Projet BE : Chargeur de batterie par panneaux solaires

L3 REL EEA

Université Paul Sabatier III

GABET Florian

DIOP Ibrahima

FAYE Issa

Tables des matières

1) Principe de fonctionnement

I) Présentation du projet :

Le but de ce projet est de pouvoir recharger une batterie à l'aide d'un panneau solaire dans des conditions optimale et de manière totalement automatique.

Le principe serait d'utiliser une carte Arduino programmable afin de commande un convertisseur pour adapter la tension aux bornes de la batterie à chaque instant en fonction de l'ensoleillement et des valeurs relevés par le capteur.

II) Cahier des charges :

Afin de réaliser le projet dans les meilleures conditions possible nous avons à notre disposition le cahier des charges suivant :

- Analyser les caractéristiques d'un module photovoltaïque.

- Comprendre l'intérêt d'intercaler un convertisseur DC/DC entre le générateur photovoltaïque et la charge continue.

- Utiliser les relations de dimensionnement d'un hacheur pour retrouver les paramètres des composants passifs (capacité, inductance).

- Analyser des commandes extrémales (recherche du maximum de puissance)

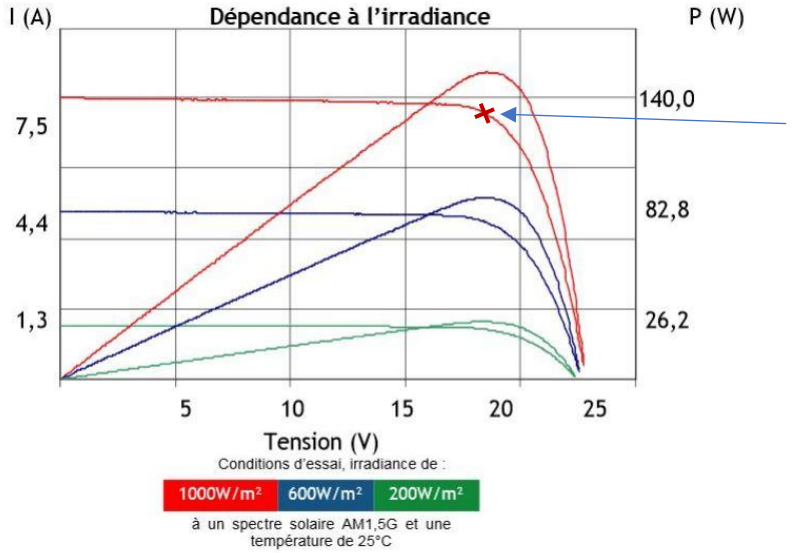
III) Liste du matériel :

Pour cela nous avons le matériel suivant à notre disposition :

- 1 Carte programmable Arduino UNO
- 1 Capteur de température / humidité
- 1 Base shield
- 1 Cable USB (alimentation Arduino)
- 1 Ecran LCD
- 1 Convertisseur
- 1 Capteur de courant
- 1 panneau solaire

IV) Etude du panneau solaire :

Afin de mieux comprendre comment un panneau fonctionne voici sa caractéristique.

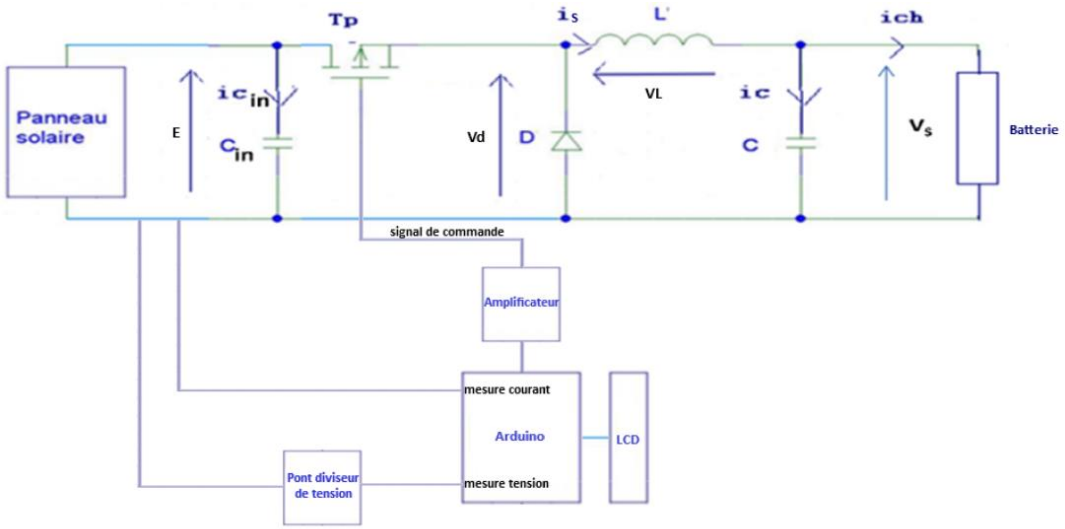


Point de Puissance Maximale

On peut apercevoir plusieurs courbes en fonction de l'ensoleillement représentant l'évolution du courant ainsi que la puissance délivrée par le panneau solaire en fonction de la tension.

On peut voir un point de puissance maximal, c'est ici que nous allons venir nous placer afin de recharger la batterie dans des conditions optimale.

V) Schéma de principe :



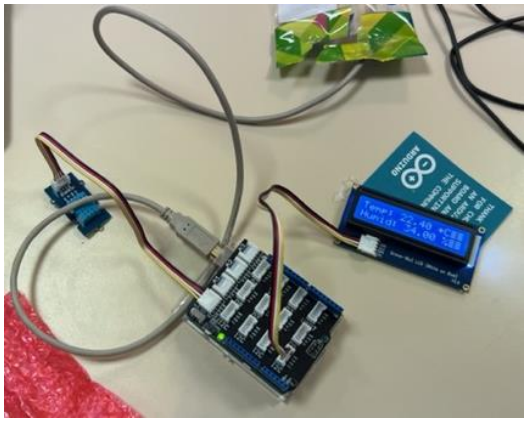
2) Prise en main du matériel

I) Utilisation du capteur DHT11 et de l'écran LCD :

Pour commencer, nous avons réalisé par un montage basique afin de se familiariser avec le matériel.

