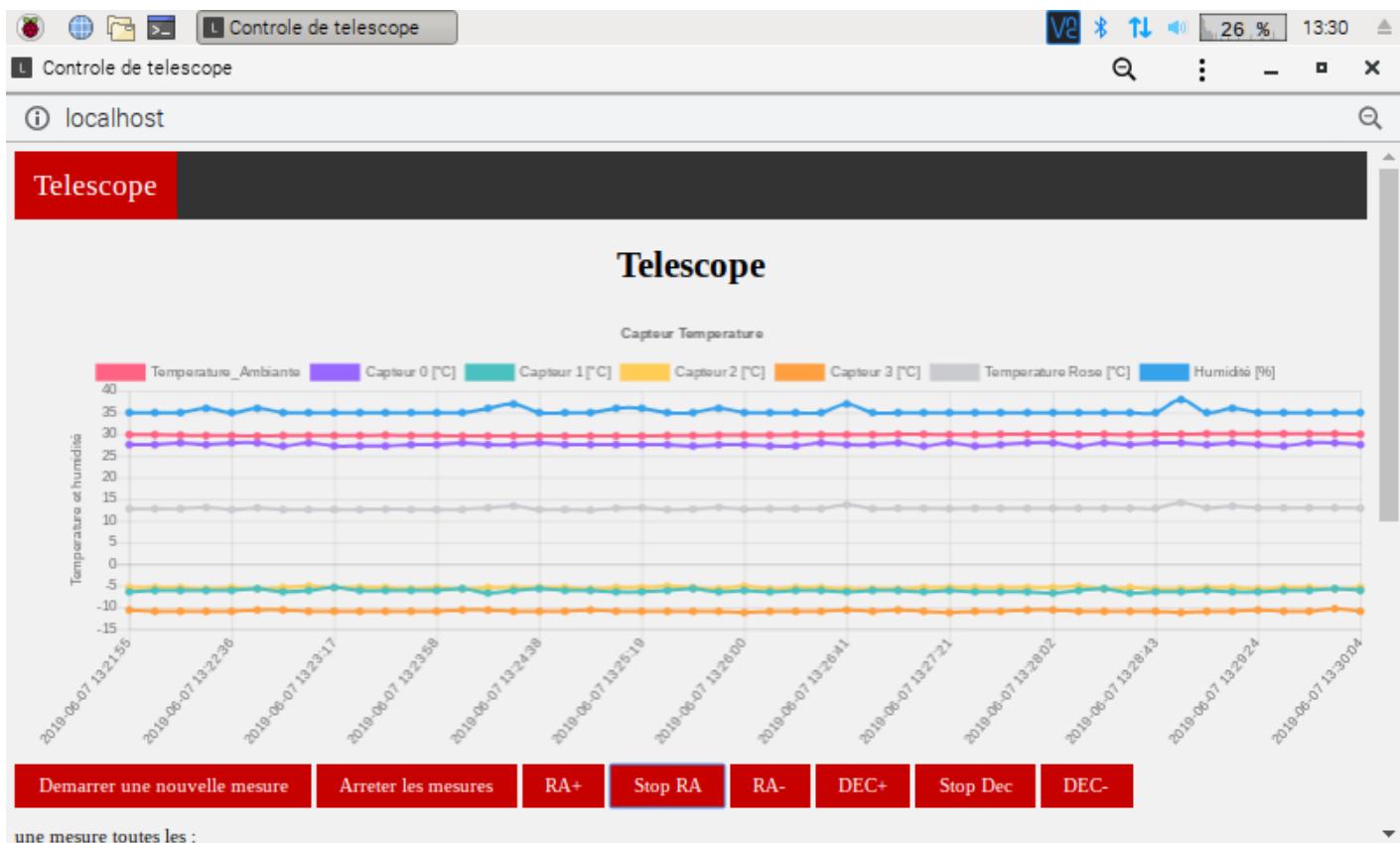


Le lancement de PHD2 et le changement de luminosité demandera une confirmation, cliquer sur Exécuter

9.1.3. Page Web

Voici la page web obtenue en utilisant l'exécutable sur le bureau :

La page web est aussi accessible en utilisant un navigateur web et en entrant l'adresse IP de la Raspberry pi.



La fréquence des mesures peut être définie sur la page à cet endroit :

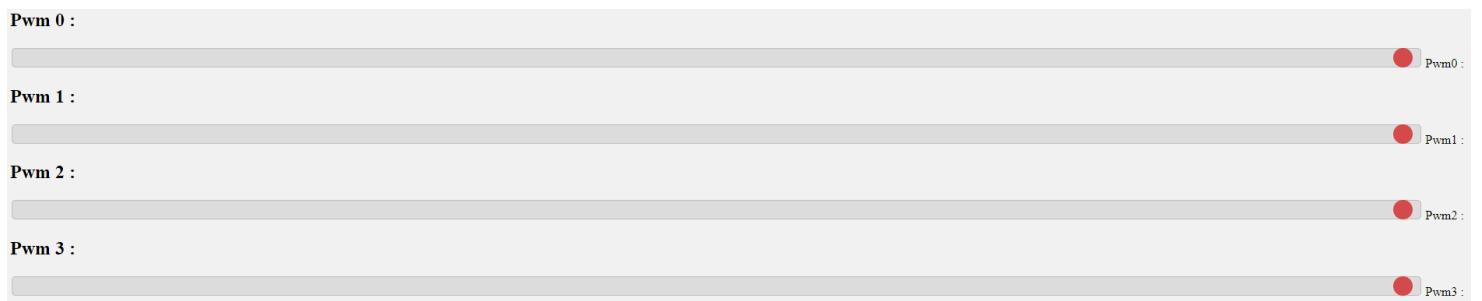
une mesure toutes les :

10

Liste des boutons disponibles sur la page :

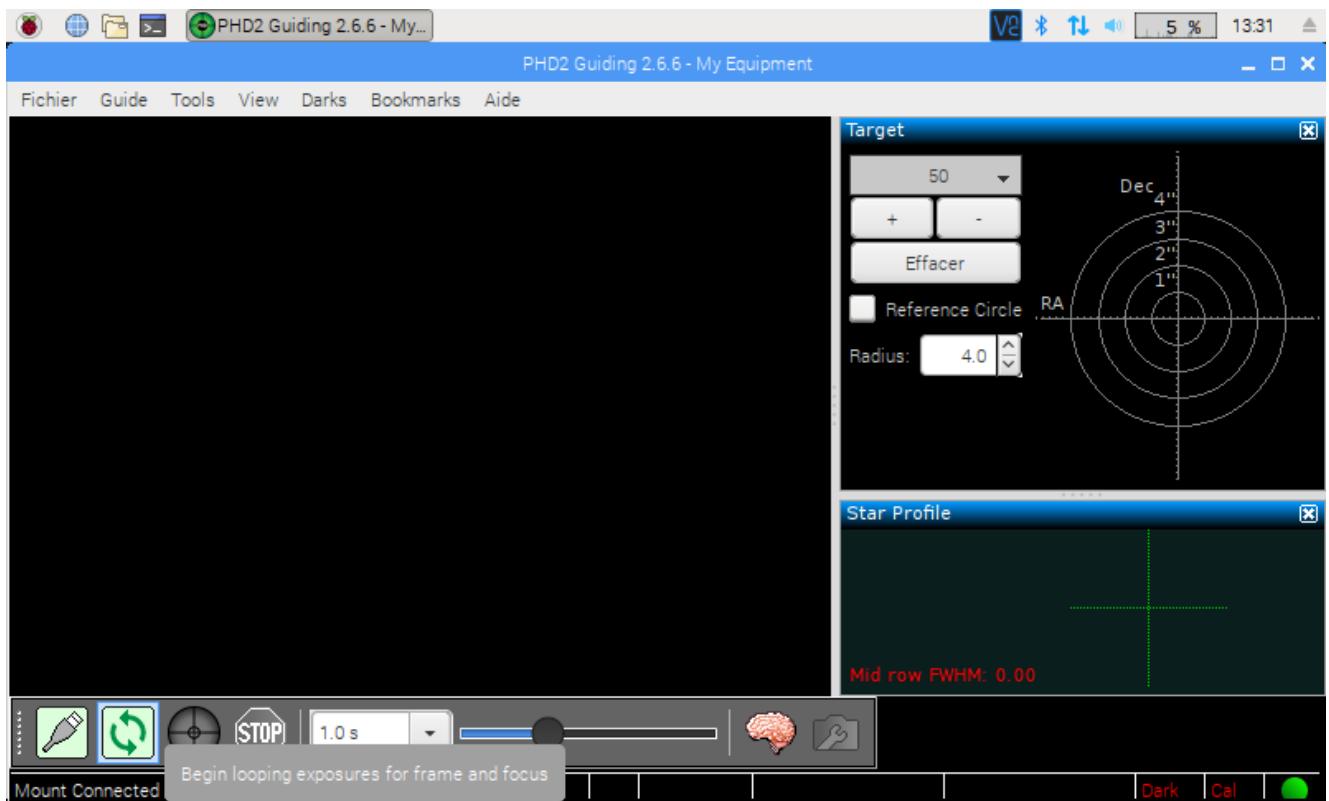
Bouton	Description
Démarrer une nouvelle mesure	Permet de recommencer des mesures dans un nouveau fichier (Date Heure.csv) Attention la page doit être rechargé.
Arrêter les mesures	Permet d'arrêter les mesures qui sont effectuées sur la carte
RA+	Déplace le télescope sur l'axe d'ascension droite dans le sens du ciel
Stop RA	Arrête un déplacement sur l'axe d'ascension droite
RA-	Déplace le télescope sur l'axe d'ascension droite dans le sens inverse du ciel
DEC+	Déplace le télescope sur l'axe de déclinaison
Stop DEC	Arrête un déplacement sur l'axe de déclinaison
DEC-	Déplace le télescope sur l'axe de déclinaison

Les curseurs disponibles sur la page permettent de choisir la puissance dissiper par les résistances de chauffe, à condition que le mode de régulation soit sur UART.



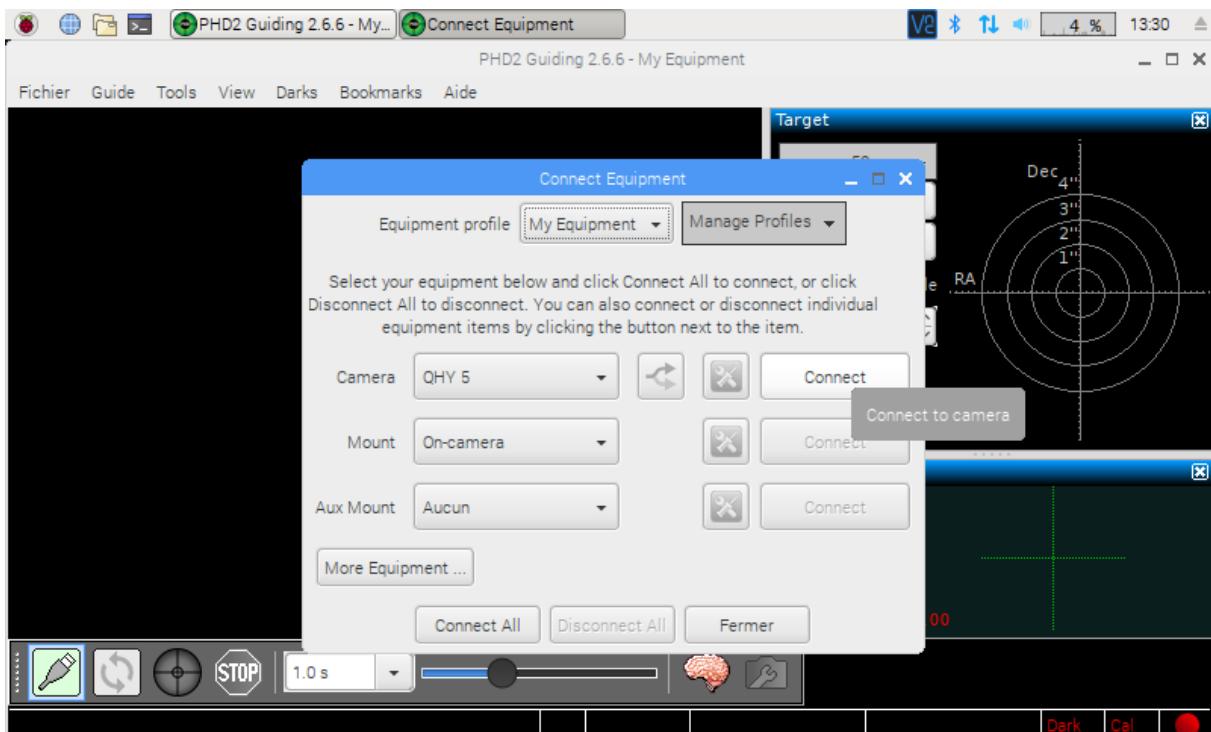
9.1.4. PHD2

L'interface de PHD2 ressembla à cela :



Pour connecter la caméra, il faut cliquer sur l'icône en bas à gauche :
Cette fenêtre devrait s'ouvrir :



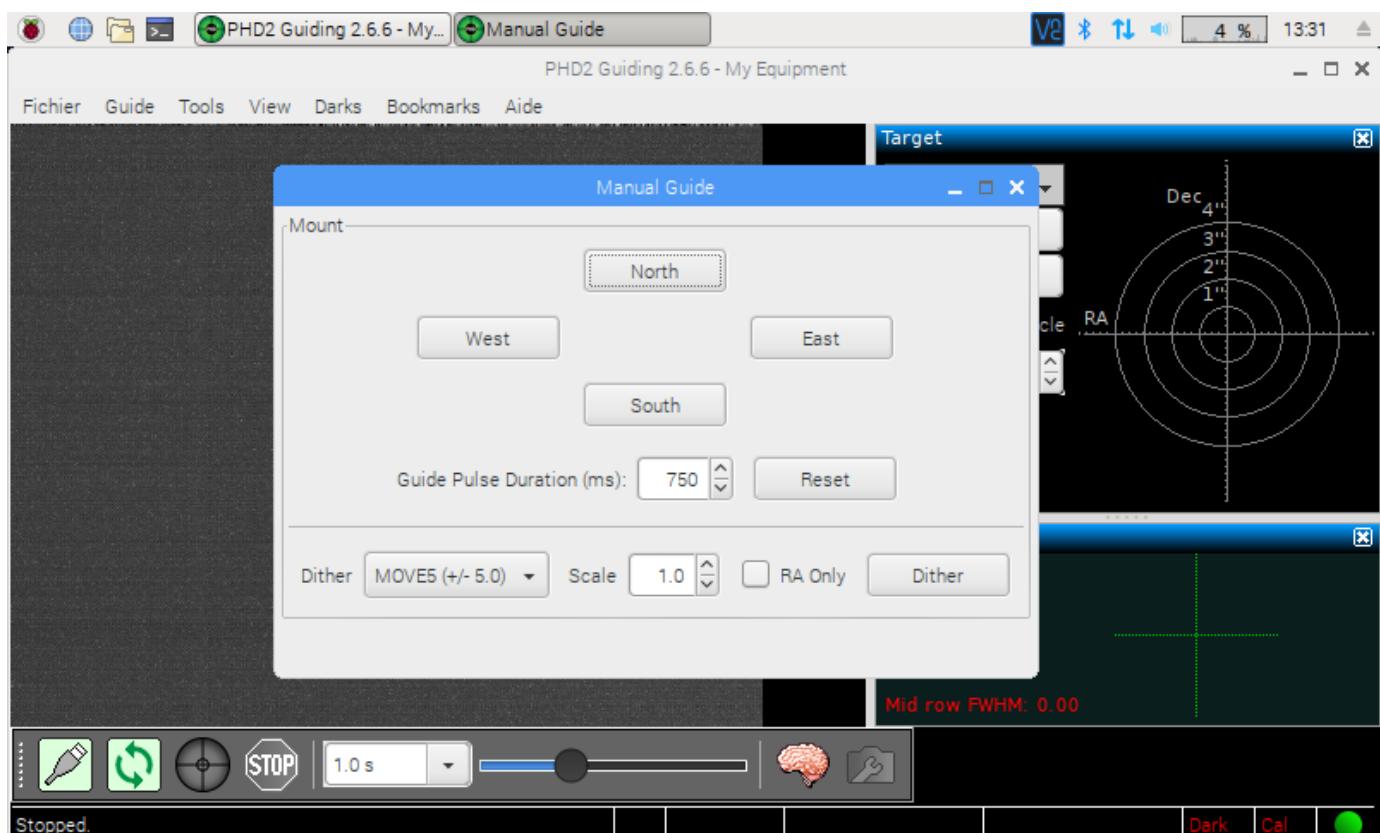


Cliquer sur connecter pour la camera et pour mount.

Pour commencer à prendre des photos avec la caméra de guidage, appuyer sur le bouton :



Il est possible de contrôler le télescope manuellement en allant dans Outils, manuel Guide :



Ces deux paramètres permettent de choisir le temps d'exposition et le contraste de l'image :

Dans ce cas, le temps de pose est d'une seconde :



9.2. Liste des source et pourcentage d'apport

<https://www.astroeq.co.uk/tutorials.php> → 10% → Moteur

<https://www.astroeq.co.uk/forum/index.php/topic,135.0.html> → 5% → Moteur

http://www.astrosurf.com/quasar95/exposes/les_solutions_contre_la_buee.pdf → 5% →
carte de température

<https://www.indilib.org/download/all.html> → 60% installation de PHD2

<https://indilib.org/download/raspberry-pi>

<https://www.cloudynights.com/topic/636559-building-a-diy-motorized-control-for-a-skywatcher-eq5-mount/> → 1 % → carte moteur

http://www.motionking.com/Products/PM_Stepper_Motors/42PM_Stepper_Motor.htm →
5% → carte moteur

<https://www.webastro.net/forums/topic/10962-vitesse-sidérale/> → 5 % → carte moteur

<https://www.wolframalpha.com/> → 10% → carte moteur

9.3. Planning

		semaine 1					semaine 2				
	Heures	lundi	mardi	mercredi	jeudi	ven	lundi	mardi	mercredi	jeudi	ven
Dietrich Tanguy											
mise à jour le 06.04.2019											
Travail de diplôme (288 heures , soit 36 jours)											
Panification, Analyse, Etude de composant	40										
Etude et mesure du protocole ST4	24										
Etude et mesure des moteur de suivie + Calcul de la vitesse de suivie.	8										
Ajout du port ST4 sur le microcontrôleur	5										
Montage et programmation du driver moteur	16										
Test du suivi des moteur(Si la meteo le permet) + mesure /Routage Kicad	16										
Cablage des capteur d'humidité et de température	16										
Test du fonctionnement du port ST4	8										
Ajout des mesure de température et d'humidité sur le micro + calcul du point de rosé	3										
Test de régulation de température	8										
Mise en place de la communication UART du F38C vers le raspberry Pi	16										
Developpement de la page WEB du Raspberry Pi	12										
Creation du schema finale sur kicad + developpement du PCB	24										
Montage de la carte et test de fonctionnement	24										
Rapport	40										
	92										

Dietrich Tanguy

mise à jour le 06.04.2019

Travail de diplôme (288 heures , soit 36 jours)

Heures	Estimées Planifiées Réalisées	semaine 3					semaine 4				
		lundi	mardi	mercredi	jeudi	ven	lundi	mardi	mercredi	jeudi	ven
22.04.19											03.05.19
23.04.19											02.05.19
24.04.19											01.05.19
25.04.19											30.04.19
26.04.19											
27.04.19											
28.04.19											
29.04.19											
30.04.19											
Panification, Analyse, Etude de compositant	40 24										
Etude et mesure du protocole ST4	8 5										
Etude et mesure des moteur de suivie + Calcul de la vitesse de survi.	16 6										
Montage et programmation du driver moteur	16 13										
Test du suivie des moteur(Si la meteo le permet) + mesure /Routage Kicad	8 0										
Ajout du port ST14 sur le microcontrôleur	34 16										
Test du fonctionnement du port ST14	8 3										
Cablage des capteur d'humidité et de température	8 24										
Ajout des mesure de température et d'humidité sur le micro + calcul du point de rosé	16 16										
Test de regulation de température	8 16										
Mise en place de la communication UART du F38C vers le raspberry Pi	16 12										
Développement de la page WEB du Raspberry Pi	24 24										
Creation du schema finale sur kicad + développement du PCB	16 32										
Montage de la carte et test de fonctionnement	40 44										
Rapport	40 92										

Dietrich Tangy

mise à jour le 06.04.2019

Travail de diplôme (288 heures , soit 36 jours)

Heures	Estimées	Réalisées	semaine 5					semaine 6				
			lundi	mardi	mercredi	jeudi	ven	lundi	mardi	mercredi	jeudi	ven
06.05.19	40	40										
07.05.19	24	24										
08.05.19	8	8										
09.05.19	5	5										
10.05.19	16	16										
11.05.19	6	6										
12.05.19	16	16										
13.05.19	13	13										
14.05.19	8	8										
15.05.19	0	0										
16.05.19	34	34										
17.05.19	8	8										
18.05.19	3	3										
19.05.19	24	24										
20.05.19	16	16										
21.05.19	12	12										
22.05.19	24	24										
23.05.19	24	24										
24.05.19	44	44										
25.05.19	40	40										
	92	92										

Dietrich Tanguy

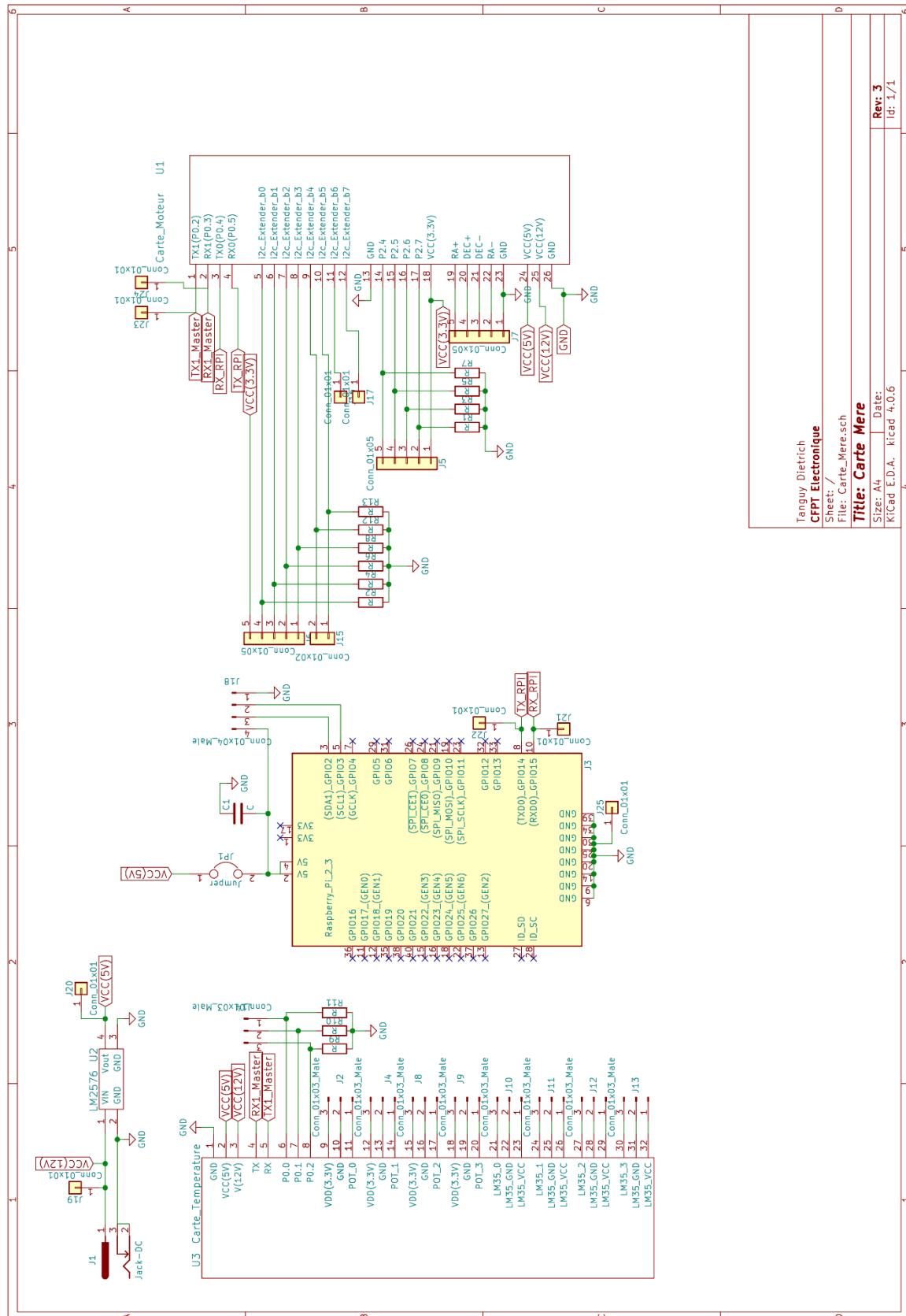
mise à jour le 06.04.2019

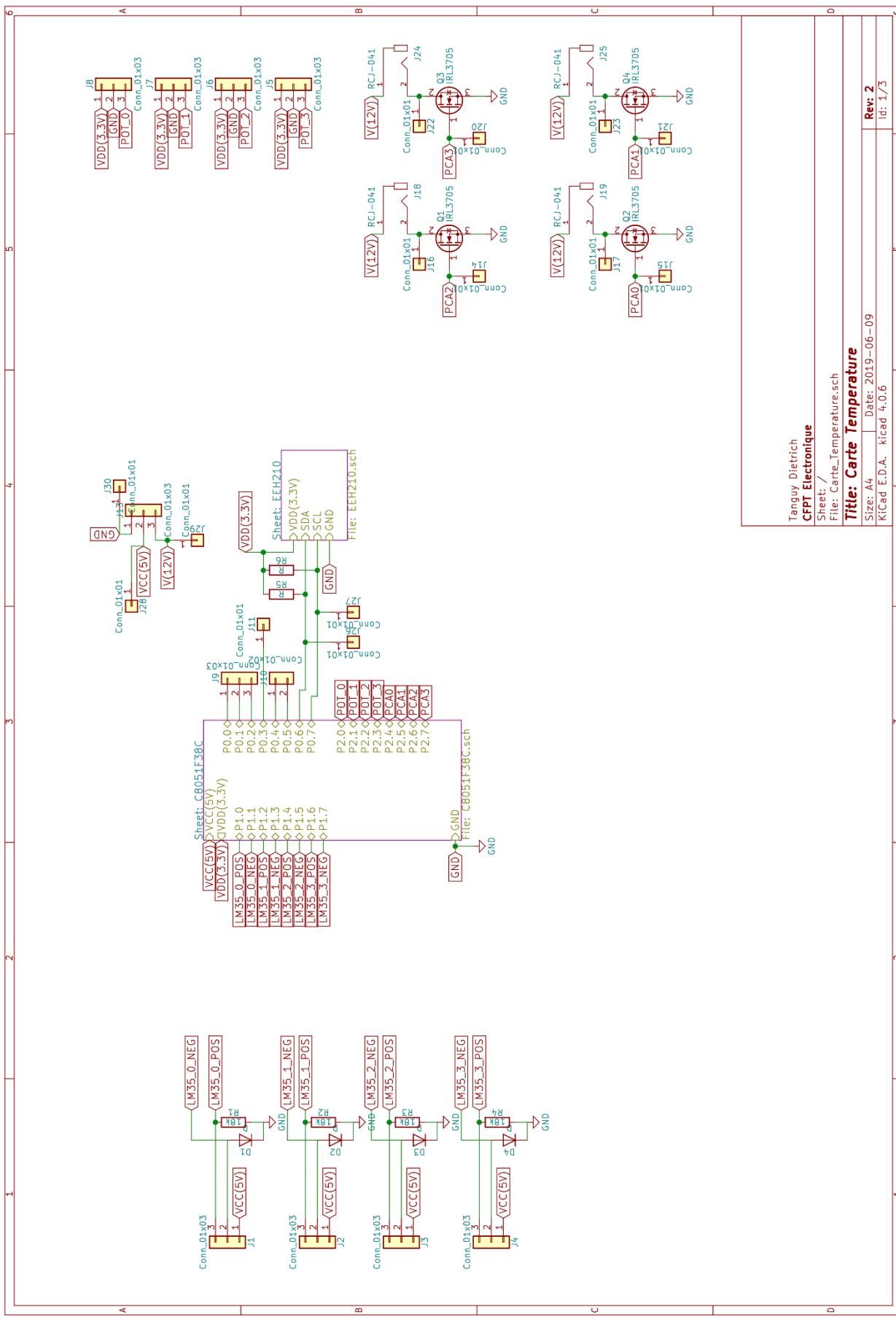
Travail de diplôme (288 heures , soit 36 jours)

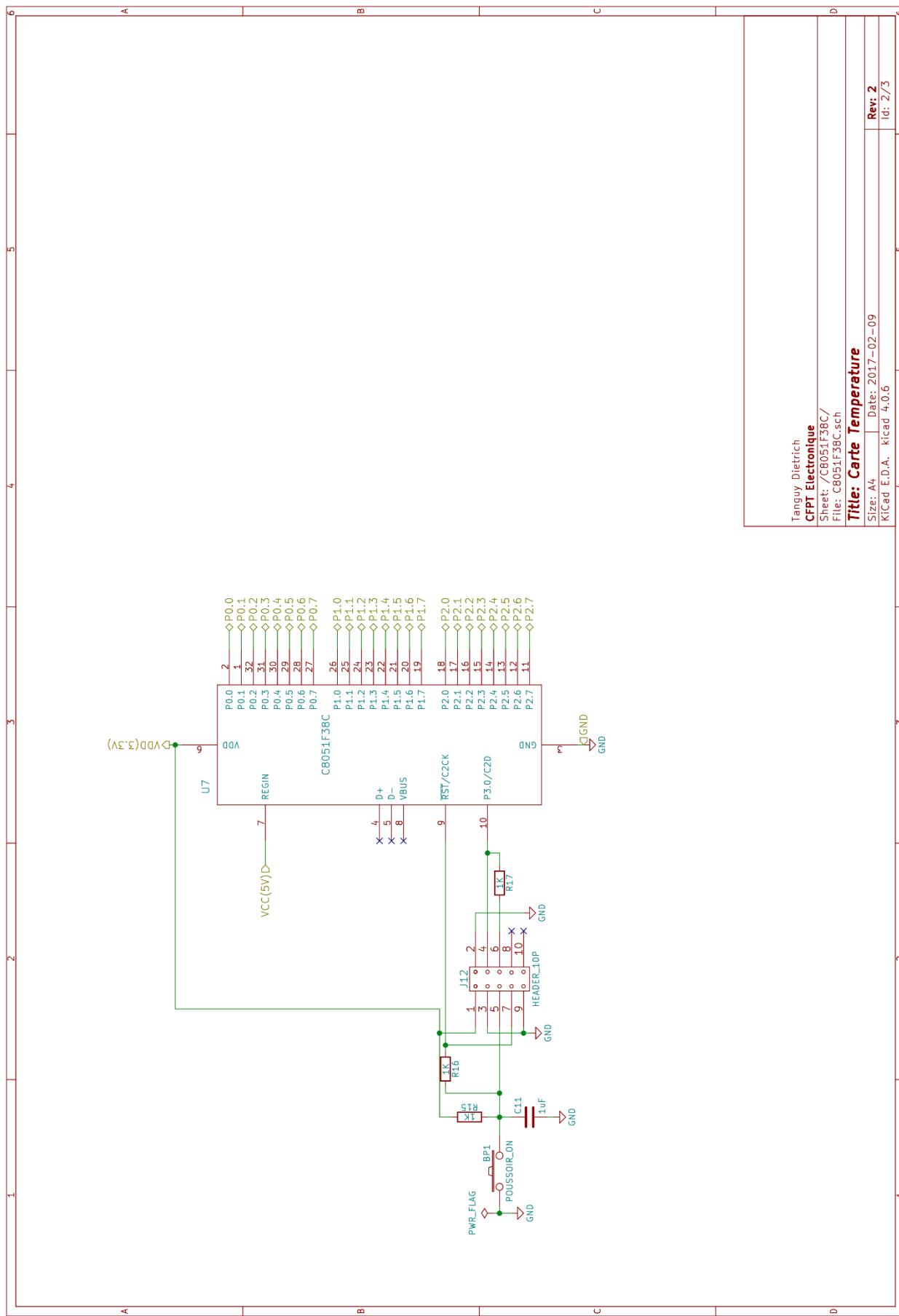
Heures	Estimées	Réalisées	semaine 7					semaine 8					
			mardi	mercredi	jeudi	ven	lundi	mardi	mercredi	jeudi	ven	lundi	
Panification, Analyse, Etude de composant	40	24											
Etude et mesure du protocole ST4	8												
Etude et mesure des moteur de suivie + Calcul de la vitesse de suivi.	5												
Montage et programmation du driver moteur	16												
Test du suivie des moteur(Si la meteo le permet) + mesure /Routage Kicad	6												
Ajout du port ST4 sur le microcontrôleur	16												
Test du fonctionnement du port ST4	8												
Cablage des capteur d'humidité et de température	3												
Ajout des mesure de température et d'humidité sur le micro + calcul du point de rosé	8												
Test de régulation de température	16												
Mise en place de la communication UART du F38C vers le raspberry Pi	16												
Développement de la page WEB du Raspberry Pi	12												
Création du schéma finale sur kicad + développement du PCB	24												
Montage de la carte et test de fonctionnement	40												
Rapport	44												
	40												
	92												

