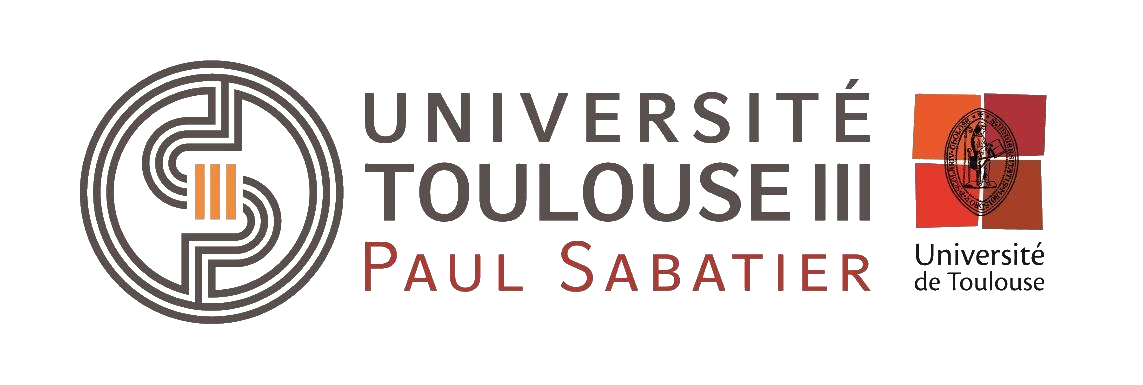
**Bureau d’étude**

**L3 EEA REL**



Tables des Matières

[Introduction 3](#_Toc3593)

[Matériels 3](#_Toc8442)

[Affichage de l’écran LCD 5](#_Toc15592)

[Méthodes 5](#_Toc3277)

[Résultats 6](#_Toc3133)

[Conclusion 7](#_Toc29188)

[La tension du panneaux solaire 8](#_Toc16408)

[Méthodes 8](#_Toc5261)

[Résultats 8](#_Toc25678)

[Conclusion 9](#_Toc12833)

[Le courant du panneaux solaire 10](#_Toc25624)

[Méthodes 10](#_Toc17359)

[Résultats 12](#_Toc11270)

[Conclusion 13](#_Toc25790)

[Commande du hacheur 15](#_Toc4138)

[Introduction 15](#_Toc20506)

[Partie 1 : PWM 15](#_Toc16606)

[Méthode 15](#_Toc14151)

[Résultats 17](#_Toc10911)

[Conclusion 17](#_Toc31437)

[Partie 2 : Amplificateur 18](#_Toc16845)

[Méthode 18](#_Toc25508)

[Résultats 19](#_Toc331)

[Conclusion 19](#_Toc10174)

[MPPT 21](#_Toc23390)

[Capteur de température 23](#_Toc7172)

[Capteur de luminosité 24](#_Toc5686)

[Annexe 25](#_Toc18210)

[Sources 25](#_Toc6661)

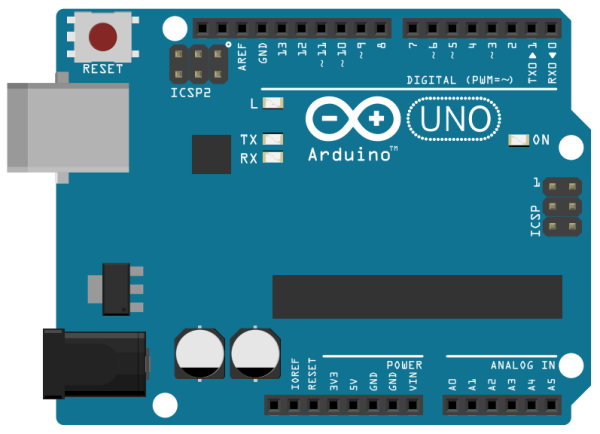
# Introduction

L’objectif de départ de ce bureau d’étude était de réaliser un chargeur de batterie alimenté par un panneau solaire, mais nous avons décider de nous concentrer sur le suivit du point maximal de puissance (MPPT) du panneaux solaire. Pour réaliser ce bureau d’étude nous avons utilisé une carte Arduino qui va s’occuper de récupérer, gérer et analyser les différentes informations afin de trouver ce point de fonctionnement.