Ce montage permet à l'Arduino de mesurer un taux d'humidité et une température et l'écrire sur un écran LCD :

Voici comment nous avons branché le capteur ainsi que l'écran LCD:



Voici le code que nous avons réalisé afin que l'écran affiche le taux d'humidité ainsi que la température :

Permet de récupérer les valeurs de température/humidité mesurée par le capteur

Permet l'affichage des valeurs de température et d'humidité sur l'écran LCD.

```
base
#include <Wire.h>
#include "rgb_lcd.h"
#include "DHT.h"

#define broche A0
#define type DHT11
DHT dht(broche, type);


float temp;
float humidite;
int pot,Low=102,Hight=920;

rgb_lcd lcd;

const int colorR = 255;
const int colorG = 0;
const int colorB = 0;

void setup()
{

    // set up the LCD's number of columns and rows:
    lcd.begin(16, 2);

    lcd.setRGB(colorR, colorG, colorB);

    delay(1000);
    Serial.begin(9600);
    dht.begin();
}

void loop()
{
    temp = dht.readTemperature();        // acquisition
    humidite = dht.readHumidity();          // acquisit

    lcd.print("Temp: ");
    lcd.print(temp);
    lcd.println(" *C");
    // set the cursor to column 0, line 1
    // (note: line 1 is the second row, since counting
    lcd.setCursor(0, 1);
    // print the number of seconds since reset:
    lcd.print("Humid: ");
    lcd.print(humidite);
    lcd.println(" %");

}
```

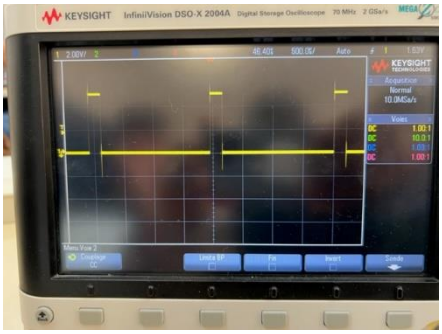
II) Création d'un signal de sortie variable :

Pour réaliser un signal de sortie variable nous avons utilisé un potentiomètre branché sur une entrée analogique de l'Arduino, le potentiomètre réglera le rapport cyclique du signal de sortie.

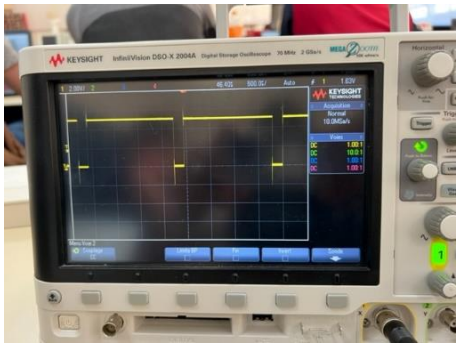
Ce montage nous sera utile par la suite pour de commander notre convertisseur.

Pour vérifier si notre code était correct nous avons utilisé un oscilloscope afin de vision le signal que délivrait notre Arduino.

On peut voir les variations de la tension de sortie lorsque nous faisons varier le potentiomètre :



Ici on a un rapport cyclique de 10%



Ici le rapport vaut 90%

Voici le code que nous avons réalisé :

base

```
#include <Wire.h>
#include "rgb_lcd.h"
#include "DHT.h"

#define broche A0
#define type DHT11
DHT dht(broche, type);


float temp;
float humidite;
int pot,Low=102,Hight=920;

rgb_lcd lcd;

const int colorR = 255;
const int colorG = 0;
const int colorB = 0;

void setup()
{

    // set up the LCD's number of columns and rows:
    lcd.begin(16, 2);

    lcd.setRGB(colorR, colorG, colorB);

    delay(1000);
    Serial.begin(9600);
    dht.begin();
}

void loop()
{
    temp = dht.readTemperature();          // acquisition
    humidite = dht.readHumidity();          // acquisit

    lcd.print("Temp: ");
    lcd.print(temp);
    lcd.println(" *C");
    // set the cursor to column 0, line 1
    // (note: line 1 is the second row, since counting
    lcd.setCursor(0, 1);
    // print the number of seconds since reset:
    lcd.print("Humid: ");
     lcd.print(humidite);
     lcd.println(" %");


     //Potentiomètre
    pot = analogRead(A1);
     float tension = pot/204.0;
     float alpha = pot/1023.0;
     if(pot>Hight)  //Limite +90%
       Serial.println(5);
     else if(pot<Low) //Limite +10%
       Serial.println(0);
     else
       Serial.println(tension);

     Serial.println(alpha);

     analogWrite(3, pot/4);

     delay(100);
}
```

On a branché notre potentiomètre sur la broche A1 importation comprise entre 0 et 1023.

On renvoie l'information du potentiomètre sur le pin de sortie 3 en divisant l'information du potentiomètre par 4 car les sorties sont comprises entre 0 et 255.

3) Mesures avec l'Arduino

I) Mesure de tension via Arduino

Pour la suite nous avons besoin de savoir la tension aux bornes de la batterie ainsi que la tension délivrée par les panneaux, pour cela on a réalisé le code suivant permettant de mesurer une tension avec l'Arduino.

```
#include "rgb_lcd.h"

rgb_lcd lcd;

const int colorR = 255;
const int colorG = 0;
const int colorB = 0;

void setup() {
  lcd.begin(16, 2);
  lcd.setRGB(colorR, colorG, colorB);
  delay(1000);
  Serial.begin(9600);

}

void loop() {
  int tens = analogRead(A0);
  float tension = tens * (5.0 / 1023.0);
  lcd.print("Tension: ");
  lcd.println(tension);
  delay(250);
}
```

On récupère la valeur de la tension aux bornes de l'entrée analogique A0

Ont converti cette valeur en une tension comprise entre 0 et 5V.

Puis on l'écrit sur notre écran LCD.

II) Pont diviseur de tension :

Maintenant que nous savons mesurer une tension, il faut adapter la tension du panneau solaire et de la batterie entre 0 et 5V afin de pouvoir la mesurer avec l'Arduino.

Pour cela nous avons utilisé un pont diviseur de tension, nous savons qu'il nous faut au maximum 5V quand a tension du panneau est maximale (environs 21V). Il suffit de fixer une valeur de résistance et déterminer l'autre.

Voici le schéma ainsi que les calculs réalisés :

Ici, V_{in} = 21V ; V_{out} = 5V ; R_1 = 9,81kΩ (valeur choisit arbitrairement).