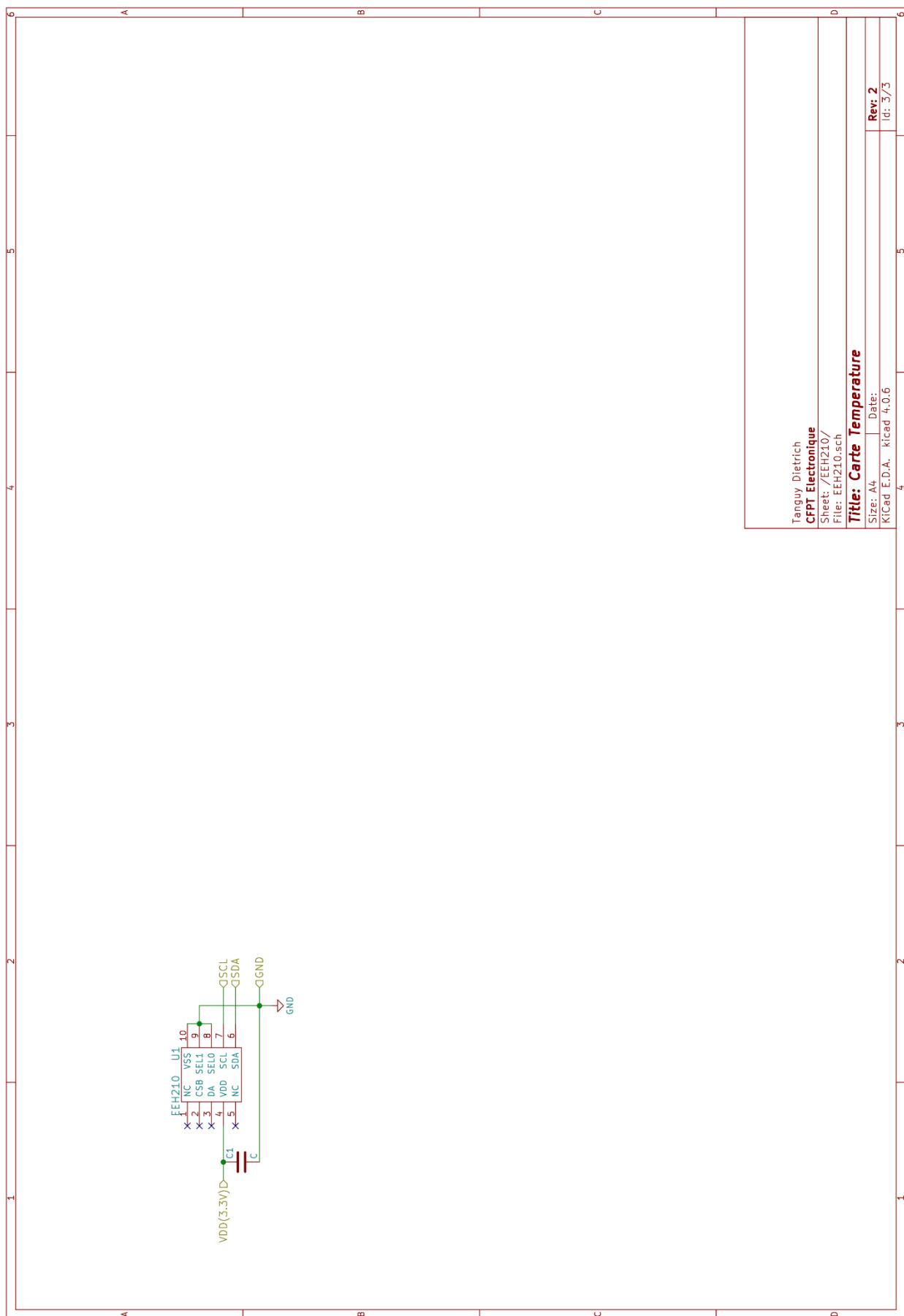
Dietrich Tanguy		semaine 9							semaine 10							semaine 11						
	mis à jour le 06.04.2019	mercredi	jeudi	ven	lundi	mardi	mercredi	jeudi	ven	lundi	mardi	mercredi	jeudi	ven	lundi	mardi	mercredi	jeudi	ven	lundi	mardi	mercredi
Estimées	Plannifiées	05.06.19	06.06.19	07.06.19	10.06.19	11.06.19	12.06.19	13.06.19	14.06.19	17.06.19	18.06.19	19.06.19										
Planifiées	Estimées	40	24	8	5	16	6	16	13	8	0	34	16	8	3	8	24	16	8	16	40	92
Panification, Analyse, Etude de composant																						
Etude et mesure du protocole ST4																						
Etude et mesure des moteur de suivie + Calcul de la vitesse de suivi.																						
Montage et programmation du driver moteur																						
Test du suivie des moteur(Si la meteo le permet) + mesure /Routage Kicad																						
Ajout du port ST4 sur le microcontrôleur																						
Test du fonctionnement du port ST4																						
Cablage des capteur d'humidité et de température																						
Ajout des mesure de température et d'humidité sur le micro + calcul du point de rosé																						
Test de regulation de température																						
Mise en place de la communication UART du F38C vers le raspberry Pi																						
Développement de la page WEB du Raspberry Pi																						
Creation du schema finale sur kicad + développement du PCB																						
Montage de la carte et test de fonctionnement																						
Rapport																						

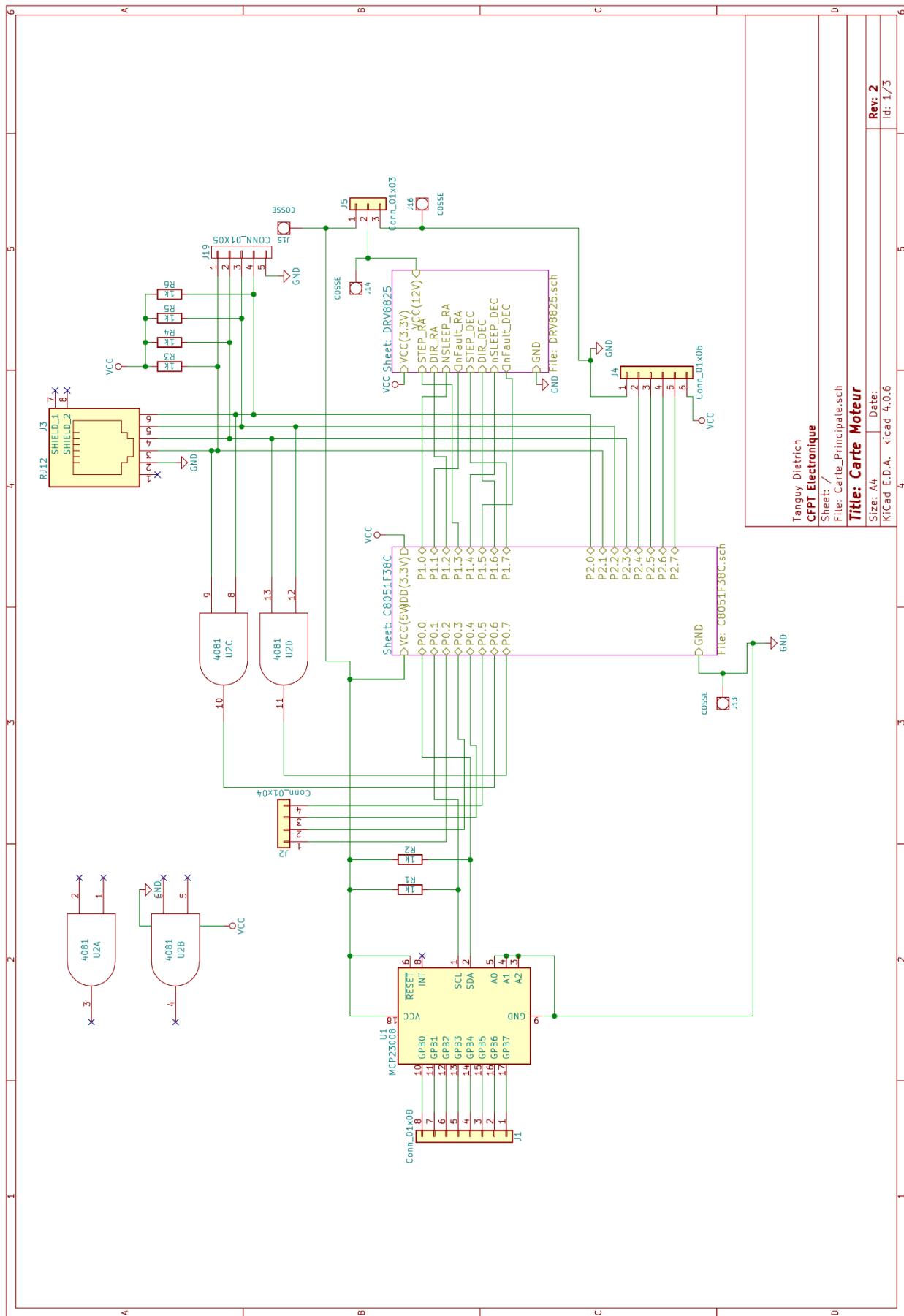
9.4. Schéma

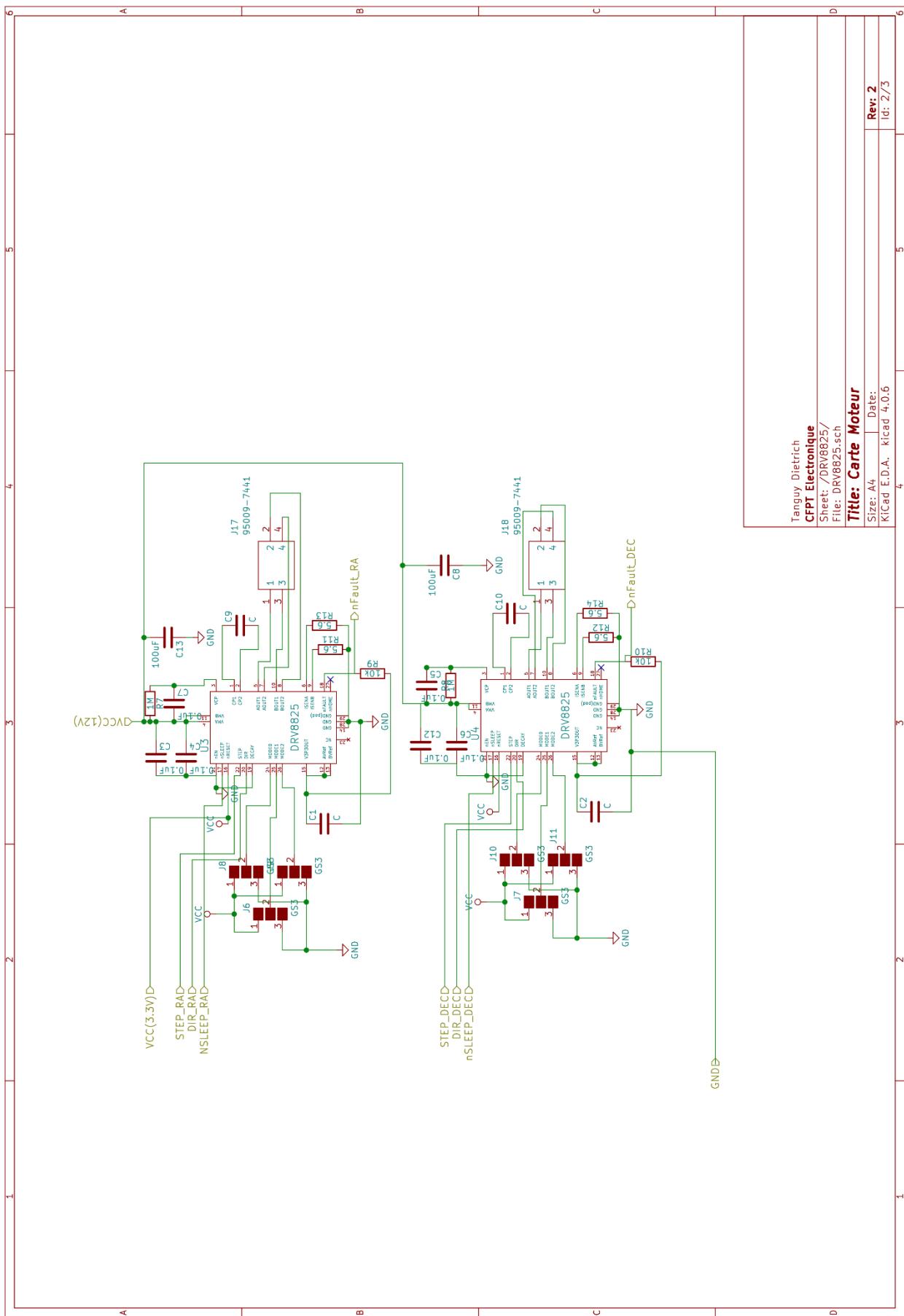


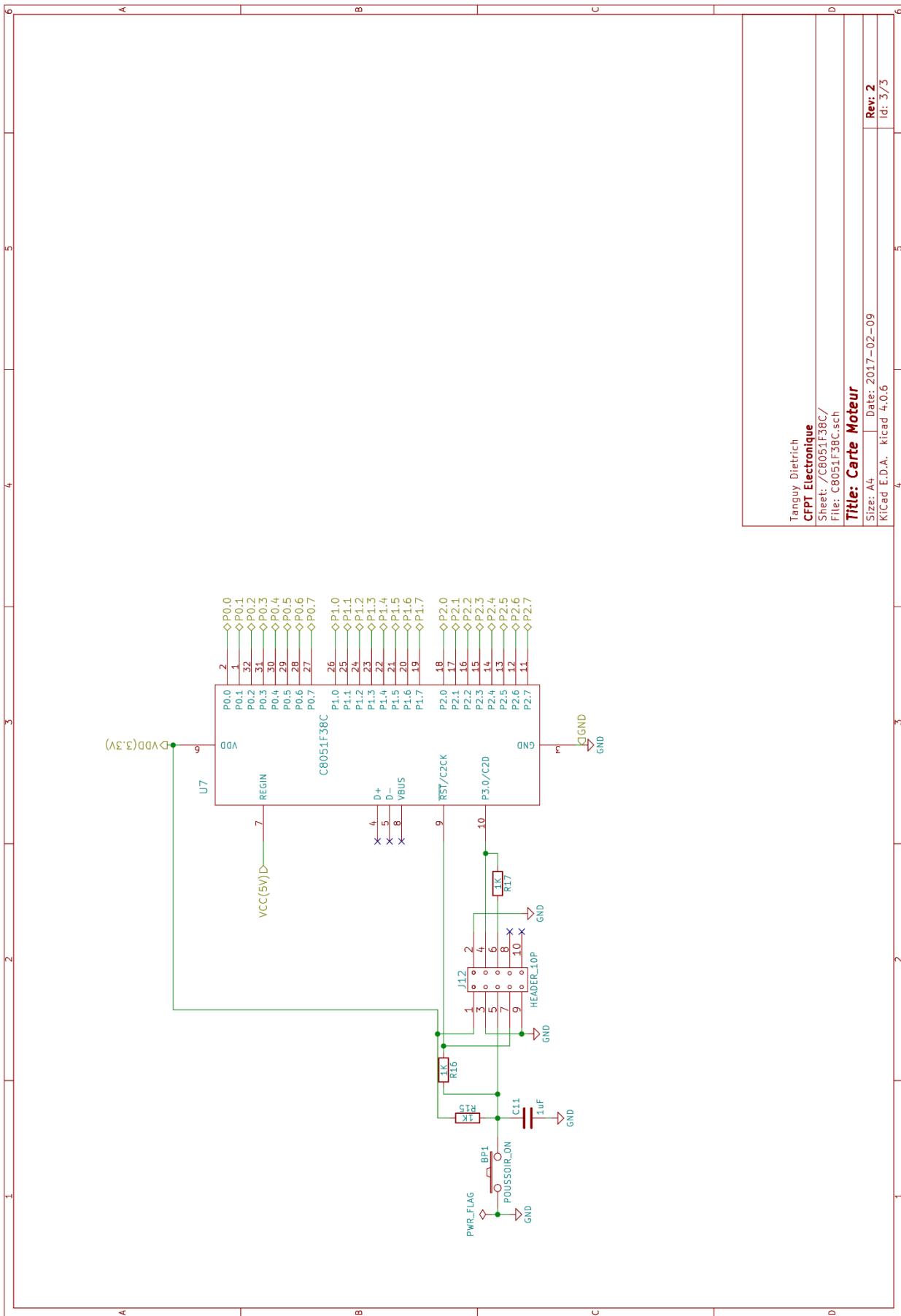












9.5. Dossier de fabrication des circuits imprimés

Projet / produit :	Diplôme 2019 - Télescope
Description :	
Quantités :	<ul style="list-style-type: none"><u>1</u> pièce<u>_</u> pièces / lot de fabrication<u>_</u> pièces / an

Circuit :

Nom :	Carte Mere		
Numéro :	ND (non disponible)		
Révision :	3		
Date de création :	28.05.2019	Auteur :	Tanguy Dietrich

Historique des modifications :

Rév.	Date	Auteur	Détails des modifications
01			
02			
03			
04			
05			
06			
07			
08			
09			
10			

Spécification du PCB :

Nature du matériel isolant :	<input type="checkbox"/> Bakélite	<input type="checkbox"/> FR-2	<input type="checkbox"/>	_____	
	<input type="checkbox"/> Epoxy	<input checked="" type="checkbox"/> FR-4	<input type="checkbox"/>	_____	
	<input type="checkbox"/> Autre : _____				
Epaisseur du PCB :	<input type="checkbox"/> 0.5 mm	<input type="checkbox"/> 0.8 mm	<input type="checkbox"/> 1.0 mm	<input type="checkbox"/> 1.2 mm	
	<input checked="" type="checkbox"/> 1.6 mm	<input type="checkbox"/> 2.0 mm	<input type="checkbox"/> autre : _____ mm		
Nombre de couches :	<input type="checkbox"/> Simple face				
	<input checked="" type="checkbox"/> Double face				
	<input type="checkbox"/> Multicouches, nombre de couches = X				
Epaisseur des couches cuivre :					
Externe	<input checked="" type="checkbox"/> 18 µ	<input type="checkbox"/> 35 µ	<input type="checkbox"/> 75 µ	<input type="checkbox"/> 105 µ	
				<input type="checkbox"/> autre : _____	
Interne	<input type="checkbox"/> 18 µ	<input type="checkbox"/> 35 µ	<input type="checkbox"/> 75 µ	<input type="checkbox"/> 105 µ	
				<input type="checkbox"/> autre : _____	
Largeur minimum des pistes :	0.508	mm	/	20m	inch
Largeur minimum entre pistes :	0.254	mm	/	10m	inch

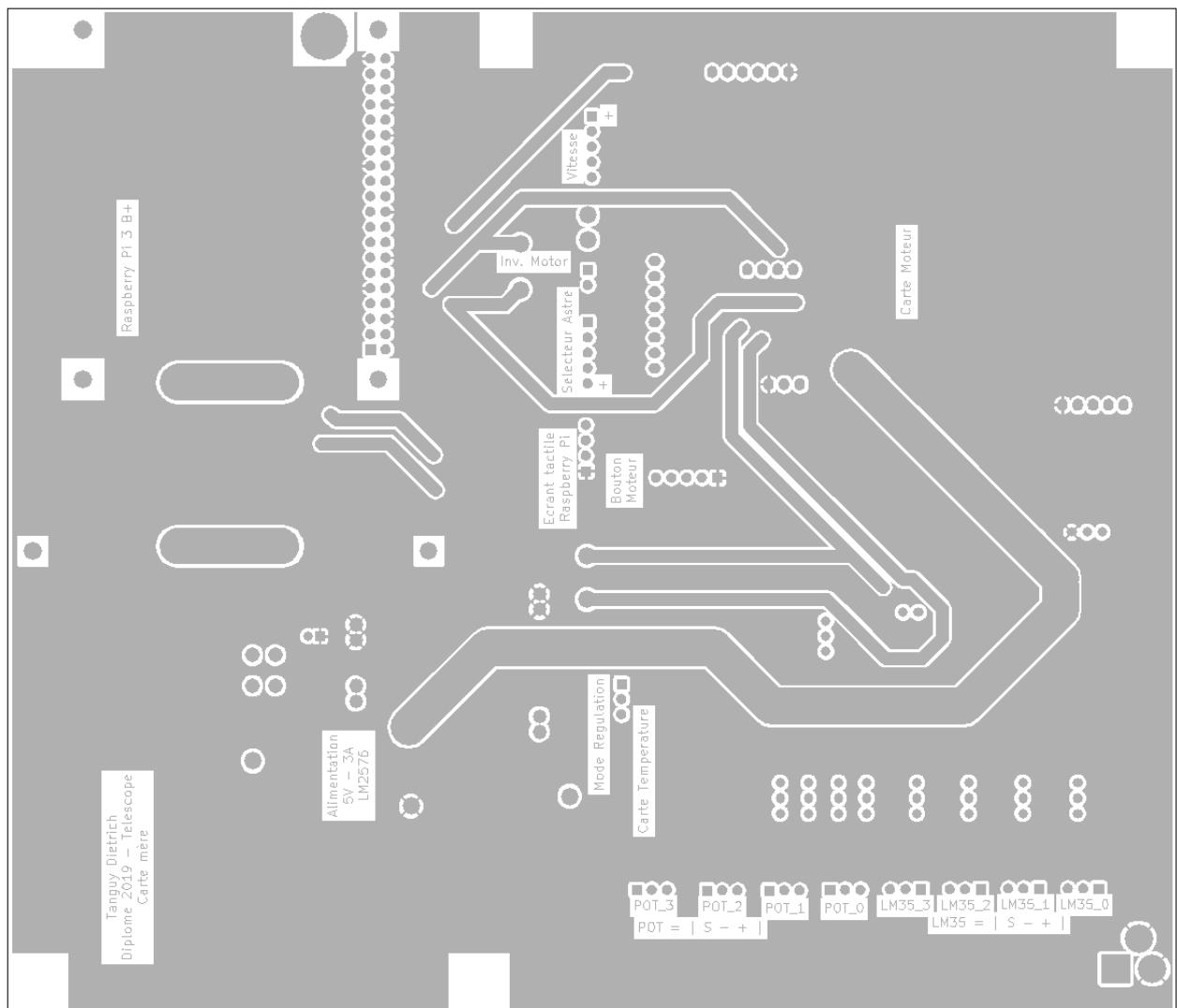
Liste des documents :

<input checked="" type="checkbox"/>	Liste du matériel	Page 3
<input checked="" type="checkbox"/>	Dimensions du PCB	Page 4
<input checked="" type="checkbox"/>	Plan de perçage	Page 5
<input checked="" type="checkbox"/>	Plan d'implantation dessous	Page 6
<input checked="" type="checkbox"/>	Plan d'implantation dessus	Page 6
<input checked="" type="checkbox"/>	Print : vue de dessous	Page 7
<input checked="" type="checkbox"/>	Print : vue de dessus	Page 7
<input checked="" type="checkbox"/>	Points tests	Page 8
<input type="checkbox"/>	Fichier Gerber	_____

Liste du matériel :

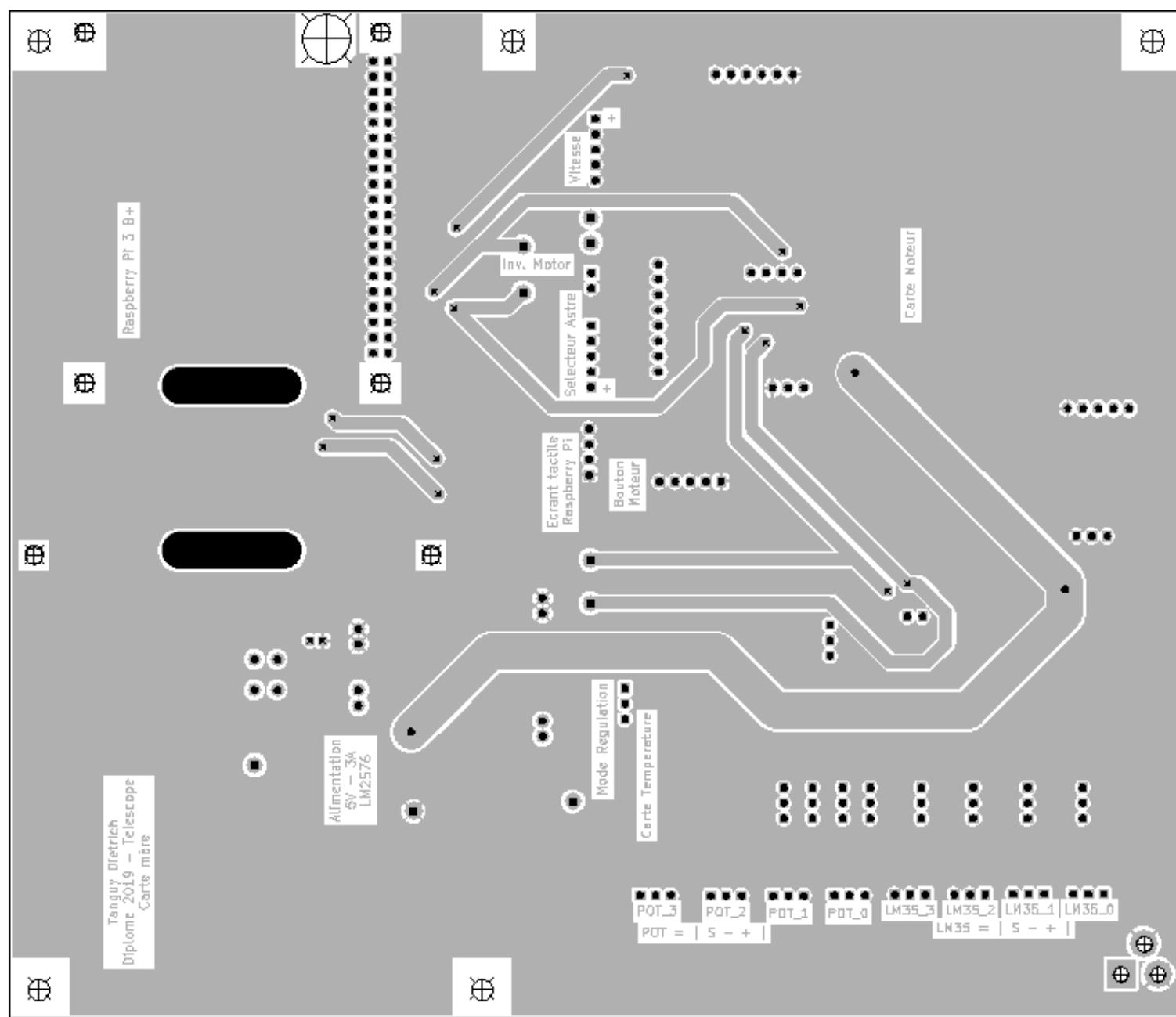
Références	Composant	N° article Distrelec	Quantité	Pu [CHF]	Ptot [CHF]
C1	100uF 16V radial 5mm	300-92-017	1	0.063	0.063
J1	Connecteur jack-DC	301-29-715	1	0.4335	0.4335
J2; J4; J8; J9; J10; J11; J12; J13; J14	Connecteur 3 pin Male	300-93-644	9	0.1317	1.1853
J3	Connecteur 2*20 Femelle (raspberry Pi)	300-93-679	1	2.05	2.05
J5; J6; J7	Connecteur 5 pins Male	300-93-646	3	0.2346	0.7038
J15	Connecteur 2 pins Male	300-93-652	1	0.176	0.176
J16; J17; J19; J20; J21; J22; J23; J24; J25	Cosse poignard		9		
J18	Connecteur 4 pins Male	300-93-645	1	0.1761	0.1671
JP1	Connecteur 3 Pin male modifier	300-93-644	2	0.1317	0.2634
R1; R2; R3; R4; R5; R6; R7; R8; R9; R10; R11; R12; R13	Résistance SMD 1206 10 [kOhm]	300-56-987	13	0.0285	0.3705
U1	Carte moteur		1	18.1567	18.1567
J3	Raspberry Pi 3 B+	301-09-158	1	39.6	39.6
J3	LCD 7 pouces pour Raspberry Pi	300-37-316	1	86.90	86.90
U2	Carte LM2576		1	4.6161	4.6161
U3	Carte Température		1	55.44	55.44
U3,U2	Connecteur 2 pins femelles	300-93-662	5	0.5385	2.6925
U3,U1	Connecteur 3 pins femelles	300-93-663	11	0.8973	9.8703
U1	Connecteur 4 pins femelles	300-93-664	1	0.8973	0.8973
U1	Connecteur 5 pins femelles	300-93-665	1	0.9972	0.9972
U1	Connecteur 6 pins femelles	300-93-666	1	0.9971	0.9971
U1	Connecteur 8 pins femelles	300-93-668	1	1.14	1.14
PCB carte Mere			1	2.98	2.98
Prix total =					229.6998 [CHF]

Dimensions de la carte :



194mm x 166.7mm

Plan de perçage :



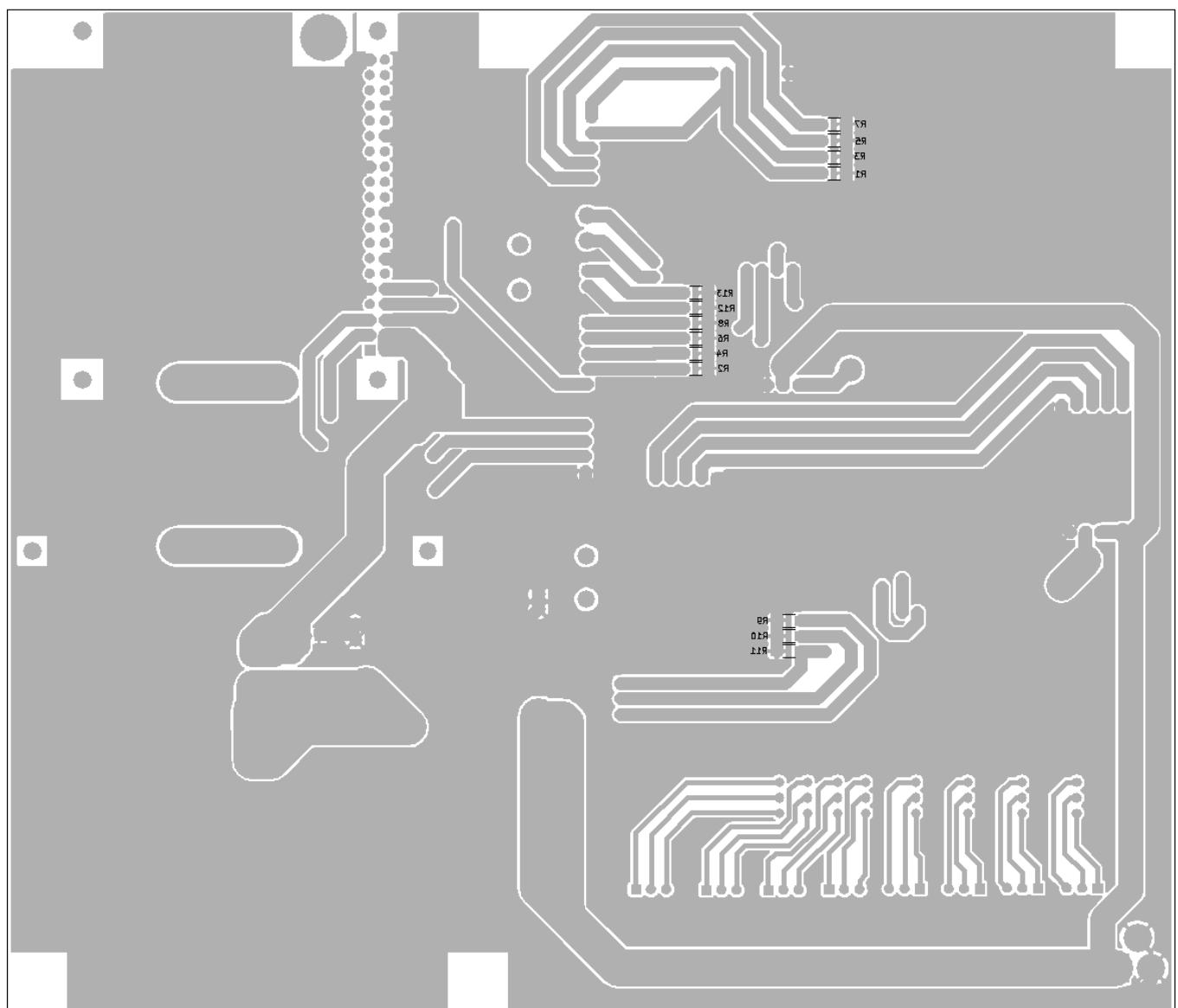
Drill Map:

- × 0.80mm / 0.031" (16 holes)
- ◊ 1.00mm / 0.039" (135 holes)
- + 1.02mm / 0.040" (26 holes)
- 1.30mm / 0.051" (9 holes)
- ◇ 2.54mm / 0.100" (1 holes + 2 slots)
- ⊗ 3.00mm / 0.118" (6 holes)
- 6.00mm / 0.236" (0 holes + 2 slots)
- 8.00mm / 0.315" (1 hole)

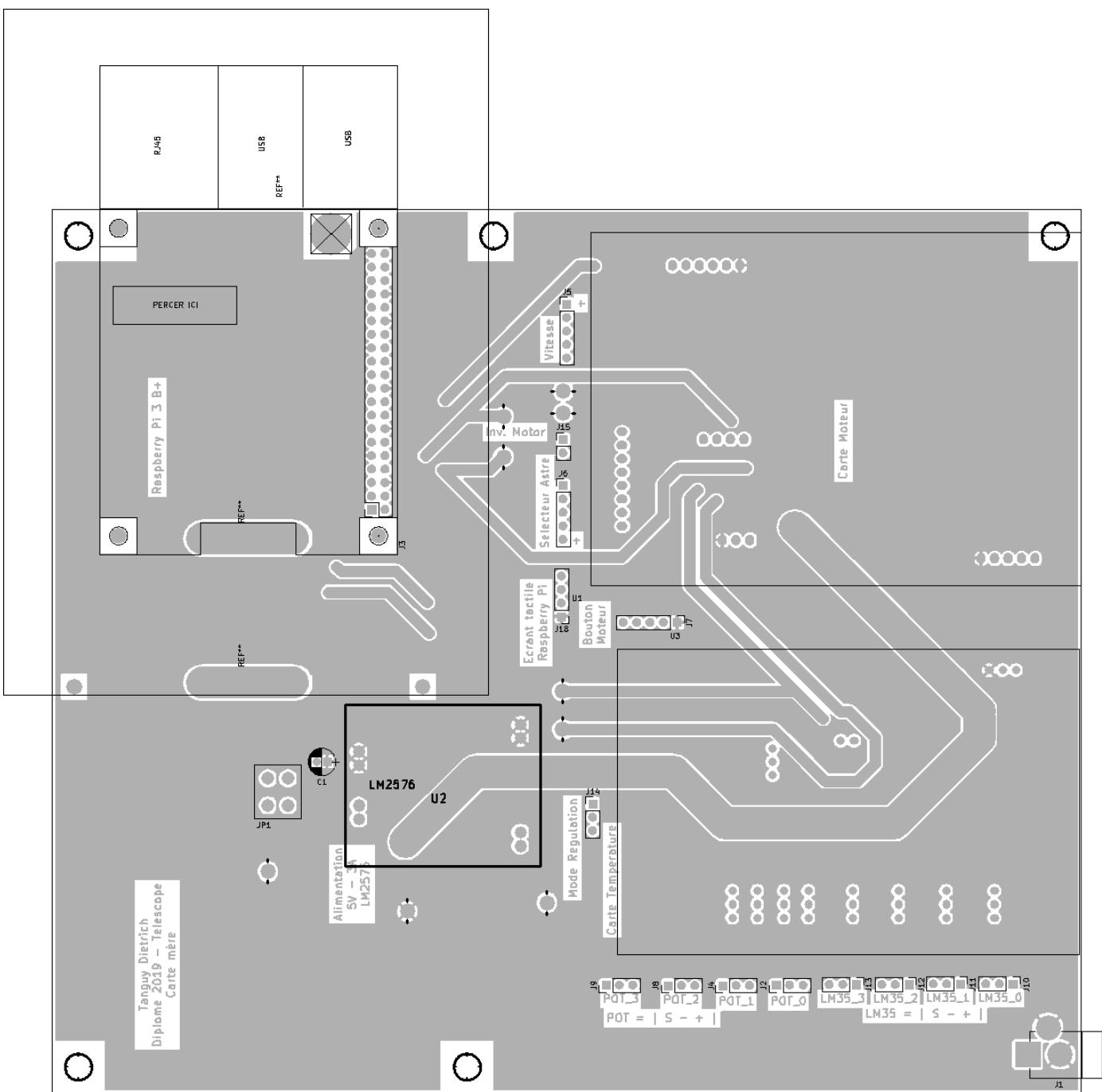
Drill Map:

- × 3.50mm / 0.138" (5 holes) (not plated)

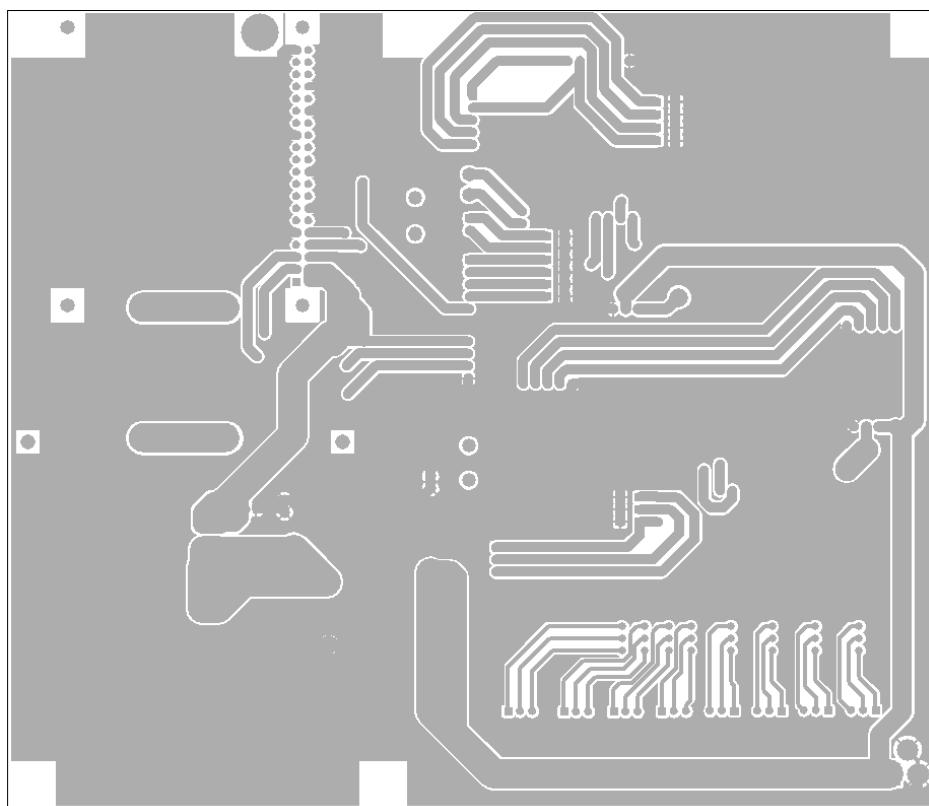
Plan d'implantation dessous :



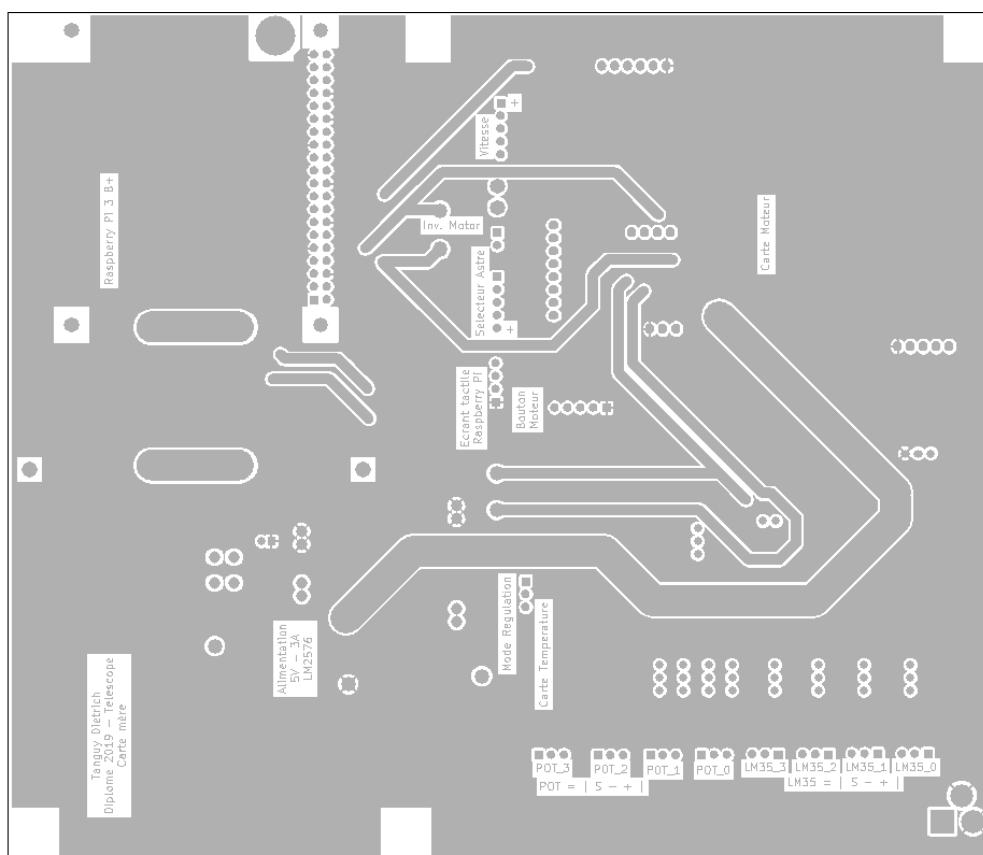
Plan d'implantation dessus :



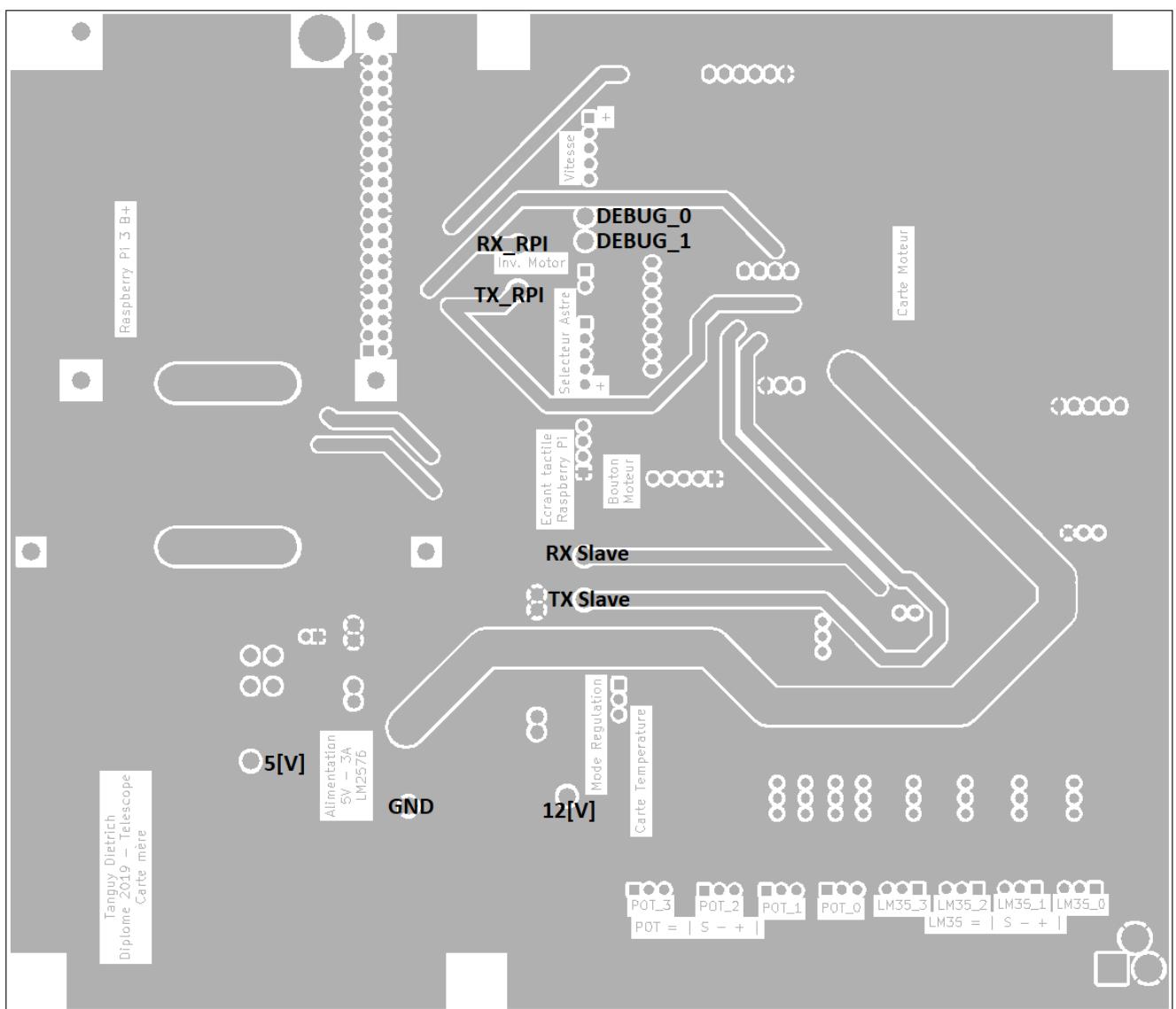
Print : vue de dessous :



Print : vue de dessus :



Points tests :



Projet / produit :	Diplôme 2019 - Télescope
Description :	
Quantités :	<p><u>1</u> pièce</p> <p><u>_</u> pièces / lot de fabrication</p> <p><u>_</u> pièces / an</p>

Circuit :

Nom :	Carte de gestion de la température		
Numéro :	ND (Non disponible)		
Révision :	0		
Date de création :	17.09.2018	Auteur :	Tanguy Dietrich

Historique des modifications :

Rév.	Date	Auteur	Détails des modifications
01			
02			
03			
04			
05			
06			
07			
08			
09			
10			

Spécification du PCB :

Nature du matériel isolant :	<input type="checkbox"/> Bakélite	<input type="checkbox"/> FR-2	<input type="checkbox"/>	_____	
	<input type="checkbox"/> Epoxy	<input checked="" type="checkbox"/> FR-4	<input type="checkbox"/>	_____	
	<input type="checkbox"/> Autre : _____				
Epaisseur du PCB :	<input type="checkbox"/> 0.5 mm	<input type="checkbox"/> 0.8 mm	<input type="checkbox"/> 1.0 mm	<input type="checkbox"/> 1.2 mm	
	<input checked="" type="checkbox"/> 1.6 mm	<input type="checkbox"/> 2.0 mm	<input type="checkbox"/> autre : _____ mm		
Nombre de couches :	<input type="checkbox"/> Simple face				
	<input checked="" type="checkbox"/> Double face				
	<input type="checkbox"/> Multicouches, nombre de couches = X				
Epaisseur des couches cuivre :					
Externe	<input checked="" type="checkbox"/> 18 µ	<input type="checkbox"/> 35 µ	<input type="checkbox"/> 75 µ	<input type="checkbox"/> 105 µ	
				<input type="checkbox"/> autre : _____	
Interne	<input type="checkbox"/> 18 µ	<input type="checkbox"/> 35 µ	<input type="checkbox"/> 75 µ	<input type="checkbox"/> 105 µ	
				<input type="checkbox"/> autre : _____	
Largeur minimum des pistes :	0.508	mm	/	20m	inch
Largeur minimum entre pistes :	0.254	mm	/	10m	inch

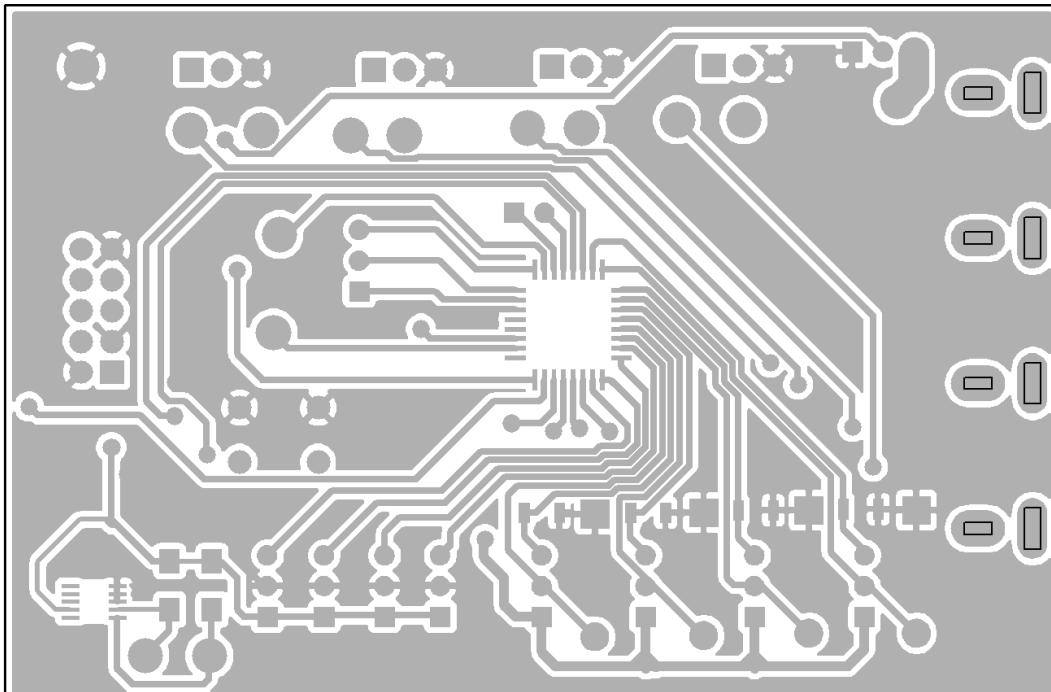
Liste des documents :

<input checked="" type="checkbox"/>	Liste du matériel	Page 3
<input checked="" type="checkbox"/>	Dimensions du PCB	Page 4
<input checked="" type="checkbox"/>	Plan de perçage	Page 5
<input checked="" type="checkbox"/>	Plan d'implantation dessous	Page 6
<input checked="" type="checkbox"/>	Plan d'implantation dessus	Page 6
<input checked="" type="checkbox"/>	Print : vue de dessous	Page 7
<input checked="" type="checkbox"/>	Print : vue de dessus	Page 7
<input checked="" type="checkbox"/>	Points tests	Page 8
<input type="checkbox"/>	Fichier Gerber	_____

Liste du matériel :

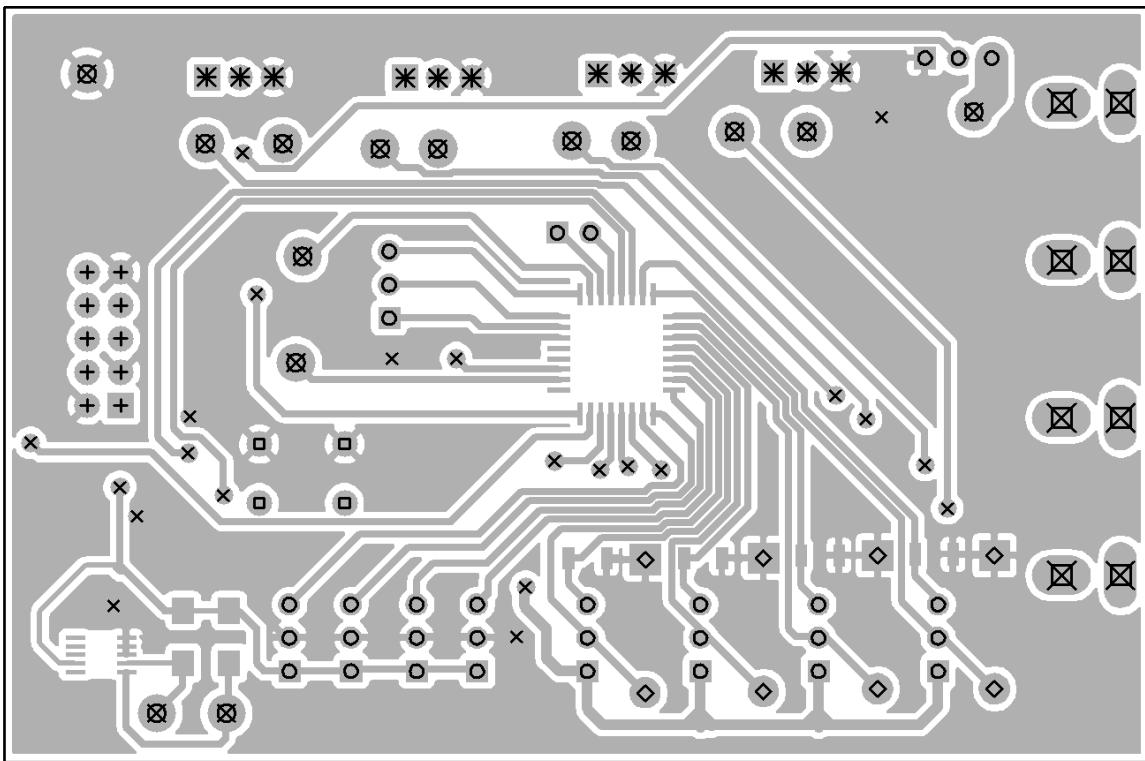
Références	Composant	Référence fabricant	Quantité	Pu [CHF]	Ptot [CHF]
BP1	Bouton Pousoir	301-18-595	1	0.2111	0.2111
C1	100nF SMD 1206	300-67-988	1	0.1094	0.1094
C11	1uF SMD 1206	300-86-875	1	0.0496	0.0496
D1,D2,D3,D4	1N4148	300-92-887	4	0.0175	0.07
J1; J2; J3; J4; J5; J6; J7; J8; J9; J13	Connecteur 3 pins male	300-93-644	10	0.1317	1.317
J12	Connecteur 10 pins	300-12-021	1	0.87	0.87
J18; J19; J24; J25	Connecteur RCJ	490-RCJ-041	4	1.36	5.44
Q1; Q2; Q3; Q4	IRL3705 TO-220	171-18-037	4	1.91	7.64
R1; R2; R3; R4,	Résistance 18k 1206	300-56-931	4	0.0344	0.1376
R5; R6 ;R15 ;R16 ;R17	Résistance 1k 1206	300-56-986	5	0.0285	0.1425
U1	EEH210	300-73-044	1	10.30	10.30
U7	C8051F38C		1	1.44	1.44
PCB	PCB carte température		1	1	1
LM35	LM35 capteur température	173-26-716	4	3.2	12.8
potentiomètre		164-35-367	4	3.65	14.6
Prix total =					55.4432 [CHF]

Dimensions de la carte :



La carte fait 87.1mm x 57.5 mm

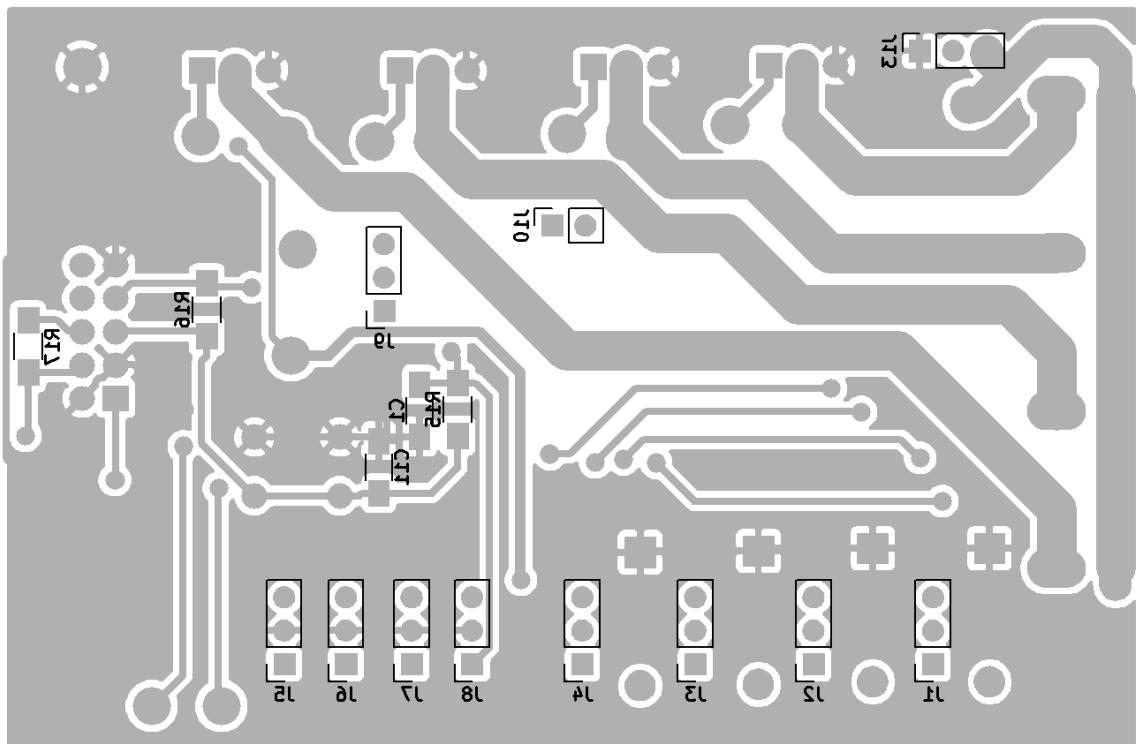
Plan de perçage :



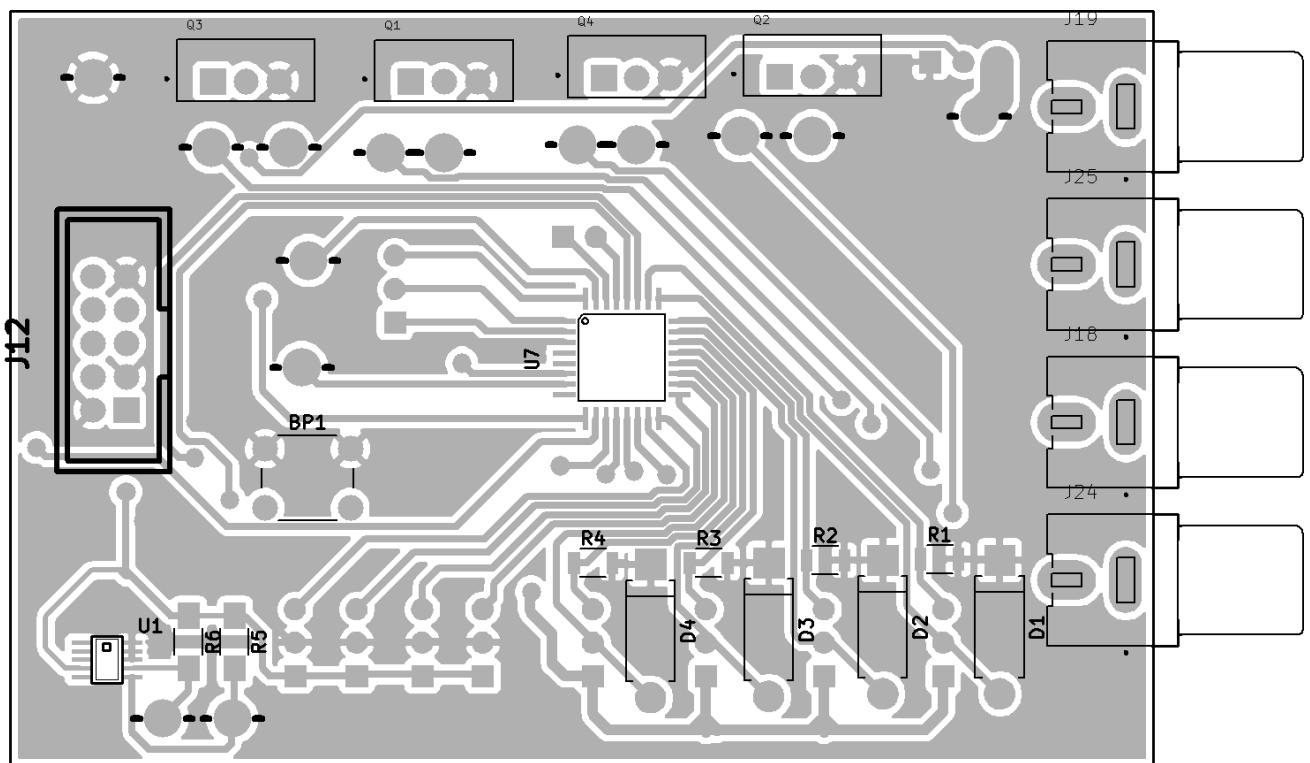
Drill Map:

x	0.80mm	/	0.031"	(22 holes)
o	1.00mm	/	0.039"	(32 holes)
+	1.00mm	/	0.039"	(10 holes)
□	1.10mm	/	0.043"	(4 holes)
◊	1.20mm	/	0.047"	(8 holes)
■	1.30mm	/	0.051"	(14 holes)
*	1.34mm	/	0.053"	(12 holes)
☒	2.10mm	/	0.083"	(8 holes)

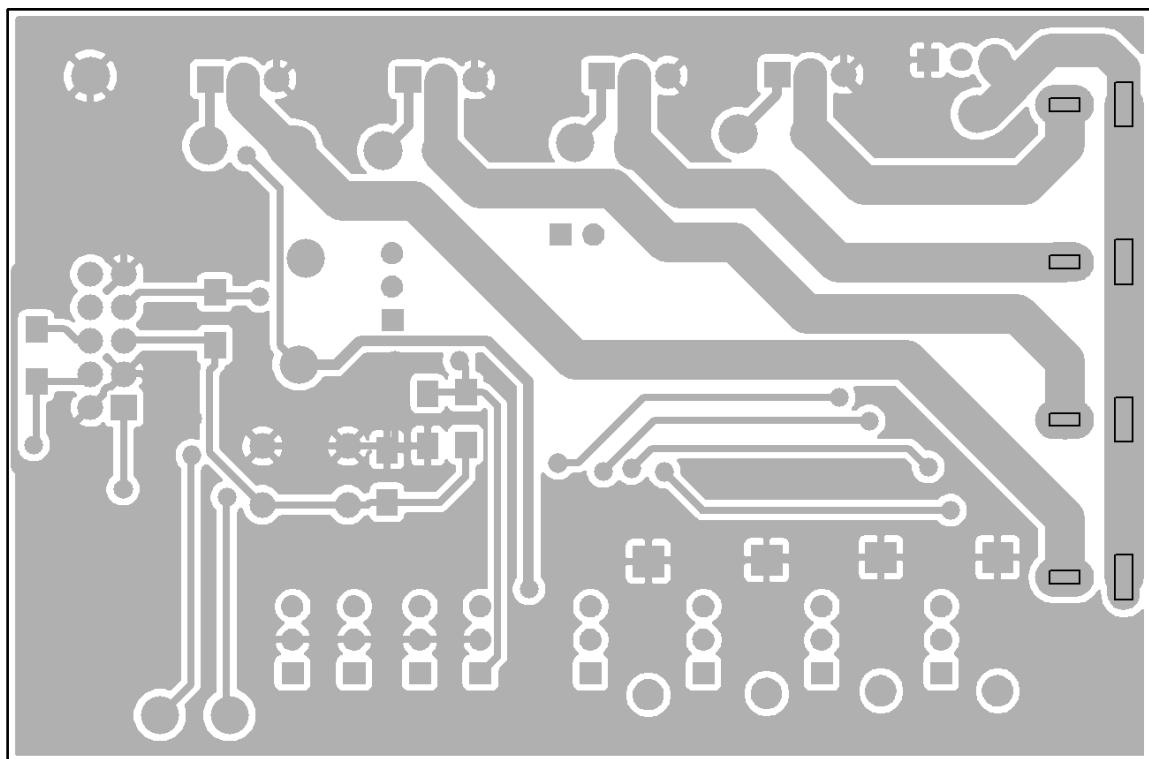
Plan d'implantation dessous :



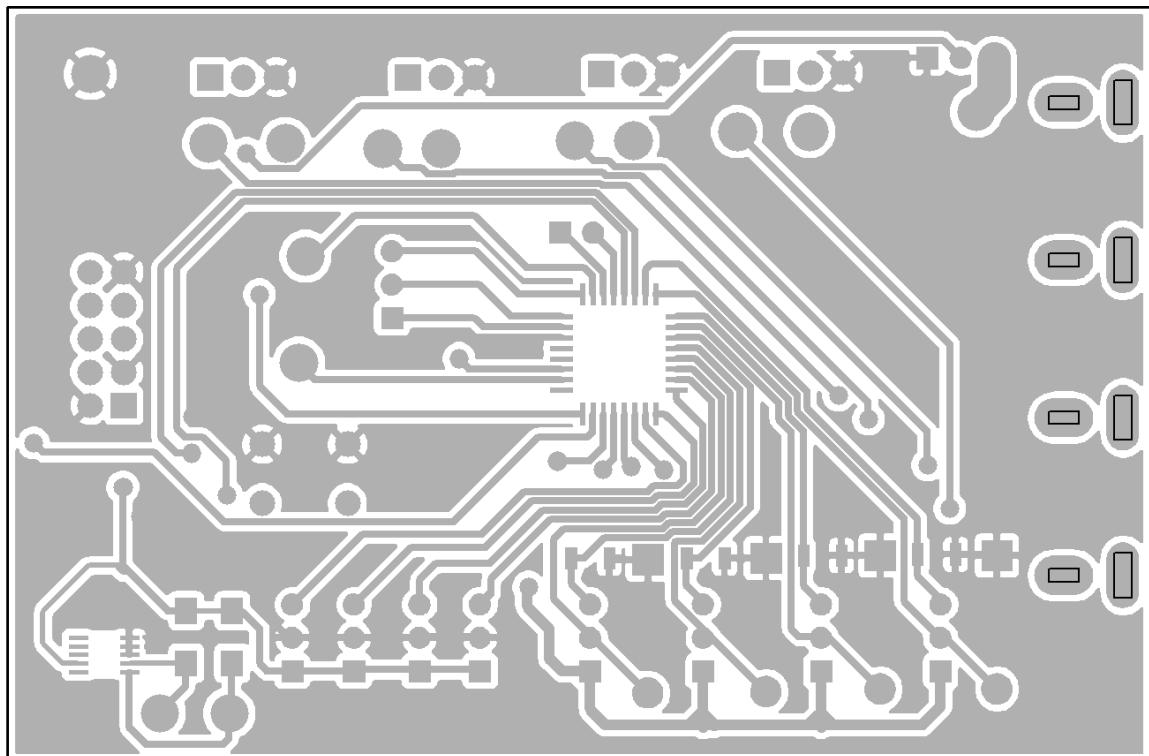
Plan d'implantation dessus :



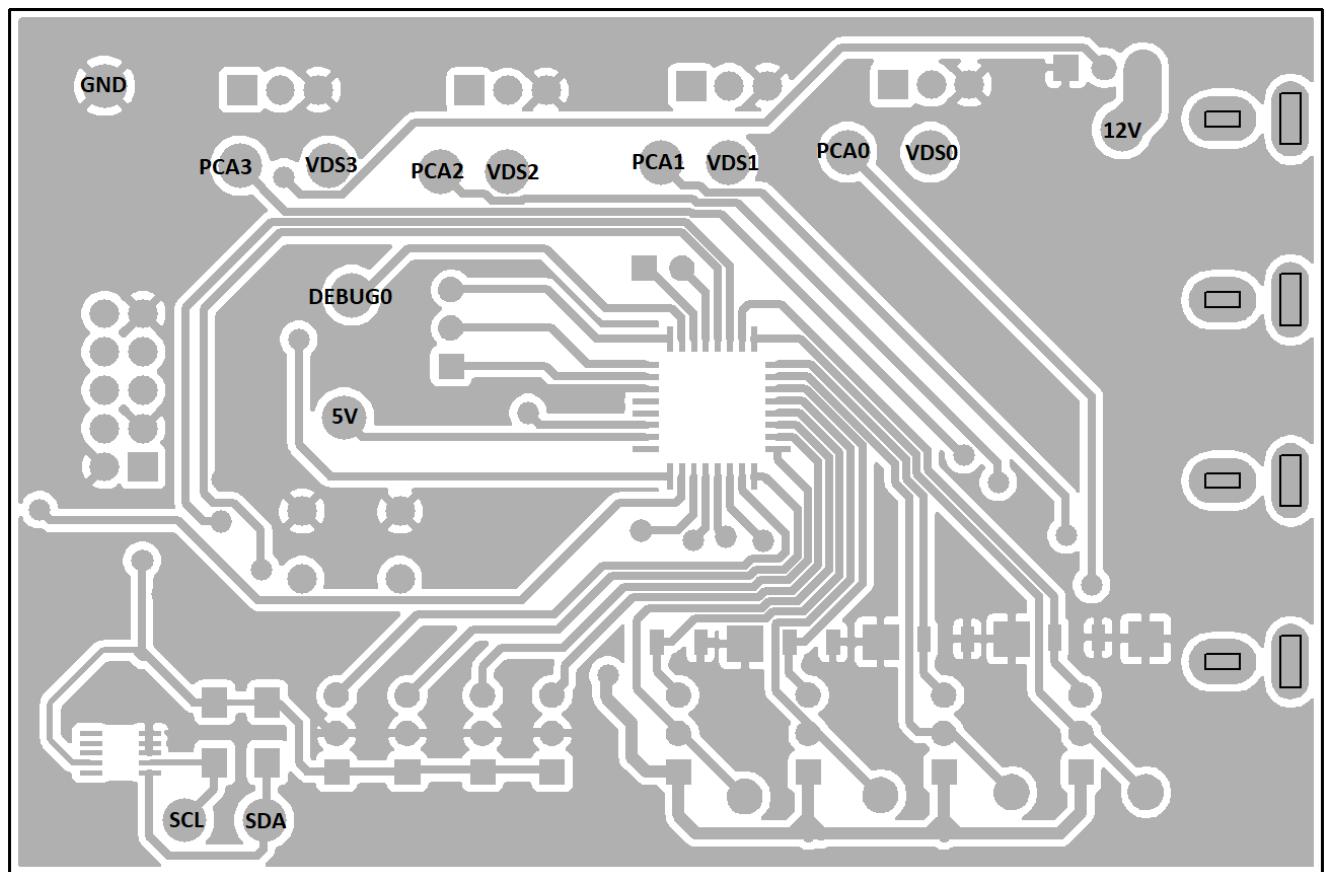
Print : vue de dessous :



