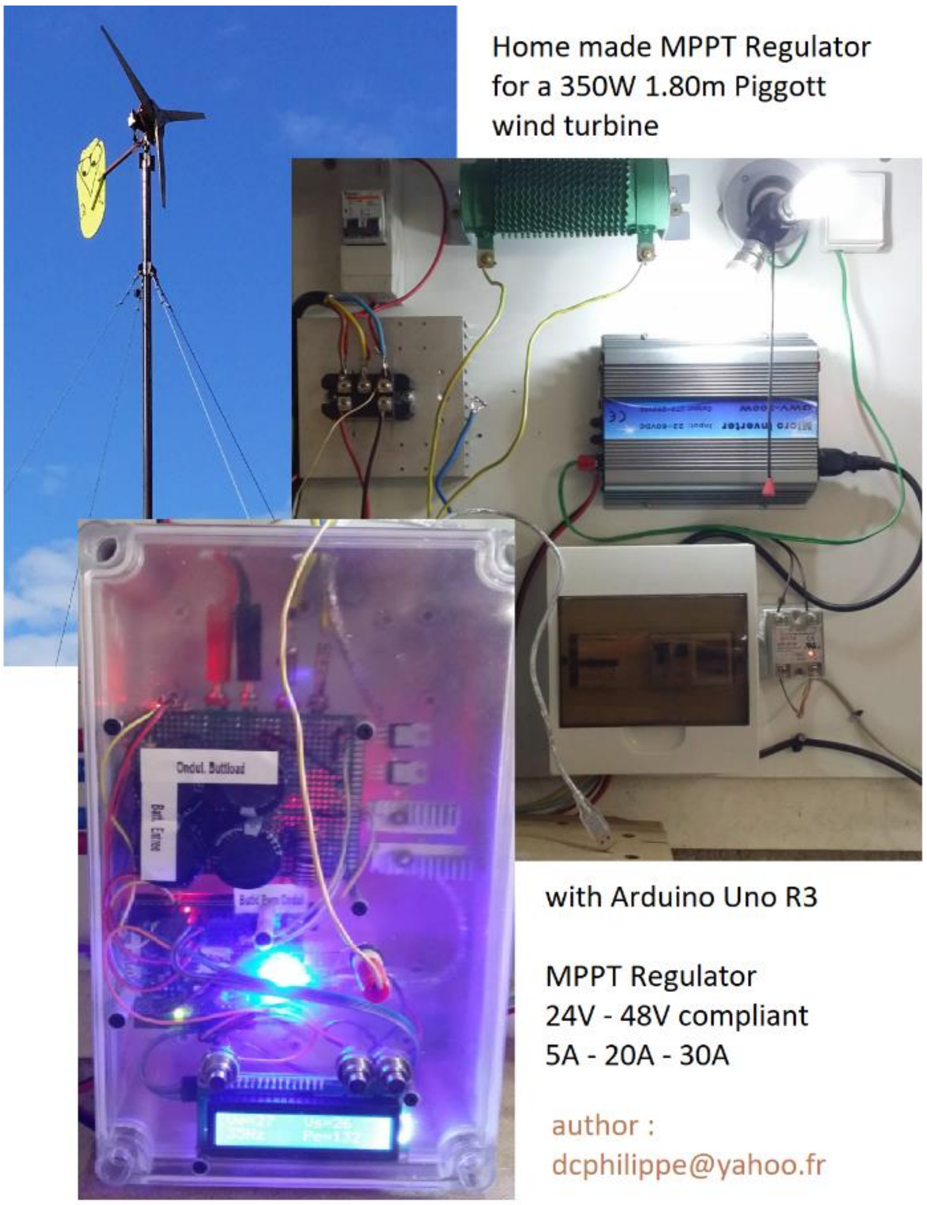
How to build a Wind Turbine MPPT Regulator within direct injection or battery configuration

Author : Philippe de Craene
dcphilippe@yahoo.fr

Date: Oct. 2018 – Oct. 2019

Manual version : 2.0

Program version : 2.0



For a wind turbine, a regulator has two goals:

- To protect the wind turbine against over speed of the turbine that may destroy it,
- To adapt the power delivered to charge a battery or to drive an inverter.

There are many regulators on the market. However, they are mostly adapted for solar panels only, and if the curve of delivered power is similar, the way to regulate is different – to resume solar panels use buck converter, wind turbines use boost converter. Many are not MPPT, the PWM regulators are very less efficient than MPPT, and also specific wind turbine MPPT regulators are very expensive.

Therefore, It can be a very good project to self-build our own regulator, after having been built our own Piggott wind Turbine ☺

This manual describes how to build such a regulator step by step.

Table of content

Introduction

Depending of the geography, there will be more or less wind. However, a wind turbine will sometime turn right, sometime will turn too much, and sometime will not turn at all. Moreover, the amount of wind can change very quickly, nearly instantaneously, in fact much quicker than the cloud that will shadow the solar panel.

Therefore, we imagine a regulator that will offer:

- A very high efficiency: it must be of the MPPT technology, which offer the advantage to “find” the best working source voltage to get the best power delivery.
- A very fast tracking state: due to the erratic behavior of the wind, the MPPT will be a real advantage if the best working voltage can be reach as fast as possible.
- A very good safe for the wind turbine: if it is too windy, or if there is no load, the turbine will turn freely, so it will turn too fast until damage. That is the reason why a wind turbine regulator is built with a Dump Load Resistor; this load will break the speed of rotation – if needed of course.

Also the regulator:

- Can operate without knowing any of the wind turbine built specifications, except nominal voltage and power.
- Can operate with wind turbine of 24V or 48V,
- Can operate until a current of 20A. However, it is possible to easily reach 30A max. For more another sensor must be adapted.
- Can drive the charging batteries, or can be directly connected to a 230V inverter, or any load of proper consumption

This version 2 contains several useful add-ons and a mistake:

- Now there is a PCB !
- Unfortunately, drawing the sheet for the PCB there was 2 confusions between Arduino Uno pinup and the Atmega 328p’s. For those who has built the shield for V1.x must perform 2 changes to make it operate in V2.x :
    - Yellow LED was pin 8 and is now pin 6
    - Red LED was pin 9 and is now pin 13
- Better default values are defined for the charge of batteries
- A optional SSR has been added to connect a battery charger in case of highly discharged and absolutely no wind…. As some SSR switch on with a ON signal, others with a OFF’s, both output are proposed.
- Several capacitors have been added to improve filtration
- LiquidChrystal library replacement by one that allow display switch off.

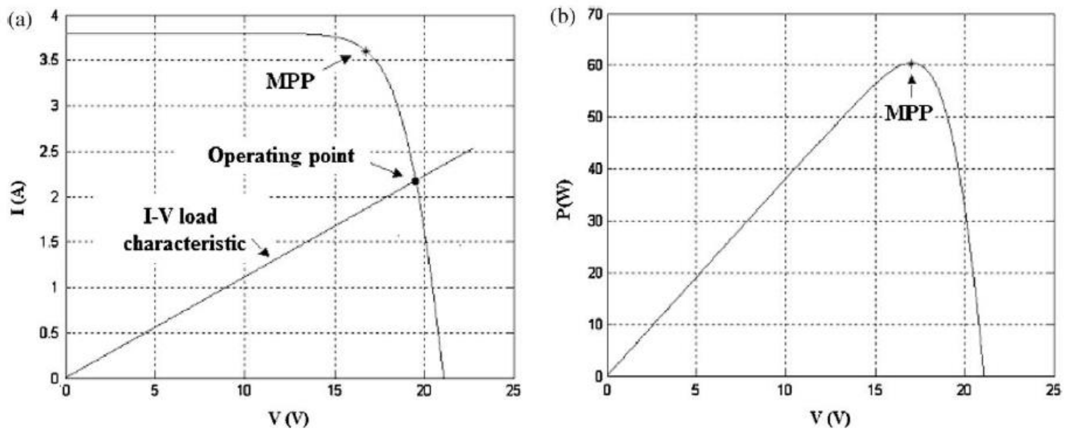
A little theory of MPPT

MPPT explanation

The power response of a solar panel or a wind turbine is not linear. If the voltage increases proportionally with the amount of sun or wind, then the power grows proportionally with the amount of sun or wind. However, not indefinitely, instead until a maximum value, then power decreases even if the voltage stays growing.

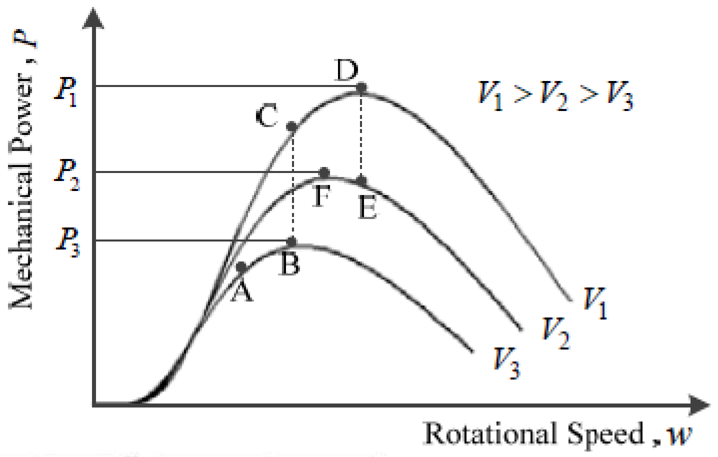
This maximum value is called MPP – Maximum Power Point.

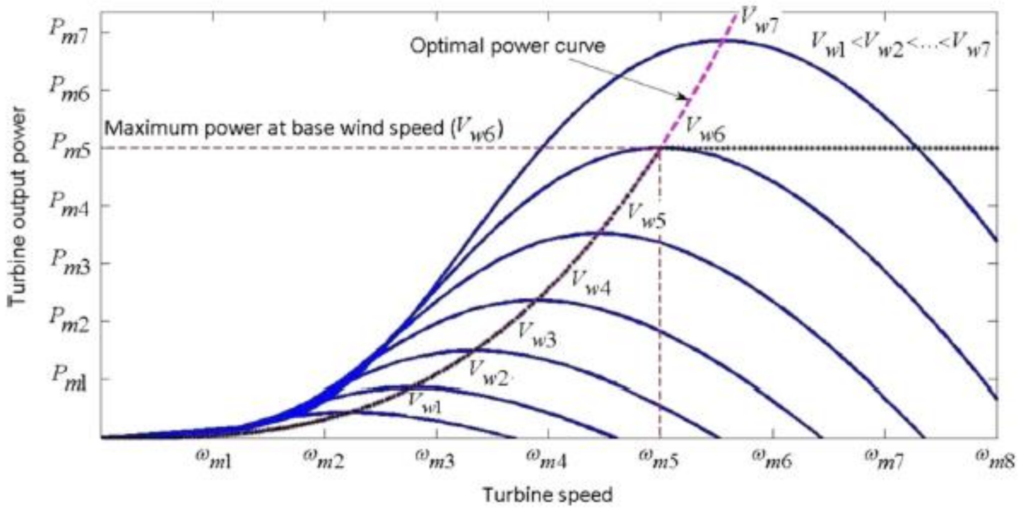
Power response of a solar panel or a wind turbine :



Power response for three rotation speeds V1 V2 and V3.

We can notice that each time the MPP takes place at a different place. That shows how the power generator is non-linear, as expressed by the graph below:





The quality of the MPPT – Maximum Power Point Tracking – algorithm will be placed in:

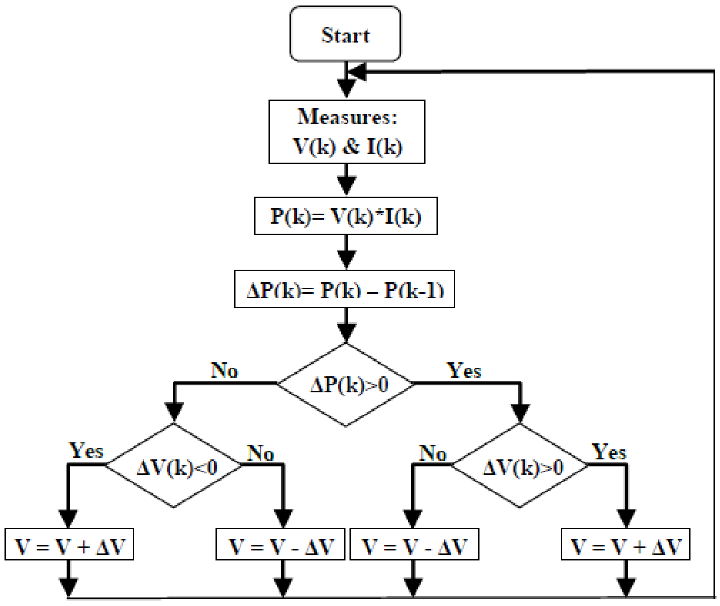
- The success to “follow” the MPP whatever the situation of the speed turbine.
- The success to track the MPPT the fastest possible.

The fastest algorithms use hardware dependent data that dress a table of characteristics of what to do for any wind speed, but this kind of tables are specific for a model of turbine. Moreover, we can suppose that the characteristics will evolve with time…

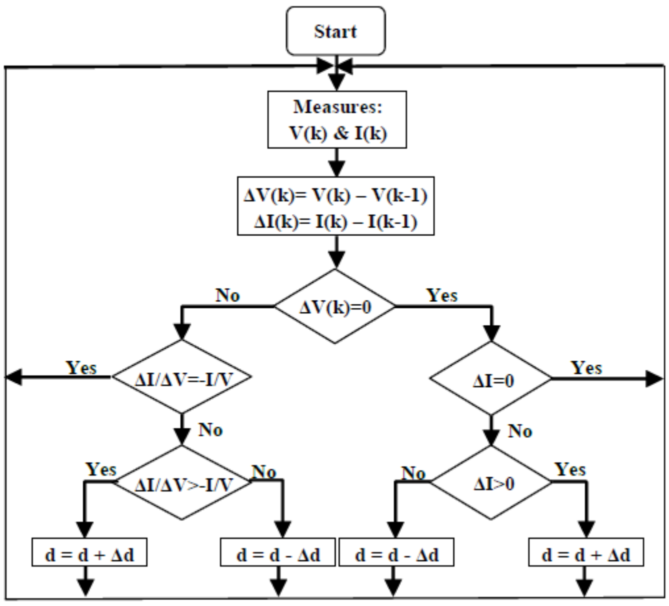
In our case, the parameters we can get are: delivered voltage, available current and turbine speed.

In the most common cases the MPP is calculated with the use of the PO algorithm – Perturb and Observe – or the INC algorithm – Incremental Conductance –

Algorithm PO



Algorithm INC



Both algorithms are based on a similar method: the increase or the decrease of the ratio that drive a DC-DC convertor, as a result of the comparison between actual measure and precedent's one. We can imagine that the succession of cycles or comparisons that makes the ration evaluates until the result is one time a little bit before the MPP, next time a little bit after; in fact, the method is called HCS for Hill Climbing Search.

As is, this operating method has three drawbacks:

➢ The MPPT spend its "time" to increase and decrease to climb over the "hill" – the maximum working state - , with no way to get it. This makes a little loss of available power. For this reason the "step" in the ratio change should be as small as possible, to be able to reach the most possible the "hill" point that will minimize the losses.

