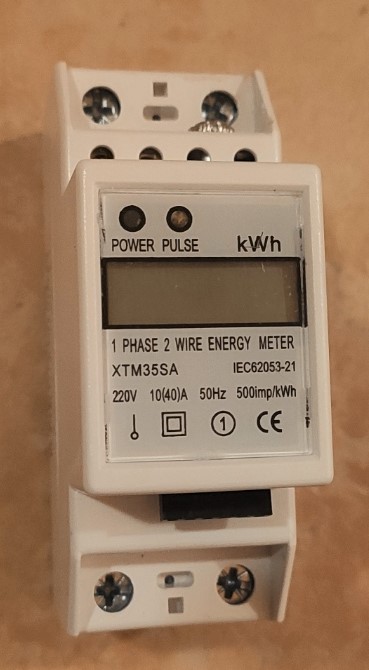
**Routeur Tignous**

J. Lemaire



Pierrefeu Février 2022

# Introduction

Le Forum photovoltaïque[[1]](#footnote-2) présente un routeur, dénommé « optimiseur » ou « Routeur Tignous » qui permet de détecter un surplus de production photovoltaïque monophasée, et de le router vers une charge :

1. purement résistive, avec contrôle de phase (S1)
2. quelconque, en tout ou rien, dès saturation de la première (S2).

La détection du surplus est assurée par un système composé :

* d’une pince ampèremétrique, à installer autour d’un des câbles (Phase ou Neutre), juste après le compteur d’électricité
* d’un petit circuit électronique, basé sur un microcontrôleur ATmega328P

Ce circuit peut se loger dans un compteur d’énergie électrique à bas coût, qui assure son alimentation en 5V continu et affiche l’énergie du courant routé.

Il commande 2 relais solides (SSR), également peu coûteux, pour l’alimentation des charges.

Pour une cinquantaine d’euros, on peut donc disposer d’un routeur photovoltaïque, programmable avec l’IDE Arduino par exemple et installable sur rails DIN dans un coffret électrique (**9** modules).

Bien décrit et très largement commenté sur le Forum Photovoltaïque, on peut l’obtenir en kit ou tout assemblé sur ce site qui fournit aussi les sources des programmes. A souligner ici toute la richesse des informations disponibles sur ce forum où les concepteurs (Tignous84, Rolrider) de ce routeur répondent rapidement.

On pourra cependant regretter l’absence d’isolation galvanique, certes contournée ici en utilisant un branchement inversé (Phase⬄ Neutre) du compteur, mais qui peut rendre dangereuse l’utilisation d’un moniteur série pour le monitoring ou la mise au point. On peut aussi craindre un échauffement du premier relais, à cause des harmoniques dues au mode « contrôle de phase ». Des modifications ont été proposées pour y pallier.

# Composants

## Kit optimiseur

**Rolrider** (Pascal Gourdon) sur Forum Photovoltaïque – 16€

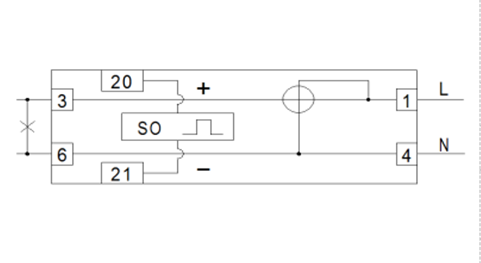
* Circuits imprimés
* Composants à souder
* MOC 3023 pour relais solde (cf. 3.2)

## Compteur d’énergie XTM 35SA[[2]](#footnote-3) [[3]](#footnote-4)

# 

**Caractéristiques**

* Numéro de phase : monophasé
* Mode d'affichage : LCD
* Courant de référence : 10(40)A
* Tension nominale : 220V
* Tension standard : 120/230V AC
* Tension de fonctionnement : 84-156 V AC/161-300 V AC
* Capacité d'isolation :
* Tenue à la tension alternative : 4 KV pendant 1 minute
* Tenue à la tension d'impulsion : 6 KV - forme d'onde 1,2 μS
* Plage de courant de fonctionnement : 0,04 A-60 A/0,08 A-100 A
* Tenue aux surintensités : 1800 A/3000 A pendant 0,01 s
* Plage de fréquence de fonctionnement : 50 Hz ±10 %
* Consommation électrique interne : ≤2W/10VA
* Taux de flash de sortie de test (LED ROUGE) : 500 impulsions par kWh[[4]](#footnote-5)
* Taux de sortie d'impulsions (broches 20 21) : 500 impulsions par kWh
* Indicateur d'alimentation (LED verte) : le compteur est connecté à 120 ou 230 V/50 Hz
* Écran LCD rétroéclairé : 7+1 chiffres
* Capacité maximale du câble : 25 mm²
* Humidité de fonctionnement : ≤85 %
* Humidité de stockage : ≤95 %
* Température de fonctionnement : -10℃ - +50℃
* Température de stockage : -30℃ - +70℃
* Classe de précision : 1
* Niveau de protection : IP53
* Couleur : Blanc
* Taille : env. 14 x 7 x 3,2 cm/5,5 x 2,7 x 1,2 pouces
* Poids : 179 g



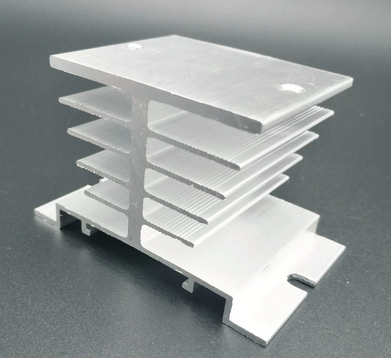
**TRES IMPORTANT**

Ici L sera connecté au Neutre[[5]](#footnote-6) et N à la Phase.

Cette inversion est justifiée ici car la basse tension est créée avec un condensateur et une diode Zener, sans séparation galvanique donc[[6]](#footnote-7).

Comme le Neutre est connecté à la terre, il sera utilisé comme référence du GND dans les circuits de commande.

## Relais solide[[7]](#footnote-8) [[8]](#footnote-9)

* 1 par voie
* Pour courant AC 24-380V
* Commande DC 3-32V
* Diode
* Radiateur possible pour voie 1 (si échauffement à cause des harmoniques)

## Pince ampèremétrique SCT-013-050[[9]](#footnote-10) [[10]](#footnote-11) [[11]](#footnote-12)



# Montage du routeur[[12]](#footnote-13) [[13]](#footnote-14)

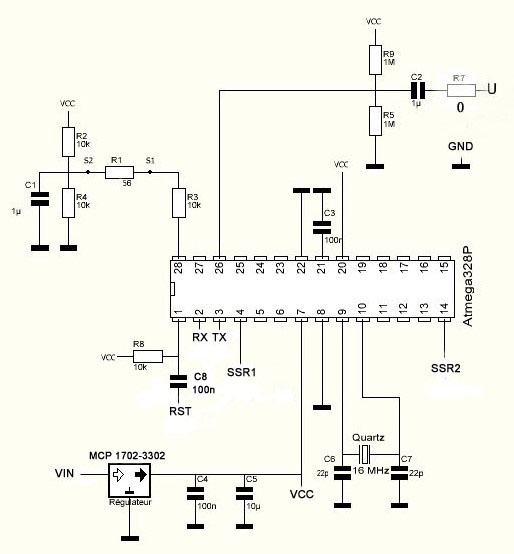
## Module commande

Il comprend 2 petites cartes installées dans le compteur XTM 35SA :

* une carte principale basée sur un microcontrôleur **ATmega328P**
* une carte pour la liaison FTDI

### Carte principale

#### Schéma



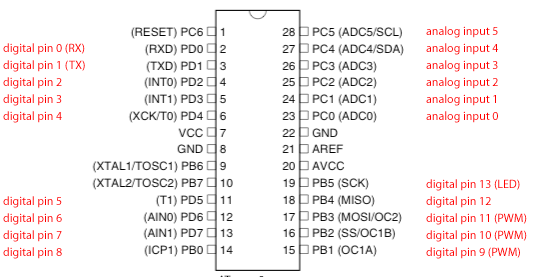
Entrées :

* GND et VIN = 5V DC fournis par le compteur XTM 35SA
* S1, S2 => pince ampèremétrique

Sorties :

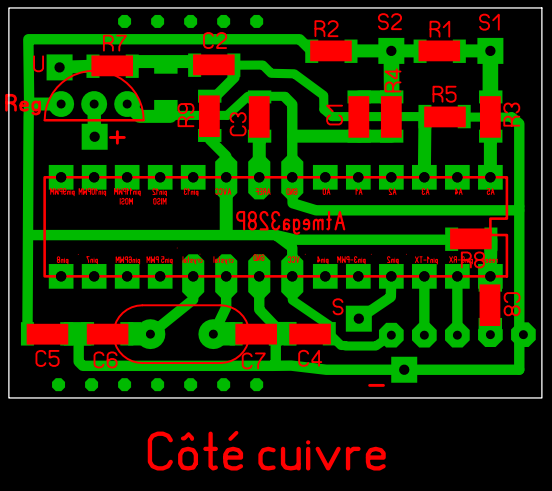
* SSR1, SSR2 => relais solides
* RST, RX, TX, VCC, GND => carte FTDI

#### ATmega328P[[14]](#footnote-15)



Consommation 15mA[[15]](#footnote-16)