Print : vue de dessus :



Points tests :



Projet / produit :	Diplôme 2019 - Télescope
Description :	
Quantités :	<p><u>1</u> pièce</p> <p><u>_</u> pièces / lot de fabrication</p> <p><u>_</u> pièces / an</p>

Circuit :

Nom :	Alimentation 5V – 3A		
Numéro :	ND (non disponible)		
Révision :	0		
Date de création :	17.09.2018	Auteur :	Tanguy Dietrich

Historique des modifications :

Rév.	Date	Auteur	Détails des modifications
01			
02			
03			
04			
05			
06			
07			
08			
09			
10			

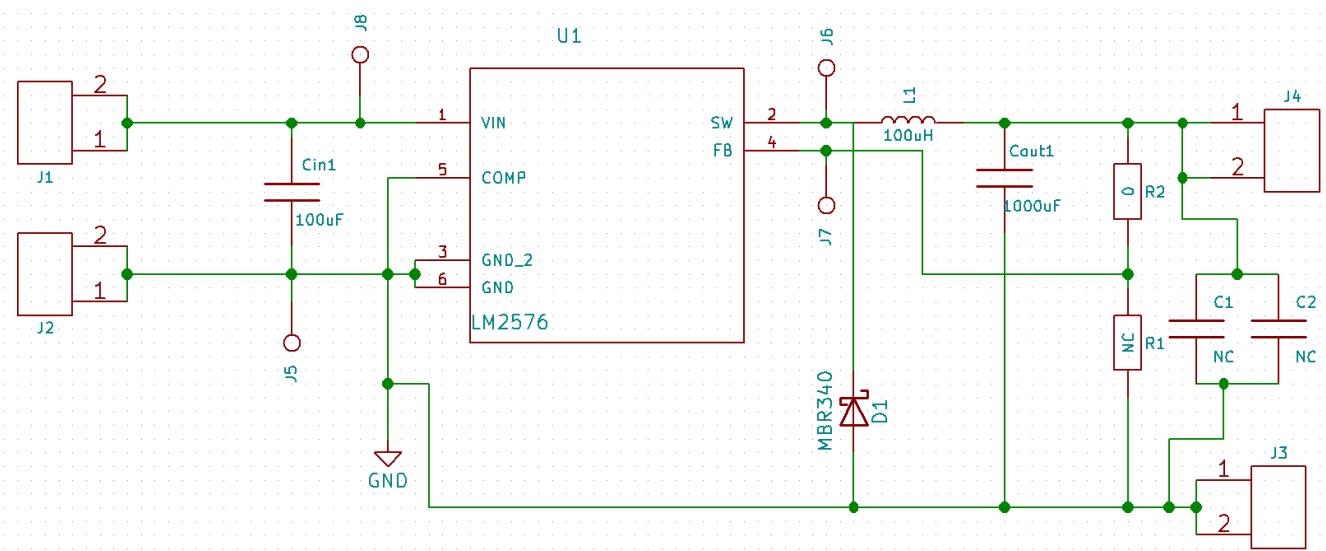
Spécification du PCB :

Nature du matériel isolant :	<input type="checkbox"/> Bakélite	<input type="checkbox"/> FR-2	<input type="checkbox"/> _____		
	<input type="checkbox"/> Epoxy	<input checked="" type="checkbox"/> FR-4	<input type="checkbox"/> _____		
	<input type="checkbox"/> Autre : _____				
Epaisseur du PCB :	<input type="checkbox"/> 0.5 mm	<input type="checkbox"/> 0.8 mm	<input type="checkbox"/> 1.0 mm	<input type="checkbox"/> 1.2 mm	
	<input checked="" type="checkbox"/> 1.6 mm	<input type="checkbox"/> 2.0 mm	<input type="checkbox"/> autre : _____ mm		
Nombre de couches :	<input type="checkbox"/> Simple face				
	<input checked="" type="checkbox"/> Double face				
	<input type="checkbox"/> Multicouches, nombre de couches = X				
Epaisseur des couches cuivre :					
Externe	<input checked="" type="checkbox"/> 18 µ	<input type="checkbox"/> 35 µ	<input type="checkbox"/> 75 µ	<input type="checkbox"/> 105 µ	
				<input type="checkbox"/> autre : _____	
Interne	<input type="checkbox"/> 18 µ	<input type="checkbox"/> 35 µ	<input type="checkbox"/> 75 µ	<input type="checkbox"/> 105 µ	
				<input type="checkbox"/> autre : _____	
Largeur minimum des pistes :	0.508	mm	/	20m	inch
Largeur minimum entre pistes :	0.254	mm	/	10m	inch

Liste des documents :

<input checked="" type="checkbox"/>	Schéma électrique	Page 3
<input checked="" type="checkbox"/>	Liste du matériel	Page 3
<input checked="" type="checkbox"/>	Dimensions du PCB	Page 4
<input checked="" type="checkbox"/>	Plan de perçage	Page 5
<input checked="" type="checkbox"/>	Plan d'implantation dessous	Page 6
<input checked="" type="checkbox"/>	Plan d'implantation dessus	Page 6
<input checked="" type="checkbox"/>	Print : vue de dessous	Page 7
<input checked="" type="checkbox"/>	Print : vue de dessus	Page 7
<input checked="" type="checkbox"/>	Points tests	Page 8
<input type="checkbox"/>	Fichier Gerber	_____

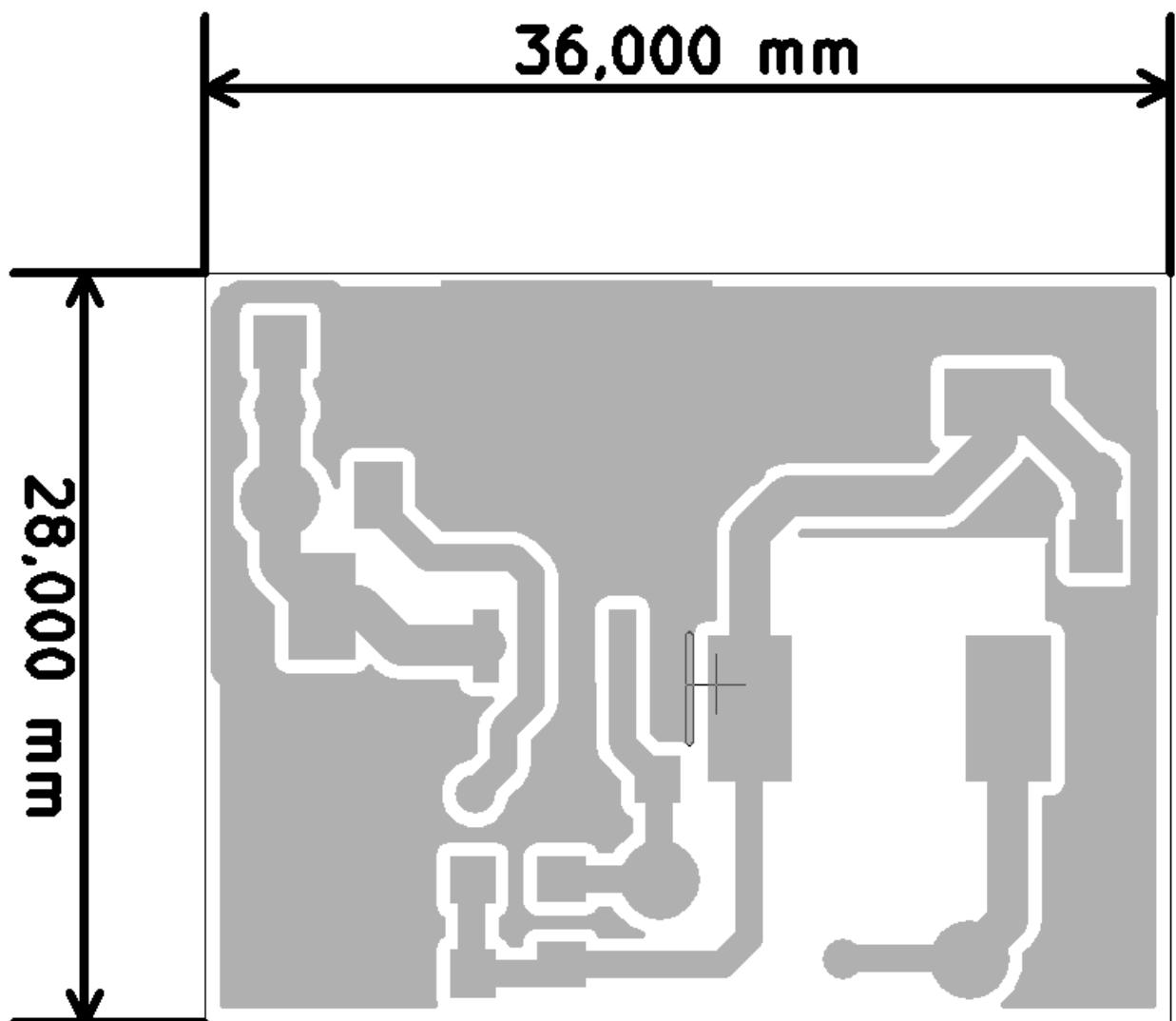
Schéma électrique :



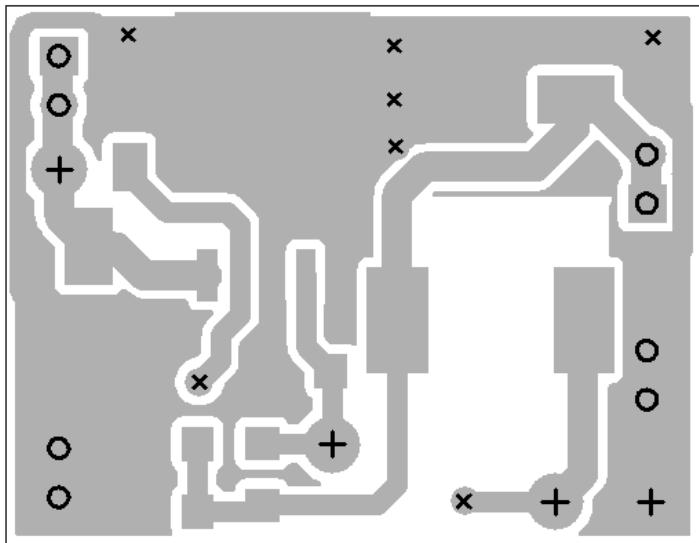
Liste du matériel :

Références	Composant	Référence fabricant	Quantité	Pu [CHF]	Ptot [CHF]
J1...J4	Connecteur 2 poles	300-93-652	4	0.176	0.704
Cin1	Condensateur 100uF SMD	167-31-200	1	0.2172	0.2172
Cout1	Condensateur 1000uF	300-92-035	1	0.2415	0.2415
J5...J8	Cosse		4		
U1	LM2576	300-19-258	1	2.35	2.35
D1	MBR340	170-09-528	1	0.4106	0.4106
L1	100uH SMD	301-28-799	1	0.6928	0.6928
Prix total =					4.6161 [CHF]

Dimensions de la carte :



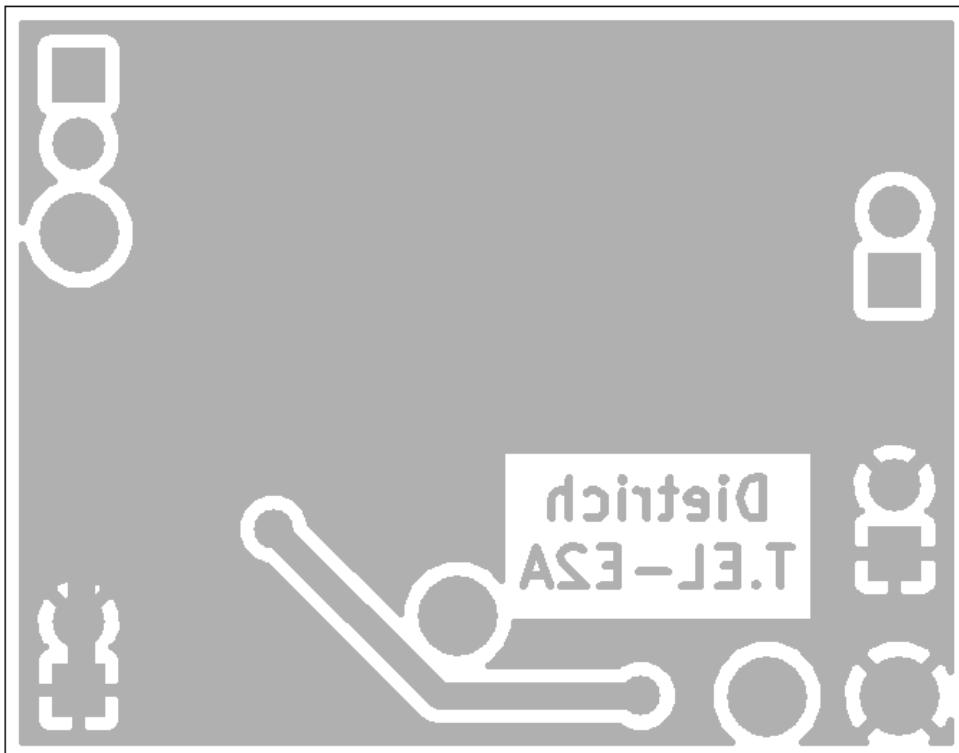
Plan de perçage :



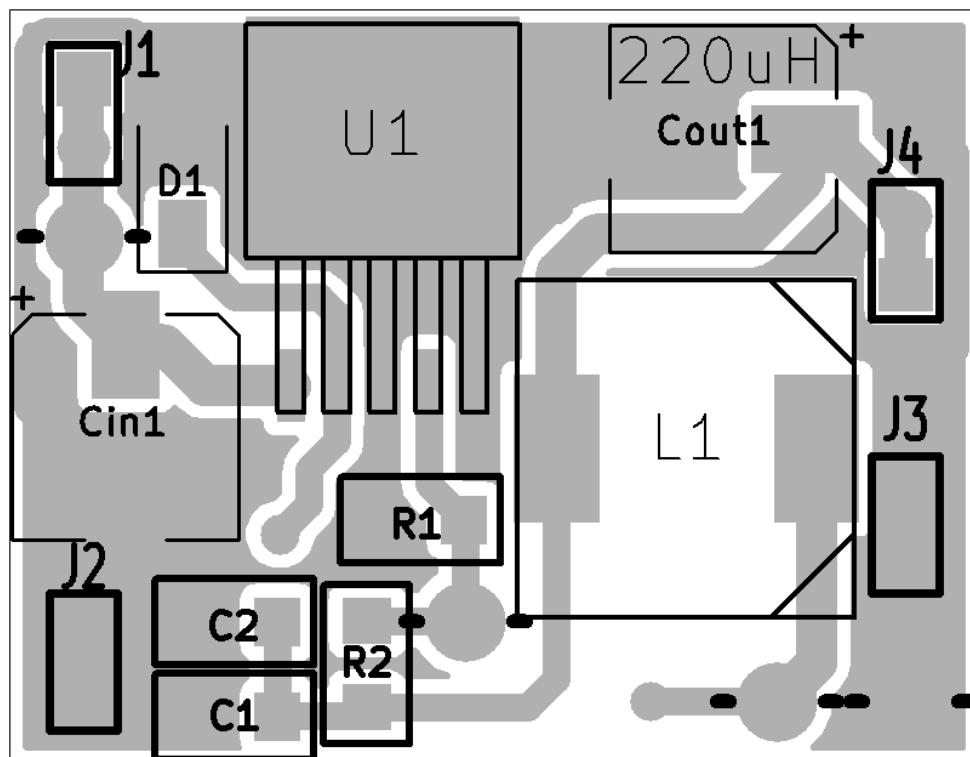
Drill Map:

- * 0.60mm / +0.024" (7 holes)
- 1.02mm / 0.040" (8 holes)
- + 1.30mm / 0.051" (4 holes)

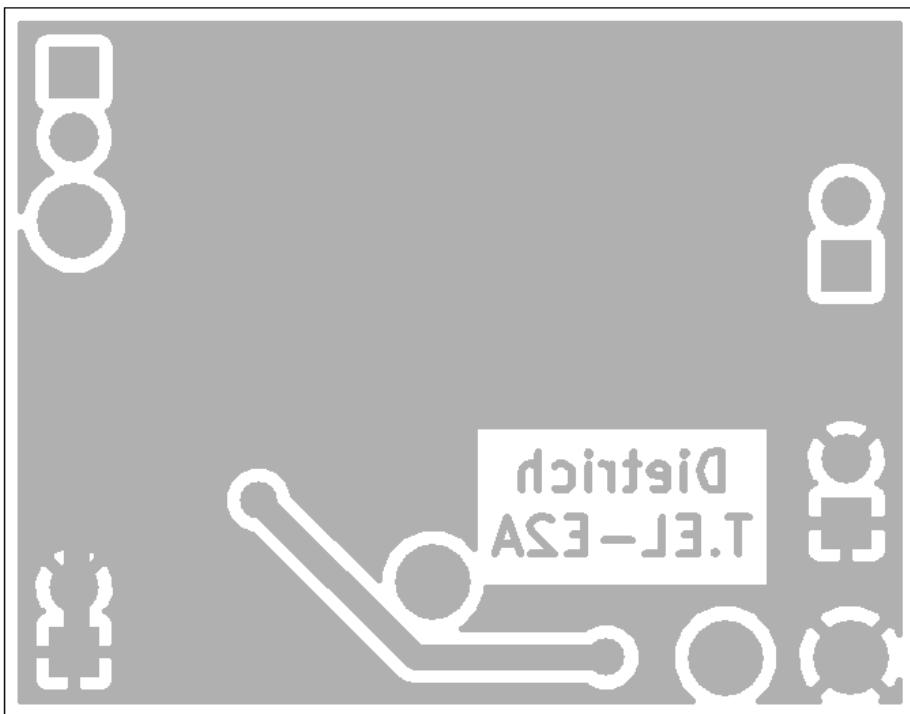
Plan d'implantation dessous :



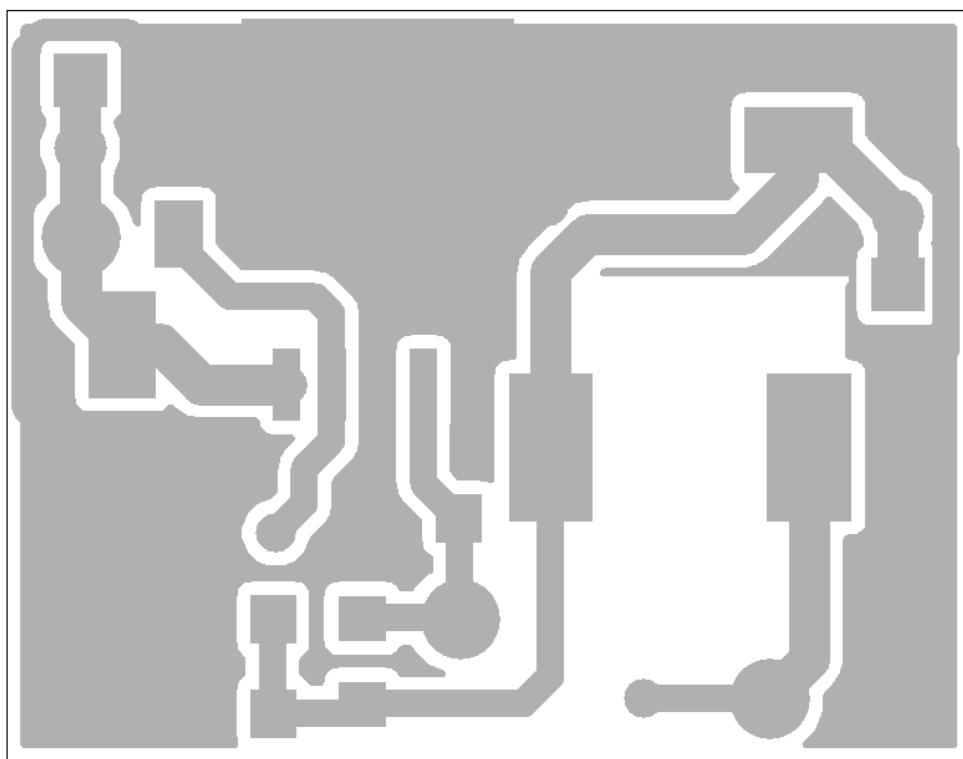
Plan d'implantation dessus :



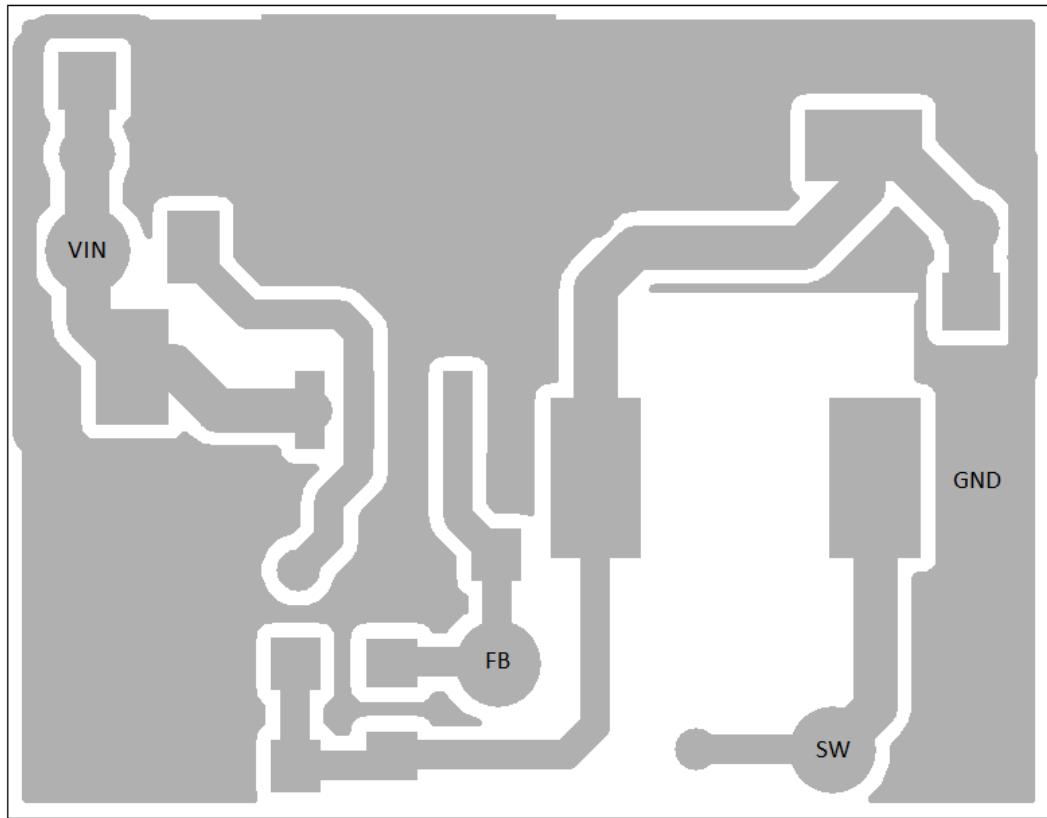
Print : vue de dessous :



Print : vue de dessus :



Points tests :



Projet / produit :	Diplôme 2019 - Télescope
Description :	
Quantités :	<p><u>1</u> pièce</p> <p><u>_</u> pièces / lot de fabrication</p> <p><u>_</u> pièces / an</p>

Circuit :

Nom :	Carte de gestion des moteur		
Numéro :	ND (Non disponible)		
Révision :	0		
Date de création :	17.09.2018	Auteur :	Tanguy Dietrich

Historique des modifications :

Rév.	Date	Auteur	Détails des modifications
01			
02			
03			
04			
05			
06			
07			
08			
09			
10			

Spécification du PCB :

Nature du matériel isolant :	<input type="checkbox"/> Bakélite	<input type="checkbox"/> FR-2	<input type="checkbox"/> _____		
	<input type="checkbox"/> Epoxy	<input checked="" type="checkbox"/> FR-4	<input type="checkbox"/> _____		
	<input type="checkbox"/> Autre : _____				
Epaisseur du PCB :	<input type="checkbox"/> 0.5 mm	<input type="checkbox"/> 0.8 mm	<input type="checkbox"/> 1.0 mm	<input type="checkbox"/> 1.2 mm	
	<input checked="" type="checkbox"/> 1.6 mm	<input type="checkbox"/> 2.0 mm	<input type="checkbox"/> autre : _____ mm		
Nombre de couches :	<input type="checkbox"/> Simple face				
	<input checked="" type="checkbox"/> Double face				
	<input type="checkbox"/> Multicouches, nombre de couches = X				
Epaisseur des couches cuivre :					
Externe	<input checked="" type="checkbox"/> 18 µ	<input type="checkbox"/> 35 µ	<input type="checkbox"/> 75 µ	<input type="checkbox"/> 105 µ	
				<input type="checkbox"/> autre : _____	
Interne	<input type="checkbox"/> 18 µ	<input type="checkbox"/> 35 µ	<input type="checkbox"/> 75 µ	<input type="checkbox"/> 105 µ	
				<input type="checkbox"/> autre : _____	
Largeur minimum des pistes :	0.508	mm	/	20m	inch
Largeur minimum entre pistes :	0.254	mm	/	10m	inch

Liste des documents :

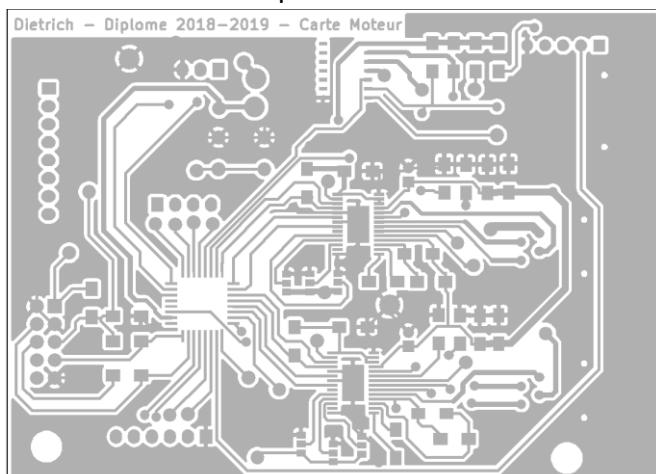
<input checked="" type="checkbox"/>	Schéma électrique	Page 3
<input checked="" type="checkbox"/>	Liste du matériel	Page 3
<input checked="" type="checkbox"/>	Dimensions du PCB	Page 4
<input checked="" type="checkbox"/>	Plan de perçage	Page 5
<input checked="" type="checkbox"/>	Plan d'implantation dessous	Page 6
<input checked="" type="checkbox"/>	Plan d'implantation dessus	Page 6
<input checked="" type="checkbox"/>	Print : vue de dessous	Page 7
<input checked="" type="checkbox"/>	Print : vue de dessus	Page 7
<input checked="" type="checkbox"/>	Points tests	Page 8
<input type="checkbox"/>	Fichier Gerber	_____

Liste du matériel :

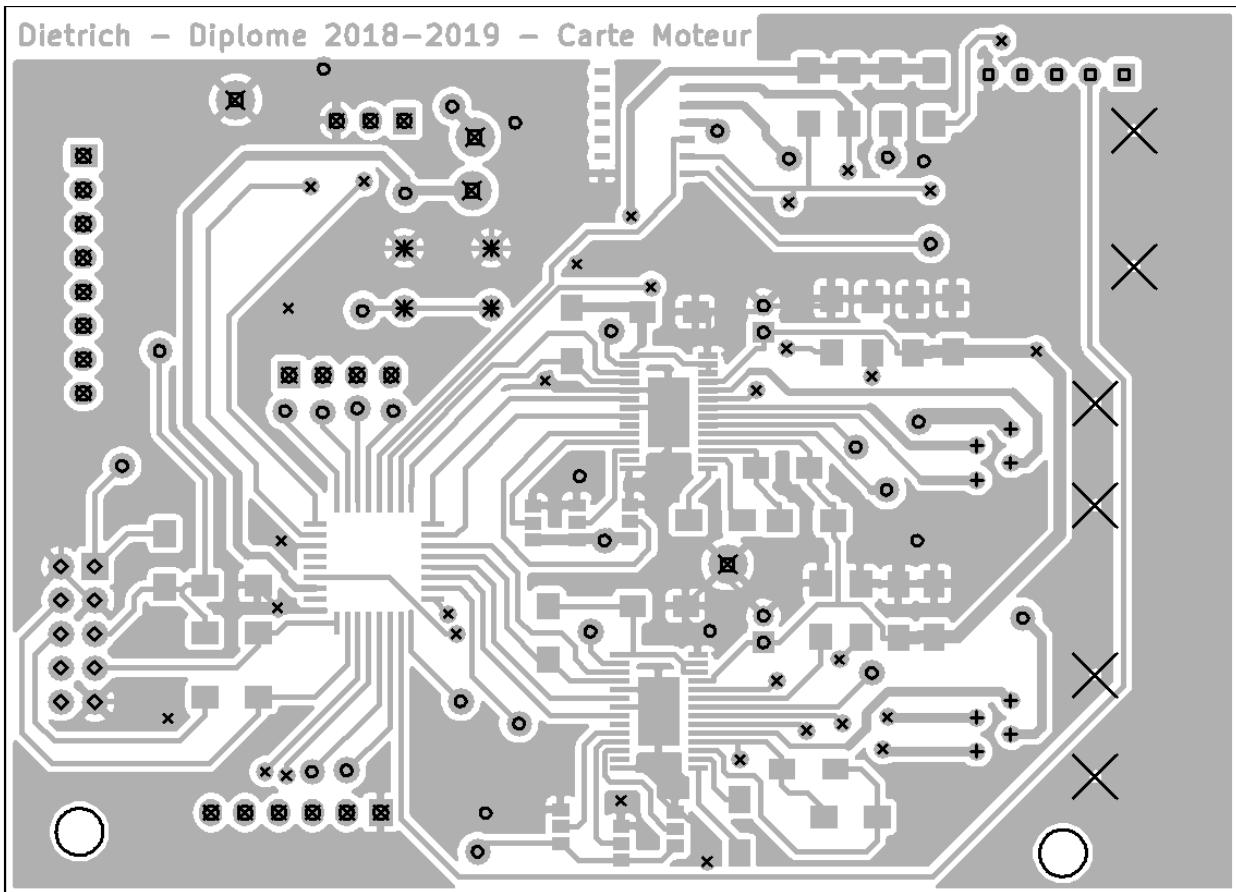
Références	Composant	Référence fabricant	Quantité	Pu [CHF]	Ptot [CHF]
BP1	Bouton Poussoir 6mm	301-18-595	1	0.2111	0.2111
C1; C2	0.47uF 1206	300-86-945	2	0.052	0.104
C3; C4; C5; C6; C7; C12	0.1uF 1206	300-67-988	6	0.1094	0.6564
C8; C13	100uF radial 5mm	300-92-017	2	0.063	0.126
C9; C10	0.01uF 1206	300-66-368	2	0.144	0.288
C11	1uF 1206	300-86-875	1	0.0496	0.0496
J1	Barrette 8 pin male	300-93-649	1	0.4189	0.4189
J2	Barrette 4 pin male	300-93-645	1	0.1761	0.1761
J3	Connecteur RJ12	Mouser :538-95501-6669	1	0.776	0.776
J4	Barrette 6 pin male	300-93-647	1	0.2991	0.2991
J5	Barrette 3 pin male	300-93-644	1	0.1317	0.1317
J12	Connecteur 10 pin	300-12-021	1	0.87	0.87
J13; J14; J15; J16	Cosse		4		
J17; J18	RJ11	Mouser : 538-95009-7441	2	0.797	1.594
J19	Barrette 5 pin male	300-93-646	1	0.2346	0.2346
R1; R2; R9; R10	Résistance 10k SMD 1206	300-56-987	4	0.0285	0.114
R3; R4; R5; R6 ;R15 ;R16 ;R17	Résistance 1k SMD 1206	300-56-986	7	0.0285	0.1995
R7; R8	Résistance 1M SMD 1206	300-56-919	2	0.0305	0.061
R11; R12; R13; R14	Résistance 5.6Ohm SMD 1206	160-18-949	4	0.1071	0.4284
U1	MCP23008	300-46-590	1	1.09	1.09
U2	CD4081	300-18-498	1	0.4809	0.4809
U3 ;U4	DRV8825	Mouser :595-DRV8825PWPR	2	3.72	7.44
U7	C8051F38C		1	1.41	1.41
PCB	PCB Carte température		1	1	1
Prix total =					18.1567 [CHF]

Dimensions de la carte :

La carte fait 92.3mm par 66.5mm



Plan de perçage :



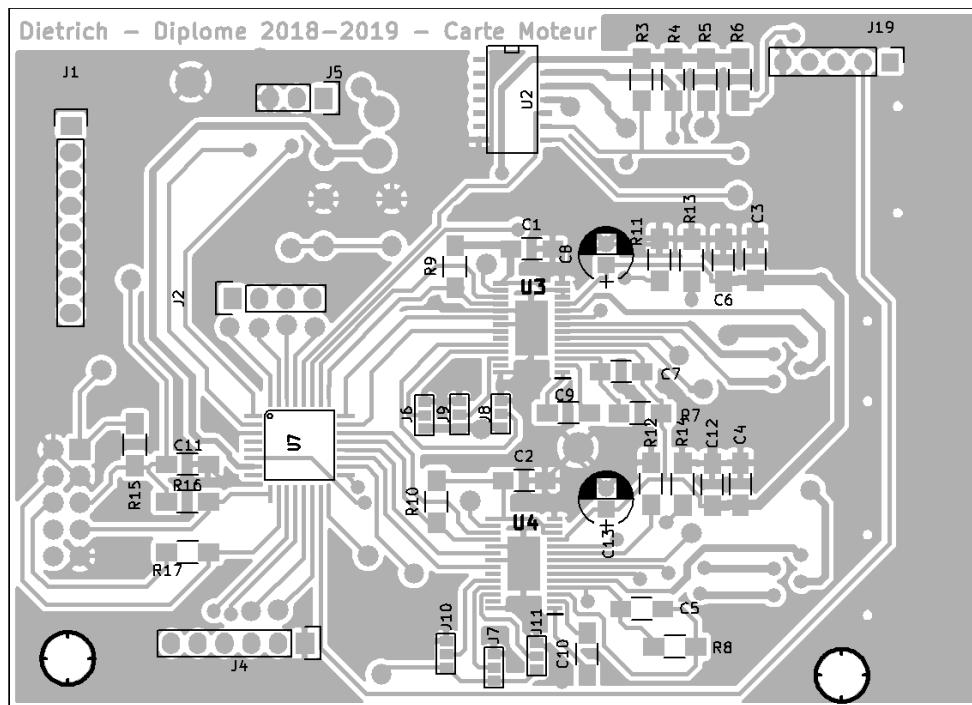
Drill Map:

- ✗ 0.60mm / 0.024" {31 holes}
- 0.80mm / 0.031" {37 holes}
- + 0.89mm / 0.035" {8 holes}
- 1.00mm / 0.039" {5 holes}
- ◊ 1.00mm / 0.039" {10 holes}
- 1.02mm / 0.040" {21 holes}
- * 1.10mm / 0.043" {4 holes}
- ▣ 1.30mm / 0.051" {4 holes}

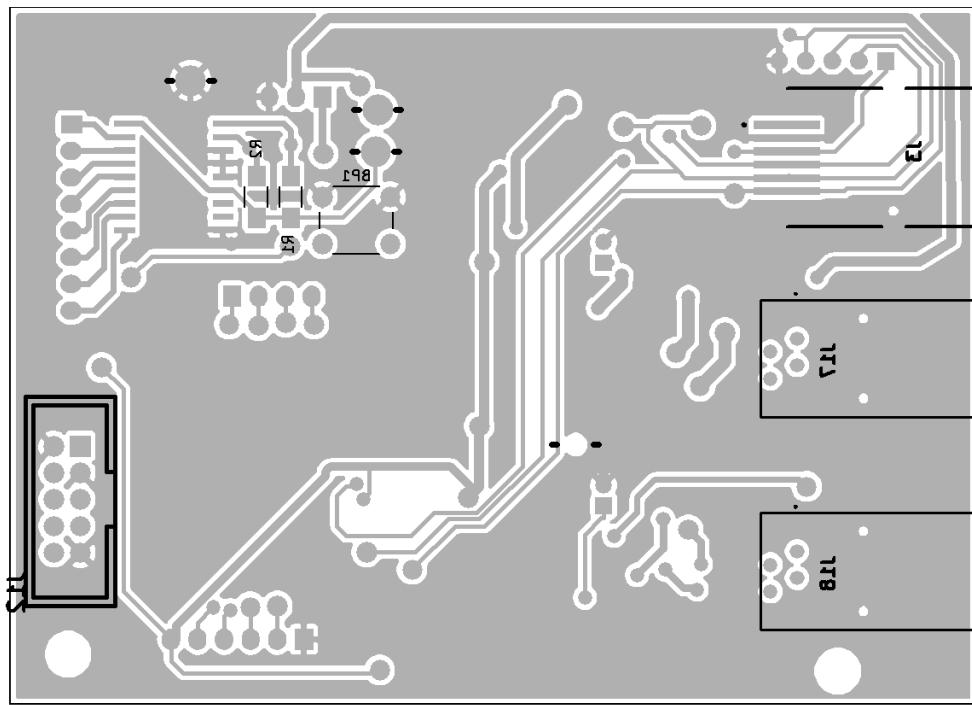
Drill Map:

- ✗ 3.25mm / 0.128" {6 holes} {not plated}
- 3.50mm / 0.138" {2 holes} {not plated}

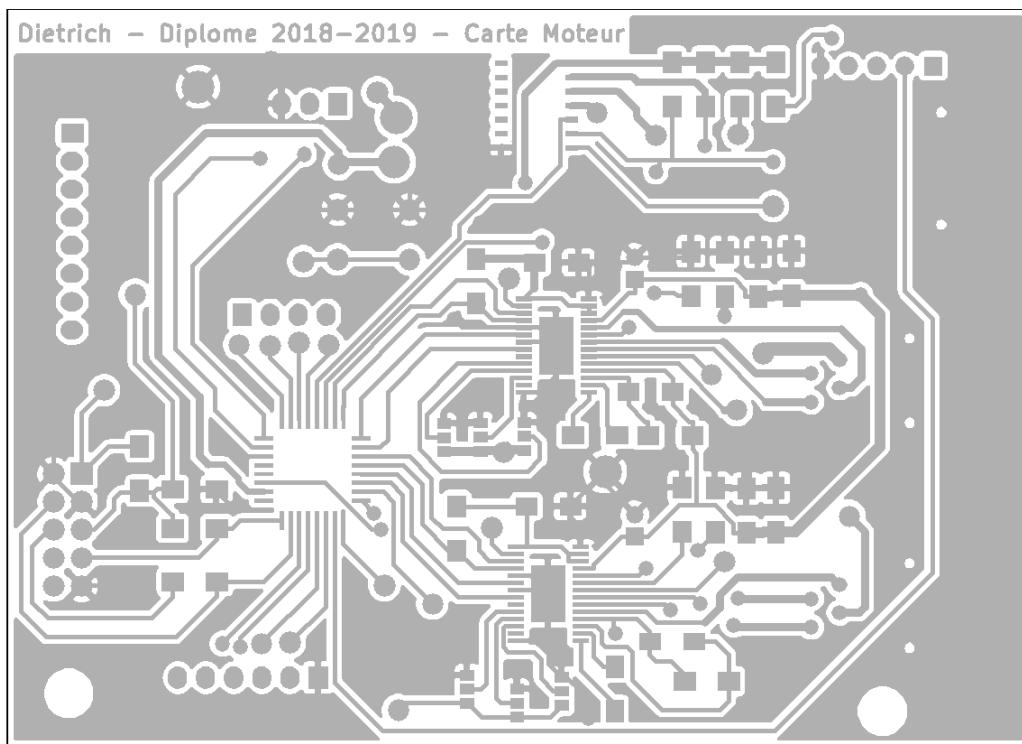
Plan d'implantation dessous :



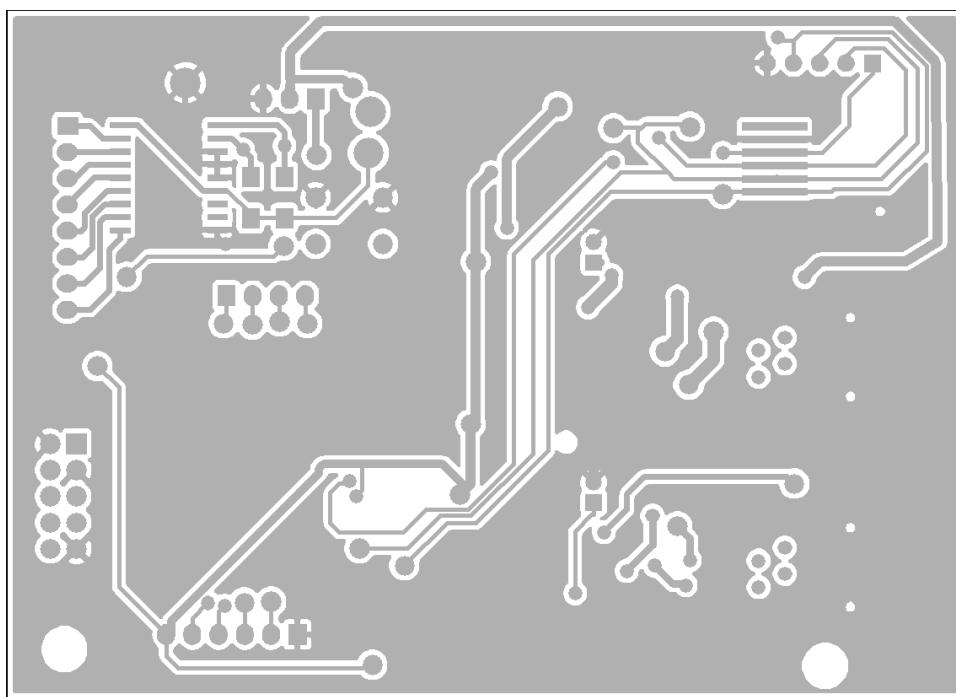
Plan d'implantation dessus :



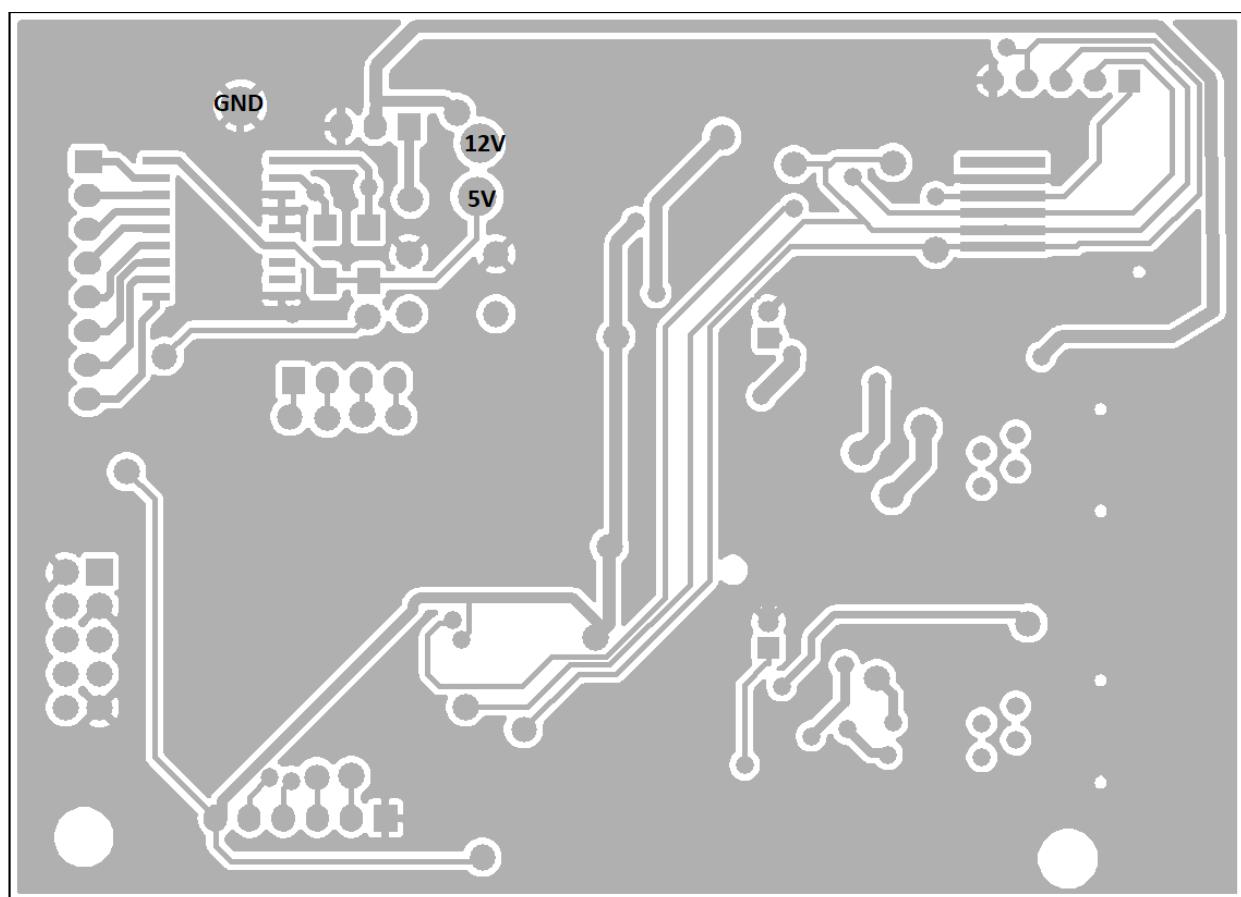
Print : vue de dessous :



Print : vue de dessus :



Points tests :



9.6. Listing des codes sources

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 #https://github.com/Svenito/Pyscope/blob/master/tcp_test.py
4 import struct
5 import time
6 import socket, select
7 import serial
8 import time
9 from math import *
10
11 M_PI = 3.1415926535897932385
12 MOVE_SPEED = 48
13 flagPosition=1
14 UNITE_DEPLACEMENT = 49843#correspond a 15.04"arc
15 DEPLACEMENT_SECONDS_DEC = UNITE_DEPLACEMENT * MOVE_SPEED
16 DEPLACEMENT_SECONDS_RA_POS = UNITE_DEPLACEMENT * (MOVE_SPEED+1)#le ciel      ↵
    continue de bouger en meme temps
17 DEPLACEMENT_SECONDS_RA_NEG = UNITE_DEPLACEMENT * (MOVE_SPEED-1)
18 serial = serial.Serial("/dev/ttyS0",57600,timeout=2)#
19 # List of socket objects that are currently open
20 open_sockets = []
21
22 # AF_INET means IPv4.
23 # SOCK_STREAM means a TCP connection.
24 # SOCK_DGRAM would mean an UDP "connection".
25 listening_socket = socket.socket( socket.AF_INET, socket.SOCK_STREAM )
26 listening_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
27
28 # The parameter is (host, port).
29 # The host, when empty or when 0.0.0.0, means to accept connections for
30 # all IP addresses of current machine. Otherwise, the socket will bind
31 # itself only to one IP.
32 # The port must greater than 1023 if you plan running this script as a
33 # normal user. Ports below 1024 require root privileges.
34 listening_socket.bind( ("", 10001) )
35
36 # The parameter defines how many new connections can wait in queue.
37 # Note that this is NOT the number of open connections (which has no limit).
38 # Read listen(2) man page for more information.
39 listening_socket.listen(5)
40
41 current_position = []
42
43 def StopRA():
44     serial.write("RA0#")
45
46 def StopDEC():
47     serial.write("DEC0#")
48
49 def RunDEC(direction):
50     if direction==1:
51         serial.write("DEC+#")
52     else:
53         serial.write("DEC-#")
54
55 def RunRA(direction):
```

```
56     if direction==1:
57         serial.write("RA+#")
58     else:
59         serial.write("RA-#")
60
61
62 def printit(ra_int, dec_int):
63     h = ra_int
64     d = floor(0.5 + dec_int*(360*3600*1000/4294967296.0));
65     dec_sign = ''
66     if d >= 0:
67         if d > 90*3600*1000:
68             d = 180*3600*1000 - d;
69             h += 0x80000000;
70         dec_sign = '+';
71     else:
72         if d < -90*3600*1000:
73             d = -180*3600*1000 - d;
74             h += 0x80000000;
75         d = -d;
76         dec_sign = '-';
77
78
79     h = floor(0.5+h*(24*3600*1000/4294967296.0));
80     ra_ms = h % 10000; h /= 10000;
81     ra_s = h % 60; h /= 60;
82     ra_m = h % 60; h /= 60;
83
84     h %= 24;
85     dec_ms = d % 1000; d /= 1000;
86     dec_s = d % 60; d /= 60;
87     dec_m = d % 60; d /= 60;
88
89     print ("ra =", h,"h", ra_m,"m",ra_s,".",ra_ms)
90     print ("dec =",dec_sign, d,"d", dec_m,"m",dec_s,".",dec_ms)
91
92
93 while True:
94     # Waits for I/O being available for reading from any socket object.
95     rlist, wlist, xlist = select.select( [listening_socket] + open_sockets,    ↴
96                                         [], [] )
97     for i in rlist:
98         if i is listening_socket:
99             new_socket, addr = listening_socket.accept()
100            open_sockets.append(new_socket)
101        else:
102            data = i.recv(1024)
103            if data == "":
104                open_sockets.remove(i)
105                flagPosition=1
106                print ("Connection closed")
107            else:
108
109                data = struct.unpack("3iIi", data)
110                ra = data[3]*(M_PI/0x80000000)
111                dec = data[4]*(M_PI/0x80000000)
```

```
111             cdec = cos(dec)
112             if flagPosition==1:
113                 current_position= []
114                 current_position.append(data[3])
115                 current_position.append(data[4])
116                 flagPosition=0
117             else :
118                 desired_pos = []
119                 desired_pos.append(cos(ra)*cdec)
120                 desired_pos.append(sin(ra)*cdec)
121                 desired_pos.append(sin(dec))
122
123                 deplacement_ra = current_position[0]-data[3]
124                 if deplacement_ra>0:
125                     temps_ra=float(abs(deplacement_ra*1000)/
126                                     DEPLACEMENT_SECONDS_RA_POS) ↗
127                 else:
128                     temps_ra=float(abs(deplacement_ra*1000)/
129                                     DEPLACEMENT_SECONDS_RA_NEG) ↗
130                 deplacement_dec = current_position[1]-data[4]
131                 #temps de deplacement en millisec
132                 temps_dec=float(abs(deplacement_dec*1000)/
133                                     DEPLACEMENT_SECONDS_DEC) ↗
134
135                 print (temps_ra)
136                 print (temps_dec)
137                 end_move_ra=0
138                 end_move_dec=0
139                 moving_RA=current_position[0]
140                 moving_DEC=current_position[1]
141                 if deplacement_dec > 0:
142                     RunDEC(1)
143                 elif deplacement_dec < 0:
144                     RunDEC(0)
145                 else:
146                     pass
147
148                 if deplacement_ra > 0:
149                     RunRA(1)
150                 elif deplacement_ra < 0:
151                     RunRA(0)
152                 else:
153                     pass
154                 now_millis = int(round(time.time() * 1000))
155                 while True:
156                     millis = int(round(time.time() * 1000))
157                     time_elapsed = millis - now_millis
158                     #print time_elapsed
159                     if (time_elapsed>=temps_ra)& (end_move_ra==0):
160                         StopRA()
161                         end_move_ra=1
162
163                     if (time_elapsed>=temps_dec)& (end_move_dec==0):
164                         StopDEC()
165                         end_move_dec=1
```

```
164             if (end_move_ra==1) & (end_move_dec==1):
165                 break
166
167             current_position[0]=data[3]
168             current_position[1]=data[4]
169             reply = struct.pack("3Ii", 24, 0, time.time(),
170                                 current_position[0], current_position[1], 0)
171             for s in range(10):##Stellarium likes to recieve the
172                             coordinates 10 times.
173                 i.send(reply)
174             print
```

```
1 # -*- coding: utf-8 -*-
2 from flask import Flask, render_template, url_for, request, jsonify
3 import datetime
4 import RPi.GPIO as GPIO
5 import serial
6 import pandas
7 import json
8 app = Flask(__name__)
9
10 GPIO.setmode(GPIO.BCM)
11 chartSize=50
12
13 @app.route("/Telescope/RA_Pos/", methods = ['POST'])
14 def Move_RA_Pos():
15     with open('Config.cfg') as f:
16         config = json.load(f)
17         f.close()
18     config[ "MoveRA"]="RA+#"
19     config[ "ChangeRA"]=True
20     with open('Config.cfg', 'w') as outfile:
21         json.dump(config, outfile)
22         outfile.close()
23     return str(1)
24
25 @app.route("/Telescope/RA_Neg/", methods = ['POST'])
26 def Move_RA_Neg():
27     with open('Config.cfg') as f:
28         config = json.load(f)
29         f.close()
30     config[ "MoveRA"]="RA-#"
31     config[ "ChangeRA"]=True
32     with open('Config.cfg', 'w') as outfile:
33         json.dump(config, outfile)
34         outfile.close()
35     return str(1)
36
37 @app.route("/Telescope/RA_Stop/", methods = ['POST'])
38 def Move_RA_Stop():
39     with open('Config.cfg') as f:
40         config = json.load(f)
41         f.close()
42     config[ "MoveRA"]="RA0#"
43     config[ "ChangeRA"]=True
44     with open('Config.cfg', 'w') as outfile:
45         json.dump(config, outfile)
46         outfile.close()
47     return str(1)
48
49 @app.route("/Telescope/DEC_Pos/", methods = ['POST'])
50 def Move_DEC_Pos():
51     with open('Config.cfg') as f:
52         config = json.load(f)
53         f.close()
54     config[ "MoveDEC"]="DEC+#"
55     config[ "ChangeDEC"]=True
56     with open('Config.cfg', 'w') as outfile:
```

```
57     json.dump(config, outfile)
58     outfile.close()
59     return str(1)
60
61 @app.route("/Telescope/DEC_Neg/", methods = ['POST'])
62 def Move_DEC_Neg():
63     with open('Config.cfg') as f:
64         config = json.load(f)
65         f.close()
66     config[ "MoveDEC"]="DEC-#"
67     config[ "ChangeDEC"]=True
68     with open('Config.cfg', 'w') as outfile:
69         json.dump(config, outfile)
70         outfile.close()
71     return str(1)
72
73 @app.route("/Telescope/DEC_Stop/", methods = ['POST'])
74 def Move_DEC_Stop():
75     with open('Config.cfg') as f:
76         config = json.load(f)
77         f.close()
78     config[ "MoveDEC"]="DEC0#"
79     config[ "ChangeDEC"]=True
80     with open('Config.cfg', 'w') as outfile:
81         json.dump(config, outfile)
82         outfile.close()
83     return str(1)
84
85 @app.route("/Telescope/Set_Pwm0/", methods = ['POST'])
86 def Set_Pwm0():
87     value = int(request.form.get("SliderValue"))
88     with open('Config.cfg') as f:
89         config = json.load(f)
90         f.close()
91     config[ "Consigne0"]="R0_"+'{:03d}'.format(value)+"#"
92     config[ "ConsigneChange"]=True
93     with open('Config.cfg', 'w') as outfile:
94         json.dump(config, outfile)
95         outfile.close()
96     return str(1)
97
98 @app.route("/Telescope/Set_Pwm1/", methods = ['POST'])
99 def Set_Pwm1():
100    value = int(request.form.get("SliderValue"))
101    with open('Config.cfg') as f:
102        config = json.load(f)
103        f.close()
104    config[ "Consigne1"]="R1_"+'{:03d}'.format(value)+"#"
105    config[ "ConsigneChange"]=True
106    with open('Config.cfg', 'w') as outfile:
107        json.dump(config, outfile)
108        outfile.close()
109    return str(1)
110
111 @app.route("/Telescope/Set_Pwm2/", methods = ['POST'])
112 def Set_Pwm2():
```

```
113     value = int(request.form.get("SliderValue"))
114     with open('Config.cfg') as f:
115         config = json.load(f)
116         f.close()
117     config["Consigne2"]="R2_"+'{:03d}'.format(value)+"#"
118     config["ConsigneChange"]=True
119     with open('Config.cfg', 'w') as outfile:
120         json.dump(config, outfile)
121         outfile.close()
122     return str(1)
123
124 @app.route("/Telescope/Set_Pwm3/", methods = ['POST'])
125 def Set_Pwm3():
126     value = int(request.form.get("SliderValue"))
127     with open('Config.cfg') as f:
128         config = json.load(f)
129         f.close()
130     config["Consigne3"]="R3_"+'{:03d}'.format(value)+"#"
131     config["ConsigneChange"]=True
132     with open('Config.cfg', 'w') as outfile:
133         json.dump(config, outfile)
134         outfile.close()
135     return str(1)
136
137 @app.route("/Telescope/SetInterval/", methods = ['POST'])
138 def SetInterval():
139     interval = request.form.get("Reponse")
140     with open('Config.cfg') as f:
141         config = json.load(f)
142         f.close()
143     print(interval)
144     config["Delai"]=interval
145     with open('Config.cfg', 'w') as outfile:
146         json.dump(config, outfile)
147         outfile.close()
148     return str(1)
149
150 @app.route("/Telescope/StartMesure/", methods = ['POST'])
151 def StartMesure():
152     start = request.form.get("Reponse")
153     with open('Config.cfg') as f:
154         config = json.load(f)
155         f.close()
156     #config["State"]=bool(start);
157     config["State"]=start in ['true', '1']
158     #print(start)
159     #print(bool(start))
160     #print(config)
161     with open('Config.cfg', 'w') as outfile:
162         json.dump(config, outfile)
163         outfile.close()
164     return str(1)
165
166 @app.route("/Telescope/CreateNewFile/", methods = ['POST'])
167 def CreateNewFile():
168     #start = request.form.get("Reponse")
```

```
169     with open('Config.cfg') as f:
170         config = json.load(f)
171         f.close()
172     config["CreateNewFile"] = True;
173     config["State"] = False;
174     with open('Config.cfg', 'w') as outfile:
175         json.dump(config, outfile)
176         outfile.close()
177     return render_template('Telescope.html')
178
179
180 @app.route('/Telescope/GetData', methods=['POST'])
181 def Telescope_Data():
182     test = request.form.get("Reponse")
183     print(test)
184     with open('Config.cfg') as f:
185         config = json.load(f)
186     colnames = ['Temperature0', 'Temperature1', 'Temperature2', 'Temperature3', ↪
187     'Temperature4', "Temperature_Rose", 'Humidite', "Temps"]
188     data = pandas.read_csv(config["FileName"], names=colnames, sep=';')
189     temperature0 = data.Temperature0.tolist()
190     temperature1 = data.Temperature1.tolist()
191     temperature2 = data.Temperature2.tolist()
192     temperature3 = data.Temperature3.tolist()
193     temperature4 = data.Temperature4.tolist()
194     temperature_rose = data.Temperature_Rose.tolist()
195     humidite = data.Humidite.tolist()
196     Temps = data.Temps.tolist()
197     myList = [temperature0[-1], temperature1[-1], temperature2[-1], temperature3[-1], ↪
198     temperature4[-1], temperature_rose[-1], humidite[-1], Temps[-1]]
199     return jsonify({'data': myList})
200
201 @app.route('/Telescope/GetAllData', methods=['POST'])
202 def Telescope_AllData():
203     test = request.form.get("Reponse")
204     print(test)
205     with open('Config.cfg') as f:
206         config = json.load(f)
207     colnames = ['Temperature0', 'Temperature1', 'Temperature2', 'Temperature3', ↪
208     'Temperature4', "Temperature_Rose", 'Humidite', "Temps"]
209     data = pandas.read_csv(config["FileName"], names=colnames, sep=';')
210     temperature0 = data.Temperature0.tolist()
211     temperature1 = data.Temperature1.tolist()
212     temperature2 = data.Temperature2.tolist()
213     temperature3 = data.Temperature3.tolist()
214     temperature4 = data.Temperature4.tolist()
215     temperature_rose = data.Temperature_Rose.tolist()
216     humidite = data.Humidite.tolist()
217     Temps = data.Temps.tolist()
218     myList = [temperature0, temperature1, temperature2, temperature3, temperature4, temperature_rose, ↪
219     humidite, Temps]
220     return jsonify({'data': myList})
221
222 @app.route('/Telescope/GetChart', methods=['POST'])
223 def Telescope_GetChart():
```

```
220     test = request.form.get("Reponse")
221     print(test)
222     with open('Config.cfg') as f:
223         config = json.load(f)
224     colnames = ['Temperature0', 'Temperature1', 'Temperature2', 'Temperature3', ↵
225                 'Temperature4', "Temperature_Rose", 'Humidite',"Temps"]
226     data = pandas.read_csv(config["FileName"],names=colnames,sep=';')
227     temperature0 = data.Temperature0.tolist()
228     temperature1 = data.Temperature1.tolist()
229     temperature2 = data.Temperature2.tolist()
230     temperature3 = data.Temperature3.tolist()
231     temperature4 = data.Temperature4.tolist()
232     temperature_rose = data.Temperature_Rose.tolist()
233     humidite=data.Humidite.tolist()
234     Temps=data.Temps.tolist()
235     #Trie les dernier element en fonction de chartSize
236     sizeMesure=len(Temps)
237     if(sizeMesure>=chartSize):
238         temperature0=temperature0[sizeMesure-chartSize:sizeMesure]
239         temperature1=temperature1[sizeMesure-chartSize:sizeMesure]
240         temperature2=temperature2[sizeMesure-chartSize:sizeMesure]
241         temperature3=temperature3[sizeMesure-chartSize:sizeMesure]
242         temperature4=temperature4[sizeMesure-chartSize:sizeMesure]
243         temperature_rose=temperature_rose[sizeMesure-chartSize:sizeMesure]
244         humidite=humidite[sizeMesure-chartSize:sizeMesure]
245         Temps=Temps[sizeMesure-chartSize:sizeMesure]
246     myList = [temperature0, ↵
247               temperature1,temperature2,temperature3,temperature4,temperature_rose, ↵
248               humidite,Temps]
249     return jsonify({'data': myList})
250
251 @app.route('/Telescope/')
252 def Telescope():
253     return render_template('Telescope.html')
254
255 @app.route('/')
256 def Telescope2():
257     return render_template('Telescope.html')
258
259 @app.route('/Mesure')
260 def Mesure():
261     with open('Config.cfg') as f:
262         config = json.load(f)
263     f = open(config["FileName"], "r")
264     text=f.read()
265     text = text.replace('\n', '')
266     return (text)
267
268 @app.route('/Config.cfg')
269 def GetConfig():
270     f = open("Config.cfg", "r")
271     return (f.read())
272
273 if __name__ == "__main__":
274     app.run(host='0.0.0.0', port=80, debug=True)
```

```
1 import datetime
2 import time
3 import RPi.GPIO as GPIO
4 import serial
5 import json
6 import time
7 import csv
8
9 serial = serial.Serial("/dev/ttyS0", 57600, timeout=1)
10
11
12 def GetTemperature_0():#temperature Ambiente
13     serial.write("GET_T0#")
14     temperature = serial.readline()
15     #print temperature
16     if temperature=="":
17         return "ERREUR"
18     else:
19         #temperature=temperature[:4]+". "+temperature[4:]
20         temperature=temperature.replace('\r',"")
21     return temperature
22
23 def GetTemperature_1():
24     serial.write("GET_T1#")
25     temperature = serial.readline()
26     if temperature=="":
27         return "ERREUR"
28     else:
29         #temperature=temperature[:4]+". "+temperature[4:]
30         temperature=temperature.replace('\r',"")
31     return temperature
32
33 def GetTemperature_2():
34     serial.write("GET_T2#")
35     temperature = serial.readline()
36     if temperature=="":
37         return "ERREUR"
38     else:
39         #temperature=temperature[:4]+". "+temperature[4:]
40         temperature=temperature.replace('\r',"")
41     return temperature
42
43 def GetTemperature_3():
44     serial.write("GET_T3#")
45     temperature = serial.readline()
46     if temperature=="":
47         return "ERREUR"
48     else:
49         #temperature=temperature[:4]+". "+temperature[4:]
50         temperature=temperature.replace('\r',"")
51     return temperature
52
53 def GetTemperature_4():
54     serial.write("GET_T4#")
55     temperature = serial.readline()
56     if temperature=="":
```

```
57         return "ERREUR"
58     else:
59         #temperature=temperature[:4]+". "+temperature[4:]
60         temperature=temperature.replace('\r',"")
61         return temperature
62
63 def GetTemperature_Rose():
64     serial.write("GET_TR#")
65     temperature = serial.readline()
66     if temperature=="":
67         return "ERREUR"
68     else:
69         #temperature=temperature[:4]+". "+temperature[4:]
70         temperature=temperature.replace('\r',"")
71         return temperature
72
73 def GetHumidity():
74     serial.write("GET_H#")
75     humidity = serial.readline()
76     if humidity=="":
77         return "ERREUR"
78     else:
79         humidity=humidity.replace('\r',"")
80         return humidity
81
82
83
84 cptDelai=0
85 while True:
86     try:
87         with open('Config.cfg') as f:
88             config = json.load(f)
89             f.close()
90     except:
91         with open('Config.cfg') as f:
92             #config = json.load(f)
93             print f.read()
94             #f.close()
95     if config["State"]==True: #Mode Mesure
96         cptDelai+=1
97         if cptDelai>(int(config["Delai"])*2):
98             cptDelai=0
99             now= int(round(time.time() * 1000))
100            temperature0=GetTemperature_0()
101            temperature1=GetTemperature_1()
102            temperature2=GetTemperature_2()
103            temperature3=GetTemperature_3()
104            temperature4=GetTemperature_4()
105            temperature_rose=GetTemperature_Rose()
106            humidite=GetHumidity()
107            temps=datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
108            with open(config["FileName"], mode='a') as MesureFile:#Attention ↵
109                I'ecriture du fichier prend du temps
110                MesureFile = csv.writer(MesureFile, delimiter=';',
111                                         quotechar='"', quoting=csv.QUOTE_MINIMAL)
110                MesureFile.writerow(
```

```
        ([temperature0,temperature1,temperature2,temperature3,temper ↪
         ature4,temperature_rose,humidite,tempo])
111     elapsed_time =int(round(time.time() * 1000))-now
112     #print elapsed_time
113     if elapsed_time >= 500:
114         delai_wait = 0.01#10ms
115     else:
116         delai_wait =500-elapsed_time#compense le temps
117     #print delai_wait
118     time.sleep(delai_wait/1000)#fais un delai de 500ms
119     else:
120         time.sleep(0.5)#fais un delai de 500ms
121 elif(config["CreateNewFile"]==True)&(config["State"]==False): #Cree un ↪
nouveau fichier de mesure
122     config["CreateNewFile"]=False
123     config["FileName"]=datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%% ↪
S")+".csv"
124     config["State"]=True
125     with open('Config.cfg', 'w') as outfile:
126         json.dump(config, outfile)
127         outfile.close()
128     with open(config["FileName"], mode='w') as MesureFile:
129         MesureFile = csv.writer(MesureFile, delimiter=';', quotechar='', ↪
quoting=csv.QUOTE_MINIMAL)
130         MesureFile.writerow(['Temperature0', 'Temperature1', ↪
'Temperature2', 'Temperature3', ↪
'Temperature4', "Temperature_Rose", 'Humidite', "Temps"])
131 elif (config["CreateNewFile"]==False)&(config["State"]==False):#Tout est ↪
arreter
132     time.sleep(0.5)
133     if (config["ConsigneChange"]==True):
134         print "Changement de consigne"
135         serial.write(config["Consigne0"].encode())
136         time.sleep(0.01)
137         serial.write(config["Consigne1"].encode())
138         time.sleep(0.01)
139         serial.write(config["Consigne2"].encode())
140         time.sleep(0.01)
141         serial.write(config["Consigne3"].encode())
142         config["ConsigneChange"]=False
143         with open('Config.cfg', 'w') as outfile:
144             json.dump(config, outfile)
145             outfile.close()
146     if (config["ChangeRA"]==True):
147         serial.write(config["MoveRA"].encode())
148         print "Move RA"
149         config["ChangeRA"]=False
150         with open('Config.cfg', 'w') as outfile:
151             json.dump(config, outfile)
152             outfile.close()
153     if (config["ChangeDEC"]==True):
154         serial.write(config["MoveDEC"].encode())
155         print "Move DEC"
156         config["ChangeDEC"]=False
157         with open('Config.cfg', 'w') as outfile:
158             json.dump(config, outfile)
```

159 outfile.close()
160