➢ The time spent to reach the MPP is another way of power losses. To increase the speed the number of cycles - of ratio changes to join the "hill"- should be the lowest possible. For this reason the "step" in the ratio change must be the highest possible.
We can see that the "step" value is a compromise between losses in:
- the difference between the maximum to obtain and the 2 working values on both sides of the maximum, the smaller the "step" the better it will be,
- the speed to reach this maximum area, the biggest the "step" the best it will be.

The solution will be to adopt a variable "step".

➢ In case of sudden drop of wind, the very low power available may confuse the MPPT : in this case the ratio will reach the maximum and then go down to the right value

The solution will be to use high accurate voltage and current sensors.

Specifications and rules

To determine the MPP there are three variables available from any wind turbine:

• Generated voltage after rectifier :	Vpri	
• Available current :	Ipri	=> Vpri*Ipri = Puiss
• Turbine speed :	Fpri	

1- The MPPT will start when Vpri > VprimMin which means that the turbine will not have any load under this voltage to facilitate the speed increase

2- The Dump Load Resistor will be connect as soon as Vpri > VpriMax. In fact, for such a voltage the MPPT is yet stopped by its own fact.
3- The MPPT will stop if Ipri > IpriMax to prevent overcharge or short-circuit.
4- The MPPT algorithm will state as follow :
    - Case Vpri increase :
        - If Puiss decrease we can increase the load by increasing the current : Step is positive to increase pwm
        - If Puiss increase we must do the opposite in respect of the HCS operation : Step is negative to decrease pwm
    - Case Vpri decrease : the opposite actions are taken :
        - If Puiss decrease then Step is negative to decrease pwm so Vpri may increase
        - If Puiss increase Step is positive to increase pwm
5- Fpri can be used to determine the value of Step: the higher the rotation speed changes, the higher will be Step.
6- If Vpri < VpriMin a brutal fall down is done on Step to unload the turbine and trying so to increase Vpri. If not the MPPT will finish to stop: with no load the turbine has much more facilities to increase speed, and voltage as well.

As a result the output of the regulator will give:

• Output voltage:	Vsor
• Charging battery current:	Ibat

Additional specific operations are performed:

7- The MPPT will stop if Vsor > VsorMax to prevent overcharge battery or inverter
8- If Vsor < VsorMin the Step is forced positive – or null – in order to increase voltage, the inverter is disconnected
9- If Ibat > IbatMax the Step is forced negative – or null - in order to decrease pwm
10- If Ibat > IbatMax and MPPT stopped then the Dump Load Resistor is connected
11- If Vsor > VsorFlo the inverter is connected

We assume these parameters:

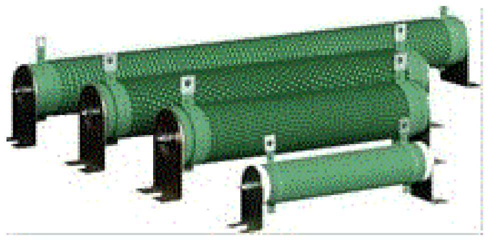
- VpriMin : minimal input voltage to make the regulator works
- VpriMax : maximal input voltage before risk of damage on the turbine

- VsorMin : minimal output voltage for a discharged battery, or under voltage for the inverter
- VsorFlo : floating output voltage for the battery, or starting working voltage for the inverter
- VsorMax : maximal output voltage for battery or the inverter
- IbatMax : maximal charging current, usually 0.23 the capacity of the battery : for instance IbatMax # 13A for a 54A/H battery.

List of materials

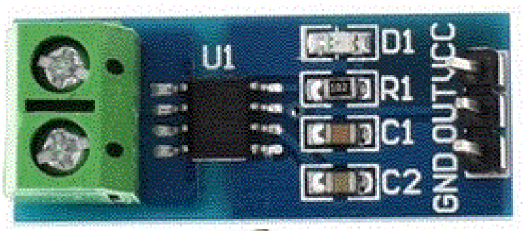
1- One 3-phases power rectifier done for 150A is fine

2- One Dump Load Resistor that can absorb the total power of the wind turbine

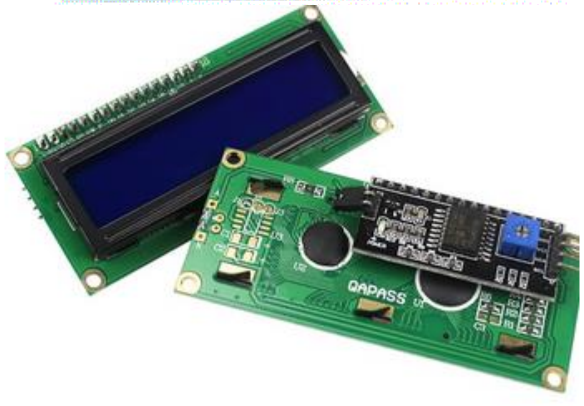


3- One Arduino Uno R3

4- Two current sensors ACS712 module; there are 3 versions: 5A, 20A et 30A. For our purpose, we chose the 20A model.



5- One LCD display 1602 with I2C interface



6- At least (*) four power transistor CMOS, for instance:
   - IRFB4110 that works until 100V, 180A max current and 370W of dissipation. Very low Rds.
   - IRFP4227 can work until 200V so it will be preferred for big 48V wind turbine. However, the Rds is much higher so 2 or 3 of them in parallel and the use of a (small) radiator will be great.
   - IRFB4321 can work until 150V, 83A max current and 330W of dissipation.

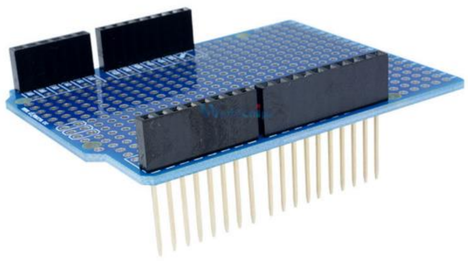
   (*) at least because it can be a very good idea to parallelize them in case of heat.

7- At least (*) one Schottky diode that can stand 150V (for a 48V wind turbine) and twice the max current. For a 350W/24V wind turbine the MBR30200 is great. A (small) radiator is required.
   (*) at least because it can be a very good idea to parallelize them in case of heat. In fact, the TO220 ship contains 2 diodes.

8- One self of 100uH at least 15A like this one :
   https://fr.aliexpress.com/item/32804332271.html?spm=a2g0s.9042311.0.0.41826c37UDsQky

9- At least (*) four electrolytic capacitors of 4700uF / 80V.
   (*) at least because 4700uF/80V is very less expensive than 10000uF/80V or 4700uF/200V. Then we will combine capacitors in series and/or parallel to get the biggest capacity possible (10000uF or more) at the right voltage rate.

10- Two power CMOS driver circuits TC428.
   Power CMOS need voltage and current on the gate that the Arduino cannot offer.



11- Two transistors like 2N2222, few 4.7V zeners, few electrolytic capacitors, few resistors….

12- One prototype shield sized for Arduino Uno R3 or the ready to use PCB

13- One 12V power supply. It can be either a 230V/12V or a 24V/12V or 48V/12V for those who works with a battery.

14- For a realization with the PCB : a 12MHz crystal with 2x 22pF capacitors, and a Atmega328p, 1 DIP28 adapter.

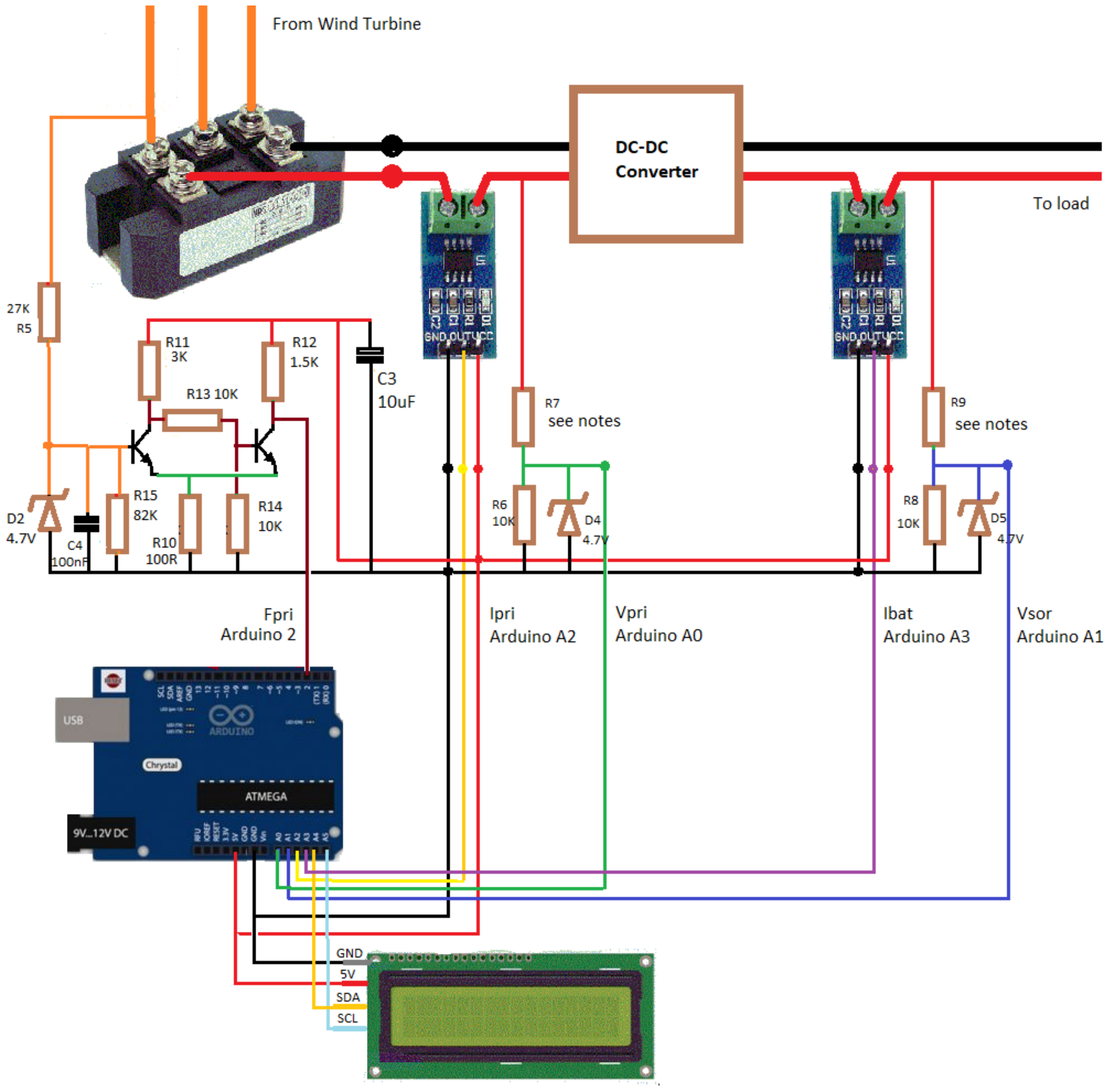
Circuit around sensors

List of sensors

We will need 5 sensors to measure:

• Vpri	range from 0 to twice the nominal wind turbine voltage : ▪ 24V : 0 – 50V divided to comply with the analog read of the Arduino, or ▪ 48V : 0 – 100V divided to comply with the analog read of the Arduino.
• Ipri	defined by the ACS72 circuit module: 20A in our case.
• Fpri	we can get the frequency above the rectifier, picking up from one of the 3 coils of the turbine. The result signal must match a digital of the Arduino by a 0-5V square.
• Vsor	range from 0 to twice the maximum Vpri voltage, because the DC-DC converter is a Boost so Then output voltage is supposed to be able to reach twice the input voltage: ▪ 24V : 0 - 100V divided to comply with the analog read of the Arduino, or ▪ 48V : 0 – 200V divided to comply with the analog read of the Arduino.
• Ibat	defined by the ACS72 circuit module: 20A in our case.

Circuit diagram around sensors



Current sensors for Ipri and Ibat

Each sensor is built with a ACS712 current sensor module for Arduino. The voltage available at their output varies from 0 to 5V, according to the current from –Imax to +Imax. So 0A gives 2.5V, and the current measured will give a variation of the output either over 2.5V, or under 2.5V, depending of the way of wiring. The conversion ratio I/U is:

- ACS712 – 5A max module => 185mV/A
- ACS712 – 20A max module => 100mV/A
- ACS712 – 30A max module => 66mV/A

Voltage sensors Vpri and Vsor

These sensors are simple voltage dividers built with R6 and R7 for Vpri, R8 and R9 for Vsor, according the formula:

VA0 = Vpri * R6 / (R6+R7)

with R6 = 10K :

- 24V turbine : Vpri maxi = 50V => R7 # 91K
- 48V turbine : Vpri maxi = 100V => R7 # 180K

VA1 = Vsor * R8 / (R8+R9)

with R6 = 10K :

- 24V turbine : Vpri maxi = 100V => R9 # 180K
- 48V turbine : Vpri maxi = 200V => R9 # 360K

D4 and D5 are 4.7V zeners that protect the Arduino entries for any surge.

R7 can be 91K + 91K in series with a switch that can shunt one of these resistors; likewise, R9 can be 180K + 180K. If so, the regulator will comply with any 24V or 48V turbine.

Vsor must be accurate. The life of the battery depends of this accuracy. However, the resistors used has a tolerance of 5%, so the resistor divider has at least 10% of tolerance. If I have 23.7V instead of an expected value of 24V, it is a problem. Further, we will set Vsor_calibration to correct this tolerance.

For 12V Win Turbine

Above calculations can be done for a 12V wind turbine, or for any voltages with these 3 rules:

$$VpriMaxRef = 2 * \ Win\ Turbine\ nominal\ voltage$$

$$Arduino\ maximum\ input\ voltage = 5V = VpriMaxRef\ .\ \frac{R6}{R6+R7}$$

$$Arduino\ maximum\ input\ voltage = 5V = VsorMax\,.\frac{R8}{R8+R9} = 2\,.VpriMaxRef..\frac{R8}{R8+R9}$$

Results given for a 12V Wind Turbine : R7 = 38K and R8 = 95K # 91K

Turbine speed sensor (optional)

The turbine speed sensor is built with R5, D2 and C4 that offer a positive half-square signal lower than 5V.

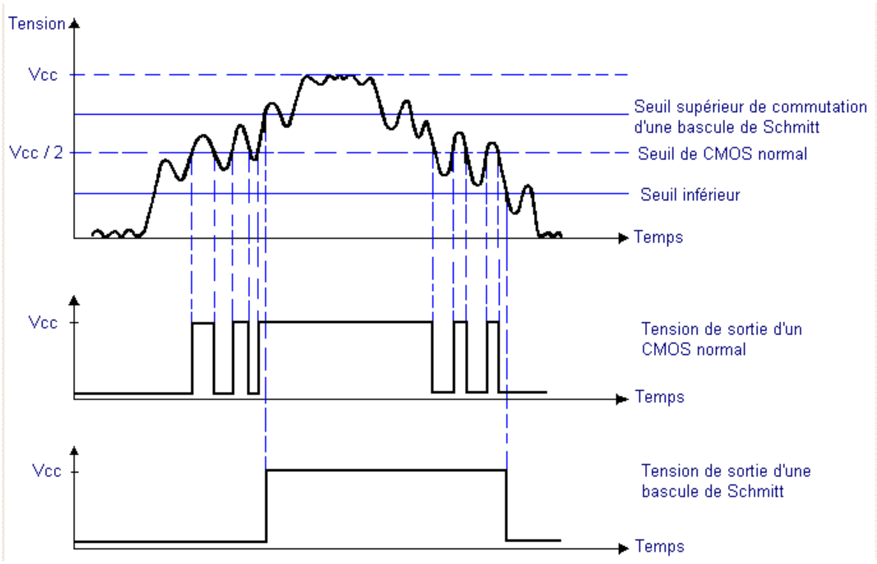
C4 and R5 form a low frequencies pass filter according to the formula : $fc = \frac{1}{2\pi R_5 C_4}$

With Fc = 50Hz (which is yet a high speed of turbine rotation), R5 = 27K, we get C# 100nF.

