

AASun

Routeur multi voies pour installation
photovoltaïque

11/05/2024

Sommaire

1	Introduction.....	6
1.1	Réflexions préliminaires	6
1.2	Les bases du projet	6
1.3	Conception préliminaire.....	7
	1.3.1 Le MCU.....	7
	1.3.2 Mesure de la tension secteur.....	8
	1.3.3 Mesure des courants	9
	1.3.4 Alimentations	10
	1.3.5 Les accessoires	10
1.4	Sous quelle forme ?.....	11
2	Conception matérielle	13
2.1	Le module MCU	13
	2.1.1 Caractéristiques	13
	2.1.2 Schéma.....	14
	2.1.3 ADC	16
2.2	L'alimentation et le capteur de tension	16
2.3	La mesure de courant.....	18
	2.3.1 CST013.....	19
	2.3.2 SCT016.....	19
	2.3.3 CT08CL10	19
2.4	La commande des relais statiques	21
2.5	Interface TIC du Linky	21
2.6	Le connecteur d'extension.....	22
2.7	Choix de l'ampli opérationnel	22
3	Conception logicielle	26
3.1	Timer d'acquisition analogique	26
3.2	Acquisition analogique.....	26
3.3	Commande SSR.....	27
	3.3.1 Déterminer la formule théorique.....	28
	3.3.2 Application	28
	3.3.3 Correction de powerDiverterMax	29
3.4	Mesure de température	30
	3.4.1 Interface matérielle	30
	3.4.2 Logiciel.....	32
	3.4.3 Configuration.....	33
3.5	Date et heure.....	34
3.6	Architecture du logiciel	34
	3.6.1 meterTask() : la priorité haute	35
	3.6.2 AASun() : la priorité moyenne	35
	3.6.3 lowProcessesTask () : la priorité basse	36

3.6.4	Les interruptions	37
3.6.5	L'afficheur OLED.....	37
3.6.6	La LED D1	39
4	WIFI	40
4.1	Le module.....	40
4.2	Implémentation.....	41
4.2.1	Le mode « Access Point »	41
4.2.2	Le mode « Station ».....	42
4.2.3	Alimentation du module	42
4.3	Un peu de technique	42
4.4	Schéma du module.....	44
5	Câblage.....	46
5.1	Carte d'extension.....	47
6	Mise en route	49
6.1	Installation	49
6.2	Etalonnage	50
6.3	Calibration de la tension	52
6.4	Correction de la phase	52
6.5	Calibration des intensités	53
6.5.1	Correction de l'offset de l'ADC	54
6.5.2	Calibration de l'intensité	54
6.6	Calibration de la puissance actives	55
6.6.1	Correction de l'offset de puissance active.....	55
6.6.2	Calibration de la puissance active.....	55
6.7	Autres valeurs de configuration du routeur.....	56
6.8	Configuration réseau	56
6.9	Sauvegarde	56
6.10	Configuration du MCU	58
6.10.1	Les « Option Bits ».....	59
6.10.2	Mise en flash du logiciel	60
7	Routage et forçage	61
7.1	Les voies de sortie.....	62
7.2	Les voies d'entrées.....	62
7.3	Hystérésis de température	63
8	Routage	64
8.1	Syntaxe d'une règle.....	64
8.2	L'IHM	65
9	Forçage.....	67
9.1	Règles de forçage	67
9.2	Le mode STD	68

9.2.1	DMIN	68
9.2.2	DMAX	69
9.3	Le mode AUTO	69
9.3.1	Fonctionnement du filtrage	70
9.4	Le paramètre ratio	71
9.4.1	Ratio normal.....	71
9.4.2	Ratio burst	71
9.5	Actions manuelles	71
9.5.1	Cas du forçage en mode AUTO.....	72
9.6	L'IHM	73
10	L'historique	74
11	Anti légionellose.....	79
12	Mise en FLASH des pages HTTP	81
12.1.1	Mise en flash avec ST-LINK.....	81
12.1.2	Mise en flash avec un adaptateur USB/UART	84
13	Telnet.....	86
14	Conversion puissance – délai	88
14.1	Ampoule halogène.....	89
15	Défauts et améliorations.....	90



Le routeur présenté dans ce document est une réalisation personnelle, pas un produit commercial.

Il est réalisé soigneusement mais ne répond à aucune norme de sécurité particulière.

Ce document n'est pas un manuel d'utilisation. Il s'agit plutôt de notes de réalisation et d'un aide mémoire.

Si le lecteur décide de réaliser ce montage je serais heureux de le renseigner et de l'aider, mais toute garantie expresse ou implicite de qualité ou d'adéquation à un usage particulier sont exclues.

Ce montage est connecté au secteur 230V : toujours débrancher l'alimentation de l'appareil avant toute intervention.

1 Introduction

Ce document accompagne la version 1.18 de AASun, qui introduit entre autre le WIFI.

La production d'électricité photovoltaïque par les particuliers est devenue abordable. Et j'ai eu envie d'y toucher, à petite échelle. Après avoir installé 2 panneaux solaires de 420Wc chacun, le complément naturel c'est d'avoir un routeur pour consommer la production excédentaire.

Cela peut paraître paradoxal de vouloir un routeur pour une si petite installation, et quand on n'a pas de chauffe eau électrique. En réalité la motivation est de l'ordre de la curiosité : avoir à réfléchir à un problème et arriver au bout de la réalisation ont été des motivations fortes. Accessoirement j'ai signé une CACSI avec Enedis, donc en principe je ne devrais pas injecter mon (petit) surplus sur le réseau national.

1.1 Réflexions préliminaires

La première étape est de fouiller sur le web. Je suis rapidement tombé sur le site openenergymonitor.org qui est une mine d'information, et probablement un précurseur dans le domaine. Puis sur le site d'un autre précurseur Robin Emley : <https://mk2pvrouter.co.uk/index.html>.

Après ces lectures je commence à avoir une idée de ce qu'il est possible de faire. En fouillant un peu plus je tombe sur le [forum photovoltaïque](#), qui présente plein de réalisations diverses. Ce forum est une mine d'or !

Une des réalisations qui a retenu mon attention est l'[EcoPV](#) devenu le [MaxPV!](#) de Jetblack. Après avoir lu attentivement plusieurs milliers de messages sur cette réalisation mes idées sont plus claires : je vais réaliser mon routeur.

Il a fallu lui trouver un nom... Comme j'utilise le nom AdAstra pour signer mes réalisations je l'ai appelé AASun.

1.2 Les bases du projet

Les routeurs sont généralement basés sur plusieurs éléments :

- Un microcontrôleur (MCU) pour gérer l'appareil en particulier la partie temps réel.
- Une mesure de la tension secteur après conditionnement sur une entrée analogique (ADC) du MCU.
- Au moins 1 mesure de courant à l'aide d'un tore après conversion en tension puis amplification sur une entrée ADC.
- Une référence de tension pour l'ADC et le conditionnement des mesures analogiques.
- Un relais statique (SSR) à base de TRIAC pour la dérivation du surplus électrique.
- Une alimentation pour toute cette électronique.

En plus de ces éléments indispensables, on peut envisager des options agréables ou qui augmentent considérablement les fonctionnalités du routeur :

- Un afficheur OLED
- Une interface réseau filaire
- Une interface WIFI
- Une seconde sortie de routage
- Un relais pour un autre mode de routage
- Une ou plusieurs mesures de température pour un routage plus « intelligent »
- Des entrées TOR pour le comptage d'impulsions ou autre contact
- Une interface logicielle vers des solutions domotique
- La récupération des informations de la TIC du compteur Linky
- Etc la liste pourrait être très longue !

Clairement il faut choisir entre toutes ces possibilités pour définir notre routeur.

1.3 Conception préliminaire

Beaucoup de réalisations utilisent des cartes de type Arduino avec un MCU Atmel AVR. Je suis très admiratif des réalisations qui sont faites avec ces petits MCU

J'ai beaucoup utilisé ces MCU dans le temps (ils sont vraiment pratiques à utiliser même s'il faut débugger au printf et au scope), mais depuis j'utilise des STM32. Donc que faire avec ce type de MCU ?

1.3.1 Le MCU

J'ai quelques exemplaires de STM32G071CB qui restent d'une [réalisation précédente](#), et vu la pénurie de composants (mi 2021) je décide d'utiliser ceux là. Quelques caractéristiques pour situer la bête :

- Boitier TQFP 48 broches
- 128 KB de FLASH, 36 KB de RAM, horloge max 64 MHz
- 2 I2C
- 4 USART + 1 LPUART
- 2 SPI
- Plusieurs timers de 16 bits et 1 de 32 bits
- Une référence de tension interne et accessible à l'extérieur (ce n'est pas toujours le cas chez ST !), autant l'utiliser.
- 1 ADC et 16 entrées analogiques.

Mais il n'y a pas d'EEPROM interne, dommage...

Comme chez tous les microcontrôleurs il faudra jongler pour répartir l'ensemble des fonctionnalités nécessaires sur les pattes. Tout n'est pas utilisable simultanément...



Avec ce cœur équipé de mon RTOS maison je devrais pouvoir faire quelque chose.

1.3.2 Mesure de la tension secteur

Pour la mesure de tension, plutôt que d'utiliser un transformateur d'alimentation difficile à trouver je préfère d'utiliser un capteur dédié facilement achetable : ZMPT101B ou DL-PT202D (Le ZMPT101B semble avoir plusieurs sources avec des présentations différentes). Ils semblent avoir de bonnes caractéristiques



Technical Data:

	DL-PT202D	Unit
Rated input	2	mA (AC)
Rated output	2	mA (AC)
Rated sampling load	500	Ω
Rated sampling voltage	1	V (AC)
Turns ratio	1000/1000	---
Rated Phase shift	≤8 (RL=0Ω)	'
Accuracy	0.1	Level
Linearity	0.1	%
Maximum input	0~8 (RL≤125Ω)	mA (AC)
Applicable voltage	0~1000	V (AC)
DCR	113 ±10	Ω
Weight	14.1	g

1.3.3 Mesure des courants

J'ai un abonnement EDF de 6 KW et une installation solaire de 840Wc. Donc des courants pas très importants, je peux prendre un capteur d'un modèle bien connu : YHDC CST013 30A/1V.

Ce capteur fournit 1Vrms pour un courant de 30A qui le traverse (il a une résistance « burden » interne). Cela correspond 2.8V crête à crête.

Plus tard j'ai essayé le capteur YHDC SCT016 50A/50mA. Ce capteur n'a pas de résistance « burden » interne, et la valeur recommandée est de 10Ω. Cela donne $10 \times 0.05 = 0.5$ Vrms, soit 1.4Vpp pour 50A.

Valeurs à retenir lors de la conception du schéma électronique.



Il faut au moins 1 mesure de courant sur la ligne d'arrivée au tableau pour détecter les injections. Mais il est intéressant de mesurer aussi la production photovoltaïque et surtout la puissance routée, qui est difficile à estimer par logiciel.

Donc au moins 2 ou 3 mesures de courant sont nécessaires.

1.3.4 Alimentations

Le MCU a peu de besoins (~10mA), mais une interface réseau ou WIFI en a beaucoup plus, environ 300mA, en 3.3V.

A priori il est bon d'avoir du 5V en plus du 3.3V sur une carte.

J'aimerais limiter le câblage et n'avoir qu'une seule source à débrancher pour isoler le montage. Donc une seule connexion au secteur pour la mesure de la tension et la génération du 5V et du 3.3V.

L'alimentation est basée sur un module HI-Link de 3W qui peut fournir 5V/600mA, suivi d'un LT1117 pour le 3.3V

1.3.5 Les accessoires

- Un afficheur OLED en SPI. Sur un STM32 l'I2C est difficile à utiliser. Et deux boutons pour pouvoir faire défiler les pages dans les 2 sens
- Une LED présence 5V, une LED utilisateur (usage non défini).
- Une FLASH W25Q64 de 64 Mbits en SPI (parce que j'en ai quelques unes en stock) pour l'enregistrement de la configuration et des énergies mesurées. Comme cette FLASH est de grande taille, on peut aussi y loger les pages d'un petit serveur, ou envisager d'enregistrer plus de données (logger).
- Un module réseau filaire Wiznet W5500 en SPI. Il est autonome contient toute la stack UDP/TCP donc peu de logiciel nécessaire côté MCU. Ma maison est câblée, c'est donc ma première option pour la mise en réseau. Cela permet une liaison Telnet avec le AASun, et de fournir le serveur HTTP.



- Je n'ai aucune connaissance en développement WEB, je ne sais donc pas quoi choisir pour avoir un petit serveur en liaison WIFI. J'installe donc des empreintes pour un module Wemos D1 mini et un ESP-01S, tous les deux utilisent un UART. Il faudra choisir, les deux ne peuvent pas être utilisés en même temps.

Evolution : Interface WIFI utilisant un module ESP32-C3 Super Mini :



Donc finalement ni WemosD1 ni ESP-01 ! Pour le prototype j'ai fais une petite carte d'adaptation qui s'insère à la place du ESP-01, et peut cohabiter avec le module W5500.

- Interface Linky : elle est simple et ne nécessite que l'entrée RX d'un UART. C'est sympa de savoir ce que raconte le Linky à travers une liaison telnet !
- Entrées TOR générique ou de comptage. Il faut conditionner ces entrées pour supporter des tensions supérieures à 3.3V.
- Sorties SSR. Utilise un timer du MCU, qui lui même à 3 ou 4 canaux. Donc conditionnement de 2 sorties SSR en 5V.

J'ai commandé des SSR Jotta comme recommandé mais j'ai eu la surprise de voir que leur entrée est spécifiée à 4-32VDC, et non 3-32V comme sur la photo. Donc 5V impératif, mais j'ai pu vérifier sur le 1^{er} prototype qu'il marche aussi en 3.3V...

- Un relais pour un délestage secondaire.
- Un connecteur pour la mesure de températures avec des DS18B20. Ca peut servir...

1.4 Sous quelle forme ?

J'ai quelques connaissances de base en électronique et routage, donc je peux réaliser un circuit imprimé simple.

Je souhaite un système facile à installer et à câbler. Je choisis donc de réaliser une mono carte, qui puisse être installée sur un rail DIN. Cependant le SSR est externe : la connexion ne nécessite que 2 fils, et avec son radiateur il est encombrant.

C'est une première réalisation, donc une carte d'expérimentation pour avancer en terrain inconnu : pas de contrainte sur la taille, ni sur le coût (en restant raisonnable !).

Je prévois un module réutilisable pour supporter le MCU qui a un boîtier TQFP, et des amplificateurs opérationnels en DIP pour pouvoir les remplacer ou en essayer plusieurs modèles (quitte à utiliser un adaptateur pour ceux en SO-8).

2 Conception matérielle

Après cette étape de réflexion, les fonctionnalités à implémenter sont :

- Une alimentation 5V 500mA qui permet de générer 3.3V 400 mA
- La génération d'une ½ référence de tension
- Une référence de tension externe TL431 de 2.5V (réserve)
- 1 mesure de la tension secteur
- 2 mesures de courant sur jack 3.5mm
- 2 mesures de courant sur jack 3.5mm optionnelles
- 2 entrées conditionnées supportant au moins 27V pour comptage ou autre
- 2 commandes 5V pour SSR asynchrones (random)
- 1 relais mécanique
- 2 entrées/sorties non conditionnées (~~capteur DS18b20~~ ou autre)
- 1 écran 128x64 et 2 boutons. Possibilité de déport de l'écran et des boutons à quelques mètres.
- 1 module réseau filaire W5500
- 1 module Wemos D1 mini (inutilisé)
- 1 module ESP32-C3 Super Mini (remplace le ESP-01S)
- 1 FLASH W25Q64
- Interface TIC pour le Linky (en fait un optocoupleur)
- Un connecteur d'extension avec les broches inutilisées du MCU pour d'éventuelles extensions.

La suite de ce chapitre présente les principaux points de l'architecture matérielle.

2.1 Le module MCU

Réutilisation d'un module CPU utilisable sur plaque d'expérimentation ou PCB.

C'est très pratique pour les expérimentations.

2.1.1 Caractéristiques

Toutes les pattes du MCU sont disponibles sur les connecteurs

Le module supporte le connecteur 2x5 points de programmation/debug standard pour mes réalisations : SWD, SWC, Reset, RX, TX, 3.3V, GND, SWO (inexistant sur les Cortex-M0).

Un oscillateur 2532 optionnel (connectable par des gouttes de soudure)

Utilisation optionnelle du 3.3V filtré comme référence de tension externe pour l'ADC.

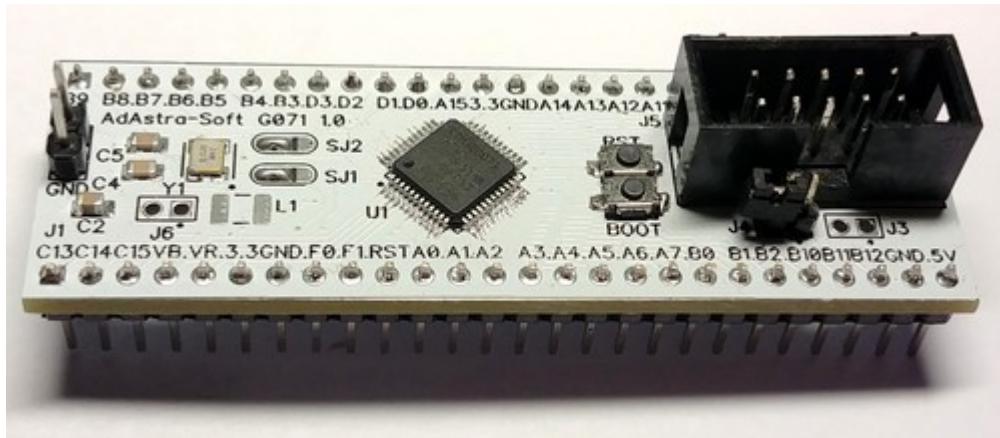
Bouton BOOT

Bouton Reset

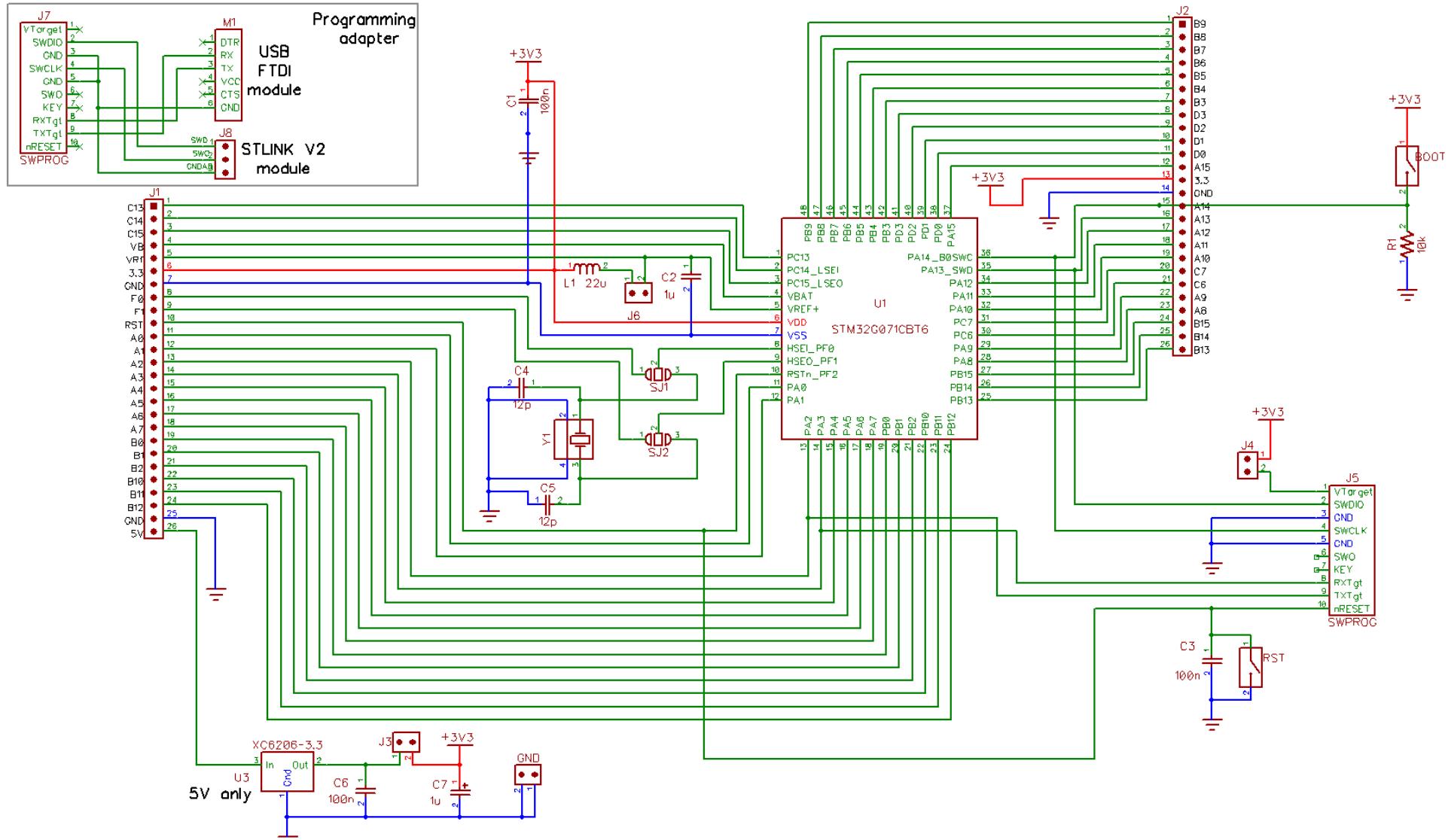
Choix de l'alimentation 3.3V à partir de :

- 3.3V du connecteur de programmation
- 3.3V externe
- 3.3V produit localement à partir du 5V externe (100 mA max)

Avec le circuit du module vient un petit circuit adaptateur permettant d'utiliser un module ST-LINK-V2 (type chinois...), ou un module convertisseur UART/USB pour mettre à jour le firmware et avoir une console.



2.1.2 Schéma



J3 is used to connect the external 5V power supply to the board 3.3V.
 J4 is used to connect the debug probe power output to the board 3.3V.
 CAUTION: J3 and J4 are exclusive: do not connect both power supplies at the same time!

Exception: when the debug probe needs a voltage reference (VTARGET is then a probe input)
 you can connect both J3 and J4.

When the board is powered from the 3.3V pins, do not connect J3.

STM32G071CB Bread board adapter

1.1

Nov 14, 2022
 G71CB_AD.dch

AC
 1/1

2.1.3 ADC

Le MCU dispose de plusieurs entrées analogiques et d'une référence de tension interne. Cette référence de tension peut sortir sur la broche VREF+, elle est configurée pour une valeur de 2.5V (c'est la valeur max possible). L'étendue des signaux analogiques mesurables est donc de 0 à 2.5V max.

A l'extérieur du MCU VREF est routée vers un pont diviseur qui permet d'avoir une tension $VREF/2 = 1.25V$. Ce 1.25V sert à polariser les signaux analogiques de tension et de courants à mesurer afin de les positionner au milieu de l'étendue de mesure.

Quelques caractéristiques de l'ADC :

- Horloge de conversion max de 35 MHz. J'utilise 32 MHz qui est la fréquence du cœur divisée par 2.
- Conversion en 12, 10, 8 ou 6 bits
- Rapide : 0.37 μ s/voie en 10 bits. Il est possible de spécifier le temps d'acquisition avant la conversion (en coups d'horloge ADC).
- Sur échantillonnage programmable de 0 à 256 en 2^n : L'ADC fait automatiquement plusieurs conversions et fournit la moyenne.

Après de multiples essais le mode 12 bits est trop bruité, et en utilisant le sur échantillonnage cela s'améliore un peu, mais le temps de conversion de 5 voies est prohibitif.

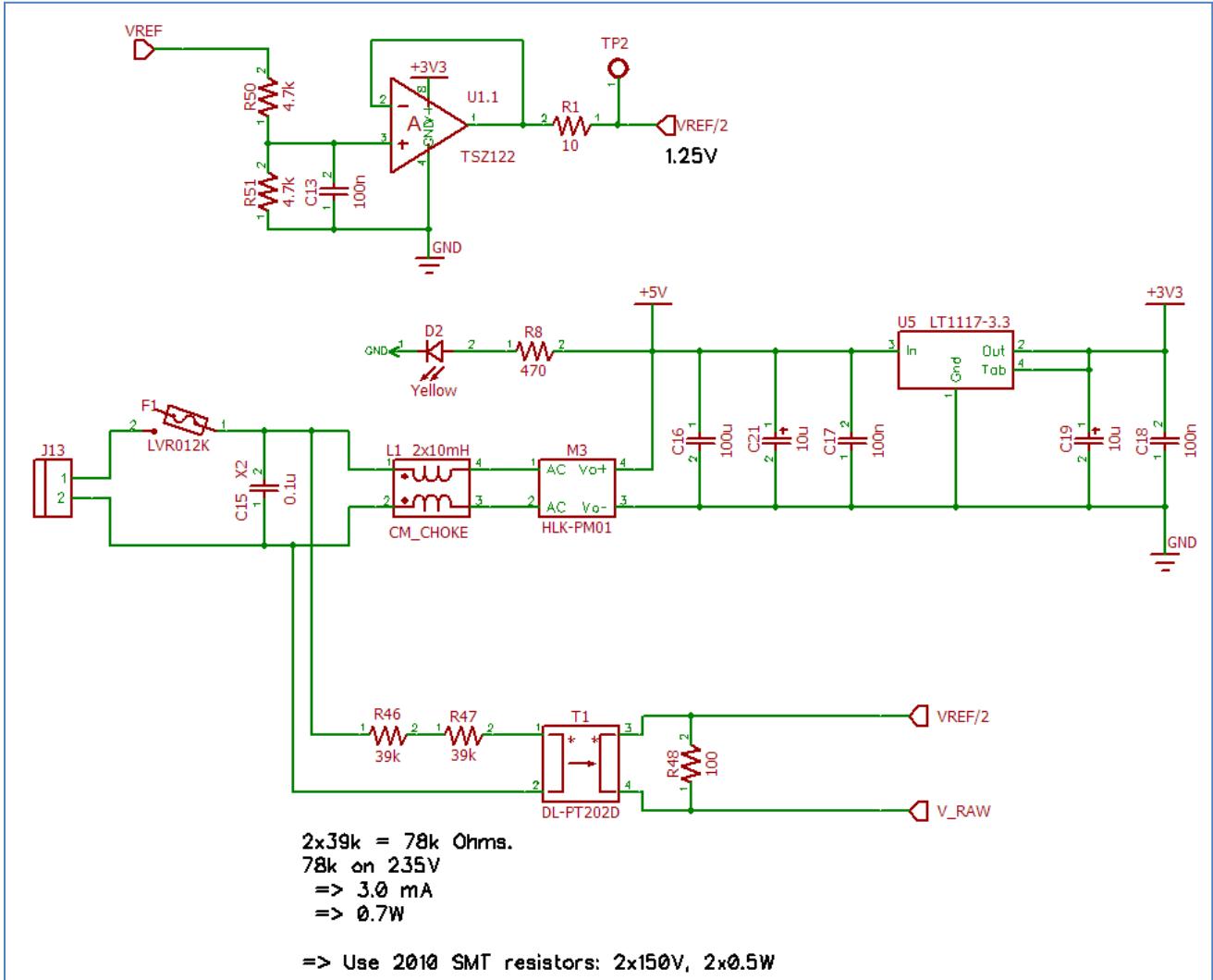
L'optimal semble d'utiliser 10 bits avec un sur échantillonnage de 16 et une durée d'échantillonnage de 7.5 coups d'horloge. On obtient un temps de conversion de 9 μ s par voie, soit 27 μ s pour 3 voies (volt + 2 courants).

Pour 5 voies cela donne 45 μ s. En réduisant la durée d'échantillonnage à 3.5 coups d'horloge, (~110ns) on peut obtenir 35 μ s.

2.2 L'alimentation et le capteur de tension

La datasheet du module convertisseur 230V/5V HLK_PM01 impose coté entrée l'utilisation d'un filtrage : Une self de mode commun de 10 à 15 mH et un condensateur X2 de 0.1 μ f.

Coté sortie l'habituel condensateur de lissage et le LT1117 avec ses capa.