On sait que :

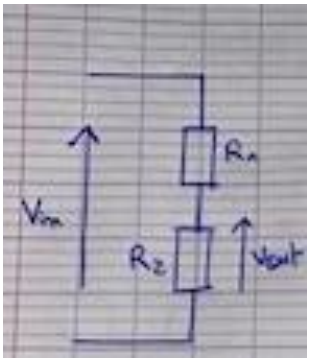
$$V_{out} = \frac{V_{in} \cdot R_2}{R_1 + R_2}$$

D'où,

$$R_2 = \frac{V_{out} R_1}{V_{in} - V_{out}}$$

$$R_2 = \frac{5 \cdot 9{,}81 \cdot 10^3}{21 - 5} = 3065\ \Omega$$

R_2 = 3065Ω



III) Mesure du courant via Arduino :

Pour donner suite à ça nous avons voulu utiliser le capteur de courant afin de réaliser les mesures avec l'Arduino mais celui-ci nous donnait des valeurs incohérente et inexploitable pour la suite, nous avons donc étudié ce capteur afin de savoir comment il fonctionnait.

Pour ce faire, on a réalisé un montage avec une rhéostat permettant de faire varier le courant qui traversait la résistance.

Nous avons aussi utilisé un oscilloscope pour comprendre le fonctionnement du capteur afin d'en traduire le programme à développer pour déterminer le courant traversant la résistance.

Voici notre montage :

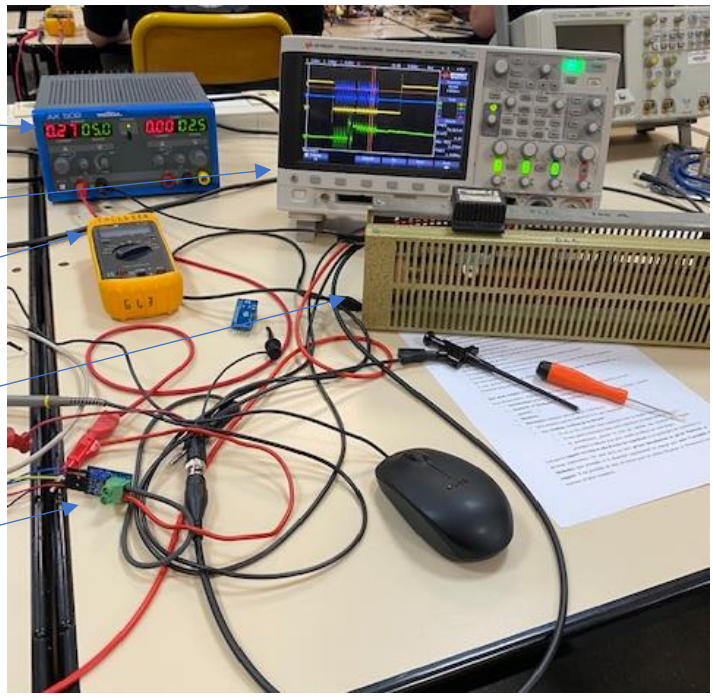
Alimentation

Oscilloscope

Ampèremètre

Rhéostat

Capteur de courant



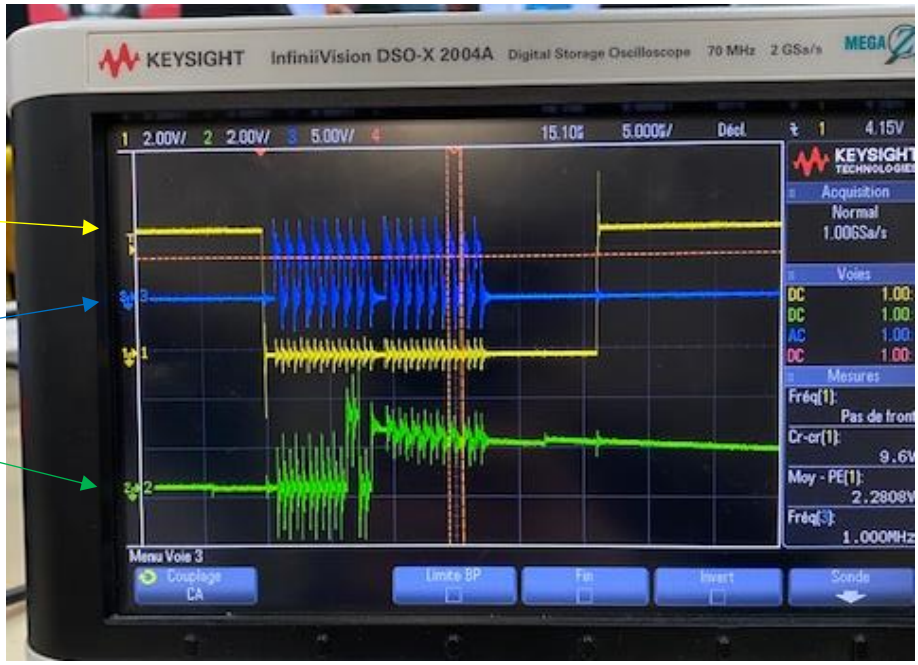
Il faut savoir que le protocole de communication du capteur de courant est en SPI, ceci fonctionne de la manière suivante :

Il a besoin d'un signal que l'on doit générer lui disant quant-il doit réaliser les mesures ici c'est Chip Select, un signal d'horloge ici le Serial Clock générer par l'Arduino et enfin le capteur nous retourne le signal en vert sur l'oscilloscope juste en dessous :

Chip select (généré par l'Arduino)

Serial Clock (généré par l'Arduino)