At the beginning of this project, the signal on D2 and C4 went directly to input 2 of the Arduino. However, for low wind – and very low frequencies too - the voltage is low with many accuracy maters in frequencies measures. Moreover, it is worth once the DC-DC converter in place, because it generates many electrical parasites.

So after first disappointing tests, a homemade flip-flop is added, built with R10 to R15 and 2 bipolar small signal transistors – like 2N2222 -

The following illustration explain the gain of a flip-flop :



Tension
Vcc
Vcc / 2
Seuil supérieur de commutation d'une bascule de Schmitt
Seuil de CMOS normal
Seuil inférieur
Temps
Vcc
Tension de sortie d'un CMOS normal
Temps
Vcc
Tension de sortie d'une bascule de Schmitt
Temps

The flip-flop is a Smith trigger with a small hysteresis to insure an anti-bounce final square signal.

About the way to measure the frequency, in the program we first utilized the instruction `pulseIn()`. But despite everything, there is a lake of accuracy/precision. Then the final program uses 0V interruptions and count the time between two interruptions.

This solution offers a much better accuracy.

The final program gives the choice to use or not the turbine speed sensor.

The LCD 1602 display with I2C extension

It is not required – just like the turbine speed sensor – but it is almost a good idea to easily see how the regulator is working.

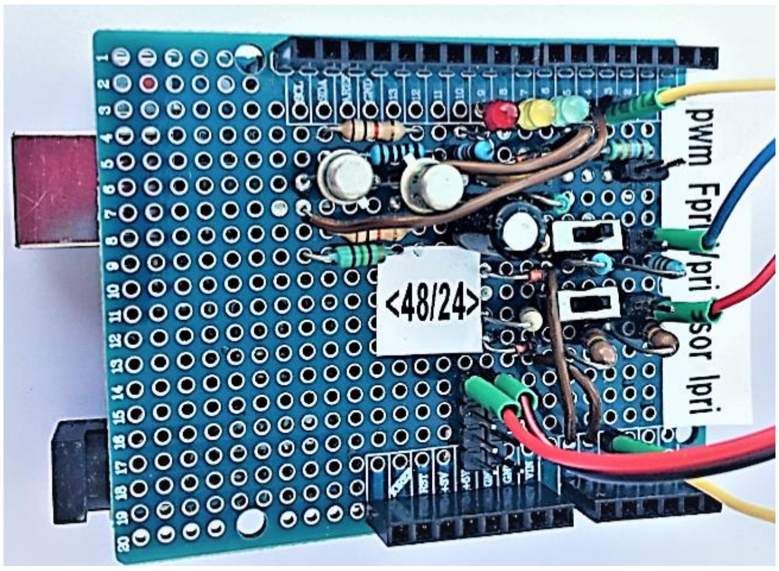
The 1602 LCD is very basic and very little expensive. As it requires many wires, we added a I2C extension which makes it driven with 2 wires only SDA and SCL, respectively A4 and A5 Arduino Uno R3 inputs – these inputs are hardware required -.

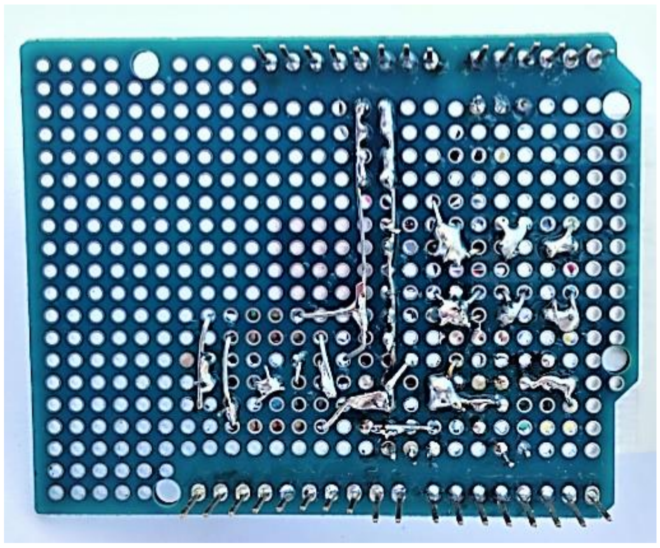
To be able to use the 1602 LCD I2C a specific library is needed, and must be installed on the Arduino IDE.

For information: https://andrologiciels.wordpress.com/arduino/lcd/lcd-1602-i2c/

To get the library: https://app.box.com/s/czde88f5b9vpulhf8z56

Some photos of the realization





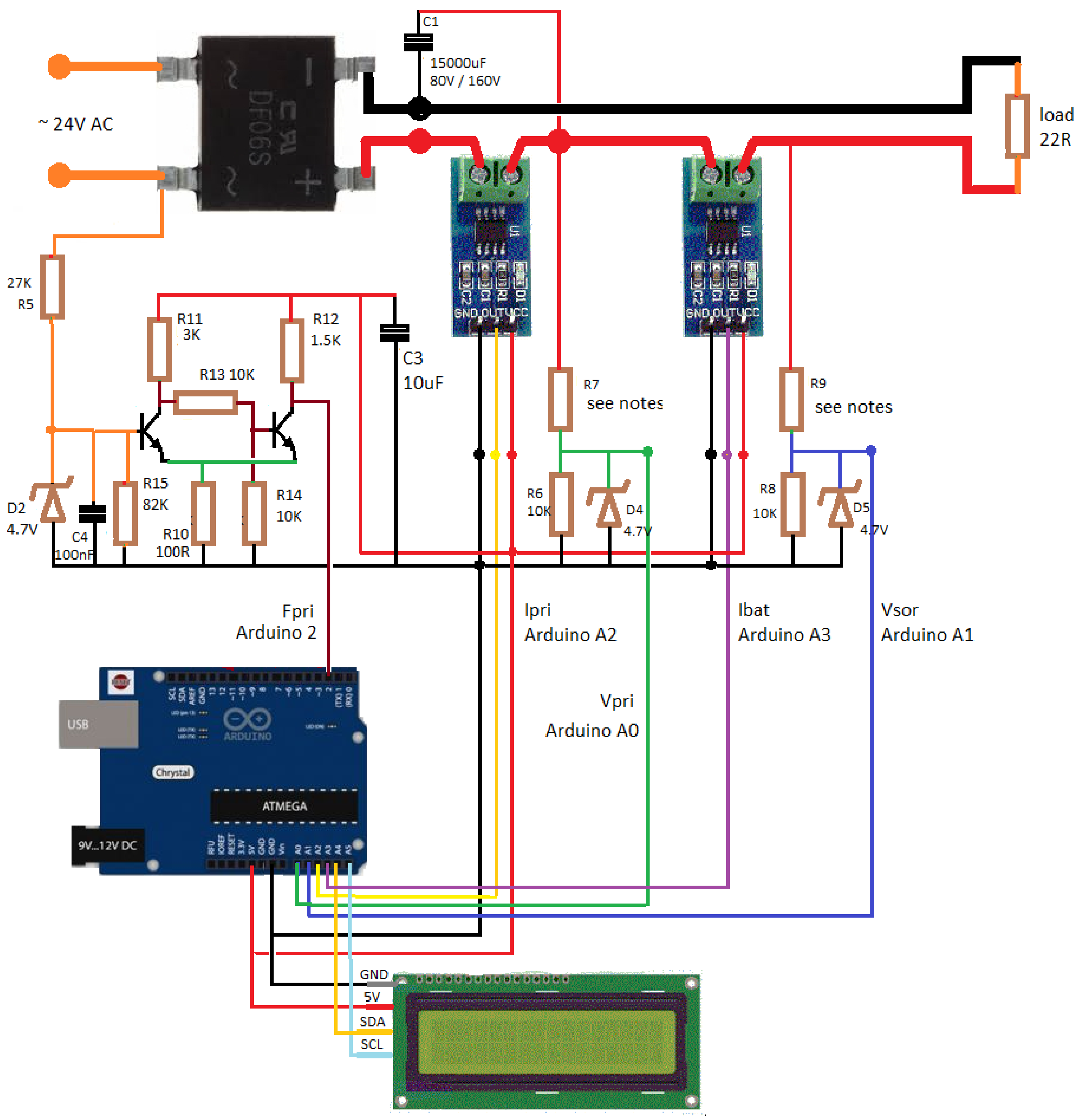
Note that this shield had undergone multiple modifications, thus the setting-up can be gratefully improved to limit electric wires. Keep enough place for the two TC428.

Tests and measures

All components will take place on the shield. A place must be let for future add-ons: 2 circuits TC428 and 4 LEDs.

I do not have photos of the realization, as my prototype is the result of many experiments and the components do not take place in the best final way…

To check if everything works fine we can test the following diagram:

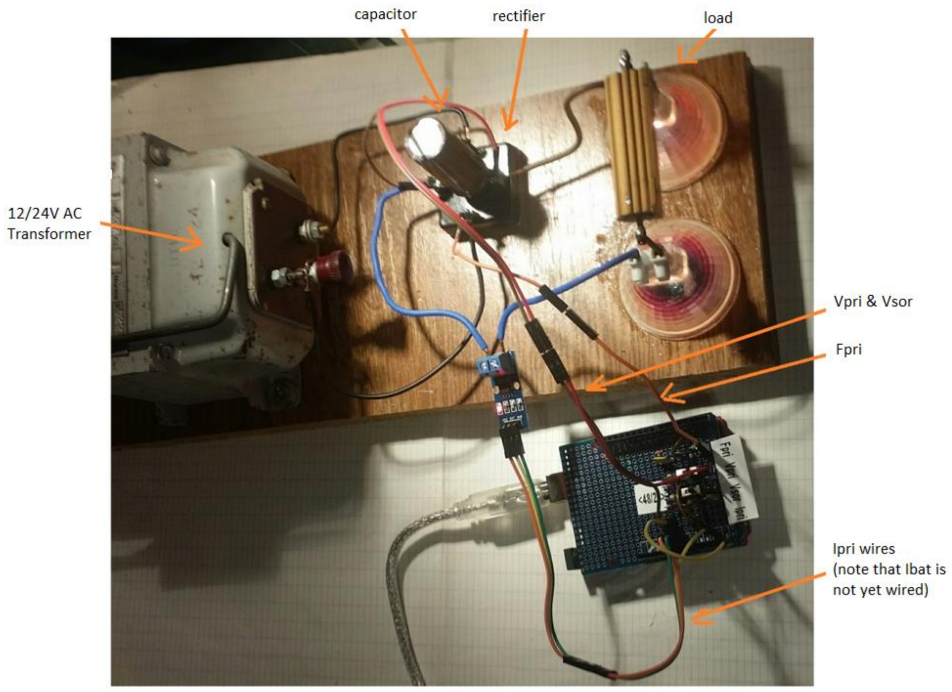


A simple 24V AC transformer with a rectifier and the capacitor C1 in input, output on a load of 22R 50W if allowed by the transformer power (at least 50VA), otherwise the lowest value possible belong the transformer's power.

Then the final program is loaded to test the circuit.

With such a configuration we should get the following measures:

Vpri = Vsor # 31.0V, Ipri = Ibat # 1,5A, Pe # 46W, Fpri = 50Hz



The DC-DC boost converter

The DC-DC converter

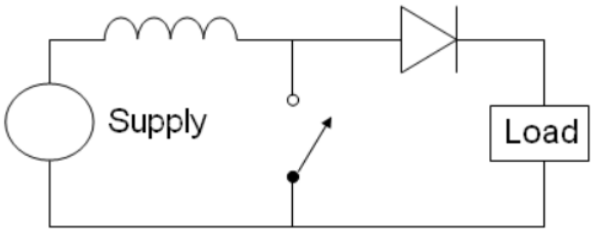
The MPPT Regulator must be able to control the current flow to the load by playing with the voltage available on the load, according to the law P = U * I.

Either the turbine will produce a voltage lower than the nominal value (24V or 48V), and the converter will have to develop to reach it, either the regulator will determine the correct tension, because the available current is emitted by the law I = U / load, the correct tension is set for the best P = U* I. For a power available from the turbine, the more is the voltage, the more is the current. Thus, the Converter will always boost voltage.

In fact, since the beginning of the project 4 different converters had been tested, in particular the buck-boost inverter converter, but the results gave that such a circuit was always working in boost operation mode, offered a poor efficiency of 78%, and the disadvantage to invert voltage polarity in output.

A very well description of the theory of a boost converter is given there: https://en.wikipedia.org/wiki/Boost_converter

According to the diagram



- the supply is the wind turbine followed by the 3 phases rectifier,
- the switch is a CMOS that cyclically short circuit the self, each time it is opened the power is available to the load through the diode.
- The load is the battery and/or the inverter

The self must allow the nominal working current, in our case 15A, and must not saturate for very high current surge.

The diode must have the lowest forward voltage possible, that is a loss of available power and generate heat. In addition, a fast answer, otherwise it is also a loss of available power and generate heat. So, we use a Shottky diode. The current should be twice the need, in our case 30A. Several diodes can be placed in parallel to divide the heat between them. The inconvenient of these diodes is the quite low inverse voltage possible before avalanche. It must be at least twice the maximum voltage, we have chosen 200V by security.

