Fabriquer un banc de mesures pour éoliennes ou PV

Autheur : Association P'TIWATT

avec Philippe de Craene dcphilippe@yahoo.fr
et Dominique Boucherie ptiwatt@mailoo.org

Date : octobre 2018

Version 1.1

Un banc de mesures permet de relever les courants /tensions en amont et en aval du dispositif de régulation et ou d'injection.

Les données collectées sont enregistrées sur une carte SD sous la forme d'un fichier, ce fichier pouvant ensuite être récupéré et traité par un tableur.

Ce document décrit comment réaliser cet appareil.

A noter : la réalisation de ce programme a demandé plusieurs dizaines d'heures de développement, apprendre à utiliser l'Arduino, comprendre son comportement, en cramer deux… apprendre son langage, faire des tests sur différents composants, tester différents algorithmes, et imaginer tous les tests possibles pour fiabiliser au maximum l'appareil. Toute contribution en vue de l'amélioration de l'appareil est la bienvenue ! Il vous est juste demandé de conserver mon nom et mon email dans l'entête du programme, et bien sûr de partager avec moi cette amélioration. Merci.

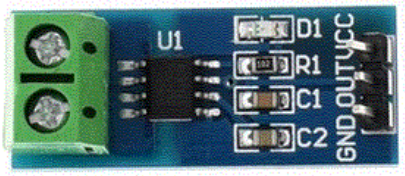
Table des matières

Liste des courses

1- Un arduino Uno R3 : https://fr.aliexpress.com/item/One-set-New-2016-UNO-R3-ATmega328P-CH340G-MicroUSB-Compatible-for-Arduino-UNO-Rev-3-0/32696412561.html?spm=a2g0s.9042311.0.0.27426c37rrsgNO

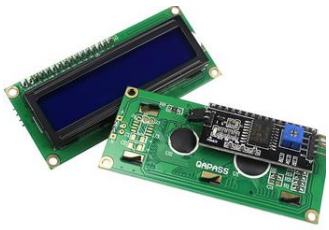
2- Une carte d'extension : https://fr.aliexpress.com/item/Free-Shipping-UNO-Proto-Shield-prototype-expansion-board-with-SYB-170-mini-bread-board-based-For/32502867722.html?spm=a2g0s.9042311.0.0.27426c37rrsgNO

3- Un ou 2 capteurs de courant à base de l' ACS712, il en existe 3 modèles : 5A, 20A et 30A. Le choix se portera sur le modèle 20A si l'éolienne ne peut pas fournir plus : https://fr.aliexpress.com/item/WAVGAT-Hot-Sale-ACS712-20A-Range-Hall-Current-Sensor-Module-ACS712-Module-For-Arduino-20A/32827933262.html?spm=a2g0s.13010208.99999999.258.16fd3c00mx02AD



4- Suivant la configuration retenue un capteur de courant alternatif sensible : https://fr.aliexpress.com/item/Free-shipping-0-30A-sensor-split-core-current-transformer-SCT006/32579590465.html?spm=a2g0s.9042311.0.0.27426c37zd6RJS
Le capteur de courant est une sorte de mini transformateur ne comportant qu'un secondaire qui doit être chargé sur une résistance (dite de Burden) de l'ordre de 100Ω. Inutile d'augmenter la valeur de cette résistance pour tenter d'obtenir un meilleur taux de transformation, ça sature très vite ces petits capteurs.
Ce modèle possède sa propre résistance de Burden : https://fr.aliexpress.com/item/5A-Range-of-Single-Phase-AC-Current-Sensor-Module-for-Arduino-Free-Shipping/32713284483.html?spm=a2g0s.9042311.0.0.27426c37TNdWoA

5- Un afficheur LCD 1602 avec I2C : https://fr.aliexpress.com/item/1602-LCD-Module-Bleu-Jaune-Vert-cran-avec-IIC-I2C-16x2-LCD-R-tro-clairage-Module/32891917063.html?spm=a2g0s.9042311.0.0.458d6c37pXdIJP

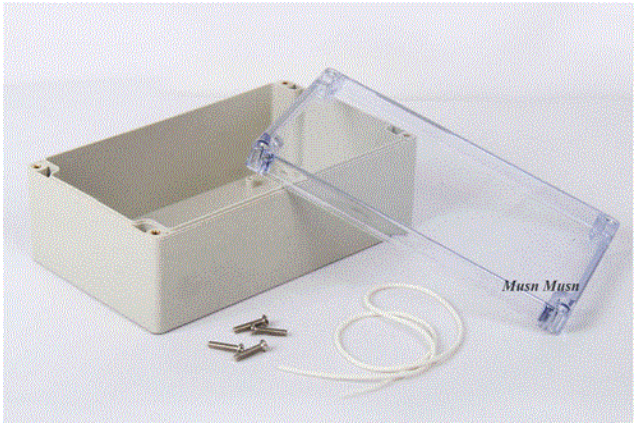


6- Un module de lecteur de carte SD pour le cas où vous souhaitez conserver les données de mesure et vérifier le fonctionnement et le rendement du montage : https://fr.aliexpress.com/item/New-Data-Module-Logging-Shield-SD-Card-Data-Recorder-Shield-V1-0-UNO-SD-Card-Hot/32815793580.html?spm=a2g0s.9042311.0.0.27426c377SIMdS

7- Quelques zeners 4,7V, résistances, condensateurs dont les valeurs sont données dans le circuit électrique…

8- Une alimentation 5V pour l'Arduino, c'est-à-dire un chargeur de téléphone portable de récupération, du câble Dupont (bof bof c'est plein de mauvais contacts, rien de vaut mieux que la soudure), une toute petite poignée de composants électriques dont la liste est un peu plus bas.

9- Une jolie boite pour intégrer durablement le montage (c'est d'ailleurs l'élément le très loin le plus cher)



Banc de mesures pour configuration en injection directe :

Le banc de mesure va collecter les informations suivantes :

Si l'éolienne tourne, et dans ce cas à intervalles réguliers :

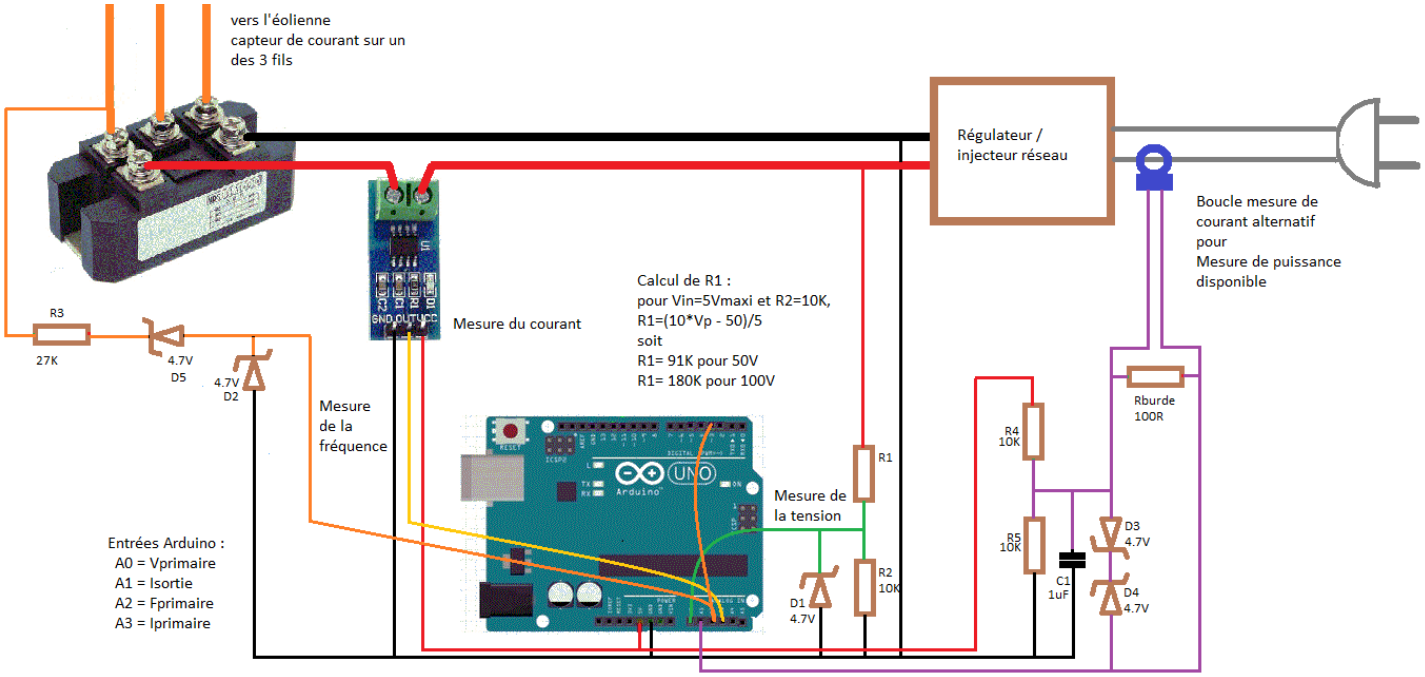
- Date et l'heure,
- Vitesse de rotation Fprimaire
- Tension délivrée au pont de diode Vprimaire
- Courant délivré au pont de diode Iprimaire
- Courant injecté sur le réseau Isortie
  On suppose que la tension du réseau est stable à 230V.

Vprimaire * Iprimare = Pprimaire

230 * Isortie = Psortie

D'où calcul du rendement Psortie/Pprimaire, mesure de la courbe Iprimaire et Vprimaire en fonction de la vitesse de rotation, mesure du temps de recherche du MPPT du régulateur/injecteur, etc….

Mesure de la tension, des courants et de la fréquence



Le capteur de vitesse de rotation de l'hélice est construit autour de R3, D2 et D5. Le signal alternatif est récupéré sur l'une des 3 entrées du pont de diode. Entre chacune de ces 3 entrées le signal est sinusoïdal, mais par rapport à la masse le signal n'est plus que la demie-alternance positive. R3 limite le courant à moins de 5 milliampères, D5 sécurise qu'il n'y est aucun signal négatif et crée par la même occasion un offset à 4,7V, car ça ne sert pas à grand-chose d'effectuer des relevés de mesure quand l'hélice tourne si doucement qu'aucun courant récupérable n'est produit. Enfin D2 limite le signal à 4,7V compatible avec l'Arduino. Pour les tests du circuit le capteur est branché sur l'entrée analogique A2 pour vérifier l'amplitude du signal, et l'entrée numérique 3 pour calculer la fréquence.