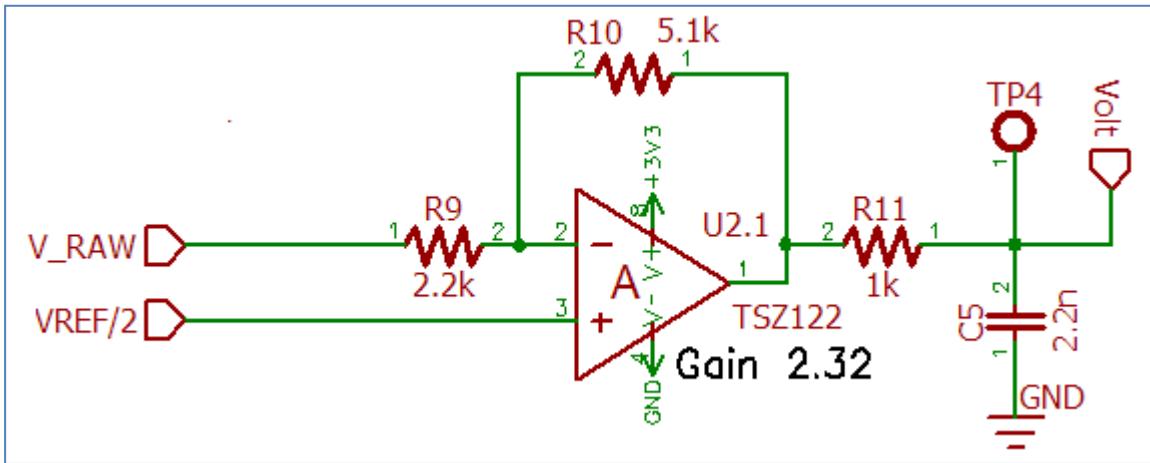


Le transformateur de mesure de tension est donné pour 2mA sur le primaire, mais la datasheet indique de meilleurs résultats à 3mA. La résistance sur le primaire doit donc être de $235\text{V} / 0.003\text{A} = 78 \text{ k}\Omega$. Cela induit une puissance dissipée de $235 \times 0.003 = 0.7\text{W}$.

Donc en utilisant comme résistance dans le primaire deux résistances format 2010 de 39 kΩ on obtient une résistance de 78 kΩ pouvant supporter 1W et 300Vrms.

En ce qui concerne la résistance dans le secondaire (dite de burden) la datasheet montre que plus la résistance est faible plus le déphasage est faible. On choisit une résistance de 100Ω ce qui donnera 0.85 Vpp.

Pour utiliser au mieux l'étendue de mesure de l'ADC il faut amplifier ce signal pour le rapprocher de 2.5Vpp. Un gain de 2.32 donne 1.97Vpp.



2.3 La mesure de courant

La mesure de courant utilise un transformateur de courant. Il fournit donc un courant qui est transformé en tension quand il passe dans une résistance dite de burden.

Exemple : avec le CT SCT013 100A/ 50mA bien connu, il est recommandé d'utiliser une résistance de burden de 22Ω . Cela donne :

$$0.05 \times 22 = 1.1V_{rms}, \text{ soit } 3.11V_{pp} \text{ pour } 100 \text{ A.}$$

Cette tension est trop élevée pour notre ADC. Il faut donc l'atténuer.

D'un autre coté beaucoup d'utilisations n'ont pas besoin de mesurer 100A. Si on se limite à 50A, on obtient :

$$0.05 \times 0.5 \times 22 = 0.55V_{rms}, \text{ soit } 1.55V_{pp} \text{ pour } 50 \text{ A.}$$

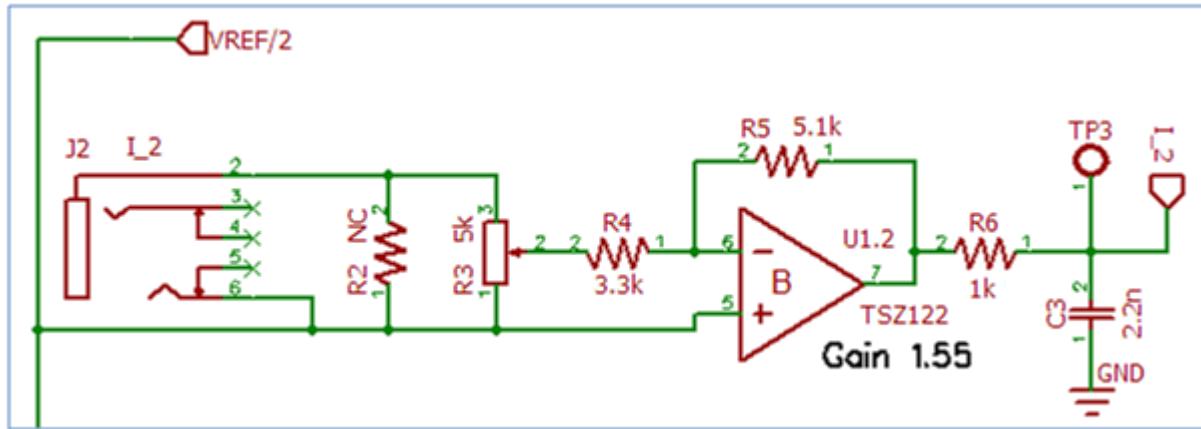
Dans ce cas il faut amplifier le signal.

Le conditionnement de la mesure de courant doit donc être capable d'atténuer ou d'amplifier suivant les cas pour utiliser au mieux la précision de l'ADC.

D'autre part il y a des CT avec résistance de burden intégrée, et qui généralement fournissent 1Vrms pour l'intensité nominale. Par exemple SCT013 30A/1Vrms. Le raisonnement reste le même que pour une burden externe.

Le conditionnement du signal du CT se fait en 3 étapes :

- Une résistance optionnelle de burden si elle n'est pas intégrée au CT.
- Un pont diviseur constitué d'un potentiomètre de $10K\Omega$ qui permet un facteur d'atténuation réglable de 0 à 50% (il n'est pas raisonnable d'aller au delà de 50%, il vaut mieux dans ce cas là changer de CT pour éviter trop de bruit)
- Un amplificateur de 1.55 qui amplifie le signal et isole la résistance de burden de l'ADC.



2.3.1 CST013

Dans ces conditions, en acceptant une valeur ADC maximale de 2.4V, un CST013 100A/50mA permet de mesurer :

$$2.4 / 1.55 / 2 / 1.44 = 0.55 \text{VRms}$$

soit $0.55 / 1.1 * 100 = 49.7 \text{A}$

Avec 50% d'atténuation on peut utiliser l'intensité nominale de CT : 100A.

Ce raisonnement est valable pour tous les CT à burden intégré et qui génèrent 1VRMS: 50% du courant nominal sans atténuation, et 100% du courant nominal avec 50% d'atténuation.

2.3.2 SCT016

Cas du CT SCT016 50A/50mA avec le burden recommandé de 10Ω . Pour 50A ce CT fournit :

$$0.05 * 10 = 0.5 \text{VRMS} \text{ soit } 1.41 \text{ Vpp}$$

$$1.41 * 1.55 = 2.17 \text{Vpp}$$

Ce CT peut être utilisé dans les conditions optimales sans atténuation. Si le potentiomètre d'atténuation est monté sur support, il peut être enlevé et remplacé par un bout de fil entre les pattes 2 et 3.

Le déphasage par rapport à la tension est d'environ $300\mu\text{s}$ (5.4°).

2.3.3 CT08CL10

Pour mesurer la production photovoltaïque et la puissance routée j'utilise de petits tores à enfiler sur le fil : CT08CL10. Bonne qualité et bon marché (3€ pièce en commandant un lot de 4).

En particulier le déphasage par rapport à la tension est quasi nul. Peut-être faudrait-il utiliser la tension non corrigée en phase pour calculer la puissance mesurée par ces tores ?



<https://www.aliexpress.com/item/1005004284693644.html>

Dans mon cas de petite production (760W max) j'ai fait 5 tours dans le tore pour améliorer la précision de la mesure.

C'est un transformateur 1:2000 tours donc avec une sensibilité de 0.5mA/A. Je l'utilise avec un burden de 47Ω .

$$(760W * 5\text{tours}) / 235V = 16.2 \text{ A donc } 8.1 \text{ mA RMS.}$$

$$47 * 0.0081 * 2 * 1.414 = 1.07 \text{ Vpp}$$

$$\text{Une fois amplifié l'ADC reçoit : } 1.07 * 1.55 = 1.66\text{Vpp} \text{ (soit } 2/3 \text{ de } 2.5\text{V)}$$

Donc utilisation directe sans atténuation.

Technical Data:

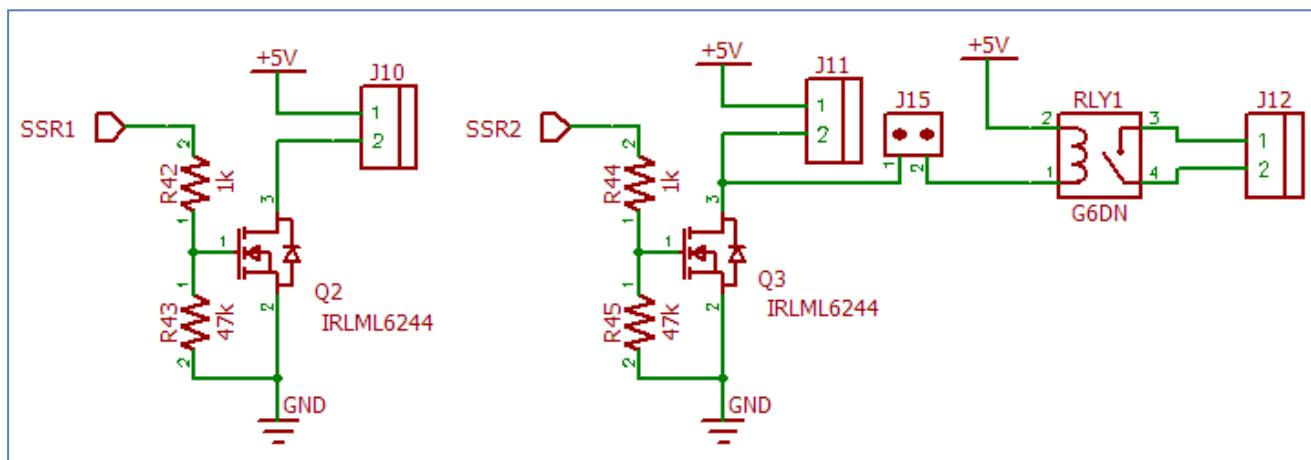
	DL-CT08CL10-2000/1	Unit
Rated input	20	A (AC)
Rated output	10	mA (AC)
Rated sampling load	100	Ω
Rated sampling voltage	1	V (AC)
Turns ratio	2000/1	---
Rated Phase shift	≤ 5 ($RL=0\Omega$)	'
Accuracy	0.1	Level
Linearity	0.1	%
Measuring current	0~60 ($RL \leq 40\Omega$)	A (AC)
Maximum current	0~80 ($RL \leq 10\Omega$)	A (AC)
DCR	99 ± 10	Ω
Weight	11.5	g

2.4 La commande des relais statiques

Le routage se fait par l'intermédiaire d'un relais statique (SSR). Ce relais est externe afin de pouvoir le choisir en fonction des besoins de l'installation.

Les relais sont commandés par les sorties d'un timer, et conditionnées par un MOSFET-N. cela permet de commander les SSR avec 5V, ce qui devrait convenir dans la plupart des cas.

Une des sorties peut être commandée par logiciel et commuter un relais mécanique. Cela nécessite de déplacer un cavalier sur la carte (J15).



2.5 Interface TIC du Linky

Le compteur Linky installé par Enedis a une sortie « Télé Information Consommateur », qui peut être utilisée pour exploiter quelques informations émises par le Linky.

Il ya des sites qui expliquent bien comment faire, tels que :

<http://hallard.me/demystifier-la-teleinfo/>

J'ai testé ce montage, et il a fallu faire des adaptations à cause de mon optocoupleur SFH620 chinois sans marque, et donc de caractéristiques inconnues, et apparemment pas très bonnes.

Pour avoir un fonctionnement stable en mode standard j'ai adapté les valeurs de 2 résistances : $2.2\text{k}\Omega$ dans le primaire, et $3.3\text{k}\Omega$ comme charge du phototransistor. Je pense qu'avec un SFH620A-2 original cela fonctionnerait mieux.

Donc il ne semble pas y avoir de configuration universelle, il faut s'adapter à l'optocoupleur utilisé.

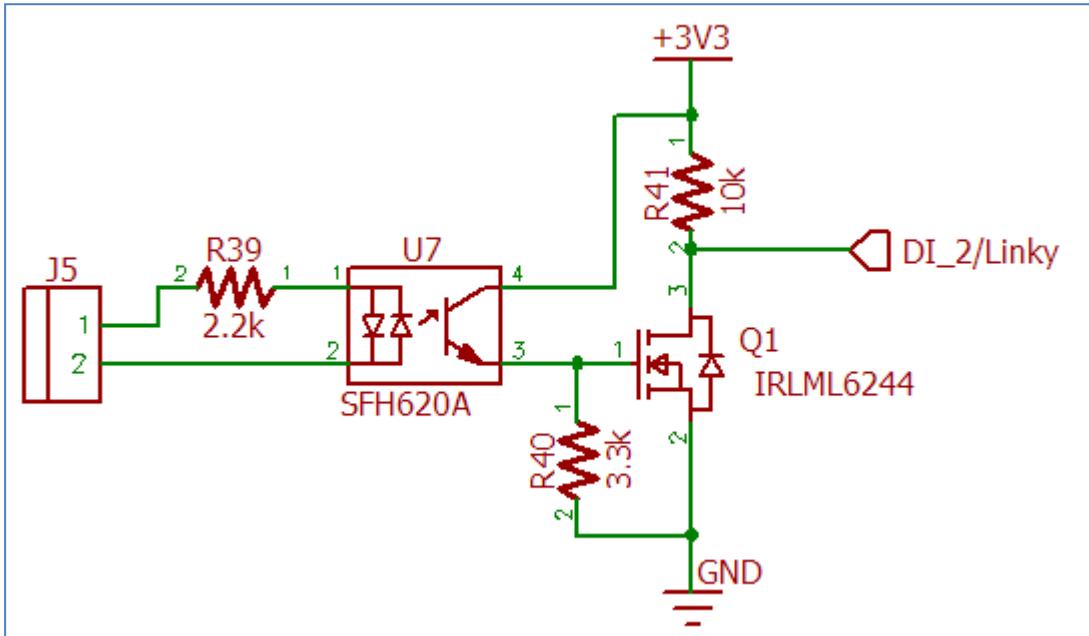
La résistance R39 de $2.2\text{k}\Omega$ permet d'avoir 2 récepteurs sur la ligne (je l'ai testé), tout en respectant les spécifications d'Enedis : charge $> 500\Omega$.

La TIC du linky est très bavarde, mais peu d'informations sont utiles à part :

- La tension secteur
 - La puissance apparente en VA. Utilité douteuse, sauf que si elle passe à 0 c'est qu'il y a injection !
 - Le compteur de Wh facturés

- La date et l'heure

Il manque la puissance réelle instantanée soutirée en W et la puissance injectée en W. Ce sont des informations universellement réclamées, mais Enedis...



2.6 Le connecteur d'extension

Un connecteur IDC de 20 broches est utilisé pour permettre une extension de l'équipement. Il regroupe des broches inutilisées du MCU:

- 3.3V
- 5V
- GND
- 1 bus I2C (ou 2 GPIO)
- 1Bus SPI (commun avec l'écran et la FLASH)
- 1UART (ou 2 GPIO)
- VREF/2
- 1 GPIO
- 1 GPIO / ADC / DAC
- 1GPIO qui est un canal du timer SSR, donc pouvant éventuellement commander un SSR de routage.

2.7 Choix de l'ampli opérationnel

Il n'y a pas si longtemps les amplificateurs opérationnels (AOP) utilisaient des tensions d'alimentation élevées telles que $\pm 15V$, pour qu'ils puissent avoir de bonnes caractéristiques. Par exemple leur technologie ne permettait pas aux signaux traités d'approcher à plus de 1.5 ou 2 volts des tensions d'alimentation. Et pour pouvoir traiter les signaux autour de 0V des alimentations symétriques étaient utilisées.

Depuis les équipements n'utilisent plus qu'une seule tension beaucoup plus faibles, comme 5V. Les AOP ont du évoluer parce que par exemple la marge de 1.5V autour des tensions d'alimentation n'est plus acceptable. Ils sont donc devenus « rail-to-rail », c'est à dire que la plage de tension des signaux en entrée et en sortie peut atteindre les tensions d'alimentation.

Les premières générations d'AOP « rail-to-rail », ont des étages d'entrée constitués de deux paires de transistors : une pour traiter les signaux de bas niveau, et l'autre pour les signaux de tension plus élevée jusqu'à Vcc. Ces deux paires ayant des caractéristiques différentes, il y a donc une zone de transition entre les deux autour de 1.5V sous Vcc (input crossover distortion), où les performances de l'OPA sont dégradées.

Un cas typique de cette génération d'AOP est le OPA2340. Voici une représentation issue de la datasheet :

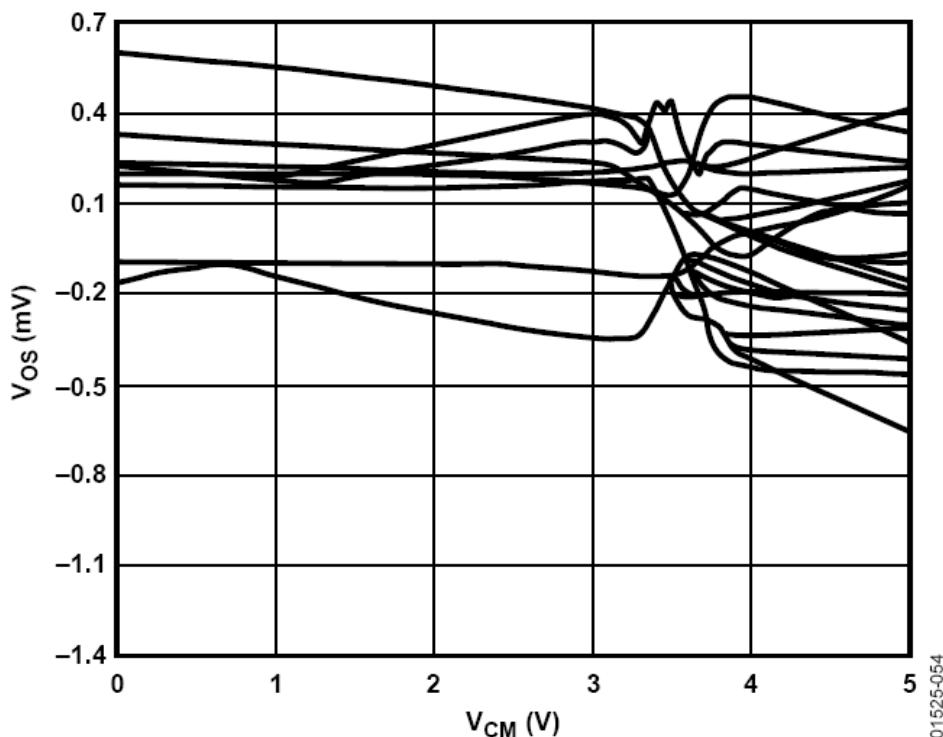


Figure 54. Burr-Brown OPA2340UR Input Offset Voltage vs. Common-Mode Voltage, 24 SOIC Units @ 25°C

On distingue nettement la zone de transition autour de 3.5V où l'offset change de valeur. Ce type d'AOP ne respecte donc ses caractéristiques qu'entre 0V et Vcc-1.5V. Il faut le savoir et utiliser l'AOP dans de bonnes conditions.

L'évolution suivante de l'électronique est d'encore abaisser la tension d'alimentation à 3.3V ou 2.5V. A ce niveau la plage d'utilisation d'AOP tel que l'AOP2340 est de 0V à 1.8V ou 1V ! Ce n'est plus acceptable dans beaucoup de cas.

Il n'est pas pénalisant d'utiliser un AOP avec un offset de plusieurs millivolts, s'il est constant. Cela peut se corriger par matériel ou logiciel avec un étalonnage. Mais un offset qui varie suivant le niveau d'entrée introduit des distorsions qui ne sont pas corrigéable. Dans le cas d'un MCU AVR en 5V et un ADC avec une plage d'entrée < 3.5V, l'OPA2340 est parfaitement adapté. Avec un STM32 en 3.3V il ne l'est plus.

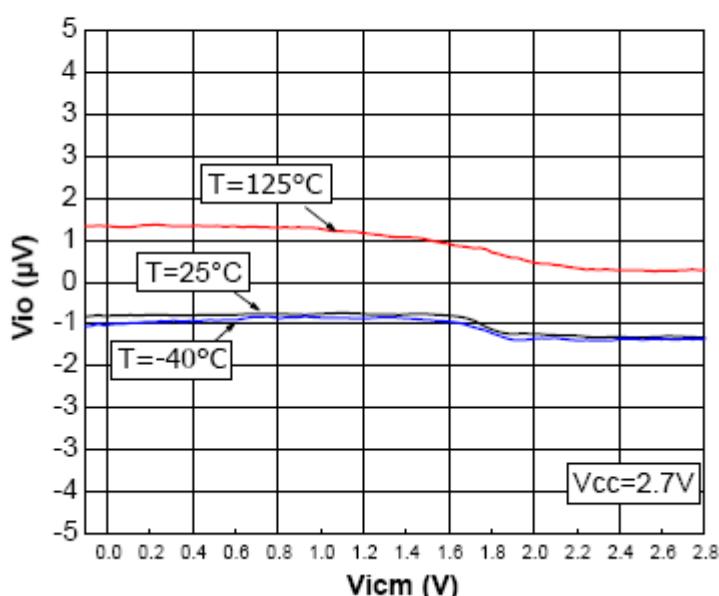
Il faut aussi relativiser : une variation d'offset de 1mV correspond à 1/5 de pas pour un ADC 5V et 10 bits (1024 pas).

Il y a eu plusieurs réponse des concepteurs d'AOP, par exemple :

- Utiliser une pompe de charge interne pour alimenter l'étage d'entrée de l'AOP avec une tension supérieure à l'alimentation et donc pourvoir utiliser une seule paire de transistors ce qui élimine la zone de transition. Cette technique est appelée « Zero-crossover », voir OPA2388 ou ADA4500-2.
- Utiliser une technique « Zero Drift » pour corriger en permanence l'offset, comme le TSZ122 ou le TSZ182 plus rapide chez STMicroelectronics, ou AD8629 chez Analog Devices.

Voici la caractéristique Vio (Chez STM) ou Vos (Chez TI) du TSZ122 :

Figure 10. Input offset voltage vs. input common-mode at $V_{CC} = 2.7 \text{ V}$



On peut voir que la zone de transition est très atténuee, et par ailleurs le Vio est annoncé à 8μV entre -40°C et + 125°C.

Ces AOP ont une plage d'utilisation d'entrée qui couvre entièrement la plage de tension d'alimentation, et souvent un peu plus : 0.1V à 0.3V au dessous de 0V et en dessus de Vcc. La sortie s'approche à quelques dizaines de mV de 0V et de Vcc.

Pour notre application :

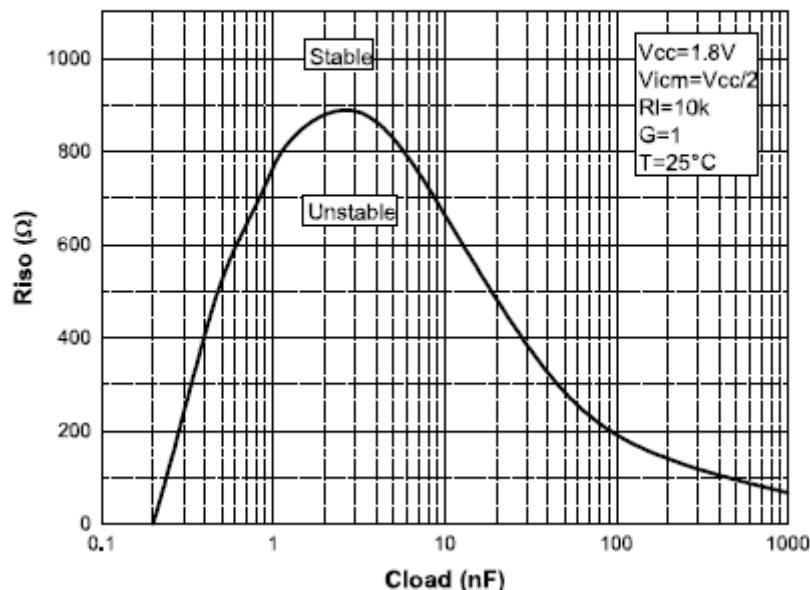
- Alimentation 3.3V. Il ne faut pas risquer d'envoyer plus de 3.3V sur les entrées analogiques du MCU qui ne le supporterait pas.
- Conditionner des signaux de quelques millivolts à 1.6V centrés sur 1.25V, et sortie de 0V à 2.5V. Peut être même 3V dans l'avenir
- Plus l'offset est faible et constant plus fiable est la mesure.

Les AOP Zero Drift sont donc bien adaptés à cette utilisation. J'avais quelques TSZ122 et TSZ182 alors je les ai utilisés. Ils ne sont pas plus chers que des AOP classiques.

Il faut lire attentivement les datasheets pour les utiliser correctement. Par exemple :

- Ils sont « lents » à démarrer. Un AOP classique atteint ses caractéristiques en quelques dizaines de μ s après sa mise sous tension. Ces Zero Drift mettent de 100 à 1000 μ s. Dans notre cas cela n'a pas d'importance.
- Les AOP ont des difficultés avec les charges capacitives et généralement on utilise une résistance d'isolation de 10 à 33Ω entre la sortie et le condensateur. La datasheet du TSZ122 a des graphes pour indiquer la résistance à utiliser en fonction de la tension d'alimentation et la valeur du condensateur. Et donc j'utilise 1 k Ω . C'est la valeur minimale pour le TSZ122, mais elle convient aussi à tout autre AOP.
- La datasheet indique que le TSZ122 a été conçu pour une résistance de feedback d'au moins 10 k Ω . Je n'avais pas lu ça et j'ai utilisé 5.1 k Ω sans problème. Peut-être qu'un jour je ferai des tests avec la valeur préconisée.

Figure 44. Stability criteria with a serial resistor at $V_{DD} = 1.8$ V



3 Conception logicielle

Le principe du routeur est simple : détecter s'il y a surplus de production d'énergie, et router cette énergie vers un consommateur. En regardant les réalisations des autres, j'ai vu qu'il y a beaucoup de façons différentes de réaliser cela.

Mon objectif est de minimiser l'injection et pour cela il faut tenir compte des caractéristiques du Linky (j'habite en France ou tout est plus compliqué qu'ailleurs...). Donc il faut équilibrer la consommation à chaque demi-période du secteur.

Le principe de base du routage est donc le suivant : Pendant chaque demi-période on mesure la puissance utilisée, et on ajuste le routage avec cette valeur pour la demi-période suivante. En supposant que la puissance utilisée varie peu d'une demi-période à l'autre. Un contrôleur Proportionnel/Integral logiciel est utilisé pour ces ajustements.

La mesure de puissance est réalisée en mesurant la tension du secteur et le courant sur le câble d'arrivée au tableau électrique. Ces signaux sont analogiques, on utilise donc des entrés ADC du MCU.

3.1 Timer d'acquisition analogique

L'échantillonnage de la tension et des courants se fait 200 fois par période secteur de 20ms, donc toute les 100 μ s :

Le 1^{er} échantillon d'une période doit être synchronisé avec le début de la période secteur, c'est à dire quand la tension vaut 0 (du moins en théorie...). Donc il est nécessaire de démarrer le timer lors du 0 dans le sens croissant de la tension.

Si on laisse le timer compter librement il y a rapidement une désynchronisation entre le timer et le secteur. Un asservissement PI (Proportionnel/Integral) de la période du timer assure que toute les 200 périodes du timer la tension lue est bien de 0. Si la valeur est éloignée de 0 (>LOCK_THR) un flag de synchronisation hors limite est positionné. Quand la synchronisation est correcte pendant LOCK_COUNT périodes, le flag est effacé.

Ce timer (nommé TIMSYNC dans le logiciel) a une horloge de 16 MHz, et donc sa période peut varier par pas de 0.0625 μ s, ce qui fait un pas de

$$0.0625 \times 200 = 12.5\mu\text{s}$$

par période secteur. L'asservissement sur le début de période secteur est donc relativement fin.

Ce timer déclenche la séquence d'acquisition de l'ADC toute les 100 μ s, et il peut être observé à l'oscilloscope sur la patte B1/TP8.

3.2 Acquisition analogique

Le MCU dispose de nombreuses broches d'entrées analogiques, mais d'un seul ADC, autrement dit le résultat de la conversion d'une voie se trouve dans un seul registre, et la conversion suivante écrase le résultat précédent. Pour convertir plusieurs voies consécutives il y a 2 méthodes :

- Déclencher les conversions une par une et lire le résultat à la fin de chaque conversion avant de déclencher la suivante. C'est coûteux en temps de traitement. Mais c'est très facile si on a qu'une seule voie à convertir.

- Utiliser le mode séquentiel : on programme une liste de voies à convertir et l'ADC enchaîne automatiquement les conversions. Pour ne pas perdre de donnée, on utilise un DMA, qui exécute automatiquement le transfert d'une donnée à la fin de chaque conversion.