#### PCB[[16]](#footnote-17)

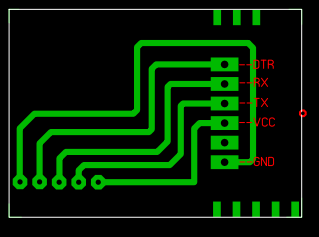


#### Composants

|  |  |  |
| --- | --- | --- |
| **Label** | **Type** | **Caractéristiques** |
| R1 | Résistance | 56Ω |
| R2 | Résistance | 10kΩ |
| R3 | Résistance | 10kΩ |
| R4 | Résistance | 10kΩ |
| R5 | Résistance | 1MΩ |
| R7 | Résistance | 0Ω |
| R8 | Résistance | 10kΩ |
| R9 | Résistance | 1MΩ |
| C1 | Condensateur | 1µF |
| C2 | Condensateur | 1µF |
| C3 | Condensateur | 100nF |
| C4 | Condensateur | 100nF |
| C5 | Condensateur | 10µF |
| C6 | Condensateur | 22pF |
| C7 | Condensateur | 22pF |
| C8 | Condensateur | 100nF |
| VR1 | Régulateur 3,3V | MCP 1702-3302E/TO[[17]](#footnote-18) |
| IC1 | Microcontrôleur | ATmega328P-U |
| X1 | Quartz | 16MHz |

### Carte pour la liaison FTDI

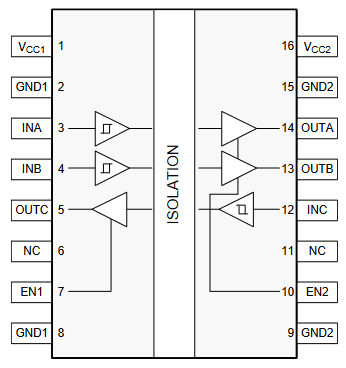
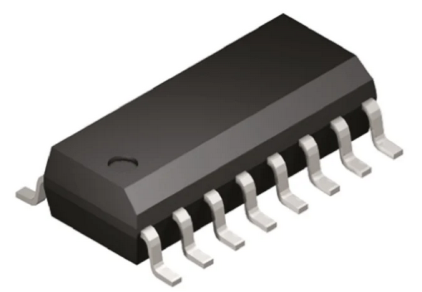
Le PCB fourni ici établit une liaison directe entre les 5 sorties RST, RX, TX, VCC, GND précédentes et un connecteur Dupont femelle à 6 entrées, où l’on pourra brancher un convertisseur USB=>Série TTL :



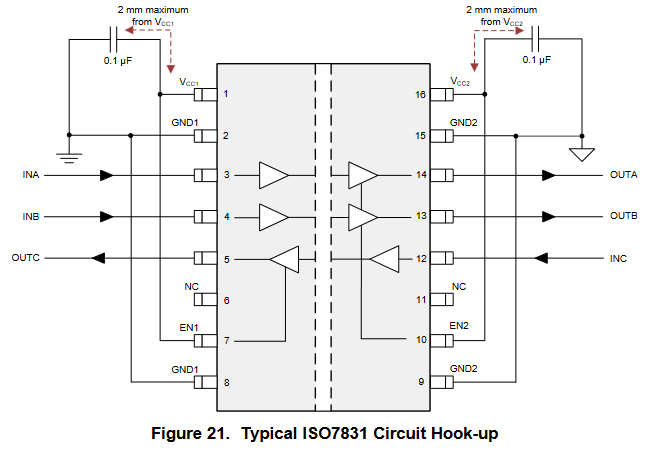
Le convertisseur pourra donc alimenter la carte principale en 3,3V et il est bien recommandé d’utiliser cette liaison SANS alimenter cette carte par le compteur XTM 35SA. On ne pourra donc pas l’utiliser pour monitorer son fonctionnement à chaud et ce d’autant plus que la carte principale ne dispose pas d’isolation galvanique par rapport au réseau EDF (GND est connecté au Neutre).

Ce problème a été résolu par **bgrigoriu** sur le **Forum photovoltaïque[[18]](#footnote-19)**, en utilisant un isolateur numérique ADUM1201[[19]](#footnote-20) (2 canaux directs) ou ADUM1301[[20]](#footnote-21) (3 canaux dont 1 inversé). On reprend ici sa solution à 3 canaux[[21]](#footnote-22), en remplaçant son PCB spécifique par une simple plaque d’essai de mêmes dimensions et l’ADUM 3101 par un composant Skyworks équivalent, soudé sur un support adaptateur SOIC16 => DIP16.

#### Isolateur numérique Skyworks Si8431AB-D-IS1[[22]](#footnote-23) [[23]](#footnote-24)



* 3 canaux dont 1 inversé
* 2,7-5.5V
* Format NSOIC16 (larg. = 6mm, esp. broches = 1.27mm)
* Isolation 2500V
* Montage avec condensateurs céramique 0,1µF

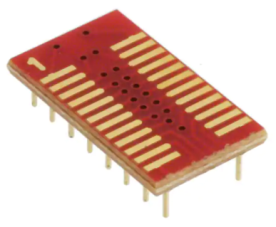


#### Condensateurs céramique 0,1µF Wurth Elektronik 885012207128[[24]](#footnote-25) [[25]](#footnote-26)



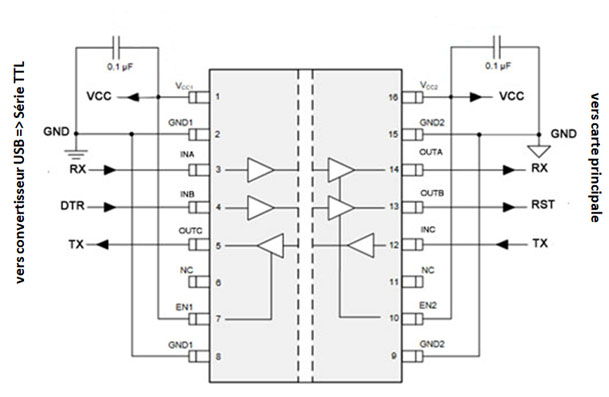
* 0,1µF
* CMS 805 => Long. = 2mm

#### Support adaptateur SOIC16 => DIP16 Aries 16-350000-11-RC[[26]](#footnote-27) [[27]](#footnote-28)

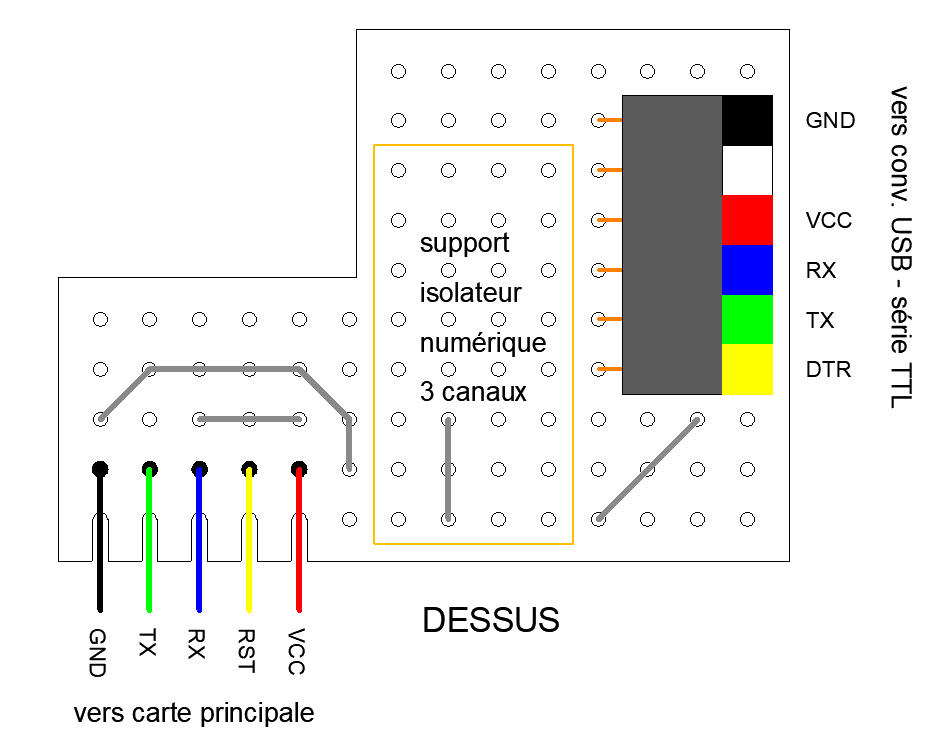


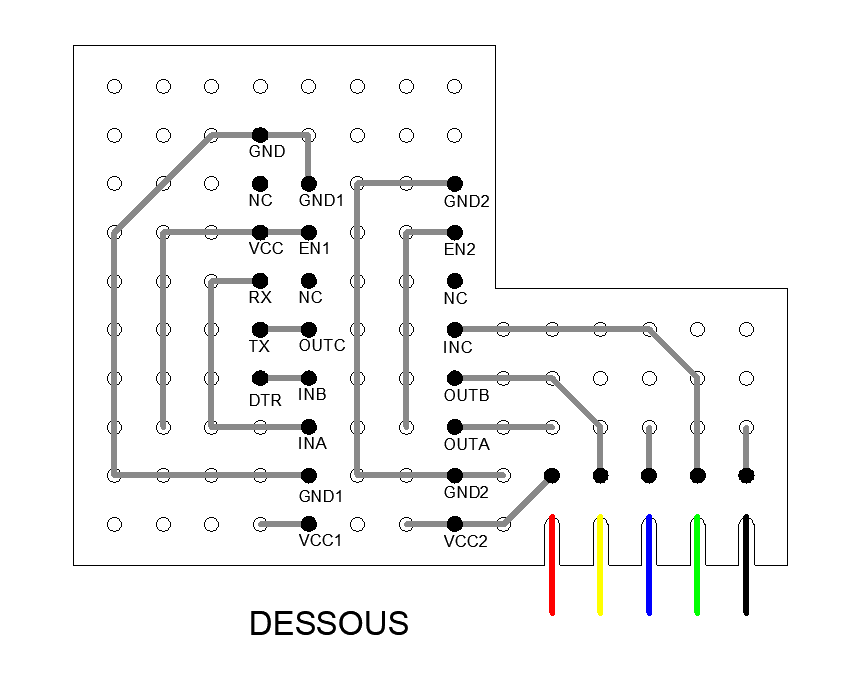
* Long. (pins) = 7x2,54 = 17,78mm
* Larg. (pins) ) = 3x2,54 = 7,62mm