Signal délivré par le capteur

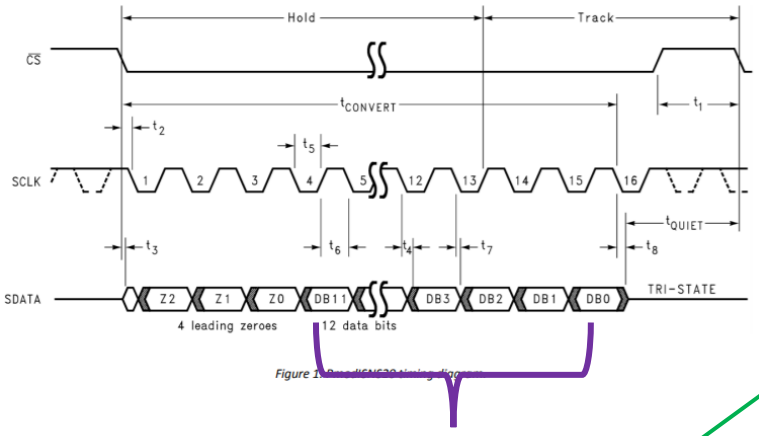


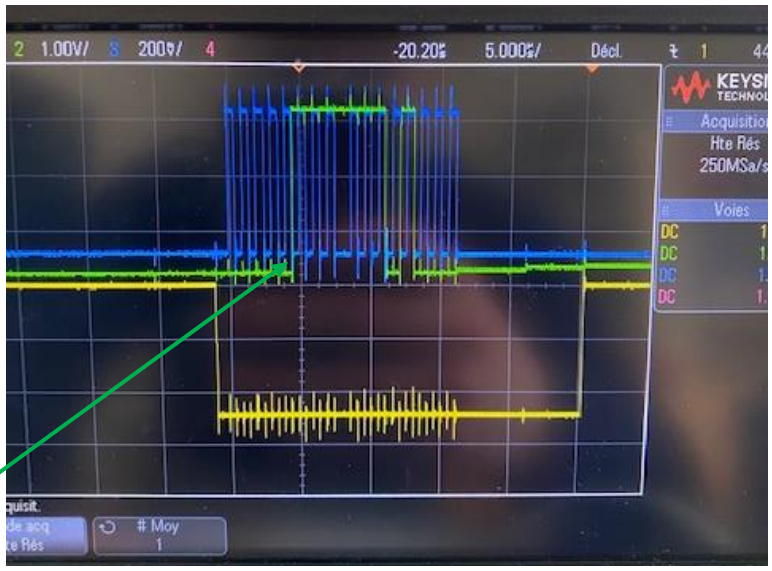
Dans la documentation technique il y avait une formule donnée permettant de convertir le signal mesuré par le capteur en milliampère, en utilisant la formule ci-dessous on obtenait des valeurs totalement incohérentes avec la réalité.

$$I_{mA} = \frac{1000}{89.95}(ADC_{VALUE} - 2048)$$

Nous avons alors trouvé une solution qui était de comprendre comment le capteur fonctionné et de tracer sa caractéristique afin de déterminer l'équation pour convertir les valeurs délivrées par le capteur en un courant.

Pour cela on a décortiqué le signal délivré par le capteur et les informations donné par la documentation :





A droite on peut voir le signal délivré par le capteur pour une certaine valeur du courant.

A gauche on voit comment le capteur fonctionne.

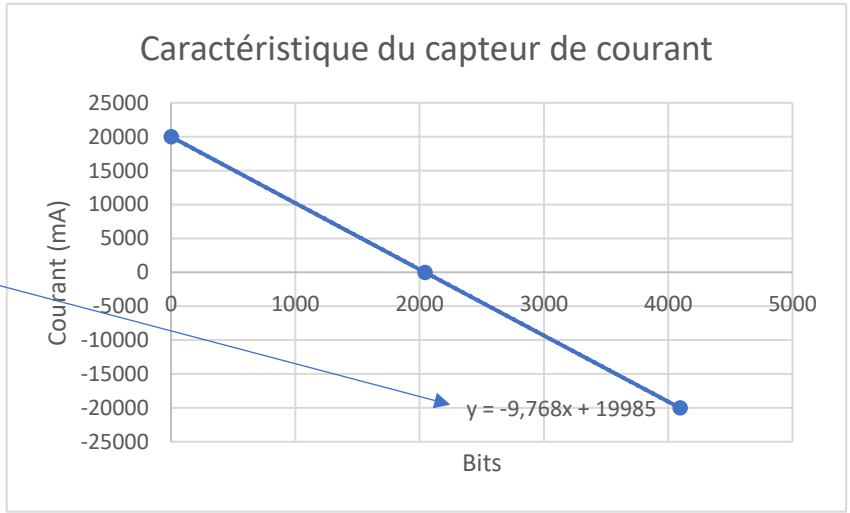
Le capteur génère 12 data bits soit 2^{12} = 4096 valeurs.

En réalisant les essais on a remarqué :

- Une valeur de 2048 pour un courant de 0A.
- Une valeur de 1800 pour un courant de 2,4A.
- Nous avons réalisé plusieurs mesures afin d'être le plus précis possible.

Grace à ça nous avons pu tracer la caractéristique car celle-ci est linéaire.

Grace à ça nous avons déterminé la nouvelle équation permettant la conversion.



Voici le code que nous avons développé afin de mesurer le courant à l'aide de la carte Arduino :

Ancienne équation donnée par la documentation.

Conversion des données du capteur en bits.

Équation déterminée avec les valeurs relevées.

Écriture du courant sur l'écran LCD.

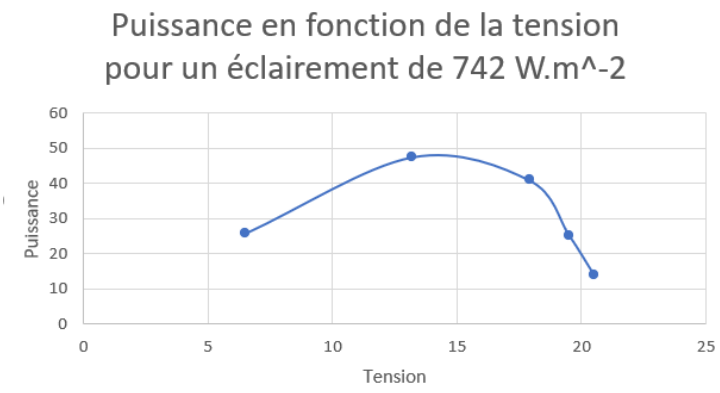
```
void loop()
{
 // milli_amps = (10000 * (((MSB<<8) | LSB) - 2048)) / 899; //formule donnée dans la documentation
  int count=0;
  digitalWrite(CS, LOW);// activation de la ligne CS
  digitalWrite(CSMesure, LOW);
  MSB=SPI.transfer(0x00); // récupération des bit de poids forts
  LSB=SPI.transfer(0x00); // récupération des bit de poids faibles
  digitalWrite(CS, HIGH); // désactivation de la ligne CS
  digitalWrite(CSMesure, HIGH);

  int m = ((MSB<<8) | LSB);
  float courant = 1000.0*((-m+2043.0)*(1.89/155.0));
  lcd.setCursor(0,0);
  lcd.print("Courant: ");
  lcd.setCursor(10,0);
  lcd.print(courant);
  Serial.print("LSB=");
  Serial.println(LSB);
  Serial.print("MSB=");
  Serial.println(MSB);
  Serial.print("en bits=");
  Serial.println(((MSB<<8) | LSB));
  Serial.print("courant=");
  Serial.println(courant);



  delay(500);
}
```

IV) Mesure de puissance via Arduino :

Maintenant que nous savons mesurer un courant et une tension à l'aide de la carte Arduino nous pouvons alors mesurer une puissance.

Nous avons donc mesuré la puissance en fonction de la tension délivrée par le panneau solaire.