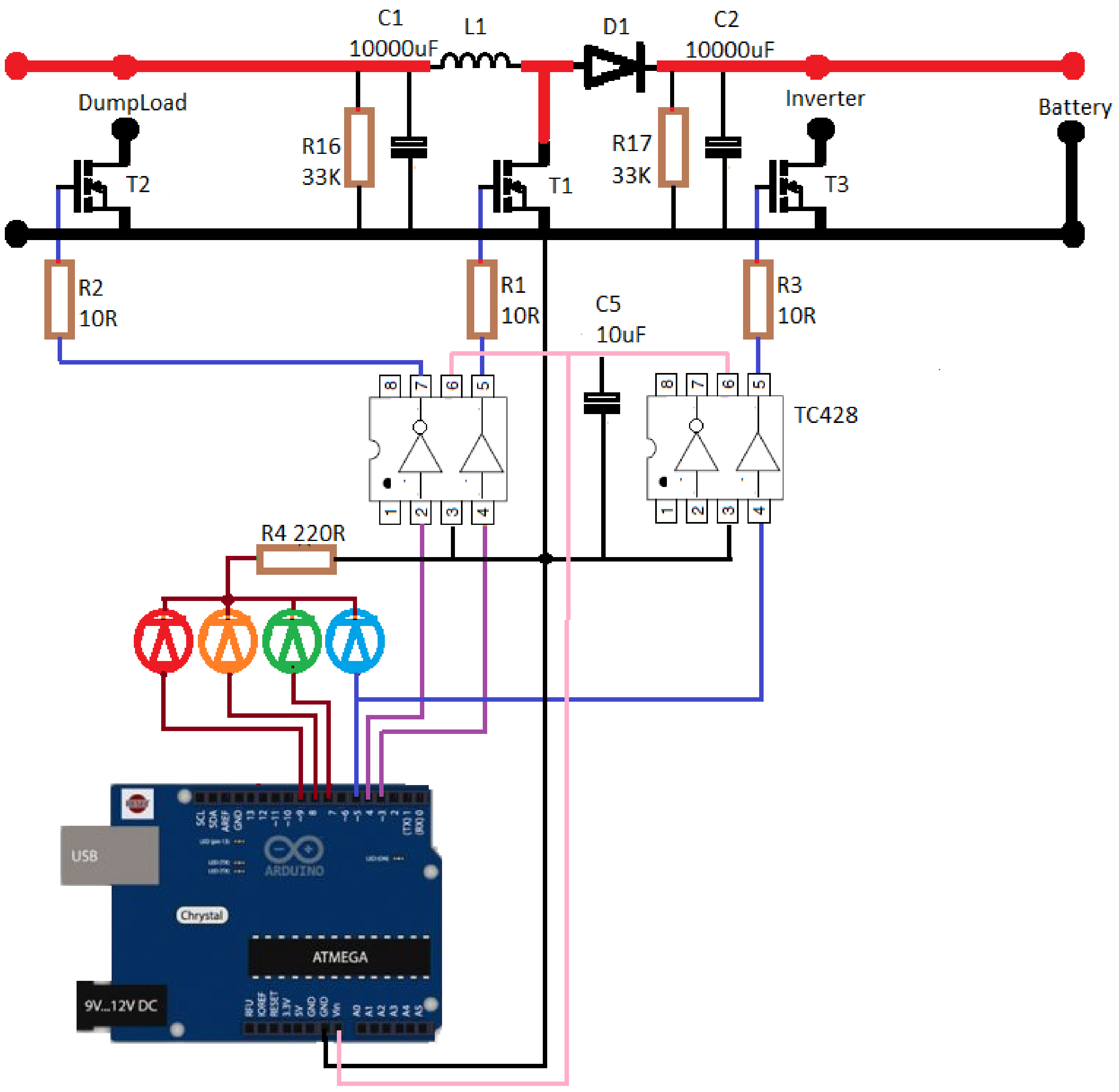
The transistor is a channel N CMOS that is fast, can stand very high surge current and at least twice the maximum voltage, The main loss of CMOS is due to the low residual resistance when it is in the switching position: Rds. The lower Rds is the lower heat will be to dissipate.

The efficiency of a boost converter is more than 90%. Our converter is more than 94%, only the Shottky diode need a small dissipation heat. Efficiency can be improved by replacing this diode by a CMOS for instance.

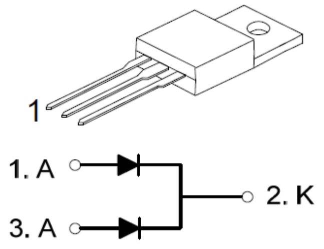
Extra "switches" are added to the converter:

- one in the input (before the self) to be able to switch the Dum Load Resistor for security
- one in the output (in parallel with the load) to allow power to the inverter: in the charging battery configuration, we must take care to its discharge level. So in this case the inverter is not always connected, but only if the battery is correctly charged.

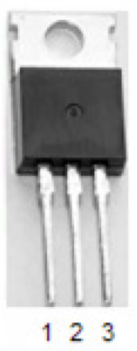
Circuit diagram around DC-DC converter

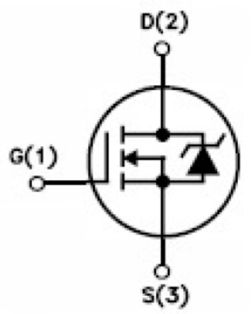


The UTC **MBR30200C** is a 30A schottky barrier rectifier



IRFB4110

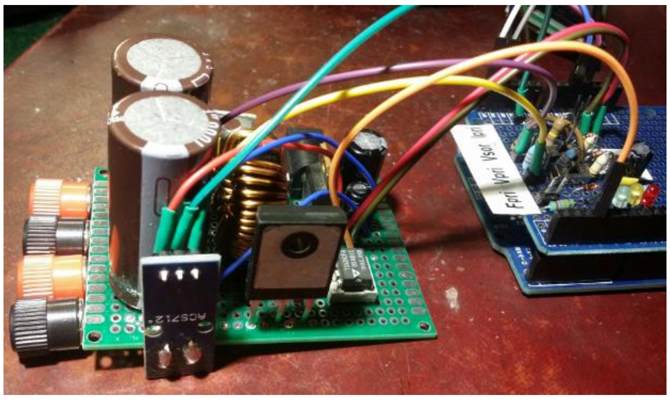




pin 1, Gate
2, Drain
3, Source
TO-220C package

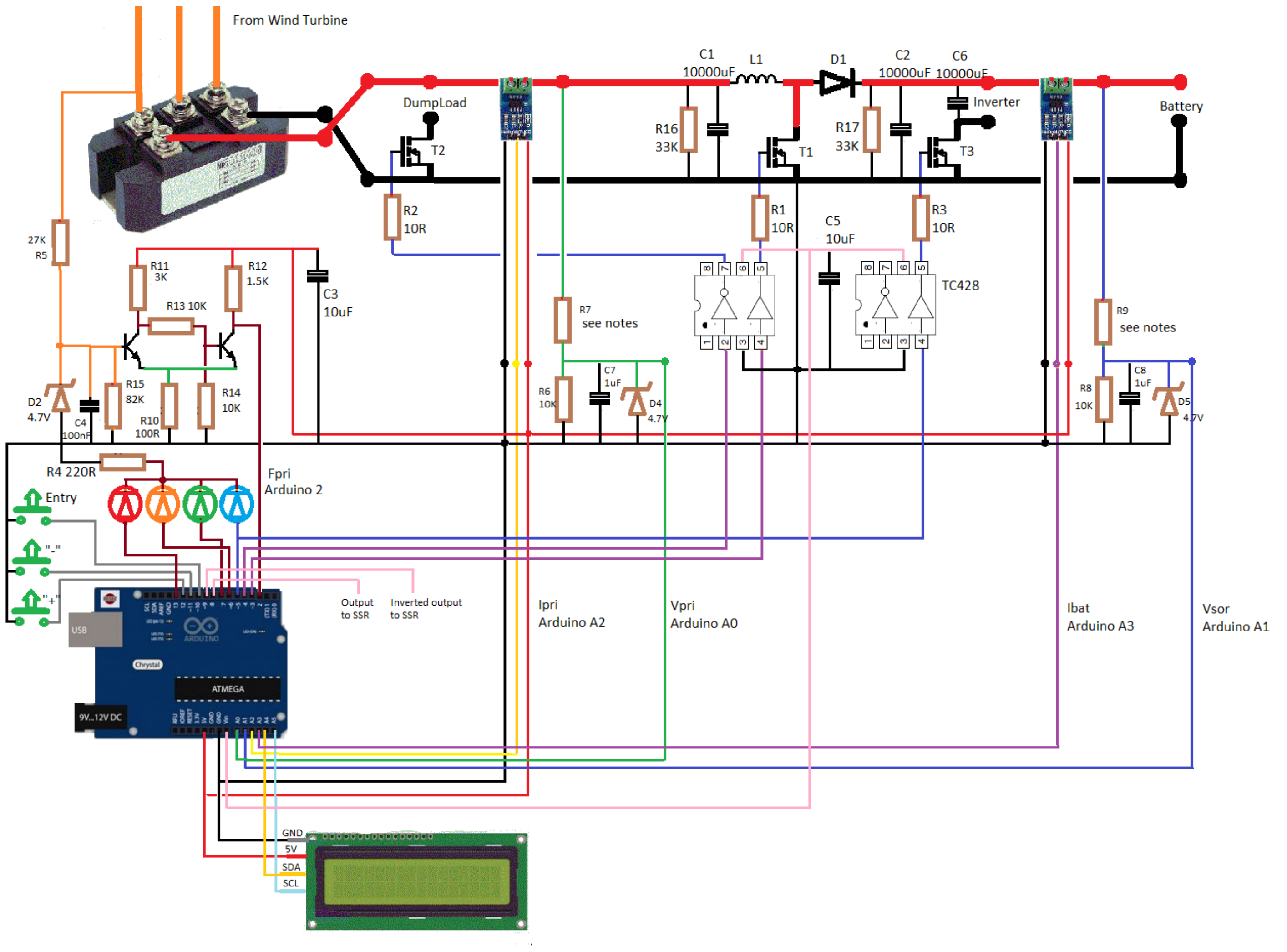
- Drain Current $-I_D= 180A@ T_C=25℃$
- Drain Source Voltage-
  : $V_{DSS}= 100V(Min)$
- Static Drain-Source On-Resistance
  : $R_{DS(on)} = 4.5mΩ (Max)$

Here is a picture as a witness to a previous version:



All together for the final regulator

Complete diagram circuit



Please note: relative to the two precedent diagrams, the 3 push-buttons are now added. There is also an Output and an Inverted Output to connect a SSR: this SSR can connect an external battery charger if the batteries are becoming too empty. Some SSR are active at LOW level, others are active at HIGH level, so both is proposed.