# Matériels

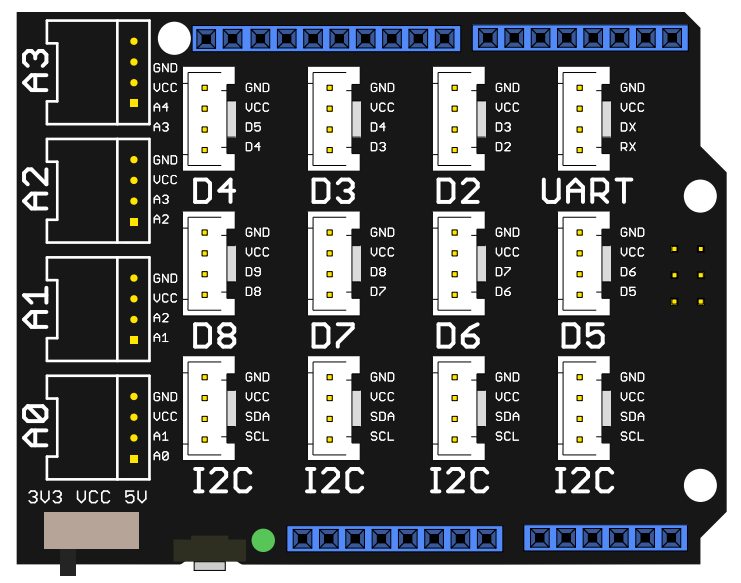
Pour la réalisation du projet nous disposons de plusieurs panneaux solaire se situant sur le toit du bâtiment mais pour notre système nous en utiliserons qu’un seul.

Pour la gestion du programme nous disposons d’une carte Arduino Uno en figure 1.



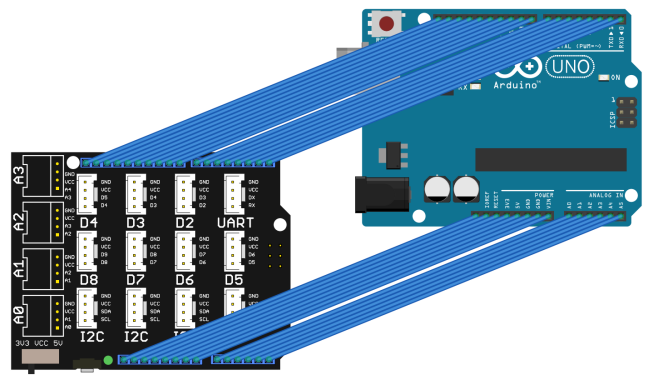
1. : représentation graphique de la carte Arduino

Pour simplifier le câblage et pour pouvoir utiliser plus de modules nous disposons d’une extension qui se superpose à la carte Arduino, il s’agit d’une carte Shield V2 visible en figure 2.



1. : Schéma de la carte Shield

Le raccordement entre les deux composants est très simple il suffit seulement de branche le shield sur la carte Arduino car il possède exactement les mêmes connecteurs.



1. : Schéma de raccordement de la carte Arduino et le Shield

Pour réaliser le programme et la réalisation pratique du projet nous avons utiliser les documentations officiel sur la carte Arduino [1] et le Shield V2 [2].

# Affichage de l’écran LCD

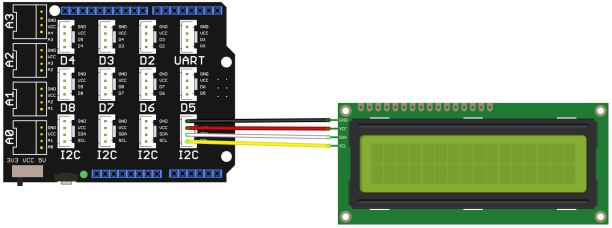
Dans notre système, nous avons afficher des informations en continu du panneaux solaire tel que la tension et le courant maximale qu’il peut fournir . Pour ce faire nous avons utiliser un afficheur LCD qui peut afficher 16 caractère sur 2 lignes, il affiche du texte de couleur noir sur fond jaune. L’afficheur se connecte en I2C ce qui simplifie grandement le câblage mais qui nécessite d’utiliser une libraire

## Méthodes

Pour afficher du texte sur l’écran il nous faut raccorder l’afficheur à la carte Arduino, étant donné que nous avons la carte d extension Shield, nous avons simplement représenté le raccordement au niveau de l’extension. Puisque que l’afficheur se raccorde en I2C, nous utilisons que 4 broches comme indiqué dans le tableau de la figure 4, la figure 5 représente le schéma de câblage électrique des deux modules.