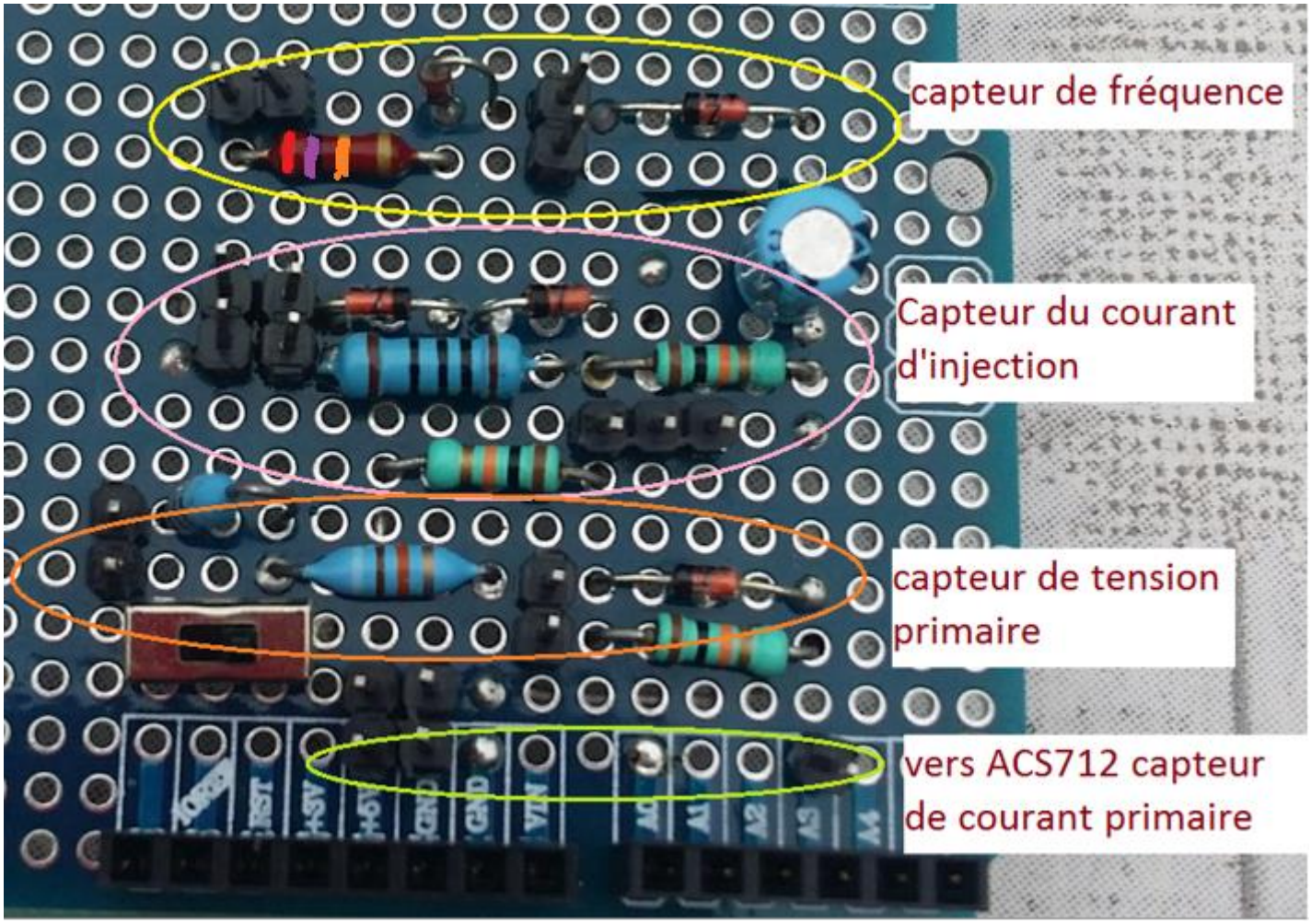
Le capteur de tension primaire est un simple pont diviseur construit autour de R1 et R2. La mesure s'effectue aux bornes de R2, la tension recueillie est V = Vprimaire * R2 / (R1+R2). D1 est une sécurité qui protège l'Arduino en cas de surtension.

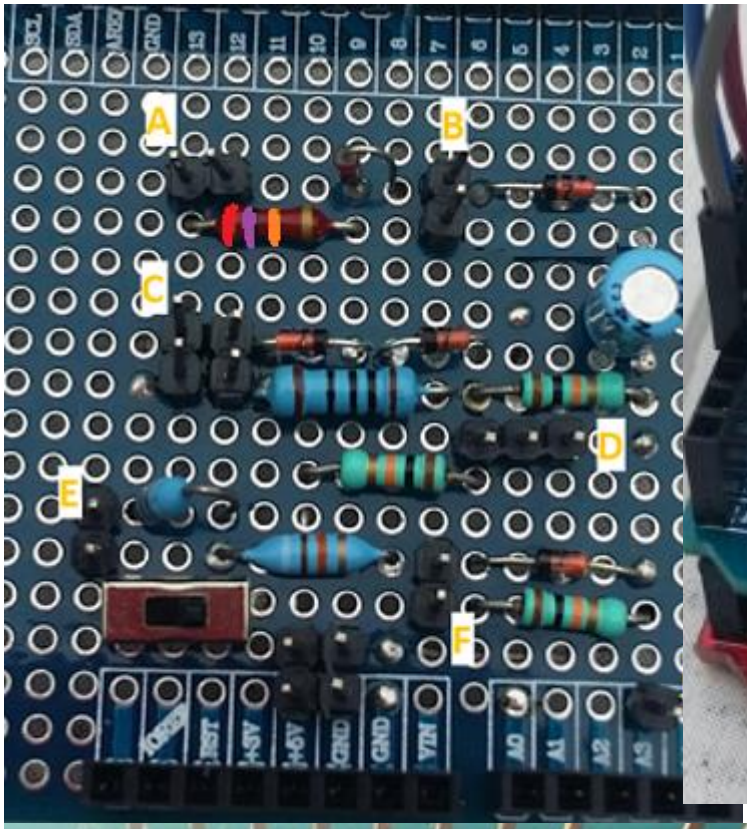
Le capteur de courant primaire est constitué du module ACS712, qui fournit une tension de mesure comprise entre 0 et 5V, et proportionnelle au courant qui le traverse.

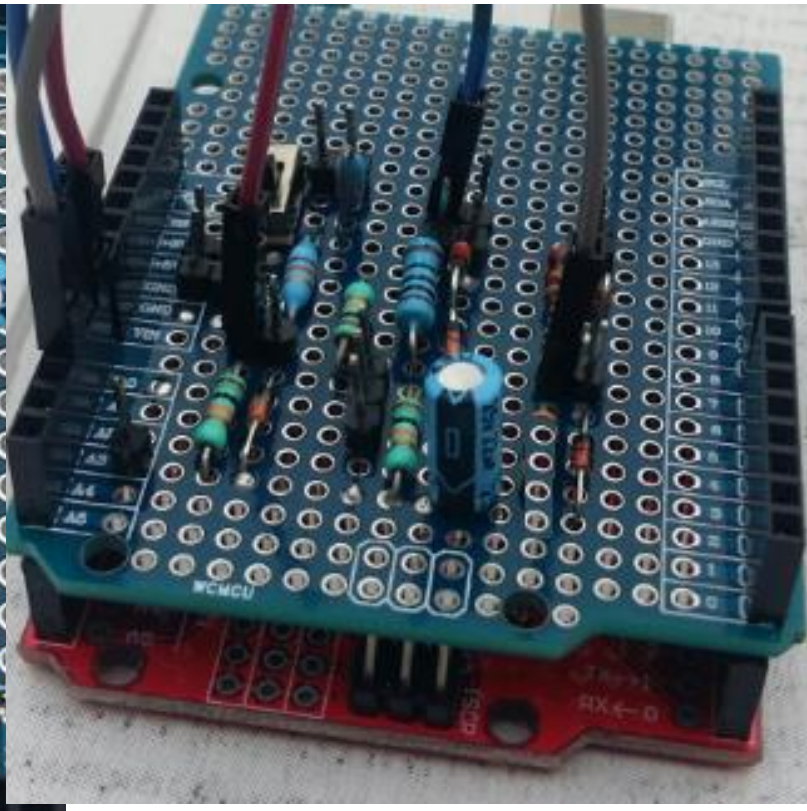
Le capteur de courant d'injection est constitué d'un capteur à effet Hall, c'est une sorte de bobinage traversé par le fil dont on veut effectuer la mesure. Cette méthode préserve de manipuler la haute tension du secteur…
Pour fonctionner correctement ce bobinage doit être connecté à une résistance de Burden, qui peut être dans le capteur lui-même. Aux bornes de cette résistance il y aura une tension (alternative) proportionnelle au courant alternatif parcouru dans le fil. Pour ramener cette tension tantôt négative tantôt positive sur une fourchette de 0 à 5V, un côté de Rburden est branché au pont diviseur R4 et R5 où le potentiel vaut 2,5V. C1 découple le pont diviseur pour éliminer tout résidu alternatif. 2 zéners tête-bêche de 4,7V aux bornes de Rburden limitent l'amplitude de la tension à 5V.

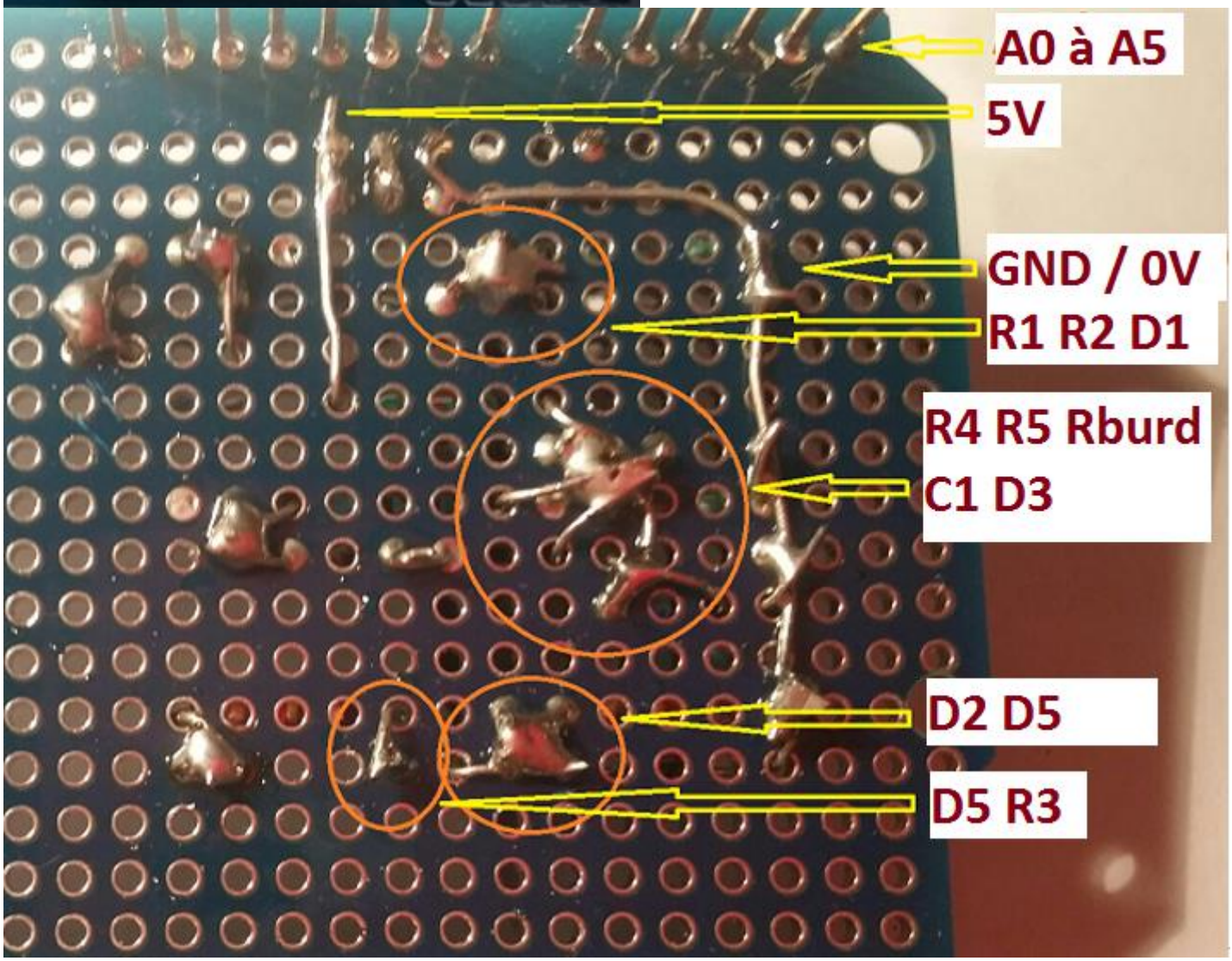
Quelques photos du circuit autour des capteurs

Comme déjà fait pour le montage de sécurité étudié précédemment, R1 est composée de 2 résistance en série de 91K avec un interrupteur miniature permettant le shunt de l'une des 2. Cette configuration permet les essais aussi bien sur des éoliennes de 24V que de 48V.