The final program

```
/*
  Wind Turbine MPTT Regulator, for direct injection or battery charging

  ________________________________________________________________
  |                                                              |
  |       author : Philippe de Craene <dcphilippe@yahoo.fr       |
  |       Free of use - Any feedback is welcome                  |
  ________________________________________________________________

  Materials :
  • 1* Arduino Uno R3 - IDE version 1.8.7
  • 2* 20A current sensor ACS712 modules
  • 2* Power MOSFET drivers TC428
  • 1* LCD 1602 with I2C extension
  • 1 DC-DC boost converter : PCB is proposed for the use of a DIP28 Atmega328p

  Arduino Uno pinup (with Atmega328p matching:
• input voltage sensor   : VpriPin   => input A0 = ADC0
• output voltage sensor  : VsorPin   => input A1 = ADC1
• input current sensor   : IpriPin   => input A2 = ADC2
• battery current sensor : IbatPin   => input A3 = ADC4
• SDA for I2C LCD        : SDA       => output A4 = ADC4
• SCL for I2C LCD        : SCD       => output A5 = ADC5
• wind turbine speed     : FpriPin   => input 2 = PD2
• driving PWM signal     : gatePin   => output 3 = PD3 - DC-DC converter driver signal
• dumpload signal        : loadPin   => output 4 = PD4 - dumpload resistor
• inverter enable signal : ondulPin  => output 5 = PD5 - blue LED + inverter
• limit MPPT indicator   : limitPin  => output 6 = PD6 - yellow LED (!!! pin8 in V1.x!!!)
• MPPT indicator         : mpptPin   => output 7 = PD7 - green LED
• external charger       : ssrN_Pin  => output 8 = PB0 - Normal output to SSR
• external charger       : ssrI_Pin  => output 9 = PB1 - Inverted output to SSR
• "ok" push button       : pbE_Pin   => output 10 = PB2
• "-" push button        : pbM_Pin   => output 11 = PB3
• "+" push button        : pbP_Pin   => output 12 = PB4
• overhead alarm         : alarmPin  => output 13 = PB5 - red LED (!!! pin9 in V1.x!!!)



  Versions history :
  version 0.4 - 26 march 2019 - Fpri sensor rebuild with interrupt function
  version 0.5 - 27 march 2019 - Ipri sensor rebuild for average value
  version 0.6 - 26 april 2019 - MPPT algorithm rebuild without Fpri
  version 0.7 - 27 april 2019 - DC-DC converter rebuilt from buck-boost inverter to boost
  version 1.0 -  2 june  2019 - First full working version
  version 1.1 -  5 july  2019 - added EEPROM and menus
  version 1.2 -  5 july  2019 - improvment of the display of voltages
  version 1.3 - 13 july  2019 - improvment of security underload and overload and Ibat measure
  version 1.4 -  8 oct  2019 - new LiquidCrystal-I2C-library and direct injection bug correction
  version 2.0 -  9 oct  2019 - update for PCB

*/

#include <EEPROM.h>                // EEPROM to keep redifined parameters data
#include <LiquidCrystal_I2C.h>     // github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library

const bool VERBOSE = true;//false;        // if true : debugging mode => very slow !
const bool REGLAGE = false;        // if true : for current sensor offset settings
bool USAGE_FPRI = true;            // if true : the turbine speed is calculated
bool mode_injection = false;       // if true : direct injection : no batteries needed

// Wind turbine voltage model : 24 ou 48V
// VpriMax is twice the optimal wind turbine voltage
// a 24V wind turbine can reach 50V  => VpriMaxRef = 50.0V
// a 48V wind turbine can reach 100V => VpriMaxRef = 100.0V
int VpriMaxRef = 50;               // 24V model
int VpriMax = 31;                  // MAx value until DumpLoad Resistor security

// Current sensor model for ACS712 :
// 5A ACS712 module  => 185mV/A
// 20A ACS712 module => 100mV/A
// 30A ACS712 module => 66mV/A
const float convI = 100.0;         // 20A model (float number is required)

// Depending of the way the ACS712 module is wired, polarity ajustment may be required
// to get the charging batteries current positive, values can be 1 or -1
const int IbatPolarity = 1;

// General parameters
const int   Ioffset = 510;         // offset is set with REGLAGE = true, to get Ipri=0 with no
current (~512)
const byte  pwm_gate_Max = 220;    // Max PWM allowed (<250)
int   IbatMax = 15;          // must be = 0,23 time the battery capacity => 13A for 54Ah
int   VpriMin = 15;          // the voltage that will start the MPPT process. Too low the wind
turbine may have difficulties to start
int Vsor_calibrate = 100;    // to adjust Vsor to match real value with multimeter 100 = 100%

// Battery mode parameters (floats numbers)
float VsorMin = 24.0;        // discharged battery voltage : see battery datasheet for exact value
float VsorFlo = 26.6;        // floating voltage : see battery datasheet for exact value
float VsorMax = 29.8;        // maximum voltage : see battery datasheet for exact value

// Direct injection parameters (int numbers)
byte VsorMin_injection = 23;      // see inverter datasheet for correct values
byte VsorFlo_injection = 28;      // injection will start over the "floating value"
byte VsorMax_injection = 59;      // and will stop under the "Min value"

/// inputs outputs declaration
#define VpriPin  A0          // input to Vpri sensor
#define VsorPin  A1          // input to Vsor sensor
#define IpriPin  A2          // input to Ipri sensor
#define IbatPin  A3          // input to Ibat sensor
#define FpriPin   2          // input to Fpri sensor
#define gatePin   3          // pwm output to drive the DC-DC converter circuit (pwm_gate)
#define loadPin   4          // inverted output (because of TC428) dumpload resistor
#define ondulPin  5          // output inverter enabling
#define limitPin  6          // output to yellow LED
#define mpptPin   7          // output to green LED
#define ssrN_pin  8          // output for SSR for external battery charger
#define ssrI_pin  9          // inverted output for external battery charger
#define pbE_Pin  10          // push-button for parameters access
#define pbM_Pin  11          // push-button -
#define pbP_Pin  12          // push-button +
#define alarmPin 13          // output to red LED

// variables for treatment
```

```
float Vpri, memo_Vpri, Vsor;          // input and output voltage
float Ipri, Ibat;                     // input and battery current
float Puiss = 0, memo_Puiss;          // input power
unsigned int Fpri = 0;                // turbine speed in Hertz
int kept_VpriMax = 0;                 // Max measured Vpri for display
int kept_IpriMax = 0;                 // Max measured Ipri for display
int kept_PuissMax = 0;                // Max calculated Puiss for display
int kept_FpriMax = 0;                 // Max measured Fpri for display
unsigned int lect_Ipri_count = 0;     // number of Ipri measures
unsigned long somme_lect_Ipri = 0;    // Ipri measures added between two interrupts (in bytes)
unsigned long somme_lect_Ibat = 0;    // Ibat measures added between two interrupts (in bytes)
volatile bool Fpri_flag = false;      // Fpri flag interruption
unsigned int Fpri_tempo = 0, memo_Fpri_tempo, duration;    // time spent for Fpri measure
int Step = 0;                         // pwm ratio update for pwm_gate
int pwm_gate = 0;                     // pwm signal command for DC-DC converter
byte VsorMinInt, VsorMinDec, VsorFloInt, VsorFloDec, VsorMaxInt, VsorMaxDec;
byte overflow_count = 0;              // count any averflow cycle

// variables for display and menus

unsigned int memo_tempo = 0;          // time flag when Fpri=0
unsigned int memo_tempo_LCD = 0;      // time flag for LCD refresh
unsigned int refresh_tempo = 1000;    // refresh delay for LCD update
bool pbM, memo_pbM, pbP, memo_pbP;
byte ret_push_button = 0;
byte window = 0;
byte count_before_timeout = 0;
byte timeout = 20;


// LCD with I2C declaration :
// documentation : 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, 16, 2);
// => Arduino Uno R3 pin connexion : SDA to A4, SCL to A5


//
// SETUP
//___________________________________________________________________________________________

void setup() {
// inputs outputs declaration
  pinMode(VpriPin, INPUT);          // input for Vpri sensor - input voltage
  pinMode(VsorPin, INPUT);          // input for Vsor sensor - output voltage
  pinMode(IpriPin, INPUT);          // input for Ipri sensor - input current
  pinMode(IbatPin, INPUT);          // input for Ibat sensor - battery current
  pinMode(FpriPin, INPUT);          // input for Fpri sensor - turbine speed
  pinMode(gatePin, OUTPUT);         // pwm output pwm_gate
  pinMode(loadPin, OUTPUT);         // inverted output for dumpload resistor
  pinMode(ondulPin, OUTPUT);        // output for enabling injection
  pinMode(mpptPin, OUTPUT);         // output to green LED
  pinMode(limitPin, OUTPUT);        // output to yellow LED
  pinMode(alarmPin, OUTPUT);        // output to red LED
  pinMode(ssrN_pin, OUTPUT);        // output for SSR for external battery charger
  pinMode(ssrI_pin, OUTPUT);        // inverted output for external battery charger
  pinMode(pbE_Pin, INPUT_PULLUP);   // push-button for menus acces
  pinMode(pbM_Pin, INPUT_PULLUP);   // push-button -
  pinMode(pbP_Pin, INPUT_PULLUP);   // push-button +

// outputs initialisation for no signal
  analogWrite(gatePin, 0);
  digitalWrite(loadPin, HIGH);      // with MOSFET driver circuit TC428 pin 7 output is inverted
  digitalWrite(ondulPin, LOW);      // with MOSFET driver circuit TC428 pin 5 output is non-
inverted
  digitalWrite(ssrN_pin, LOW);      // output for SSR for external battery charger
  digitalWrite(ssrI_pin, HIGH);     // inverted output for external battery charger

  // EEPROM check and data upload :
  // stored data are always positive from 0 to 255.
  // it seems that in cas of first use all are set to 255.
  if (EEPROM.read(0) < 2)   USAGE_FPRI = EEPROM.read(0);     else EEPROM.write(0, USAGE_FPRI);
  if (EEPROM.read(1) < 2)   mode_injection = EEPROM.read(1); else EEPROM.write(1, mode_injection);
  if (EEPROM.read(2) < 131) VpriMax = EEPROM.read(2);        else EEPROM.write(2, VpriMax);
  if (EEPROM.read(3) < 51)  VpriMin = EEPROM.read(3);        else EEPROM.write(3, VpriMin);
  if (EEPROM.read(4) < 41)  IbatMax = EEPROM.read(4);        else EEPROM.write(4, IbatMax);
  VsorMinInt = VsorMin;
  VsorMinDec = 10 * (VsorMin - VsorMinInt);
  VsorFloInt = VsorFlo;
  VsorFloDec = 10 * (VsorFlo - VsorFloInt);
  VsorMaxInt = VsorMax;
  VsorMaxDec = 10 * (VsorMax - VsorMaxInt);
  if (EEPROM.read(5) < 65)  VsorMinInt = EEPROM.read(5);     else EEPROM.write(5, VsorMinInt);
  if (EEPROM.read(6) < 100) VsorMinDec = EEPROM.read(6);     else EEPROM.write(6, VsorMinDec);
  if (EEPROM.read(7) < 65)  VsorFloInt = EEPROM.read(7);     else EEPROM.write(7, VsorFloInt);
  if (EEPROM.read(8) < 100) VsorFloDec = EEPROM.read(8);     else EEPROM.write(8, VsorFloDec);
  if (EEPROM.read(9) < 131) VsorMaxInt = EEPROM.read(9);     else EEPROM.write(9, VsorMaxInt);
  if (EEPROM.read(10) < 100) VsorMaxDec = EEPROM.read(10);   else EEPROM.write(10, VsorMaxDec);
  if (EEPROM.read(11) < 65)  VsorMin_injection = EEPROM.read(11); else EEPROM.write(11,
VsorMin_injection);
  if (EEPROM.read(12) < 100) VsorFlo_injection = EEPROM.read(12); else EEPROM.write(12,
VsorFlo_injection);
  if (EEPROM.read(13) < 200) VsorMax_injection = EEPROM.read(13); else EEPROM.write(13,
VsorMax_injection);
  if ( mode_injection == true ) {
    VsorMin = VsorMin_injection;
    VsorFlo = VsorFlo_injection;
    VsorMax = VsorMax_injection;
  }
  else {
    VsorMin = 1.0 * VsorMinInt + (1.0 * VsorMinDec) / 10.0;
    VsorFlo = 1.0 * VsorFloInt + (1.0 * VsorFloDec) / 10.0;
    VsorMax = 1.0 * VsorMaxInt + (1.0 * VsorMaxDec) / 10.0;
  }
  if (EEPROM.read(14) < 120) Vsor_calibrate = EEPROM.read(14); else EEPROM.write(14,
Vsor_calibrate);


  // Set clock divider for timer 2 at 1 = PWM frequency of 31372.55 Hz
  // Arduino Uno R3 pins 3 and 11
  // https://etechnophiles.com/change-frequency-pwm-pins-arduino-uno/
  TCCR2B = TCCR2B & 0b11111000 | 0x01;

  attachInterrupt(digitalPinToInterrupt(FpriPin), Fpri_detect, RISING);
  // Every state update from down to up of FpripPin the function 'Fpri_detect' is called
  // documentation : https://www.arduino.cc/reference/en/language/functions/external-
interrupts/attachinterrupt/

  // Console initialisation
  Serial.begin(250000);
  Serial.println();
  Serial.println("Ready to start...");
  if( VERBOSE == true ) Serial.println("Verbose mode");
  if( REGLAGE == true ) Serial.println("Current offset mode");
  Serial.println();

  // LCD initialisation
  lcd.begin();               // initialize the lcd for 16 chars 2 lines
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("  Wind Turbine  ");
  lcd.setCursor(0, 1);
  lcd.print(" MPPT regulator ");
  delay(1000);
  lcd.clear();
}   // fin de setup

//
// Fpri_detect : what is done at each interruption
//____________________________________________________________________________________________

void Fpri_detect() {
  Fpri_flag = true;
}

//
// LOOP
//_____________________________________________________________________________________________

void loop() {

  unsigned int tempo = millis();             // time count
  int lect_Ipri = analogRead(IpriPin);
  delayMicroseconds(100);
  int lect_Ibat = analogRead(IbatPin);
  delayMicroseconds(100);

  // overcurrent security
  //_______________________________________________

  if( lect_Ipri < 1 || lect_Ipri > 1022 ) {   // reading in bytes
    analogWrite(gatePin, 0);                 // no driving control to MPPT
    digitalWrite(alarmPin, HIGH);
    return;                                  // nothing else is done
  }

  // cumulative Ipri and Ibat measures between 2 interupts = one turbine rotation
  somme_lect_Ipri += lect_Ipri;
  delayMicroseconds(100);
  somme_lect_Ibat += lect_Ibat;
  delayMicroseconds(100);
  lect_Ipri_count++;
  // Every turbine period, or every second if no wind, or every 100ms is Fpri not measured
  //__________________________________________________

  if( (USAGE_FPRI == true && (Fpri_flag == true || tempo - memo_tempo > 1000))
       || (USAGE_FPRI == false && tempo - memo_tempo > 100) ) {
    noInterrupts();                     // disable any possible interruption
    Fpri_flag = false;                  // flag reset, will be ready to be set at next interrupt
    memo_tempo = tempo;
    memo_Puiss = Puiss;                 // memorization of the previous Puiss measurement
    memo_Vpri = Vpri;                   // memorization of the previous Vpri measurement

    // Measures and calculation of power values
    //__________________________________________________

    //  time spent between 2 interrupts => Fpri calculation
    memo_Fpri_tempo = Fpri_tempo;               // memorization of the previous time measurement
    Fpri_tempo = tempo;                         // memorization of the actual time measurement
    duration = Fpri_tempo - memo_Fpri_tempo;
    if ( duration < 1001 ) Fpri = 1000 / duration; // in Hertz = number of rotations/seconde
    else Fpri = 0;                                // set Fpri = 0 if delay over 1s

    // analogRead measures are in bits : from 0 to 1023, to convert to :
    // -> voltage from 0 to VpriMaxRef for Vpri and Vsor
    // -> current : 511 in bits is the 0mA, 0 in bits matches -Imax, 1023 matches +Imax

    Vpri = (analogRead(VpriPin) / 1023.0) * VpriMaxRef; // Vpri measure
    delayMicroseconds(100);
    Vsor = (analogRead(VsorPin) / 512.0) * VpriMaxRef * (Vsor_calibrate / 100.0); // Vsor measure
    delayMicroseconds(100);
    Ipri = ((float)(somme_lect_Ipri / lect_Ipri_count) - Ioffset) * 5000 / convI / 1023.0; // the
average value of Ipri
    somme_lect_Ipri = 0;                        // reset of the counter of the number of measures
of Ipri
    if ( Ipri < 0 ) Ipri = -Ipri;               // to be sure to get a positive value despite the
way of wiring
    Ibat = ((float)(somme_lect_Ibat / lect_Ipri_count) - Ioffset) * 5000 * IbatPolarity / convI /
1023.0;
    somme_lect_Ibat = 0;                        // reset of the counter of the number of measures
of Ipri
    lect_Ipri_count = 0;

    Puiss = Vpri * Ipri;

    // keep the maximum measured values for display only
    if ( Vpri > kept_VpriMax ) kept_VpriMax = Vpri;
    if ( Ipri > kept_IpriMax ) kept_IpriMax = Ipri;
    if ( Puiss > kept_PuissMax ) kept_PuissMax = Puiss;
    if ( Fpri > kept_FpriMax && Fpri < 100 ) kept_FpriMax = Fpri;

    // setup of the MPPT algorithm
    //__________________________________________________

    // 2 ways that make a lower power :
    // either the wind turbine runs too fast, Vpri increases, so 'step' increases to increase the
current (and so Vpri may decrease)
    // either less wind, Vpri decreases, so 'step' decreases to decreases the current (and so Vpri
may increase)

    if ( memo_Vpri <= Vpri ) {
      if ( Puiss <= memo_Puiss ) Step = 1;
      else Step = -1;
    }
    else {
      if ( Puiss <= memo_Puiss ) Step = -1;
      else Step = 1;
    }

    // Management of DC-DC converter cutting control
    //__________________________________________________

    if ( Vpri < VpriMin ) {                   // lower limit input voltage value reached
      Step = Step - 10;                       // sharp decline of step to try to increase Vpri
      digitalWrite(mpptPin, LOW);             // green LED is OFF
      digitalWrite(limitPin, LOW);
      digitalWrite(ondulPin, LOW);
    }
    else if ( Vsor < VsorMin) {               // lower limit output volatge
      digitalWrite(ondulPin, LOW);            // lower limit output voltage value reached
      if ( Step < 0 ) {
        Step = -Step;                         // 'Step' is forced to be positive
        digitalWrite(limitPin, HIGH);         // yellow LED is ON
      }
    }
    else {
      digitalWrite(limitPin, LOW);
      digitalWrite(mpptPin, HIGH);
      if ( Vsor > VsorFlo ) digitalWrite(ondulPin, HIGH);  // inverter is ON as soon as VsorFloat
is reached
    }
    if ( Ibat > IbatMax && Step > 0 ) Step = -Step;          // 'Step' is forced to be negative



    // Overcharge security & pwm ratio update
    //_________________________________________________

    if ( Vsor > VsorMax ) {
      Step = Step - 10;                       // decrease pwm ratio
      overflow_count++;                       // count the number of overfolw cycles
      digitalWrite(alarmPin, HIGH);           // red LED is ON
      if ( overflow_count > 5 ) {             // after 5 cycles of overflow
        pwm_gate = -10;                       // cutting control is stopped
        digitalWrite(loadPin, LOW); }         // dumpload is ON - remember that TC428 pin 7 is
inverted
    }
    else {
      digitalWrite(loadPin, HIGH);            // dumpload is OFF
      digitalWrite(alarmPin, LOW);            // red LED is OFF
      overflow_count = 0;                     // reset the overflow cycles count
    }

    // constrain the pwm ratio
    pwm_gate += Step;
    if ( pwm_gate > pwm_gate_Max ) pwm_gate = pwm_gate_Max;  // high value limit
    else if ( pwm_gate < 0 )  pwm_gate = 0;                  // low value limit

    analogWrite(gatePin, pwm_gate);           // cutting command update before any dumpload
evaluation
    interrupts();                             // interrupts enable again
/*
    // for debugging purpose only
    if ( VERBOSE == true ) {
      Serial.print("Fpri= ");               Serial.print(Fpri);
      Serial.print("  Vpri= ");             Serial.print(Vpri);
      Serial.print("  Ipri= ");             Serial.print(Ipri);
      Serial.print("  Puiss= ");            Serial.print(Puiss);
      Serial.print("  Puiss-memo_Puiss= "); Serial.print(Puiss - memo_Puiss);
      Serial.print("  Step : ");            Serial.print(Step);
      Serial.print("  pwm_gate : ");        Serial.print(pwm_gate);
      Serial.print("  Vsor= ");             Serial.print(Vsor);
      Serial.print("  Ibat= ");             Serial.print(Ibat);
      Serial.println();
    }*/
  }   // end of Fpri 1 period cycle
/*
  if ( REGLAGE == true ) {
    Serial.print("valeur de I=0 en bits : ");
    Serial.print(analogRead(IpriPin) - Ioffset);
    Serial.println();
  }*/

  // LCD and menus management + ext battery charger
  //_________________________________________________

  // every seconds look for push-button activity and update display
  if ( tempo - memo_tempo_LCD > refresh_tempo ) {
    memo_tempo_LCD = tempo;
    ret_push_button = push_button();         // reading push-button status here only
    lcd.setCursor(0, 0);
    count_before_timeout++;
    if ( count_before_timeout > timeout ) lcd.noBacklight();
    if ( ret_push_button == 1 ) {
      if ( window != 4 ) next_window();
      else {
        window = 0;
        lcd.clear();
      }
    }
    if ( mode_injection == true && window == 7 ) window++;

    // usual display with 2 choises : window 0 and window 1
    if ( window < 2 ) {
      lcd.print("Ve=");
      lcd.print(String(Vpri, 1));
      lcd.setCursor(9, 0);
      lcd.print("Vs=");
      lcd.print(String(Vsor, 1));
      lcd.setCursor(0, 1);
      if ( window == 0 ) {
        if ( USAGE_FPRI == true ) {
          lcd.print(Fpri);
          lcd.print("Hz");
        }
        lcd.setCursor(8, 1);
        lcd.print("Pe=");
        lcd.print(String(Puiss, 1));
      }
      else if ( window == 1 ) {
        lcd.print("Ie=");
        lcd.print(String(Ipri, 1));
        lcd.setCursor(9, 1);
        lcd.print("Ib=");
        lcd.print(String(Ibat, 1));
      }
    }    // end of usual display
    else {

      // if window >= 2 we are entering in max values display and parameters setup

      if ( count_before_timeout > timeout ) {  // timeout to return to usual display if no job done
        count_before_timeout = 0;
        window = 0;
        lcd.clear();
      }
      if ( window == 2 ) {
        lcd.print("VM=");
        lcd.print(kept_VpriMax);
        lcd.setCursor(9, 0);
        lcd.print("IM=");
        lcd.print(kept_IpriMax);
        lcd.setCursor(0, 1);
        if ( USAGE_FPRI == true ) {
          lcd.print(kept_FpriMax);
          lcd.print("Hz");
        }
        lcd.setCursor(8, 1);
        lcd.print("PM=");
        lcd.print(kept_PuissMax);
      }  // end of window 2
      if ( window == 3 ) {
        lcd.print("Reset MAX val. ?");
        lcd.setCursor(0, 1);
        lcd.print("push + to reset");
        if (ret_push_button == 2) {
          kept_VpriMax = 0;
          kept_IpriMax = 0;
          kept_PuissMax = 0;
          kept_FpriMax = 0;
          lcd.setCursor(0, 1);
          lcd.print("values reseted ");
          window = 2;
          lcd.clear();
        }
      }  // end of window 3
      if ( window == 4 ) {
        lcd.print("Parameters setup");
        lcd.setCursor(0, 1);
        lcd.print("push + to review");
        if ( ret_push_button > 1 ) next_window();
      }   // end of wondows 4
      if ( window == 5 ) {
        if ( ret_push_button > 1 ) USAGE_FPRI = ! USAGE_FPRI;
        lcd.print("Turbine speed :");
        lcd.setCursor(0, 1);
        if ( USAGE_FPRI == true ) lcd.print("measured");
        else lcd.print("not measured");
      }   // end of window 5
      if ( window == 6 ) {
        if ( ret_push_button > 1 ) mode_injection = ! mode_injection;
        if ( mode_injection == true ) {
          VsorMin = VsorMin_injection;
          VsorFlo = VsorFlo_injection;
          VsorMax = VsorMax_injection;
          lcd.print("Injection mode");
        } else {
          VsorMin = 1.0 * VsorMinInt + (1.0 * VsorMinDec) / 10.0;
          VsorFlo = 1.0 * VsorFloInt + (1.0 * VsorFloDec) / 10.0;
          VsorMax = 1.0 * VsorMaxInt + (1.0 * VsorMaxDec) / 10.0;
          lcd.print("Battery mode");
        }
        lcd.setCursor(0, 1);
        lcd.print("-/+ to modify");
      }   // end of wondows 6
      if ( window == 7 ) {
        if (ret_push_button == 2) VpriMax++;   // if "+" pushed
        if (ret_push_button == 3) VpriMax--;   // if "-" pushed
        VpriMax = constrain(VpriMax, 24, 130);
        lcd.print("U input MAXI");
        lcd.setCursor(0, 1);
        lcd.print("VpriMax = ");
        lcd.setCursor(10, 1);
        lcd.print(VpriMax);
        lcd.print("V");
      }  // end of window 7
      if ( window == 8 ) {
        if (ret_push_button == 2) VpriMin++;
        if (ret_push_button == 3) VpriMin--;
        lcd.print("U input MINI");
        lcd.setCursor(0, 1);
        lcd.print("VpriMin = ");
        lcd.setCursor(10, 1);
        lcd.print(VpriMin, 1);
        lcd.print("V");
      }  // end of wondows 8
      if ( window == 9 ) {
        if (ret_push_button == 2) IbatMax++;
        if (ret_push_button == 3) IbatMax--;
        lcd.print("I battery MAXI");
        lcd.setCursor(0, 1);
        lcd.print("IbatMax = ");
        lcd.setCursor(10, 1);
        lcd.print(IbatMax);
        lcd.print("A");
      }  // end of window 9
      if ( window == 10 ) {
        if ( mode_injection == true ) {
          if (ret_push_button == 2) VsorMin++;
          if (ret_push_button == 3) VsorMin--;
          VsorMin_injection = VsorMin;
          lcd.print("U inverter STOP");
        } else {
          if (ret_push_button == 2) VsorMin = VsorMin + 0.1;
          if (ret_push_button == 3) VsorMin = VsorMin - 0.1;
          VsorMinInt = VsorMin;
          VsorMinDec = 10 * (VsorMin - VsorMinInt);
          lcd.print("U Battery MINI");
        }
        lcd.setCursor(0, 1);
        lcd.print("VsorMin = ");
        lcd.setCursor(10, 1);
        lcd.print(VsorMin, 1);
        lcd.print("V");
      }  // end of window 10
      if ( window == 11 ) {
        if ( mode_injection == true ) {
          if (ret_push_button == 2) VsorFlo++;
          if (ret_push_button == 3) VsorFlo--;
          VsorFlo_injection = VsorFlo;
          lcd.print("U inverter START");
        } else {
          if (ret_push_button == 2) VsorFlo = VsorFlo + 0.1;
          if (ret_push_button == 3) VsorFlo = VsorFlo - 0.1;
          VsorFloInt = VsorFlo;
          VsorFloDec = 10 * (VsorFlo - VsorFloInt);
          lcd.print("U Battery FLOAT");
        }
        lcd.setCursor(0, 1);
        lcd.print("VsorFlo = ");
        lcd.setCursor(10, 1);
        lcd.print(VsorFlo, 1);
        lcd.print("V");
      }  // end of window 11
      if ( window == 12 ) {
        if ( mode_injection == true ) {
          if (ret_push_button == 2) VsorMax++;
          if (ret_push_button == 3) VsorMax--;
          VsorMax_injection = VsorMax;
          lcd.print("U inverter MAXI");
        } else {
          if (ret_push_button == 2) VsorMax = VsorMax + 0.1;
          if (ret_push_button == 3) VsorMax = VsorMax - 0.1;
          VsorMaxInt = VsorMax;
          VsorMaxDec = 10 * (VsorMax - VsorMaxInt);
          lcd.print("U Battery MAXI");
        }
        lcd.setCursor(0, 1);
        lcd.print("VsorMax = ");
        lcd.setCursor(10, 1);
        lcd.print(VsorMax, 1);
        lcd.print("V");
      }  // end of window 12
      if ( window == 13 ) {
        if (ret_push_button == 2) Vsor_calibrate++;
        if (ret_push_button == 3) Vsor_calibrate--;
        lcd.print("U Battery adjust");
        lcd.setCursor(0, 1);
        lcd.print("-/+ modify: ");
```

```
        lcd.print(String(Vsor, 1));
      }  // end of window 13
      if ( window == 14 ) {
        lcd.print("Debug mode");
        lcd.setCursor(0, 1);
        lcd.print("s:"); lcd.print(Step); lcd.print("  ");
        lcd.setCursor(8, 1);
        lcd.print("p:"); lcd.print(pwm_gate); lcd.print("  ");
      }  // end of window 14

      // EEPROM updated if needed
      EEPROM.update(0, USAGE_FPRI);
      EEPROM.update(1, mode_injection);
      EEPROM.update(2, VpriMax);
      EEPROM.update(3, VpriMin);
      EEPROM.update(4, IbatMax);
      EEPROM.update(5, VsorMinInt);
      EEPROM.update(6, VsorMinDec);
      EEPROM.update(7, VsorFloInt);
      EEPROM.update(8, VsorFloDec);
      EEPROM.update(9, VsorMaxInt);
      EEPROM.update(10, VsorMaxDec);
      EEPROM.update(11, VsorMin_injection);
      EEPROM.update(12, VsorFlo_injection);
      EEPROM.update(13, VsorMax_injection);
      EEPROM.update(14, Vsor_calibrate);
    }    // end of parameters reviewm
  }      // end of LCD display
}        // end of loop

//
// NEXT_WINDOW : next window procedure
//______________________________________________________________________________________________

void next_window() {

  window = (window + 1) % 15;       // next window modul 10
  ret_push_button = 0;              // reset the buttun state
  lcd.clear();
  lcd.setCursor(0, 0);
}     // end of next_window function

//
// PUSH_BUTTON : return value depending of the state of the 3 push-buttons
//______________________________________________________________________________________________

byte push_button() {

  memo_pbM = pbM; memo_pbP = pbP;     // memorization for past state of + - push-button
  pbP = digitalRead(pbP_Pin);
  pbM = digitalRead(pbM_Pin);

  if ( digitalRead(pbE_Pin) == 0 ) {
    count_before_timeout = 0;         // reset the timeout counter
    refresh_tempo = 1000;
    lcd.backlight();                  // switch on display
    lcd.clear();
    return 1;
  }
  if ( pbP == 0 ) {
    count_before_timeout = 0;                  // reset the timeout counter
    if ( memo_pbP == 0 ) refresh_tempo = 300;  // temporary lower display update duration
    lcd.backlight();                           // switch on display
    return 2;
  }
  if ( pbM == 0 ) {
    count_before_timeout = 0;                  // reset the timeout counter
    if ( memo_pbM == 0 ) refresh_tempo = 300;  // temporary lower display update duration
    lcd.backlight();                           // switch on display
    return 3;
  }
  refresh_tempo = 1000;             // return back to usual display update duration
  return 0;
}     // end of push_button function
```