C'est le second mode qui est utilisé. Lorsque le DMA a transmis la donnée de la dernière voie à convertir il génère une interruption pour prévenir le logiciel que les données sont prêtes à être utilisées.

Une fois le timer d'acquisition synchronisé et l'ADC et le DMA démarrés, l'acquisition analogique de toutes les voies se fait entièrement par matériel, le logiciel est simplement prévenu périodiquement par une interruption qu'un tableau de valeurs analogiques est prêt.

L'interruption ne fait que réveiller la tâche de traitement, ce qui peut être aussi simple que (en pratique il y a un traitement des éventuelles erreurs) :

```
void DMA1_Channel1_IRQHandler (void)
{
    dma_t          * pDma = (dma_t *) DMA1 ;
    uint32_t       isr ;
    uint32_t       mask ;

    aaIntEnter () ;

    isr = pDma->ISR ; // Get interrupt status register only once

    // Check transfer complete
    mask = 1u << DMA_ISR_TCIF1_Pos ;
    if ((isr & mask) != 0u)
    {
        // Clear flag
        pDma->IFCR = mask ;

        // Signal ADC processing task
        aaSignalSend (meterTaskId, 1u) ;
    }
    aaIntExit () ;
}
```

La tâche meterTask qui attendait ce signal et qui a la plus haute priorité commence aussitôt le traitement.

3.3 Commande SSR

Le routage de puissance vers une charge (chauffe eau par exemple) utilise un relais statique (SSR : Solid State Relay). Ce relais à la particularité de n'avoir aucun composant mécanique, donc d'être rapide. Cela permet de fermer le relais au moment choisi dans la demi-période secteur, et le relais s'ouvre automatiquement lorsque la tension secteur passe à 0.

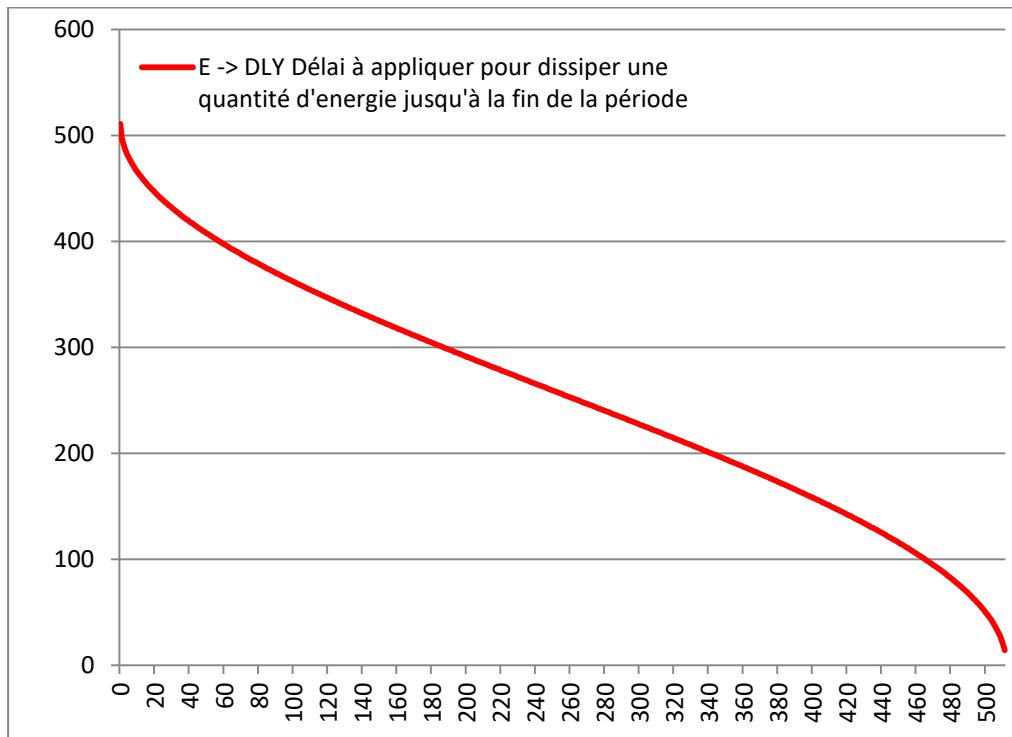
L'énergie routée dépend donc du délai entre le début de la demi période et la fermeture du délai : plus le délai est faible plus l'énergie routée et importante. Il est donc nécessaire d'utiliser une table qui donne le délai à utiliser pour une puissance donnée.

3.3.1 Déterminer la formule théorique

Le courant est sinusoïdal. L'énergie est l'intégration de ce signal donc un Cosinus.

Ce cosinus donne l'énergie accumulée depuis le début de la demi-alternance. Ce qui nous intéresse c'est la réciproque : l'énergie entre le délai et la fin de la demi-alternance : c'est un ArcCosinus.

Si on construit une table nommée `p2Delay` pour une puissance normalisée entre 0 et 511 et un délai entre 0 et 511 on obtient :



En X l'énergie, en Y le délai à appliquer pour router cette énergie.

3.3.2 Application

La puissance mesurée par le CT 1 est utilisée pour déterminer la puissance à router : Si la puissance est négative, il y a injection sur le réseau, il faut donc router cette puissance excédentaire.

La puissance maximale de la charge `powerDiverterMax` (par exemple 3000W) est connue et correspond à 511 dans le graphe. Le délai à appliquer pour une puissance `Pdiv` à router est :

$$\text{delai} = \text{p2Delay} [\text{Pdiv} / (\text{powerDiverterMax} / 512)]$$

ou :

$$\text{delai} = \text{p2Delay} [(\text{Pdiv} * 512) / \text{powerDiverterMax}]$$

Si la puissance `Pdiv` est \geq à `powerDiverterMax` alors le délai est de 0, et il faut déclencher le relais dès le début de la demi-alternance.

Inversement si `Pdiv` vaut 0, on ne déclenche pas le relais.

Le timer qui génère le délai puis l'impulsion de déclenchement du SSR est configuré de façon à pouvoir utiliser directement la valeur issue de la table (entre 0 et 511) dans son registre de délai.

En pratique le délai minimal est de 3 (~60µs), et le délai maximal de 502 pour avoir une largeur correcte de l'impulsion de déclenchement.

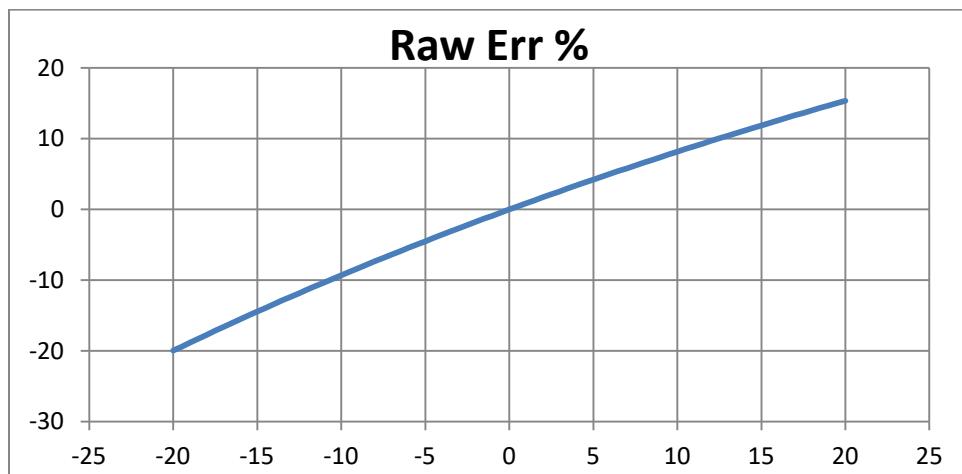
Un autre paramètre à prendre en compte est le type de charge utilisé, voir §14 [Conversion puissance – délai](#).

3.3.3 Correction de powerDiverterMax

Pour calculer la puissance à router il faut tenir compte de la puissance maximale de la résistance vers laquelle est routée l'énergie, elle est appelée `powerDiverterMax` dans le logiciel. Par convention elle est exprimée en W pour une tension de 230Vrms.

Le problème est que la tension du secteur n'est pas constante, donc la puissance dissipée par cette résistance ne l'est pas non plus, et par conséquent le calcul de la puissance à router est faussé.

Voici l'erreur sur la puissance de la résistance si on la considère constante en fonction de U. Le 0 en abscisse correspond à 230V, et à 10 (soit 240V) l'erreur est de 8.2% :



Il est donc introduit une correction de `powerDiverterMax` en fonction de la tension effective au moment du calcul.

On utilise les équations suivantes :

$$U = R \cdot I \quad \text{donc} \quad I = U/R \quad \text{et} \quad P = U \cdot I \quad \Rightarrow \quad P = U^2/R$$

Si on considère R constante, la puissance ne varie qu'en fonction de U

Remarque : La variation de puissance de la résistance en fonction de sa température n'est pas corrigée et est négligée.

U est mesurée chaque seconde et est appelée Vrms, il faut donc connaître R.

On sait que dans la configuration Pmax est mesurées à 230V. On peut donc en déduire :

$$R_{div} = 230^2 / P_{max}$$

Rdiv est calculée une fois au démarrage de l'application et est utilisée chaque seconde pour en déduire la puissance de la charge à ce moment là :

$$\text{powerDiverterMax} = (\text{Vrms} * \text{Vrms}) / \text{Rdiv}$$

Cette valeur de puissance est utilisée pendant la seconde qui suit par le routage à chaque $\frac{1}{2}$ période secteur, en considérant que la tension variera peu pendant ce temps là.

Pour que tout cela soit valable, il reste à connaître la puissance réelle de la résistance à 230V !

Si à une tension V_{RMS} de 238V on mesure une puissance de P_{max} Watts, on peut:

- 1) Déterminer $R = V_{\text{RMS}}^2 / P_{\text{max}} = 238^2 / P_{\text{max}}$
- 2) Utiliser R pour en déduire: $P_{230} = 230^2 / R$

En combinant les 2 équations :

$$P_{230} = (230^2 / V_{\text{RMS}}^2) * P_{\text{max}}$$

3.4 Mesure de température

La mesure de température est réalisée avec des capteurs DS18B20.

3.4.1 Interface matérielle

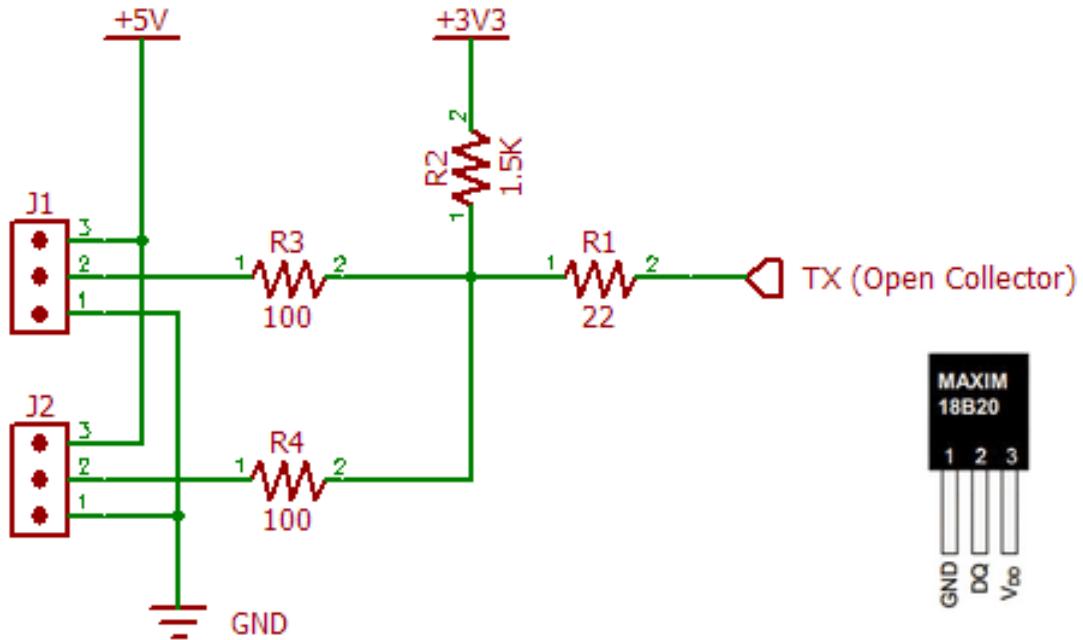
La datasheet du DS18B20a été lu rapidement lors de la réalisation du circuit imprimé. Grossièrement il suffit d'une GPIO pour gérer le protocole OneWire. Le circuit a donc été réalisé en dédiant 2 GPIO à la mesure de température sur 2 lignes de capteurs.

Lors de la réalisation du logiciel, il est apparu qu'il était nécessaire de respecter des timings de l'ordre de la microseconde, ce qui n'est pas compatible avec l'utilisation d'un noyau temps réel qui utilise des sections critiques et des interruptions. La solution est d'utiliser un UART pour gérer ces capteurs (exit les GPIO qui bien sûr n'étaient pas reliées à un UART).

Les DS18B20 utilisent donc le connecteur d'extension qui dispose d'un UART.

Il s'avère que l'interface matérielle de ces capteurs est très dépendante de la topologie du câblage, du type de câble employé et des longueurs utilisées. La littérature donne beaucoup de conseils, plus ou moins contradictoires.

Après quelques essais l'interface choisie est la suivante :



Les valeurs de toutes ces résistances sont à adapter au câblage choisi. Mes tests ont été faits avec une longueur de câble téléphonique de 11m et 5m sur J1 et J2.

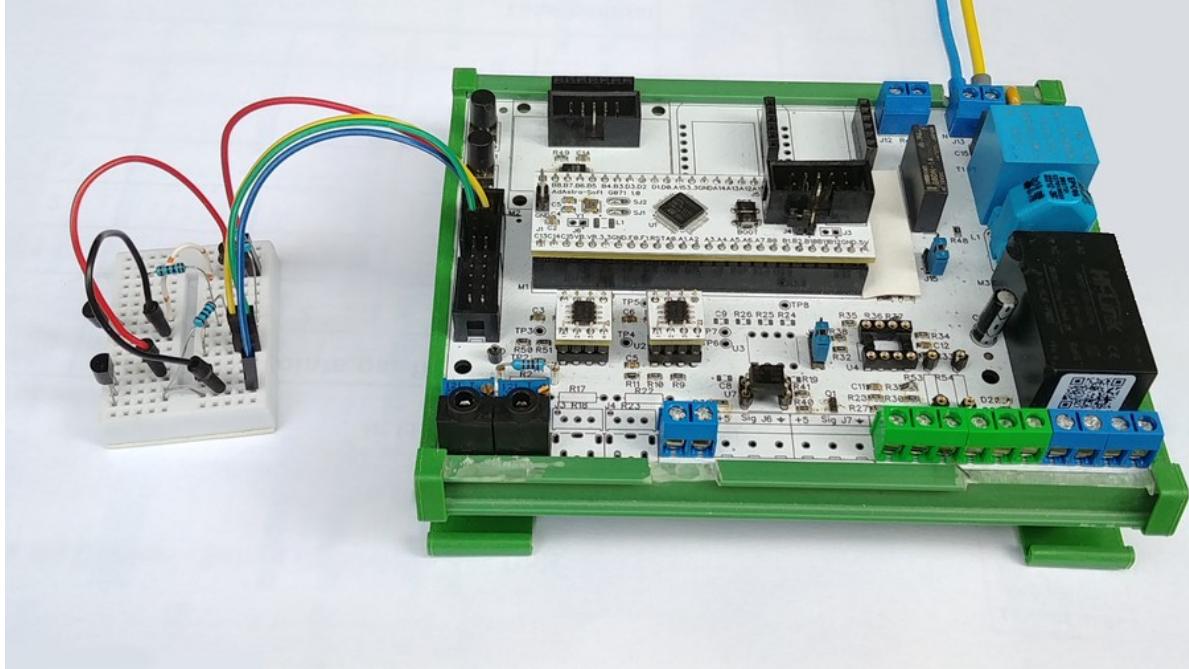
Le port TX doit être configuré en collecteur ouvert, avec la vitesse (slew rate) la plus lente. Même dans ce cas le front de descente du signal est de 35 ns, ce qui est rapide et induit des undershoots au bout du câble. R1 permet de calmer ce front et de diminuer les undershoots.

Le port série a un mode « half duplex » dans lequel RX et TX sont connectés intérieurement. En utilisant ce mode seule la broche TX est utilisée, et la broche RX peut être utilisée pour autre chose.

Le choix de 2 lignes permet de positionner des capteurs à des endroits différents. Il s'agit donc d'une topologie en étoile, ce qui n'est normalement pas recommandé. Mais les résistances R3 et R4 permettent de désolidariser les 2 lignes (Si R3 et R4 sont utilisées, R1 peut être supprimée).

Le pull-up R2 de 1.5kΩ permet d'avoir des fronts suffisamment raides au bout des câbles. Le pull-up est relié au 3.3V qui est la tension d'alimentation du MCU.

Les DS18B20 sont alimentés par le 5V pour avoir une tension suffisante au bout de longs câbles.



Montage pour la mise au point de l'interface et du logiciel DS18B20

3.4.2 Logiciel

Les opérations à réaliser avec les DS18B20 sont :

- Rechercher les capteurs
- Démarrer la conversion sur tous les capteurs en même temps
- Attendre que tous les capteurs aient fini leur conversion
- Lire une par une les températures des capteurs

La première étape est faite une seule fois lors de la configuration. Ensuite la capture des températures est cyclique et se fait environ 1 fois par seconde.

Les DS18B20 sont identifiables uniquement par leur identifiant interne, récupéré lors de la recherche des capteurs. Les capteurs sont donc découverts dans l'ordre de leurs identifiants. Si un capteur est remplacé (suite à une panne par exemple), l'ordre des capteurs dans la liste n'est plus le même. Et par exemple le capteur n°1 dans la liste ne correspond plus au capteur physique qui était le n°1 avant la modification.

Pour résoudre ce problème le logiciel permet d'associer un numéro de rang à chaque capteur. De cette façon l'utilisateur peut associer un rang fixe à un capteur physique, mais il faut refaire cette association à chaque changement de configuration.

Sur la page HTTP de AASun les températures sont affichées dans l'ordre des rangs, donc indépendamment des identifiant des capteurs.

Le logiciel permet de gérer jusqu'à 4 DS18B20, avec une résolution de 12bits qui est celle par défaut du capteur (résolution de 0.0625°C !). Mais les températures sont stockées au 1/16 de °C et affichées au 1/10 de °C.

3.4.3 Configuration

La configuration des DS18B20 se fait en deux étapes :

- Rechercher les capteurs pour en constituer la liste
- Affectation de son rang à chaque capteur

La commande « tss » (Temperature Sensor Search) recherche les capteurs et construit la liste des présents. Le rang par défaut de chaque capteur est son index dans la liste.

La commande « dts » (Display Temperature Sensor) permet de voir la liste des capteurs de température. Les capteurs sont affichés dans l'ordre de la liste constituée lors de leur découverte.

La commande « tsr » (Temperature Sensor Rank) permet d'affecter son rang à chaque capteur. Le rang va de 0 à n-1, n étant le nombre de capteurs dans la liste. Les rangs sont donnés dans l'ordre de la liste. Syntaxe :

```
tsr r0 r1 r2 r3
```

La commande « tsc » (Temperature Sensor Check) permet de vérifier si les capteurs de la configuration sont physiquement présents. Si un capteur n'est pas présent il est qualifié d'absent (A dans la liste) et ne sera pas interrogé. Cette commande est effectuée automatiquement à chaque démarrage de AASun.

Cette liste de capteurs fait partie de la configuration, elle est donc mémorisée en flash par la commande cwr.

Exemple de configuration:

```
tss
Device 000 Type 0x28 ID 3CE1E380003C CRC 0xD8
Device 001 Type 0x28 ID 3CE1E38181B0 CRC 0xC6
TS Search done

dts
0 1 P Type 0x28 ID 3CE1E380003C CRC 0xD8    23.5
1 2 P Type 0x28 ID 3CE1E38181B0 CRC 0xC6    23.7
2 3 A Type 0x00 ID 000000000000 CRC 0x00    0.0
3 4 A Type 0x00 ID 000000000000 CRC 0x00    0.0

tsr 2 1
dts
0 2 P Type 0x28 ID 3CE1E380003C CRC 0xD8    23.5
1 1 P Type 0x28 ID 3CE1E38181B0 CRC 0xC6    23.6
2 3 A Type 0x00 ID 000000000000 CRC 0x00    0.0
3 4 A Type 0x00 ID 000000000000 CRC 0x00    0.0

cwr
Ok
```

Dans la capture d'écran ci dessus :

- La commande tss a trouvé 2 capteurs

- La commande `dts` qui suit affiche 2 capteurs présents (P) et 2 capteurs absents (A). La 2^{ème} colonne est celle des rangs, qui sont dans l'ordre de la liste. En fin de ligne la température en °C.
- La commande `tsr` permet d'affecter les rangs des 2 premiers capteurs
- La commande `dts` qui suit affiche bien les nouveaux rangs.
- La commande `cwr` enregistre cette configuration en flash.

Exemple de vérification après avoir physiquement enlevé le 2^{ème} capteur :

```
tsc
ts 0
TS Check done
dts
0 2 P Type 0x28 ID 3CE1E380003C CRC 0xD8    23.4
1 1 A Type 0x28 ID 3CE1E38181B0 CRC 0xC6    200.0
2 0 A Type 0x00 ID 0000000000000000 CRC 0x00    0.0
3 0 A Type 0x00 ID 0000000000000000 CRC 0x00    0.0
```

On voit que le capteur de rang 1 est toujours dans la liste, mais est qualifié d'absent, et affiche une température invalide de 200°C.

3.5 Date et heure

Pour le routeur connaître la date et l'heure sont indispensable pour gérer les historiques d'énergie et de puissance. Cela est important aussi pour les forçages utilisant un jour ou une heure donnée.

Il y a quatre sources possibles pour la date et l'heure :

- Manuelle. Ces informations peuvent être entrées manuellement par la console avec la commande « time »
- Réseau filaire (via la box de la maison)
- WIFI (via la box de la maison)
- Linky.

Lorsque le routeur a besoin de l'heure (à la mise sous tension par exemple), il interroge les sources disponibles à tour de rôle, jusqu'à ce que l'une d'elle réponde une date valide (> 1970 !). A ce moment là les historiques sont configurés et démarrés.

Chaque jour à 01:07:00 le routeur cherche à mettre à jour date et heure pour annuler une éventuelle dérive de l'horloge locale.

Les passages heure d'été / heure d'hiver sont gérés automatiquement.

Le module WIFI en mode station utilise SNTP pour se remettre à l'heure toute les 60 minutes.

3.6 Architecture du logiciel

Le logiciel a des actions à réaliser à différentes échéances temporelles :

- Toutes les 100 µs

- Toutes les secondes
- et d'autres sans échéances particulières.

Le MCU a suffisamment de ressources pour utiliser un noyau temps réel, j'utilise donc le noyau maison AA-RTK (AdAstra Real Time Kernel).

Le principe de base assuré par un noyau temps réel est qu'il active toujours parmi les tâches prêtes à s'exécuter celle qui a la plus haute priorité.

Le logiciel utilise donc 3 tâches de priorité décroissantes, dont les principales actions sont :

3.6.1 meterTask() : la priorité haute

Cette tâche est activée toutes les 100µs par l'interruption de fin d'acquisition analogique (en réalité la fin du DMA), et réalise :

- Le traitement des échantillons analogiques : calcul de Volt, I1, I2, etc.
- L'asservissement du timer d'échantillonnage sur le 0 de la tension secteur toute les 20 ms.
- L'accumulation des données pour le routage puis l'asservissement du routage toute les 10 ms.
- L'accumulation des données pendant 1s puis signalement de la disponibilité de ces données à la tâche AASun()
- La gestion de l'anti rebond des boutons (toute les 20 ms).
- La gestion de l'anti rebond des entrées de comptage (toute les 10 ms).

Il est pratique de faire la gestion d'anti rebond et le comptage d'impulsions dans cette tâche, car son timing est très régulier puisque c'est la tâche de plus haute priorité : elle ne peut pas être interrompue par une autre tâche. De cette façon on ne devrait pas louper d'impulsion de comptage même si elles ne durent que 30ms.

3.6.2 AASun() : la priorité moyenne

C'est la première tâche exécutée, elle assure la plus grande partie des initialisations, puis elle crée les deux autres tâches.

Cette tâche assure les traitements qui doivent être faits toutes les secondes, en utilisant les données fournies par meterTask() :

- Calcul de V_{RMS} , des puissances (réelles et apparentes), des cos Phi.
- Cumul des énergies.
- Récupération des compteurs d'impulsion.
- Mise à jour de la page sur l'afficheur OLED.

Toute les 15mn, elle ajoute une entrée au tableau de l'historique des puissances du jour, tableau qui est affiché sous forme de courbes par une page web.

Toute les 2h elle enregistre une sauvegarde des compteurs journaliers (pour ne pas les perdre lors d'un reset par exemple).

Quelques secondes après minuit la tâche réalise les actions suivantes :

- Mémorise en flash les compteurs d'énergies journaliers qui constituent les compteurs historiques.
- Mémorise en flash le tableau des puissances (tableau historique).

Et à 01:07:00 elle démarre l'acquisition de la date courante par DNS WIFI ou Linky suivant les disponibilités. Cela permet d'annuler quotidiennement la dérive du compteur de datation local.

Cette tâche assure la gestion des températures. Le principe est de demander une action concernant les températures (découverte, vérification de présence, acquisition) à la tâche de priorité basse lowProcessesTask() puis de tester chaque seconde si cette action est terminée. Une fois l'action terminée et traitée on peut en lancer une autre.

Les commandes interactives de l'utilisateur provenant de la console (série ou Telnet) sont aussi gérées par cette tâche. Elle reçoit les caractères, et une fois le '\n' reçu, la commande est complète et elle est exécutée. Il faut que l'exécution de ces commandes prenne moins de 1 seconde.

3.6.3 lowProcessesTask () : la priorité basse

Cette tâche réalise toutes les actions longues ou annexes que les 2 tâches précédentes ne peuvent pas réaliser. Dans un premier temps elle réalise des initialisations :

- Initialisation de l'interface réseau filaire
- Initialisation du serveur telnet filaire
- Initialisation du serveur HTTP
- Initialisation des librairies SNTP et DNS
- Initialisation de la communication par liaison série avec le module WIFI

Dans un second temps elle réalise les actions cycliques nécessaires pour :

- Le serveur Telnet filaire
- Répondre aux requêtes HTTP
- Les requêtes SNTP et DNS
- La gestion des capteurs de température DS18B20
- Répondre aux requêtes du module WIFI
- Teste et signale la perte de connexion du réseau filaire ou du module WIFI

3.6.4 Les interruptions

Les interruptions ont une priorité encore plus haute que les tâches : toutes les interruptions interrompent la tâche en cours. Il faut donc respecter un principe simple : les handlers d'interruptions doivent être courts : s'exécuter le plus rapidement possible.

L'application n'utilise qu'une seule interruption : celle de fin de DMA d'acquisition analogique, et son handler est très court.

Mais le système utilise aussi des interruptions avec des handlers aussi rapides que possible :

- Le tic système à la fréquence de 1 KHz : réveil des tâches à l'issue d'un délai
- Interruptions des voies RX et TX de la liaison série de la console : à 57600 bauds cela génère une interruption toute les 174µs.
- Interruption de la voie RX du Linky à 9600 bauds : une interruption toute les 1041µs.

Il est donc sûr que la tâche meterTask() (avec la priorité la plus haute) peut être fréquemment interrompue par des interruptions !

3.6.5 L'afficheur OLED

Le routeur dispose d'un afficheur OLED et de deux boutons pour faire défiler les pages.

L'afficheur est géré par SPI (plus facile à gérer et plus rapide qu'en I2C) et sa présence n'est pas obligatoire : il n'est utilisé qu'en écriture et son absence n'est pas perçue par le logiciel.

D'un point de vue technique 2 polices de caractères sont gérées et permettent d'obtenir 8 ou 4 lignes. Pour une meilleure visibilité la grosse police est utilisée, donc 4 lignes.

Deux types d'afficheurs sont supportés :

- Afficheur de 0.96" avec un contrôleur SSD-1306.
- Afficheur de 1.3" avec un contrôleur SH-1106.

Attention : il faut prendre le modèle SPI avec 7 broches, et la broche de gauche doit être GND. On trouve des afficheurs avec GND et VCC inversés.

La sélection du modèle d'afficheur se fait lors de la configuration avec la commande « ccpy ».

La configuration par défaut n'indique pas d'afficheur (valeur 0). Il faut donc passer par la configuration pour indiquer l'afficheur à utiliser, sauver la configuration, puis relancer le routeur (reset). Sur la console pour choisir le 1.3" cela donne :

```
ccpy 1
 cwd
 reset
```

Une douzaines de pages sont disponibles, et il est facile d'en ajouter, de les modifier ou d'en modifier l'ordre.