#### Schéma



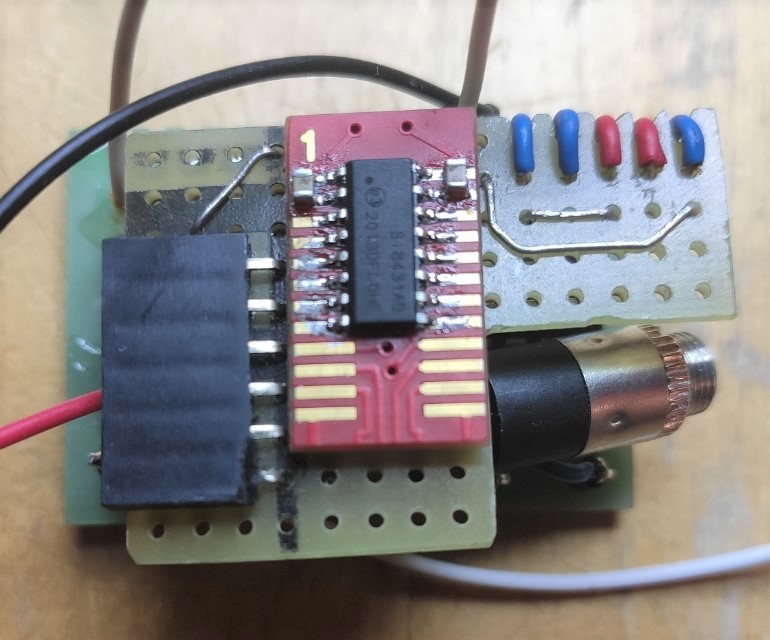
#### Plaque d’essai





#### Réalisation

L’isolateur numérique et les 2 condensateurs de 0,1µF sont soudés à la pâte à souder sur le support :

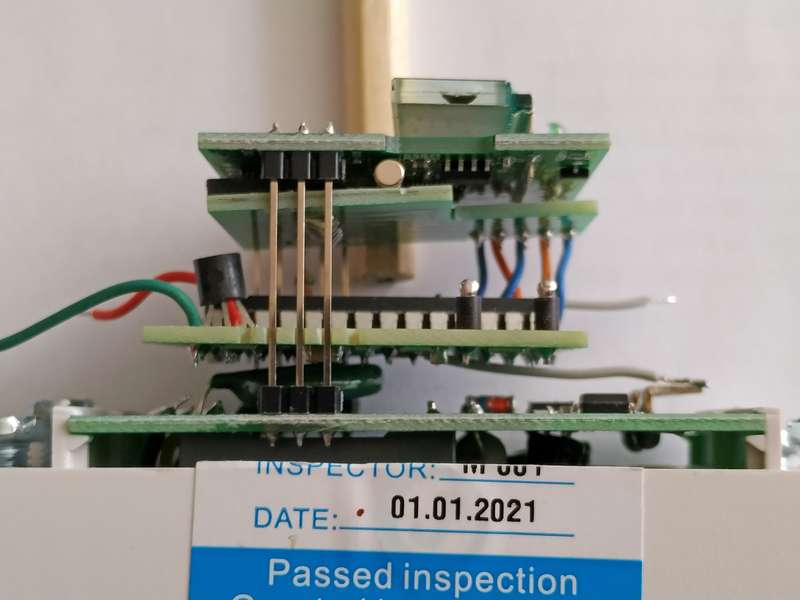


Et il faut croiser les fils RST et TX pour connecter les deux cartes.

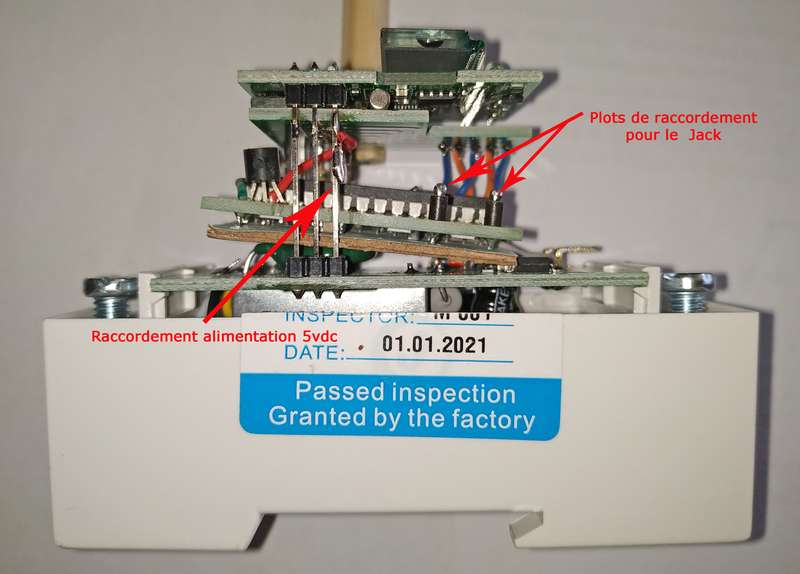
### Modification du compteur XTM 35A et insertion des 2 cartes

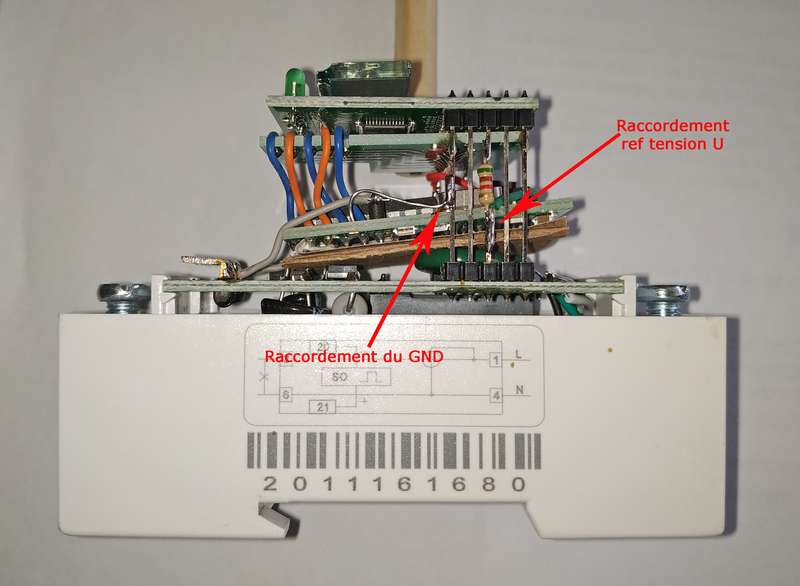
On suit les consignes de Pascal Gourdon[[28]](#footnote-29).

Pour insérer facilement les 2 cartes, il suffit de dessouder la connexion à 3 broches de la platine supérieure :



Connexions avec le compteur :

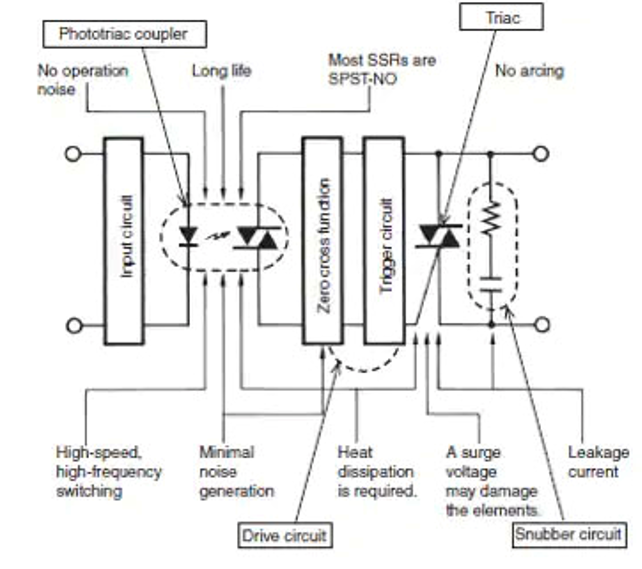




## Modules de puissance[[29]](#footnote-30)

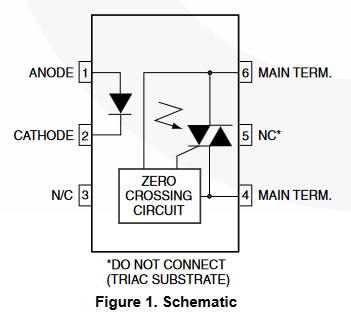
Ils sont ici constitués de 2 relais statiques (SSR), commandés par un signal continu.

Ils sont essentiellement constitués d’un optocoupleur qui commande un triac de puissance[[30]](#footnote-31):

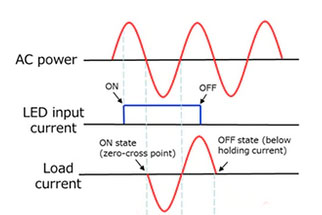


Le triac[[31]](#footnote-32) de puissance agit comme un contacteur normalement ouvert, dont la fermeture est commandée par l’optocoupleur, qui se réouvre automatiquement quand le courant repasse par 0.