Some tips about the program

- Some parameters can be modified once the program is running: VpriMin, VpriMAx, IbatMax, VsorMin, VsorFlo, VsorMax and Vsor_calibrate. Predefined values are proposed in the program for general use.

- Some parameter that are mostly hardware dependent cannot be modified once the program is compiled like: the model of current sensor stored in conI, the current offset Ioffset, the max ratio of cutting signal pwm_gate_Max, IbatPolarity, VERBOSE and REGLAGE.
  - pwm_gate_Max must be under 250. The highest is this value, the highest can reach Vsor, but more delicate the life of the CMOS transistor T1.
  - IbatPolarity is set to 1 or -1, in order to read a positive Ibat value when the battery is charging
  - REGLAGE allows the setup of Ioffset in order to get 0.0 when there is no current. This is done only one time during tests. Value should be 511 or 512.
  - VERBOSE allows watching most important variables on the console. Remind that this slows a lot the program, and should be set during test only.

  Please note : VERBOSE and REGLAGE mode cannot be anymore useful if the circuit is directly made on the PCB, because of the lack of the USB serial port only available on the Arduino Uno.

- Pwm frequency is set to 31 KHz. It is a low frequency that allows quite long wires between the CPU to the booster TC428 and then to the CMOS transistor.
- Turbine speed rotation is set by calculating the time spent between two interrupts. The interrupts are set on each rising of the square signal issue from one coil of the turbine. It gives a result in Hertz, not in rpm (tour per minute) as the number of poles of the generator varies depending of the model.
- Ipri and Ibat are calculated from the average of multiple measures between two interrupts. The reason why is that we are working in high power cutting voltage, and the current is not linear.

Instructions for use

1. Please note : **The wind turbine must be stopped before Regulator power on. The battery must also be switched off**.
   Each time you will switch on the regulator under wind turbine power, you will burn the cutting CMOS transistor. This is because "indeterminate things" happens during the short power on time. For the moment the goal is to build a quite simple device, to prevent against those "indeterminate things" will need a dozen of external components that would complicate the circuit.
2. Plug the wind turbine, the Dump Load Resistor, the inverter and the battery (if any) at the appropriate places.
3. The converter needs 12V DC to operate. Power on the converter.
4. Once on, the LCD displays :
   - Ve for Vpri : input voltage of the wind turbine (after the rectifier)
   - Vs for Vsor : output voltage of the regulator which is the battery voltage
   - Fe for Fpri : rotation speed of the turbine in Hertz or rotations per seconds
   - Pe for Puiss : Ve * Ipri
5. Push "Entry" : the display is changed to :
   - Ve for Vpri : input voltage of the wind turbine (after the rectifier)
   - Vs for Vsor : output voltage of the regulator which is the battery voltage
   - Ie for Ipri : the input current
   - Ib for Ibat : the battery current
6. Push "Entry": the display shows the maximum measured values :
   - VM for kept_VpriMax is the maximum value measured for Vpri
   - IM for kept_IpriMax is the maximum value measured for Ipri
   - The second line will indicate, if installed, the maximum speed measured for Fpri in Hz
   - PM for kept_PuissMax is the maximum input power calculated for Puiss
7. Push again "Entry" : the display proposes to reset these values. Press "+" will reset. Of course, these values need some storms to become interesting…
8. Push again "Entry" : the display proposes to enter in parameters if "+" pushed.
9. Push "+": we enter in the parameters menu. Any change is immediately taken in account. Each push "Entry" will review one by one parameters in this order :

10. Display: Fpri measured or not: we can change with “+” or “-“.
    If not measured, MPPT will update every 100ms. Otherwise, update time is proportional to the turbine speed.
11. Display: Battery mode or Inverter mode: we can change with “+” or “-“.
    In battery mode, the parameters that come will define the absolute characteristics of the battery.
    In inverter mode, they will define the working area of the inverter.
12. Display : VpriMax: absolute highest wind turbine voltage
13. Display : VpriMin: minimum voltage for regulator starts
14. Display : IbatMax: absolute highest battery charging current
15. Display : VsorMin: defined according the battery or inverter mode :
    - Absolute lower voltage to consider the battery as discharged.
    - Lower value that shutoff the inverter.
16. Display: VsorFlo defined according the battery or inverter mode :
    - Floating voltage for battery.
    - Level value that startup the inverter.
17. Display: VsorMax : defined according the battery or inverter mode :
    - Absolute higher voltage during the battery charge.
    - Absolute higher allowable voltage for the inverter
18. Display: Vsor_calibrate : define the correction factor for the most accurate reading measure of Vsor. At this state, Push “-“ or “+” to adjust Vsor (displayed at the same time) according to a multimeter reading. Default value is 100 for 100%.
19. Display debug information: ‘s’ for the ‘Step’ value and ‘p’ for’ pwm_gate.

After a while the display will automatically light off, the pull of any of the 3 push-button for arrould 1 second will light on it back.

How to charge a battery

The main mater with green energy is that it is rarely available when we need it. A solar panel alone will never light any lamp during night, nor a wind turbine will help if there is no wind.