Les pages que j'utilise :

- Logo (un peu de pub et c'est joli au démarrage !)

- Page générale avec

- o la version du logiciel
 - o l'adresse IP de base
 - o le jour et l'heure
 - o la tension secteur.



- Puissance instantanée : importée (soutirée), exportée (injectée), routée estimée, CT2
- Energie totale : importée (soutirée), exportée (injectée), routée, CT2, CT3, CT4, Pulse1, Pulse2.
- Etat du routage et estimation de la puissance instantanée routée
- 1 page pour chaque CT :
 - o Intensité
 - o Puissance réelle
 - o Puissance active
 - o Cosinus Phi
- Page Linky (mode standard)
 - o Voltage
 - o Base
 - o VA
- Page pour afficher les températures
- Pages pour afficher les pics min/max de chaque voie analogique.
- Page d'affichage des bits d'état et d'erreurs du routeur.

Toutes ces informations permettent de consulter l'état du routeur et de le configurer confortablement.

3.6.6 La LED D1

Le clignotement de D1 donne une idée du fonctionnement de l'application :

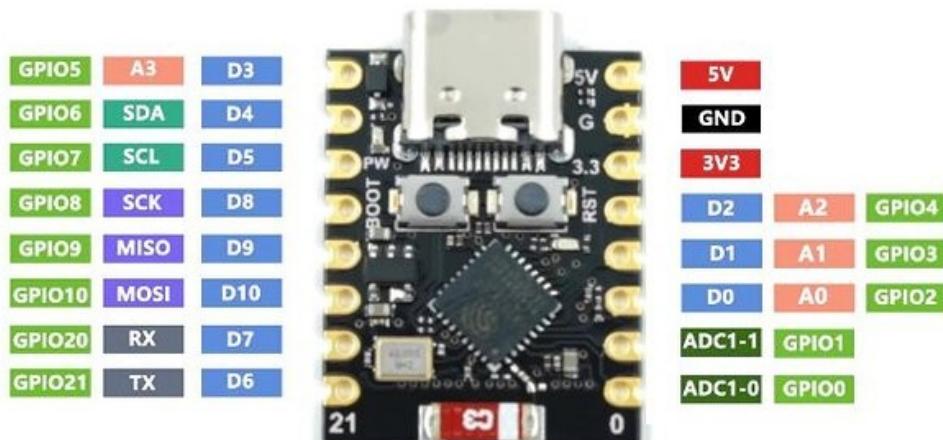
- Elle est allumée dès que l'application est démarrée et le reste pendant toute la phase d'initialisation.
- Pendant la phase de synchronisation elle clignote à 50 HZ (baisse de luminosité).
- Elle est allumée par la tâche meterTask() quant 1s de données sont accumulées et transmises à la tâche AASun().
- Elle est éteinte par la tâche AASun() quand elle a fini de traiter les données.

Si D1 clignote à la fréquence de 1s c'est que les 2 tâches sont « vivantes ». La durée d'allumage indique le temps de traitement de chaque seconde par la tâche AASun().

4 WIFI

4.1 Le module

Le module WIFI choisi est le « ESP32-C3 Super Mini ». L'ESP32-C3 est considéré comme le successeur de l'ESP8266 bien connu.



Principales caractéristiques du module :

- 13 GPIO disponibles, qui peuvent être affectées par programmation à presque n'importe quel périphérique. L'affectation des GPIO dans la figure ci-dessus est une suggestion.
- Seules GPIO2, GPIO8 et GPIO9 ont une fonction particulière au boot. Mais ces broches sont correctement conditionnées sur le module et peuvent quand même être utilisées par une application :
 - GPIO2 pull up, peut être utilisée comme une entrée
 - GPIO8 pull up, LED bleue du module
 - GPIO9 pull up, bouton BOOT.
- 3 liaisons séries.
- 2 SPI (1 est réservé pour la flash interne).
- 2 ADC (1 est réservé au WIFI)
- 4 Mo de flash
- Interface USB spécialisée : utilisable seulement pour la programmation de la flash, et comme port série virtuel pour une console. Ce qui est très pratique pour le développement : pas besoin de convertisseur USB/UART externe, et cela réduit la taille du module.

Il y a une contrainte concernant l'alimentation du module. La broche 5V est directement reliée au VBUS de l'USB. Cette broche est donc une sortie du module quand l'USB est branché.

Plus généralement il y a 3 modes pour alimenter le module et ils sont exclusifs :

- Le connecteur USB
- La broche 5V
- La broche 3.3V

4.2 Implémentation

Le module WIFI est considéré comme une interface par le routeur : aucun traitement ne lui est sous traité.

Le routeur communique avec l'interface WIFI par une liaison série à 230.4 k bauds. La synchronisation des 2 parties est automatique, et l'absence temporaire ou permanente de l'interface WIFI ne perturbe pas le fonctionnement du routeur.

L'interface WIFI donne accès au serveur HTTP de AASun et a une liaison Telnet donnant accès à toutes les commandes de la console.

Le module permet deux modes WIFI : « Station » dit STA et « Access Point » dit AP. La sélection se fait à l'aide d'un cavalier sur GPIO0:

STA : cavalier absent

AP : cavalier en place

4.2.1 Le mode « Access Point »

Le mode AP permet au module WIFI de créer un réseau particulier auquel d'autres équipements (comme un téléphone) peuvent se connecter. Les informations de connexion en mode AP :

SSID	AASun
Mot de passe	AASun***
Adresse IP	192.168.4.1

Le mode AP est utile pour les situations suivantes :

- S'il n'y a pas de box Ethernet dans le logement, le mode AP permet de se connecter au routeur et de le configurer. D'autre part le serveur HTTP est disponible dans ce mode et permet de consulter facilement les données du routeur. Cependant la page d'affichage des courbes n'est pas disponible : La librairie graphique utilisée fait ~1 Mo, et ne loge pas dans la flash du module.
- Pour se connecter en mode STA le module a besoin de connaître le SSID et le mot de passe de la box ou du routeur WIFI auquel il va se connecter. Ces informations d'identifications doivent être au préalable fournies au module WIFI qui les stocke dans sa flash. Pour cela :

Se connecter à AASun en mode AP

Puis se connecter au serveur HTTP en avec l'URL : 192.168.4.1/id

Puis entrer le SSID et le mot de passe requis.

Il est aussi possible de personnaliser le mot de passe du mode AP.

Dans le mode AP le WIFI n'a pas accès à Internet, il ne peut donc pas utiliser des services tels que DNS ou SNTP.

4.2.2 Le mode « Station »

Le mode STA est le plus utilisé et donne accès à Internet par l'intermédiaire de la box ou du routeur Internet du logement. L'interface WIFI peut donc servir de source de datation à AASun.

AASun permet d'utiliser simultanément les interfaces WIFI et réseau filaire, il faut donc attribuer deux adresses IP à AAsun. Par convention l'adresse IP fournie par la configuration est l'adresse IP attribuée au réseau filaire, et l'adresse WIFI est cette adresse incrémentée de 1. Par exemple :

192.168.1.130	Adresse du réseau filaire attribuée lors de la configuration
192.168.1.131	Adresse IP de l'interface WIFI en mode STA

Ce sont des adresses IP fixes.

Pour donner les informations d'identification du mode STA (SSID et mot de passe) il faut passer par le mode AP, puis par le serveur HTTP ou par les commandes disponibles sur la console série/USB (commande ?).

4.2.3 Alimentation du module

Le module WIFI est alimenté à partir du 5V

Pour le développement ou la mise à jour du logiciel il est utile de pouvoir connecter l'USB du module alors qu'il est en place sur le routeur. Pour cela le routeur dispose d'un cavalier qui débranche l'alimentation 5V venant du routeur.

Pour brancher le câble USB sur le module WIFI en place sur le routeur, il est impératif d'enlever le cavalier d'alimentation 5V du module au préalable.

La mise à jour du logiciel du module WIFI utilise l'USB. L'OTA n'est pas implémenté.

4.3 Un peu de technique

La communication entre le routeur et l'interface WIFI est surveillée, et en cas de désynchronisation ou de timeout, une procédure de resynchronisation est lancée.

Cela permet par exemple de faire un reset du routeur ou de l'interface WIFI sans perturber l'autre partie.

Les pages du serveur HTTP sont placées dans la flash de AASun afin de pouvoir être utilisées avec l'interface réseau filaire. Pour accélérer l'affichage des pages en WIFI en leur évitant de passer par la liaison série entre le routeur AASun et l'interface WIFI, l'interface WIFI fait une copie du système de fichiers dans sa propre flash.

Au boot l'interface WIFI compare le CRC de son système de fichier avec celui du routeur, et s'ils sont différents il demande le système de fichier au routeur.

Les requêtes GET et POST CGI sont intégralement passées au routeur qui renvoie la réponse. L'interface WIFI n'a aucune connaissance des données du routeur.

De même pour l'interface Telnet. L'interface WIFI s'occupe de la connexion et de la déconnexion, mais ne fait que transférer le trafic à travers la liaison série. Ensuite le routeur traite les commandes de la même façon que le Telnet filaire ou la console manuelle.

Donc pour une mise à jour de l'interface WIFI il n'y a que l'application à mettre à flasher. Les identifiants WIFI et le système de fichier HTTP sont conservés.

Cela marche assez bien, mais parfois il semble qu'une requête se perde dans le serveur HTTP de l'ESP, et il faut recharger la page.

Si le module est placé directement sur le circuit du routeur, la connexion WIFI échoue. Il m'a fallu éloigner le module de 3cm du circuit. A investiguer...

4.4 Développement et téléversement

L'application du module WIFI est développée avec l'environnement Espressif-IDE, qui est un environnement pratique sous Eclipse : <https://dl.espressif.com/dl/esp-idf/>.

Le dossier du projet est `http`. Il est à copier dans le workspace Eclipse créé au lancement de l'IDE.

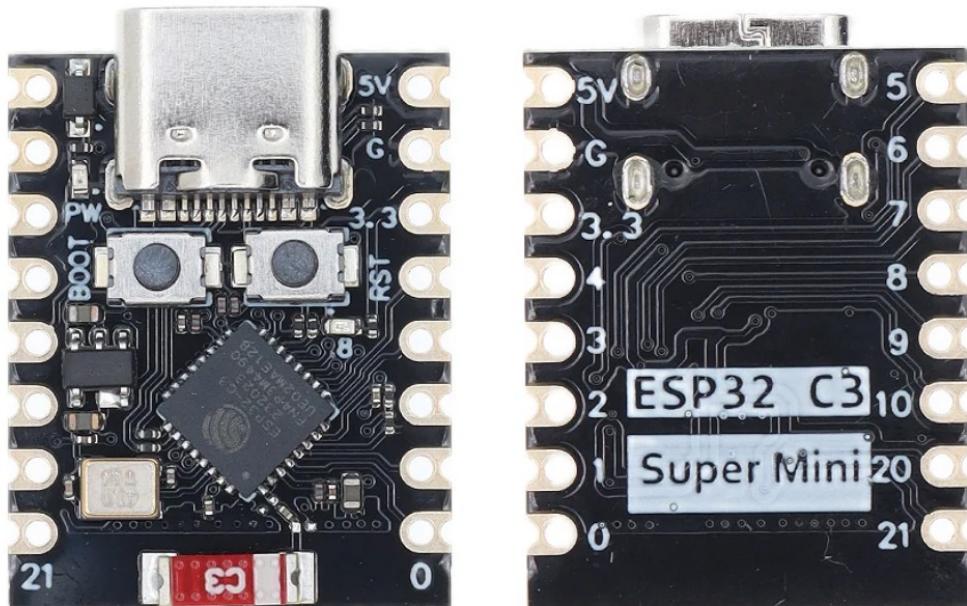
L'environnement permet d'éditer et de compiler le projet. Ensuite il permet de le téléverser dans la cible et de le débugger.

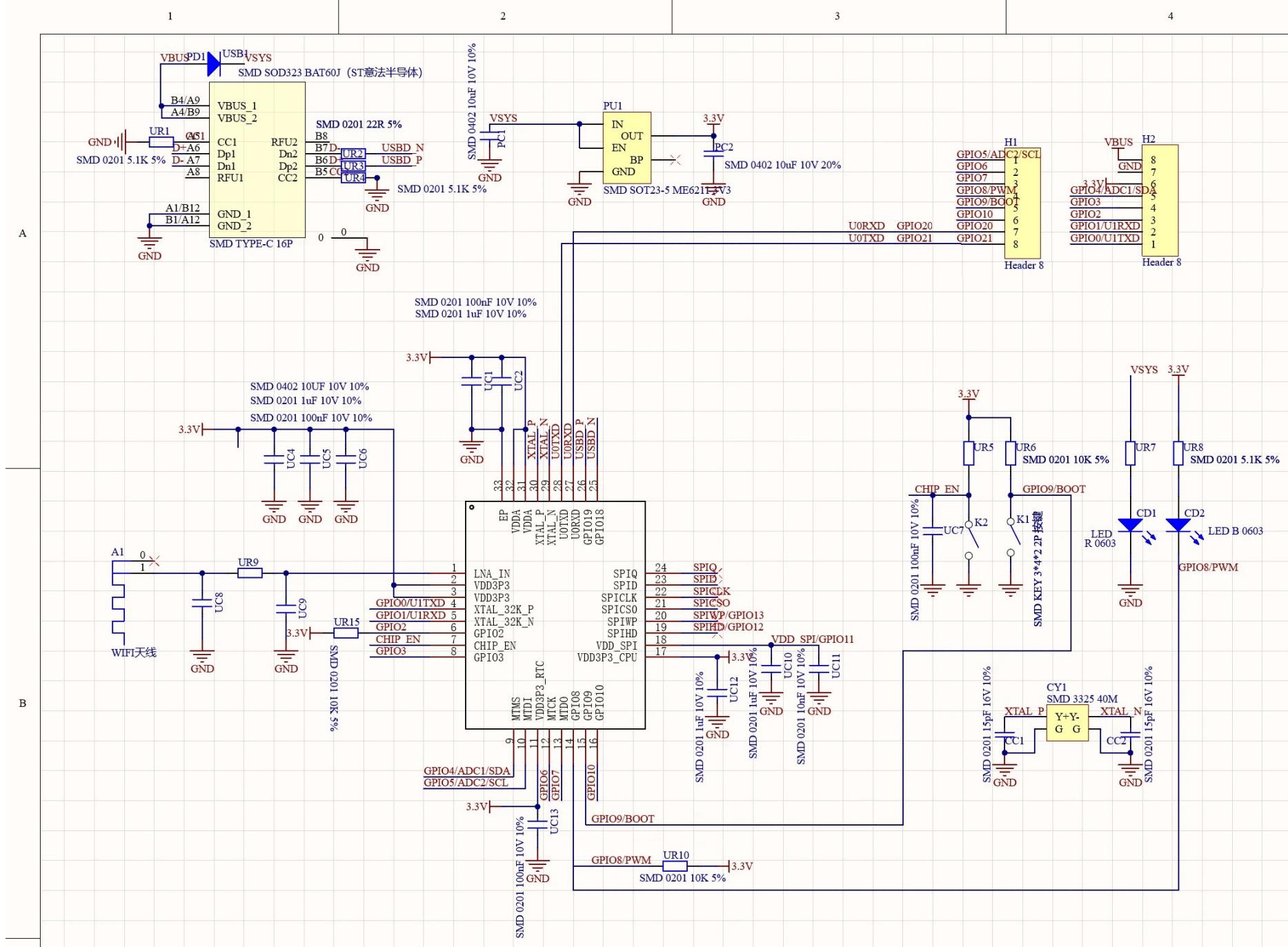
Dans mon cas je n'ai pas réussi à faire fonctionner le débugger de l'IDE. Mais l'installation fournit une console de commande (ESP-IDF CMD) qui permet de s'en sortir. Pour cela :

- Configurer l'adresse IP de base dans le routeur.
- Compiler le projet.
- Lancer la fenêtre de commande ESP-IDF CMD, et changer le dossier courant pour le dossier du projet dans l'IDE (dossier `http`).
- Connecter le module ESP par USB, en place sur AASun, avec le cavalier d'alimentation 5V retiré, et le cavalier de boot AP en place. Repérer le numéro du port COM associé, par exemple COM15
- Taper : `idf.py -p COM15 flash monitor`
Cela configure le partitionnement de la flash puis programme le bootstrap et l'application.
- A la fin de la programmation le module rebooté en mode AP et met à jour le système de fichier dur serveur HTTP.
- Avec un téléphone se connecter au module avec le SSID et le mot de passe du mode AP indiqués plus haut, puis charger la page `192.168.4.1/id` pour rentrer le SSID et le mot de passe de votre box ou routeur Ethernet.
- Enlever le cavalier de boot AP du module WIFI et faire un reset de celui-ci. Il rebooté en mode STA et dans les commentaires affichés il indique la connexion à la box et l'adresse IP qui lui est affectée.
- Vérifier que tout marche en chargeant la page d'accueil du serveur HTTP.
- Mettre le routeur hors tension, débrancher le câble USB du module WIFI, replacer le cavalier d'alimentation 5V du module WIFI. Remettre sous tension le routeur.
- Quitter le monitoring de l'ESP en tapant « `CTRL-t x` »

4.5 Schéma du module

Il est souvent difficile de trouver le schéma d'un module chinois copié par de multiples fournisseurs. Celui-ci me semble correspondre au module que j'ai acheté :





5 Câblage

Il n'est pas nécessaire de câbler l'intégralité des composants si des fonctionnalités ne sont pas nécessaires. Par exemple :

- supprimer U3 et les composants associés si les CT3 et CT4 ne sont pas utilisés.
- Si le Linky n'est pas utilisé, supprimer l'optocoupleur U7 et les composants associés.
- Même si on n'utilise qu'une entrée compteur, il faut câbler tous les composants autour du LM390(U4).
- Il peut être utile de câbler le connecteur d'extension (J14), l'évolution du prototype l'utilise pour plusieurs fonctionnalités. Et il peut servir pour des signaux de debug.

Il y a une erreur sur le schéma : il manque une liaison entre R4 et R5, facile à ajouter sur le PCB.

Erreur sur le PCB : le boîtier de la FLASH W25Q64 n'est pas en SOIC mais un SOIC-208 (2mm plus large). Elle est un peu délicate à souder...

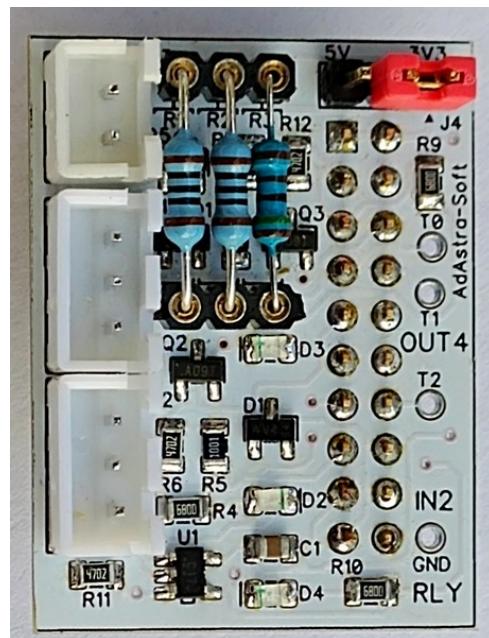
Les capteurs de température DS18B20 n'utilisent pas les broches PB2 et PB10 prévues mais le port UART sur le connecteur d'extension.

Par conséquent ces broches deviennent disponibles ce qui permet de pouvoir utiliser simultanément le Linky et le compteur d'impulsion 2. Voir §[Carte d'extension](#) qui regroupe les modifications à effectuer dans la version finale) pour le PCB V1

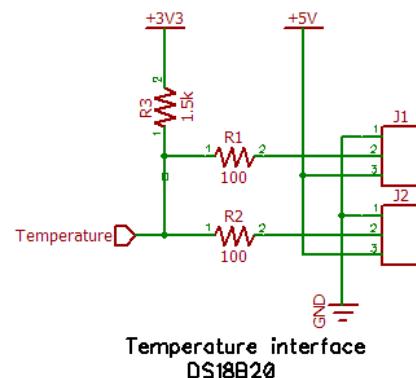
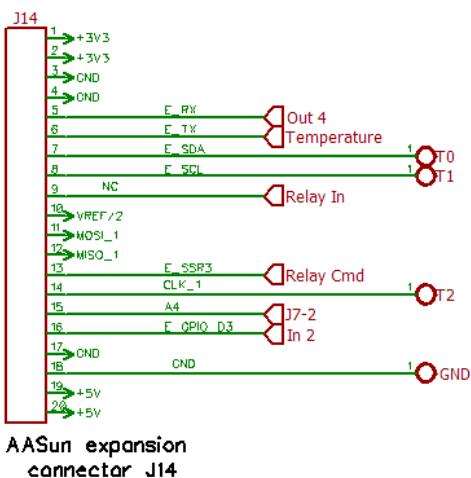
5.1 Carte d'extension

La réalisation a révélé un certain nombre de défauts et imperfections de la conception du matériel (voir le dernier chapitre). Il a été décidé de réaliser un petit circuit imprimé à placer sur le connecteur d'extension J14 pour éliminer quelques unes de ces imperfections :

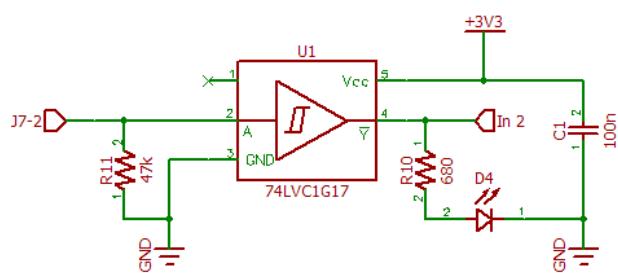
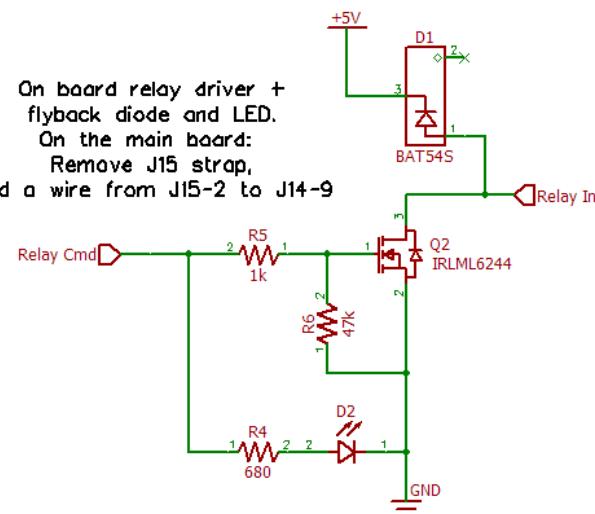
- Il manque une diode de roue libre pour le relais RLY1. De plus le relais et la sortie OUT2 utilisent la même I/O du MCU, c'est dommage.
La commande du relais est déportée sur l'extension, en y ajoutant la diode de roue libre et une LED pour un rendu visuel de l'état du relais.
Le cavalier J15 ne doit plus être utilisé et être retiré.
Le relais (OUT3) et la sortie SSR2 (OUT2) peuvent maintenant être utilisés simultanément ce qui simplifie le logiciel.
- Une sortie logique OUT4 est ajoutée sur l'extension. Elle peut sortir 3.3V ou 5V (sélection par cavalier), et peut commander un relai externe (diode de roue libre), et son état est visualisé par une LED.
- Interface pour capteurs de température DS18B20, utilisant la broche TX de la liaison série disponible sur l'extension.
Cela libère les broches TEMP_1 (PB2) et TEMP_2 (PB10) sur le circuit principal. J6 devient IN1, et PB10 est utilisé comme interface pour le LINKY (Souder un fil entre la broche 3 de J16 et la broche PB10 du MCU). Comme le Linky ne partage plus l'I/O avec l'entrée PULSE2 ils peuvent être utilisés simultanément : le cavalier J16 doit être positionné en 1-2 et ne plus être bougé. Simplification du logiciel et de l'utilisation.
- J7 (ex TEMP_2) devient disponible en coupant la piste qui le relie à PB10.
Un fil est ajouté entre J7-2 et le connecteur d'extension, et le signal est conditionné sur l'extension avec un circuit compatible 5V, et une LED pour en visualiser l'état. J7 devient donc IN2.
- Puisque quelques I/O sont encore disponibles, ajout des points de test T0, T1 et T2.



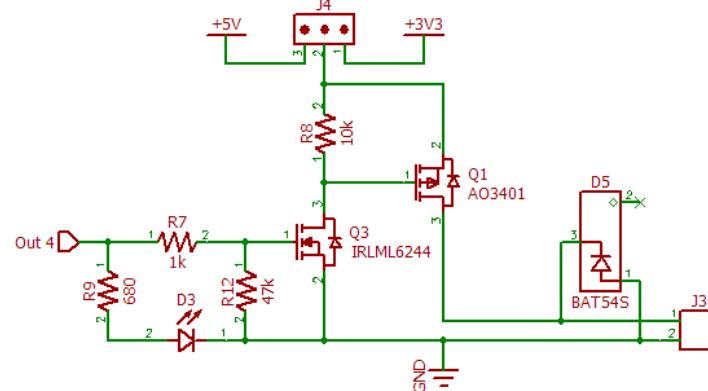
Cela peut paraître un peu compliqué, mais le schéma explique tout :



On board relay driver + flyback diode and LED.
On the main board:
Remove J15 strap,
Add a wire from J15-2 to J14-9



On the main board:
Cut the trace of PA4 near the MPU on bottom side
Cut trace at J7-2 (top side)
Add a wire fram J7-2 to J14-15



Flyback diode:
BAT54, BAT54C, BAT54S, BAV99

6 Mise en route

Il est recommandé de tester et étalonner le routeur sur banc avant de l'installer.

En cas de dysfonctionnement ou de doute c'est beaucoup plus facile de faire les manips.

Pour que le routeur fonctionne il faut que ses paramètres de mesure de tension et de courants soient au moins grossièrement corrects.

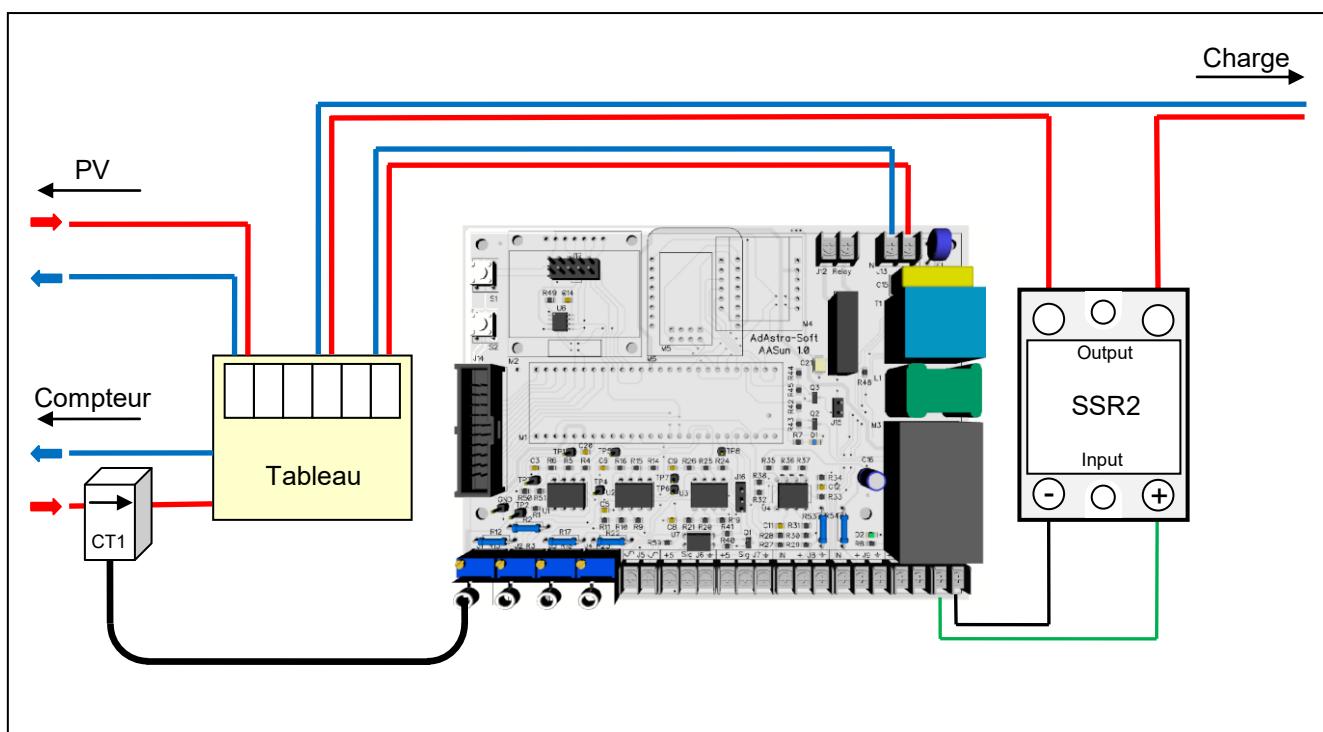
La définition `I_SENSOR_COUNT` dans le fichier `cfgParameters.h` permet de spécifier combien de CT le routeur doit gérer, entre 2 et 4.