Pour cela nous avons branché le panneau solaire aux bornes d'une charge résistive variable afin de faire varier le courant, voici le résultat obtenu :



Ces mesures on étaient mesuré pour un ensoleillement de 742 $W.m^{-2}$ que nous avons pu relever à l'aide d'un pyranomètre.

Sur celui-ci était indiqué 14,45 µV/($W.m^{-2}$), on a mesuré une tension de 10,7 mv ce qui nous donne bien un ensoleillement de 742 $W.m^{-2}$.

V) Comparaison des mesures avec la théorique :

Pour savoir si nos mesures étaient cohérentes avec la théorie nous avons comparé nos mesures avec les caractéristiques du panneau photovoltaïque.

CARACTERISTIQUES ELECTRIQUES				
PW6-110		Configuration 12 V		
Puissance typique	W	90	100	110
Puissance minimale	W	85	95,1	105,1
Tension à la puissance typique	V	16,4	16,7	17,2
Intensité à la puissance typique	A	5,5	6,0	6,4
Intensité de court circuit	A	6,1	6,5	6,9
Tension en circuit ouvert	V	21,1	21,5	21,7
Tension maximum du circuit	V	1000V DC		
Coefficients de température		a = +2,085 mA/°C ; β = -79 mV/°C ; γ P/P = - 0,43 % /°C		
Spécifications de puissance à 1000 W/m² : 25°C : AM 1,5				

On peut voir que la tension en circuit ouvert est d'environ 21V comme ce que nous avons pu mesurer.

Pour ce qui est de la puissance théorique les caractéristiques constructeur nous donne une puissance de 90W atteinte pour une tension d'environs 16V sous un ensoleillement de 1000 $W.m^{-2}$.

Sur notre graphique on peut voir que notre pic de puissance est bien atteint pour une tension avoisinant les 16V mais notre puissance maximale est bien inférieure au 90W annoncé ce qui est normal car lors des mesures nous avons mesuré un ensoleillement de 740W.m-2.

4) Convertisseur DC/DC :

I) Présentation du convertisseur :

Comme la tension de la batterie n'est pas égale à la tension du panneau solaire nous devons utiliser un convertisseur DC/DC qui va permettre d'abaisser la tension de sortie tout en forçant le panneau solaire à travailler à sa puissance maximale.

Le convertisseur utilisé est un hacheur série, ce convertisseur utilise un thyristor qui va permettre de laisser passer le courant ou non.

En réalité le convertisseur découpe la tension d'entrée pour abaisser la valeur moyenne de celle-ci.
Or comme la tension de sortie sera découpé elle sera donc variable et non continue.

Pour pallier ce problème il faut alors ajouter une inductance et une capacité qui vont jouer un rôle de lissage afin d'obtenir une tension exploitable pour la batterie.

Voici le schéma que nous avons déterminé :

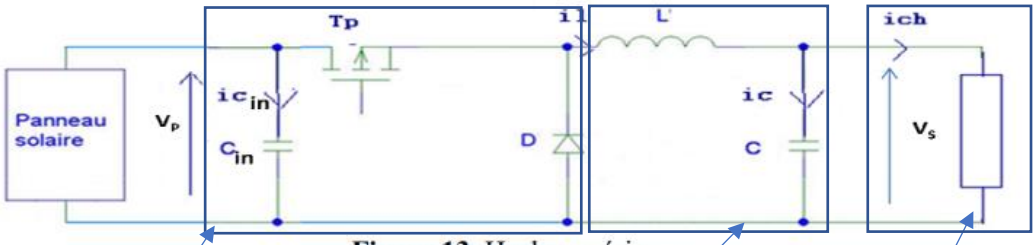


Figure 13. Hacheur série.

Convertisseur DC/DC — Lissage — Charge (Batterie)

II) Détermination des composants :

Pour lisser il faut utiliser un condensateur et une bobine, la bobine permettra de lisser le courant et le condensateur de lisser la tension.

Pour cela il faut déterminer leurs valeurs, nous nous sommes alors imposé le cahier des charges suivant :

- Une ondulation maximale de courant : $\Delta I_{Lmax} = 120$ mA
- Une ondulation maximale de tension : $\Delta I_{Cmax} = 50$mV

Voici les démarches que nous avons entreprises pour déterminer les valeurs de l'inductance ainsi que de la capacité :

Premièrement nous faisons l'hypothèse que le courant i_s circulant dans l'inductance est toujours positif.

- Si T_r conduit alors D (diode) est bloqué donc la tension aux bornes de la diode (V_D) est égale à E.
- Si T_r est bloquée alors D conduit donc la tension aux bornes de la diode (V_D) est nulle.

Voici un schéma pour mieux comprendre :

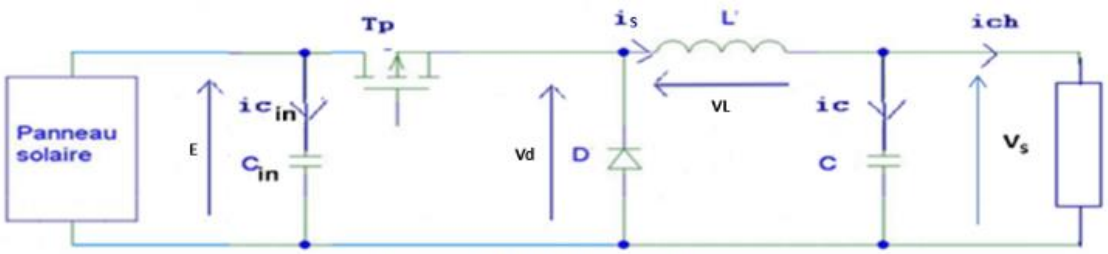
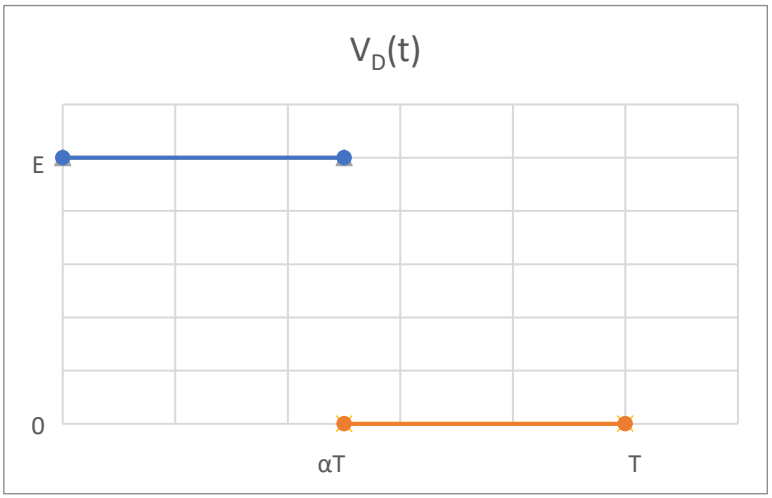


Figure 13. Hacheur série.