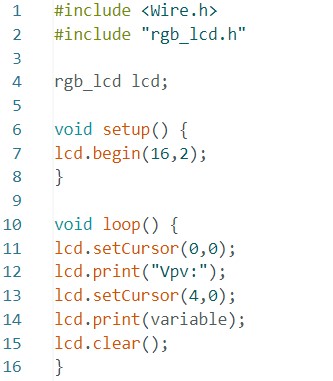
|  |  |  |
| --- | --- | --- |
| Port Arduino | Broche | Couleur |
| GND (I2C) | GND | Noir |
| Vcc (I2C) | Vcc | Rouge |
| SDA (I2C) | SDA | Blanc |
| SCL(I2C) | SCL | Jaune |

1. : Tableau de raccordement entre l’afficheur et le shield



1. : Schéma de raccordement entre l’afficheur et le shield

Pour ce qui en est de la programmation le principe de la communication I2C est de réduire le nombre de fils et d’utiliser un bus de communication. Nous devons donc ajouter une librairie qui a déjà était créer pour ce système d’affichage tel que la librairie «rgb\_lcd» visible à la ligne 2 de la figure 6. De plus il faut ajouter le libraire Arduino «Wire.h» qui est faite pour l’utilisation des communications en bus I2C.



1. :Programme de l’affichage sur l’écran LCD

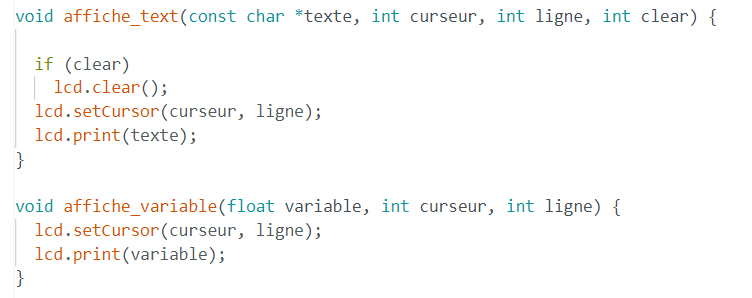
Dans le programme nous pouvons voir la fonction «lcd.» qui est la simplification de «rgb\_lcd» qui est plus longue à écrire, ce paramétrage est visible ligne 4.

Pour pourvoir afficher du contenu sur l’afficheur LCD il faut définir les caractéristiques de l’afficheur par la fonction «lcd.begin» en définissant le nombre de caractères possible est le nombre de linges. Il faut également définir la position du curseur, ligne 11, on définit le curseur sur première li gne au premier caractère. Puis avec la fonction «print», on affiche du texte en le mettant entre guillemet, si on veut afficher la valeur d’une variable, il faut seulement mettre le nom de la variable. Si on veut changer le texte sur l’afficheur il faut l’effacer puis réaffirmer le texte ou la variable souhaité.

## Résultats

Au premier test, l’afficheur n’a afficher que la moitier du texte souhaité, ce problème provenais du fait que la fonction loop afficher puis effacer le texte à la même fréquence que horloge de la carte Arduino, mais l’afficheur n’a pas le temps d’écrire tout les caractère. Pour résoudre ce soucis il à fallu rajouter une ligne pour mettre en pause l’affichage de quelques secondes.

Mais une fois ce problème résolut, l’affichage est clair et le texte est suffisamment lisible, les lettres qui sont en majuscules sont facilement differentiable de celles en minuscules. Le premier programme réaliser pour afficher du texte et des variables étant long et peu pratique pour afficher en continu les différents textes et variables à la suite, nous avons décider de créé deux fonctions externe à la fonction void loop afin de simplifier et alléger la fonction principale visible sur la figure7.



1. : fonction permettant d’afficher du texte et des variables

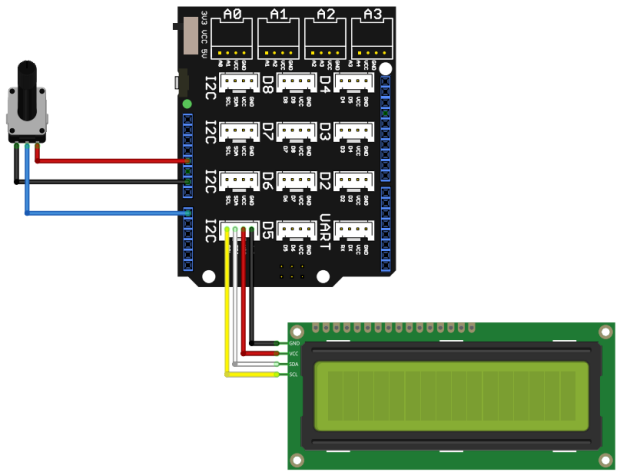
## Conclusion

L’affichage nous permet pour la suite de l’évolution du projet, voir les valeurs reçu par les différents capteurs tel que la valeur de la tension ou le courant. Grâce au deux fonctions d’affichage nous pouvons facilement voir le programme que nous somme modifiant sans chercher parmi les lignes de codes ce qui permet l’affichage et ce qui nous intéresse.