Liste des lettres :

A- Vers une des 3 phases de l'éolienne pour mesure de la fréquence du signal alternatif,

B- Vers A2 et 3 pour mesure de la vitesse de rotation de l'éolienne Fprimaire,
C- Vers A1 + Sonde de courant alternatif de mesure du courant d'injection Isortie,
D- Vers sonde de courant alternatif de mesure du courant d'injection,
E- Vers le pôle + du pont de diode pour mesure de la tension primaire,
F- Vers A0 pour mesure de la tension Vprimaire
G- (=broche A3) Vers module ACS712 pour mesure du courant primaire Iprimaire.

Premier test de mesures des capteurs

1- Sans enficher la carte d'extension dans l'Arduino, brancher l'Arduino sur le port USB du PC
2- Si ce n'est déjà fait, Installer l'interface de programmation, le programme est téléchargeable ici : https://www.arduino.cc/en/Main/OldSoftwareReleases
3- On installe les drivers pour l'arduino UNO : http://283.mytrademe.info/ch340.html
4- On lance le programme Arduino :
1- Menu outils-> type de carte-> UNO
2- Menu outils-> PORT-> ComX ou X représente le port sur lequel est installé votre arduino.
3- On efface ce qu'il y a dans la fenêtre d'édition
4- On y colle le code ci-dessous :

```
// minMaxAndRangeChecker
// A simple tool to investigate the ADC values that are seen at the
// first four analogue inputs of an Atmega chip, as used on an emonTx
//
// Robin Emley (calypso_rae on the Open Energy Monitor forum)
//
// 20th April 2013
//

int val_a0, val_a1, val_a2, val_a3;
int minVal_a0, minVal_a1, minVal_a2, minVal_a3;
int maxVal_a0, maxVal_a1, maxVal_a2, maxVal_a3;

int loopCount = 0;
unsigned long timeAtLastDisplay = 0;
byte displayLineCounter = 0;

void setup(void) {
  Serial.begin(9600);
  Serial.print("ready ...");
  delay(700);
  Serial.println ();
  Serial.println(" The Min, Max and Range ADC values for analog inputs 0 to 3:");
}

void loop(void) {
  val_a0 = analogRead(0);
  val_a1 = analogRead(1);
  val_a2 = analogRead(2);
  val_a3 = analogRead(3);

  if (val_a0 < minVal_a0) { minVal_a0 = val_a0;}
  if (val_a0 > maxVal_a0) { maxVal_a0 = val_a0;}
  if (val_a1 < minVal_a1) { minVal_a1 = val_a1;}
  if (val_a1 > maxVal_a1) { maxVal_a1 = val_a1;}
  if (val_a2 < minVal_a2) { minVal_a2 = val_a2;}
  if (val_a2 > maxVal_a2) { maxVal_a2 = val_a2;}
  if (val_a3 < minVal_a3) { minVal_a3 = val_a3;}
  if (val_a3 > maxVal_a3) { maxVal_a3 = val_a3;}

  unsigned long timeNow = millis();
  if ((timeNow - timeAtLastDisplay) >= 3000) {
    timeAtLastDisplay = timeNow;
    displayVal(minVal_a0);
    displayVal(maxVal_a0);
    displayVal(maxVal_a0 - minVal_a0);
    Serial.print(";  ");

    displayVal(minVal_a1);
    displayVal(maxVal_a1);
    displayVal(maxVal_a1 - minVal_a1);
    Serial.print(";  ");

    displayVal(minVal_a2);
```

```
    displayVal(maxVal_a2);
    displayVal(maxVal_a2 - minVal_a2);
    Serial.print(";  ");

    displayVal(minVal_a3);
    displayVal(maxVal_a3);
    displayVal(maxVal_a3 - minVal_a3);
    Serial.println();

    resetMinAndMaxValues();
    displayLineCounter++;

    if (displayLineCounter >= 5) {
      Serial.println();
      displayLineCounter = 0;
      delay(2000); // to allow time for data to be accessed
    }
  }
}

void resetMinAndMaxValues() {
  minVal_a0 = 1023, minVal_a1 = 1023, minVal_a2 = 1023, minVal_a3 = 1023;
  maxVal_a0 = 0, maxVal_a1 = 0, maxVal_a2 = 0, maxVal_a3 = 0;
}
void displayVal(int intVal){
  char strVal[4];
  byte lenOfStrVal;
  itoa(intVal, strVal, 10); // decimal conversion to string
  lenOfStrVal = strlen(strVal); // determine length of string

  for (int i = 0; i < (4 - lenOfStrVal); i++)  {
    Serial.print(' ');
  }

  Serial.print(strVal);
}
```

5- Enregistrer ce programme sur le PC sous le nom de testminmax.ino par exemple.

6- Puis : Menu croquis -> téléverser.

7- Ouvrir le moniteur série : Menu outils -> moniteur série : une autre fenêtre s'ouvre. Régler le débit (baud rate à 9600 si ce n'est pas le réglage par défaut)

Nous devons obtenir les résultats suivants :

Les entrées sont en l'air, c'est proche de zéro mais pas tout à fait.

```
The Min, Max and Range ADC values for analog inputs 0 to 3:
  0 340 340;     0 334 334;     0 326 326;     0 318 318
  0  65  65;     0  62  62;     0  59  59;     0  68  68
  0  66  66;     0  62  62;     0  60  60;     0  68  68
  0  68  68;     0  64  64;     0  60  60;     0  70  70
```

8- Débrancher la carte Arduino, insérer la carte d'extension sans brancher les capteurs.

9- Rebrancher la carte Arduino, et réouvrir le moniteur série.

Nous devons obtenir les résultats suivants :

La première ligne n'est pas significative, il faut laisser le temps au programme d'effectuer les calculs après initialisation des variables.

```
The Min, Max and Range ADC values for analog inputs 0 to 3:
  0   0   0;     0 513 513;     0 102 102;     0 511 511;
  0   2   2;   509 514   5;    55 145  90;   508 514   6;
  0   2   2;   509 514   5;    55 145  90;   507 515   8;
  0   2   2;   508 514   6;    55 145  90;   508 515   7;
```

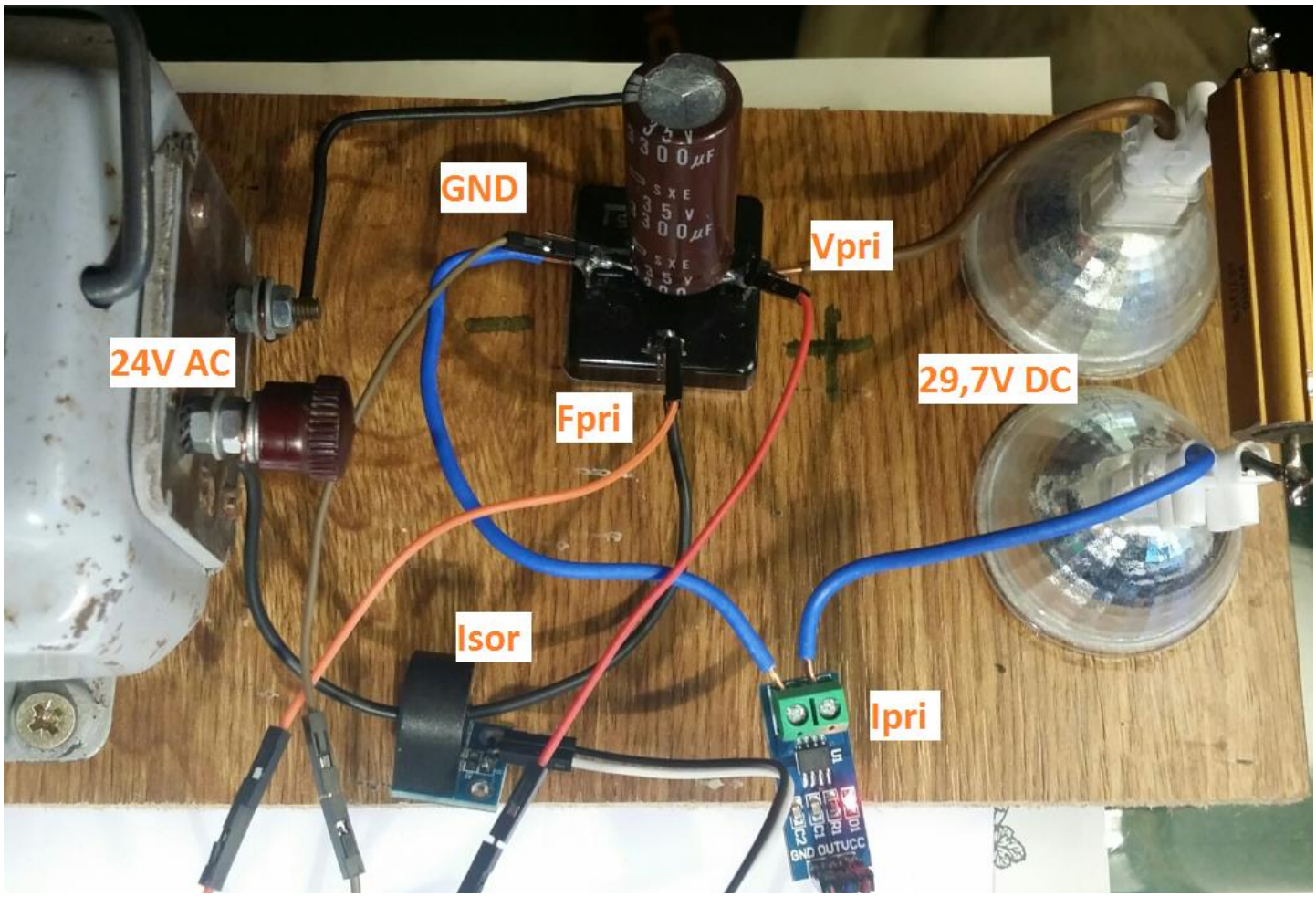
Les entrées analogiques passent par un convertisseur numérique qui fournit 0 à 1024 bits pour une valeur lue de 0 à 5V.

A vérifier : A0 = 0 avec un bruit de l'ordre de 2 bits, A1 = 512 (soit 2,5V ce qui correspond au 5V divisé par R4 et R5) avec un bruit de 6 bits, A2 est branché sur une diode sans courant, c'est comme si elle n'était pas connectée, et enfin A3 donne 512 (soit 2,5V délivrée par l'ACS712) avec un bruit de 6 bits. Décidément ces CAN ont une mauvaise précision à bas niveau, ce qui ne nous arrange pas pour les mesures de petites puissances.

10- Brancher les capteurs :

Nous pouvons fabriquer un environnement d'essais à partir d'une alimentation expérimentale avec un transformateur 230V/20V suivi d'un pont de diode, d'un condensateur de filtrage – ce qui permet d'avoir une tension continue et non

pas une succession de demie-alternances positives - et une charge constituée de 2 ampoules halogènes 12V/20W branchées en série. Ainsi nous avons des grandeurs continue et alternatives pour effectuer les essais.