```
1  <!DOCTYPE html>
2  <html lang="fr">
3      <head>
4          <meta charset="utf-8"/>
5
6          <title>Controle de telescope</title>
7          <link href="/static/mon_style.css" rel="stylesheet" type="text/css" />
8          <script src="/static/jquery-3.2.1.min.js"></script>
9          <script src="/static/Chart.bundle.js"></script>
10         <script src="/static/utils.js"></script>
11
12         <script>
13             var delai=10;
14             var dumb;
15             window.onload = function() {
16                 var lineChartData =
17                 {
18                     title : "Capteur de temperature",
19                     type: 'line',
20                     data:
21                     {
22                         labels: [],
23                         datasets:
24                         [
25                             {
26                                 label: 'Temperature_Ambiante',
27                                 backgroundColor: window.chartColors.red,
28                                 borderColor: window.chartColors.red,
29                                 data: [],
30                                 fill: false,
31                             },
32                             {
33                                 label: 'Capteur 0 [°C]',
34                                 fill: false,
35                                 backgroundColor: window.chartColors.purple,
36                                 borderColor: window.chartColors.purple,
37                                 data: [],
38                             },
39                             {
40                                 label: 'Capteur 1 [°C]',
41                                 fill: false,
42                                 backgroundColor: window.chartColors.green,
43                                 borderColor: window.chartColors.green,
44                                 data: [],
45                             },
46                             {
47                                 label: 'Capteur 2 [°C]',
48                                 fill: false,
49                                 backgroundColor: window.chartColors.yellow,
50                                 borderColor: window.chartColors.yellow,
51                                 data: [],
52                             },
53                             ,
54                             {
55                                 label: 'Capteur 3 [°C]',
56                                 fill: false,
```

```
57                     backgroundColor: window.chartColors.orange,
58                     borderColor: window.chartColors.orange,
59                     data: [],
60                 }
61             ,
62         {
63             label: 'Temperature Rose [°C]',
64             fill: false,
65             backgroundColor: window.chartColors.grey,
66             borderColor: window.chartColors.grey,
67             data: [],
68         }
69     ,
70     {
71         label: 'Humidité [%]',
72         fill: false,
73         backgroundColor: window.chartColors.blue,
74         borderColor: window.chartColors.blue,
75         data: [],
76     }
77 ]
78 },
79 options:
80 {
81     responsive: true,
82     title:
83     {
84         display: true,
85         text: 'Capteur Temperature'
86     },
87     tooltips:
88     {
89         mode: 'index',
90         intersect: false,
91     },
92     hover:
93     {
94         mode: 'nearest',
95         intersect: true
96     },
97     scales:
98     {
99         xAxes:
100        [
101            {
102                //display: true,
103                //scaleLabel:
104                //{
105                    // display: true,
106                    // labelString: 'Temps'
107                //}
108                //type: 'time',
109                //time:
110                //{
111                    //displayFormats:
112                //}
113            }
114        ]
115    }
116 }
```

```
113                     // quarter: 'MMM YYYY'  
114                     //}  
115                 //}  
116             },  
117             yAxes:  
118             [{  
119                 display: true,  
120                 scaleLabel:  
121                 {  
122                     display: true,  
123                     labelString: 'Temperature et humidité'  
124                 }  
125             }]  
126         }  
127     }  
128 };  
129 dumb=lineChartData;  
130 var ctx = document.getElementById('canvas').getContext('2d');  
131 window.myLine = new Chart(ctx, lineChartData);  
132  
133 $('button#StartMesure').click(function(){  
134     var start=0;  
135     if(document.getElementById  
("StartMesure").innerHTML=="Arreter les mesures")  
136     {  
137         document.getElementById  
("StartMesure").innerHTML="Demarrer les mesures";  
138         start=0;  
139     }  
140     else  
141     {  
142         document.getElementById  
("StartMesure").innerHTML="Arreter les mesures";  
143         start=1;  
144     }  
145     $.ajax  
({  
146  
147         url: "/Telescope/StartMesure/",  
148         type: "POST", //send it through get method  
149         data: {"Reponse": start},  
150         success: function(response)  
151         {  
152  
153         }  
154     });  
155 });  
156});  
157  
158 $('button#CreateNewFile').click(function(){  
159     document.getElementById("StartMesure").innerHTML="Arreter les  
mesures";  
160     alert("La page va etre recharger");  
161     //window.location.reload(1);  
162     $.ajax  
({  
163  
164
```

```
165                     url: "/Telescope/CreateNewFile/",
166                     type: "POST", //send it through get method
167                     data: {"Reponse": "OK"},
168                     success: function(response)
169                     {
170
171                     }
172                 });
173             });
174
175             $('button#RA_Pos').click(function(){
176                 $.ajax
177                 ({
178
179                     url: "/Telescope/RA_Pos/",
180                     type: "POST", //send it through get method
181                     data: {"Reponse": "OK"},
182                     success: function(response)
183                     {
184
185                     }
186                 });
187             });
188
189             $('button#RA_Neg').click(function(){
190                 $.ajax
191                 ({
192
193                     url: "/Telescope/RA_Neg/",
194                     type: "POST", //send it through get method
195                     data: {"Reponse": "OK"},
196                     success: function(response)
197                     {
198
199                     }
200                 });
201             });
202
203             $('button#RA_Stop').click(function(){
204                 $.ajax
205                 ({
206
207                     url: "/Telescope/RA_Stop/",
208                     type: "POST", //send it through get method
209                     data: {"Reponse": "OK"},
210                     success: function(response)
211                     {
212
213                     }
214                 });
215             });
216
217             $('button#DEC_Pos').click(function(){
218                 $.ajax
219                 ({
220
```



```
274         $('input#sliderPwm1').click(function()
275     {
276         $.ajax
277     (
278         url: "/Telescope/Set_Pwm1/",
279         type: "POST", //send it through get method
280         data:{"SliderValue": document.getElementById
281             ("sliderPwm1").value},
282         success: function(response)
283         {
284             valueConsigne1 = Math.round(document.getElementById
285                 ("sliderPwm1").value);
286             document.getElementById("valuePwm1").innerHTML =
287             valueConsigne1;
288         }
289     });
290 });
291
292         $('input#sliderPwm2').click(function()
293     {
294         $.ajax
295     (
296         url: "/Telescope/Set_Pwm2/",
297         type: "POST", //send it through get method
298         data:{"SliderValue": document.getElementById
299             ("sliderPwm2").value},
300         success: function(response)
301         {
302             valueConsigne2 = Math.round(document.getElementById
303                 ("sliderPwm2").value);
304             document.getElementById("valuePwm2").innerHTML =
305             valueConsigne2;
306         }
307     });
308 });
309
310         $('input#sliderPwm3').click(function()
311     {
312         $.ajax
313     (
314         url: "/Telescope/Set_Pwm3/",
315         type: "POST", //send it through get method
316         data:{"SliderValue": document.getElementById
317             ("sliderPwm3").value},
318         success: function(response)
319         {
320             valueConsigne3 = Math.round(document.getElementById
321                 ("sliderPwm3").value);
322             document.getElementById("valuePwm3").innerHTML =
323             valueConsigne3;
324         }
325     });
326 });
327
328         $('input#nMesure').on('change', function () {
329             if(document.getElementById("nMesure").value>=3)
```

```
321         {
322             }
323         }
324     else
325     {
326         document.getElementById("nMesure").value = 3;
327     }
328     delai=document.getElementById("nMesure").value*1000;
329     //console.log(delai)
330     $.ajax
331     ({
332         url: "/Telescope/SetInterval/",
333         type: "POST", //send it through get method
334         data:{"Reponse": document.getElementById("nMesure").value},
335         success: function(response)
336         {
337
338         }
339     });
340
341     //alert("changed");
342 });
343 //Code a lancer a l'ouverture de la page ici :
344 //ouverture du fichier json Config.cfg
345 //var delai=0;
346 var rawFile = new XMLHttpRequest();
347 rawFile.open("GET", "/Config.cfg", false);
348 rawFile.onreadystatechange = function ()
349 {
350     if(rawFile.readyState === 4)
351     {
352         if(rawFile.status === 200 || rawFile.status == 0)
353         {
354             var allText = rawFile.responseText;
355             var obj = JSON.parse(allText);
356             if(obj.State==true)
357             {
358                 document.getElementById("StartMesure").innerHTML =
359                 "Arreter les mesures";
360                 delai=obj.Delai*1000;
361                 document.getElementById("nMesure").value =
362                 obj.Delai;
363                 //setInterval(TimerMesure,obj.Delai*1000);
364                 window.setTimeout(TimerMesure, delai);
365                 //console.log(delai)
366                 //Charge les donne des mesure en cours
367                 var ctx = document.getElementById('canvas').getContext('2d');
368                 window.myLine = new Chart(ctx, lineChartData);
369                 $.ajax
370                 ({
371                     url: "/Telescope/GetChart",
372                     type: "POST", //send it through get method
373                     data: {"Reponse": "ok"},
374                     success: function(response)
```

```
373                     {
374
375                         for (i = 1; i < response.data[0].length; i++)
376                             ++
377
378                             var x = 0;
379                             //window.myLine.data.labels.push
380                             (window.myLine.data.labels.length);
381                             window.myLine.data.labels.push
382                             (response.data[response.data.length-1][i]);
383                             window.myLine.data.datasets.forEach
384                             (function(dataset)
385
386                             {
387
388                                 dataset.data.push(parseFloat
389                                 (response.data[x][i]));
390                                 x++;
391                                 });
392                                 window.myLine.update();
393
394                             }
395                         }
396                     );
397                     rawFile.send(null);
398
399                 };
400
401             function readTextFile(file)
402             {
403                 var rawFile = new XMLHttpRequest();
404                 rawFile.open("GET", file, false);
405                 rawFile.onreadystatechange = function ()
406                 {
407                     if(rawFile.readyState === 4)
408                     {
409                         if(rawFile.status === 200 || rawFile.status == 0)
410                         {
411                             var allText = rawFile.responseText;
412                             //alert(allText);
413                             return allText;
414                         }
415                     }
416                 }
417
418                 rawFile.send(null);
419             }
420
421             function TimerMesure() {
422                 $.ajax
```

```
423         ({
424             url: "/Telescope/GetData",
425             type: "POST", //send it through get method
426             data: {"Reponse": "ok"},
427             success: function(response)
428             {
429                 var x = 0;
430                 console.log(window.myLine.data.labels.length);
431                 if(window.myLine.data.labels.length>=49)
432                 {
433                     window.myLine.data.labels.shift();
434                 }
435                 window.myLine.data.labels.push(response.data
436 [response.data.length-1]);
437                 window.myLine.data.datasets.forEach(function(items)
438                 {
439                     if(window.myLine.data.labels.length>=49)
440                     {
441                         items.data.shift();
442                     }
443                     items.data.push(parseFloat(response.data[x]));
444                     x++;
445                 });
446                 window.myLine.update();
447             }
448         });
449         //console.log(delai)
450         window.setTimeout(TimerMesure, delai);
451     }
452
453
454     </script>
455 </head>
456 <body>
457     <ul>
458         <li><a class = "active" href="">Telescope</a></li>
459     </ul>
460     <h1>Telescope</h1>
461     <canvas id="canvas" width="300" height="100"></canvas>
462     <button id="CreateNewFile" >Demarrer une nouvelle mesure</button>
463     <button id="StartMesure" >Demarrer les mesures</button>
464     <button id="RA_Pos" >RA+</button>
465     <button id="RA_Stop" >Stop RA</button>
466     <button id="RA_Neg" >RA-</button>
467     <button id="DEC_Pos" >DEC+</button>
468     <button id="DEC_Stop" >Stop Dec</button>
469     <button id="DEC_Neg" >DEC-</button><br>
470     <br>
471     une mesure toutes les :
472     <input type="number" name="nMesure" id="nMesure" min="1" max="300"
473         width="50"><br>
474     <a href="/Mesure">Telecharger toutes les mesure</a><br>
475     <h2>Pwm 0 :</h2>
476     <input type="range" min="0" max="255" value = "255" class="slider"
477         id="sliderPwm0"></input>
```

```
476      <span id="valuePwm0">Pwm0 : </span>
477
478      <h2>Pwm 1 :</h2>
479      <input type="range" min="0" max="255" value = "255" class="slider"      ↵
480          id="sliderPwm1"></input>
481      <span id="valuePwm1">Pwm1 : </span>
482
483      <h2>Pwm 2 :</h2>
484      <input type="range" min="0" max="255" value = "255" class="slider"      ↵
485          id="sliderPwm2"></input>
486      <span id="valuePwm2">Pwm2 : </span>
487
488      <h2>Pwm 3 :</h2>
489      <input type="range" min="0" max="255" value = "255" class="slider"      ↵
490          id="sliderPwm3"></input>
491      <span id="valuePwm3">Pwm3 : </span>
492
493  </body>
494 </html>
495
```

```

1  /*=====
2   Master_Telescope - Tanguy Dietrich
3  =====
4  Descriptif:
5  Gestion de moteur d'une monture NEQ5.
6  Le moteur R.A a une vitesse constante correspondant a la vitesse sideral.
7  Le moteur DEC est activer lorsque la pin p2.2 ou P2.3(ST4) sont a 0
8  Le tout est commander avec un port ST4
9  =====*/
10
11 #include <reg51f380.h>      // registres 51f38C
12 #include "Vitesse_Moteur.h"
13 #include "Delay48M.h"
14 #include "SMBus0.h"
15 #include "string.h"
16
17 #define CONFIG_PAGE 0x0F
18 #define LEGACY_PAGE 0x00
19 #define BAUD_UART0 247 //247=57600
20
21 //PCB
22 sbit STEP_RA = P1^3;
23 sbit DIR_RA = P1^2;
24 sbit NSLEEP_RA = P1^0;
25 sbit NFAULT_RA = P1^1;
26
27 sbit STEP_DEC = P1^7;
28 sbit DIR_DEC = P1^6;
29 sbit NSLEEP_DEC = P1^4;
30 sbit NFAULT_DEC = P1^5;
31
32 sbit ST4_RA_NEG = P2^0;
33 sbit ST4_RA_POS = P2^1;
34 sbit ST4_DEC_NEG = P2^2;
35 sbit ST4_DEC_POS = P2^3;
36
37 sbit SDA = P0^0;
38 sbit SCL = P0^1;
39
40
41 // ===== FONCTIONS PROTOTYPES=====
42 void ClockInit ();           // init. clock systme
43 void PortInit ();           // init. config des ports
44 void TimerInit();
45 void Init_int();
46 void SMBus_Init (void);
47 void Timer3_Init (void);
48 void Timer2_Init (void);
49 void SMB_Write (void);
50 void SMB_Read (void);
51 void UART_Init();
52 void decodeUartRaspberryPi();
53 void setMotorRA(unsigned char vitesse,bit direction);
54 void setMotorDEC(unsigned char vitesse,bit direction);
55 void stopMotorDec();
56 void configMCP23008(unsigned char input);
57 unsigned char read_i2c_port();
58 void reset_i2c();
59 // ===== MAIN =====
60
61 //SMBus0
62 unsigned char gSMBNumBytesToWR = 2; // Number of bytes to write
63                                     // Master -> Slave
64 unsigned char gSMBNumBytesToRD = 3; // Number of bytes to read
65                                     // Master <- Slave
66
67 // Global holder for SMBus data
68 // All receive data is written here
69 unsigned char gSMBDataIN[NUM_BYTES_MAX_RD];
70
71 // Global holder for SMBus data.
72 // All transmit data is read from here
73 unsigned char gSMBDataOUT[NUM_BYTES_MAX_WR];
74
75 unsigned char gTarget;          // gTarget SMBus slave address
76
77 bit SMB_BUSY;                 // Software flag to indicate when the

```

```

78                                     // SMB_Read() or SMB_Write() functions
79                                     // have claimed the SMBus
80
81     bit SMB_RW;                      // Software flag to indicate the
82                                     // direction of the current transfer
83
84     unsigned long NUM_ERRORS;         // Counter for the number of errors.
85
86
87
88
89 //Gestion Moteur
90 unsigned char gVitesseRAH=VITESSE_RA_SIDERAL_HIGH;
91 unsigned char gVitesseRAL=VITESSE_RA_SIDERAL_LOW;
92 unsigned int gVitesseDEC=VITESSE_DEC_MAX;
93 unsigned char gVitesseDeplacement=VITESSE_MAX;
94
95 //Uart
96 xdata unsigned char gUart0Tx[10]="";
97 xdata unsigned char gUart0Rx[10];
98 unsigned char gUart0NbrByteTx=0;
99 unsigned char gUart0NbrByteRx=0;
100 bit gUart0FlagReceive;
101
102 xdata unsigned char gUart1Tx[10]="";
103 xdata unsigned char gUart1Rx[10];
104 unsigned char gUart1NbrByteTx=0;
105 unsigned char gUart1NbrByteRx=0;
106 bit gUart1FlagReceive;
107 bit gDirRa=0;
108 bit gDirDec=0;
109 unsigned char gAstrSuivi=0;
110 bit gFlagMovingRA=0;
111
112 void main ()
113 {
114     unsigned char portExtender;
115     unsigned char memoAstreSuivi=0;
116     bit waitEndMove=0;
117     PCA0MD &= ~0x40;      // WDTE = 0 (disable watchdog timer)
118     ClockInit ();        // init. clock system
119     PortInit ();         // init. config des ports
120     SDA=1;
121     reset_i2c();
122     Init_int();
123     Timer3_Init();       // Configure Timer3 for use with SMBus
124                                     // low timeout detect
125     Timer2_Init();
126
127     SMBus_Init ();        // Configure and enable SMBus
128     UART_Init();
129     TimerInit();
130     EA=1; //Autorise toutes les interruption
131     EIE1 |= 0x81;
132     EIE2 |= 0x12;
133     IE |= 0x97;
134     TR0 = 1; //lance le timer 0
135     TR1=1;
136     NSLEEP_RA=1; //Active le chip drv8825
137     NSLEEP_DEC=0;
138     DIR_RA=0;
139     DIR_DEC=0;
140
141     SDA=1;
142     Delay_1ms (20);
143     configMCP23008(0xFF);
144     portExtender=read_i2c_port();
145     gAstrSuivi=(portExtender>>4);
146     //memoAstreSuivi=gAstrSuivi;
147     while (1)
148     {
149         memoAstreSuivi=gAstrSuivi;
150         gVitesseDeplacement=(P2>>4)+VITESSE_X05;
151         portExtender=read_i2c_port();
152         gAstrSuivi=(portExtender>>4);
153         gDirRa=(portExtender&0x04)>>2;
154         gDirDec=(portExtender&0x08)>>3;

```

```

155     if( (memoAstreSuivi!=gAstrSuivi) || (waitEndMove==1) )
156     {
157         waitEndMove=1;
158         if(gFlagMovingRA==0)
159         {
160             setMotorRA(gAstrSuivi,gDirRa);
161             waitEndMove=0;
162         }
163     }
164 }
165
166 if( (NFAULT_RA==0) && (NSLEEP_RA==1) )//probleme sur le DRV8825 de l'axe R.A
167 {
168     NSLEEP_RA=0;
169     Delay_1ms (10);
170     NSLEEP_RA=1;
171     Delay_1ms (10); //1.7ms necessaire au rallumage
172 }
173
174 if( (NFAULT_DEC==0) && (NSLEEP_DEC==1) )//probleme sur le DRV8825 de l'axe DEC
175 {
176     NSLEEP_DEC=0;//Eteint
177     Delay_1ms (10);
178     NSLEEP_DEC=1;//Allume
179     Delay_1ms (10);
180 }
181
182 if(gUart0FlagReceive)
183 {
184     gUart0FlagReceive=0;
185     while(gUart0NbrByteTx!=0);
186     decodeUartRaspberryPi();
187 }
188
189 if(gUart1FlagReceive)
190 {
191     while(gUart0NbrByteTx!=0);
192     gUart1FlagReceive=0;
193     strcpy(gUart0Tx,gUart1Rx);
194     TI0=1;
195 }
196
197 } // End while (1)
198 } // main =====
199
200 //INTERRUPTION
201 /*-----*/
202 interruption0()
203 -----
204 Descriptif: Fonction d'interruption INT0 vecteur 0
205         Pin :P0.6 - Descendant
206 Entree   : --
207 Sortie   : --
208 */-----*/
209 void interruption_ST4_RA() interrupt 0
210 {
211     IT01CF=IT01CF^0x08;
212     if((!ST4_RA_POS)&&(!ST4_RA_NEG))
213     {
214         setMotorRA(gAstrSuivi,gDirRa);
215         gFlagMovingRA=0;
216     }
217     else if((ST4_RA_POS)&&(ST4_RA_NEG))
218     {
219         setMotorRA(gAstrSuivi,gDirRa);
220         gFlagMovingRA=0;
221     }
222     else if((ST4_RA_POS)&&(!ST4_RA_NEG))
223     {
224         setMotorRA(gVitesseDeplacement,!gDirRa);
225         gFlagMovingRA=1;
226     }
227     else if((!ST4_RA_POS)&&(ST4_RA_NEG))
228     {
229         setMotorRA(gVitesseDeplacement,gDirRa);
230         gFlagMovingRA=1;
231     }

```

```

232     }
233     /*-----*/
234     interruption1()
235     -----
236     Descriptif: Fonction d'interruption 1 vecteur 2
237         Pin :P0.7 - Descendant
238     Entree    : --
239     Sortie    : --
240     ----- */
241 void interruption_ST4_DEC() interrupt 2
242 {
243     IT01CF=IT01CF^0x80;
244     if((!ST4_DEC_POS)&&(!ST4_DEC_NEG))//00
245     {
246         stopMotorDec();
247     }
248     else if((ST4_DEC_POS)&&(ST4_DEC_NEG))//11
249     {
250         stopMotorDec();
251     }
252     else if((ST4_DEC_POS)&&(!ST4_DEC_NEG))//10
253     {
254         setMotorDEC(gVitesseDeplacement,gDirDec);
255     }
256     else if((!ST4_DEC_POS)&&(ST4_DEC_NEG))//01
257     {
258         setMotorDEC(gVitesseDeplacement,!gDirDec);
259     }
260 }
261
262 /*-----*/
263 timer0()
264 -----
265 Descriptif: Fonction d'interruption Timer0 vecteur 1
266     Temporisation de 51,94milli
267     Mode : 16bit
268     Entree    : --
269     Sortie    : --
270     ----- */
271 void timer0() interrupt 1
272 {
273     TR0=0;
274     TH0=gVitesseRAH ;//Charge la valeur dans le registre MSB du timer 0
275     TL0=gVitesseRAL;//Charge la valeur dans le registre LSB du timer 0
276     TR0=1;
277     STEP_RA=!STEP_RA;
278 }
279
280 ///*-----*/
281 //timer1()
282 //-----
283 //Descriptif: Fonction d'interruption Timer1 vecteur 3
284 //        Mode : 8 bit 9600baud
285 //Entree    : --
286 //Sortie    : --
287 //----- */
288 //void timer1() interrupt 3
289 //{
290 ////
291 //}
292
293 /*-----*/
294 timer4()
295 -----
296 Descriptif: Fonction d'interruption Timer4 vecteur 19
297     Mode : 16bit autoreload
298     Entree    : --
299     Sortie    : --
300     ----- */
301 void timer4() interrupt 19
302 {
303     static unsigned int cpt=0;
304     SFRPAGE = CONFIG_PAGE;
305     TMR4CN &=~0xC0;//Clear pending flag
306     SFRPAGE = LEGACY_PAGE;
307     cpt++;
308     if(cpt>=gVitesseDEC)

```

```

309     {
310         STEP_DEC=!STEP_DEC;
311         cpt=0;
312     }
313 }
314 /*-----*/
315 uart0()
316 -----
317 Descriptif: Fonction d'interruption de l'uart0 vecteur 4
318 Entrée : --
319 Sortie : --
320 */
321 void uart0() interrupt 4
322 {
323     if(TI0)
324     {
325         TI0=0;
326         if(gUart0Tx[gUart0NbrByteTx]!=0)
327         {
328             SBUF0=gUart0Tx[gUart0NbrByteTx];
329             gUart0NbrByteTx++;
330         }
331     else
332     {
333         gUart0NbrByteTx=0;
334     }
335 }
336
337 if(RI0)
338 {
339     RI0=0;
340     gUart0Rx[gUart0NbrByteRx]=SBUF0;
341     if(gUart0Rx[gUart0NbrByteRx]=='#')
342     {
343         gUart0FlagReceive=1;
344         gUart0Rx[gUart0NbrByteRx+1]=0;
345         gUart0NbrByteRx=0;
346     }
347     else
348     {
349         gUart0NbrByteRx=(gUart0NbrByteRx+1)%20;
350     }
351 }
352 }
353
354 /*-----*/
355 uart1()
356 -----
357 Descriptif: Fonction d'interruption de l'uart1 vecteur 16
358 Entrée : --
359 Sortie : --
360 */
361 void uart1() interrupt 16
362 {
363     if(SCON1&0x02)//TI1
364     {
365         SCON1=SCON1&~0x02;
366         if(gUart1Tx[gUart1NbrByteTx]!=0)
367         {
368             SBUF1=gUart1Tx[gUart1NbrByteTx];
369             gUart1NbrByteTx++;
370         }
371     else
372     {
373         gUart1NbrByteTx=0;
374     }
375 }
376
377 if(SCON1&0x01)//RI1
378 {
379     SCON1=SCON1&~0x01;
380     gUart1Rx[gUart1NbrByteRx]=SBUF1;
381     if(gUart1Rx[gUart1NbrByteRx]=='\n')
382     {
383         gUart1FlagReceive=1;
384         gUart1Rx[gUart1NbrByteRx+1]=0;
385         gUart1NbrByteRx=0;

```

```

386         }
387     else
388     {
389         gUart1NbrByteRx=(gUart1NbrByteRx+1)%20;
390     }
391 }
392 }
393
394 //FONCTION
395
396 /*-----*
397 reset_i2c()
398 -----
399 Descriptif: Verifie si le bus I2C est bloquer, et effectue un reset du bus
400 Entree   : --
401 Sortie   : --
402 -----*/
403 void reset_i2c()
404 {
405     unsigned char i=0;
406     if(!SDA)
407     {
408         //Desactive tout l'I2C
409         POSKIP    = 0x03;//skip les pin SCL SDA
410         XBR0      = 0x01;//desactive l'I2C
411         //force le slave a sortir des donnee
412         while(!SDA)
413         {
414             // Provide clock pulses to allow the slave to advance out
415             // of its current state. This will allow it to release SDA.
416             SCL = 0;                                // Drive the clock low
417             for(i = 0; i < 100; i++);               // Hold the clock low
418             SCL = 1;                                // Release the clock
419             for(i = 0; i < 100; i++);               // Hold the clock low
420         }
421         POSKIP    = 0x00;//Enleve les skip de pin SDA et SCL
422         XBR0      = 0x05;//reactive l'i2c
423     }
424 }
425
426 /*-----*
427 configMCP23008()
428 -----
429 Descriptif:
430 Entree   : unsigned char input (0 ... 255) - choix de pins a mettre en sortie
431 Sortie   : --
432 -----*/
433 void configMCP23008(unsigned char input)
434 {
435     gTarget = MCP23008_ADDR;           // gTarget the Slave for next SMBus transfer
436     gSMBDataOUT[0] = MCP23008_IODIR;
437     gSMBDataOUT[1] = input;           // Set as INPUT
438     gSMBNumBytesToWR = 2;
439     SMB_Write(); // Initiate SMBus write
440 }
441
442 /*-----*
443 read_i2c_port()
444 -----
445 Descriptif: Lis le port du MCP23008
446 Entree   : --
447 Sortie   : unsigend char (0 ... 255)
448 -----*/
449 unsigned char read_i2c_port()
450 {
451     while(SMB_BUSY);
452     gTarget = MCP23008_ADDR;
453     gSMBDataOUT[0] = MCP23008_GPIO;
454     gSMBNumBytesToRD = 1;
455     SMB_Write(); // Initiate SMBus write
456     while(SMB_BUSY);
457     gTarget = MCP23008_ADDR;
458     gSMBNumBytesToRD = 1;
459     SMB_Read(); // Initiate SMBus write
460     return gSMBDataIN[0];
461 }
462

```

```

463 /*-----*
464 setMotorRA()
465 -----
466 Descriptif: Applique une vitesse predefinit au moteur et une direction
467 Entrée   :
468     - unsigned char vitesse (0 ... 10)
469         - bit direction      (0 ... 1)
470 Sortie    : --
471 -----*/
472 void setMotorRA(unsigned char vitesse,bit direction)
473 {
474     unsigned int i=0;
475     TR0=0;
476     STEP_RA=0;
477     //for(i=0;i<500;i++); //voire figure 1
478     DIR_RA=direction;
479     //for(i=0;i<;i++); //voire figure 1
480     switch(vitesse)
481     {
482         case VITESSE_RA_SIDERAL://suivi sideral
483             gVitesseRAH=VITESSE_RA_SIDERAL_HIGH;
484             gVitesseRAL=VITESSE_RA_SIDERAL_LOW;
485             break;
486         case VITESSE_RA_LUNE://suivi Lune
487             gVitesseRAH=VITESSE_RA_LUNE_HIGH;
488             gVitesseRAL=VITESSE_RA_LUNE_LOW;
489             break;
490         case VITESSE_RASOLEIL://suivi Soleil-Planete generale
491             gVitesseRAH=VITESSE_RA_SOLEIL_HIGH; //
492             gVitesseRAL=VITESSE_RA_SOLEIL_LOW;
493             break;
494         case VITESSE_RA_SATURNE://suivi Soleil-Planete generale
495             gVitesseRAH=VITESSE_RA_SATURNE_HIGH; //
496             gVitesseRAL=VITESSE_RA_SATURNE_LOW;
497             break;
498         case VITESSE_RA_JUPITER://suivi Soleil-Planete generale
499             gVitesseRAH=VITESSE_RA_JUPITER_HIGH; //
500             gVitesseRAL=VITESSE_RA_JUPITER_LOW;
501             break;
502         case VITESSE_RA_ISS://suivi ISS
503             gVitesseRAH=VITESSE_RA_ISS_HIGH;
504             gVitesseRAL=VITESSE_RA_ISS_LOW;
505             break;
506         case VITESSE_X05://x0,5
507             DIR_RA=0;
508             if(direction==0)
509             {
510                 gVitesseRAH=VITESSE_RA_X1_5_HIGH;
511                 gVitesseRAL=VITESSE_RA_X1_5_HIGH;
512             }
513             else
514             {
515                 gVitesseRAH=VITESSE_RA_X05_HIGH;
516                 gVitesseRAL=VITESSE_RA_X05_LOW;
517             }
518             break;
519         case VITESSE_X2://x2
520             gVitesseRAH=VITESSE_RA_X2_HIGH;
521             gVitesseRAL=VITESSE_RA_X2_LOW;
522             break;
523         case VITESSE_X8://x8
524             gVitesseRAH=VITESSE_RA_X8_HIGH;
525             gVitesseRAL=VITESSE_RA_X8_LOW;
526             break;
527         case VITESSE_X16://x16
528             gVitesseRAH=VITESSE_RA_X16_HIGH;
529             gVitesseRAL=VITESSE_RA_X16_LOW;
530             break;
531         case VITESSE_MAX://max
532             gVitesseRAH=VITESSE_RA_MAX_HIGH;
533             gVitesseRAL=VITESSE_RA_MAX_LOW;
534             break;
535         default://suivi sideral
536             gVitesseRAH=VITESSE_RA_SIDERAL_HIGH;
537             gVitesseRAL=VITESSE_RA_SIDERAL_LOW;
538             break;
539     }
}

```

```

540     TH0=gVitesseRAH ;//Charge la valeur dans le registre MSB du timer 0
541     TL0=gVitesseRAL; //Charge la valeur dans le registre LSB du timer 0
542     TR0=1;
543 }
544
545 /*-----*/
546 setMotorDEC()
547 -----
548 Descriptif: Applique une vitesse predefinit au moteur et une direction
549 Entrée   :
550         - unsigned char vitesse (6 ... 10)
551         - bit direction      (0 ... 1)
552 Sortie    : --
553 */
554 void setMotorDEC(unsigned char vitesse,bit direction)
555 {
556     unsigned int i=0;
557     SFRPAGE = CONFIG_PAGE;
558     TMR4CN&=~0x04;
559     SFRPAGE = LEGACY_PAGE;
560     //for(i=0;i<50000;i++); //voire figure 1
561     NSLEEP_DEC=1; //Attention il faut 1.7ms au chip pour se rallumer
562     STEP_DEC=0;
563     DIR_DEC=direction;
564     //for(i=0;i<50000;i++); //voire figure 1
565     switch(vitesse)
566     {
567         case VITESSE_X05://x0,5
568             gVitesseDEC=VITESSE_DEC_X05;
569             SFRPAGE = CONFIG_PAGE;
570             TMR4CN|=0x04;
571             SFRPAGE = LEGACY_PAGE;
572             break;
573         case VITESSE_X2://x2
574             gVitesseDEC=VITESSE_DEC_X2;
575             SFRPAGE = CONFIG_PAGE;
576             TMR4CN|=0x04;
577             SFRPAGE = LEGACY_PAGE;
578             break;
579         case VITESSE_X8://x8
580             gVitesseDEC=VITESSE_DEC_X8;
581             SFRPAGE = CONFIG_PAGE;
582             TMR4CN|=0x04;
583             SFRPAGE = LEGACY_PAGE;
584             break;
585         case VITESSE_X16://x16
586             gVitesseDEC=VITESSE_DEC_X16;
587             SFRPAGE = CONFIG_PAGE;
588             TMR4CN|=0x04;
589             SFRPAGE = LEGACY_PAGE;
590             break;
591         case VITESSE_MAX://max
592             gVitesseDEC=VITESSE_DEC_MAX;
593             SFRPAGE = CONFIG_PAGE;
594             TMR4CN|=0x04;
595             SFRPAGE = LEGACY_PAGE;
596             break;
597         default://suivi sideral
598             stopMotorDec();
599             break;
600     }
601 }
602
603 /*
604 stopMotorDec()
605 -----
606 Descriptif: Stop le moteur de declinaison et met le drv8825 en sommeil
607 Entrée   : --
608 Sortie    : --
609 */
610 void stopMotorDec()
611 {
612     SFRPAGE = CONFIG_PAGE;
613     TMR4CN&=~0x04;
614     SFRPAGE = LEGACY_PAGE;
615     NSLEEP_DEC=0;
616 }

```