On en déduit donc :

- Sur [0 ; αT] , T_r conduit.
- Sur [αT ; T] , T_r est bloqué.

Voici l'allure de la tension aux bornes de la diode :



On en déduit donc que la valeur moyenne de la tension aux bornes de la diode est la suivante :

$\langle V_D \rangle = \alpha E$

En faisant une loi des mailles sur le graphique plus haut on obtient l'équation suivante :

$\langle V_D \rangle - V_L - V_S = 0$
On sait que $V_L = L di_s(t)/dt$
⇨ $\langle V_D \rangle = \langle L di_s(t)/dt \rangle + \langle Vs \rangle$

Mais, en régime permanent, $i_s(t) = i_s(t + T)$

⇨ $\langle L di_s(t)/dt \rangle = L[i_s(t+T) - i_s(t)] / Tc = 0$

Donc,

$\langle V_D \rangle = \langle Vs \rangle = \alpha E$

Calcul de Δi_s sur [0 ; αT] :

On cherche à déterminer le courant aux bornes de l'inductance pour cela nous avons une seconde loi des mailles:

E – VL – VS = 0

⇨ E = VL + VS
⇨ E = Ldis/dt + αE
⇨ $Ldi_s/dt = E - \alpha E$
⇨ $di_s/dt = (E - \alpha E)/L$
⇨ $\Delta i_s/\alpha T = (E - \alpha E)/L$
⇨ $\Delta i_s = (E - \alpha E)\alpha/(L.f) = \alpha E(1-\alpha)/(L.f)$

Il nous manque plus qu'à déterminer L :

$L = \alpha E(1-\alpha)/\Delta i_s.f$

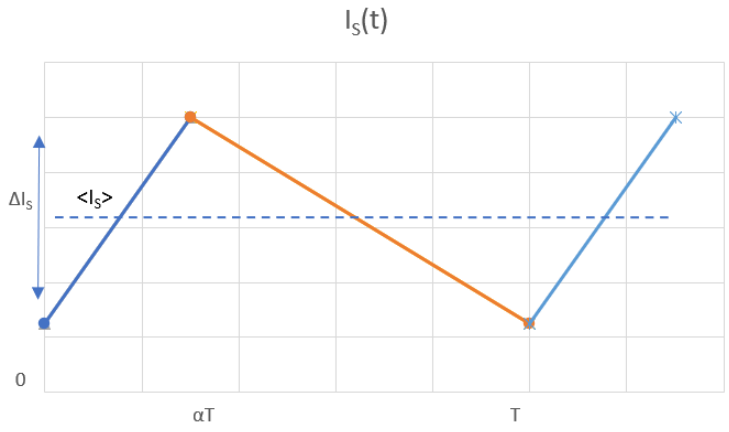
L'ondulation est maximale pour un rapport cyclique α = 0.5, E = 12V, la fréquence sera celle délivrée par l'Arduino que nous verrons juste après f= 62,5kHz et Δi_s = 120 mA.

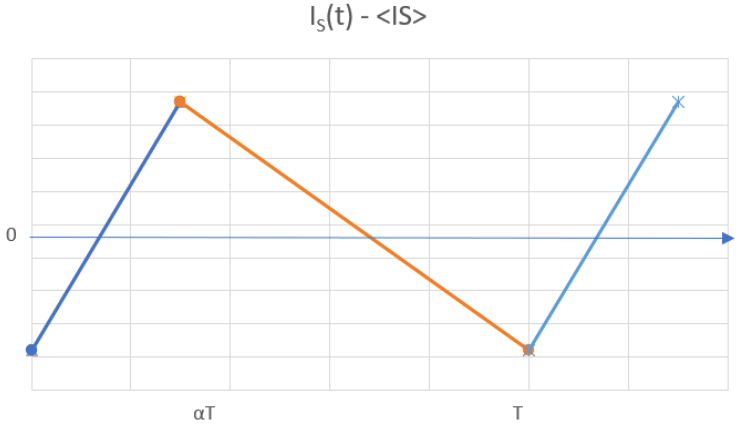
⇨ $L = 0.5*12*0.5/(120*10^{-3}*62.5*10^{3}) = 0.4$ mH .

Calculons maintenant ΔV_s :

En faisant une loi des nœuds on obtient la relation suivante :

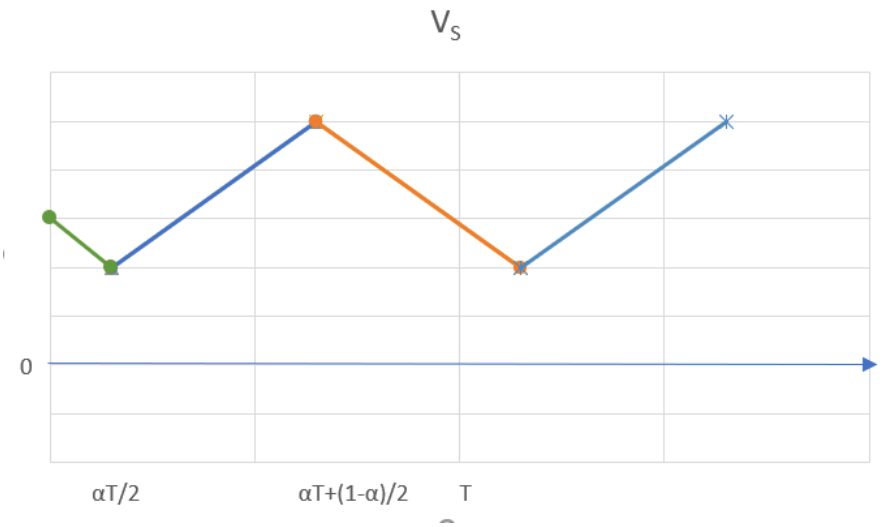
⇨ $i_s = \langle i_s \rangle + CdV_s/dt$
⇨ $CdV_s/dt = i_s - \langle i_s \rangle$





Comme $CdV_S/dt = i_s - \langle i_s \rangle$: - quand $i_s > \langle i_s \rangle$ alors V_S est croissant.