Nous obtenons le résultat suivant sur la console :

```
The Min, Max and Range ADC values for analog inputs 0 to 3: +ms +Hz
  0 594 594;      0 351 351;      0 874 874;      0 545 545;  19.74 50.66
570 627  57;    350 673 323;      0 878 878;    539 549  10;  19.73 50.67
571 628  57;    349 672 323;      0 878 878;    539 549  10;  19.73 50.69
571 628  57;    349 673 324;      0 878 878;    539 549  10;  19.73 50.68
```

Les mesures au multimètre donnent les informations : Vac = 24V / 50Hz, Vprimaire = 29,7V, Isortie = 2,55A

- A0 indique 571 à 628, un tel delta est dû à l'ondulation du signal. Une mesure charge débranchée (sans les ampoules halogènes) donne 636 à 642 pour 34V.
- A1 indique Isortie, qui oscille de 349 à 673, c'est normal puisque le courant est alternatif, la grandeur à mesurer est le delta 324.
- A2 indique Fpri, la mesure est une oscillation qui va de 0 à 878.
- A3 indique Ipri, la sortie de l'ACS712 va de 539 à 549, par rapport à 512. L'écart de 10 est lié à l'ondulation résiduelle en plus du bruit.

Vous obtenez grosso-modo les mêmes résultats ? Alors on continue.

Passer des bits à des valeurs normalisées

Vpri : il suffit d'appliquer une règle de 3. Si « 639 » (moyenne de 636 à 642) indique 34V, il y a donc 18,8 bits par volt.

Ipri : le module ACS712 envoie « 512 » pour 0A, « 0 » pour -20A et « 1023 » pour +20A. Déjà on remarquera que si la mesure est négative il suffit d'inverser le sens de la sonde. Il s'agit donc de convertir les bits en tension, puis d'appliquer la conversion courant/tension suivante :

- Module ACS712 de 5A => 185mV/A
- Module ACS712 de 20A => 100mV/A
- Module ACS712 de 30A => 66mV/A

Fpri : l'instruction `pulseln(pin, HIGH)` permet de calculer le nombre de microseconde entre 2 instants où le signal est à l'état haut. De même `pulseln(pin, LOW)` fait la même chose pour l'état bas. La somme de ces 2 mesures de temps donne la période du signal. Cette instruction ne fonctionne que sur une entrée numérique, d'où le choix de l'entrée 3.

Isor : Il est plus compliqué de mesurer un courant ou une tension alternative, parce que justement elle n'est pas statique. Un procédé consiste à prendre un échantillonnage de mesures sur un temps donné, de déterminer la valeur min et max, et d'en tirer la valeur efficace. Il est possible de réaliser cette programmation qui est la base de fonctionnement de testminmax.ino. Cependant il existe une librairie qui fait le job : *EmonLib.h*
A voir ici : https://learn.openenergymonitor.org/electricity-monitoring/ctac/how-to-build-an-arduino-energy-monitor

Utiliser un afficheur LCD 1602

A terme les données seront stockées sur une carte SD, néanmoins il est intéressant d'avoir une vue globale que les mesures sans nécessiter le branchement d'un PC sur le port USB et d'ouvrir le moniteur série.

Nous allons utiliser l'afficheur le plus classique qui soit : le LCD 1602 pour 16 caractères sur 2 lignes, avec l'extension I2C qui réduit considérablement la complexité du montage qui se résume à 2 fils de data. Toutes les infos sont décrites là : https://andrologiciels.wordpress.com/arduino/lcd/lcd-1602-i2c/

La bibliothèque est ici : https://app.box.com/s/czde88f5b9vpulhf8z56

Attention ! L'installation de la bibliothèque met à disposition une nouvelle bibliothèque *LiquidChrystal.h* ; il faut donc absolument supprimer – ou renommer – tous les fichiers de même nom : *LiquidCrystal.h* et *LiquidCrystal.cpp*, qui se trouvent quelque part sous le répertoire *C:\Program Files (x86)\Arduino\librairies*. Heureusement il n'y a rien de destructifs, les nouvelles librairies apportant les fonctionnalités compatibles avec les LCD classiques.

Programme initial de banc de test

1- Télécharger la librairie *emonLib.h* : https://github.com/openenergymonitor/EmonLib/archive/master.zip
2- Télécharger la librairie *LiquidCrystal_I2C.h* : https://app.box.com/s/czde88f5b9vpulhf8z56
3- Les déclarer : (In the Arduino IDE) Sketch > Include Library > Add .ZIP Library > select the downloaded file > Open
4- Ajouter l'extension *.orig* à *LiquidCrystal.h* et *LiquidCrystal.cpp*, qui se trouvent quelque part sous le répertoire *C:\Program Files (x86)\Arduino\librairies*
5- Créer un nouveau programme avec le code ci-dessous :

```
/*
Banc de mesures pour éoliennes 24 ou 48V
_____________________________________________________________
|                                                            |
|       auteur : Philippe de Craene <dcphilippe@yahoo.fr     |
|             pour l' Association PTIWATT                    |
_____________________________________________________________

*/

#include <Wire.h>
#include <LiquidCrystal_I2C.h> // https://app.box.com/s/czde88f5b9vpulhf8z56
```

```
// Attention ! Renomer la librairie d'origine de l'IDE LiquidCrystal.h et LiquidCrystal.cpp
#include <EmonLib.h>           // https://github.com/openenergymonitor/EmonLib
EnergyMonitor injection;       // Création de l'instance injection

// Brochage des entrées des capteurs et paramétrage

const byte broche_Vpri = A0;   // broche lecture tension pont de diodes
const byte broche_Isor = A1;   // broche lecture courant d'injection
const byte broche_Ipri = A3;   // broche lecture courant pont de diodes
const byte broche_Fpri = 3;    // broche numérique calcul fréquence

// choisir le modèle d'éolienne 24V ou 48V à vérifier expérimentalement :
// éolienne 24V avec maxi à 54V => conv_Vpri = 20.2
// éolienne 48V avec maxi à 95V => conv Vpri = 11.2
const float conv_Vpri = 20.2;  // conversion bytes/tension pour éolienne 24V

// Choisir le modèle de sonde ACS712 :
// Module ACS712 de 5A => 0.185V/A
// Module ACS712 de 20A => 0.1V/A
// Module ACS712 de 30A => 0.066mV/A
const float conv_Ipri = 0.1;   //  pour modèle 20A

// Calibration de Isor à définir expérimentalement suivant la sonde utilisée
// => valeur à saisir dans le setup()
// offset pour supprimer le bruit pour avoir 0 en absence de courant
const float Isor_offset = 0.06;

// Variables de traitement

float Vpri;                 // Tension au pont de diode
float Ipri;                 // Courant délivré au pont de diode
float Fpri;                 // Vitesse de rotation
int Vsor = 230;             // Tension du réseau
float Isor;                 // Courant injecté sur le réseau
unsigned long duree_H;      // durée de la demie période état haut
unsigned long duree_L;      // durée de la demie période état bas

int B_Vpri, B_Ipri;         // lecture des capteurs en bytes

// Déclaration du LCD en mode I2C :
// toutes les infos ici : http://arduino-info.wikispaces.com/LCD-Blue-I2C
// Set the pins on the I2C chip used for LCD connections:
//                    addr, en,rw,rs,d4,d5,d6,d7,bl,blpol
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
// => connexion des 2 broches I2C sur l'Arduino Uno R3 : SDA en A4, SCL en A5

//
// SETUP
//_____________________________________________________________________________________________

void setup() {

  pinMode(broche_Fpri, INPUT);

// Calibration de Isor à définir expérimentalement suivant la sonde utilisée
  injection.current(broche_Isor, 9.85);    // Current: input pin, calibration.

// initialisation de l'affichage et du mode console
  Serial.begin(9600);            // préparation du moniteur série
  Serial.println ();
  Serial.println("ready ...");
  Serial.println ();

// initialisation du LCD
  lcd.begin(16,2);               // initialize the lcd for 16 chars 2 lines
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("*** BONJOUR ***");
  lcd.setCursor(0, 1);
  lcd.print("    *******");

  delay(1500);

}                                 // fin de setup

//
// LOOP
//_____________________________________________________________________________________________

void loop() {

// Calcul de Vprimaire
  B_Vpri = analogRead( broche_Vpri );
  Vpri = B_Vpri / conv_Vpri;

// Calcul de Iprimaire
  B_Ipri = analogRead( broche_Ipri );
```

```
  Ipri = abs(( B_Ipri * 5.0 / 1023.0 ) - 2.5 ) / conv_Ipri;

// Calcul de Isortie    //initialement Isor = injection.calcIrms(1480);
// https://openenergymonitor.org/forum-archive/node/846.html
//  Isor donne 0.06mA en l'absence de courant, dû au bruit des 2 derniers bits du CAN
//  donc on soustrait 0.06 à la lecture et on coupe la résolution à 0.01
  Isor = 0.01 * ( abs( int( 100 * (injection.calcIrms(1172) - Isor_offset))));

// Calcul de la vitesse de rotation Fpri
  duree_H = pulseIn(broche_Fpri, HIGH);
  duree_L = pulseIn(broche_Fpri, LOW);
  if( duree_H + duree_L > 1 ) { Fpri = 1000000.0/(duree_H + duree_L);}
  else { Fpri = 0;}

// Affichage des résultats sur le moniteur série
  Serial.println(" Vpri  | Ipri  || Isor  | Fpri ");
  Serial.print(Vpri);
  Serial.print("V | ");
  Serial.print(Ipri);
  Serial.print("A || ");
  Serial.print(Isor);
  Serial.print("A | ");
  Serial.print(Fpri);
  Serial.println("Hz ");

// Affichage du résultat sur le LCD
  lcd.setCursor(0, 0);
  lcd.print(String(Vpri,1));
  lcd.print("V ");
  if( Vpri < 10) { lcd.print(" ");}
  lcd.print("Pe=");
  if( Vpri*Ipri < 100) { lcd.print(" ");}
  if( Vpri*Ipri < 10) { lcd.print(" ");}
  lcd.print(String(Vpri*Ipri,1));
  lcd.print("W ");
  lcd.setCursor(0, 1);
  lcd.print(String(Fpri,0));
  lcd.print("Hz ");
  lcd.print(" Ps=");
  if( Vsor*Isor < 100) { lcd.print(" ");}
  if( Vsor*Isor < 10) { lcd.print(" ");}
  lcd.print(String(Vsor*Isor,1));
  lcd.print("VA");

}
```

6- Enregistrer ce programme sur le PC sous le nom de mesures_eolienne.ino par exemple.

7- Puis : Menu croquis -> téléverser.