Une fois l'application compilée la flasher dans le MCU.

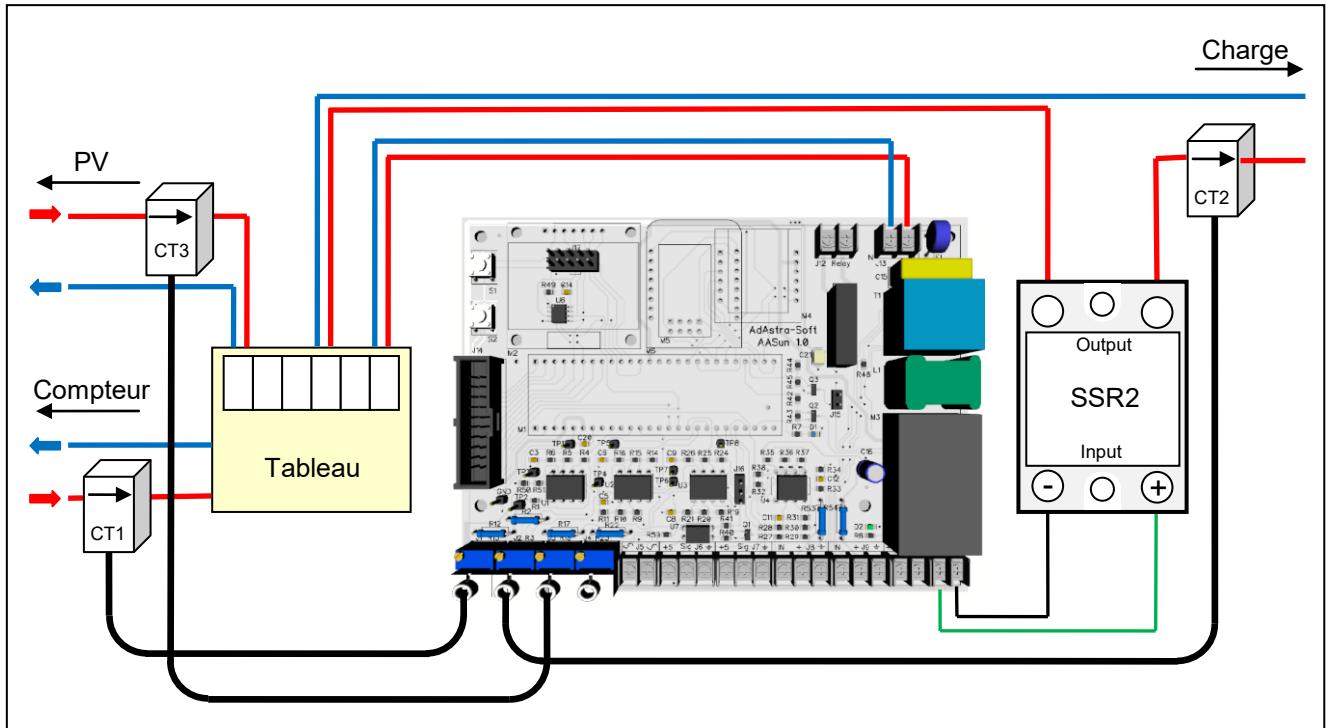
Pour utiliser le serveur HTTP il faut aussi flasher le système de fichiers correspondants dans la flash externe.

6.1 Installation

Le câblage minimal est le suivant :



Un câblage plus complet : CT2 permet de mesurer la puissance réelle routée, et CT3 permet de mesurer la production photovoltaïque.



Il est important de respecter le sens de la flèche indiqué sur le CT.

Si le CT ne peut pas être placé sur le câble de phase, il peut être placé sur le câble de neutre correspondant, en inversant le sens de la flèche.



Certains CT n'ont pas de résistance de «burden» interne, et peuvent donc développer des tensions très élevées s'ils ne sont pas connectés sur une résistance externe.

Il est impératif de connecter le CT sur le routeur avant de le clipser sur le câble.



Vérifiez d'avoir placé les résistances de «burden» sur le routeur si elles ne sont pas intégrées au CT.

Sinon les fortes tensions générées pourraient endommager les AOP

6.2 Etalonnage

Etalonner le routeur consiste à étalonner la mesure de la tension secteur et les mesures de courant. Le matériel nécessaire est :

- Un multimètre numérique, si vous lui faites confiance, ou un compteur tel que celui-ci (pas forcément plus fiable...).



Exemple pour la tension : le Linky annonce 235V, le compteur 238 et le multimètre 233. Je me suis dit que le plus fiable est le Linky (le multimètre est à moins de 1% du Linky).

Pour l'intensité de mon petit radiateur à bain d'huile le compteur et le multimètre donnent la même valeur : 2.09A.

- Un appareil électrique qui se présente sous la forme d'une résistance pure : radiateur à bain d'huile, chauffe eau, grille pain...
- Un câble d'alimentation de l'appareil avec un des fils de phase ou de neutre accessible pour y placer les transformateurs de courant (CT).
- Une liaison série ou Ethernet avec le routeur pour avoir une console interactive (PuTTY).

L'appareil électrique doit être d'une puissance correspondant à peu près à l'usage futur du routeur. S'il est de puissance très inférieure, il faudra faire plusieurs tours de fils dans le CT. Les CT n'étant pas très linéaires, un étalonnage à 100W donnera des mesures légèrement faussées à 3 kW, et inversement.

L'étalonnage se fait en utilisant les commandes à travers la console ou les pages du serveur HTTP. Sur la console les commandes de calibration commencent par 'c'. Par exemple :

c?	Affiche toutes les valeurs de calibration
cv	Affiche la valeur de calibration de la tension
cv 1715	Fixe la valeur de calibration de la tension à 1715
cwr	Ecrit la configuration en Flash
crd	Lit la configuration dans la Flash
d1	Affiche toutes les secondes les données du CT 1

- d2 Affiche toutes les secondes les données du CT 2
- ? Affiche le menu de toutes les commandes

En général la configuration initiale est plus pratique sur la console, et les ajustements fins à partir des pages HTTP.

6.3 Calibration de la tension

Afficher les données avec d1, puis ajuster la calibration de la tension avec cv xxx
La tension est calibrée, relever la valeur xxx.

6.4 Correction de la phase

Les transformateurs de courant ont tous des caractéristiques différentes, en particulier le retard qu'ils introduisent dans les mesures. Pour un routage correct il est important que la mesure de tension et la mesure du courant I1 du CT N°1 soient en phase.

La commande ds (Display Samples) permet d'afficher les valeurs ADC pour une période complète du secteur. La commande dss (Display Samples Short) permet de n'afficher que la partie centrale de la période, c'est à dire la transition entre l'alternance positive et l'alternance négative. Les colonnes correspondent à (pour une configuration avec 2 CT):

- La tension brute
- I1
- I2
- La tension avec phase corrigée.

Le but de la correction de phase est que la tension et I1 changent de signe au même moment. On corrige donc la phase de la tension pour s'adapter au CT de I1. Cela ne corrige pas forcément la phase par rapport à I2, I3..., et même parfois la dégrade pour ces courants.

Exemple :

Tableau de gauche sans correction de phase : La tension (colonne 4) change de signe à la ligne 11, et I1 (colonne 2) à la ligne 7, donc il y a un déphasage correspondant au temps de 4 échantillons (400 µs).

Tableau de droite après correction de phase avec la commande cph 8192, la tension et I1 changent de signe sur la même ligne. Par contre I2 change un peu plus tard.

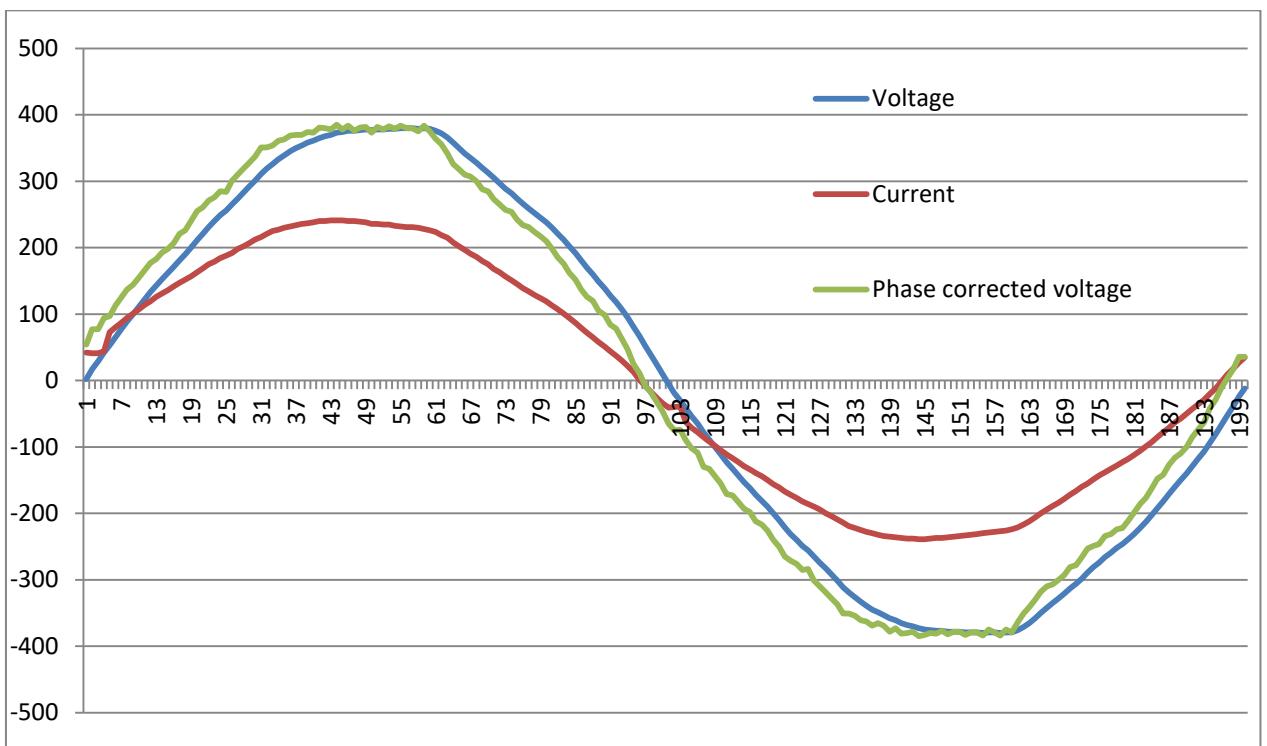
Sans correction de phase				Avec correction de phase			
dss				dss			
132	46	62	132	133	47	62	93
122	40	55	122	122	40	56	78
111	33	48	111	112	33	48	72
99	25	39	99	100	25	40	52
86	16	30	86	87	17	31	35
71	6	21	71	73	7	21	17
57	-4	11	57	58	-4	11	-2
42	-15	-1	42	43	-14	-1	-17
27	-23	-11	27	28	-23	-11	-32
12	-32	-20	12	13	-32	-20	-47
-3	-40	-28	-3	-3	-40	-28	-67

-16	-40	-27	-16	-15	-40	-27	-63
-28	-39	-27	-28	-27	-40	-27	-75
-39	-62	-50	-39	-38	-61	-50	-82
-51	-69	-58	-51	-51	-69	-57	-103
-64	-77	-66	-64	-62	-76	-64	-106
-75	-84	-73	-75	-74	-83	-72	-122
-87	-91	-81	-87	-85	-90	-79	-129
-98	-98	-88	-98	-97	-97	-87	-145
-110	-104	-95	-110	-109	-103	-94	-157

Pour étalonner il faut faire changer le déphasage par pas de 1024 jusqu'à ce que les changements de signe soient sur la même ligne.

Il ne faut pas s'attendre à ce que ce soit parfait tout le temps : les signaux sont bruités et un résultat oscillant à ± 1 ligne ou 2 est correct.

Affichage graphique des données fournies par ds :



Les lignes bleue et verte montrent la tension avant et après déphasage.

Quand la phase est ajustée les lignes verte et rouge se croisent en 0.

Il est à remarquer qu'après la mise en phase, les alternances positives et négatives ne sont plus égales... Peut-être vaudrait-il mieux mettre en phase les intensités, ce qui poserait d'autres problèmes...

6.5 Calibration des intensités

CT branchés et appareil (ou autre source de courant) éteint.

Commande `sd 0` pour arrêter le routage (**stop diverter**). Avec la commande ? au bout de la ligne `sd` il y a (0) quand le routage est à suspendu et (1) quand il est en marche.

6.5.1 Correction de l'offset de l'ADC

Sur l'afficheur afficher la page des valeurs de l'ADC « Peak (LSB) ». Il faut corriger au mieux un éventuel offset. Les deux valeurs sur une ligne doivent être en valeur absolue égales à ± 1 près. La calibration se fait avec la commande `cio i xxx` où `i` est le numéro du CT et `xxx` la valeur qui sera ajoutée aux mesures.

Par exemple si on voit :

`I1 0 -3`

La correction se fait avec :

`cio 1 1`

Tt on doit voir des valeurs qui oscillent entre :

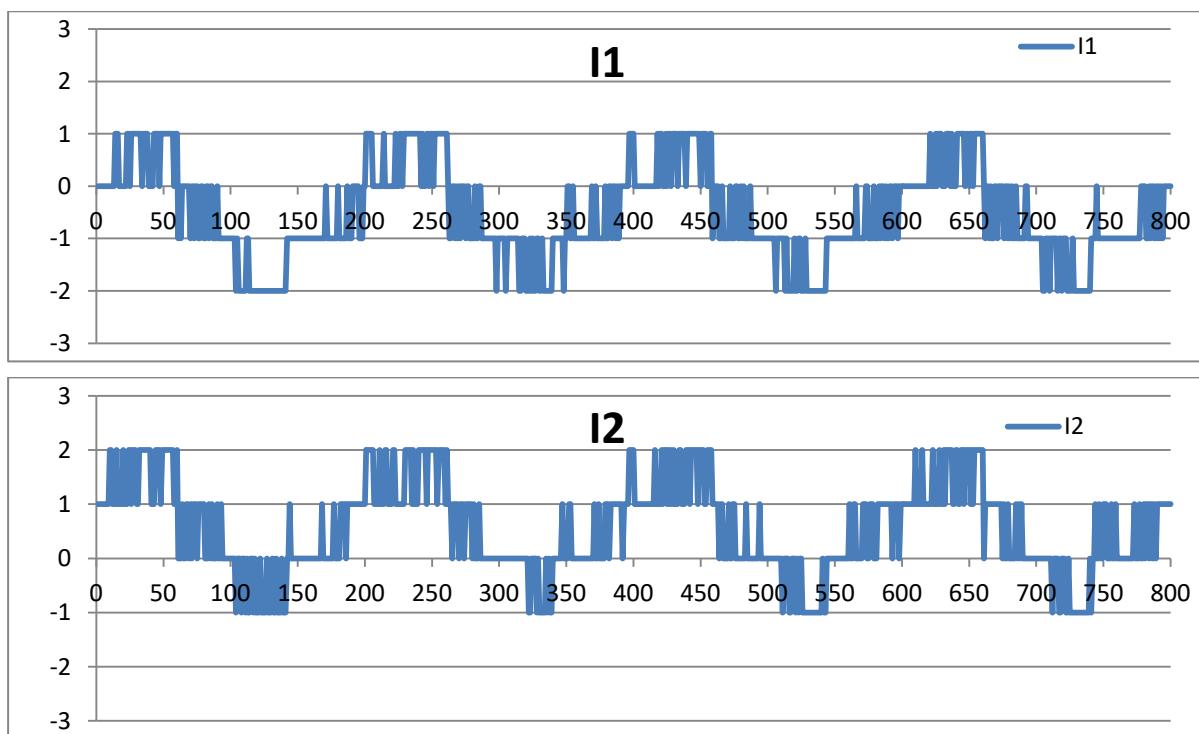
`I1 2 -2`

`I1 2 -3`

`I1 1 -2`

Corriger aussi l'offset des autres voies.

Les graphes ci-dessous montrent 4 périodes secteur près correction d'offset ADC :



On remarque :

- Les signaux sont propres.
- Il y a un léger parasitage de l'intensité par la tension secteur : ± 1 LSB, ce n'est pas grave.
- Pour vraiment corriger l'offset il faudrait utiliser $\frac{1}{2}$ LSB !

6.5.2 Calibration de l'intensité

Clipser les CT autour du fil de phase avec les flèches en direction de l'appareil pour avoir un courant positif.

Mettre l'appareil sous tension et mesurer l'intensité qu'il consomme avec le multimètre ou le compteur.

La commande d1 ou l'afficheur affichent l'intensité I1 calculée toute les secondes. Avec la commande 'ci 1 xxx' ajuster la valeur de calibration xxx de I1.

d1 pour arrêter l'affichage de I1

d2 pour démarrer l'affichage de I2 (ou sélectionner la page I2 de l'afficheur)

Calibrer I2 I3 et I4 de la même manière avec ci 2 xxx etc.

d2 pour arrêter l'affichage de I2

Les intensités sont calibrées.

6.6 Calibration de la puissance actives

Une fois la tension et les intensités calibrées, l'affichage des puissances apparentes en VA est juste.

Comme on utilise un appareil purement résistif les puissances apparentes en VA et les puissances réelles en W doivent être égales.

6.6.1 Correction de l'offset de puissance active

Appareil éteint la puissance active doit être à 0 ou très proche. Exemple pour le CT1 :

d1 Affiche les données (ou sélectionner la page I1 de l'afficheur)

cpo 1 0 Annule l'offset de p1Real Relever la valeur xxx affichée pour p1Real, c'est l'offset à annuler

cpo 1 xxx Utiliser la partie entière de xxx. La puissance p1Real affichée doit être autour de 0.

Calibrer de la même manière l'offset des autres puissances actives avec cpo 2, etc.

6.6.2 Calibration de la puissance active

Allumer l'appareil. La puissance active doit être égale à la puissance apparente :

d1 Affiche les données (ou sélectionner la page I1 de l'afficheur)

cp 1 Affiche la valeur courante de la calibration

cp 1 xxx Modifier la valeur de calibration pour obtenir l'égalité des puissances actives et apparentes.

Le Cos-Phi est aussi affiché. La calibration peut aussi être effectuée pour que le Cos-Phi oscille entre 999 et 1000 ou 1001.

Calibrer de la même manière les autres puissances actives.

6.7 Autres valeurs de configuration du routeur

cpm n xxx	Fixe la marge de puissance du routage n en W avec n égal à 1 ou 2. Lorsqu'il y a routage, le routeur s'assure qu'il y a quand même un soutirage de cette puissance pour éviter toute injection. Valeurs courantes : 0 ou 10.
cpmax n w v	Valeur maximale de la puissance à router pour le routage n égal 1 ou 2. La puissance est w en watt, et v est le voltage correspondant à cette puissance. Avec un chauffe eau annoncé pour 3000W, c'est une valeur autour de 3000 à 230V.
cks p i	Coefficients du contrôleur PI de synchronisation sur la période secteur. Ne pas toucher.
ckd p i	Coefficients du contrôleur PI du routeur. Eventuellement à ajuster en fonction de l'équipement recevant la puissance routée.
cfp n	Numéro de la page à présenter sur l'afficheur OLED à la mise sous tension du routeur.
cpc n p	Configure le compteur d'impulsion n égal 1 ou 2. La valeur p est le nombre d'impulsions par kWh. Si p vaut 0, alors ce compteur n'est pas utilisé et l'entrée peut être utilisé comme entrée logique (IN3 ou IN4).
cen n nom	Attribue un nom au compteur d'énergie n, de 0 à 8.
cal s v	Configure l'anti légionellose. Voir § Anti légionellose .

6.8 Configuration réseau

Les commandes de configuration du réseau commencent par cl. Exemple :

clip	192	168	1	130
clmask	255	255	255	0
clgw	192	168	1	254
cldns	1	1	1	1
cldhcp	s			

Pour cldhcp seul le mode adressage statique « s » est supporté (pas de DHCP).

6.9 Sauvegarde

Pour afficher la configuration courante : c?

Exemple pour une configuration avec 3 CT:

c?				
cv	1750			
cph	4096			
cio	1 0			
ci	1 595			
cp	1 1023			
cpo	1 9			
cio	2 0			
ci	2 579			

```

cp      2 995
cpo     2 10
cio     3 0
ci      3 579
cp      3 900
cpo     3 10
cpc     1 2000
cpc     2 0
cpm     1 0
cpmax   1 2832
cpm     2 0
cpmax   2 740
cdr     1 OFF
cdr     2 ON
cfr     3 ON & OUT2 & STD & V2 > 6 / DMAX = 120 & V2 < 6
cen     0 Imported
cen     1 Exported
cen     2 Div1(est)
cen     3 Div2(est)
cen     4 CT2
cen     5 CT3
cen     6 CT4
cen     7 Cnt1
cen     8 Cnt2
cks     1000 40
ckd     2 25
clip    192 168 1 132
clmask  255 255 255 0
clgw    192 168 1 254
cldns   1 1 1 1
cldhcp  s
cfp     1
cal     T1 54

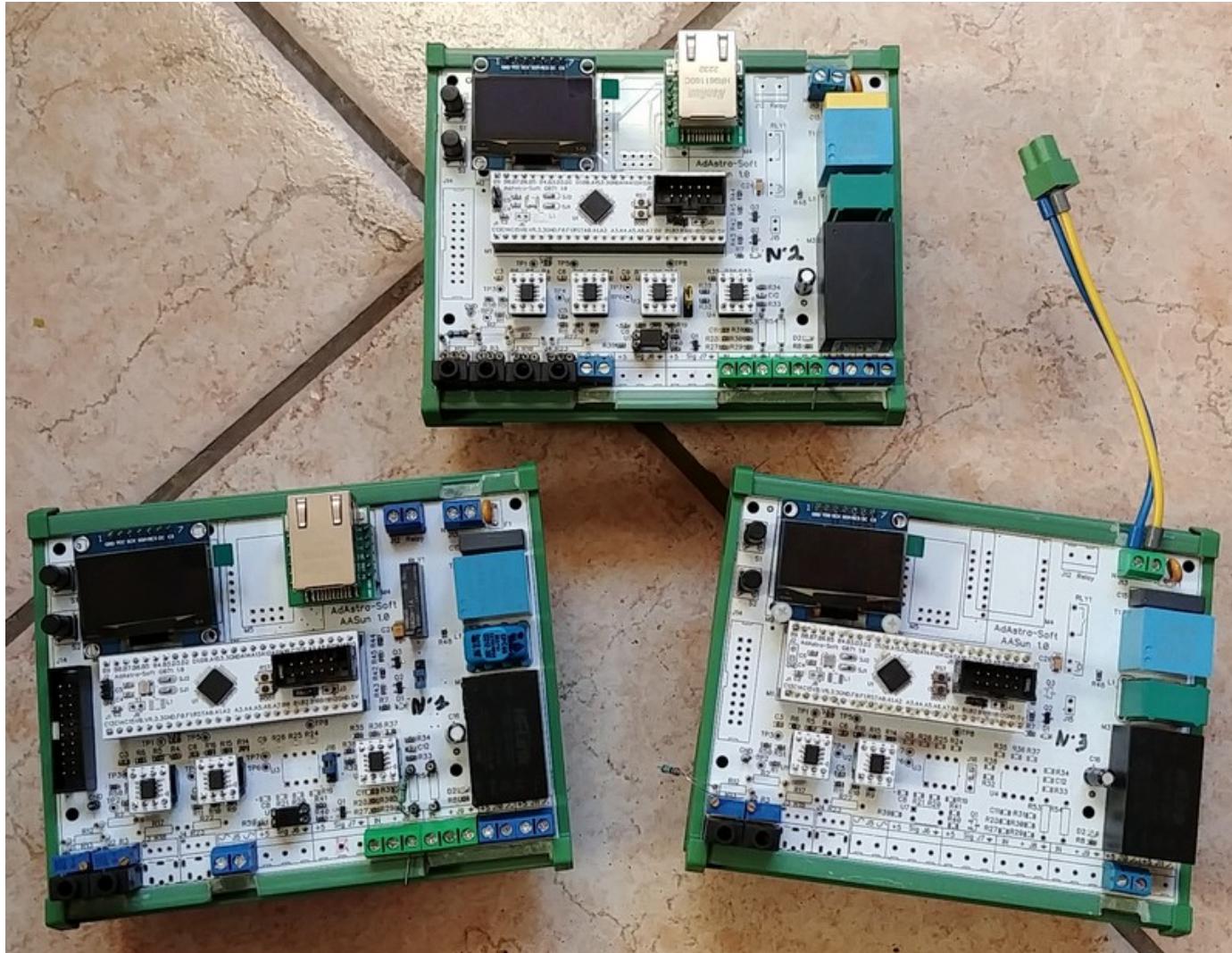
```

Il est recommandé de copier ce qui est affiché et de le sauver dans un fichier.

Si un jour il est nécessaire de restaurer la configuration, il suffira de coller cette configuration dans le terminal.

Après vérification il faudra utiliser `cwr` pour la sauver dans la FLASH du routeur.

Pour la configuration des capteurs de température se reporter au chapitre [Configuration](#) des DS18B20



3 prototypes de routeurs diversement équipés suivant leur utilisation

6.10 Configuration du MCU

Il faut disposer du logiciel STM32CubeProgrammer qui est gratuit et disponible sur le site ST.

Il y a plusieurs choses à faire avant de pouvoir utiliser un MCU neuf (qui n'a jamais été programmé) :

- Configurer les « Option Bits »
- Ecrire l'application dans la flash du MCU
- Ecrire le fichier contenant les page HTTP dans la flash externe.

Les deux premières étapes se font avec l'application STM32CubeProgrammer, en utilisant soit ST-LINK soit un adaptateur USB/UART (FTDI par exemple).

La dernière peut se faire de deux façons :

- STM32CubeProgrammer et un ST-LINK et un « External Loader » spécialement développé (fourni)
- L'application SerEL.exe et un adaptateur USB/UART

6.10.1 Les « Option Bits »

Lors de la première utilisation du MCU il est nécessaire de configurer les « Option Bits » afin de pouvoir utiliser le bouton Boot et le bootloader interne si une liaison série est utilisée pour flasher l'application.

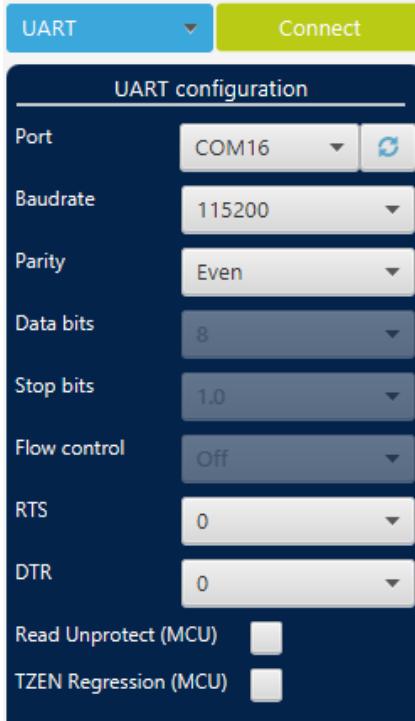
C'est la configuration par défaut sauf nBOOT_SEL qui est décoché.

Après configuration des Option Bits il faut impérativement mettre hors tension le MCU avant de continuer pour que la nouvelle configuration soit prise en compte (un reset ne suffit pas !)

User Configuration		
Name	Value	Description
nRST_STOP	<input checked="" type="checkbox"/>	Unchecked : Reset generated when entering Stop mode Checked : No reset generated when entering Stop mode
nRST_STDBY	<input checked="" type="checkbox"/>	Unchecked : Reset generated when entering Standby mode Checked : No reset generated when entering Standby mode
nRST_SHDW	<input checked="" type="checkbox"/>	Unchecked : Reset generated when entering the Shutdown mode Checked : No reset generated when entering the Shutdown mode
IWDG_SW	<input checked="" type="checkbox"/>	Unchecked : Hardware independant watchdog Checked : Software independant watchdog
IWDG_STOP	<input checked="" type="checkbox"/>	Unchecked : Freeze IWDG counter in stop mode Checked : IWDG counter active in stop mode
IWDG_STDBY	<input checked="" type="checkbox"/>	Unchecked : Freeze IWDG counter in standby mode Checked : IWDG counter active in standby mode
WWDG_SW	<input checked="" type="checkbox"/>	Unchecked : Hardware window watchdog Checked : Software window watchdog
RAM_PARITY_CHECK	<input checked="" type="checkbox"/>	Unchecked : SRAM2 parity check enable Checked : SRAM2 parity check disable
nBOOT_SEL	<input type="checkbox"/>	Unchecked : BOOT0 signal is defined by BOOT0 pin value (legacy mode) Checked : BOOT0 signal is defined by nBOOT0 option bit
nBOOT1	<input checked="" type="checkbox"/>	Unchecked : Boot from Flash if BOOT0 = 1, otherwise Embedded SRAM1 Checked : Boot from Flash if BOOT0 = 1, otherwise system memory
nBOOT0	<input checked="" type="checkbox"/>	Unchecked : nBOOT0=0 Checked : nBOOT0=1
NRST_MODE	<input type="button" value="3"/>	0 : Reserved 1 : Reset Input only: a low level on the NRST pin generates system reset, internal RESET not produced 2 : GPIO: standard GPIO pad functionality, only internal RESET possible 3 : Bidirectional reset: NRST pin configured in reset input/output mode (legacy mode)
IRHEN	<input checked="" type="checkbox"/>	Internal reset holder enable bit Unchecked : Internal resets are propagated as simple pulse on NRST pin Checked : Internal resets drives NRST pin low until it is seen as low level

Configuration de la liaison série dans CubeProgrammer

Le baudrate est à ajuster suivant la qualité de la liaison série (longueur, type de câble...). Pour une liaison de 12m en câble réseau Cat5e j'utilise 57600 bauds.