So, the first idea that come in mind is to use a battery. Unfortunately a battery has a very bad efficiency, 50% of energy is lost for charging a battery. So, we should consider using a battery only for required equipment like lights during nights.

The second problem of a battery is its lifetime. However, well kept a battery can stay “alive” for 10 years…

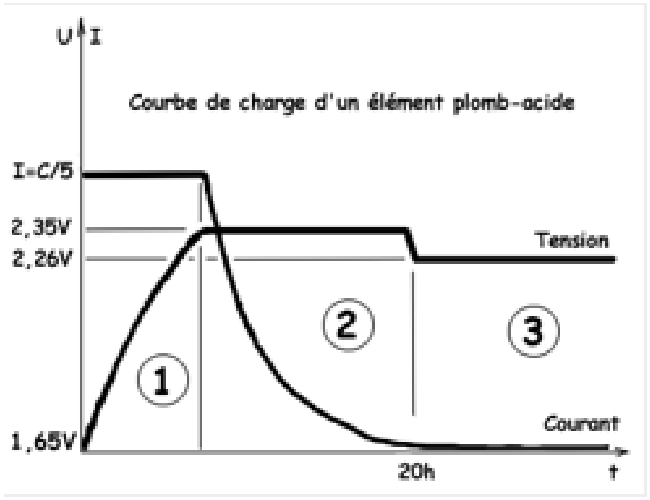
One of the elements guaranteeing a long lifetime is the respect of its electrical characteristics:

Each model of battery owns its proper characteristics. Each technologies owns its advantages and drawbacks. The lead acid battery has the advantage to be very cheap and robust, with a poor self-discharge. Its first disadvantage is its low watt/kg rate.

Usually for any 12V lead acid battery, the characteristics are:

- The nominal working voltage is 12.6V
- When discharged, the voltage fall down around 11V. The lower is this voltage, the lower is the lifetime, a battery suffers a lot when over discharged. The problem with a wind turbine is that a battery can stay a very long time discharged if there is no wind… Moreover, more the discharge is slow, the better it is.
  For the regulator the default value for VsorMin = 24.0V = 2*12.0V
- The highest voltage of charge is around 14.4V. Never go over. It is the problem of cheap car battery charger which most of them go over.
  For the regulator the default value for VsorMax = 28.8V = 2*14.4V

- The highest charging current. It is 0.23 time the capacity of the battery. It is the other problem of cheap car battery charger, the current is not controlled and will remain the same for small or big capacity battery.
  For the regulator the IbatMax = 13A, which is approximatively the maximum current that a 350W wind turbine can deliver. The best battery capacity in this case is 54A/h.
- The floating voltage, it is the voltage to ensure a complete charge.



The time spent for a complete charge cycle is 20 hours. It begins under Imax (phase 1 in draw).

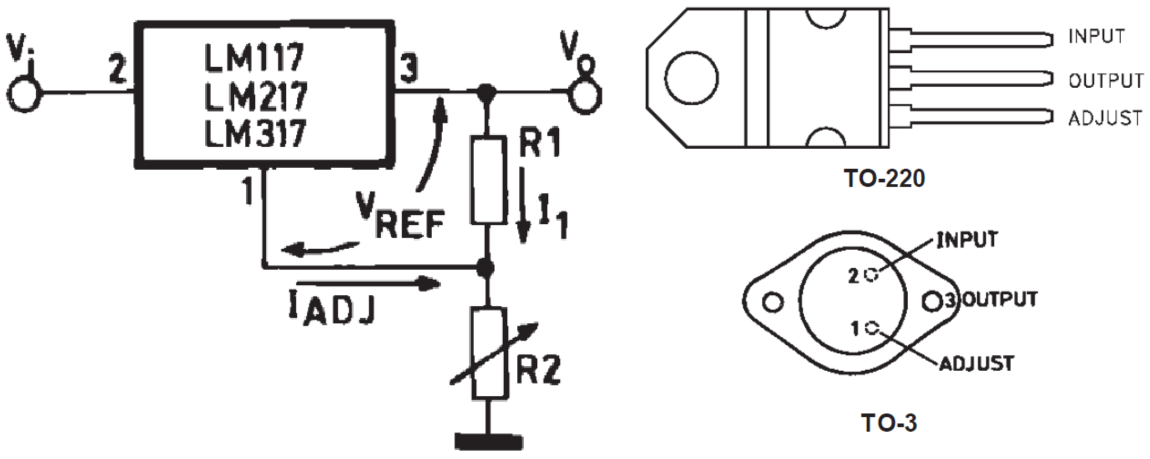
After a while the current becomes lower and lower, so the voltage become higher and higher, until Vmax (phase 2).

20 hours after the voltage is forced declined until Vfloat (phase 3). The regulator does not perform such a sophisticate battery charge procedure. In fact we suppose that there is not wind enough for 20 hours, and we suppose also that the battery will discharge every day. So we suppose the battery will never be full charged.

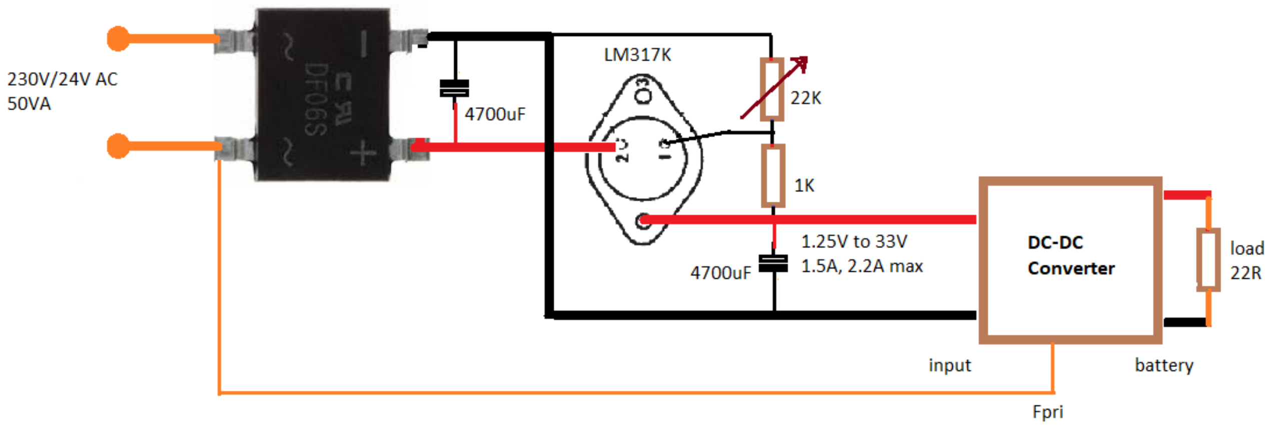
How to test the regulator

The Simple way to test if MPPT is working well is with the help of a power supply that uses the LM317K (TO3 case seems to give more current).

The LM317K is a voltage regulator that offers an output adjustable voltage from 1.2V to 37V, with a maximum current of 2.2A and a security current of 1.5A. That means if the current exceed 2.2A then a security drop it to a limit of 1.5A.



Here is an example of circuit for tests:



First upload the program in `VERBOSE = TRUE;` (this let you see all variables evolve and slow down a lot the program execution) then adjust the voltage to the lowest value. The display should show all values near zero except Fpri = 50Hz.

Then slowly increase the voltage. You must notice that as soon as input voltage exceeds VpriMin the MPPT starts : pwm increases and Vsor increases as well. Whatever the voltage applied you should notice that current stays around 2A, in this case Puiss is about twice the voltage. With 30V input you can get 60W on the load. Take care of the LM317 and the load become very hot. However, the converter CMOS transistor T1 and the barrier Shottky D1 stay cold.

With accurate measures with the help of a multimeter the efficiency overcomes 92%.

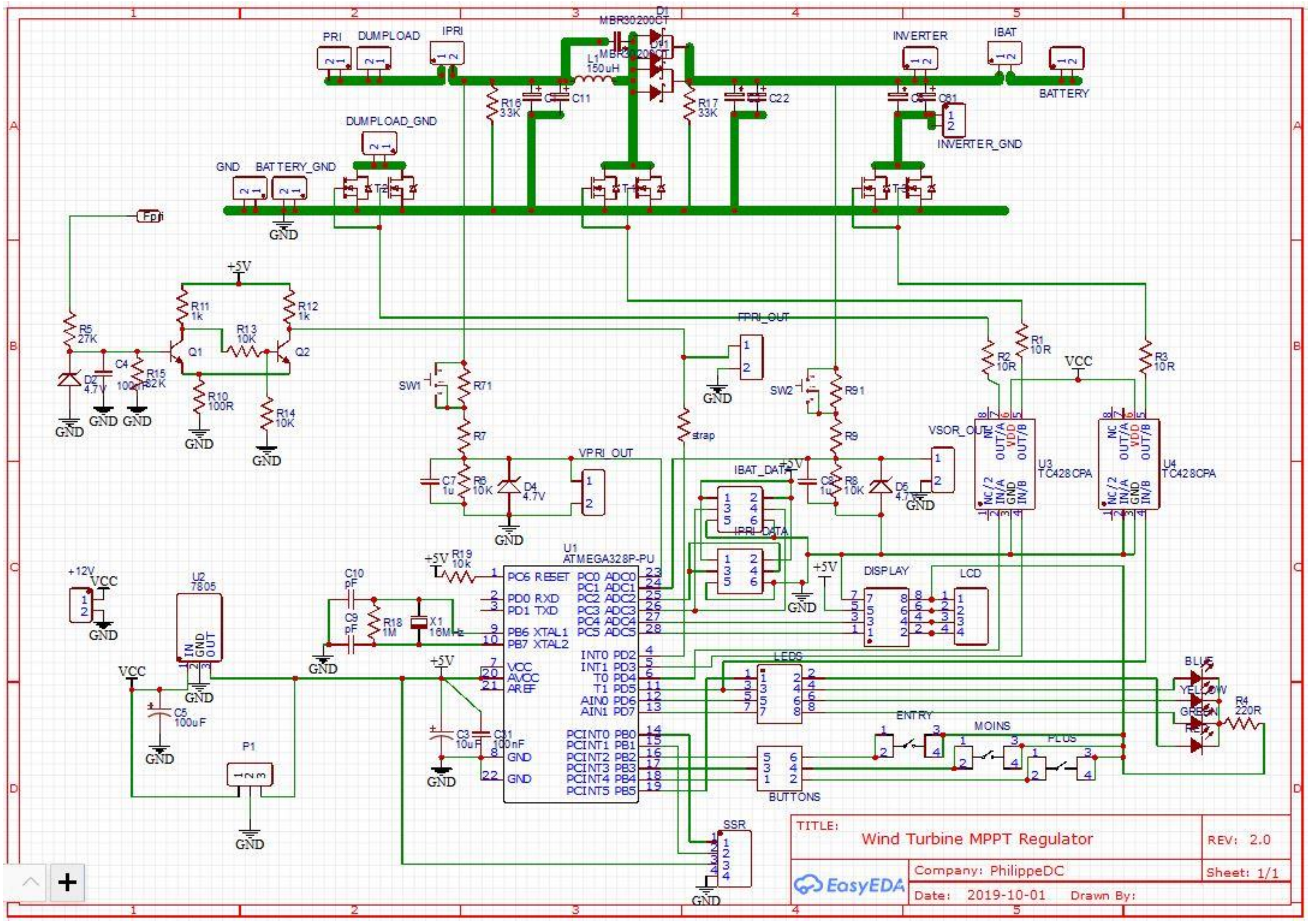
Putting all on a PCB

For the PCB I use the the EasyEDA's services : https://easyeda.com

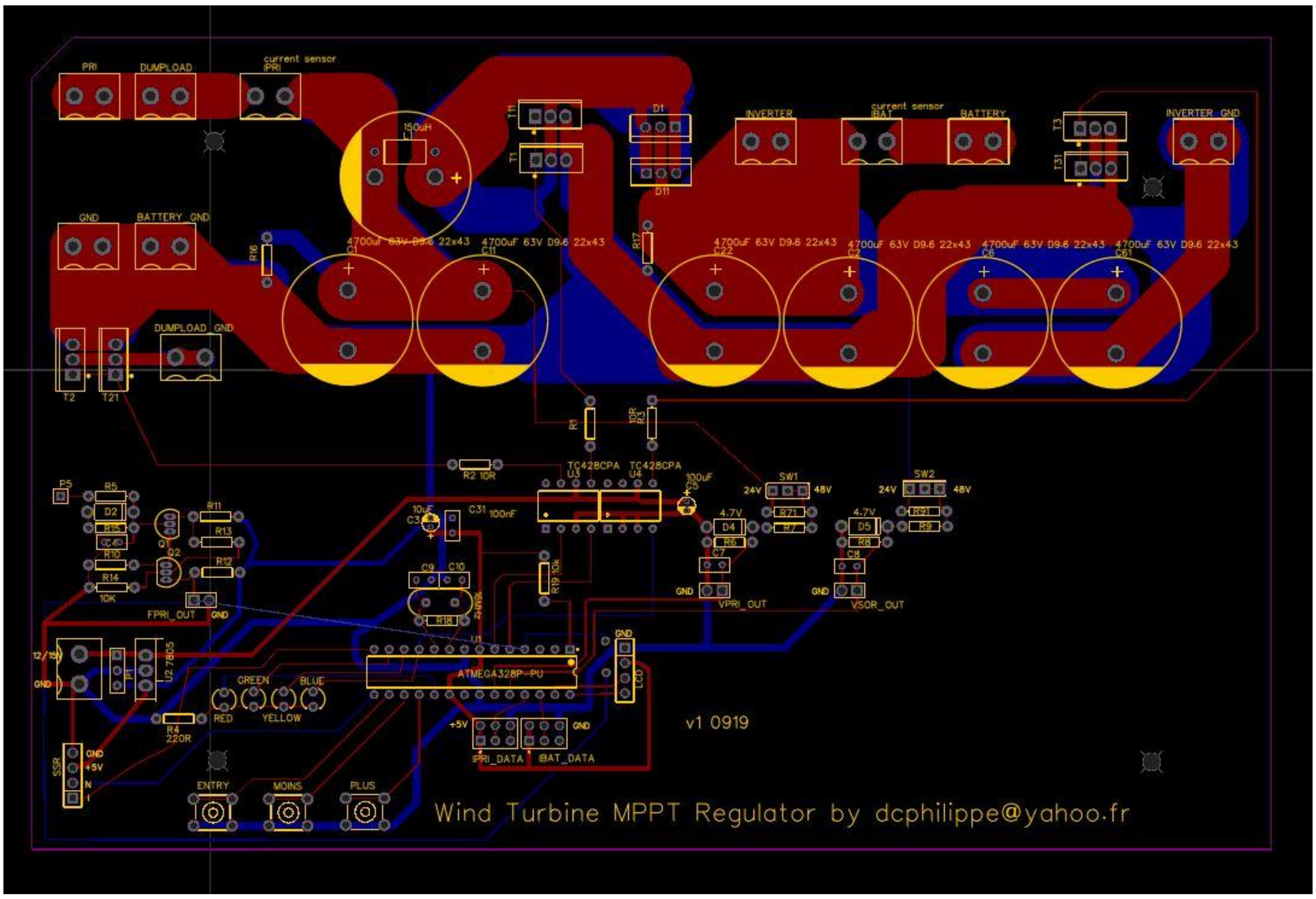
First a circuit diagram must be drawn. Then a PCB is proposed, we are free to place components wherever we want…

With the use of the PCB, only the ATmega328p DIP-28 microcontroller came from the Arduino Uno. For the clock service, a 16Mhz crystal and 2 capacitors of 22pF must be added (X1, C9 and C10 on the diagram below).