8- Ouvrir le moniteur série :

Montage d'essai éteint :

```
 Vpri  | Ipri  || Isor  | Fpri
0.00V | 0.07A || -0.01A | 0.00Hz
 Vpri  | Ipri  || Isor  | Fpri
0.00V | 0.07A || 0.00A | 0.00Hz
 Vpri  | Ipri  || Isor  | Fpri
0.00V | 0.07A || 0.00A | 0.00Hz
```

Montage d'essai allumé :

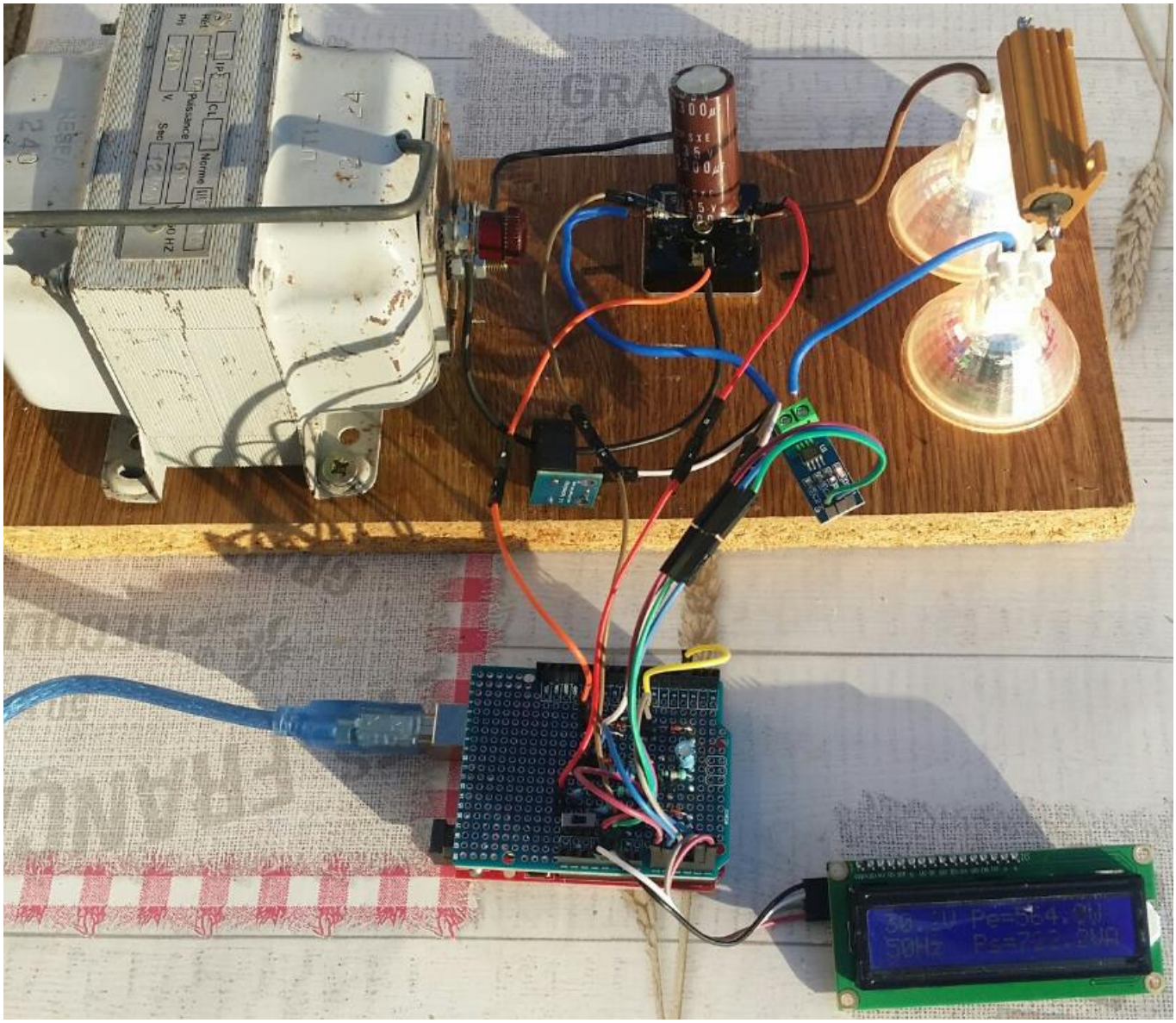
```
 Vpri  | Ipri  || Isor  | Fpri
30.35V | 1.69A || 3.02A | 50.32Hz
 Vpri  | Ipri  || Isor  | Fpri
30.35V | 1.69A || 3.02A | 50.33Hz
 Vpri  | Ipri  || Isor  | Fpri
30.40V | 1.69A || 3.02A | 50.32Hz
```

Le montage d'essai éteint, ajuster :

- `const float Isor_offset = 0.06;` => pour obtenir Isor =0 (c'est au 1/100ème près !)

Le montage d'essai allumé, ajuster :

- `const float conv_Vpri = 20.2;` => pour Vpri = valeur du multimètre
- `injection.current(broche_Isor, 9.85);` => pour Isor = valeur du multimètre



Mise en œuvre du module de carte SD et mesure du temps

Il eut été trop simple qu'il n'exista qu'une seule version de module de carte SD. Il en existe 2 ! La nouvelle qui utilise les librairies déjà installées sur le programme PC, et l'ancienne version … ET la carte dont j'ai donné le lien dans la liste des courses est l'ancienne version.

Il faut donc installer les bonnes librairies, et surtout retrouver *SD.h* et *SD.cpp* quelque part dans *C:\Program Files (x86)\Arduino\librairies\SD\src* et les renommer pour les rendre inutilisables, par exemple en leur ajoutant l'extension *.orig*.

Toutes les informations sont disponibles ici : https://learn.adafruit.com/adafruit-data-logger-shield/overview

La procédure pour installer la librairie : https://learn.adafruit.com/adafruit-data-logger-shield/for-the-mega-and-leonardo

Et le lien de téléchargement de la bibliothèque : https://github.com/adafruit/SD

Ensuite il faudra installer la librairie pour utiliser la partie RTC du module. Toutes les information sont disponibles ici : https://learn.adafruit.com/adafruit-data-logger-shield/using-the-real-time-clock

Et le lien de téléchargement de la bibliothèque : https://github.com/adafruit/RTClib/archive/master.zip

A l'aide d'une loupe il faudra regarder la référence du circuit de gestion du temps : *PCF8523* ou *DS1307.* Celui du lien donné en liste des courses est l'ancien modèle DS1307.

Ensuite à la 1ère utilisation du module, ou bien à chaque fois qu'elle n'est plus alimentée en énergie – il parait que la pile possède une durée de vie de 5 ans – il faudra programmer l'heure avec l'instruction :
`rtc.adjust(DateTime(2018, 10, 14, 14, 51, 0));` pour 14 octobre 2018 à 14h51. Sinon `rtc.isrunning()` ne se lance pas.

Test d'écriture de la date et l'heure sur la carte SD

1- Télécharger la librairie *SD.h* : https://github.com/adafruit/SD
2- Télécharger la librairie *RTClib.h* : https://app.box.com/s/czde88f5b9vpulhf8z56
3- Les déclarer : (In the Arduino IDE) Sketch > Include Library > Add .ZIP Library > select the downloaded file > Open
4- Ajouter l'extension *.orig* à *SD.h* et *SD.cpp* dans *C:\Program Files (x86)\Arduino\librairies\SD\src* et les renommer pour les rendre inutilisables.
5- Créer un nouveau programme avec le code ci-dessous :

```
// Date et l'heure enregistées sur carte SD

#include <Wire.h>
#include <RTClib.h>
#include <SPI.h>
#include <SD.h>

// On the Ethernet Shield, CS is pin 4. Note that even if it's not
// used as the CS pin, the hardware CS pin (10 on most Arduino boards,
// 53 on the Mega) must be left as an output or the SD library
// functions will not work.
const int chipSelect = 4;

File dataFile;

// Choix du modèle de RTC
RTC_DS1307 rtc;
//RTC_PCF8523 rtc;

char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday"};

void setup () {
  Serial.begin(9600);
  if(! rtc.begin()) { Serial.println("Couldn't find RTC"); while (1);}
//  if(! rtc.isrunning()) { Serial.println("RTC is NOT running!");

// Mise à l'heure du module qui ne sera fait qu'une seule fois :
// suivant le modèle suivant :
// pour 14 octobre 2018 à 15h30 :
  rtc.adjust(DateTime(2018, 10, 14, 15, 30, 0));
//  }

  Serial.print("Initializing SD card...");
  // make sure that the default chip select pin is set to
  // output, even if you don't use it:
  pinMode(SS, OUTPUT);

  // see if the card is present and can be initialized:
  if (!SD.begin(chipSelect)) {
    Serial.println("Card failed, or not present");
    // don't do anything more:
    while (1) ;
  }
  Serial.println("card initialized.");

  // Open up the file we're going to log to!
  dataFile = SD.open("datalog.txt", FILE_WRITE);
  if (! dataFile) {
    Serial.println("error opening datalog.txt");
    // Wait forever since we cant write data
    while (1) ;
  }
}

void loop () {
```

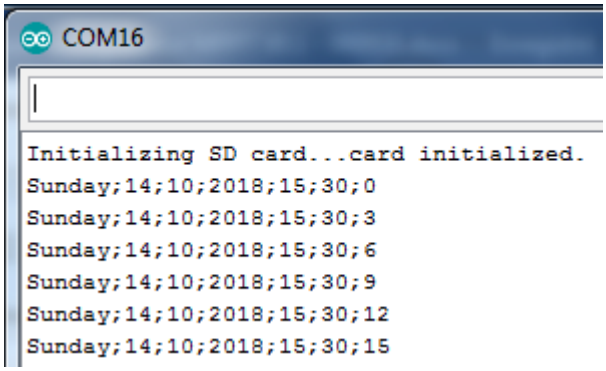
```
  // make a string for assembling the data to log:
  String dataString = "";

  DateTime now = rtc.now();

  dataString += String(daysOfTheWeek[now.dayOfTheWeek()]);
  dataString += ";";
  dataString += String(now.day(), DEC);
  dataString += ";";
  dataString += String(now.month(), DEC);
  dataString += ";";
  dataString += String(now.year(), DEC);
  dataString += ";";
  dataString += String(now.hour(), DEC);
  dataString += ";";
  dataString += String(now.minute(), DEC);
  dataString += ";";
  dataString += String(now.second(), DEC);

  dataFile.println(dataString);     // enregistre la ligne
  Serial.println(dataString);       // et l'affiche à la console

// The following line will 'save' the file to the SD card after every line of data
// this will use more power and slow down how much data you can read but it's safer!
// If you want to speed up the system, remove the call to flush() and it will save
// the file only every 512 bytes - every time a sector on the SD card is filled.
  dataFile.flush();

  delay(3000);
}
```

6- Effectuer la mise à jour de la ligne correspondante à la déclaration de la date et l'heure

7- Enregistrer ce programme sur le PC sous le nom de test_module_SD_RTC.ino par exemple.

8- Puis : Menu croquis -> téléverser.

9- Ouvrir le moniteur série, si les données sont bonnes insérer la pile dans son compartiment.



A noter : en principe ce programme n'est à exécuter qu'une fois, sinon la date et l'heure seront systématiquement réinitialisées à la valeur saisie.

Ensuite nous devons retrouver le même résultat dans le fichier *datafile.txt* présent sur la carte SD.