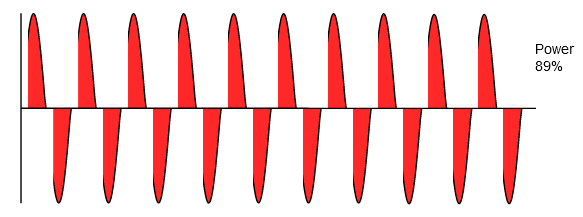
Les relais ici recommandés (Jotta SSR-40 DA) sont équipés d’optocoupleurs MO3063[[32]](#footnote-33) avec détection de zéro :



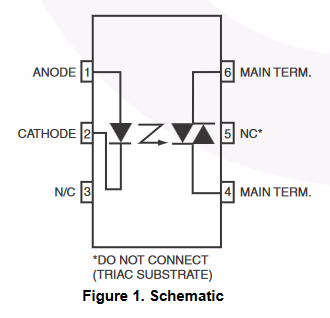
Ceci impliquera que le contacteur attendra aussi que le courant repasse par 0 pour se fermer :



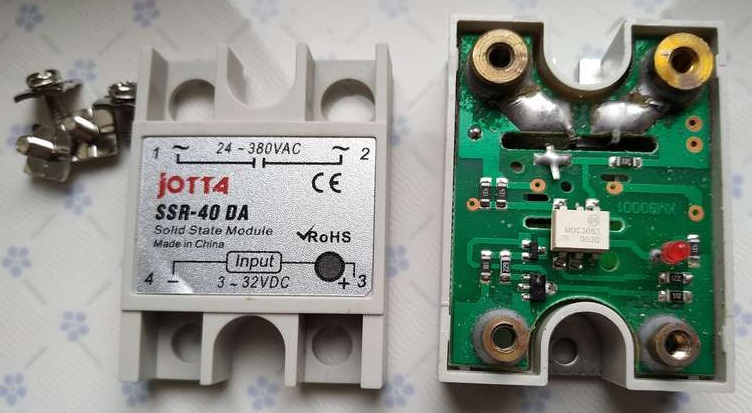
De manière à pouvoir déclencher le relais de la voie 1 en mode contrôle de phase, c'est-à-dire sans attendre que le courant repasse par 0[[33]](#footnote-34) :



il faut dans ce relais remplacer ce composant par un composant analogue, sans détection de zéro, par exemple un MOC3023[[34]](#footnote-35) :



Il suffit pour cela d’ouvrir le relais (facile), de dessouder le MOC3063 (sans arracher les pastilles) et de ressouder à la place le MOC3023 :



A noter aussi le remplacement des 2 petites résistances CMS en parallèle de 430Ω par une de 220Ω tolérant plus de puissance (1W), évoqué par Rolrider sur le forum précité :

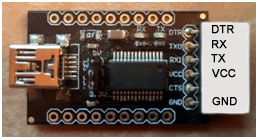
 

# Tests du routeur

## Outils de programmation

### Convertisseur USB⬄Série TTL

* De DfRobot[[35]](#footnote-36)
* Lignes utilisées :
  + 1 – DTR
  + 2 – TXD (=RX sur le convertisseur)[[36]](#footnote-37)
  + 3 – RXD (=TX sur le convertisseur)
  + 4 – VCC (3,3V ou 5V)
  + 6 - GND

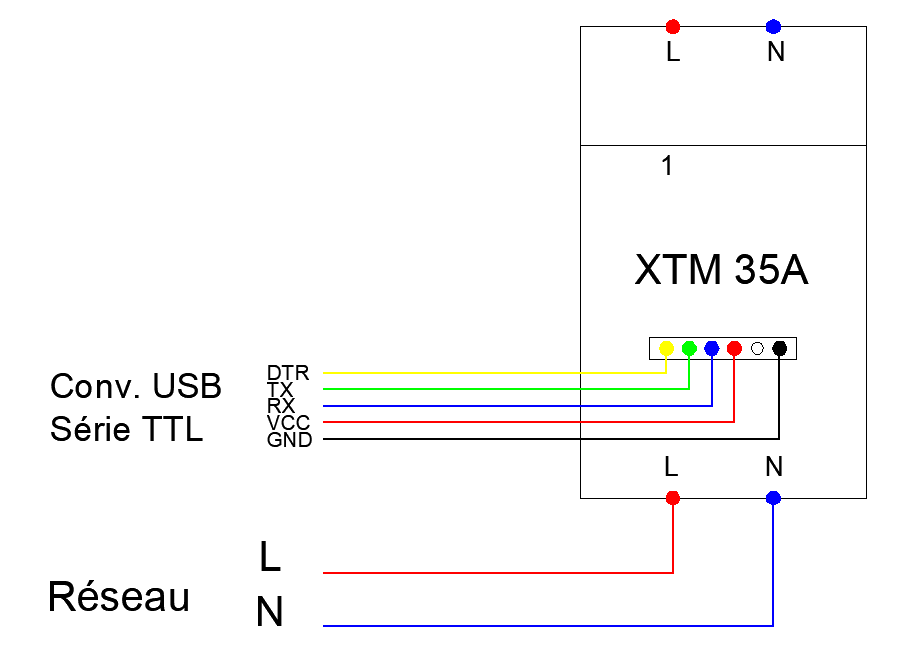


### Environnement de Développement Intégré Arduino 1.8.13

* Type de carte : Arduino Uno
* Port : COMx port (9600b/s)

## Tests

### Communication avec le microcontrôleur



**// TestSerial.ino**

void setup**()**

**{**

Serial**.**begin**(**9600**);**

**}**

void loop**()**

**{**

**if(**Serial**.**available**()>**0**)**

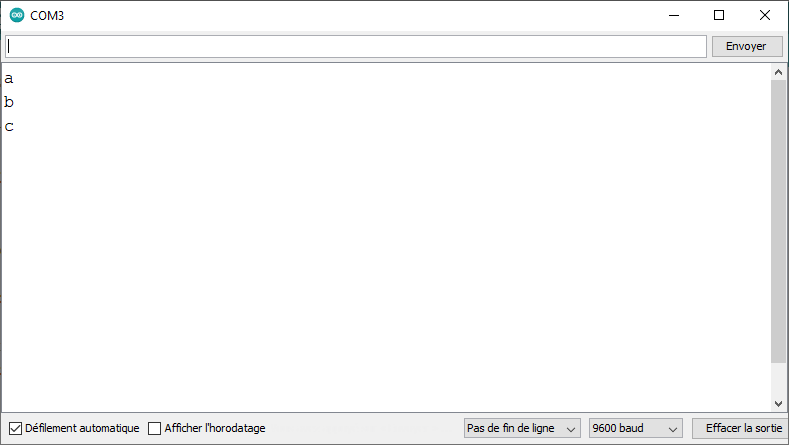
**{**

char x **=** Serial**.**read**();**

Serial**.**println**(**x**);**

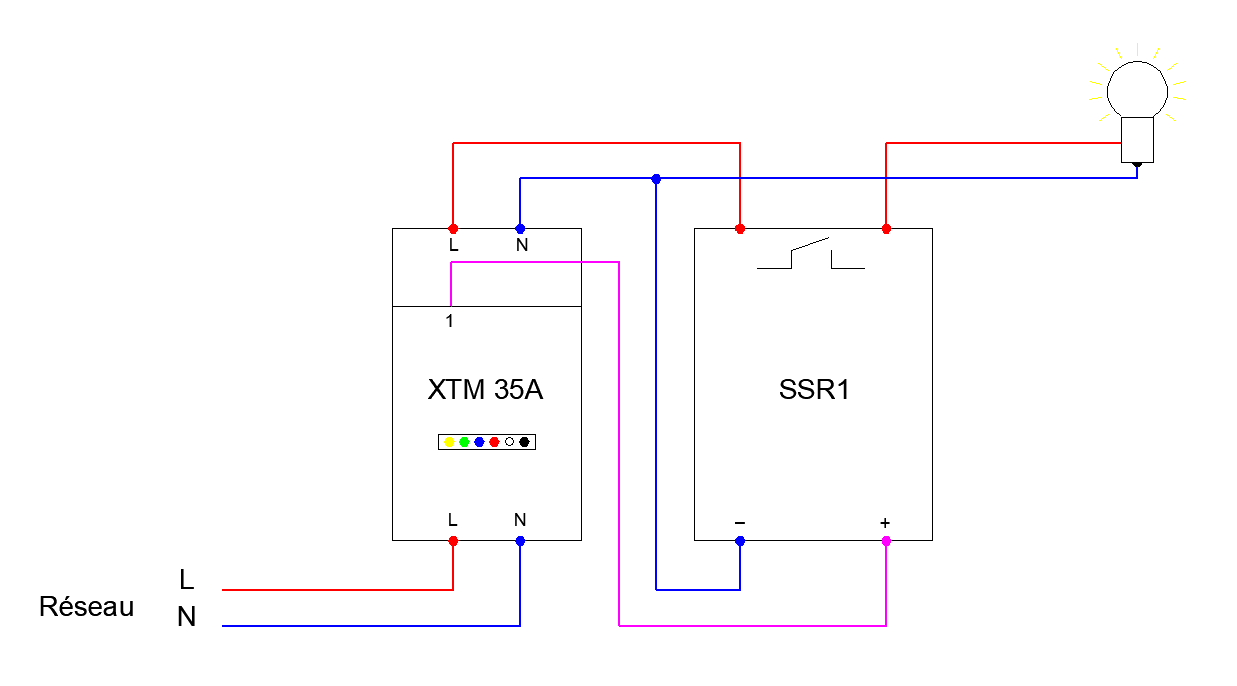
**}**

**}**



Le moniteur série affiche les caractères saisis.