```

617
618 /*-----*
619 decodeUartRaspberryPi()
620 -----
621 Descriptif: Decode les message envoyé par la raspberry PI
622 Entrée : -- (Utilisation de variable globale)
623 Sortie : --
624 -----*/
625 void decodeUartRaspberryPi()
626 {
627     if((gUart0Rx[0]=='R') && (gUart0Rx[1]=='A'))
628     {
629         if((gUart0Rx[2]=='+'))
630         {
631             setMotorRA(gVitesseDeplacement,gDirRa);
632         }
633         else if (gUart0Rx[2]=='-')
634         {
635             setMotorRA(gVitesseDeplacement,!gDirRa);
636         }
637         else//=='0'
638         {
639             setMotorRA(gAstrSuivi,gDirRa);
640         }
641     }
642     else if((gUart0Rx[0]=='D') && (gUart0Rx[1]=='E') && (gUart0Rx[2]=='C'))
643     {
644         if((gUart0Rx[3]=='+'))
645         {
646             setMotorDEC(gVitesseDeplacement,gDirDec);
647         }
648         else if (gUart0Rx[3]=='-')
649         {
650             setMotorDEC(gVitesseDeplacement,!gDirDec);
651         }
652         else//=='0'
653         {
654             stopMotorDec();
655         }
656     }
657     else//Ne reconnaît pas la commande, fais suivre au slave
658     {
659         //gUart1Tx[0]
660         strcpy(gUart1Tx,gUart0Rx);
661         SCON1|=0x02;
662     }
663 }
664
665 /*-----*
666     UART_Init ()
667 -----
668     Descriptif: Configure l'UART par ces valeur par défaut
669                 (il ne fait rien)
670     Entrée : --
671     Sortie : --
672 -----*/
673 void UART_Init()
674 {
675     SCON0      = 0x10;
676     //SBRLL1    = 0x3C;
677     //SBRLH1    = 0xF6;
678     //SCON1     = 0x10;
679     //SBCON1    = 0x43;
680
681     //57600 Uart1
682     SBRLL1    = 0x5F;
683     SBRLH1    = 0xFE;
684     SCON1     = 0x10;
685     SBCON1    = 0x43;
686
687     //115200 Uart1
688     // SBRLL1    = 0x30;
689     // SBRLH1    = 0xFF;
690     // SCON1     = 0x10;
691     // SBCON1    = 0x43;
692 }
693

```

```

694  /*-----*/
695  Init_int ()
696  -----
697  Descriptif:
698  Interruption 0 : P0.6 - Descendant
699  Interruption 1 : P0.7 - Descendant
700  Entree   : --
701  Sortie    : --
702  ----- */
703  void Init_int()
704  {
705      EX0=0;//Autorise l'interruption externe 0
706      EX1=0;//Autorise l'interruption externe 1
707      IT0=1;//interruption 0 sur flanc
708      IT1=1;//interruption 1 sur flanc
709      IT01CF= IT01CF &~ 0xFF;//Clear le registre des interruption
710          // +----- INT1 Polarite
711          // |+++++ Selection du canal P0.x de l'interruption 1
712          // |||||+--- INT0 Polarite
713          // |||||++- Selection du canal P0.x de l'interruption 0
714          // ||||||| (000: Select P0.0)
715          // ||||||| ...
716          // ||||||| ...
717          // ||||||| (111: Select P0.7)
718      IT01CF= IT01CF | 0x76;// 01110110
719
720      IE0=0; IE0=0; IE0=0;//clear du flag d'interruption 0 (3* pour eviter les bug)
721      IE1=0; IE1=0; IE1=0;//clear du flag d'interruption 1 (3* pour eviter les bug)
722      EX0=1;//Active l'interruption 0
723      EX1=1;//Active l'interruption 1
724  }
725
726  /*-----*/
727  TimerInit ()
728  -----
729  Descriptif:
730  Timer 0 : Mode 16bit - Prediv 48 - vitesse definit par gVitesseRAH-L
731  Timer 1 : Mode 8bit - Prediv 48 - baudrate 57600
732  Entree   : --
733  Sortie    : --
734  ----- */
735  void TimerInit()
736  {
737      //Timer 0 et 1
738      TR0 = 0;//Stop le timer 0
739      TR1 = 0;
740      ET0 = 0;//Desactive l'interruption du timer 0
741      ET1=0;
742      TMOD &= ~0x0F;//Clear le registre du mode
743          // +----- Timer 1 Gate Control
744          // |----- Counter/Timer1 Select
745          // ||++--- choix du mode du timer 1
746          // ||||+--- Timer 0 Gate Control
747          // |||||+-- Counter/Timer0 Select
748          // ||||||+- choix du mode du timer 0
749          // ||||||| (00 : Mode 0, 13-bit Counter/Timer)
750          // ||||||| (01 : Mode 1, 16-bit Counter/Timer)
751          // ||||||| (10 : Mode 2, 8-bit Counter/Timer with reload
752          // ||||||| (11 : Mode 3, Two 8-bit Counter/Timers)
753      TMOD |= 0x21;// 00000001
754      CKCON &= ~0x07;//clear les bit de selection pour les timer 0 et 1
755          // +----- Timer 3 High Byte Clock Select.
756          // |----- Timer 3 Low Byte Clock Select.
757          // ||++--- Timer 2 High Byte Clock Select
758          // ||||+--- Timer 2 Low Byte Clock Select.
759          // |||||+-- Timer 1 Clock Select.
760          // |||||+-- Timer 0 Clock Select.
761          // ||||||+- Timer 0/1 Prescale Bits.
762          // ||||||| (00: System clock divided by 12)
763          // ||||||| (01: System clock divided by 4)
764          // ||||||| (10: System clock divided by 48)
765          // ||||||| (11: External clock divided by 8
766      CKCON |= 0x02;// 00000010
767      TH0=gVitesseRAH;//Charge la valeur dans le registre MSB du timer 0
768      TL0=gVitesseRAL;//Charge la valeur dans le registre LSB du timer 0
769      TH1=TL1=BAUD_UART0;//Charge la valeur dans le registre LSB du timer 0
770      TF0 = 0;//clear le flag d'interruption du timer 0

```

```

771     TF1 = 0;
772     ET0 = 1; // Autorise l'interruption du timer 0
773     //ET1 = 1;
774
775     //Init Timer 4
776     SFRPAGE = CONFIG_PAGE;
777     TMR4CN = 0x00; //Mode 16 bit auto reload
778     TMR4RLH=0xF7;
779     TMR4RLL=0x8A;
780     SFRPAGE = LEGACY_PAGE;
781 }
782
783 /*-----*
784     ClockInit ()
785 -----*/
786 Descriptif: Initialisation du mode de fonctionnement du clock systeme
787     choix : SYSCLK : oscillateur HF interne à 48 MHz
788
789     Entrée : --
790     Sortie : --
791 -----*/
792 void ClockInit()
793 {
794
795     // +----- clock interne LF
796     // | (1 : oscillateur LF : enable)
797     // | (0 : oscillateur LF: desable)
798     // |+----- en lecture seule 1 : signal que oscillateur
799     // ||      interne fonctionne à sa valeur de prog.
800     // ||+----- réglage fin de la fréquence de l'osc. LF
801     // |||||+--- choix du diviseur :
802     // |||||    (00 : Osc LF /8 -> f = 10 KHz)
803     // |||||    (01 : Osc LF /4 -> f = 20 KHz)
804     // |||||    (10 : Osc LF /2 -> f = 40 KHz)
805     // |||||    (11 : Osc LF /1 -> f = 80 KHz)
806     OSCLCN |= 0x00; // 00000000
807
808     // +----- non utilisé
809     // |+----- Sélection du clock USB
810     // ||      (010 : Oscil ext. : limiter la conso.)
811     // ||+---- clock out select
812     // ||||    (0 : sortie sysclk non synchronisée)
813     // ||||    (1 : sortie sysclk synchronisée)
814     // ||||+--- choix du clock systeme
815     // |||||    (000 : Oscil interne 48/4 = 1.5, 3, 6 ou
816     // |||||      12 MHz selon le choix du diviseur
817     // |||||      dans OSCICN
818     // |||||    (001 : Oscil externe = x MHz)
819     // |||||    (010 : Oscil interne 48/2 = 24 MHz)
820     // |||||    (011 : Oscil interne 48/1 = 48 MHz)
821     // |||||    (100 : Oscil interne LF = 80 KHz max)
822     // |||||    (101 à 111 : réservés)
823     CLKSEL = 0x03; // 00000011
824
825     // +----- clock interne HF
826     // |      (1 : oscillateur LF : enable)
827     // |      (0 : oscillateur LF: desable)
828     // |+----- en lecture seule 1 : signal que oscillateur
829     // ||      interne fonctionne à sa valeur de prog.
830     // ||+----- 1 : suspend l'oscillateur interne
831     // ||+--- non utilisés
832     // |||||+--- choix du diviseur :
833     // |||||    (00 : 12/8 -> f = 1.5 MHz)
834     // |||||    (01 : 12/4 -> f = 3 MHz)
835     // |||||    (10 : 12/2 -> f = 6 MHz)
836     // |||||    (11 : 12/1 -> f = 12 MHz)
837     OSCICN = 0xC3; // 11000011
838
839     FLSCL = 0x90; // A utiliser si le clock system est à 48 MHz
840
841 } // ClockInit -----*/
842
843 /**
844 // SMBus_Init
845 //-----*/
846
847 */

```

```

848 // Return Value : None
849 // Parameters   : None
850 //
851 // SMBus configured as follows:
852 // - SMBus enabled
853 // - Slave mode inhibited
854 // - Timer2 used as clock source. The maximum SCL frequency will be
855 // approximately 1/3 the Timer1 overflow rate
856 // - Setup and hold time extensions enabled
857 // - Bus Free and SCL Low timeout detection enabled
858 //
859 void SMBus_Init (void)
860 {
861     SMB0CF = 0x5E;                                // Use Timer2 overflows as SMBus clock
862                                         // source;
863                                         // Disable slave mode;
864                                         // Enable setup & hold time
865                                         // extensions;
866                                         // Enable SMBus Free timeout detect;
867                                         // Enable SCL low timeout detect;
868
869     SMB0CF |= 0x80;                               // Enable SMBus;
870 }
871
872
873 //-----
874 // Timer2_Init
875 //-----
876 //
877 // Return Value : None
878 // Parameters   : None
879 //
880 // Timer2 configured as the SMBus clock source as follows:
881 // - Timer2 in 8-bit auto-reload mode
882 // - SYSCLK or SYSCLK / 4 as Timer1 clock source
883 // - Timer2 overflow rate => 3 * SMB_FREQUENCY
884 // - The resulting SCL clock rate will be ~1/3 the Timer2 overflow rate
885 // - Timer2 enabled
886 //
887 void Timer2_Init (void)
888 {
889     TMR2CN    |= 0x0C; //0x0C
890     TMR2RLH   |= 0x7B; //0x7B
891 }
892
893 //-----
894 // Timer3_Init
895 //-----
896 //
897 // Return Value : None
898 // Parameters   : None
899 //
900 // Timer3 configured for use by the SMBus low timeout detect feature as
901 // follows:
902 // - Timer3 in 16-bit auto-reload mode
903 // - SYSCLK/12 as Timer3 clock source
904 // - Timer3 reload registers loaded for a 25ms overflow period
905 // - Timer3 pre-loaded to overflow after 25ms
906 // - Timer3 enabled
907 //
908 void Timer3_Init (void)
909 {
910     TMR3CN = 0x00;                                // Timer3 configured for 16-bit auto-
911                                         // reload, low-byte interrupt disabled
912
913     CKCON &= ~0x40;                               // Timer3 uses SYSCLK/12
914
915     TMR3RL = -(SYSCLK/12/40);                   // Timer3 configured to overflow after
916     TMR3 = TMR3RL;                                // ~25ms (for SMBus low timeout detect):
917                                         // 1/.025 = 40
918
919     EIE1 |= 0x80;                                // Timer3 interrupt enable
920     TMR3CN |= 0x04;                               // Start Timer3
921 }
922
923 //-----
924 // SMBus Interrupt Service Routine (ISR)

```

```

925 //-----+
926 //
927 // SMBus ISR state machine
928 // - Master only implementation - no slave or arbitration states defined
929 // - All incoming data is written to global variable array <gSMBDataIN>
930 // - All outgoing data is read from global variable array <gSMBDataOUT>
931 //
932 void SMBus_ISR (void) interrupt 7
933 {
934     bit FAIL = 0;                                // Used by the ISR to flag failed
935                                         // transfers
936
937     static unsigned char sent_byte_counter;
938     static unsigned char rec_byte_counter;
939
940     if (ARBLOST0 == 0)                            // Check for errors
941     {
942         // Normal operation
943         switch (SMB0CN & 0xF0)                  // Status vector
944         {
945             // Master Transmitter/Receiver: START condition transmitted.
946             case SMB_MTSTA:
947                 SMB0DAT = gTarget;                // Load address of the gTarget slave
948                 SMB0DAT &= 0xFE;                // Clear the LSB of the address for the
949                                         // R/W bit
950                 SMB0DAT |= SMB_RW;              // Load R/W bit
951                 STA0 = 0;                   // Manually clear START bit
952                 rec_byte_counter = 1;        // Reset the counter
953                 sent_byte_counter = 1;       // Reset the counter
954             break;
955
956             // Master Transmitter: Data byte transmitted
957             case SMB_MTDB:
958                 if (ACK0)                      // Slave ACK0?
959                 {
960                     if (SMB_RW == WRITE)        // If this transfer is a WRITE,
961                     {
962                         if (sent_byte_counter <= gSMBNumBytesToWR)
963                         {
964                             // send data byte
965                             SMB0DAT = gSMBDataOUT[sent_byte_counter-1];
966                             sent_byte_counter++;
967                         }
968                         else
969                         {
970                             STO0 = 1;                // Set STO0 to terminate transfer
971                             SMB_BUSY = 0;            // And free SMBus interface
972                         }
973                     }
974                     else {}                  // If this transfer is a READ,
975                                         // proceed with transfer without
976                                         // writing to SMB0DAT (switch
977                                         // to receive mode)
978
979                 }
980             }
981             // If slave NACK,
982             {
983                 STO0 = 1;                // Send STOP condition, followed
984                 STA0 = 1;                // By a START
985                 NUM_ERRORS++;          // Indicate error
986             }
987             break;
988
989             // Master Receiver: byte received
990             case SMB_MRDB:
991                 if (rec_byte_counter < gSMBNumBytesToRD)
992                 {
993                     gSMBDataIN[rec_byte_counter-1] = SMB0DAT; // Store received
994                                         // byte
995                     ACK0 = 1;                  // Send ACK0 to indicate byte received
996                     rec_byte_counter++;      // Increment the byte counter
997                 }
998                 else
999                 {
1000                     gSMBDataIN[rec_byte_counter-1] = SMB0DAT; // Store received
1001                                         // byte

```

```

1002             SMB_BUSY = 0;           // Free SMBus interface
1003             ACK0 = 0;            // Send NACK to indicate last byte
1004                                         // of this transfer
1005
1006             STO0 = 1;            // Send STOP to terminate transfer
1007         }
1008         break;
1009
1010     default:
1011         FAIL = 1;             // Indicate failed transfer
1012                                         // and handle at end of ISR
1013         break;
1014
1015     } // end switch
1016 }
1017 else
1018 {
1019     // ARBLOST = 1, error occurred... abort transmission
1020     FAIL = 1;
1021 } // end ARBLOST if
1022
1023 if (FAIL)                                // If the transfer failed,
1024 {
1025     SMB0CF &= ~0x80;                      // Reset communication
1026     SMB0CF |= 0x80;
1027     STA0 = 0;
1028     STO0 = 0;
1029     ACK0 = 0;
1030
1031     SMB_BUSY = 0;                         // Free SMBus
1032
1033     FAIL = 0;
1034     //LED = 0;
1035
1036     NUM_ERRORS++;                        // Indicate an error occurred
1037 }
1038
1039     SI0 = 0;                            // Clear interrupt flag
1040 }
1041
1042 //-----
1043 // Timer3 Interrupt Service Routine (ISR)
1044 //-----
1045 //
1046 // A Timer3 interrupt indicates an SMBus SCL low timeout.
1047 // The SMBus is disabled and re-enabled here
1048 //
1049 void Timer3_ISR (void) interrupt 14
1050 {
1051     SMB0CF &= ~0x80;                      // Disable SMBus
1052     SMB0CF |= 0x80;                       // Re-enable SMBus
1053     TMR3CN &= ~0x80;                      // Clear Timer3 interrupt-pending
1054                                         // flag
1055     STA0 = 0;
1056     SMB_BUSY = 0;                         // Free SMBus
1057 }
1058
1059 //-----
1060 // Support Functions
1061 //-----
1062
1063 //-----
1064 // SMB_Write
1065 //-----
1066 //
1067 // Return Value : None
1068 // Parameters   : None
1069 //
1070 // Writes a single byte to the slave with address specified by the <gTarget>
1071 // variable.
1072 // Calling sequence:
1073 // 1) Write gTarget slave address to the <gTarget> variable
1074 // 2) Write outgoing data to the <gSMBDataOUT> variable array
1075 // 3) Call SMB_Write()
1076 //
1077 void SMB_Write (void)
1078 {

```

```

1079     while (SMB_BUSY);                                // Wait for SMBus to be free.
1080     SMB_BUSY = 1;                                    // Claim SMBus (set to busy)
1081     SMB_RW = 0;                                     // Mark this transfer as a WRITE
1082     STA0 = 1;                                      // Start transfer
1083 }
1084 -----
1085 //-----SMB_Read-----
1086 // SMB_Read
1087 //-----
1088 //
1089 // Return Value : None
1090 // Parameters : None
1091 //
1092 // Reads a single byte from the slave with address specified by the <gTarget>
1093 // variable.
1094 // Calling sequence:
1095 // 1) Write gTarget slave address to the <gTarget> variable
1096 // 2) Call SMB_Write()
1097 // 3) Read input data from <gSMBDataIN> variable array
1098 //
1099 void SMB_Read (void)
1100 {
1101     while (SMB_BUSY);                                // Wait for bus to be free.
1102     SMB_BUSY = 1;                                    // Claim SMBus (set to busy)
1103     SMB_RW = 1;                                     // Mark this transfer as a READ
1104
1105     STA0 = 1;                                      // Start transfer
1106
1107     while (SMB_BUSY);                                // Wait for transfer to complete
1108 }
1109
1110 /*
1111 -----PortInit ()-----
1112 Descriptif: autorise le fonctionnement du crossbar et de l'uart0
1113 Entrée : --
1114 Sortie : --
1115 */
1116
1117 void PortInit ()
1118 {
1119     // P0.0 - SDA (SMBus0), Open-Drain, Digital
1120     // P0.1 - SCL (SMBus0), Push-Pull, Digital
1121     // P0.2 - TX1 (UART1), Push-Pull, Digital
1122     // P0.3 - RX1 (UART1), Open-Drain, Digital
1123     // P0.4 - TX0 (UART0), Push-Pull, Digital
1124     // P0.5 - RX0 (UART0), Open-Drain, Digital
1125     // P0.6 - Unassigned, Open-Drain, Digital
1126     // P0.7 - Unassigned, Open-Drain, Digital
1127
1128     // P1.0 - Skipped, Push-Pull, Digital
1129     // P1.1 - Skipped, Push-Pull, Digital
1130     // P1.2 - Skipped, Push-Pull, Digital
1131     // P1.3 - Skipped, Open-Drain, Digital
1132     // P1.4 - Skipped, Push-Pull, Digital
1133     // P1.5 - Skipped, Push-Pull, Digital
1134     // P1.6 - Skipped, Push-Pull, Digital
1135     // P1.7 - Skipped, Open-Drain, Digital
1136
1137     // P2.0 - Skipped, Open-Drain, Digital
1138     // P2.1 - Skipped, Open-Drain, Digital
1139     // P2.2 - Skipped, Open-Drain, Digital
1140     // P2.3 - Skipped, Open-Drain, Digital
1141     // P2.4 - Unassigned, Open-Drain, Digital
1142     // P2.5 - Unassigned, Open-Drain, Digital
1143     // P2.6 - Unassigned, Open-Drain, Digital
1144     // P2.7 - Unassigned, Open-Drain, Digital
1145
1146     // P3.0 - Unassigned, Open-Drain, Digital
1147
1148     P0MDOUT = 0x16;
1149     P1MDOUT = 0xDD;//0xFF = schema
1150     P1SKIP = 0xFF;
1151     P2SKIP = 0x0F;
1152     XBR0 |= 0x05;
1153     XBR1 |= 0x40;
1154     XBR2 |= 0x01;
1155

```

```
1156  
1157 } // PortInit -----
```

```

1  /*=====
2   Slave_telescope - Tanguy Dietrich
3  =====
4  Descriptif: Permet de reguler la temperature des miroir ou des lentille
5      d'un telescope en utilisant la temperature du point de rose
6      ou des potentiometre
7
8  =====*/
9
10 #include <reg51f380.h>      // registres 51f38C
11 #include "SmBus0.h"
12 #include "PID.h"
13 #include "Delay48M.h"
14 #include "math.h"
15
16 #define LOADVALUE_T0 15536
17 #define LOADVALUE_BAUD 247
18
19 // === FONCTIONS PROTOTYPES=====
20 void ClockInit ();           // init. clock systme
21 void PortInit ();           // init. config des ports
22 void SMBus_Init (void);
23 void Timer3_Init (void);
24 void Timer2_Init (void);
25 void Uart0Init();
26 void SMB_Write (void);
27 void SMB_Read (void);
28 void PCA_Init();
29 void decodeUart0();
30 void TimerInit();
31 void ADCInit();
32 void configEeh210(unsigned char mode);
33 long convertToTemp_i2c(unsigned int value);
34 unsigned int convertToHumi(unsigned int value);
35 long getTempAmbiente();
36 unsigned int GetHumidite();
37 void actu_Pca0_Pid(int tempCapt,int consigne);
38 void actu_Pca1_Pid(int tempCapt,int consigne);
39 void actu_Pca2_Pid(int tempCapt,int consigne);
40 void actu_Pca3_Pid(int tempCapt,int consigne);
41 int convertLm35ToTemp(unsigned int adc_pos,unsigned int adc_neg);
42 void reset_i2c();
43 int calculRose(float t_ambiente, float humidite);
44 unsigned char ConvertToPca(unsigned int adc);
45 void ActuAvgMesure(unsigned int tabMesure[12][NMESURE], unsigned int tabSortie[]);
46 sbit SDA = P0^6;
47 sbit SCL= P0^7;
48 // === MAIN =====
49
50 //SMBus0
51 unsigned char gSMBNumBytesToWR = 2; // Number of bytes to write
52                                     // Master -> Slave
53 unsigned char gSMBNumBytesToRD = 3; // Number of bytes to read
54                                     // Master <- Slave
55
56 // Global holder for SMBus data
57 // All receive data is written here
58 xdata unsigned char gSMBDataIN[NUM_BYTES_MAX_RD];
59
60 // Global holder for SMBus data.
61 // All transmit data is read from here
62 xdata unsigned char gSMBDataOUT[NUM_BYTES_MAX_WR];
63
64 unsigned char gTarget;           // gTarget SMBus slave address
65
66 bit SMB_BUSY;                 // Software flag to indicate when the
67                                // SMB_Read() or SMB_Write() functions
68                                // have claimed the SMBus
69
70 bit SMB_RW;                   // Software flag to indicate the
71                                // direction of the current transfer
72
73 xdata unsigned long NUM_ERRORS; // Counter for the number of errors.
74
75
76
77 //Uart

```

```

78 xdata unsigned char gUart0Tx[15] ="";
79 xdata unsigned char gUart0Rx[15];
80 unsigned char gUart0NbrByteTx=0;
81 unsigned char gUart0NbrByteRx=0;
82 bit gUart0FlagReceive;
83
84 //Mesure ADC
85 xdata unsigned int gMesure[12][NMESURE];
86 xdata unsigned int gMesureAvg[12];
87 bit gFlagMesureAvg=0;
88
89 bit gFlagActuI2c=0;
90 long gTempAmbiante=0;
91 unsigned int gHumidite=0;
92 unsigned char gModeRegulation=MODE_UART;
93 bit gFlagPot=0;
94 void main ()
95 {
96     //unsigned char i=0;
97     //unsigned char temp_pwm=0;
98     int tempSensor;
99     int temp_rose=0;
100    PCA0MD &= ~0x40;           // WDTE = 0 (disable watchdog timer)
101    ClockInit ();             // init. clock systme
102    PortInit ();              // init. config des ports
103    Timer3_Init();            // Configure Timer3 for use with SMBus
104                                // low timeout detect
105    Timer2_Init();
106    SMBus_Init ();             // Configure and enable SMBus
107    PCA_Init();
108    TimerInit();
109    Uart0Init();
110    ADCInit();
111
112 EA=1;//Autorise toutes les interruption
113 IE|=0x90;
114 EIE1 |= 0x09;
115 TR1=1;
116 TR0=1;
117 SDA=1;
118 reset_i2c();
119 configEeh210(0);
120 gHumidite=GetHumidite();
121 gTempAmbiante=getTempAmbiante();
122 while (1)
123 {
124     gModeRegulation=P0&0x07;
125     if(gUart0FlagReceive)
126     {
127         gUart0FlagReceive=0;
128         decodeUart0();
129     }
130
131     if(gFlagMesureAvg)
132     {
133         gFlagMesureAvg=0;
134         ActuAvgMesure(gMesure,gMesureAvg);
135         if(gModeRegulation==MODE_PID)
136         {
137             tempSensor=convertLm35ToTemp(gMesureAvg[LM35_0_POS],gMesureAvg[LM35_0_NEG]);
138             actu_Pca0_Pid(tempSensor,temp_rose);
139
140             tempSensor=convertLm35ToTemp(gMesureAvg[LM35_1_POS],gMesureAvg[LM35_1_NEG]);
141             actu_Pca1_Pid(tempSensor,temp_rose);
142
143             tempSensor=convertLm35ToTemp(gMesureAvg[LM35_2_POS],gMesureAvg[LM35_2_NEG]);
144             actu_Pca2_Pid(tempSensor,temp_rose);
145
146             tempSensor=convertLm35ToTemp(gMesureAvg[LM35_3_POS],gMesureAvg[LM35_3_NEG]);
147             actu_Pca3_Pid(tempSensor,temp_rose);
148         }
149     }
150
151     if(gFlagPot)
152     {
153         gFlagPot=0;
154         if(gModeRegulation==MODE_POT)

```

```

155     {
156         PCA0CPH0=ConvertToPca(gMesure[POT_0][0]);
157         PCA0CPH1=ConvertToPca(gMesure[POT_1][0]);
158         PCA0CPH2=ConvertToPca(gMesure[POT_2][0]);
159         PCA0CPH3=ConvertToPca(gMesure[POT_3][0]);
160     }
161 }
162
163 if(gFlagActuI2c)
164 {
165     gHumidite=GetHumidite();
166     gTempAmbiante=getTempAmbiante();
167     temp_rose=calculRose(gTempAmbiante, gHumidite);
168     temp_rose+=CORRECTION_TEMPERATURE;
169     //temp_rose=DEMO_CONSIGNE;
170     gFlagActuI2c=0;
171     //temp_rose=calculRose(300, 60);
172 }
173 } // End while (1)
174 } // main =====
175
176 /*-----*/
177 timer0()
178 -----
179 Descriptif: Fonction d'interruption Timer0 vecteur 1
180     Temporisation de 50milli
181     Mode : 16bit
182 Entree   : --
183 Sortie   : --
184 */
185 void timer0() interrupt 1
186 {
187     static unsigned char cpt_i2c=0;
188     static unsigned char cpt_adc=0;
189     TR0=0;
190     TH0=LOADVALUE_T0/256;//Charge la valeur dans le registre MSB du timer 0
191     TL0=LOADVALUE_T0%256;//Charge la valeur dans le registre LSB du timer 0
192     //TR0=1;
193     if(cpt_adc>=DELAI_MESURE)
194     {
195         cpt_adc=0;
196         AD0BUSY=1;
197     }
198     cpt_adc++;
199     if(cpt_i2c>=20)
200     {
201         cpt_i2c=0;
202         gFlagActuI2c=1;
203     }
204     cpt_i2c++;
205     TR0=1;
206 }
207
208 /*-----*/
209 ADCComplete()
210 -----
211 Descriptif: fonction d'interruption de fin de conversion de l'ADC vecteur 10
212 Pin + : P1.1
213 Pin - : GND
214 Aligner a Droite
215 Conversion sur : AD0BUSY=1;
216 Clock SAR : 2000000
217 Entree   : --
218 Sortie   : --
219 */
220 void ADCComplete() interrupt 10
221 {
222     unsigned char i=0;
223     static unsigned char cptMesure=0;
224     gMesure[AMX0P][cptMesure]=ADC0L+(ADC0H<<8);
225     if(AMX0P<=0x0A)
226     {
227         AMX0P++; //change le canal de l'ADC a P2.3
228         for(i=0;i<25;i++)//Attend que les condensateur se decharge
229         {
230             }
231     }

```

```

232     AD0BUSY=1; //lance une conversion
233 }
234 else
235 {
236     AMX0P=0x00; //remet le canal P2.0
237     //MovingAvg(gMesure,gAvgMesure);
238     cptMesure++;
239     gFlagPot=1;
240     if(cptMesure==NMESURE)
241     {
242         gFlagMesureAvg=1; //Mets le flag a 1 pour signifier au main d'actualiser
243         cptMesure=0;
244     }
245 }
246 AD0INT=0;
247 }
248
249 /*-----*
250 uart0()
251 -----
252 Descriptif: Fonction d'interruption de l'uart0 vecteur 4
253 Entrée : --
254 Sortie : --
255 *-----*/
256 void uart0() interrupt 4
257 {
258     if(TIO)
259     {
260         TIO=0;
261         if(gUart0Tx[gUart0NbrByteTx]!=0)
262         {
263             SBUF0=gUart0Tx[gUart0NbrByteTx];
264             gUart0NbrByteTx++;
265         }
266         else
267         {
268             gUart0NbrByteTx=0;
269         }
270     }
271
272     if(RIO)
273     {
274         RIO=0;
275         gUart0Rx[gUart0NbrByteRx]=SBUF0;
276         if(gUart0Rx[gUart0NbrByteRx]=='#')
277         {
278             gUart0FlagReceive=1;
279             gUart0NbrByteRx=0;
280         }
281         else
282         {
283             gUart0NbrByteRx=(gUart0NbrByteRx+1)%20;
284         }
285     }
286 }
287
288 /*-----*
289 ActuAvgMesure ()
290 -----
291 Descriptif: Calcul la moyenne de mesure effectuer par l'ADC
292 Entrée :
293     - unsigned int tabMesure[12][NMESURE] - tableau de mesure
294     - unsigend int tabSortie[] - tableau de sortie pour les moyenne
295 Sortie : --
296 *-----*/
297 void ActuAvgMesure(unsigned int tabMesure[12][NMESURE], unsigned int tabSortie[])
298 {
299     unsigned char canal=0;
300     unsigned char nMesure=0;
301     unsigned long somme;
302     for(canal=0;canal<12;canal++)
303     {
304         somme=0;
305         for(nMesure=0;nMesure<NMESURE;nMesure++)
306         {
307             somme+=tabMesure[canal][nMesure];
308         }
}

```

```

309         tabSortie[canal]=somme/NMESURE;
310     }
311 }
312
313 /*-----*
314  ConvertToPca ()
315 -----*
316 Descriptif: Convertie une valeur entre 0 et 1023, de 0 a 255
317 Entrée   : unsigend int adc (0 ... 1023)
318 Sortie    : unsigend char (0 ... 255)
319 */
320 unsigned char ConvertToPca(unsigned int adc)
321 {
322     return 255-((unsigned long)adc*255)/1023;
323 }
324
325 /*-----*
326  calculRose ()
327 -----*
328 Descriptif:
329 Entrée   :
330     - float t_ambiente
331     - float humidite
332 Sortie    : int (-32767 ... +32767) la temperature du point de rosé
333     Ex : -2767 --> -27.67[C]
334 */
335 int calculRose(float t_ambiente, float humidite)
336 {
337     float tRose=0;
338     tRose=pow(((float)humidite/100),0.125);
339     tRose=tRose*(112+(9*t_ambiente)/1000);
340     tRose+=((t_ambiente)/1000)-112;
341     return tRose*10;
342 }
343
344 /*-----*
345  convertLm35ToTemp ()
346 -----*
347 Descriptif:
348 Entrée   : --
349 Sortie    : --
350 */
351 int convertLm35ToTemp(unsigned int adc_pos,unsigned int adc_neg)
352 {
353     int temp=0;
354     temp=adc_pos-adc_neg;
355     temp=((long)temp*REF_ADC)/100;
356     return temp;
357 }
358
359 /*-----*
360  actu_Pca0_Pid ()
361 -----*
362 Descriptif: calcul le pwm a appliquer sur la resistance en fonction du PID
363 Entrée   :
364     - int tempCapt
365     - int consigne
366 Sortie    : --
367 */
368 void actu_Pca0_Pid(int tempCapt,int consigne)
369 {
370     static int pwml=0;
371     static unsigned char old_error=0;
372     int error=0;
373     if(tempCapt<consigne)
374     {
375         error=consigne-tempCapt;
376         pwml=255-((int)error*KP)+((old_error-error)*KD);
377         //pwml=0;//mode tout ou rien
378     }
379     else
380     {
381         pwml=255;//0%
382         error=0;
383     }
384     old_error=error;
385     if(pwml<=0)

```

```

386     {
387         PCA0CPM0|=0x40; //reactive le pca0 si il a ete desactiver
388         PCA0CPH0=0;
389     }
390     else if(pwm1>=255)
391     {
392         PCA0CPH0=255;
393         PCA0CPM0&=~0x40; //Clear ecom0 afin d'obtenir 0%
394     }
395     else
396     {
397         PCA0CPH0=pwm1;
398     }
399 }
400
401 /*-----*
402 actu_Pca1_Pid ()
403 -----*/
404 Descriptif: calcul le pwm a appliquer sur la resistance en fonction du PID
405 Entrée   :
406             - int tempCapt
407             - int consigne
408 Sortie    : --
409 */-----*/
410 void actu_Pca1_Pid(int tempCapt,int consigne)
411 {
412     static int pwm1=0;
413     static unsigned char old_error=0;
414     int error=0;
415     if(tempCapt<consigne)
416     {
417         error=consigne-tempCapt;
418         pwm1=255-((int)error*KP)+((old_error-error)*KD);
419         //pwm1=0;//mode tout ou rien
420     }
421     else
422     {
423         pwm1=255;//0%
424         error=0;
425     }
426     old_error=error;
427     if(pwm1<=0)
428     {
429         PCA0CPM1|=0x40; //reactive la pca0 si il a ete desactiver
430         PCA0CPH1=0;
431     }
432     else if(pwm1>=255)
433     {
434         PCA0CPH1=255;
435         PCA0CPM1&=~0x40; //Clear ecom0 afin d'obtenir 0%
436     }
437     else
438     {
439         PCA0CPH1=pwm1;
440     }
441 }
442
443 /*-----*
444 actu_Pca2_Pid ()
445 -----*/
446 Descriptif: calcul le pwm a appliquer sur la resistance en fonction du PID
447 Entrée   :
448             - int tempCapt
449             - int consigne
450 Sortie    : --
451 */-----*/
452 void actu_Pca2_Pid(int tempCapt,int consigne)
453 {
454     static int pwm1=0;
455     static unsigned char old_error=0;
456     int error=0;
457     if(tempCapt<consigne)
458     {
459         error=consigne-tempCapt;
460         pwm1=255-((int)error*KP)+((old_error-error)*KD);
461         //pwm1=0;//mode tout ou rien
462     }

```