# La tension du panneaux solaire

## Méthodes

Pour lire la tension au borne du panneaux solaire nous allons utiliser les ports d’entrée analogique de la carte Arduino. Dans un premier temps nous avons fait un programme de test avec une résistance variable pour verifier si l’on réceptionner une tension. Pour ce faire nous avons alimentée un potentiomètre qui a pour sortie l’entré A0, le schéma de cable est visible sur la figure 8.



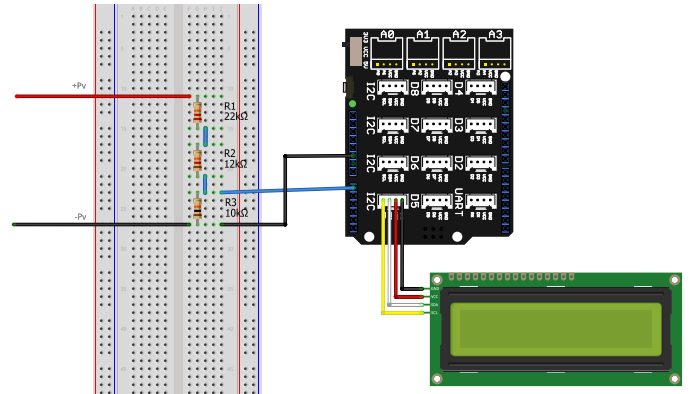
1. : Schéma de mesure de tension en A0

Le panneaux solaire fournit en permanence une tension qui va nous permettre d’alimenter la batterie, pour calculer la tension maximale qui peut être produite par le panneaux solaire nous avons mit en place un pont diviseur de tension.

Le panneaux solaire peut fournir une tension maximale de 22V mais la carte Arduino ne peut réceptionné maximum 5V au borne de ses ports analogique, nous devons donc abaisser la tension pour que la tension maximale du panneaux solaire soit perçu comme 5V pour la carte Arduino. D’où le système de pont diviseur de tension.

## Résultats

Pour mettre en place ce système, il faut utiliser la formule est Vs=Vcc\*(R2/(R1+R2)).Où R1 est la résistance en amont du système alimenté par la tension Vcc et R2 celle reliée à la masse. Entre les deux résistance, nous tirons un fil jusqu’à l’entrée A0 de la carte Arduino. Dans notre cas, nous avons une tension Vcc de 22V et une tension Vs qui doit être de 5V, il nous faut donc un rapport de réduction de 0.227. Nous avons choisit une résistance de valeur de 10kΩ et par le calcul nous trouvons qu’il faut une résistance de 34kΩ pour R2, mais n’ayant pas de résistance de cette valeur, nous avons choisit deux résistance en série de 12kΩ et 22kΩ. En figure 9, nous avons le schéma électrique du pont diviseur de tension raccorder au Shield de la carte Arduino, la valeur de le tension est afficher sur l’écran LCD.



1. : Schéma du pont diviseur de tension du panneaux solaire

## Conclusion

Après un raccordement au panneaux solaire nous avons effectuer un programme qui permet d’afficher le tension reçu en volt directement sur l’afficheur LCD et prenant soit d’annuler le rapport de réduction afin d’avoir la vrai valeur du Panneaux Solaire.