### Commande du relais



**// TestSSR.ino**

#define SSR\_PIN 2 // SSR1

// #define SSR\_PIN 8 // SSR2

void setup**()**

**{**

pinMode**(**SSR\_PIN**,** OUTPUT**);**

**}**

void loop**()**

**{**

digitalWrite**(**SSR\_PIN**,** HIGH**);**

delay**(**2000**);**

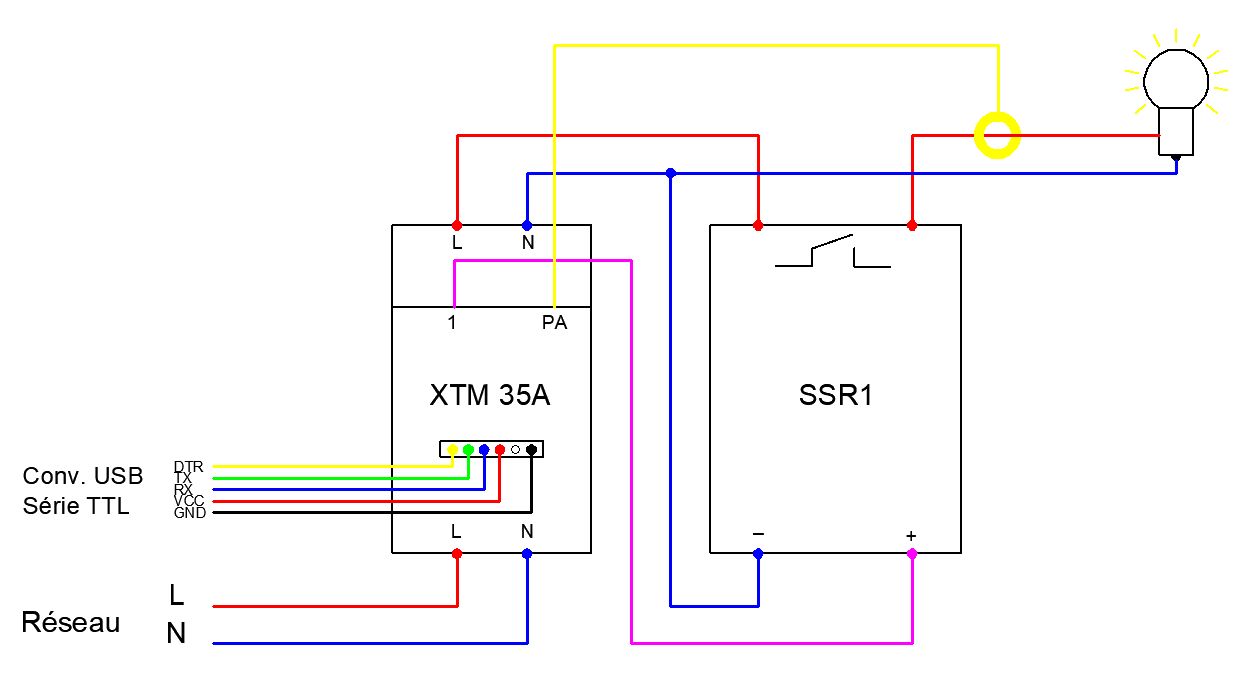
digitalWrite**(**SSR\_PIN**,** LOW**);**

delay**(**2000**);**

**}**

La lampe s’allume et s’éteint périodiquement (comme dans **Blink.ino**).

### Etalonnage et monitoring



On utilise ici la classe **PowerMonitor** (cf. Annexes)

**// PowerMonitor.ino**

// Test the PowerMonitor class

#include "PowerMonitor.h"

#include <elapsedMillis.h>

byte VPin = 3;

byte CPin = 5;

byte SSR\_PIN = 2;

**double VCal = 14.3;**

**double CCal = 0.116;**

**int pha = 280;**

PowerMonitor pm(125, 2, VPin, CPin, VCal, CCal, pha);

elapsedMillis tmr;

double Vrms, Irms, Pr, Pa;

void setup()

{

pinMode(SSR\_PIN, OUTPUT);

digitalWrite(SSR\_PIN, HIGH);

Serial.begin(9600);

pm.Init();

}

void loop()

{

// delay(30);

pm.Process();

if (tmr>2000)

{

tmr = 0;

#ifdef TESTING

Serial.print("MaxSizeBuffer = "); Serial.print(pm.MaxSize());

Serial.print(" NbSamples = "); Serial.print(pm.NbSamples());

Serial.print(" MaxProcTime = "); Serial.print(pm.MaxProcTime()); Serial.println("us");

#else

Vrms = pm.RmsVoltage();

Irms = pm.RmsCurrent();

Pr = pm.RealPower();

Pa = Vrms\*Irms;

Serial.print("Vrms = "); Serial.print(Vrms); Serial.print("V");

Serial.print(" Irms = "); Serial.print(Irms); Serial.print("A");

Serial.print(" Pr = "); Serial.print(Pr); Serial.print("W");

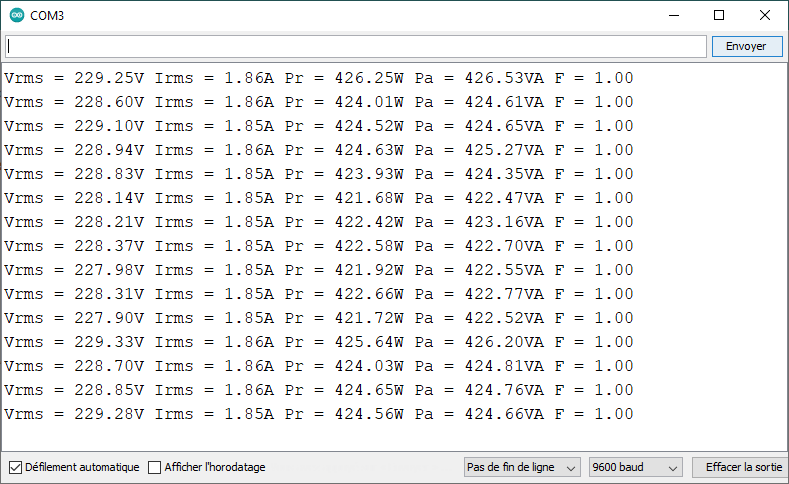
Serial.print(" Pa = "); Serial.print(Vrms\*Irms); Serial.print("VA");

Serial.print(" F = "); Serial.println(abs(Pr)/Pa);

#endif

}

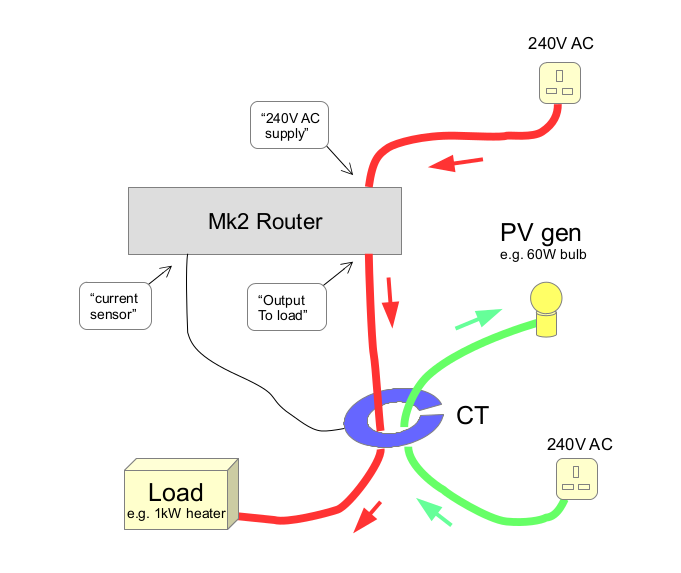
}



Et on observe un flash toutes les 17s environ sur le compteur (=> W)

### Contrôle de la charge

#### Connexions



# Mise en œuvre dans une installation photovoltaïque

## Tableau électrique

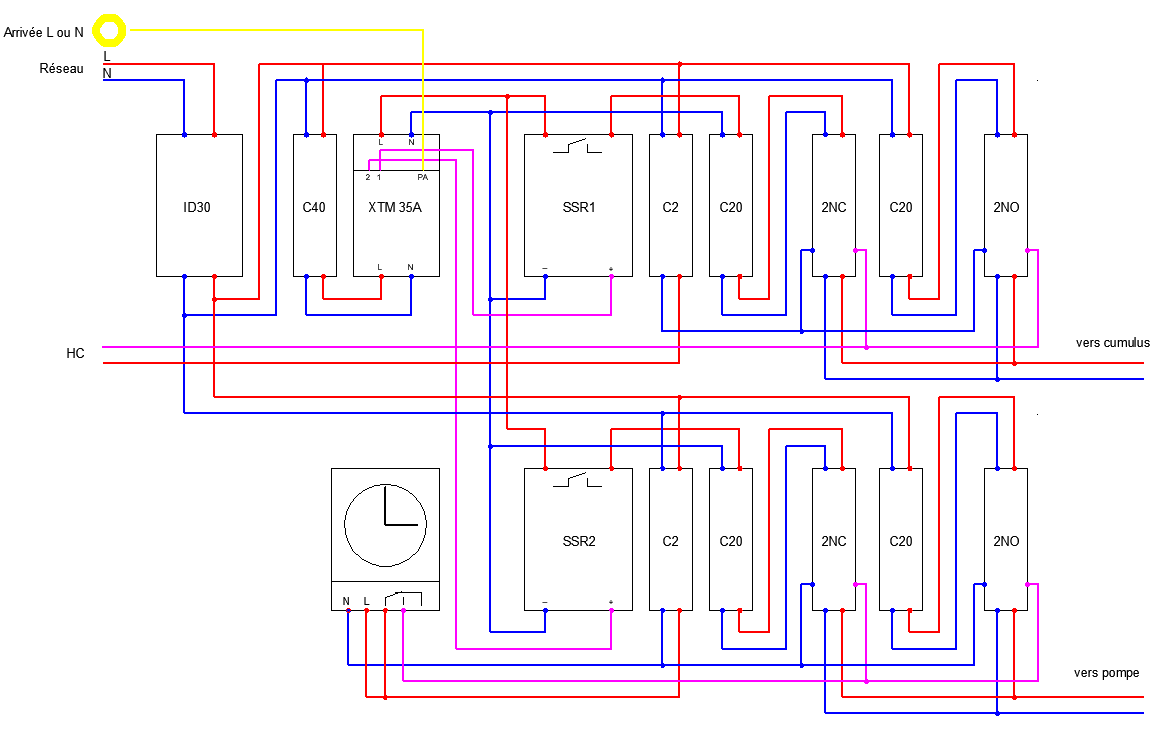
On considère ici le routage vers 2 charges :