Programme final du banc de mesures avec enregistrement sur carte SD

```
/*

Banc de mesures pour éoliennes 24 ou 48V sur injecteur réseau, avec
enregistrement sur carte SD

Liste des mesures :
Vitesse de rotation              Fprimaire
Tension délivrée au pont de diode      Vprimaire (DC)
Courant délivré      au pont de diode   Iprimaire (DC)
Courant injecté sur le réseau          Isortie (AC)
On suppose que la tension du réseau est stable à 230V.

_____________________________________________________________
|                                                           |
|       auteur : Philippe de Craene <dcphilippe@yahoo.fr        |
|           pour l' Association P'TIWATT                    |
_____________________________________________________________

Toute contribution en vue de l'amélioration de l'appareil est la bienvenue ! Il vous est juste
demandé de conserver mon nom et mon email dans l'entête du programme, et bien sûr de partager
avec moi cette amélioration. Merci.

*/

#include <Wire.h>
#include <LiquidCrystal_I2C.h> // https://app.box.com/s/czde88f5b9vpulhf8z56
// Attention ! Renomer la librairie d'origine de l'IDE LiquidCrystal.h et LiquidCrystal.cpp
#include <SD.h>                // https://github.com/adafruit/SD
// Attention ! Renomer la librairie d'origine de l'IDE SD.h et SD.cpp
#include <EmonLib.h>           // https://github.com/openenergymonitor/EmonLib
#include <RTClib.h>            // https://github.com/adafruit/RTClib/archive/master.zip
#include <SPI.h>

EnergyMonitor injection;       // Création de l'instance injection pour EmonLib.h
File dataFile;                 // Création de l'instance pour gérer le fichier sur SD

// Brochage des entrées des capteurs et paramétrage

const byte broche_Vpri = A0;   // broche lecture tension pont de diodes
const byte broche_Isor = A1;   // broche lecture courant d'injection
const byte broche_Ipri = A3;   // broche lecture courant pont de diodes
const byte broche_Fpri = 3;    // broche numérique calcul fréquence
// On the Ethernet Shield, CS is pin 4. Note that even if it's not used as the CS pin, the
// hardware CS pin must be left as an output or the SD library functions will not work.
const byte chipSelect  = 4;

// choisir le modèle d'éolienne 24V ou 48V à vérifier expérimentalement :
// éolienne 24V avec maxi à 54V => conv_Vpri = 20.2
// éolienne 48V avec maxi à 95V => conv Vpri = 11.2
const float conv_Vpri = 20.2;  // conversion bytes/tension pour éolienne 24V

// Choisir le modèle de sonde ACS712 :
// Module ACS712 de 5A => 0.185V/A
// Module ACS712 de 20A => 0.1V/A
// Module ACS712 de 30A => 0.066mV/A
const float conv_Ipri = 0.1;   //  pour modèle 20A

// Calibration de Isor à définir expérimentalement suivant la sonde utilisée
// => valeur à saisir dans le setup()
// offset pour supprimer le bruit pour avoir 0 en absence de courant
const float Isor_offset = 0.059;

// Choisir le modèle de RTC sur le module carte SD :
RTC_DS1307 rtc;
//RTC_PCF8523 rtc;

// Variables de traitement

float Vpri;                 // Tension au pont de diode
float Ipri;                 // Courant délivré au pont de diode
float Fpri;                 // Vitesse de rotation
int Vsor = 230;             // Tension du réseau
float Isor;                 // Courant injecté sur le réseau
unsigned long duree_H;      // durée de la demie période état haut
unsigned long duree_L;      // durée de la demie période état bas
int B_Vpri, B_Ipri;         // lecture des capteurs en bytes
byte prev_second = 0;       // pour le compteur

char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday"};

// Déclaration du LCD en mode I2C :
```

```
// toutes les infos ici : http://arduino-info.wikispaces.com/LCD-Blue-I2C
// Set the pins on the I2C chip used for LCD connections:
//                    addr, en,rw,rs,d4,d5,d6,d7,bl,blpol
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
// => connexion des 2 broches I2C sur l'Arduino Uno R3 : SDA en A4, SCL en A5

//
// SETUP
//_____________________________________________________________________________

void setup() {

  pinMode(broche_Fpri, INPUT);

// Calibration de Isor à définir expérimentalement suivant la sonde utilisée
  injection.current(broche_Isor, 8.35);    // Current: input pin, calibration.

// initialisation de l'affichage et du mode console
  Serial.begin(9600);            // préparation du moniteur série
  Serial.println ();
  if(! rtc.begin()) { Serial.println("RTC introuvable");}
  if(! rtc.isrunning()) { Serial.println("Date et heure à programmer sur le RTC");}
  Serial.print("Initializing SD card...");
// make sure that the default chip select pin is set to output, even if you don't use it:
  pinMode(SS, OUTPUT);
// see if the card is present and can be initialized:
  if (!SD.begin(chipSelect)) {  Serial.println("Card failed, or not present"); }
  else { Serial.println("card initialized."); }
// Open up the file we're going to log to!
  dataFile = SD.open("datalog.txt", FILE_WRITE);
  if (! dataFile) { Serial.println("erreur sur fichier datalog.txt"); }
  Serial.println("ready ...");
  Serial.println ();

// initialisation du LCD
  lcd.begin(16,2);               // initialize the lcd for 16 chars 2 lines
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("*** BONJOUR ***");
  lcd.setCursor(0, 1);
  lcd.print("    *******");

  delay(1500);

}                                  // fin de setup

//
// LOOP
//_____________________________________________________________________________

void loop() {

// Calcul de Vprimaire
  B_Vpri = analogRead( broche_Vpri );
  Vpri = B_Vpri / conv_Vpri;

// Calcul de Iprimaire
  B_Ipri = analogRead( broche_Ipri );
  Ipri = abs(( B_Ipri * 5.0 / 1023.0 ) - 2.5 ) / conv_Ipri;

// Calcul de Isortie    //initialement Isor = injection.calcIrms(1480);
// https://openenergymonitor.org/forum-archive/node/846.html
//  Isor donne 0.06mA en l'absence de courant, dû au bruit des 2 derniers bits du CAN
//  donc on soustrait 0.06 à la lecture et on coupe la résolution à 0.01
  Isor = 0.01 * ( abs( int( 100 * (injection.calcIrms(1172) - Isor_offset))));

// Calcul de la vitesse de rotation Fpri
  duree_H = pulseIn(broche_Fpri, HIGH);
  duree_L = pulseIn(broche_Fpri, LOW);
  if( duree_H + duree_L > 1 ) { Fpri = 1000000.0/(duree_H + duree_L);}
  else { Fpri = 0;}

  DateTime now = rtc.now();        // récupération de la date et l'heure
  if( now.second() != prev_second ) {    // toutes les secondes

// Affichage du résultat sur le LCD

    lcd.setCursor(0, 0);
    lcd.print(String(Vpri,1));
    lcd.print("V ");
    if( Vpri < 10) { lcd.print(" ");}
    lcd.print("Pe=");
    if( Vpri*Ipri < 100) { lcd.print(" ");}
    if( Vpri*Ipri < 10) { lcd.print(" ");}
    lcd.print(String(Vpri*Ipri,1));
    lcd.print("W ");
    lcd.setCursor(0, 1);
```

```
    lcd.print(String(Fpri,0));
    lcd.print("Hz ");
    lcd.print(" Ps=");
    if( Vsor*Isor < 100) { lcd.print(" ");}
    if( Vsor*Isor < 10) { lcd.print(" ");}
    lcd.print(String(Vsor*Isor,1));
    lcd.print("VA");

// Affichage du résultat sur carte SD

    if( Fpri > 0 ) {                  // dès que l'éolienne tourne
      String dataString = "";         // initialisation d'une chaine de caractères
      dataString += String(daysOfTheWeek[now.dayOfTheWeek()]);
      dataString += ";";
      dataString += String(now.day(), DEC);
      dataString += ";";
      dataString += String(now.month(), DEC);
      dataString += ";";
      dataString += String(now.year(), DEC);
      dataString += ";";
      dataString += String(now.hour(), DEC);
      dataString += ";";
      dataString += String(now.minute(), DEC);
      dataString += ";";
      dataString += String(now.second(), DEC);
      dataString += ";";
      dataString += ";";
      dataString += String(Vpri);
      dataString += ";";
      dataString += String(Ipri);
      dataString += ";";
      dataString += String(Isor);
      dataString += ";";
      dataString += String(Fpri);
      dataFile.println(dataString);     // enregistre la ligne
      dataFile.flush();                 // l'enregistre sur SD
      Serial.println(dataString);       // et l'affiche à la console
    }      // fin test Fpri
    prev_second = now.second();
  }        // fin compteur de seconde
}          // fin de loop()
```

Banc de mesures pour configuration en stockage sur batteries :

Même si l'injection directe offre l'avantage d'une solution plus économique et écologique, il peut être nécessaire de posséder un système à batterie lorsque par exemple l'énergie produite en journée excède la consommation, qui peut être expliquée par la présence d'un panneau photovoltaïque. S'il y a du vent en journée l'énergie est stockée en batterie, et mise à disposition au réseau la nuit ou en absence de soleil.

A noter que le Power Router permet cette configuration. Avec le système de délestage, à 25W de l'injection – c'est-à-dire quand la différence entre la consommation et la production atteint le seuil de 25W, le Power Router à l'aide d'un SSR coupe la connexion de l'injecteur au réseau, forçant la production d'énergie éolien à la charge de batteries. L'injecteur lui-même peut produire une puissance supérieure à la consommation, ce qui le fera fonctionner par intermitance.

Dans cette version le banc de mesures va collecter les informations suivantes :

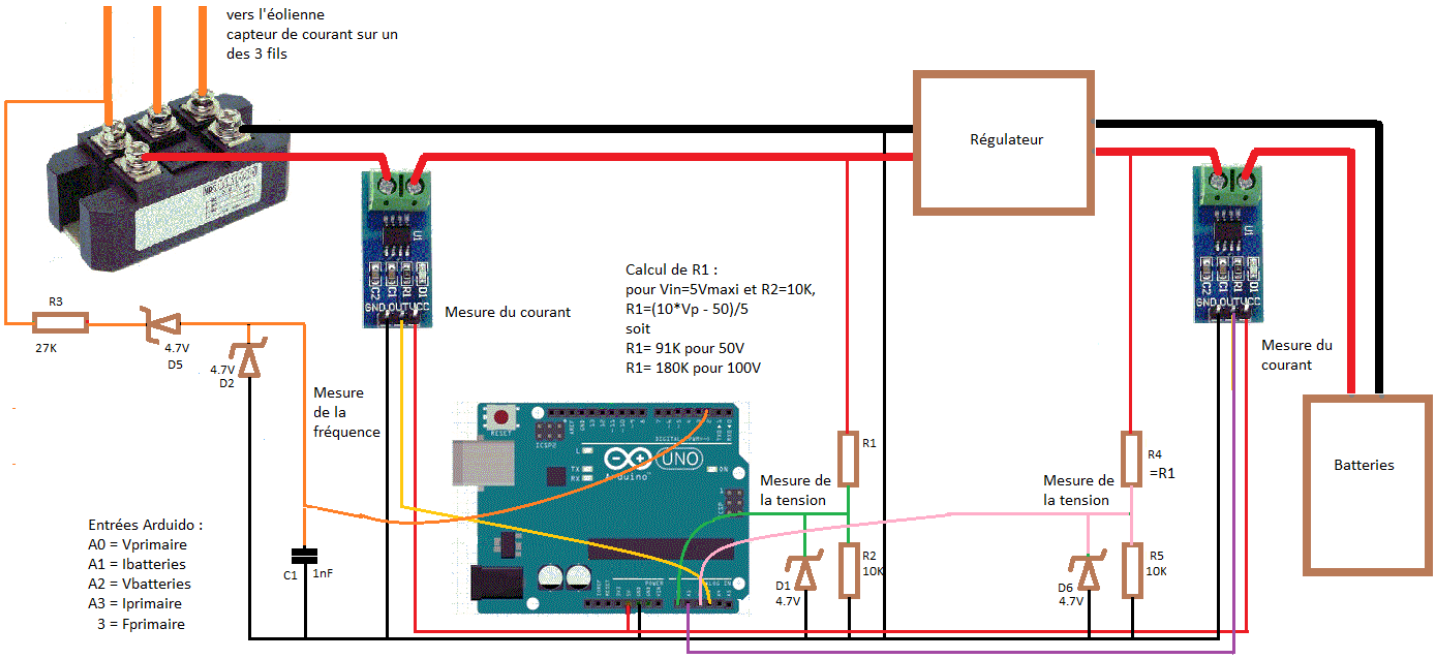
Si l'éolienne tourne, et dans ce cas à intervalles réguliers :

- Date et l'heure,
- Vitesse de rotation Fprimaire
- Tension délivrée au pont de diode Vprimaire
- Courant délivré au pont de diode Iprimaire
- Tension de charge des batteries Vsortie
- Courant de charge des batteries Isortie

Vprimaire * Iprimare = Pprimaire

Vsortie * Isortie = Psortie

Mise à jour du montage électrique



A noter : la sonde de courant Isortie est identique à Iprimaire. De même la sonde de tension Vsortie est de construction identique à celle pour Vprimaire : si jamais les batteries se coupaient, l'élévation de la tension sera mesurée sans risque de destruction de l'Arduino.