```

463     else
464     {
465         pwm1=255;//0%
466         error=0;
467     }
468     old_error=error;
469     if(pwm1<=0)
470     {
471         PCA0CPM2|=0x40;//reactive la pca0 si il a ete desactiver
472         PCA0CPH2=0;
473     }
474     else if(pwm1>=255)
475     {
476         PCA0CPH2=255;
477         PCA0CPM2&=~0x40;//Clear ecom0 afin d'obtenir 0%
478     }
479     else
480     {
481         PCA0CPH2=pwm1;
482     }
483 }
484
485 /*-----*/
486 actu_Pca3_Pid ()
487 -----
488 Descriptif: calcul le pwm a appliquer sur la resistance en fonction du PID
489 Entrée   :
490         - int tempCapt
491         - int consigne
492 Sortie    : --
493 */
494 void actu_Pca3_Pid(int tempCapt,int consigne)
495 {
496     static int pwm1=0;
497     static unsigned char old_error=0;
498     int error=0;
499     if(tempCapt<consigne)
500     {
501         error=consigne-tempCapt;
502         pwm1=255-((int)error*KP)+((old_error-error)*KD);
503         //pwm1=0;//mode tout ou rien
504     }
505     else
506     {
507         pwm1=255;//0%
508         error=0;
509     }
510     old_error=error;
511     if(pwm1<=0)
512     {
513         PCA0CPM3|=0x40;//reactive la pca0 si il a ete desactiver
514         PCA0CPH3=0;
515     }
516     else if(pwm1>=255)
517     {
518         PCA0CPH3=255;
519         PCA0CPM3&=~0x40;//Clear ecom0 afin d'obtenir 0%
520     }
521     else
522     {
523         PCA0CPH3=pwm1;
524     }
525 }
526
527 /*-----*/
528 reset_i2c()
529 -----
530 Descriptif: Verifie si le bus I2C est bloquer, et effectue un reset du bus
531 Entrée   : --
532 Sortie   : --
533 */
534 void reset_i2c()
535 {
536     unsigned char i=0;
537     if(!SDA)
538     {
539         P0SKIP      = 0xCF;

```

```

540     XBR0      = 0x01;
541     // If slave is holding SDA low because of an improper SMBus reset or error
542     while(!SDA)
543     {
544         // Provide clock pulses to allow the slave to advance out
545         // of its current state. This will allow it to release SDA.
546         SCL = 0;                      // Drive the clock low
547         for(i = 0; i < 100; i++);    // Hold the clock low
548         SCL = 1;                      // Release the clock
549         for(i = 0; i < 100; i++);    // Hold the clock low
550     }
551     POSKIP    = 0x0F;
552     XBR0      = 0x05;
553 }
554 }
555
556 /*-----*
557     getTempAmbiante ()
558 -----*/
559     Descriptif: demande la temperature au capteur EEH210
560     Entrée   : --
561     Sortie   : long
562 *-----*/
563 long getTempAmbiante()
564 {
565     while(SMB_BUSY);
566     gTarget = 0x80;//0x81      (0x40 7 bit + 1 bit = 0x81 1 == read
567     gSMBDataOUT[0] = 0xF3;//E3 = Hold master / F3 = no hold master
568     gSMBNumBytesToWR = 1;
569     SMB_Write(); // Initiate SMBus write
570     //TEST=~TEST;
571     while(SMB_BUSY);
572     Delay_1ms(20);//attend la fin de mesure
573     gTarget = 0x81;
574     gSMBNumBytesToRD = 4;
575     SMB_Read(); // Initiate SMBus read
576     //Delay_1ms(20);//attend la fin de mesure
577
578     return convertToTemp_i2c((unsigned int )gSMBDataIN[1]<<8)+gSMBDataIN[2]);
579 }
580
581 /*-----*
582     GetHumidite ()
583 -----*/
584     Descriptif: demande l'humidite ambiante au capteur EEH210
585     Entrée   : --
586     Sortie   : --
587 *-----*/
588 unsigned int GetHumidite()
589 {
590     gTarget = 0x80;//0x81      (0x40 7 bit + 1 bit = 0x81 1 == read
591     gSMBDataOUT[0] = 0xF5;
592     gSMBNumBytesToWR = 1;
593     SMB_Write(); // Initiate SMBus write
594     while(SMB_BUSY);
595     Delay_1ms(20);//attend la fin de mesure
596     gTarget = 0x81;
597     gSMBNumBytesToRD = 4;
598     SMB_Read(); // Initiate SMBus read
599     return convertToHumi((unsigned int )gSMBDataIN[1]<<8)+gSMBDataIN[2]);
600 }
601
602
603 /*-----*
604 configEeh210()
605 -----*/
606     Descriptif: Configure le capteur EEH210
607     fonction incomplete, il faut lire les parametre avant d'crire
608     dessus
609     Entrée   : unsigned char tab[] - la chaine de caractere a decoder
610     Sortie   : --
611 *-----*/
612 void configEeh210(unsigned char mode)
613 {
614     while(SMB_BUSY);
615     gTarget = 0x80;//0x81      (0x40 7 bit + 1 bit = 0x81 1 == read
616     gSMBDataOUT[0] = 0xE6;

```

```

617     switch(mode)
618     {
619         case 0 :
620             gSMBDataOUT[1] = 0x3A;
621             break;
622         case 1 :
623             gSMBDataOUT[1] = 0x3B;
624             break;
625         case 2 :
626             gSMBDataOUT[1] = 0xBA;
627             break;
628         case 3 :
629             gSMBDataOUT[1] = 0xBB;
630             break;
631     }
632
633     gSMBNumBytesToWR = 2;
634     SMB_Write(); // Initiate SMBus write
635 }
636
637 /*-----*/
638 convertToTemp()
639 -----
640 Descriptif: Convertie une valeur sur 16 bit en temperature
641 Entrée : unsigned int value - la donne a convertir
642 Sortie : long - la temperature convertie avec un facteur 100
643 */
644 long convertToTemp_i2c(unsigned int value)
645 {
646     value = value &~ 0x03;
647     return (((long)value*17572)>>16)-4685;
648 }
649
650 /*-----*/
651 convertToHumi()
652 -----
653 Descriptif: Convertie une valeur sur 16 bit en humidité
654 Entrée : unsigned int value - la donne a convertir
655 Sortie : unsigned int - la température convertie
656 */
657 unsigned int convertToHumi(unsigned int value)
658 {
659     value = value &~ 0x03;
660     return (((unsigned long)value*125)>>16)-6;;
661 }
662
663 /*-----*/
664 ADCInit()
665 -----
666 Descriptif:
667 Pin + : P1.1
668 Pin - : GND
669 Aligner a Droite
670 Conversion sur : AD0BUSY=1;
671 Clock SAR : 2000000
672 ref sur
673 Entrée : --
674 Sortie : --
675 */
676 void ADCInit()
677 {
678
679     AMX0P=0x00;//Entrée positive sur P2.0
680     AMX0N=0x1f;//Entrée négative sur GND
681         // ++++++ ADC0 SAR Conversion Clock Period Bits.
682         // |||||+--- AD0LJST ajustement a droite
683         // |||||+--- Reserver
684         // |||||+--- |
685     ADC0CF=0xb8;// 10111000 - clk SAR de 2000000Hz aligner a droite
686         // +----- ADC0 Enable Bit
687         // |+---- ADC0 Track Mode Bit.
688         // ||+---- AD0INT ADC0 Conversion Complete Interrupt Flag
689         // |||+---- AD0BUSY
690         // |||+---- ADC0 Window Compare Interrupt Flag
691         // |||||+--- ADC0 Start of Conversion Mode Select
692         // |||||+--- |
693     ADC0CN=0x80;// 10000000

```

```

694             // +----- Reference Buffer Gain Select.
695             // |+----- Unused
696             // |||+---- Regulator Reference Override
697             // ||||+--- Voltage Reference Select.
698             // |||||+-- Temperature Sensor Enable Bit
699             // |||||+-- Internal Analog Bias Generator Enable Bit.
700             // |||||+- On-chip Reference Buffer Enable Bit.
701             // |||||||
702 REF0CN=0x08;// 00010000
703
704     EIE1 |=0x08;
705 }
706
707
708 void decodeUart0()
709 {
710     unsigned int testPwm=0;
711     int temp=0;
712 //PCA0
713 if((gUart0Rx[0]=='R') && (gUart0Rx[1]=='0') && (gUart0Rx[2]=='_'))
714 {
715     testPwm=(gUart0Rx[3]-'0')*100;
716     testPwm+=(gUart0Rx[4]-'0')*10;
717     testPwm+=gUart0Rx[5]-'0';
718     if(testPwm>255)
719     {
720         //Erreur
721         testPwm=255;
722     }
723     testPwm=255-testPwm;
724     if(testPwm==255)//0%
725     {
726         PCA0CPH0=0xFF;
727         PCA0CPM0=&~0x40;//RAZ de ECOMn pour forcer le pwm a 0%
728     }
729     else
730     {
731         PCA0CPH0=testPwm;
732         PCA0CPM0|=0x40;//ECOMn a 1 pour autoriser le fonctionnement
733     }
734 }
735
736 //PCA1
737 if((gUart0Rx[0]=='R') && (gUart0Rx[1]=='1') && (gUart0Rx[2]=='_'))
738 {
739     testPwm=(gUart0Rx[3]-'0')*100;
740     testPwm+=(gUart0Rx[4]-'0')*10;
741     testPwm+=gUart0Rx[5]-'0';
742     if(testPwm>255)
743     {
744         //Erreur
745         testPwm=255;
746     }
747     testPwm=255-testPwm;
748     if(testPwm==255)//0%
749     {
750         PCA0CPH1=0xFF;
751         PCA0CPM1=&~0x40;//RAZ de ECOMn pour forcer le pwm a 0%
752     }
753     else
754     {
755         PCA0CPH1=testPwm;
756         PCA0CPM1|=0x40;//ECOMn a 1 pour autoriser le fonctionnement
757     }
758 }
759
760 //PCA2
761 if((gUart0Rx[0]=='R') && (gUart0Rx[1]=='2') && (gUart0Rx[2]=='_')) //maximum 255
762 {
763     testPwm=(gUart0Rx[3]-'0')*100;
764     testPwm+=(gUart0Rx[4]-'0')*10;
765     testPwm+=gUart0Rx[5]-'0';
766     if(testPwm>255)
767     {
768         //Erreur
769         testPwm=255;
770     }

```

```

771     testPwm=255-testPwm;
772     if(testPwm==255) //0%
773     {
774         PCA0CPH2=0xFF;
775         PCA0CPM2&=~0x40;//RAZ de ECOMn pour forcer le pwm a 0%
776     }
777     else
778     {
779         PCA0CPH2=testPwm;
780         PCA0CPM2|=0x40;//ECOMn a 1 pour autoriser le fonctionnement
781     }
782 }
783
784 //PCA3
785 if((gUart0Rx[0]=='R') && (gUart0Rx[1]=='3') && (gUart0Rx[2]=='_'))
786 {                                                 //maximum 255
787     testPwm=(gUart0Rx[3]-'0')*100;
788     testPwm+=(gUart0Rx[4]-'0')*10;
789     testPwm+=gUart0Rx[5]-'0';
790     if(testPwm>255)
791     {
792         //Erreur
793         testPwm=255;
794     }
795     testPwm=255-testPwm;
796     if(testPwm==255) //0%
797     {
798         PCA0CPH3=0xFF;
799         PCA0CPM3&=~0x40;//RAZ de ECOMn pour forcer le pwm a 0%
800     }
801     else
802     {
803         PCA0CPH3=testPwm;
804         PCA0CPM3|=0x40;//ECOMn a 1 pour autoriser le fonctionnement
805     }
806 }
807
808 if((gUart0Rx[0]=='G') && (gUart0Rx[1]=='E') && (gUart0Rx[2]=='T') && (gUart0Rx[3]=='_') && (gUart0Rx[4]=='T'))
809 {
810     if(gUart0Rx[5]=='0')//Capteur_i2c
811     {
812         temp=getTempAmbiante();
813         temp=temp/10;
814     }
815     else if(gUart0Rx[5]=='1')//LM35_0
816     {
817         temp=convertLm35ToTemp(gMesureAvg[LM35_0_POS],gMesureAvg[LM35_0_NEG]);
818     }
819     else if(gUart0Rx[5]=='2')//LM35_1
820     {
821         temp=convertLm35ToTemp(gMesureAvg[LM35_1_POS],gMesureAvg[LM35_1_NEG]);
822     }
823     else if(gUart0Rx[5]=='3')//LM35_2
824     {
825         temp=convertLm35ToTemp(gMesureAvg[LM35_2_POS],gMesureAvg[LM35_2_NEG]);
826     }
827     else if(gUart0Rx[5]=='4')//LM35_3
828     {
829         temp=convertLm35ToTemp(gMesureAvg[LM35_3_POS],gMesureAvg[LM35_3_NEG]);
830     }
831     else if(gUart0Rx[5]=='R')//LM35_3
832     {
833         temp=calculRose(getTempAmbiante(), GetHumidite());
834     }
835     else
836     {
837         temp=999;//Erreur
838     }
839
840 //envoie sur l'uart0
841 if(temp>=0)
842 {
843     gUart0Tx[0]='+';
844 }
845 else
846 {

```

```

847     gUart0Tx[0]='-';
848     temp=temp*(-1);
849 }
850 //ex pour 28.3°C --> 283
851 //gUart0Tx[1]=(temp/1000) +'0';//2
852 gUart0Tx[1]=(temp/100) +'0';//2
853 gUart0Tx[2]=((temp%100)/10) +'0';//8
854 gUart0Tx[3]='.'; //.
855 gUart0Tx[4]=((temp%10)) +'0';//3
856 gUart0Tx[5]='\n';
857 gUart0Tx[6]=0;
858 gUart0NbrByteTx=0;
859 TI0=1;
860 }
861
862
863 if((gUart0Rx[0]=='G') && (gUart0Rx[1]=='E') && (gUart0Rx[2]=='T') && (gUart0Rx[3]=='_') && (gUart0Rx[4]=='H'))
864 {
865     temp=GetHumidite();
866     gUart0Tx[0]=(temp/100) +'0';
867     gUart0Tx[1]=(temp/10) +'0';
868     gUart0Tx[2]=((temp%10))+ '0';
869     gUart0Tx[3]='\n';
870     gUart0Tx[4]=0;
871     gUart0NbrByteTx=0;
872     TI0=1;
873 }
874
875 /*-----*
876 TimerInit ()
877 -----*/
878 Descriptif:
879 Timer 0 : Mode 16bit - Prediv 48 - tempo 50milli
880 Timer 1 : Mode 8bit - Prediv 48 - vitesse : 57600bit/s
881 Entrée : --
882 Sortie : --
883 */-----*/
884 void TimerInit()
885 {
886 TR0 = 0;//Stop le timer 0
887 TR1 = 0;//Stop le timer 1
888 ET0 = 0;//Desactive l'interruption du timer 0
889 ET1 = 0;//Desactive l'interruption du timer 1
890 TMOD &= ~0xFF;//Clear le registre des mode
891         // +----- Timer 1 Gate Control
892         // |+----- Counter/Timer1 Select
893         // ||+---- choix du mode du timer 1
894         // ||||+--- Timer 0 Gate Control
895         // |||||+-- Counter/Timer0 Select
896         // |||||++ choix du mode du timer 0
897         // |||||| (00 : Mode 0, 13-bit Counter/Timer)
898         // |||||| (01 : Mode 1, 16-bit Counter/Timer)
899         // |||||| (10 : Mode 2, 8-bit Counter/Timer with Auto-Reload)
900         // |||||| (11 : Mode 3, Two 8-bit Counter/Timers)
901 TMOD |= 0x21;// 00100001
902 CKCON &= ~0x07;//clear les bit de selection pour les timer 0 et 1 et le prescalaire
903         // +----- Timer 3 High Byte Clock Select.
904         // |+----- Timer 3 Low Byte Clock Select.
905         // ||+---- Timer 2 High Byte Clock Select
906         // |||+--- Timer 2 Low Byte Clock Select.
907         // ||||+--- Timer 1 Clock Select.
908         // |||||+-- Timer 0 Clock Select.
909         // |||||++ Timer 0/1 Prescale Bits.
910         // |||||| (00: System clock divided by 12)
911         // |||||| (01: System clock divided by 4)
912         // |||||| (10: System clock divided by 48)
913         // |||||| (11: External clock divided by 8 (synchronized with the system clock))
914 CKCON |= 0x02;// 00000010
915 TH0=LOADVALUE_T0/256;//Charge la valeur dans le registre MSB du timer 0
916 TL0=LOADVALUE_T0%256;//Charge la valeur dans le registre LSB du timer 0
917 TL1=TH1=LOADVALUE_BAUD;//Charge la valeur du timer 1
918 TF0 = 0;//clear le flag d'interruption du timer 0
919 TF1 = 0;//clear le flag d'interruption du timer 1
920 ET0 = 1;//Autorise l'interruption du timer 0
921 ET1 = 0;//Autorise l'interruption du timer 1
922 }

```

```

923
924
925
926 void PCA_Init()
927 {
928     PCA0CN      = 0x40; //Active le PCA
929     PCA0MD      |= 0x02; //PCA SYSCLK Prediv
930     PCA0CPM0    = 0x42; //active le mode PWM et le comparateur du PCA0
931     PCA0CPM1    = 0x42; //active le mode PWM et le comparateur du PCA1
932     PCA0CPM2    = 0x42; //active le mode PWM et le comparateur du PCA0
933     PCA0CPM3    = 0x42; //active le mode PWM et le comparateur du PCA1
934     PCA0CPH0    = 0xFF; //valeur a charger pour obtenir %
935     PCA0CPH1    = 0xFF; //valeur a charger pour obtenir %
936     PCA0CPH2    = 0xFF; //valeur a charger pour obtenir %
937     PCA0CPH3    = 0xFF; //valeur a charger pour obtenir %
938 }
939
940 /*-----*
941     ClockInit ()
942 -----*/
943     Descriptif: Initialisation du mode de fonctionnement du clock systeme
944     choix : SYSCLK : oscillateur HF interne @ 48 MHz
945
946     Entrée      : --
947     Sortie      : --
948 *-----*/
949 void ClockInit()
950 {
951
952     // +----- clock interne LF
953     // | (1 : oscillateur LF : enable)
954     // | (0 : oscillateur LF: desable)
955     // |+----- en lecture seule 1 : signal que oscillateur
956     // ||          interne fonctionne @ sa valeur de prog.
957     // ||+----- ralage fin de la fréquence de l'osc. LF
958     // |||||+--- choix du diviseur :
959     // |||||      (00 : Osc LF /8 -> f = 10 KHz)
960     // |||||      (01 : Osc LF /4 -> f = 20 KHz)
961     // |||||      (10 : Osc LF /2 -> f = 40 KHz)
962     // |||||      (11 : Osc LF /1 -> f = 80 KHz)
963     OSCLCN |= 0x00; // 00000000
964
965     // +----- non utilis
966     // |+----- Slection du clock USB
967     // |||      (010 : Oscil ext. : limiter la conso.)
968     // |||+---- clock out select
969     // |||      (0 : sortie sysclk non synchronis)
970     // |||      (1 : sortie sysclk synchronis)
971     // |||+--- choix du clock systeme
972     // |||      (000 : Oscil interne 48/4 = 1.5, 3, 6 ou
973     // |||                  12 MHz selon le choix du diviseur
974     // |||                  dans OSCICN
975     // |||      (001 : Oscil externe = x MHz)
976     // |||      (010 : Oscil interne 48/2 = 24 MHz)
977     // |||      (011 : Oscil interne 48/1 = 48 MHz)
978     // |||      (100 : Oscil interne LF = 80 KHz max)
979     // |||      (101 @ 111 : rbservs)
980     CLKSEL = 0x03; // 00000011
981
982     // +----- clock interne HF
983     // |          (1 : oscillateur LF : enable)
984     // |          (0 : oscillateur LF: desable)
985     // |+----- en lecture seule 1 : signal que oscillateur
986     // ||          interne fonctionne @ sa valeur de prog.
987     // ||+----- 1 : suspend l'oscillateur interne
988     // |||+--- non utilis
989     // |||||+--- choix du diviseur :
990     // |||||      (00 : 12/8 -> f = 1.5 MHz)
991     // |||||      (01 : 12/4 -> f = 3 MHz)
992     // |||||      (10 : 12/2 -> f = 6 MHz)
993     // |||||      (11 : 12/1 -> f = 12 MHz)
994     OSCICN = 0xC3; // 11000011
995
996     FLSCL = 0x90; // A utiliser si le clock system est @ 48 MHz
997
998 } // ClockInit -----
999

```

```

1000
1001 //-----
1002 // SMBus_Init
1003 //-----
1004 //
1005 // Return Value : None
1006 // Parameters : None
1007 //
1008 // SMBus configured as follows:
1009 // - SMBus enabled
1010 // - Slave mode inhibited
1011 // - Timer2 used as clock source. The maximum SCL frequency will be
1012 // approximately 1/3 the Timer1 overflow rate
1013 // - Setup and hold time extensions enabled
1014 // - Bus Free and SCL Low timeout detection enabled
1015 //
1016 void SMBus_Init (void)
1017 {
1018     SMB0CF = 0x5E;                                // Use Timer2 overflows as SMBus clock
1019                                         // source;
1020                                         // Disable slave mode;
1021                                         // Enable setup & hold time
1022                                         // extensions;
1023                                         // Enable SMBus Free timeout detect;
1024                                         // Enable SCL low timeout detect;
1025
1026     SMB0CF |= 0x80;                               // Enable SMBus;
1027 }
1028
1029
1030 //-----
1031 // Timer2_Init
1032 //-----
1033 //
1034 // Return Value : None
1035 // Parameters : None
1036 //
1037 // Timer2 configured as the SMBus clock source as follows:
1038 // - Timer2 in 8-bit auto-reload mode
1039 // - SYSCLK or SYSCLK / 4 as Timer1 clock source
1040 // - Timer2 overflow rate => 3 * SMB_FREQUENCY
1041 // - The resulting SCL clock rate will be ~1/3 the Timer2 overflow rate
1042 // - Timer2 enabled
1043 //
1044 void Timer2_Init (void)
1045 {
1046     TMR2CN |= 0x0C;
1047     TMR2RLH |= 0xD4;
1048 }
1049
1050 //-----
1051 // Timer3_Init
1052 //-----
1053 //
1054 // Return Value : None
1055 // Parameters : None
1056 //
1057 // Timer3 configured for use by the SMBus low timeout detect feature as
1058 // follows:
1059 // - Timer3 in 16-bit auto-reload mode
1060 // - SYSCLK/12 as Timer3 clock source
1061 // - Timer3 reload registers loaded for a 25ms overflow period
1062 // - Timer3 pre-loaded to overflow after 25ms
1063 // - Timer3 enabled
1064 //
1065 void Timer3_Init (void)
1066 {
1067     TMR3CN = 0x00;                                // Timer3 configured for 16-bit auto-
1068                                         // reload, low-byte interrupt disabled
1069
1070     CKCON &= ~0x40;                             // Timer3 uses SYSCLK/12
1071
1072     TMR3RL = -(SYSCLK/12/40);                  // Timer3 configured to overflow after
1073     TMR3 = TMR3RL;                            // ~25ms (for SMBus low timeout detect):
1074                                         // 1/.025 = 40
1075
1076     EIE1 |= 0x80;                               // Timer3 interrupt enable

```

```

1077     TMR3CN |= 0x04;                                // Start Timer3
1078 }
1079
1080 //-----
1081 // SMBus Interrupt Service Routine (ISR)
1082 //-----
1083 //
1084 // SMBus ISR state machine
1085 // - Master only implementation - no slave or arbitration states defined
1086 // - All incoming data is written to global variable array <gSMBDataIN>
1087 // - All outgoing data is read from global variable array <gSMBDataOUT>
1088 //
1089 void SMBus_ISR (void) interrupt 7
1090 {
1091     bit FAIL = 0;                                // Used by the ISR to flag failed
1092                                         // transfers
1093
1094     static unsigned char sent_byte_counter;
1095     static unsigned char rec_byte_counter;
1096
1097     if (ARBLOST0 == 0)                            // Check for errors
1098     {
1099         // Normal operation
1100         switch (SMB0CN & 0xF0)                  // Status vector
1101         {
1102             // Master Transmitter/Receiver: START condition transmitted.
1103             case SMB_MTSTA:
1104                 SMB0DAT = gTarget;           // Load address of the gTarget slave
1105                 SMB0DAT &= 0xFE;          // Clear the LSB of the address for the
1106                                         // R/W bit
1107                 SMB0DAT |= SMB_RW;        // Load R/W bit
1108                 STA0 = 0;                // Manually clear START bit
1109                 rec_byte_counter = 1;    // Reset the counter
1110                 sent_byte_counter = 1;   // Reset the counter
1111             break;
1112
1113             // Master Transmitter: Data byte transmitted
1114             case SMB_MTDB:
1115                 if (ACK0)                   // Slave ACK0?
1116                 {
1117                     if (SMB_RW == WRITE)      // If this transfer is a WRITE,
1118                     {
1119                         if (sent_byte_counter <= gSMBNumBytesToWR)
1120                         {
1121                             // send data byte
1122                             SMB0DAT = gSMBDataOUT[sent_byte_counter-1];
1123                             sent_byte_counter++;
1124                         }
1125                         else
1126                         {
1127                             STO0 = 1;            // Set STO0 to terminate transfer
1128                             SMB_BUSY = 0;        // And free SMBus interface
1129                         }
1130                     }
1131                     else {}                  // If this transfer is a READ,
1132                                         // proceed with transfer without
1133                                         // writing to SMB0DAT (switch
1134                                         // to receive mode)
1135
1136
1137     }
1138     else                                         // If slave NACK,
1139     {
1140         STO0 = 1;            // Send STOP condition, followed
1141         STA0 = 1;            // By a START
1142         NUM_ERRORS++;       // Indicate error
1143     }
1144     break;
1145
1146     // Master Receiver: byte received
1147     case SMB_MRDB:
1148         if (rec_byte_counter < gSMBNumBytesToRD)
1149         {
1150             gSMBDataIN[rec_byte_counter-1] = SMB0DAT; // Store received
1151                                         // byte
1152             ACK0 = 1;                // Send ACK0 to indicate byte received
1153             rec_byte_counter++;    // Increment the byte counter

```

```

1154         }
1155     else
1156     {
1157         gSMBDataIN[rec_byte_counter-1] = SMB0DAT; // Store received
1158                                         // byte
1159         SMB_BUSY = 0;                      // Free SMBus interface
1160         ACK0 = 0;                        // Send NACK to indicate last byte
1161                                         // of this transfer
1162
1163         STO0 = 1;                         // Send STOP to terminate transfer
1164     }
1165     break;
1166
1167 default:
1168     FAIL = 1;                          // Indicate failed transfer
1169                                         // and handle at end of ISR
1170     break;
1171
1172 } // end switch
1173 }
1174 else
1175 {
1176     // ARBLOST = 1, error occurred... abort transmission
1177     FAIL = 1;
1178 } // end ARBLOST if
1179
1180 if (FAIL)                                // If the transfer failed,
1181 {
1182     SMB0CF &= ~0x80;                   // Reset communication
1183     SMB0CF |= 0x80;
1184     STA0 = 0;
1185     STO0 = 0;
1186     ACK0 = 0;
1187
1188     SMB_BUSY = 0;                     // Free SMBus
1189
1190     FAIL = 0;
1191     //LED = 0;
1192
1193     NUM_ERRORS++;                  // Indicate an error occurred
1194 }
1195
1196 SI0 = 0;                                 // Clear interrupt flag
1197 }
1198
1199 //-----
1200 // Timer3 Interrupt Service Routine (ISR)
1201 //-----
1202 //
1203 // A Timer3 interrupt indicates an SMBus SCL low timeout.
1204 // The SMBus is disabled and re-enabled here
1205 //
1206 void Timer3_ISR (void) interrupt 14
1207 {
1208     SMB0CF &= ~0x80;                 // Disable SMBus
1209     SMB0CF |= 0x80;                  // Re-enable SMBus
1210     TMR3CN &= ~0x80;                // Clear Timer3 interrupt-pending
1211                                         // flag
1212
1213     STA0 = 0;
1214     SMB_BUSY = 0;                  // Free SMBus
1215
1216
1217 //-----
1218 // Support Functions
1219 //-----
1220
1221 //
1222 // SMB_Write
1223 //-----
1224 //
1225 // Return Value : None
1226 // Parameters   : None
1227 //
1228 // Writes a single byte to the slave with address specified by the <gTarget>
1229 // variable.
1230 // Calling sequence:

```

```

1231 // 1) Write gTarget slave address to the <gTarget> variable
1232 // 2) Write outgoing data to the <gSMBDataOUT> variable array
1233 // 3) Call SMB_Write()
1234 //
1235 void SMB_Write (void)
1236 {
1237     while (SMB_BUSY);                                // Wait for SMBus to be free.
1238     SMB_BUSY = 1;                                    // Claim SMBus (set to busy)
1239     SMB_RW = 0;                                     // Mark this transfer as a WRITE
1240     STA0 = 1;                                      // Start transfer
1241 }
1242
1243 //-----
1244 // SMB_Read
1245 //-----
1246 //
1247 // Return Value : None
1248 // Parameters   : None
1249 //
1250 // Reads a single byte from the slave with address specified by the <gTarget>
1251 // variable.
1252 // Calling sequence:
1253 // 1) Write gTarget slave address to the <gTarget> variable
1254 // 2) Call SMB_Write()
1255 // 3) Read input data from <gSMBDataIN> variable array
1256 //
1257 void SMB_Read (void)
1258 {
1259     while (SMB_BUSY);                                // Wait for bus to be free.
1260     SMB_BUSY = 1;                                    // Claim SMBus (set to busy)
1261     SMB_RW = 1;                                     // Mark this transfer as a READ
1262
1263     STA0 = 1;                                      // Start transfer
1264
1265     while (SMB_BUSY);                                // Wait for transfer to complete
1266 }
1267
1268 *-----*
1269 Uart0Init()
1270 -----*
1271 Descriptif: Active la reception de l'UART0 - Mode 8 bit
1272 Entree   : --
1273 Sortie   : --
1274 -----* */
1275 void Uart0Init()
1276 {
1277     //Les bit font partie du registre SCON0
1278
1279     //Serial Port 0 Operation Mode.
1280     //0: 8-bit UART with Variable Baud Rate.
1281     //1: 9-bit UART with Variable Baud Rate.
1282     S0MODE=0;
1283
1284     //Multiprocessor Communication Enable.
1285     //The function of this bit is dependent on the Serial Port 0 Operation Mode:
1286     //Mode 0: Checks for valid stop bit.
1287     //0: Logic level of stop bit is ignored.
1288     //1: RIO will only be activated if stop bit is logic level 1.
1289     //Mode 1: Multiprocessor Communications Enable.
1290     //0: Logic level of ninth bit is ignored.
1291     //1: RIO is set and an interrupt is generated only when the ninth bit is logic 1.
1292     MCE0=0;
1293
1294     //Receive Enable.
1295     //0: UART0 reception disabled.
1296     //1: UART0 reception enabled.
1297     REN0=1;
1298
1299     //Ninth Transmission Bit.
1300     //The logic level of this bit will be sent as the ninth transmission bit in 9-bit UART Mode
1301     //(Mode 1). Unused in 8-bit mode (Mode 0).
1302     TB80=0;
1303
1304     //Ninth Receive Bit.
1305     //RB80 is assigned the value of the STOP bit in Mode 0; it is assigned the value of the
1306     //9th data bit in Mode 1.
1307     RB80=0;

```

```

1308     //Transmit Interrupt Flag.
1309     TI0=0;
1310
1311     //Receive Interrupt Flag.
1312     RI0=0;
1313 } // Uart0Init -----*
1315
1316
1317 /*-----*/
1318 PortInit ()
1319 -----
1320 Descriptif: autorise le fonctionnement du crossbar et de l'uart0
1321 Entrée   : --
1322 Sortie    : --
1323 -----*/
1324 void PortInit ()
1325 {
1326     // P0.0 - Skipped,      Open-Drain, Digital
1327     // P0.1 - Skipped,      Open-Drain, Digital
1328     // P0.2 - Skipped,      Open-Drain, Digital
1329     // P0.3 - Skipped,      Open-Drain, Digital
1330     // P0.4 - TX0 (UART0), Push-Pull,  Digital
1331     // P0.5 - RX0 (UART0), Open-Drain, Digital
1332     // P0.6 - SDA (SMBus0), Open-Drain, Digital
1333     // P0.7 - SCL (SMBus0), Push-Pull,  Digital
1334
1335     // P1.0 - Skipped,      Open-Drain, Analog
1336     // P1.1 - Skipped,      Open-Drain, Analog
1337     // P1.2 - Skipped,      Open-Drain, Analog
1338     // P1.3 - Skipped,      Open-Drain, Analog
1339     // P1.4 - Skipped,      Open-Drain, Analog
1340     // P1.5 - Skipped,      Open-Drain, Analog
1341     // P1.6 - Skipped,      Open-Drain, Analog
1342     // P1.7 - Skipped,      Open-Drain, Analog
1343
1344     // P2.0 - Skipped,      Open-Drain, Analog
1345     // P2.1 - Skipped,      Open-Drain, Analog
1346     // P2.2 - Skipped,      Open-Drain, Analog
1347     // P2.3 - Skipped,      Open-Drain, Analog
1348     // P2.4 - CEX0 (PCA), Push-Pull,  Digital
1349     // P2.5 - CEX1 (PCA), Push-Pull,  Digital
1350     // P2.6 - CEX2 (PCA), Push-Pull,  Digital
1351     // P2.7 - CEX3 (PCA), Push-Pull,  Digital
1352
1353     // P3.0 - Unassigned,   Open-Drain, Digital
1354
1355     P1MDIN    = 0x00;
1356     P2MDIN    = 0xF0;
1357     P0MDOUT   = 0x90;
1358     P2MDOUT   = 0xF0;
1359     P0SKIP    = 0x0F;
1360     P1SKIP    = 0xFF;
1361     P2SKIP    = 0x0F;
1362     XBRO     = 0x05;
1363     XBR1     = 0x44;
1364
1365 } // PortInit -----*

```