* un cumulus sur la voie 1, alimenté normalement pendant les heures creuses
* une pompe de piscine sur la voie 2, commandée normalement par une horloge

Ces 2 charges peuvent être alimentées en permanence par le routeur, sauf quand elles le sont normalement : on utilise des contacteurs 2NF[[37]](#footnote-38) et 2NO[[38]](#footnote-39) pour cela.

Tous les circuits sont protégés contre les courts-circuits par des disjoncteurs divisionnaires magnéto-thermiques[[39]](#footnote-40) et contre les fuites de courant à la terre par un interrupteur différentiel 30mA[[40]](#footnote-41).

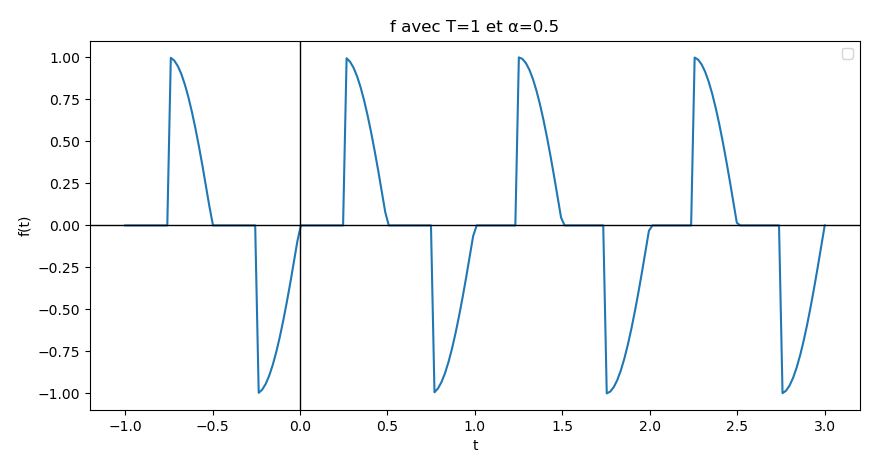
A noter ici l’inversion phase-neutre pour le compteur XTM 35A et l’utilisation de sa sortie droite (borne n°3) comme référence – pour les relais solides (borne n°4). Et les connexions sont faites pour que seul le courant routé soit comptabilisé par le compteur.



# Annexes

## harmoniques du sinus tronqué

On considère une **période** , , un coefficient et la fonction -périodique obtenue en tronquant la fonction au début de chaque demi-alternance :



Sur l’intervalle  :

si ou

sinon

On cherche les **harmoniques** de , c’est-à-dire les termes de sa décomposition en **série de Fourier** :

avec :

et si  :

De manière évidente :

Ensuite, si et  :

En effectuant le changement de variable , avec  :

En utilisant les relations trigonométriques[[41]](#footnote-42), on arrive à :

et de manière analogue :

D’abord, si  :

soit :

et :

soit :

En résumé :

|  |  |
| --- | --- |
|  |  |

Ensuite, si  :

Le premier terme est égal à :

et le deuxième à :

En réduisant au même dénominateur les termes sans cosinus, on arrive à :

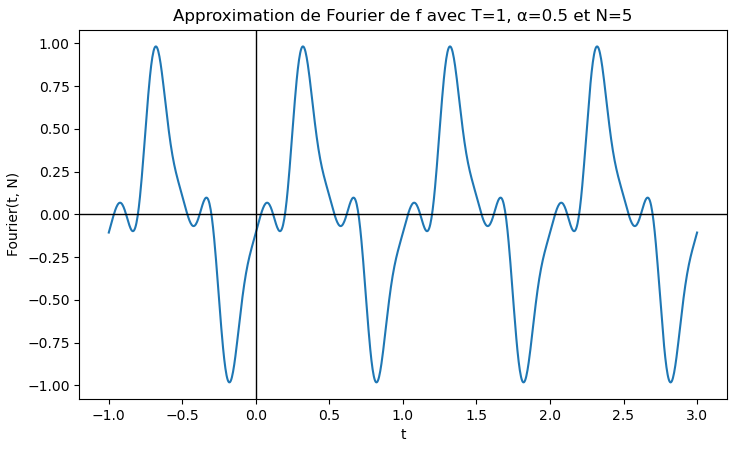
De manière analogue :

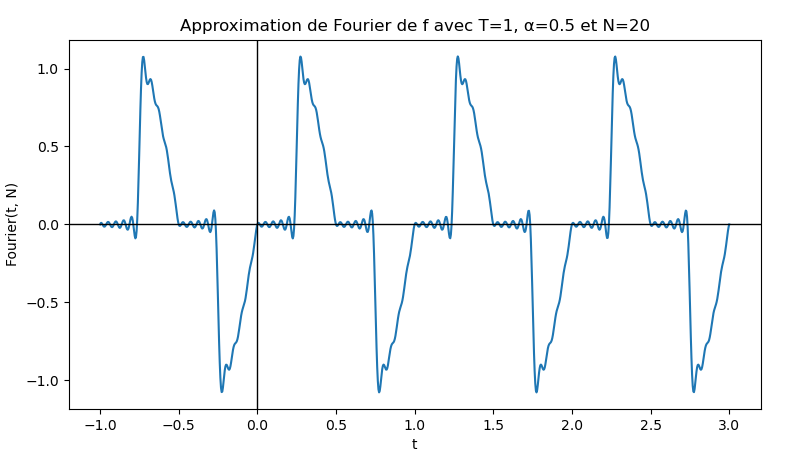
Le premier terme est égal à :

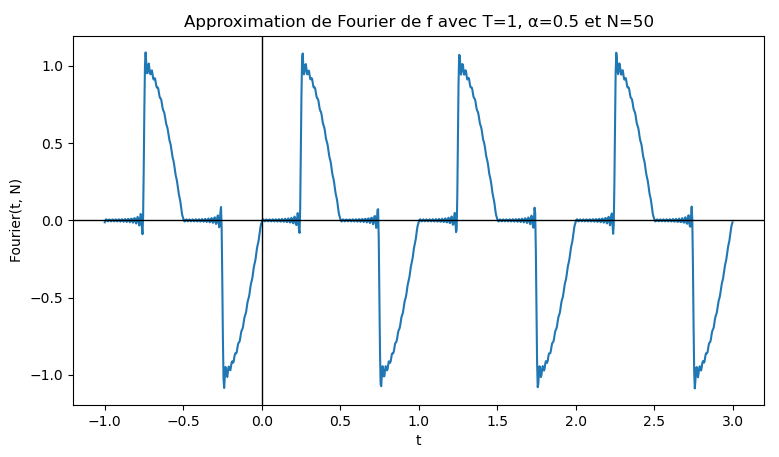
et le deuxième à :

Ainsi :

**Approximations obtenues :**







On constate ici la nécessité d’utiliser de nombreuses harmoniques pour obtenir une bonne approximation, avec toutefois la persistance d’oscillations aux discontinuités à cause du phénomène de Gibbs[[42]](#footnote-43).

Du point de vue pratique, on peut donc penser que le contrôle de phase va engendrer de nombreuses harmoniques qui perturberont le bon fonctionnement des circuits dès lors qu’ils comportent des éléments inductifs ou capacitifs : un filtrage HF peut ici s’avérer nécessaire !