6.10.2 Mise en flash du logiciel

Avec ST-LINK et STM32CubeProgrammer ou STM32CubeIDE, suivre les manuels. ST-LINK utilise l'USB, il faut donc être à proximité du routeur.

Avec une liaison série et STM32CubeProgrammer :

Il faut d'abord passer le MCU en mode « Boot Loader ». Pour cela 2 moyens :

- Si on a un accès physique au routeur il faut appuyer sur le bouton BOOT et tout en le maintenant pressé appuyer sur le bouton RESET.
Il faut impérativement utiliser ce moyen avec un MCU dans lequel l'application n'a jamais été flashée.
- A distance il faut se connecter au routeur par la liaison série (avec PuTTY par exemple) ou par Telnet et taper la commande : *reset boot*
puis quitter PuTTY ou Telnet (pour libérer la ligne).
Ce moyen n'est pas utilisable avec un MCU neuf.

Ensuite utiliser STM32CubeProgrammer pour flasher l'application.

A la fin de la mise en flash il faut malheureusement aller faire un RESET du MCU pour qu'il démarre sur l'application...

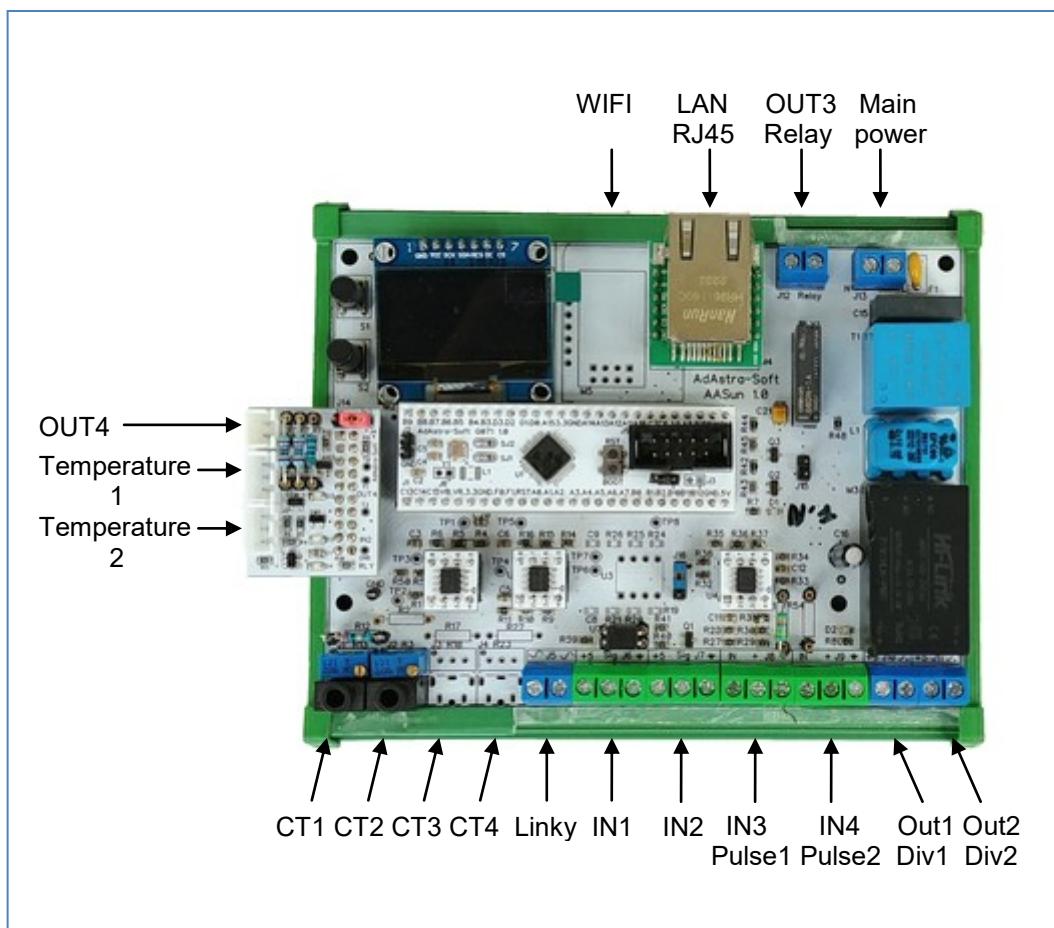
7 Routage et forçage

AASun peut réaliser simultanément du routage et du forçage sur différentes voies.

Le routage c'est la dérivation en temps réel de la puissance excédentaire (qui autrement serait injectée sur le réseau) vers un appareil. La puissance à router est calculée toutes les 10ms pour générer un signal de type PWM à destination d'un relais statique (SSR). Il s'agit donc d'une commande proportionnelle.

Le forçage consiste à commuter l'alimentation d'une charge en tout ou rien. Les conditions de forçage sont évaluées toutes les secondes. Il y a deux modes de forçages possibles.

Les tableaux d'entrées et de sorties correspondent à la configuration avec PCB d'extension sur J14 :



7.1 Les voies de sortie

AASun dispose de 4 voies de sortie disponibles pour le routage et/ou le forçage :

Nom	Connecteur	Type	Usage	Commentaire
OUT1	J10	PWM PWM/TOR	Routage Forçage	Sortie conditionnée 5V pour SSR de type « Random »
OUT2	J11	PWM PWM/TOR	Routage Forçage	Sortie conditionnée 5V pour SSR de type « Random »
OUT3	J12	Relais	Forçage	Relais sur la carte
OUT4	J3 extension (J14 :5)	TOR	Forçage	Sortie conditionnée 3.3V/5V pour relais (diode de roue libre) Connecteur XH 2 broches

Les voies OUT1 et OUT2 peuvent être utilisées en routage ou en forçage. La commutation routage / forçage se fait automatiquement en fonction de l'état vrai ou faux des règles de routage et de forçage.

Pour le routage ces voies doivent être équipées de SSR de type « Random ». A défaut elles peuvent être utilisées pour le forçage seulement en « Tout Ou Rien ».

Les voies OUT3 et OUT4 sont dédiées au forçage.

7.2 Les voies d'entrées

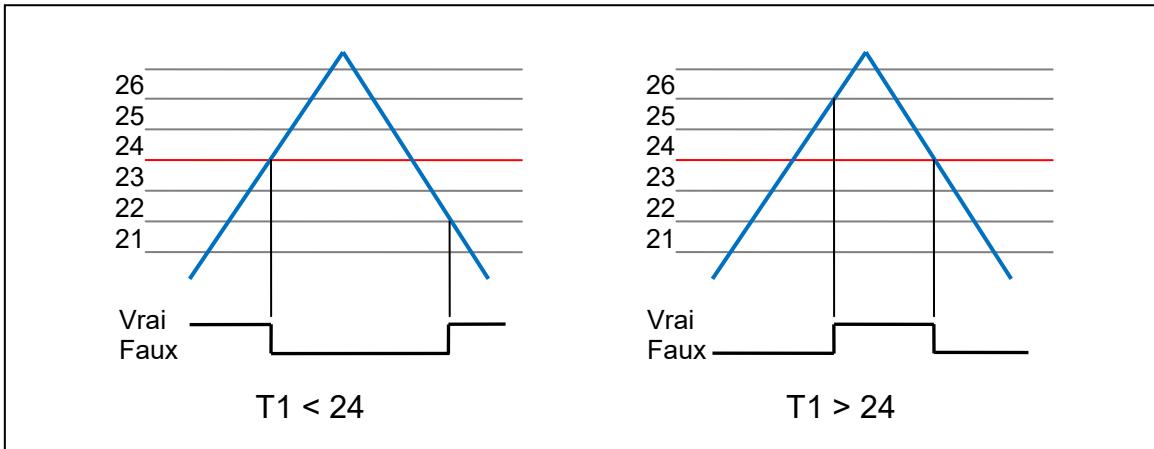
Les forçages peuvent utiliser l'état de 4 voies en entrée:

Nom	Connecteur	Type	Commentaire
IN1	J6	3.3V	Entrée 3.3V <i>Attention : J6 comporte une broche 5V qui ne doit pas être utilisée pour polariser l'entrée (sauf avec un diviseur de tension)</i>
IN2	J7 (J14-15)	3.3V/5V	Entrée compatible 3.3V et 5V
IN3	J8	30V	Entrée de compteur d'impulsion qui peut aussi être utilisée en entrée logique (3-30V)
IN4	J9	30V	Entrée de compteur d'impulsion qui peut aussi être utilisée en entrée logique (3-30V)

7.3 Hystérésis de température

Une expression faisant intervenir une température utilise une hystérésis de 2°C.

Exemple pour une règle ne contenant qu'une seule expression utilisant une température :



La gestion de l'hystérésis a été conçue afin que :

Pour l'opérateur < :

L'expression n'est vraie que si la température est inférieure à la valeur.

Pour l'opérateur > :

L'expression n'est vraie que si la température est supérieure à la valeur.

Il faut tenir compte de ce mécanisme lors de l'écriture des règles si on veut une gestion de température précise.

Exemple : dans le graphe ci-dessus si l'expression $T1 > 24$ est utilisée dans une règle stop de forçage, le forçage ne s'arrêtera que lorsque la température de $24+2$ sera atteinte soit 26°C . Dans ce cas il vaut mieux écrire $T1 > 22$.

8 Routage

Il y a deux voies de routages nommées OUT1 pour le SSR1 et OUT2 pour le SSR2, et une seule peut être active à un instant donné. Chaque voie est régie par une règle qui est évaluée chaque seconde pour déterminer la voie à activer.

La voie OUT1 est prioritaire sur la voie OUT2 : si les deux règles sont vraies en même temps c'est la voie OUT1 qui sera active.

8.1 Syntaxe d'une règle

Une règle est une équation logique comportant :

- Des paramètres tels que le numéro de la sortie concernée (OUTx), ou son état On ou OFF.
- Des expressions reliées par des opérateurs AND. Donc pour qu'une règle soit vraie il faut que toutes les expressions soient vraies.
Une règle peut contenir jusqu'à 6 expressions.

Une règle peut être inhibée (état OFF), dans ce cas elle sera toujours fausse. A l'état ON la règle est évaluée chaque seconde.

Chaque voie de routage dispose d'une seule règle : quand la règle est vraie le routage de cette voie est valide, et le routage sur cette voie est effectif seulement si c'est la règle à l'état vrai de plus haute priorité.

Les expressions et les paramètres sont toujours séparés par des caractères '&', et les éléments des expressions par des espaces.

Une expression est de la forme : Source Opérateur Valeur

Ce tableau indique les combinaisons valides :

Source	Opérateur	Valeur	Commentaire
T1 à T4	< >	entier	Une température
I1 à I3	=	0 ou 1	Une entrée logique physique
P1 à P4, PD	< >	entier	Une source de puissance : CT1 à CT4 ou PD puissance routée
V1 à V4	< = >	entier	Valeur variable interne (Forçage seulement)
HM	< = >	hhmm	Une heure + minutes (Forçage seulement)
WD	< = >	xxxxxx	Jour de la semaine (Forçage seulement)

WD indique des jours de la semaine dans l'ordre Anglo Saxon : dimanche, lundi etc.

Un '.' indique un jour invalide, un autre caractère un jour valide, et il faut donner 7 caractères. Pour spécifier lundi et jeudi on utilisera :

WD = .L..J..

ou :

WD = .X..X..

Remarque :

Une règle de routage sans expression est toujours vraie.

Une règle de forçage sans expression et toujours fausse.

Exemple 1

SSR1 OFF

SSR2 ON

C'est le cas de routage le plus simple qui est prévu dans la configuration par défaut : le routage est toujours assuré par la voie SSR2, il n'y a pas besoin d'équiper la voie 1.

Exemple 2

SSR1 ON & T1 > 55

SSR2 ON

Le routage est assuré par la voie 2 lorsque la température T1 est inférieure à 55°C, et par la voie 1 dans le cas contraire.

Autrement dit quand le ballon ECS connecté à la voie 2 a atteint une température suffisante, la puissance est envoyée ailleurs par la voie 1. Si la température du ballon redescend, la voie 1 deviendra fausse et le routage vers le ballon reprend. Voir en §7.2 [Hystérésis de température](#) l'effet de l'hystérésis de température.

Exemple 3

SSR1 ON & I3 = 0

SSR2 ON

Même exemple que ci dessus mais le ballon n'est pas équipé d'un thermomètre mais d'un thermostat mécanique qui s'ouvre quand la température est atteinte. Le thermostat est relié à l'entrée 3.

8.2 L'IHM

Il est possible de configurer les règles de routage à partir de la console avec les commandes :

cdr **Configure Diverting Rule.** Indiquer le numéro du routage 1 ou 2 puis la règle.
Exemple :

cdr 1 on & t1 > 50

dr **Diverting Rule.** Permet de changer l'état ON ou OFF de la règle d'un routage.

Exemple :

dr 1 off

Comme d'habitude c? permet de lister la configuration et donc les règles de routage.

Ne pas oublier `cwr` après avoir établi les règles pour les mémoriser en flash.

Une page HTTP est dédiée au routage / forçage. Elle permet les mêmes opérations que ci-dessus :

Diverter is : **On**

OUT1 (SSR) Diverting

OUT2 (SSR)

OUT3 (Relay)

OUT4 (Digital)

Diverting 1 (high priority)
Divertor 1 rule:

Diverting 2 (low priority)
Divertor 2 rule:

Disabled **Send**

Disabled **Send**

Le « voyant » vert du routage 2 indique qu'il est actif. Cela est aussi affiché dans le tableau de gauche.

9 Forçage

Le forçage permet d'alimenter des équipements indépendamment de la puissance en surplus.

Un forçage est défini par deux règles, utilisées à des moments différents :

- Règle START : Evaluée lorsque le forçage est inactif. Le forçage devient actif lorsque la règle devient vraie.
- Règle STOP : Evaluée lorsque le forçage est actif. Le forçage devient inactif lorsque la règle devient vraie.

Il y a 8 jeux de règles de forçage disponibles qui ont chacun une priorité : le forçage 1 a la plus haute priorité, le 8 la plus faible. Donc un forçage actif peut être supplanté par un forçage de priorité supérieure.

De plus les forçages ont une priorité supérieure au routage : si un routage et un forçage sont actifs pour la même sortie OUTx, c'est le forçage qui l'emporte.

Un forçage s'applique à une sortie OUTx. La gestion des priorités s'applique indépendamment pour chaque sortie. Pour chaque sortie l'ensemble des routages et des forçages la concernant sont évaluées chaque seconde, et celui qui est actif et de plus haute priorité est sélectionné. Donc il est possible d'avoir plusieurs routages/forçages actifs en même temps sur plusieurs sorties OUTx différentes.

9.1 Règles de forçage

La règle START du forçage contient des expressions pour évaluer les conditions de démarrage de ce forçage, formées comme indiqué dans le §8.1 [Syntaxe d'une règle](#) concernant le routage.

Il y a des paramètres de configurations qui sont obligatoires :

- ON ou OFF Si OFF est présent ce forçage est inhibé : sa règle START n'est pas évalué.
- OUTx La voie sur laquelle s'applique le forçage avec x de 1 à 4
- STD ou AUTO Le mode de forçage

D'autres paramètres sont optionnels.

La règle STOP peut contenir des paramètres en plus des expressions :

- DMIN Durée minimale d'activation
- DMAX Durée maximale d'activation

Exemple :

START ON & OUT2 & STD & HM = 2230 & T1 < 40

STOP DMAX = 60m & T1 > 50

Si à 22h30 la température du ballon ECS est inférieure à 40°C alors le forçage démarre sur la sortie 2 jusqu'à ce que la température atteigne 52 °C (il faut tenir compte de l'hystérésis de température !). Le forçage s'arrête aussi si la température n'est pas atteinte en 1 heure.

Le forçage remplace temporairement le routage sur OUT2, qui normalement à cette heure là est inactif.

9.2 Le mode STD

Le mode standard est simple et intuitif :

Quand le forçage est inactif et que la règle START devient vraie le forçage s'active.

Quand le forçage est actif et que la règle STOP devient vraie le forçage se désactive.

Il peut être intéressant de démarrer un forçage pour une certaine durée fixe, ou une durée minimale ou maximale. Pour cela on utilise les paramètres DMIN ou DMAX, qui sont spécifiques au mode STD.

Ces paramètres s'emploient dans la règle STOP. Ils sont exclusifs.

9.2.1 DMIN

Le paramètre DMIN permet de spécifier la durée minimale d'activation du forçage une fois que la règle START est vraie.

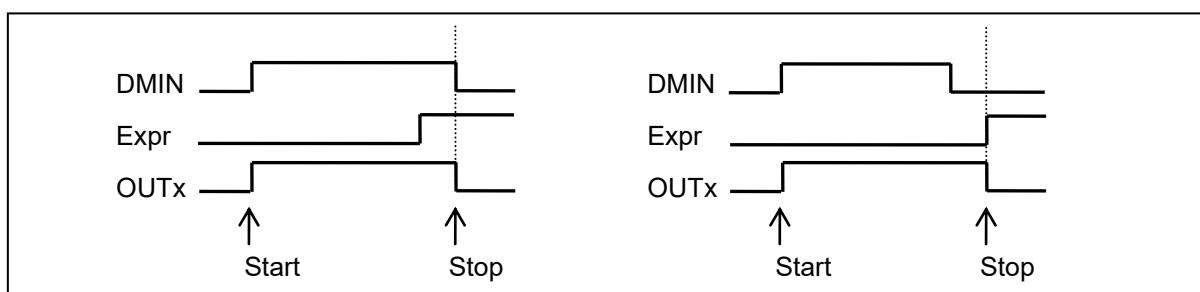
Tant que la durée minimale n'est pas écoulée les autres expressions ne sont pas évaluées.

Une fois la durée minimale écoulée le paramètre DMIN est ignoré et les autres expressions sont prises en compte.

Les durées sont exprimées en secondes ou en minutes. Exemple de règles STOP pour une durée minimale de 5 minutes :

DMIN = 300 & i3 = 1

DMIN = 5m & i3 = 1

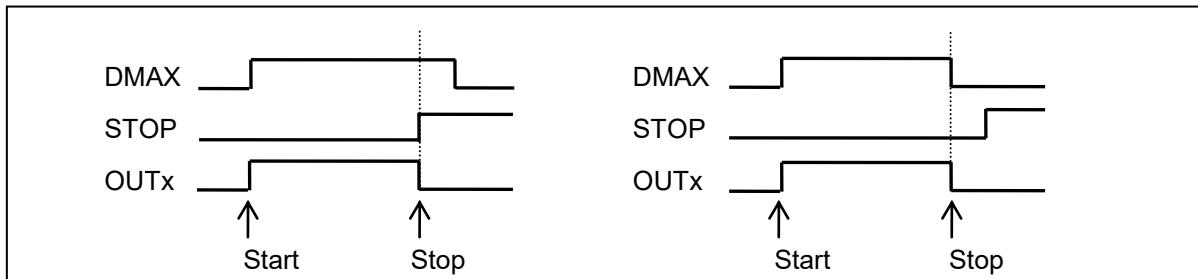


On peut considérer qu'il s'agit d'un OU entre la durée minimale et les expressions : la durée correspond à la plus longue des deux.

9.2.2 DMAX

Le paramètre DMAX permet de spécifier la durée maximale d'activation du forçage une fois que la règle START est vraie.

Une fois que la durée spécifiée par DMAX est écoulée la règle STOP devient vraie et le forçage se désactive même si toutes les autres expressions sont fausses.



On peut considérer que DMAX permet un ET entre la durée maximale et les expressions : la durée correspond à la plus courte des deux. Ou bien que DMAX est un timeout pour les expressions de la règle STOP.

Il y a un cas particulier qui est pris en compte : A l'échéance de la durée DMAX le forçage se désactive. Mais si la condition START est vraie à ce moment là alors le forçage devrait se réactive aussitôt, ce qui ferait que DMAX ne serait pas respecté.

Ce cas est traité de la façon suivante : quand DMAX est utilisé et que la règle STOP devient vraie le forçage se désactive et passe dans un état de transition jusqu'à ce que la règle START soit vue fausse. Pendant cet état de transition le forçage ne peut pas être activé.

DMAX introduit un temporisateur non ré-armable.

9.3 Le mode AUTO

Le mode STD est simple mais a des limitations :

- Il suffit que la règle START soit vraie 1 seconde pour que le routage s'active. Il n'est donc pas possible de spécifier une durée minimale d'inactivité
- Au delà du délai DMIN Il suffit que la règle STOP soit vraie 1 seconde pour que le routage se désactive.
- Il n'y a pas de filtrage des conditions START et STOP pour ignorer les variations courtes de leur état.

Cela peut produire des à-coups plus ou moins brefs du forçage, ce qui peut être préjudiciables aux équipements connectés.

Le mode AUTO est une tentative de réponse à ces limitations, en introduisant des durées minimales d'activation et de désactivation et en utilisant un filtrage des règles START et STOP.

Le mode AUTO utilise 3 paramètres spécifiques :

AON Durée minimale d'activation du forçage.

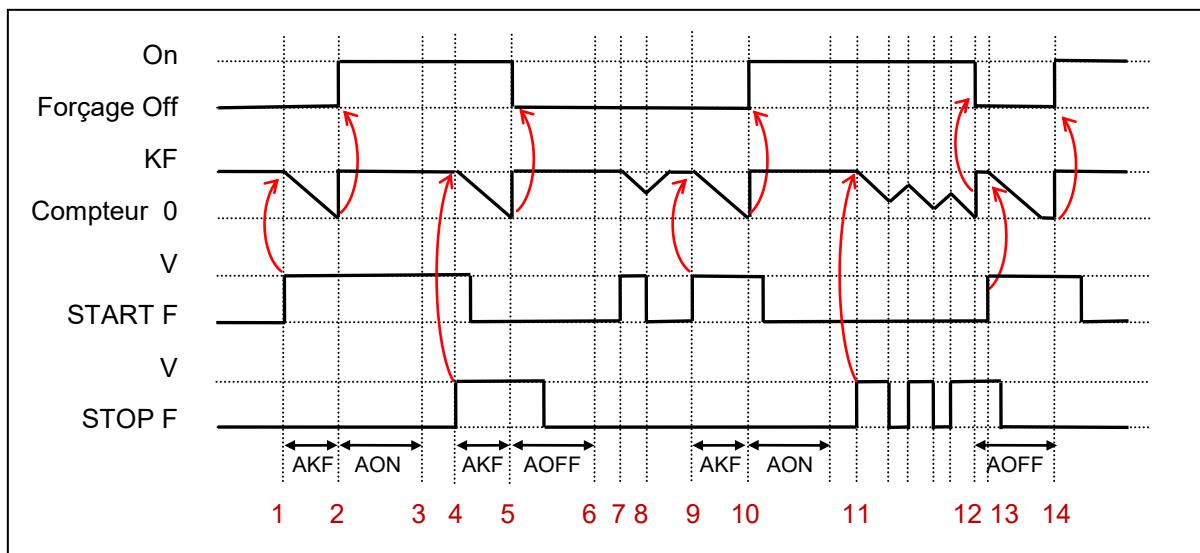
AOFF Durée minimale d'inactivité du forçage.

AKF Constante de temps du filtrage. C'est le temps minimum entre le moment où une règle devient vraie et le moment où l'action correspondante est exécutée.

9.3.1 Fonctionnement du filtrage

Le filtrage est une sorte de moyenne temporelle de la valeur de la règle. Un compteur est initialisé avec la valeur AKF, puis évolue chaque seconde avec la valeur de la règle :

- Quand la règle est fausse il est incrémenté
- Quand la règle est vraie il est décrémenté.
- Quand il atteint la valeur 0 le changement d'état du forçage est effectué, et le compteur est réinitialisé avec la valeur AKF.



Le graphique démarre avec le forçage inactif :

- 1 La règle START devient vraie : le décompte de AKF démarre.
- 2 Le décompte arrive à 0 : le forçage devient actif pour une durée AON.
- 3 La durée AON est écoulée.
- 4 La règle STOP devient vraie : le décompte d'AKF démarre
- 5 Le décompte arrive à 0 : le forçage devient inactif pour une durée AOFF.
- 6 La durée AOFF est écoulée.
- 7-8 Une durée de START est trop courte pour que le compteur AKF arrive à 0. Le filtrage a été efficace.
- 9-10 Une durée de START suffisamment longue pour activer le forçage.
- 11-12 Une série d'impulsions STOP trop courtes pour individuellement déclencher une désactivation, mais leur densité est telle que le compteur AKF peut arriver à 0 et désactiver le forçage. Le filtrage est aussi efficace.
- 13 La règle START devient vraie pendant la durée AOFF et le compteur AKF arrive à 0 avant l'échéance d'AOFF.
- 14 A la fin de AOFF le compteur AKF est à 0, donc le forçage s'active immédiatement.

9.4 Le paramètre ratio

En forçage il est possible de spécifier la puissance à fournir à l'aide d'un pourcentage appelé ici ratio.

Il y a deux modes de ratio : normal et burst.

S'il n'y a pas de ratio spécifié dans la règle START un ratio de 100% est assumé.

9.4.1 Ratio normal

Le ratio normal est utilisable sur les voies équipées de SSR dans les modes de forçage STD et AUTO.

Dans ce mode la puissance fournie est réglée en utilisant le signal PWM qui sert au routage. Il s'agit donc de hacher le courant avec une période de 100 HZ.

Ce mode de ratio est à réserver aux charges résistives comme c'est le cas pour le routage.

Le ratio normal est un paramètre spécifié dans la règle START avec un chiffre entre 1 et 100 suivi de '%'. Exemple pour un ratio de 50% :

```
ON & OUT1 & STD & 50% & I1 = 0
```

Pour un ratio normal de 100% sur une voie SSR, la commande du SSR est à 1 en continu, il n'y a donc pas de hachage du courant.

9.4.2 Ratio burst

Ce mode a pour objectif d'augmenter considérablement la période utilisée par le ratio : passer de 10ms à 4 minutes (valeur fixée dans le code mais modifiable).

Ce mode n'est utilisable que dans le mode de routage STD. Le mode de routage AUTO a son propre mécanisme de gestion des temps d'activité et d'inactivité.

Ce mode est utilisable sur toutes les voies et permet donc de moduler la puissance sur les sorties relais et logiques.

Dans ce mode la période utilisée pour régler la puissance est de 240 secondes. Cela veut dire que pour une puissance de 50% le forçage sera actif 120 secondes suivi de 120 secondes d'inactivité.

Le paramètre ratio burst est spécifié dans la règle START avec un chiffre entre 1 et 100 suivi de '%' et de 'B' (pour Burst). Exemple pour un ratio de 50% en mode burst :

```
ON & OUT1 & STD & 50%B & I1 = 0
```

Avec ce mode de ratio, pendant la phase active du forçage sur une voie SSR la commande du SSR est à 1 en continu, il n'y a donc pas de hachage du courant.

9.5 Actions manuelles

Il est possible d'activer et de désactiver manuellement un forçage. Cette action bascule l'état du forçage :

- Si le forçage est inactif il est activé : l'action manuelle simule une règle START à vrai.
- Si le forçage est actif il est inactivé : l'action manuelle simule une règle STOP à vrai.

L'action manuelle est possible même si le forçage est inhibé (état OFF). Un forçage inhibé et forcément inactif. Lorsqu'une action manuelle est appliquée à ce forçage, il est activé mais pas désinhibé, et la sortie OUTx sera commandée. Lorsque la condition de fin ou une autre action manuelle surviendra le forçage sera désactivé mais restera inhibé.

Une action manuelle peut être considérée comme un 'forçage' du forçage.

Il est donc possible de définir un forçage avec des règles jamais vraies, et de le commander manuellement :

START	OUT4 & STD
STOP	DMIN = 1

La première commande manuelle va activer le forçage, et la seconde le désactiver.