C1 a été rajouté car le régulateur MPPT pollue énormément la signal avec le découpage à haute fréquence. C1 est en parallèle avec D2.

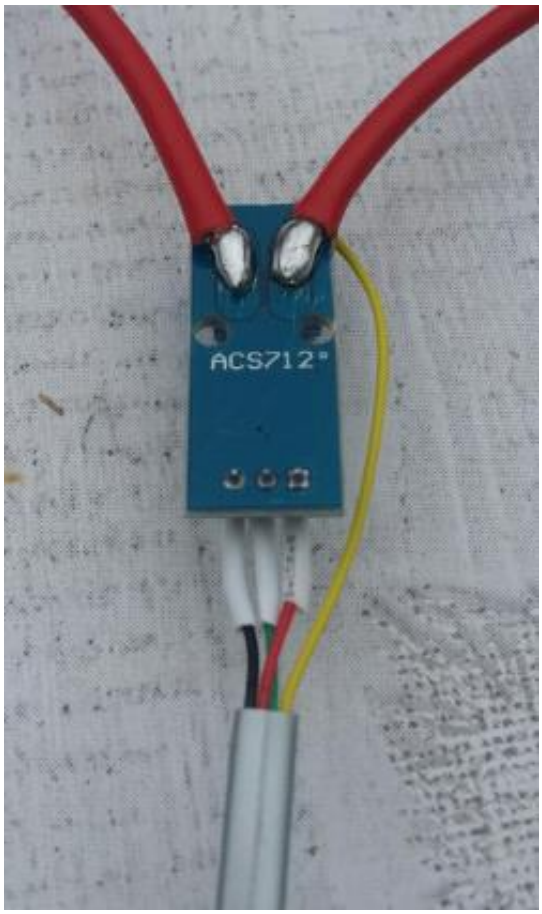
Mise à jour du programme final

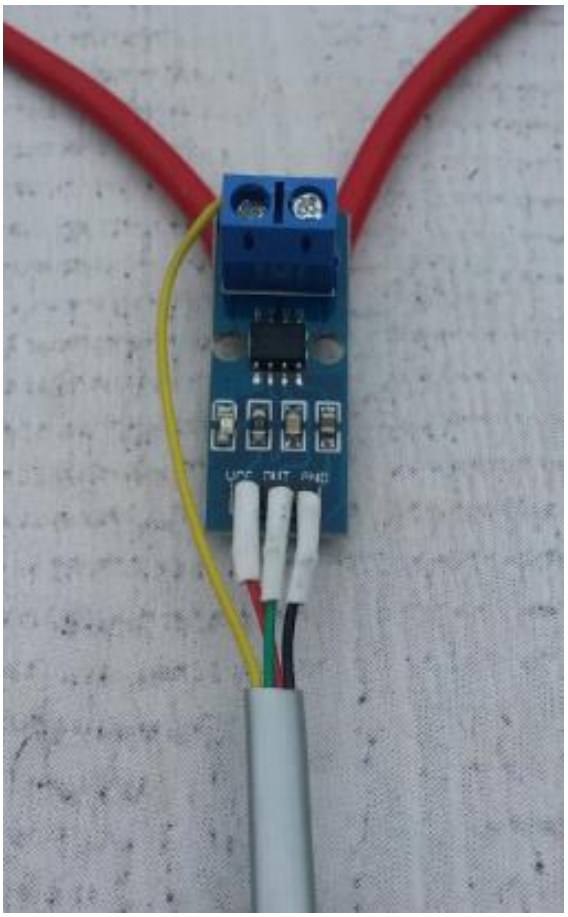
```
/*

Banc de mesures pour éoliennes 24 ou 48V sur batteries, avec
enregistrement sur carte SD
_____________________________________________________________________
|                                                                    |
|       auteur : Philippe de Craene <dcphilippe@yahoo.fr             |
|           pour l' Association P'TIWATT                             |
_____________________________________________________________________

Liste des mesures effectuées :
• Date et l'heure,
• Vitesse de rotation :      Fprimaire => entrée 3
• Tension au pont de diode : Vprimaire => entrée A0
• Courant au pont de diode : Iprimaire => entrée A3
• Tension de charge :        Vsortie   => entére A2
• Courant de charge :        Isortie   => entrée A1

*/

#include <Wire.h>
#include <LiquidCrystal_I2C.h> // https://app.box.com/s/czde88f5b9vpulhf8z56
// Attention ! Renomer la librairie d'origine de l'IDE LiquidCrystal.h et LiquidCrystal.cpp
#include <SD.h>                // https://github.com/adafruit/SD
// Attention ! Renomer la librairie d'origine de l'IDE SD.h et SD.cpp
#include <RTClib.h>            // https://github.com/adafruit/RTClib/archive/master.zip
#include <SPI.h>

File dataFile;                 // Création de l'instance pour gérer le fichier sur SD

// Brochage des entrées des capteurs et paramétrage

const byte broche_Fpri = 3;    // broche numérique calcul fréquence
const byte broche_Vpri = A0;   // broche lecture tension pont de diodes
const byte broche_Ipri = A3;   // broche lecture courant pont de diodes
const byte broche_Vsor = A2;   // broche lecture tension de charge de la batterie
const byte broche_Isor = A1;   // broche lecture courant de charge de la batterie

// On the Ethernet Shield, CS is pin 4. Note that even if it's not used as the CS pin, the
// hardware CS pin must be left as an output or the SD library functions will not work.
const byte chipSelect  = 4;

// choisir le modèle d'éolienne 24V ou 48V à vérifier expérimentalement :
// éolienne 24V avec maxi à 54V => conv_Vpri = 20.2
// éolienne 48V avec maxi à 95V => conv Vpri = 11.2
const float conv_V = 20.2;  // conversion bytes/tension pour éolienne 24V

// Choisir le modèle de sonde ACS712 :
// Module ACS712 de 5A => 0.185V/A
// Module ACS712 de 20A => 0.1V/A
// Module ACS712 de 30A => 0.066mV/A
const float conv_I = 0.1;   //  pour modèle 20A

// Choisir le modèle de RTC sur le module carte SD :
RTC_DS1307 rtc;
//RTC_PCF8523 rtc;

// Variables de traitement
float Vpri;                 // Tension au pont de diode
float Ipri;                 // Courant délivré au pont de diode
float Fpri;                 // Vitesse de rotation
float Vsor;                 // Tension de charge de la batterie
float Isor;                 // Courant de charge de la batterie
unsigned long duree_H;      // durée de la demie période état haut
unsigned long duree_L;      // durée de la demie période état bas
int B_Vpri, B_Ipri, B_Vsor, B_Isor;         // lecture des capteurs en bytes
byte prev_second = 0;       // pour le compteur

char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday"};

// Déclaration du LCD en mode I2C :
// toutes les infos ici : http://arduino-info.wikispaces.com/LCD-Blue-I2C
```

```
// Set the pins on the I2C chip used for LCD connections:
//                    addr, en,rw,rs,d4,d5,d6,d7,bl,blpol
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
// => connexion des 2 broches I2C sur l'Arduino Uno R3 : SDA en A4, SCL en A5

//
// SETUP
//_____________________________________________________________________________________________

void setup() {

  pinMode(broche_Fpri, INPUT);

// initialisation de l'affichage et du mode console
  Serial.begin(9600);            // préparation du moniteur série
  Serial.println ();
  if(! rtc.begin()) { Serial.println("RTC introuvable");}
  if(! rtc.isrunning()) { Serial.println("Date et heure à programmer sur le RTC");}
  Serial.print("Initializing SD card...");
// make sure that the default chip select pin is set to output, even if you don't use it:
  pinMode(SS, OUTPUT);
// see if the card is present and can be initialized:
  if (!SD.begin(chipSelect)) {  Serial.println("Card failed, or not present"); }
  else { Serial.println("card initialized."); }
// Open up the file we're going to log to!
  dataFile = SD.open("datalog.txt", FILE_WRITE);
  if (! dataFile) { Serial.println("erreur sur fichier datalog.txt"); }
  Serial.println("ready ...");
  Serial.println();

// initialisation du LCD
  lcd.begin(16,2);                // initialize the lcd for 16 chars 2 lines
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("*** BONJOUR ***");
  lcd.setCursor(0, 1);
  lcd.print("    *******");
  delay(1500);

  Serial.println("  Date   | Vpri | Ipri | Vsor | Isor | Tours/s ");

}     // fin de setup

//
// LOOP
//_____________________________________________________________________________________________

void loop() {

// Calcul de Vprimaire
  B_Vpri = analogRead( broche_Vpri );
  Vpri = B_Vpri / conv_V;

// Calcul de Iprimaire    // valeur absolue pour s'affranchir du sens de branchemet de la sonde
  B_Ipri = analogRead( broche_Ipri );
  Ipri = abs(( B_Ipri * 5.0 / 1023.0 ) - 2.5 ) / conv_I;

// Calcul de Vsortie
  B_Vsor = analogRead( broche_Vsor );
  Vsor = B_Vsor / conv_V;

// Calcul de Isortie
  B_Isor = analogRead( broche_Isor );
  Isor = abs(( B_Isor * 5.0 / 1023.0 ) - 2.5 ) / conv_I;

// Calcul de la vitesse de rotation Fpri
  duree_H = pulseIn(broche_Fpri, HIGH);
  duree_L = pulseIn(broche_Fpri, LOW);
  if( duree_H + duree_L > 1 ) { Fpri = 1000000.0/(duree_H + duree_L);}
  else { Fpri = 0;}

  DateTime now = rtc.now();        // récupération de la date et l'heure
  if( now.second() != prev_second ) {    // toutes les secondes

// Affichage du résultat sur le LCD

    lcd.setCursor(0, 0);
    lcd.print(String(Vpri,1));
    lcd.print("V ");
    if( Vpri < 10) { lcd.print(" ");}
    lcd.print("Pe=");
    if( Vpri*Ipri < 100) { lcd.print(" ");}
    if( Vpri*Ipri < 10) { lcd.print(" ");}
    lcd.print(String(Vpri*Ipri,1));
    lcd.print("W ");
    lcd.setCursor(0, 1);
    lcd.print(String(Fpri,0));
```