**Programmes Python[[43]](#footnote-44) pour le tracé des graphes :**

**# Plot1.py**

# Librairies

**from** numpy **import** pi**,** sin**,** linspace**,** floor

**import** matplotlib**.**pyplot **as** plt

# Paramètres

T **=** 1

α **=** 1**/**2

ω **=** 2**\***pi**/**T

# Espace temps

t **=** linspace**(-**T**,** 3**\***T**,** 1000**)**

# Fonction f

**def** f**(**x**):**

k **=** floor**(**2**\***x**/**T**)**

**if** **(**x**-**k**\***T**/**2**)** **>=** α**\***T**/**2 **:**

y **=** sin**(**ω**\***x**)**

**else** **:**

y **=** 0

**return** y

# Graphe

plt**.**plot**(**t**,** **[**f**(**x**)** **for** x **in** t**])**

plt**.**axhline**(**y**=**0**,** linewidth**=**1**,** color**=**'k'**)**

plt**.**axvline**(**x**=**0**,** linewidth**=**1**,** color**=**'k'**)**

plt**.**xlabel**(**"t"**)**

plt**.**ylabel**(**"f(t)"**)**

plt**.**title**(**"f avec T=1 et α=0.5"**)**

plt**.**show**()**

**# Plot2.py**

# Librairies

**from** numpy **import** pi**,** sin**,** cos**,** linspace

**import** matplotlib**.**pyplot **as** plt

#constantes

T **=** 1

α **=** 0.5

ω **=** 2**\***pi**/**T

αpi **=** α**\***pi

α1pi **=** **(**1**+**α**)\***pi

# Espace temps

t **=** linspace**(-**T**,** 3**\***T**,** 1000**)**

# Fonction Fourier

**def** Fourier**(**x**,** N**):**

n **=** 1

a **=** **(**cos**(**2**\***αpi**)-**1**)/(**2**\***pi**)**

b **=** 1**-**α**+**sin**(**2**\***αpi**)/(**2**\***pi**)**

y **=** a**\***cos**(**ω**\***x**)+**b**\***sin**(**ω**\***x**)**

**while** n **<** N**:**

n **=** n**+**1

a **=** **((-**1**)\*\***n**-**1**)/(**pi**\*(**1**-**n**\*\***2**))**

a **=** a **+** **(**cos**((**1**+**n**)\***αpi**)+**cos**((**1**+**n**)\***α1pi**))/(**2**\***pi**\*(**1**+**n**))**

a **=** a **+** **(**cos**((**1**-**n**)\***αpi**)+**cos**((**1**-**n**)\***α1pi**))/(**2**\***pi**\*(**1**-**n**))**

b **=** **(**sin**((**1**+**n**)\***αpi**)+**sin**((**1**+**n**)\***α1pi**))/(**2**\***pi**\*(**1**+**n**))**

b **=** b **-** **(**sin**((**1**-**n**)\***αpi**)+**sin**((**1**-**n**)\***α1pi**))/(**2**\***pi**\*(**1**-**n**))**

y **=** y **+** a**\***cos**(**n**\***ω**\***x**)+**b**\***sin**(**n**\***ω**\***x**)**

**return** y

# Graphe

plt**.**plot**(**t**,** **[**Fourier**(**x**,** 50**)** **for** x **in** t**])**

plt**.**axhline**(**y**=**0**,** linewidth**=**1**,** color**=**'k'**)**

plt**.**axvline**(**x**=**0**,** linewidth**=**1**,** color**=**'k'**)**

plt**.**xlabel**(**"t"**)**

plt**.**ylabel**(**"Fourier(t, N)"**)**

plt**.**title**(**"Approximation de Fourier de f avec T=1, α=0.5 et N=50"**)**

plt**.**show**()**

## Classes

**// RingBuffer.hpp**

#pragma once

#include <Arduino.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Class RingBuffer<M, N>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Circular Buffer FIFO data structure, with M arrays of N int elements

\*/

template **<**int M**,** int N**>** class RingBuffer

**{**

public**:**

// Constructor

RingBuffer**()**

**{**

pStart **=** buffer**;**

pHead **=** pStart**;**

pTail **=** pStart**;**

pEnd **=** **&**buffer**[**M**\***N**-**1**];**

size **=** 0**;**

**}**

// Number of elements

int Size**()** **{return** size**;}**

// Operations

void Push**(**int t**[])**

**{**

**if** **(**size **<** M**)**

**{**

size**++;**

**for** **(**int i**=**0**;** i**<**N**;** i**++)** **\***pTail**++** **=** **\***t**++;**

**if** **(**pTail**>**pEnd**)** pTail **=** pStart**;**

**}**

**}**

int Pop**(**int t**[])** // Before test if Size()>0

**{**

size**--;**

**for** **(**int i**=**0**;** i**<**N**;** i**++)** **\***t**++** **=** **\***pHead**++;**

**if** **(**pHead**>**pEnd**)** pHead **=** pStart**;**

**}**

private**:**

int buffer**[**M**\***N**];**

int **\***pHead**,** **\***pTail**,** **\***pStart**,** **\***pEnd**;**

int size**;**

**};**

**// PowerMonitor.h**

#pragma once

#include <Arduino.h>

#include "RingBuffer.hpp"

#define M 100 // Buffer size <256

//#define TESTING

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Class PowerMonitor

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Rms voltage, rms current and real power estimations for voltage and current channels

Inspired from :

Robin Emley's Cal\_bothDisplays\_2.ino sketch ( https://www.mk2pvrouter.co.uk )

EmonLib library ( https://github.com/openenergymonitor/EmonLib )

EmonLibCM library ( https://github.com/openenergymonitor/EmonLibCM )

Constructor parameters :

\_period = sampling period for the channels (in us and 105<.<8192)

\_nCycl = number of cycles to construct the estimations (>=1)

\_VPin, \_CPin = Arduino analog pins for the voltage and the current (Ax)

\_VCal, \_CCal = Calculus coefficients to convert the digitized values

\_pha = Phase shift coefficient of the voltage (int)

NB : \_pha=256 => no shift, \_pha<256 => right shift, \_pha>256 => left shift

\*/

enum Polarities **{**POSITIVE**,** NEGATIVE**};** // Alternance polarities

extern "C" void TIMER1\_OVF\_vect**();**

class PowerMonitor

**{**

public**:**

// Constructor

PowerMonitor**(**int \_period**,** int \_nCycl**,** byte \_VPin**,** byte \_CPin**,** double \_VCal**,** double \_CCal**,** int \_pha**);**

void Init**();** // Initialisation (to be called in the setup)

void Process**();** // Process calculus outside the ISR (to be called in the loop()

double RmsVoltage**();** // Vrms (V)

double RmsCurrent**();** // Irms (A)

double RealPower**();** // Pr (W)

#ifdef TESTING

byte MaxSize**();** // Maximum size of the circular buffer, on entering Process()

int NbSamples**();** // Number of samples in estimations

long MaxProcTime**();** // Max time to treat a sample (us)

#endif

private**:**

unsigned int period**;** // Sampling period for the channels

int nCycl**,** iCycl**;** // Number of cycles and index of the cycle

byte VPin**,** CPin**;** // Arduino pins for the voltage and the current

double VCal**,** CCal**;** // Calculus coefficients to convert the digitized vales

int pha**;** // Phase shift coefficients array of the voltage

int iChan**;** // Index of the channel in the digitalization

RingBuffer**<**M**,** 2**>** rbDig**;** // Circular buffer for the digitized values

int tDigISR**[**2**];** // Digitized values array pushed in the ISR

int tDig**[**2**];** // Digitized values array poped in the Process method

long V**,** C**;** // Voltage and current values (<<8 and centered)

Polarities pol**;** // Polarity of the alternance

long offV**,** offC**;** // Offsets for centering

long sumV**,** sumC**;** // Sums of the voltage and current values

long sumVV**,** sumCC**;** // Sums of the squared voltage and current values (>>10)

long savSumVV**,** savSumCC**;** // Saved sum of the squared voltage and current values (>>10)

long sumVC**;** // Sum of the VC products(>>10)

long savSumVC**;** // Saved sum of the VC products(>>10)

long preV**;** // Previous voltage value (>>2)

long phaV**;** // Phase shifted voltage value (>>2)

int nSum**;** // Number (saved) of values in the sums VV, CC and VC

int nSavSum**;** // Number (saved) of values in the sums VV, CC and VC

void FuncISR**();** // ISR function

#ifdef TESTING

byte maxSize**;** // Maximum size of the circular buffer, on entering Process()

long maxProcTime**;** // Maximum of processing time

#endif

friend void TIMER1\_OVF\_vect**();** // To use the private FuncISR() in the ISR

**};**

**// PowerMonitor.cpp**

#include "PowerMonitor.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ISR

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PowerMonitor**\*** pPowerMonitor**;** // As static

ISR**(**TIMER1\_OVF\_vect**)**

**{**

pPowerMonitor**->**FuncISR**();**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Class PowerMonitor

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PowerMonitor**::**PowerMonitor**(**int \_period**,** int \_nCycl**,** byte \_VPin**,** byte \_CPin**,** double \_VCal**,** double \_CCal**,** int \_pha**)**

**{**

pPowerMonitor **=** **this;**

period **=** \_period**;**

nCycl **=** \_nCycl**;**

VPin **=** \_VPin**;**

CPin **=** \_CPin**;**

VCal **=** \_VCal**;**

CCal **=** \_CCal**;**

pha **=** \_pha**;**

**}**

void PowerMonitor**::**Init**()**

**{**

// Set ADC parameters (prescaler, voltage reference), enable and start it with the volatge source

ADCSRA **=** **(**1**<<**ADPS0**)|(**1**<<**ADPS1**)|(**1**<<**ADPS2**);** // Prescaler = 128 (104us/conversion and max accuracy)

ADCSRA **|=** **(**1**<<**ADEN**);** // Enable ADC

ADMUX **=** 0x40**;** // Voltage reference = AVcc with external capacitor at AREF pin

ADMUX **|=** VPin**;** // Set ADPS to the voltage pin

ADCSRA **|=** **(**1**<<**ADSC**);** // Start the conversion

delay**(**1**);** // Wait to perform this conversion

// Set Timer1 interrupts

noInterrupts**();** // Disable all interrupts

TCCR1A **=** 0**;** // Normal port operation (no PWM outputs)

TCCR1B **=** **(**1**<<**WGM13**);** // Phase and frequency correct PWM mode with TOP in ICR1