Autre exemple plus utile. Démarrer manuellement un forçage qui a une règle de fin habituelle.

START	OUT4 & STD
STOP	DMAX = 45m & T1 > 53

La seule façon d'activer ce forçage est une action manuelle. Il se désactivera quand la température dépassera 55°C. Si cette condition n'arrive pas le forçage se terminera de toute façon au bout de 45 minutes. Dans ce cas il faut que le forçage ne soit pas inhibé, puisque la règle STOP doit être évaluée normalement.

Un autre type d'action manuelle (mais sans utiliser la fonctionnalité 'action manuelle') : installer un bouton poussoir sur I2 par exemple et utiliser :

START	OUT4 & STD & I2 = 1
STOP	DMAX = 45m & T1 > 53

9.5.1 Cas du forçage en mode AUTO

Un forçage en mode AUTO a un comportement particulier lors d'une commande manuelle.

Lorsque le forçage est inactif, la commande manuelle va l'activer pour la durée AON, quel que soit le résultat réel de la règle START. C'est le comportement souhaité de la commande manuelle : simuler une règle START à vrai.

Lorsque le forçage est actif, l'action manuelle le rend inactif. Si la règle START est vraie à ce moment là le forçage se réactive.

En pratique une action manuelle sur un forçage en mode AUTO a peu d'intérêt.

9.6 L'IHM

Il est possible de configurer les règles de forçage à partir de la console avec les commandes :

cfr **Configure Forcing Rule.** Indiquer le numéro du forçage 1 à 8 puis les règles START et STOP séparée par un '/'.

Exemple :

```
cfr 1 OFF & OUT4 & STD & T1 < 25 / T1 > 55
```

cfre **Configure Forcing Rule Enable.** Permet de changer l'état ON ou OFF d'un forçage. Exemples

Inhibe le forçage 3: cfre 3 0

Désinhibe le forçage 3: cfre 3 1

fman **Forcing Manual.** Permet une action immédiate de l'utilisateur : activer un forçage inactif ou désactiver un forçage actif.

Exemple changer l'état du forçage 3: fman 3

Comme d'habitude c? permet de lister la configuration et donc les règles de forçage.

Ne pas oublier cwr après avoir établi les règles pour les mémoriser en flash.

Une page HTTP est dédiée au routage / forçage. Elle permet les mêmes opérations que ci dessus. Exemple avec le forçage 1 inhibé et donc inactif, et le forçage 2 actif (voyant vert).

Force 1 (highest priority)	Force 2
Start rule:	Start rule:
OUT4 & STD & T1 < 25	OUT4 & STD & I3 = 0
Stop rule:	Stop rule:
T1 > 55	I3 = 1
<input checked="" type="radio"/> Man <input checked="" type="checkbox"/> Disabled <input type="button" value="Send"/>	<input checked="" type="radio"/> Man <input type="checkbox"/> Disabled <input type="button" value="Send"/>

10 L'historique

AASun collecte les données de 9 sources de puissance:

- Importée, puissance positive de CT1
- Exportée, puissance négative de CT1
- Estimation de la puissance routée 1
- Estimation de la puissance routée 2
- CT 2
- CT 3
- CT 4
- Compteur d'impulsion 1
- Compteur d'impulsion 2

Ces puissances sont comptabilisées en énergies journalière remises à 0 à minuit, et en énergies totales jamais remise à 0 (sauf sur action de l'utilisateur).

Les puissances sont moyennées par $\frac{1}{4}$ d'heure, ce qui donne un tableau de 96 valeurs par jour et par source.

Chaque jour à minuit les énergies journalières et le tableau des puissances sont sauvés en FLASH, avec une profondeur de 31 jours, ce qui constitue l'historique des données.

Le tableau du total des puissances est sauvé en FLASH toute les 2 heures afin de ne pas le perdre totalement en cas de coupure d'alimentation.

L'indexation du tableau de l'historique marche à rebours : le rang 0 correspond aux données les plus récentes, c'est à dire celle d'hier. On peut ainsi remonter jusqu'au rang 30 si ces données existent. On parle de rang qui est relatif à la date d'aujourd'hui, plutôt que d'indice qui est une position dans un tableau.

L'IHM de la console permet de gérer l'historique des énergies et des puissances :

La commande `e` affiche le petit menu de gestion des énergies :

```
e
?
?      Print Energy menu
R      Read total energy from FLASH
W      Write total energy to FLASH
Z n    Zero total energy counter (a|0:8)
z n    Zero daily energy counter (a|0:8)
d n    Display with history rank n
```

La commande `e d 1` affiche les énergies Totale, Journalière et l'Historique de rang 1 c'est à dire d'avant hier :

```
e d 1
Total
 2023/12/01
0 Imported      9500
1 Exported      0
2 Div1(est)     0
3 Div2(est)     0
4 Div1          -4
5 PV            42
7 Cnt1          0
8 Cnt2          0
Today
 2023/12/01
0 Imported      6394
1 Exported      0
2 Div1(est)     0
3 Div2(est)     0
4 Div1          -4
5 PV            53
7 Cnt1          0
8 Cnt2          0
History
 2023/11/29
0 Imported      3106
1 Exported      0
2 Div1(est)     0
3 Div2(est)     0
4 Div1          0
5 PV            -11
7 Cnt1          0
8 Cnt2          0
```

Le numéro dans la colonne de gauche est le numéro à utiliser pour effacer un compteur particulier avec les commandes `e Z` et `e z`.

La commande `h` affiche le petit menu de gestion des puissances :

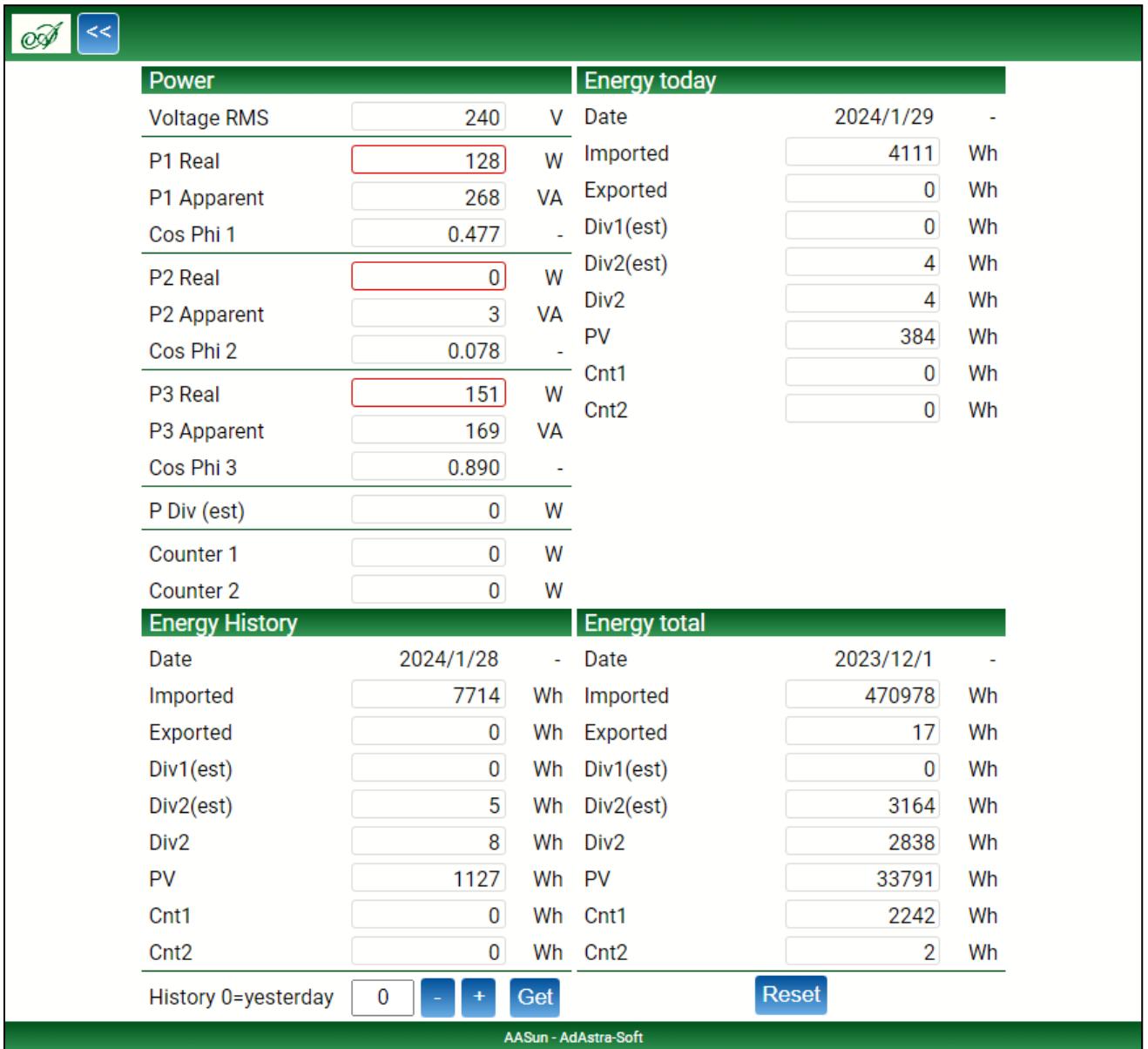
```
h
h ?  Print help
h v  Toggle debug history display
h d  Dump today power history
h h n Dump power history at rank n
h H  Dump flash history summary
h Z  Erase flash history area
h W  Write today history to flash
```

La commande `h H` affiche la liste des données disponibles dans l'historique.

La commande `h h n` affiche les données de l'historique de puissance de rang n, par exemple :

h h 1							
2023/11/25							
00:00	0	0	0	0	0	0	0
00:15	0	0	0	0	0	0	0
00:30	0	0	0	0	0	0	0
00:45	0	0	0	0	0	0	0
01:00	0	0	0	0	0	0	0
01:15	0	0	0	0	0	0	0
01:30	0	0	0	0	0	0	0
01:45	0	0	0	0	0	0	0
02:00	0	0	0	0	0	0	0
02:15	0	0	0	0	0	0	0
02:30	0	0	0	0	0	0	0
02:45	0	0	0	0	0	0	0
03:00	0	0	0	0	0	0	0
03:15	0	0	0	0	0	0	0
03:30	0	0	0	0	0	0	0
03:45	0	0	0	0	0	0	0
04:00	0	0	0	0	0	0	0
04:15	0	0	0	0	0	0	0
.							
.							
.							
21:00	0	0	0	0	0	0	0
21:15	0	0	0	0	0	0	0
21:30	0	0	0	0	0	0	0
21:45	3	0	0	-1	-9	2	2
22:00	3	0	0	-1	-9	0	0
22:15	3	0	0	-1	-9	0	0
22:30	3	0	0	-1	-8	0	0
22:45	3	0	0	-1	-8	0	0
23:00	3	0	0	-1	-8	0	0
23:15	3	0	0	-1	-9	0	0
23:30	3	0	0	-1	-9	0	0
23:45	3	0	0	-1	-9	0	0

Le serveur HTTP propose une page « Metering » qui affiche toutes les informations d'énergies, ainsi que les puissances en temps réel (à la seconde).



The screenshot shows a web-based metering interface with two main sections: 'Power' and 'Energy today' on the left, and 'Energy History' and 'Energy total' on the right.

Power		Energy today	
Voltage RMS	240 V	Date	2024/1/29 -
P1 Real	128 W	Imported	4111 Wh
P1 Apparent	268 VA	Exported	0 Wh
Cos Phi 1	0.477 -	Div1(est)	0 Wh
P2 Real	0 W	Div2(est)	4 Wh
P2 Apparent	3 VA	Div2	4 Wh
Cos Phi 2	0.078 -	PV	384 Wh
P3 Real	151 W	Cnt1	0 Wh
P3 Apparent	169 VA	Cnt2	0 Wh
Cos Phi 3	0.890 -		
P Div (est)	0 W		
Counter 1	0 W		
Counter 2	0 W		

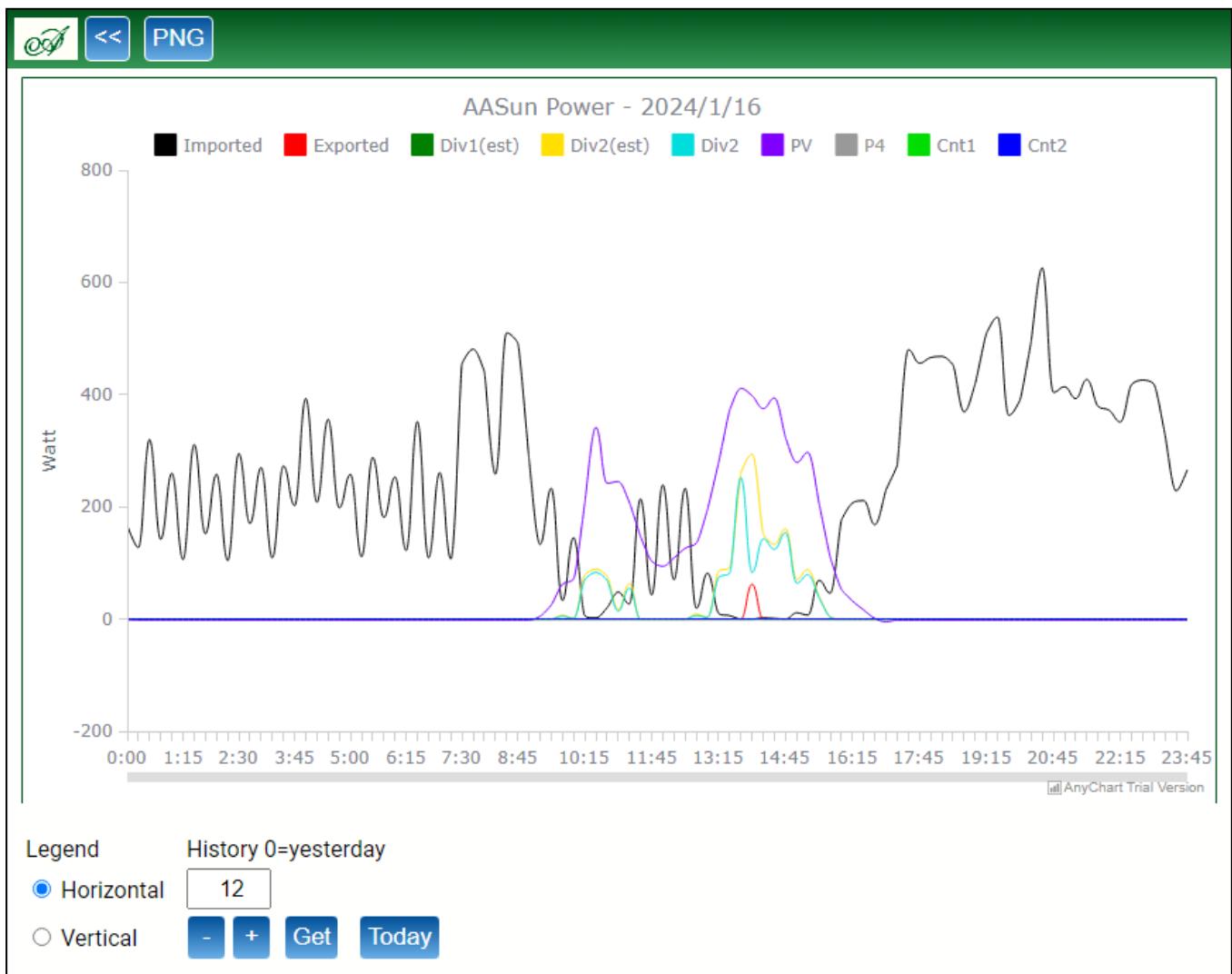
Energy History		Energy total	
Date	2024/1/28 -	Date	2023/12/1 -
Imported	7714 Wh	Imported	470978 Wh
Exported	0 Wh	Exported	17 Wh
Div1(est)	0 Wh	Div1(est)	0 Wh
Div2(est)	5 Wh	Div2(est)	3164 Wh
Div2	8 Wh	Div2	2838 Wh
PV	1127 Wh	PV	33791 Wh
Cnt1	0 Wh	Cnt1	2242 Wh
Cnt2	0 Wh	Cnt2	2 Wh

Buttons at the bottom left: History 0=yesterday, 0, -, +, Get. Button at the bottom right: Reset.

AASun - AdAstra-Soft

Note : Dans cet exemple l'application a été configurée pour ne pas gérer CT4.

Une autre page affiche l'historique journalier des puissances sous forme graphique :



Pour faciliter l'analyse on dispose de quelques outils :

- Zoom horizontal
- Cacher une courbe en cliquant sur sa légende
- Mettre une courbe en surbrillance en cliquant dessus
- En plaçant le curseur sur un point d'une courbe, un marqueur apparaît avec la valeur numérique de chaque courbe à cette abscisse.

Les boutons +, - et Get permettent de se promener dans l'historique.

Le bouton Today permet d'actualiser les courbes du jour.

11 Anti légionellose

AASun propose un dispositif pour limiter la prolifération dans un chauffe-eau sanitaire des bactéries Legionella responsables de la légionellose.

Le principe utilisé repose sur le fait suivant :

En fonction des températures, la durée nécessaire pour diminuer d'un facteur 10 la concentration des légionnelles planctoniques, non adhérentes à une surface, est de l'ordre de 20 mn à 55 °C, 2 mn à 60 °C ;

https://www.inrs.fr/dms/eficatt/FicheEficatt/EFICATT_L%C3%A9gionellose-1/Fiche_L%C3%A9gionellose.pdf

Pour lutter contre la prolifération des légionnelles il faut donc s'assurer que le CES atteigne régulièrement une température élevée, supérieure à 55°C.

Le dispositif anti-légionellose est réalisé avec trois éléments :

- Un capteur de température placé sur le CES, ou un thermostat mécanique avec une température suffisante connecté sur une entrée de AASun.
- Un compteur géré par AASun qui s'incrémentera tous les jours à minuit. Il est remis à 0 à tout moment si la température mesurée atteint le seuil fixée par l'utilisateur, ou si le thermostat commute. Ce compteur est V2.
- Une règle de forçage mise en place par l'utilisateur pour 'booster' la température du CES quand le compteur atteint le nombre de jours indiqué. Ce forçage permet de choisir le moment, la durée et la périodicité du 'boost'.

Le forçage permet donc de déclencher périodiquement une chauffe du CES si durant son utilisation normale une température seuil n'a pas été atteinte pendant cette période.

Exemple de forçage pour cette utilisation :

START : OUT2 & STD & MH = 2230 & V2 > 6

STOP : DMAX = 120M & V2 < 6

Chaque jour à 22h30 le compteur V2 est testée, et si la température de seuil n'a pas été atteinte depuis 7 jours ou plus, alors le forçage est activé. Quand la température de seuil est atteinte le compteur V2 est remis à 0 et par conséquent la règle STOP devient vraie, ce qui désactive le forçage.

Le forçage s'arrête aussi au bout de 2 heures même si la température de seuil n'est pas atteinte. Dans ce cas le forçage recommencera le lendemain.

Pour réaliser le test seulement le vendredi à 23h :

START : OUT2 & STD & WD =V. & MH = 2300 & V2 > 6

STOP : DMAX = 120M & V2 < 6

Ne pas oublier de décocher la case Disabled !

L'anti-légionellose est configuré à partir de la console avec la commande `cal`, exemple :

```
cal t1 55
```

L'anti-légionellose est activé en utilisant T1, et la température de seuil qui remet le compteur à 0 est de 55°C.

```
cal i3 1
```

L'anti-légionellose est activé en utilisant l'entrée I3, et le compteur est remis à 0 quand l'entrée vaut 1.

Pour inhiber le système anti-légionellose :

```
cal 0
```

Utiliser `cwr` pour sauver cette configuration dans la flash.

Pour tester le fonctionnement du dispositif il est possible de donner une valeur au compteur V2. Par exemple pour le mettre à 12 :

```
cal c 12
```

Cela peut aussi être fait à partie de la page HTTP « Variables ».

Variables		Anti-legionella	
V1 Day counter	16	Choose:	Off
V2 Anti-legionella	16	Value:	Send
V3	0		
V4	0		
Choose:	V1	Value:	Send

AASun - AdAstra-Soft

12 Mise en FLASH des pages HTTP

Pour flasher les ressources du serveur HTTP, il y a deux moyens :

- Utiliser une sonde ST-LINK
- Utiliser un adaptateur USB/UART

12.1.1 Mise en flash avec ST-LINK

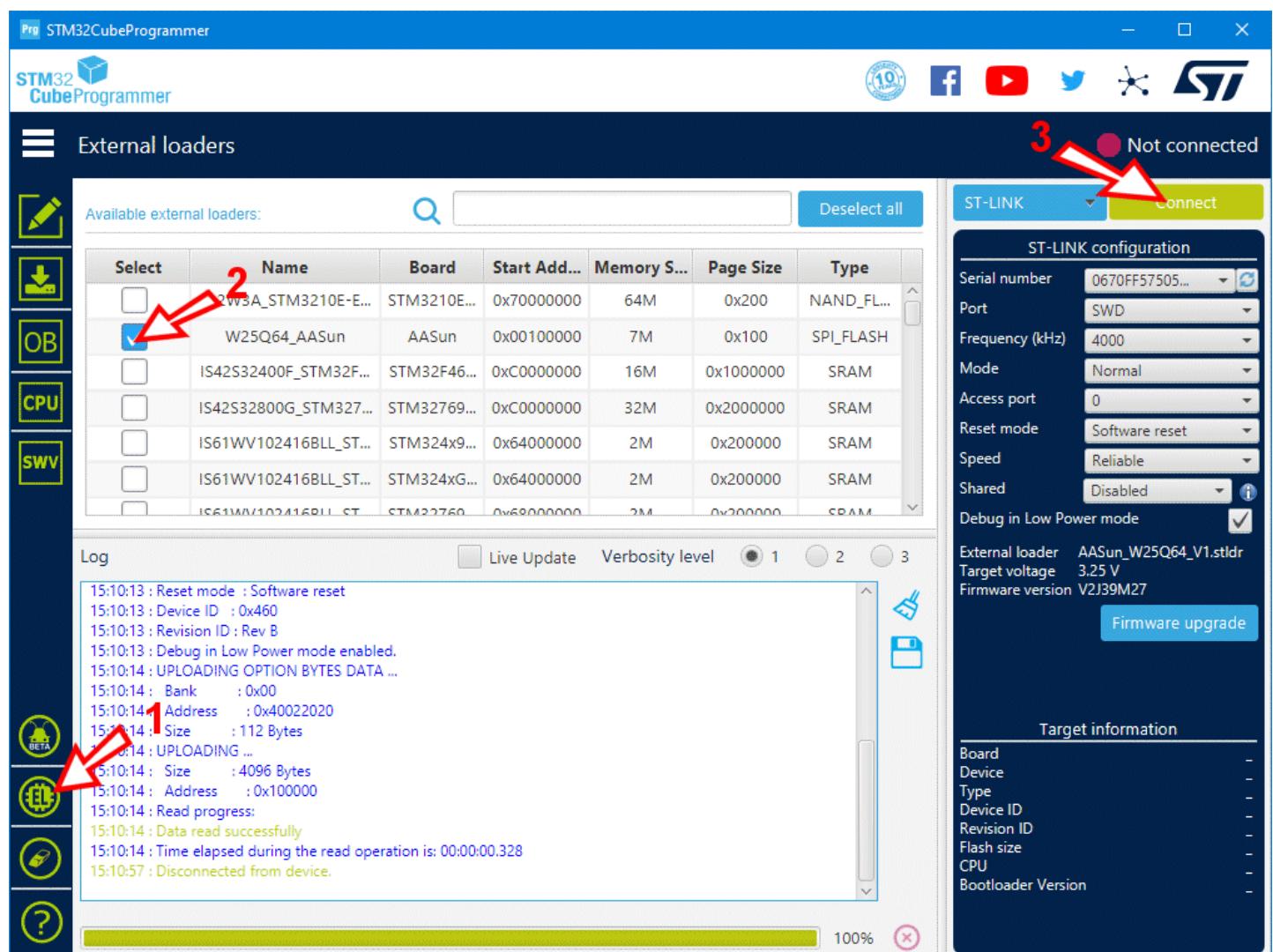
Cette procédure utilise STM32CubeProgrammer et un « External loader » développé spécialement pour AASun en tenant compte du type de MCU, du type de mémoire Flash, du SPI utilisé et des broches GPIO associées.

Ce fichier *AASun_W25Q64_V1_1.stldr* est à placer dans :

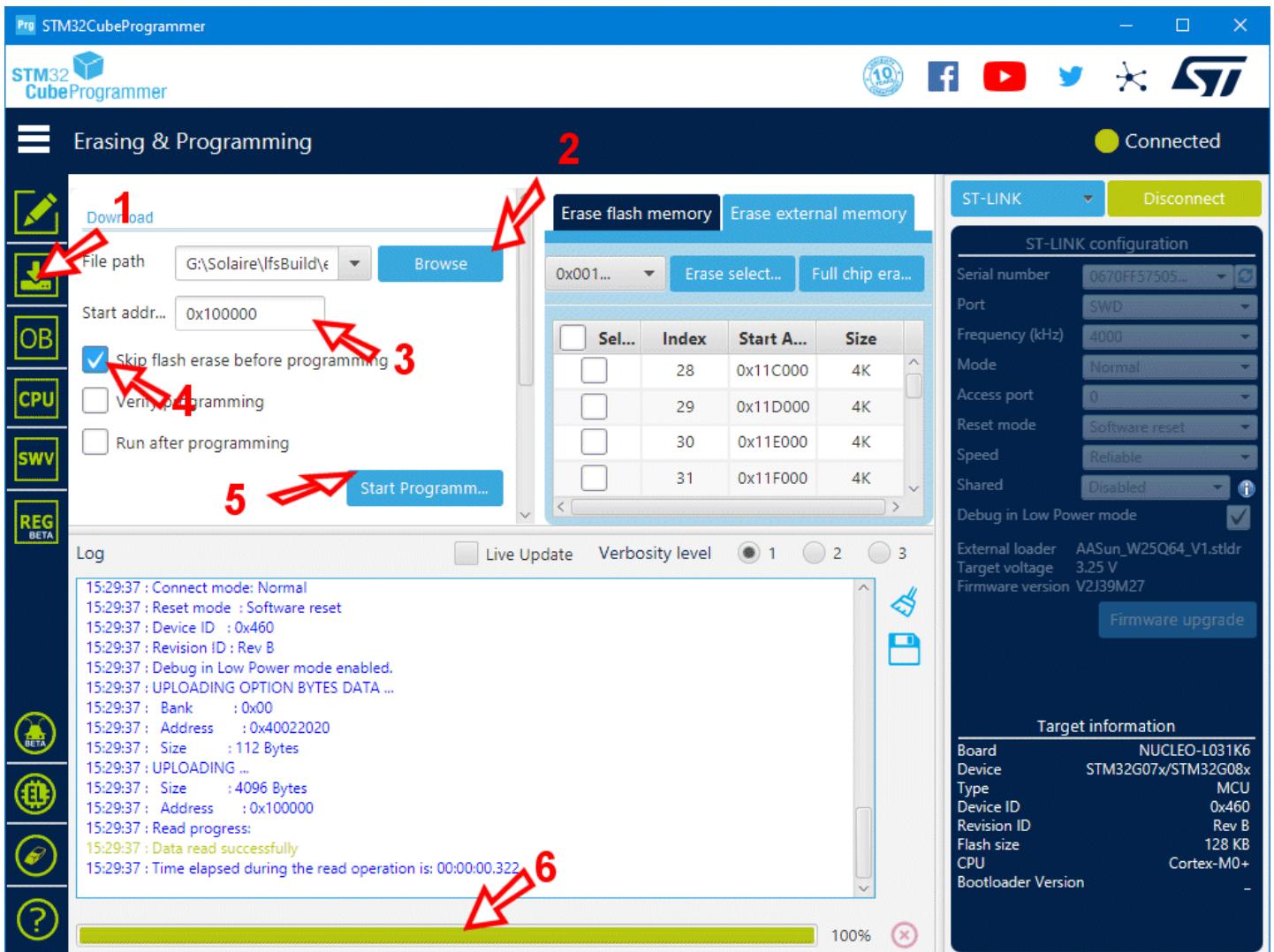
C:\Program Files\STMicroelectronics\STM32Cube\STM32CubeProgrammer\bin\ExternalLoader

Connecter la sonde au connecteur J5 du module MCU, et sur un port USB du PC.

Lancer STM32CubeProgrammer.