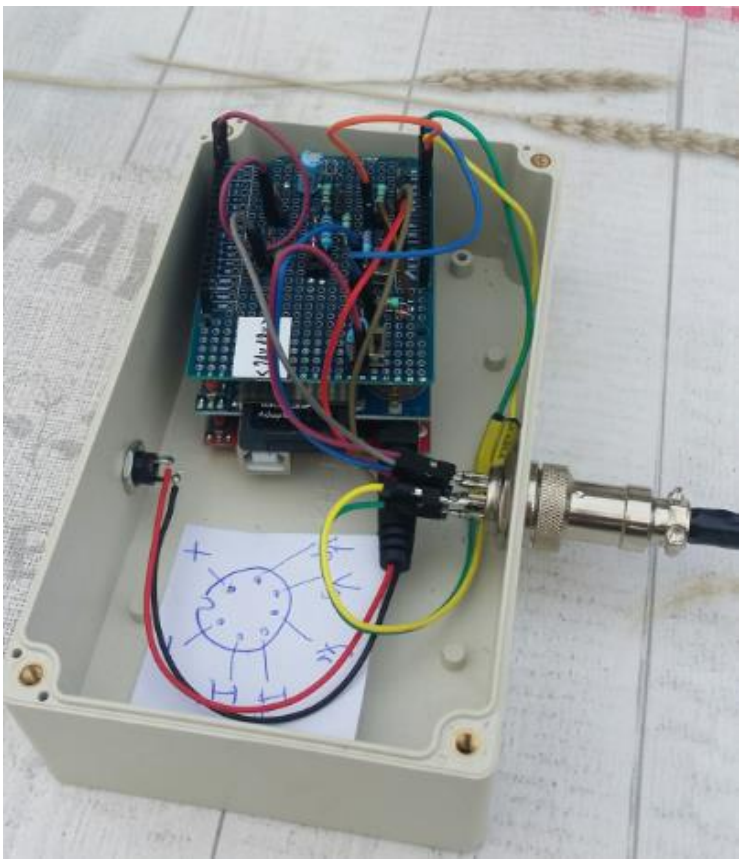
```
    lcd.print("Hz ");
    lcd.print(" Ps=");
    if( Vsor*Isor < 100) { lcd.print(" ");}
    if( Vsor*Isor < 10) { lcd.print(" ");}
    lcd.print(String(Vsor*Isor,1));
    lcd.print("W ");

// Affichage du résultat sur carte SD

    if( Fpri > 0 ) {                  // dès que l'éolienne tourne
      String dataString = "";         // initialisation d'une chaine de caractères
      dataString += String(daysOfTheWeek[now.dayOfTheWeek()]);
      dataString += ";";
      dataString += String(now.day(), DEC);
      dataString += ";";
      dataString += String(now.month(), DEC);
      dataString += ";";
      dataString += String(now.year(), DEC);
      dataString += ";";
      dataString += String(now.hour(), DEC);
      dataString += ";";
      dataString += String(now.minute(), DEC);
      dataString += ";";
      dataString += String(now.second(), DEC);
      dataString += ";";
      dataString += ";";
      dataString += String(Vpri);
      dataString += ";";
      dataString += String(Ipri);
      dataString += ";";
      dataString += String(Vsor);
      dataString += ";";
      dataString += String(Isor);
      dataString += ";";
      dataString += String(Fpri);
      dataFile.println(dataString);     // enregistre la ligne
      dataFile.flush();                 // l'enregistre sur SD
      Serial.println(dataString);       // et l'affiche à la console
    }      // fin test Fpri
    prev_second = now.second();
  }        // fin compteur de seconde
}          // fin de loop()
```

Quelques photos





Mode de branchement des sondes de courant. La soudure est préférée à ce minuscule bornier.
Le fil jaune recueille la tension.



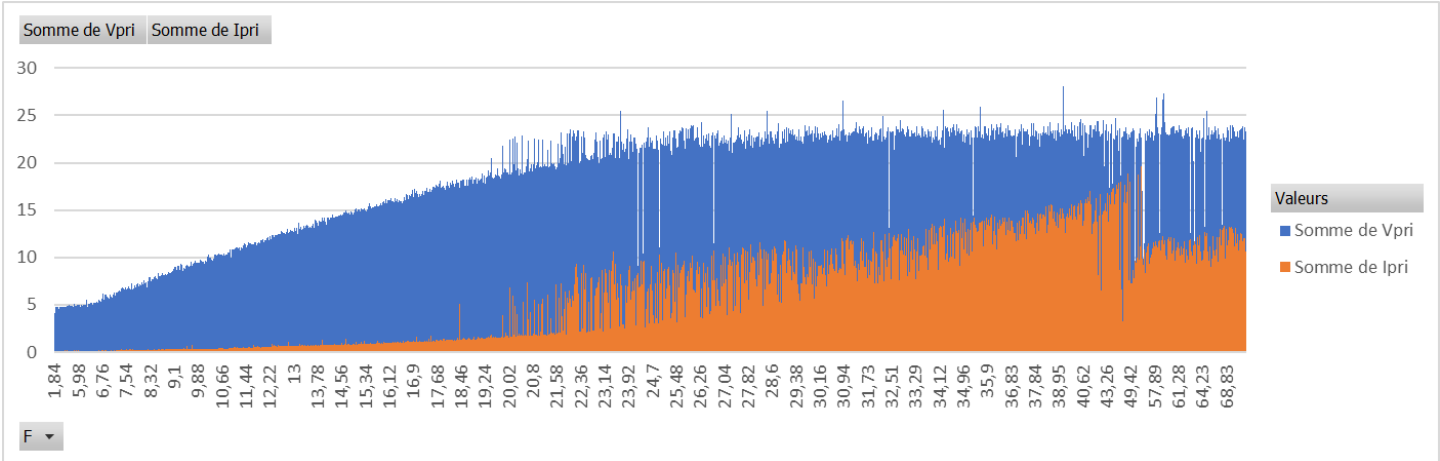
3 cartes sont superposées : en bas l'Arduino Uno, puis le lecteur de carte SD, puis le module auto construit de gestion des sondes.

Analyse des résultats

Remarque : la mesure de la fréquence de rotation est perturbée par la fréquence de découpage du régulateur MPPT qui suit le capteur. Un condensateur de 1 à 4,7nF placée entre l'entrée numérique 3 de l'Arduino et la masse (c'est-à-dire en parallèle à D2) résout le problème.

En 24 heures de tests, nous avons un fichier de 50 000 lignes, d'une taille de moins de 3Mo.

Tension et courant généré en fonction de la vitesse de rotation de l'hélice

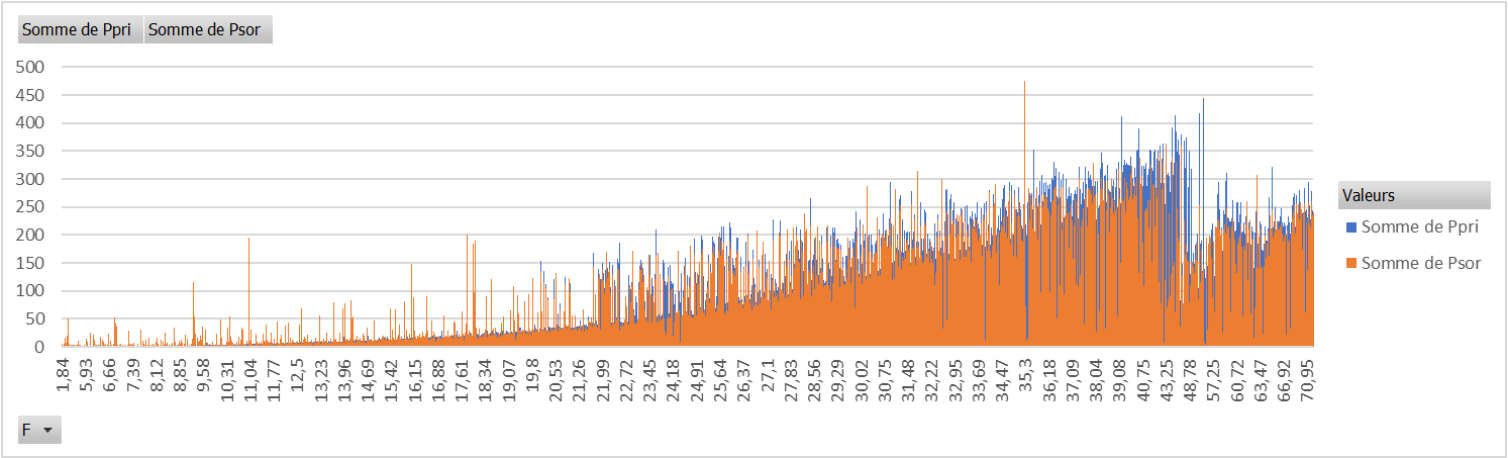


A noter :

Visiblement le régulateur est de type Buck (abaisseur de tension) car :

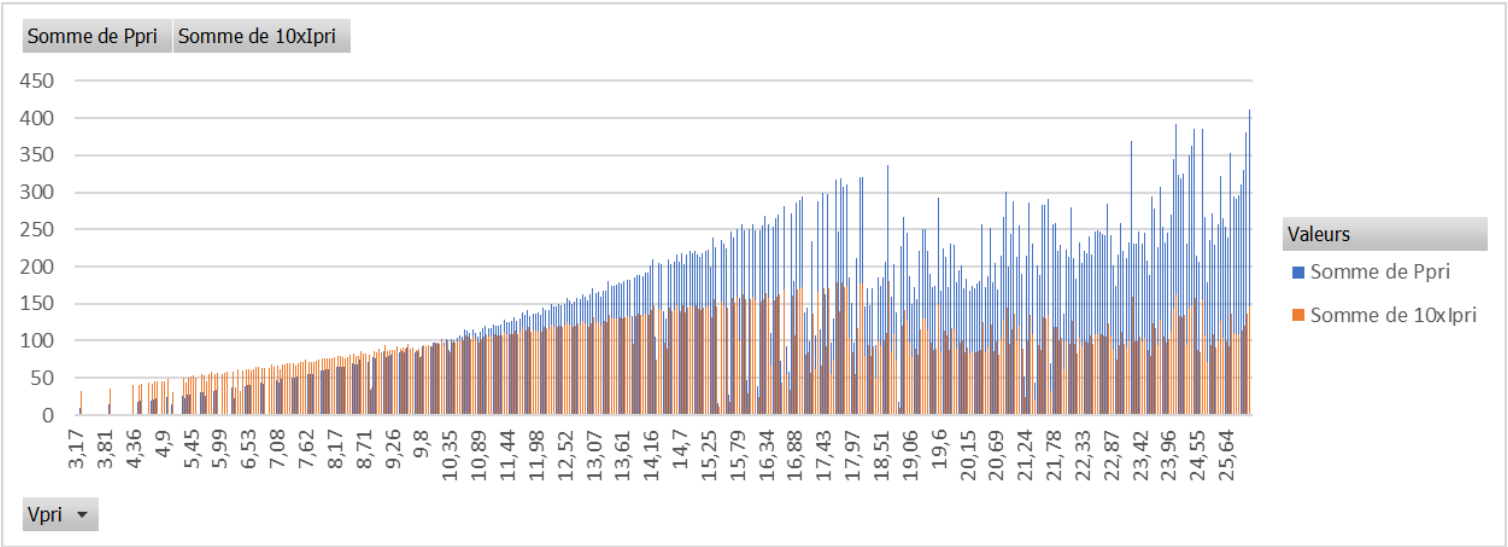
- La tension ne croit pas au-delà de 24V
- Le courant est quasi nul tant que la tension au pont de diodes n'atteint pas la tension de la batterie.

Puissance générée (Ppri) et fournie à la batterie (Psor) en fonction de la vitesse de rotation de l'hélice



Psor tient compte de la puissance fournie par la batterie pour alimenter l'injecteur. Ce qui explique que Psor puisse dépasser Ppri.

Courant et puissance générée en fonction de la tension



Etonnement le courant – et donc la puissance - chute d'un tier de sa valeur dès que la tension dépasse 18V.