- quand $i_s < \langle i_s \rangle$ alors V_S est décroissant.



On voit graphiquement que :

⇨ $\Delta V_s = V_S max - V_S min$

De la relation précédente on en déduit :

⇨ $V_S = 1/C.(i_s - \langle i_s \rangle).dt$

D'où :

⇨ $\Delta V_s = 1/C \int (i_s - \langle i_s \rangle)dt$ (borne d'intégration de $[\alpha T/2 ; \alpha T + ((1-\alpha)/2)T]$

On remarque que l'on cherche à calculer l'aide d'un triangle, donc le calcul de l'intégrale revient à ça :

$\Delta V_s = 1/C(T/2 * ½ * \Delta i_s/2) = T*\Delta i_s/(8C)$

Or,

⇨ $\Delta i_s = \alpha E(1-\alpha)/(L.f)$

D'où,

⇨ $\Delta V_s = 1/8(\alpha E(1-\alpha)/(L.f^2.C))$

On peut enfin en déduire la valeur du condensateur :

⇨ $C = 1/8(\alpha E(1-\alpha)/(L.f^2.\Delta V_s))$

⇨ $C = 1/8(0.5*12*0.5/(0.4*10^{-3}*62500^2*50*10^{-3})) = 4.7\ \mu F$

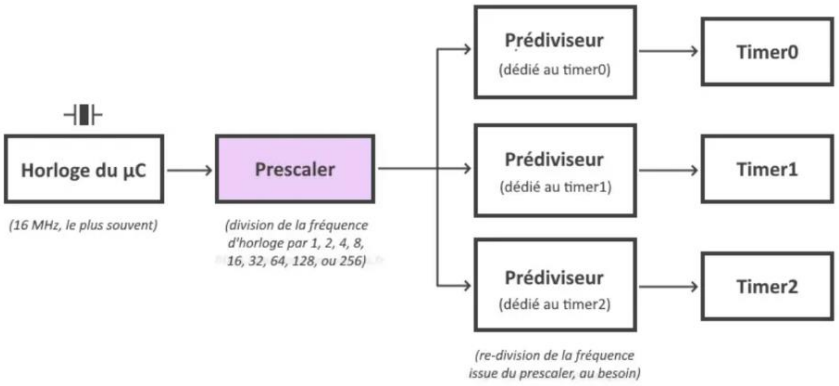
III) Création du signal de commande avec Arduino :

- Nous avons réglé le PRESCALER pour une division par 1. Ainsi, les 16 MHz du quartz qui cadence l'ATmega328P seront divisés par 1, ce qui nous donnera une fréquence toujours égale à 16MHz en sortie du PRESCALER (interne au microcontrôleur ATmega328P).

- Nous avons utilisé le TIMER 2 car il a une grande résolution. En effet, celui-ci peut diviser de 0 à 256 par puissance de 2.

- Nous avons réglé le PREDIVISEUR du TIMER 2 pour une division par 256. Ainsi, la fréquence qui cadencera le TIMER 2 sera 256 fois plus lente que celle sortant du PRESCALER, qui pour rappel était de 16 MHz. On obtient alors une fréquence en sortie de 62500 Hz.

  Le schéma ci-dessous nous indique les étapes de divisions pour avoir une fréquence de 62,5KHz.

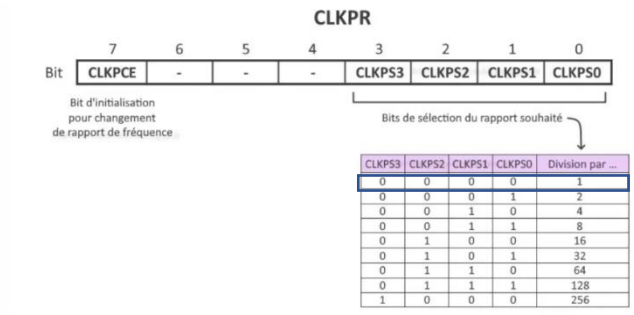


Pour faire ces divisions fréquentielles nous avons utilisé les registres.

Le registre CLKPR pour le PRESCALER est le bit « CLKPCE » qui permet d'indiquer au microcontrôleur que l'on souhaite modifier la valeur du PRESCALER.

Une fois cette indication faite, il nous a suffi juste de renseigner les bits CLKPS3, CLKPS2, CLKPS1, et CLKPS0 au microcontrôleur, pour spécifier le rapport de division souhaité qui est égale à 1 pour nous.

Ainsi, selon la valeur de ces bits, la fréquence en sortie du PRESCALER sera égale à la fréquence d'horloge en entrée, divisée par le rapport choisi.



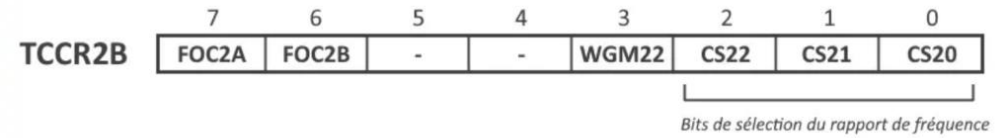
Maintenant que nous avons réglé le PRESCALER à une division par 1, nous allons voir le prédiviseur du TIMER 2 à régler.

À peu de choses près, il s'agit de la même chose. Mais alors que le PRESCALER ralentit « tout » le microcontrôleur, le prédiviseur, quant à lui, ne ralentit que le TIMER 2 auquel il est associé.

Cependant, le prédiviseur est « alimenté » avec la fréquence du PRESCALER.
C'est à dire que la fréquence du timer2 sera en fait égale à la fréquence d'horloge du microcontrôleur, divisée par le rapport de division du prescaler, puis encore divisée par le rapport du prédiviseur associé au TIMER 2.

Le rapport de division souhaité est quant à lui simplement stocké dans un registre particulier qui est le TCCR2B.

Le prédiviseur du TIMER 2 :



CS22	CS21	CS20	Conséquence
0	0	0	Timer arrêté
0	0	1	Fréquence / 1
0	1	0	Fréquence / 8
0	1	1	Fréquence / 32
1	0	0	Fréquence / 64
1	0	1	Fréquence / 128
1	1	0	Fréquence / 256
1	1	1	Fréquence / 1024

Voici le code que nous avons établie pour générer le signal souhaité avec l'Arduino :

On initialise la sortie 11, c'est sur celle-ci que nous allons créer le signal de sortie.

On règle notre PRESCALER et PREDIVISEUR afin d'avoir un signal de sortie de 62500 Hz.