Below is the diagram made on EasyEDA. All power components – CMOS transistors and Schottky Diode – can be parallelize by 2 which is better for high current.



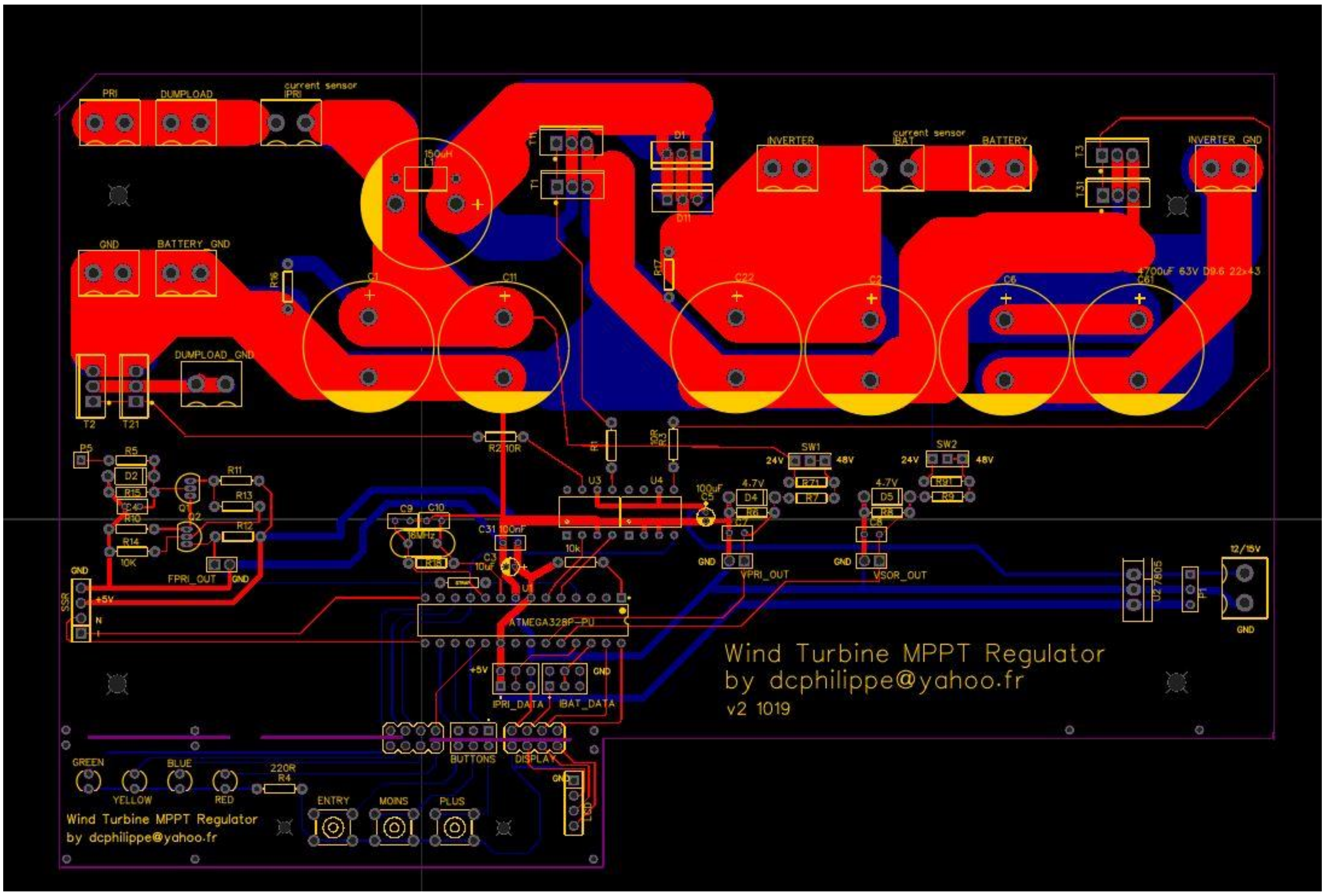
… and the PCB version 1:



Updates: PCB version 2

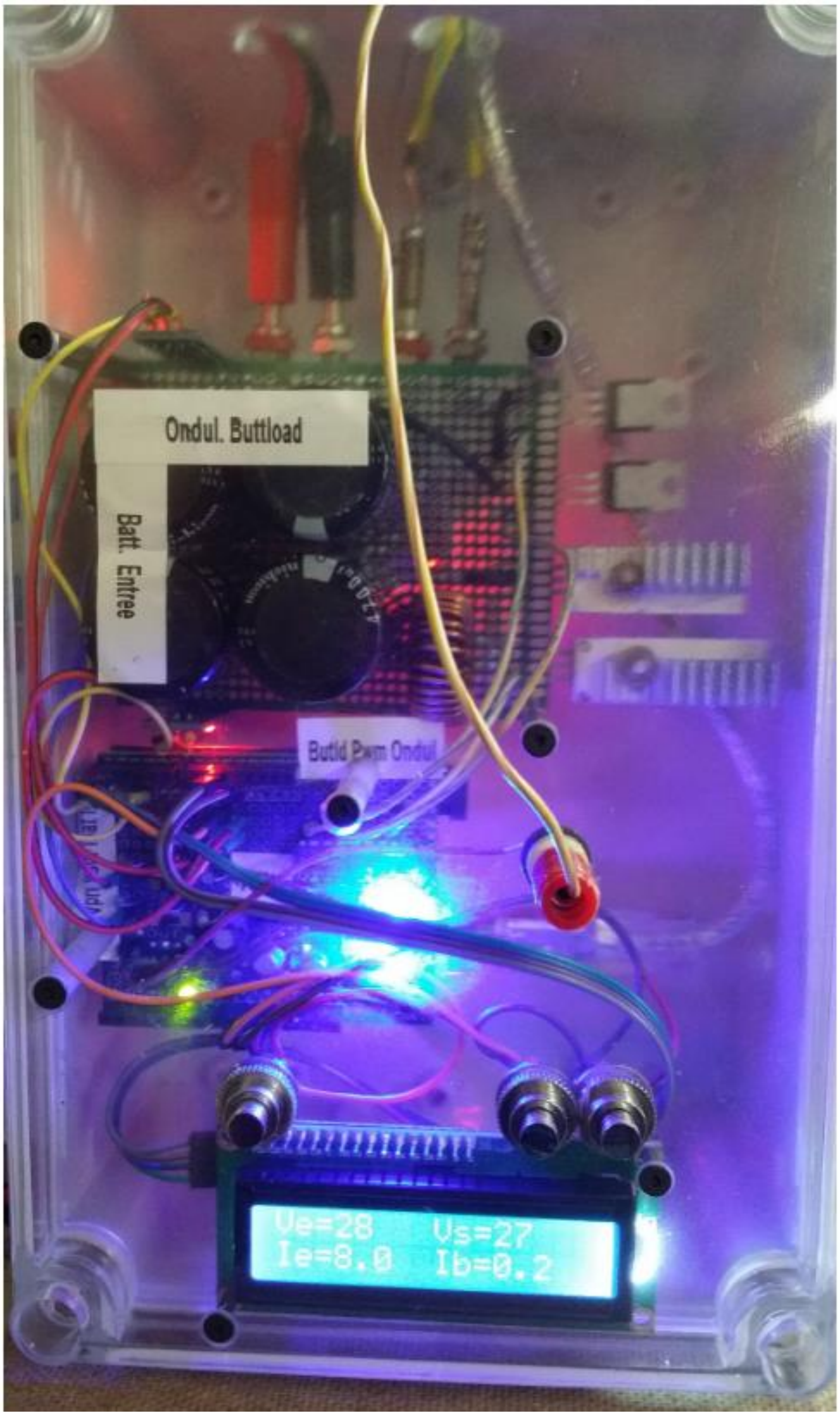
Here are feedbacks that helped to improve the regulator.

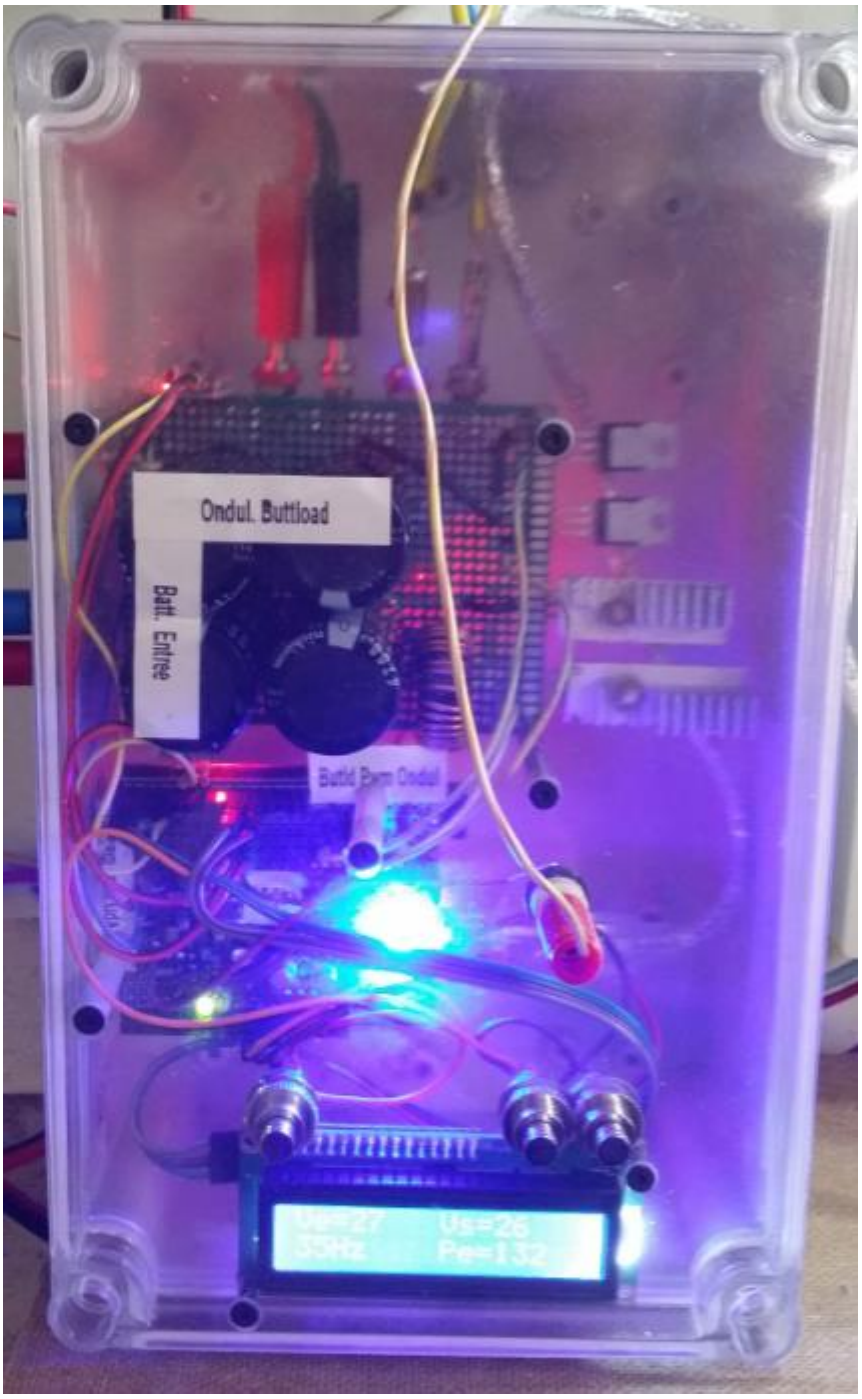
One request was to get a smaller PCB with push-buttons, leds and LCD display separated apart on the PCB :



Illustrations in use

Prototype in working conditions. Note the green led and the blue one are on:

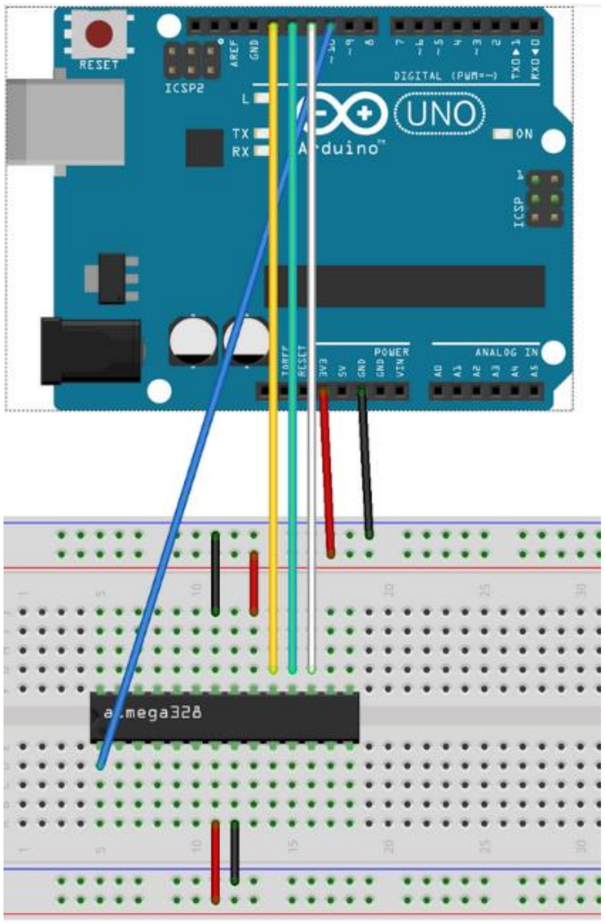

Ondul. Buttload
Batt. Entree
Butid Pwm Ondul
Ve=28 Vs=27
Ie=8.0 Ib=0.2



The PCB V1 under tests:

How to code the ATmega320p alone?

I broke 4 Arduino Uno devices during the studies… So I buy the cheapest… So when I decide of a PCB with a ATmega328p alone without its Arduino circuit, I had to buy DIP28 CPU. Within a probability of 99.99%, you will get them without bootloader, except the one you bought within the expensive native Arduino Uno. But after the realization of this Regulator what will serve an Arduino Uno without its ATmega328p?



Here is a useful explanation how to install a bootloader on a CPU – not only the ATmega328p -:

1- Make this circuit :

Arduino Uno pin	ATmega328p pin
Numeric 10	PC6 (RESET) pin1
Numeric 11 (MOSI)	PB3 (MOSI) pin 17
Numeric 12 (MISO)	PB4 (MISO) pin 18
Numeric 13	PB5 (SCK) pin 19
VCC = 5V	Pin 7 and 10
GND = 0V	Pin 8 and 22

2- Plug the Arduino to the PC, run IDE (tested version 1.8.7 and 1.8.9) with the following:

3- Upload to the Arduino Uno the example program: “11.ArduinoISP”
4- By the “Tools” menu : choose “Programmer” -> Arduino as ISP
5- By the “Tools” menu : “Burn Bootloader”

That’s all!!! Your extra ATmega328p can fit now inside your expensive official Arduino Uno.

Please note by my own experience I had to redo twice the “Burn Bootloader” to get it operate without errors and I do not know why…