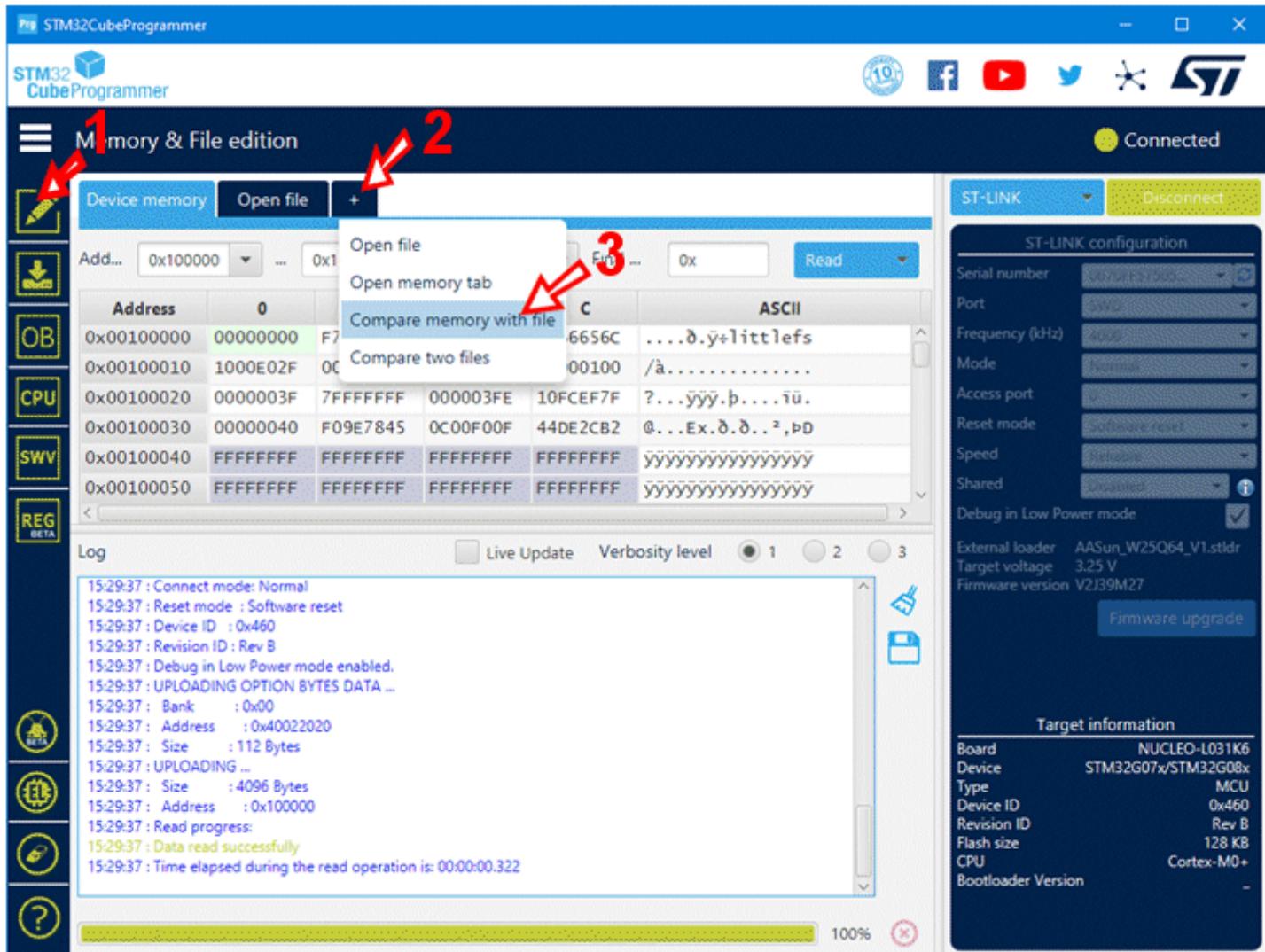
- 1 Cliquer le bouton EL pour afficher la liste des « External Loader »
- 2 Sélectionner W25Q64_AASun_V1.1.
- 3 Après avoir sélectionné ST-LINK, cliquer le bouton « Connect », une fois le logiciel connecté au MCU il affiche « Disconnect »



Pour flasher le fichier des ressources du serveur :

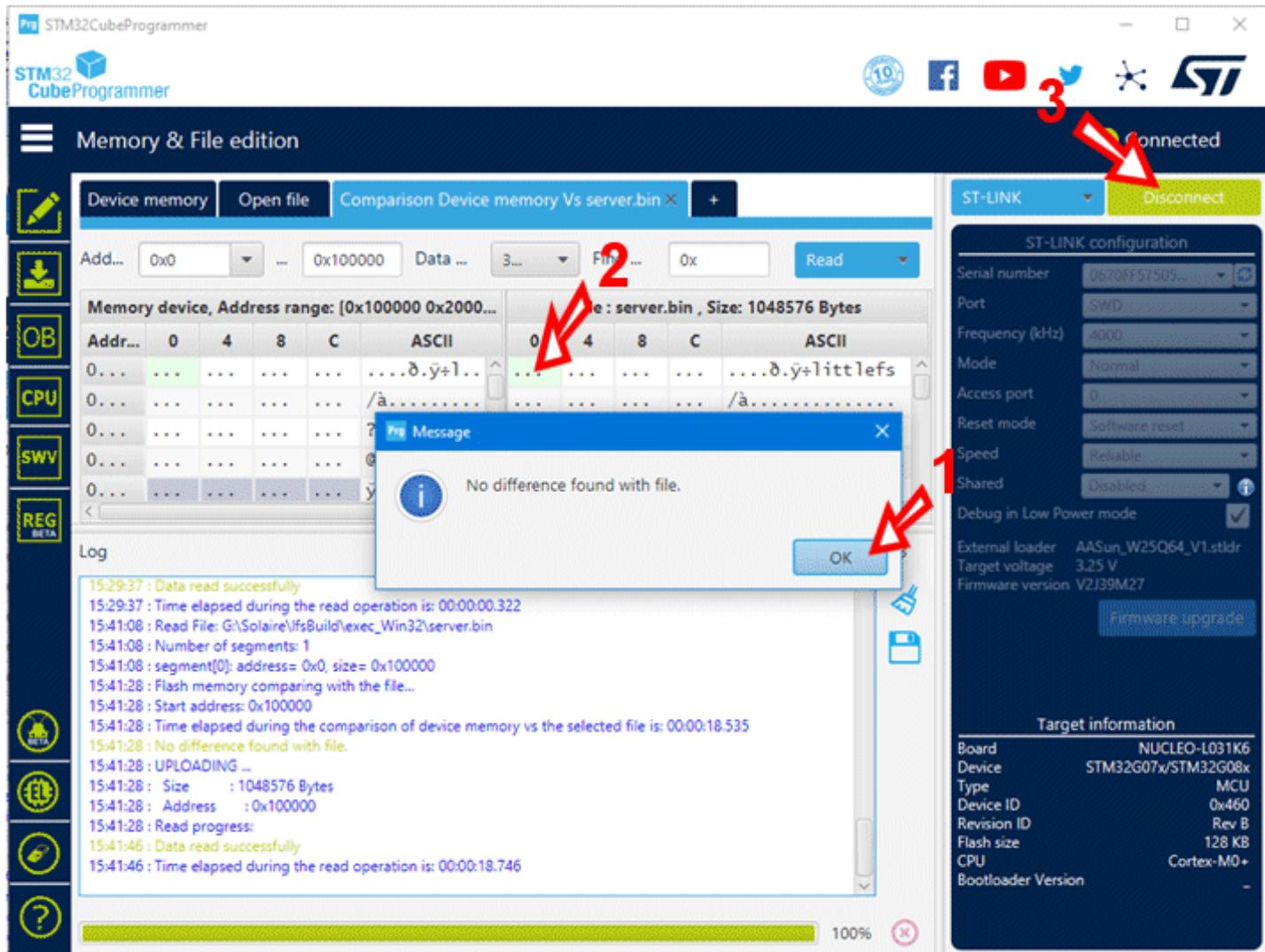
- 1 Cliquer le bouton de la configuration du téléchargement
- 2 Sélectionner le fichier à télécharger
- 3 Configurer l'adresse de chargement à 0x100000
- 4 Découcher la case « Skip flash erase before programming » : on veut effacer la Flash avant programmation
- 5 Lancer la programmation
- 6 Surveiller le déroulement de l'opération. Pour plus de détails on peut ajuster le « Verbosity level »

C'est fini, les ressources du serveur sont en Flash.



Le chargeur externe n'a pas de code pour vérifier le téléchargement. Mais on peut utiliser STM32CubeProgrammer pour cela.

- 1 Afficher la fenêtre d'inspection
- 2 Dans le menu des options
- 3 Sélectionner « Compare memory with file », puis sélectionner le fichier.



- 1 A la fin de la comparaison le résultat est affiché
- 2 Deux fenêtres permettent de comparer visuellement les contenus de la mémoire et du fichier.
- 3 Déconnecter la sonde ST-LINK avant de quitter STM32CubeProgrammer.

Déconnecter la sonde ST-LINK, et presser le RESET du MCU pour lancer AASun.

12.1.2 Mise en flash avec un adaptateur USB/UART

Le logiciel AASun contient une fonction pour écrire dans la flash externe des données provenant de la liaison série (qui sert aussi de console). Une application sur PC a été développée pour envoyer le fichier des pages HTTP au routeur : SerEL (**S**erial **E**xternal **L**oader).

SerEL admet des paramètres pour mener à bien sa fonction :

- c Numéro du port COM à utiliser
- b Baudrate, le même que celui utilisé avec la console PuTTY (par défaut 57600)
- a Adresse dans la flash externe : 0x100000 (valeur par défaut)

- f Le chemin du fichier à envoyer
- v Si cette option est ajoutée le fichier n'est pas envoyé mais il est comparé au contenu de la flash

Il faut que la liaison série soit libre : fermer la console si elle est connectée.

Pendant la programmation le routeur est arrêté, et à la fin de la programmation ou de la vérification il redémarre automatiquement avec un reset.

Exemple pour envoyer le fichier (il y a toujours une vérification après l'écriture) :

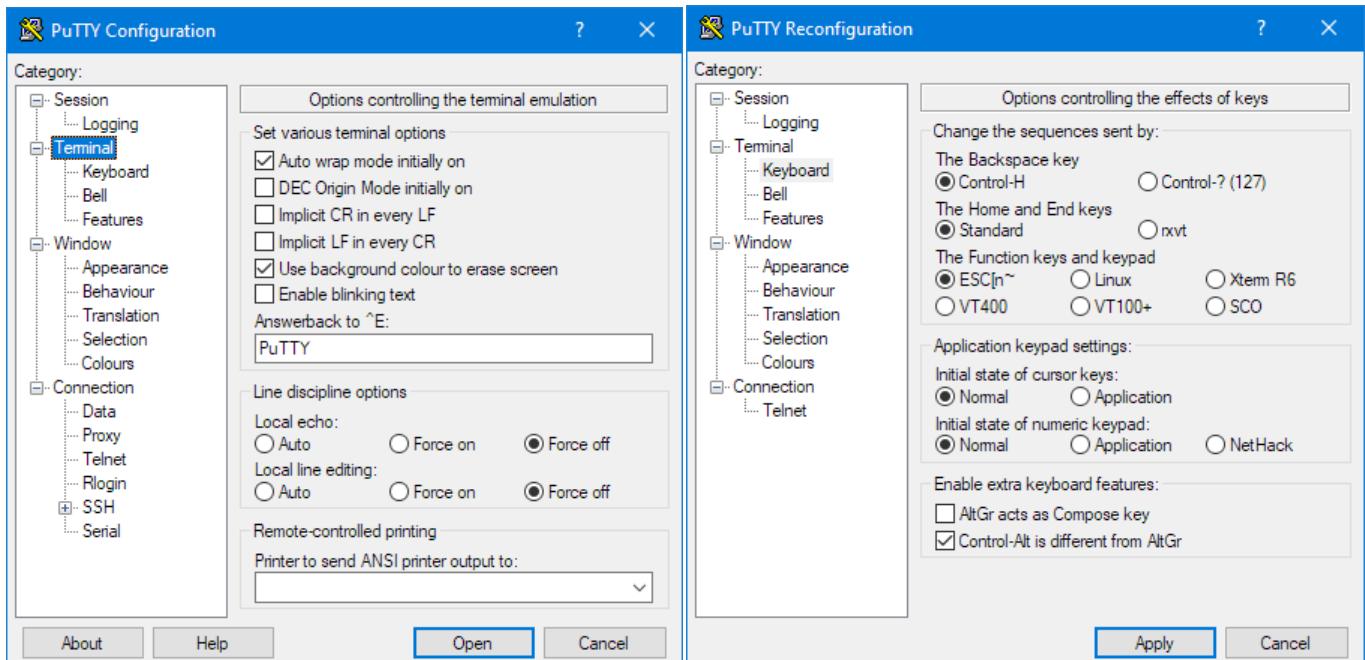
```
SerEl -c 9 -b 57400 -a 0x100000 -f AASun_web.bin
```

Exemple pour seulement vérifier si le fichier correspond à ce qui est en flash :

```
SerEl -v -c 9 -b 57400 -a 0x100000 -f AASun_web.bin
```

13 Telnet

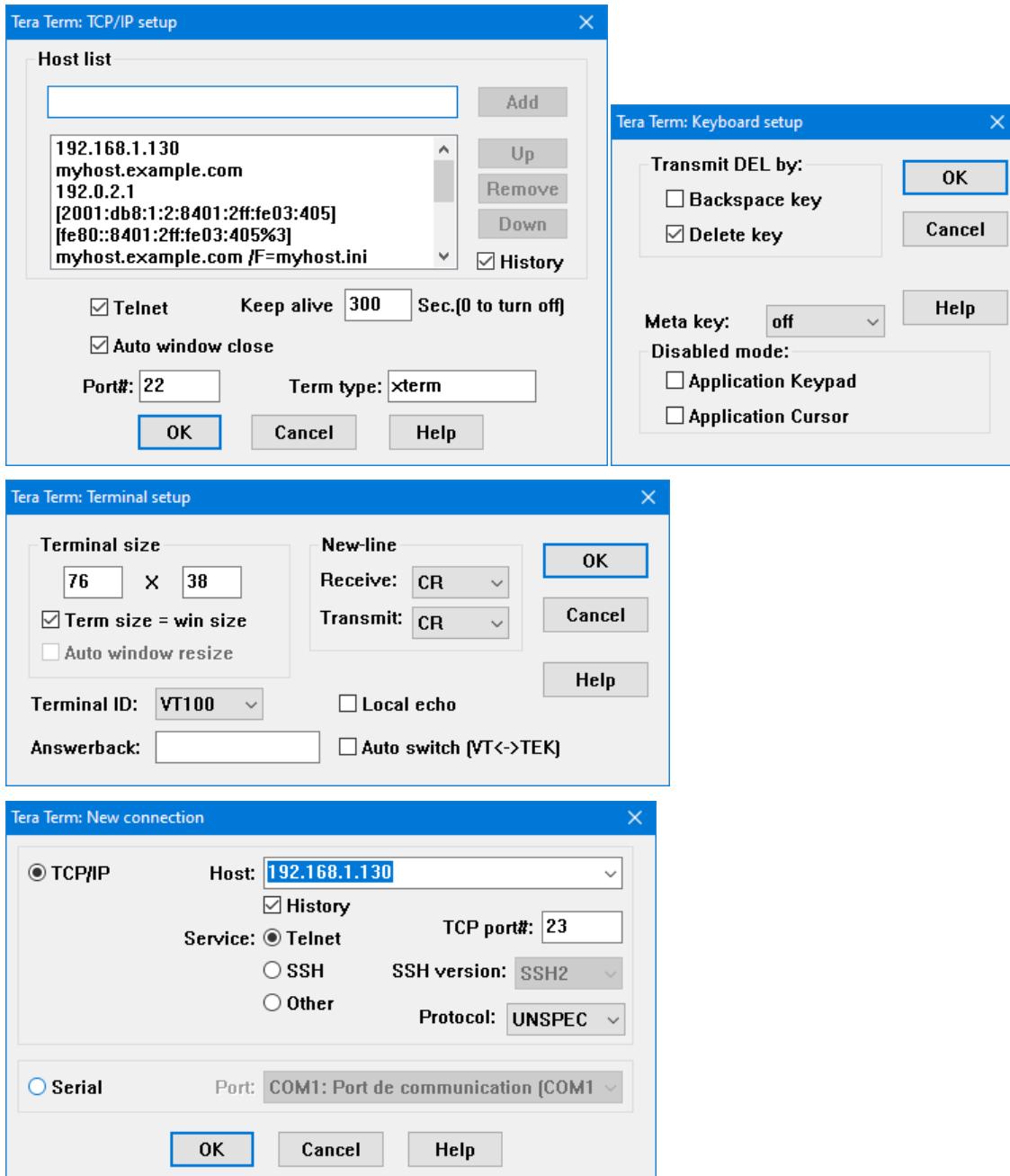
Il est possible d'utiliser Telnet pour déporter la console de AASun sur réseau, et rendre tous les outils de diagnostic et de configuration disponibles. Le logiciel PuTTY est très pratique, voici la configuration à utiliser :



Pour utiliser MS Telnet :

```
Microsoft Telnet> d
Le caractère d'échappement est 'CTRL+$'
Va authentifier (Authentification NTLM)
Écho local désactivé
Mode nouvelle ligne - force la touche Retour à envoyer CR & LF
Mode actuel : Console
Négociera le type de terminal
Le type de terminal préféré est ANSI
```

Pour TeraTerm :



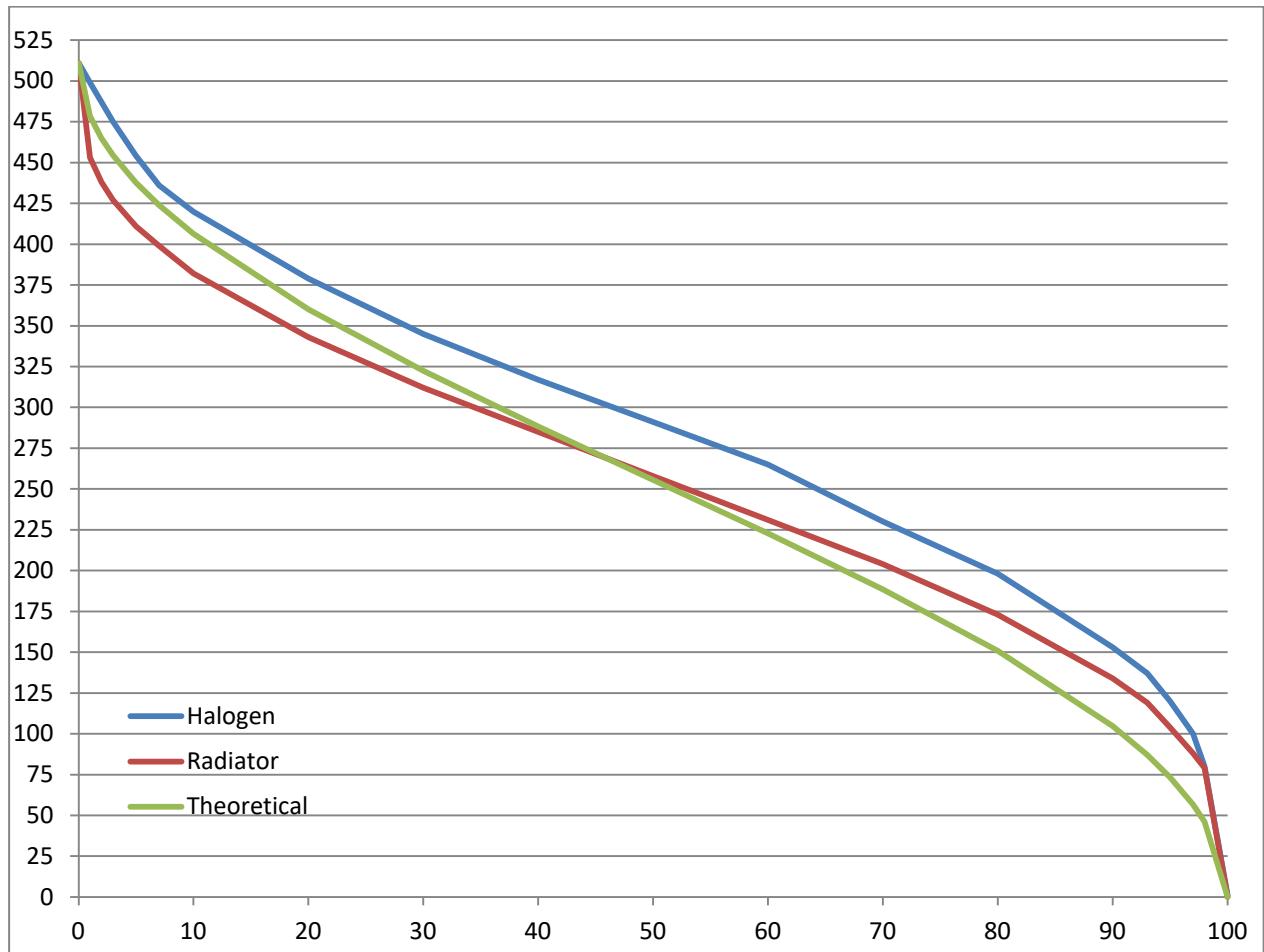
14 Conversion puissance – délai

L'algorithme de routage utilise une table pour convertir la puissance à router en délai de déclenchement du SSR (voir §2.4 [Commande SSR](#)).

Mon banc de test utilise des ampoules halogènes pour simuler la production photovoltaïque et la charge de routage. Les tests ont montré qu'il y avait un écart entre le délai obtenu avec l'algorithme et le délai théorique correspondant à la puissance réelle à router. De plus le contrôleur PI avait tendance à osciller un peu avant de se stabiliser.

J'ai donc fait un relevé pour construire la courbe puissance/délai pour les ampoules halogène : elle est différente de la courbe théorique.

Qu'en est-il avec une résistance type chauffe eau ? Je n'ai pas de chauffe eau électrique j'ai donc utilisé mon radiateur à bain d'huile qui doit être approchant : la courbe est encore différente... Et les valeurs varient un peu si on fait les mesures radiateur chaud ou froid...



En X la puissance en % de Pmax, en Y le délai correspondant entre 0 et 511.

La table puissance/délai a donc une influence sur la stabilité du contrôleur PI de routage et sur l'estimation de la puissance routée.

Pour un fonctionnement optimal du routeur, il faut utiliser une table puissance/délai adaptée au type de charge utilisé. Si la courbe n'est pas tout à fait adaptée le routage sera quand même correct grâce au contrôleur PI qui s'adaptera (pas d'injection), mais cela pourra prendre quelques $\frac{1}{2}$ périodes secteur de plus, et l'estimation de puissance routée sera légèrement fausse.

14.1 Ampoule halogène

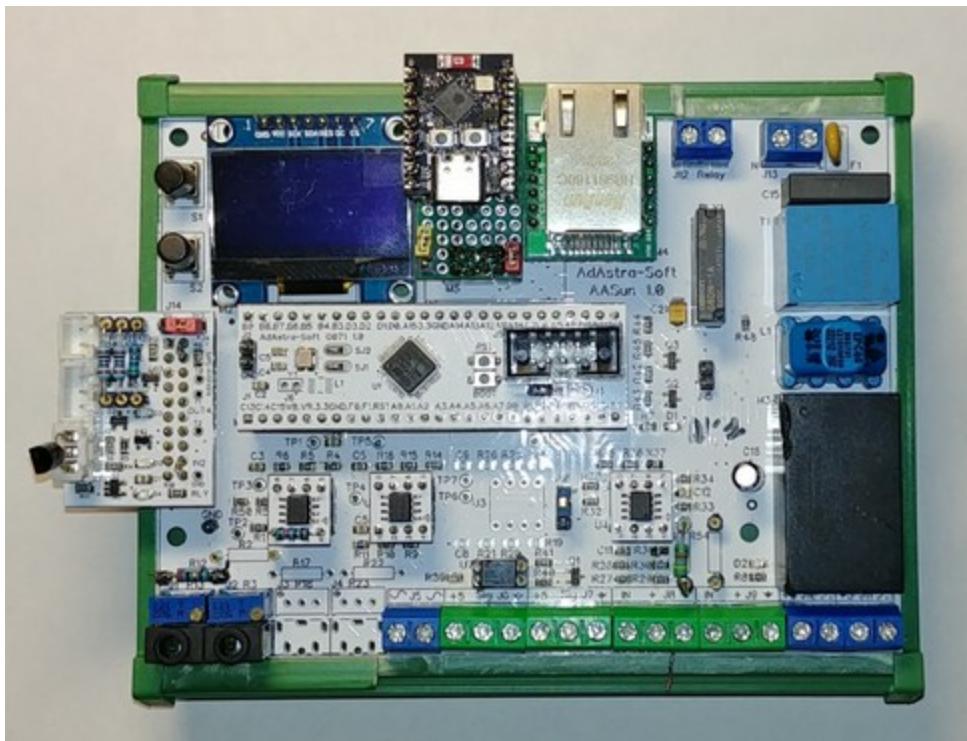
Le comportement des ampoules halogènes de mon banc de test a une explication.

Ce sont des ampoules à filament qui chauffent à blanc pour éclairer. La résistance à froid et à chaud du filament varie fortement : elle est ~ 15 fois plus faible à température ambiante qu'à température de fonctionnement ($\sim 3000^{\circ}\text{C}$).

L'hypothèse de départ d'une résistance de charge de valeur fixe n'est donc pas du tout respectée et cela perturbe le contrôleur PI. L'halogène est donc un très mauvais choix pour faire un banc de test destiné à faire des mesures. Par contre il est très pratique pour le développement du routeur :

- il loge sur ma table
- contrôle immédiat puisque visuel
- pas cher (à condition d'avoir un stock d'ampoules depuis quelques années...).
- consomme peu.

15 Défauts et améliorations



Le prototype avec presque toutes les options

Dans ce chapitre un aperçu de ce qui pourrait être corrigé ou évoluer.

Evolutions du matériel:

- Erreur sur le schéma => un fil à ajouter sur le PCB (R4-R5).
- Erreur sur le PCB : la FLASH W25Q64 n'est pas en SO8 !
- L'interface pour la mesure de température avec un DS18b20 est mal conçue (Je n'ai pas lu la doc avec attention avant de faire le schéma...). Ce capteur est plus facile à utiliser avec un UART.
- LM393 Il faut inverser les entrées + et – pour ne pas inverser la polarité de l'entrée.
- Alimentation sur les borniers d'entrée des compteurs d'impulsion de 5V : Il faut au moins 12V pour la sortie de comptage de certains compteurs.
- Blindage des câbles des CT : ajouter de quoi relier ces blindages à la terre ?
- Ajouter la terre au bornier du connecteur secteur.
- Relais et SSR 2 sur la même I/O : complication et limitation inutile.
- Linky et IN4 sur la même I/O : complication et limitation inutile. Partiellement corrigée en déplaçant le Linky sur PB10 (avec un fil) ce qui permet d'avoir Linky et IN4 en même temps (au détriment de J7 qui ne peut plus être utilisé, mais de toute façon le DS18b20 ne passe plus par là !).
- Est-il nécessaire de pouvoir utiliser le WIFI et le réseau filaire en même temps ?

- Pas besoin d'amplification pour la mesure de la tension. Cela peut permettre d'économiser un AOP, ou d'avoir une mesure de courant en plus :
Résistance primaire de 66 kΩ, secondaire de 200Ω
=> courant de 3.56 mA, 0.84 W, 2 Vpp, avec peu de modification du déphasage (qui de toute façon est bien inférieur à celui des CT à clipser).
- Utiliser seulement la référence de tension externe TL431. Pas chère et permettant d'utiliser un STM32G070CB dont la référence de tension interne ne sort pas !!!!. Avoir 2 types de MCU utilisable peut faciliter les approvisionnements.
Permettrait aussi de mesurer jusqu'à 3V en ADC pour améliorer la précision, ou simplifier le conditionnement. Les CT sortant 1Vrms seraient utilisables directement, mais la mesure de tension dégagerais plus de 1W dans le primaire.
- Carte un peu encombrante : peut être la réaliser en 10x10cm avec un PCB 4 couches.
- Les borniers 5mm utilisés ne sont pas adaptés aux boîtiers rail DIN et sont encombrants alors qu'il n'y a pas de puissance à passer. Passer à des borniers 3.96 débranchables ?



- Si nouvelle carte, utiliser un STM32U535RC ou un STM32H503 qui est sorti après le début de cette réalisation (plus performant et ~même prix...). Mais ils n'ont pas d'entrée VREF+ => difficulté avec la partie analogique ?
- Utiliser quartz LSE + RTC + batterie pour horloge interne ? (Si pas d'internet).

Evolutions logicielles :

- ~~Mettre en route le module ESP-01S pour le WIFI~~ Remplacé par ESP32-C3 Super Mini
- ~~Petit serveur HTTP pour une interface sympa.~~
- API pour une domotique (je n'en ai pas, donc...)
- ~~Utilisation du relais mécanique~~
- Ajout relais mécanique déporté sur réseau
- ~~Deuxième sortie SSR (avec priorité)~~
- ~~Gestion de modes boost manuel/périodique~~
- Autres fonctionnalité ?