Ici, on vient créer notre signal qui sera réglable grâce à un potentiomètre branché sur l'entrée analogique A0 de notre carte.

```
void setup () {
Serial.begin(9600);
pinMode (11, OUTPUT) ;

bitSet (TCCR2A, COM2B1) ;//On met le bit CoM2B1 du registre TCCR2B à 1
bitClear (TCCR2A, COM2B0) ;//On met le bit CoM2B0 du registre ICCR2B à 0
//OCR2B = 85;
//Fonctionnement FAST PWM du Timer?
bitClear (TCCR2B, WGM22) ;
bitSet (TCCR2A, WGM21); //Ici les bits WGM21 et WGM20 sont à cheval sur le registre ICCR2A
bitSet (TCCR2A, WGM20) ;
//Prédiviseur à 256 (16MGHz / 256=62500 Hz) du Timer 2
bitClear (TCCR2B, CS22) ;
bitClear (TCCR2B, CS21) ;
bitSet (TCCR2B, CS20) ;
}
void loop () {
int pot= analogRead (A0) ;
int alpha= map (pot, 0, 1023, 0, 255) ;
if (alpha < 255*0.1)
analogWrite(11,255*0.1);
else if (alpha > 255*0.9)
analogWrite(11,255*0.9);
else analogWrite (11, alpha) ;
}
```

5) Algorithme pour commander le convertisseur :

I) Détermination du MPPT :

Le MPPT de l'anglais Maximum Power Point Tracking signifie détection de la puissance maximale du champ photovoltaïque tant que la batterie n'est pas chargée.

La détermination du bon rapport cyclique pour le maximum de puissance délivré par le panneau solaire sera réalisée à l'aide d'un algorithme fournit dans le cahier des charges.

Nous allons donc essayer de comprendre et le traduire en langage Arduino afin de recharger la batterie dans les meilleures conditions.

II) Traduction de l'algorithme en langage de programmation :

Voici l'algorithme fournit par le cahier des charges :

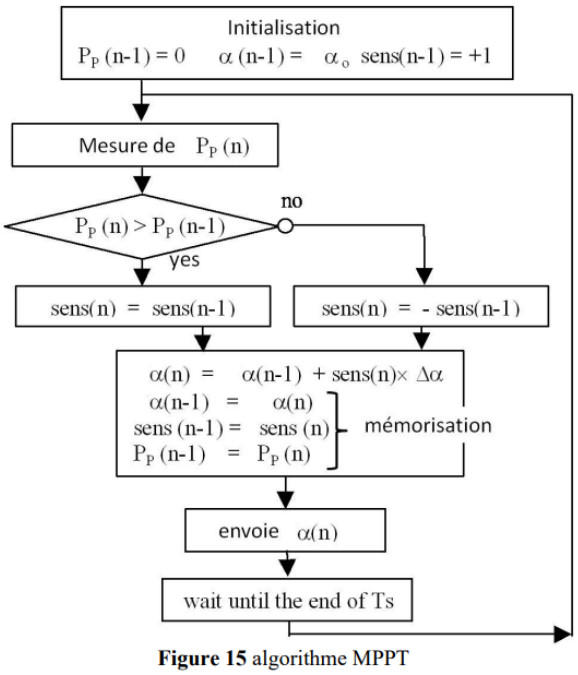


Figure 15 algorithme MPPT

Nous allons essayer de décortiquer l'algorithme :

Premièrement il faut lister le nombre et le type des différentes variable à utiliser dans le programme :

- Pp[] : Tableau qui va stocker les différentes puissances.
- α[] : Tableau qui va stocker les différentes valeurs du rapport cyclique.
- Sens[] : Tableau qui va venir ajuster les valeurs du rapport cyclique.

On peut voir lors de la phase d'initialisation que Pp est initialisé à 0 , α à α_0 et sens égal à 1.

Une première mesure de puissance est réalisée est stocké dans le tableau Pp.

Puis on vient vérifier si cette puissance est supérieure à la puissance mesurée précédemment :

SI OUI	SI NON
On conserve la valeur précédente de sens en la stockant une nouvelle fois.	On vient stocker dans Sens la valeur opposée de Sens déterminée précédemment.

Ensuite, on stocke la valeur de alpha qui sera égale au α déterminé précédemment plus la valeur de Sens actuelle multipliée par la variation de α.

On rentre ensuite dans une phase de mémorisation, cela consiste à remplacer les valeurs précédentes avec les valeurs actuelle.

Enfin, on adapte notre signal de sortie en fonction du nouveau rapport cyclique α.

Voici le code que nous avons déterminée avec le décorticage de l'algorithme :

Calcul de la tension, du courant et de la puissance.

On vient vérifier si la puissance actuelle est plus grande que la puissance mesurée précédemment.

On détermine le nouveau rapport cyclique.

Puis on mémorise nos valeurs pour le prochain passage.

```
void loop() {
  int tens = analogRead(A0);
  digitalWrite(CS, LOW);
  MSB=SPI.transfer(0x00);
  LSB=SPI.transfer(0x00);
  digitalWrite(CS, HIGH);

  int m = ((MSB<<8) | LSB);
  float courant = 1000*((-m+2043)*(1,89/155));
  float tension = tens * (5.0 / 1023.0);
  float p = courant * tension;

  //Determination de α

  if(Pp[1]>Pp[0])
  {
    sens[1] = sens[0];
  }
  else
  {
    sens[1] = -sens[0];
  }

  //Nouveau alpha

  alpha[1] = alpha[0] + sens[1]*(fabs(alpha[1]-alpha[0]));

  //Mémorisation

  alpha[0] = alpha[1];
  sens[0] = sens[1];
  Pp[0] = Pp[1];

  //On envoie le nouveau alpha..

}
```

6) Conclusion :

Ce projet nous a grandement apporté, il nous a permis de mieux comprendre le fonctionnement d'un panneau solaire, l'importance d'un hacheur série ainsi que la programmation via Arduino.
Le fait de déterminer des composant nous a également grandement appris en effet nous avons pu allier nos cours de génie électrique afin de comprendre et de calculer les composants à utiliser.

Nous n'avons malheureusement pas eu le temps de finir notre projet.
Il nous manquait une séance pour réaliser les tests afin de savoir si l'algorithme déterminant le rapport cyclique était correct, si le signal délivré par l'Arduino permettait bien de commander le hacheur série et déterminer les composants à utiliser pour amplifier notre signal de commande en sortie de l'Arduino.

Enfin nous souhaitons remercier Mr Thierry PERISSER ainsi que l'Université Paul Sabatier qui sans eux ce projet n'aurait pas pu être réalisable.