TCCR1B **|=** **(**1**<<**CS10**);** // No prescaling

ICR1 **=** 8**\***period**;** // TOP

TIMSK1 **=** **(**1**<<**TOIE1**);** // Enable timer OVF interrupt (TOV1 flag set on at BOTTOM)

interrupts**();** // Enable all interrupts

// Init values for members

iCycl **=** 0**;**

iChan **=** 0**;**

nSum **=** 0**;**

nSavSum **=** 0**;**

offV **=** 510L**\***256**;**

sumV **=** 0**;**

sumVV **=** 0**;**

offC **=** 510L**\***256**;**

sumC **=** 0**;**

sumCC **=** 0**;**

sumVC **=** 0**;**

preV **=** 220L**\***64**;**

#ifdef TESTING

maxSize **=** 0**;**

maxProcTime **=** 0**;**

#endif

**}**

void PowerMonitor**::**Process**()**

**{**

#ifdef TESTING

maxSize **=** max**(**maxSize**,** rbDig**.**Size**());**

#endif

**while** **(**rbDig**.**Size**()** **>** 0**)**

**{**

#ifdef TESTING

long startTime **=** micros**();**

#endif

// Pop the digitized values

cli**();**

rbDig**.**Pop**(**tDig**);**

sei**();**

// Voltage and current values (x256 and centering)

V **=** **(((**long**)**tDig**[**0**])<<**8**)-** offV**;**

C **=** **(((**long**)**tDig**[**1**])<<**8**)-** offC**;**

// New half alternance test

**if** **((**V **>=** 0**)** **&&** **(**pol **==** NEGATIVE**))** // Start POSITIVE (new cycle)

**{**

// Polarity new value

pol **=** POSITIVE**;**

iCycl**++;**

**if** **(**iCycl **==** nCycl**)**

**{**

// Save the sums of squared values and VC products, then reinit them

savSumVV **=** sumVV**;**

savSumCC **=** sumCC**;**

savSumVC **=** sumVC**;**

nSavSum **=** nSum**;**

sumVV **=** 0**;**

sumCC **=** 0**;**

sumVC **=** 0**;**

nSum **=** 0**;**

// Reinit iCycl

iCycl **=** 0**;**

**}**

**}**

**else** **if** **((**V **<** 0**)** **&&** **(**pol **==** POSITIVE**))** // Start NEGATIVE

**{**

// Polarity new value

pol **=** NEGATIVE**;**

// Offsets update using low pass filters

offV **+=** **(**sumV**>>**12**);**

offC **+=** **(**sumC**>>**12**);**

sumV **=** 0**;**

sumC **=** 0**;**

**}**

// Sums incrementation

sumV **+=** V**;**

V **=** **(**V**>>**2**);**

sumVV **+=** **((**V**\***V**)>>**6**);**

sumC **+=** C**;**

C **=** **(**C**>>**2**);**

sumCC **+=** **((**C**\***C**)>>**6**);**

phaV **=** preV **+** **(((**V **-** preV**)\***pha**)>>**8**);**

sumVC **+=** **((**phaV**\***C**)>>**6**);**

preV **=** V**;**

nSum**++;**

#ifdef TESTING

maxProcTime **=** max**(**maxProcTime**,** micros**()-**startTime**);**

#endif

**}**

**}**

double PowerMonitor**::**RmsVoltage**()**

**{**

**return** VCal**\***sqrt**(((**double**)**savSumVV**)/(**64.**\***nSavSum**));**

**}**

double PowerMonitor**::**RmsCurrent**()**

**{**

**return** CCal**\***sqrt**(((**double**)**savSumCC**)/(**64.**\***nSavSum**));**

**}**

double PowerMonitor**::**RealPower**()**

**{**

**return** VCal**\***CCal**\*(((**double**)**savSumVC**)/(**64.**\***nSavSum**));**

**}**

#ifdef TESTING

byte PowerMonitor**::**MaxSize**()**

**{**

byte savMaxSize **=** maxSize**;**

maxSize **=** 0**;**

**return** savMaxSize**;**

**}**

int PowerMonitor**::**NbSamples**()**

**{**

**return** nSavSum**;**

**}**

long PowerMonitor**::**MaxProcTime**()**

**{**

long savMaxProcTime **=** maxProcTime**;**

maxProcTime **=** 0**;**

**return** savMaxProcTime**;**

**}**

#endif

void PowerMonitor**::**FuncISR**()**

**{**

// Digitized value

tDigISR**[**iChan**++]** **=** ADC**;**

// Push the digitized values in the buffer

**if(**iChan **==** 2**)**

**{**

iChan **=** 0**;**

ADMUX **=** 0x40 **|** VPin**;**

rbDig**.**Push**(**tDigISR**);**

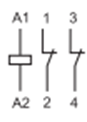
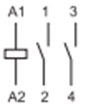
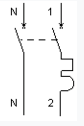
**}**

// Start the next conversion

ADMUX **=** 0x40 **|** CPin**;**

ADCSRA **|=** **(**1**<<**ADSC**);**

**}**

1. <https://forum-photovoltaique.fr> [↑](#footnote-ref-2)
2. <https://www.amazon.fr/XTM35SA-Monophas%C3%A9-Meter-Digital-10%E2%84%83/dp/B09JSYM5R1/ref=sr_1_1_sspa> [↑](#footnote-ref-3)
3. <http://www.cnhaohua.com/product/1150.html> [↑](#footnote-ref-4)
4. Si secondes entre 2 flashs => W (puissance réelle). [↑](#footnote-ref-5)
5. 0V environ entre cette ligne et la terre. [↑](#footnote-ref-6)
6. <https://forum-photovoltaique.fr/viewtopic.php?f=110&t=40512> [↑](#footnote-ref-7)
7. <https://fr.aliexpress.com/item/32297200306.html> [↑](#footnote-ref-8)
8. <https://cdn.sparkfun.com/datasheets/Components/General/SSR40DA.pdf> [↑](#footnote-ref-9)
9. <https://fr.aliexpress.com/item/32715724000.html> [↑](#footnote-ref-10)
10. <https://learn.openenergymonitor.org/electricity-monitoring/ct-sensors/files/SCT013-000_datasheet_0.pdf> [↑](#footnote-ref-11)
11. <https://learn.openenergymonitor.org/electricity-monitoring/ct-sensors/yhdc-sct-013-000-ct-sensor-report> [↑](#footnote-ref-12)
12. <https://forum-photovoltaique.fr/viewtopic.php?f=110&t=39159> [↑](#footnote-ref-13)
13. <https://forum-photovoltaique.fr/viewtopic.php?f=110&t=40512> [↑](#footnote-ref-14)
14. <https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf> [↑](#footnote-ref-15)
15. <https://www.gammon.com.au/forum/?id=11497> [↑](#footnote-ref-16)
16. <https://forum-photovoltaique.fr/viewtopic.php?t=40512> [↑](#footnote-ref-17)
17. <https://ww1.microchip.com/downloads/en/DeviceDoc/22008E.pdf> [↑](#footnote-ref-18)
18. <https://forum-photovoltaique.fr/viewtopic.php?t=48170> [↑](#footnote-ref-19)
19. <https://www.analog.com/en/products/adum1201.html> [↑](#footnote-ref-20)
20. <https://www.analog.com/en/products/adum1301.html> [↑](#footnote-ref-21)
21. De manière pouvoir utiliser l’auto-reset du microcontrôleur, via la broche DTR du convertisseur [↑](#footnote-ref-22)
22. <https://fr.rs-online.com/web/p/isolateurs-numeriques/7532232> [↑](#footnote-ref-23)
23. <https://docs.rs-online.com/d4d3/0900766b810a308f.pdf> [↑](#footnote-ref-24)
24. <https://fr.farnell.com/wurth-elektronik/885012207128/cond-0-1-f-100v-10-x7r-0805/dp/2812440> [↑](#footnote-ref-25)
25. <https://www.farnell.com/datasheets/2546917.pdf> [↑](#footnote-ref-26)
26. <https://www.digikey.fr/fr/products/detail/aries-electronics/16-350000-11-RC/1014490> [↑](#footnote-ref-27)
27. <https://www.arieselec.com/wp-content/uploads/2020/02/18010rc-rohs-compliant-soic-to-dip-adapter.pdf> [↑](#footnote-ref-28)
28. <https://forum-photovoltaique.fr/viewtopic.php?f=110&t=40512> avec juste une inversion pour les plots SSRx. [↑](#footnote-ref-29)
29. <https://forum-photovoltaique.fr/viewtopic.php?t=40617> [↑](#footnote-ref-30)
30. <https://www.digikey.fr/fr/articles/how-to-safely-and-efficiently-switch-current-or-voltage-using-ssrs> [↑](#footnote-ref-31)
31. <https://fr.wikipedia.org/wiki/Triac> [↑](#footnote-ref-32)
32. <https://www.onsemi.com/pdf/datasheet/moc3163m-d.pdf> [↑](#footnote-ref-33)
33. <https://learn.openenergymonitor.org/pv-diversion/mk2/switchdev> [↑](#footnote-ref-34)
34. <https://www.onsemi.com/pdf/datasheet/moc3023m-d.pdf> [↑](#footnote-ref-35)
35. <https://www.robotshop.com/media/files/PDF/datasheet-dfr0065.pdf> [↑](#footnote-ref-36)
36. Ici TXD et RXD indiquent les broches FTDI TX et RX sur le microcontrôleur [↑](#footnote-ref-37)
37.  [↑](#footnote-ref-38)
38.  [↑](#footnote-ref-39)
39.  [↑](#footnote-ref-40)
40. Un seul suffit car il n’y a ici que 5 circuits à protéger :

    Routeur + relais

    Commutateur HC

    Horloge

    Cumulus

    Pompe [↑](#footnote-ref-41)
41. et [↑](#footnote-ref-42)
42. <https://fr.wikipedia.org/wiki/Ph%C3%A9nom%C3%A8ne_de_Gibbs> [↑](#footnote-ref-43)
43. Par exemple : <https://sourceforge.net/projects/winpython/> [↑](#footnote-ref-44)