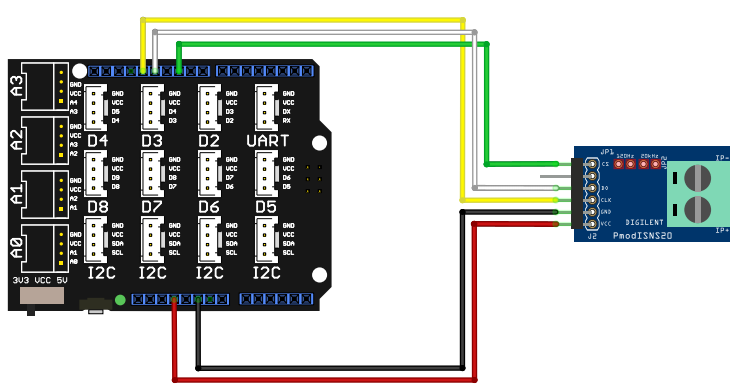
# Le courant du panneaux solaire

## Méthodes

Pour la mesure du courant nous utilisons un capteur de courant, ce dernier nous permet de brancher d’un coté le panneaux solaire et de l’autre le raccorder à la carte Arduino par des fils. Ce capteur fonctionne grâce à une communication en Bus SPI, il nous faut donc effectuer une récupération des données. Pour ce faire nous devons alimenter le capteur en 3.3V et raccorder une masse GND comme indiquer sur la documentation du constructeur et raccorder les trois pins de communication qui sont CS,CLK et MISO comme en figure 10 et le schéma de câblage est visible en figure 11.

|  |  |  |
| --- | --- | --- |
| Port Arduino | Broche | Couleur |
| +3.3V | Vcc | Rouge |
| GND | GND | Noir |
| 10 | Chip Select | Vert |
| 13 | Clock | Jaune |
| 12 | MISO | Blanc |

1. : Tableau de câblage du capteur de courant



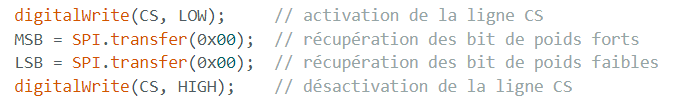
1. Schéma de raccordement entre le capteur de courant et le shield

Le port CS signifie «*Chip Selec*t», il va nous permettre de préciser lorsque l’on souhaite lire les données du capteur. Pour ce capteur il faut lire les données lorsque la valeur du CS est à 0V, nous allons donc câbler le chip select sur le port 10 de la carte Arduino. En ce qui concerne le programme, pour définir le fait que nous allons lire et enregistrer les valeurs seulement à l’état bas du CS nous allons dans un premier temps dire que nous avons le pin du Chip Select que le pin 10 de la car Arduino comme il est indiqué en figure 12.



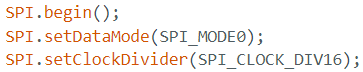
1. : Affectation de la broche 10 à la variable CS

Puis nous après avoir préciser que CS était une sortie nous la mettons à l’état bas pour après lire et enregistrer les valeurs dans les variable MSB et LSB puis nous nous remettons le CS à l’état haut pour dire que nous arrêtons de lire les valeurs, cette partie du programme est visible en figure 13.



1. : Mise à l’état bas puis haut du *Chip Select*

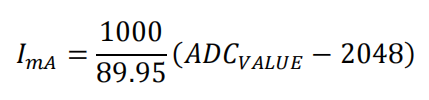
Le CLK est l’abréviation de *clock* que est l’horloge, il s’agit de la fréquence à la quel on va pouvoir lire un bit, plus la fréquence est élevée, plus les impulsions vont être proche et plus nous avons pouvoir lire de valeurs. Nous avons décider d’utiliser une fréquence de communication pouvant être générer par la carte Arduino. Nous avons donc choisit de fonctionner sur une fréquence de 1Mhz pour être sur de pouvoir récupérer toute les informations du capteur, il faut donc paramétrer le fonctionnement SPI de la carte Arduino. On commence par activer le mode SPI de la carte Arduino, étant donné que l’on doit lire les valeurs du capteur quand le Chip Select est l’état bas on doit se mettre en mode0 du SPI et on indique la fréquence de l’horloge qui est pour nous à 1Mhz. On peut voir le paramétrage du mode SPI en figure 14.



1. : paramétrage du mode SPI de la carte Arduino

Le MISO signifie «*Master In Slave Out»*, c’est sur ce port que nous allons récupérer les données qui seront sous forme de bit à l’état 1 ou 0 selon la valeur du courant, ils seront synchroniser à la fréquence de l’horloge ce qui nous permettre de lire les bits du signal. Les données du capteurs sont codés sr 12 bits avec un bit de signe, cela veux dire que le bit 212 est à 1 si le courant est positif et à 0 si il est négatif, les reste des bit permet de calculer la valeur du courant.

Le MSB signifie «*most significative bit»* ce qui signifie le bit de poids fort et LSB est son inverse «*Less significative bit»* et donc bit de poids faible. Ces deux variables contienne 8 bits de données et nous allons les rassembler pour former un Mo qui contiendra la valeur du courant. Puis de cette valeur nous appliquons la formule de la figure 15 qui est donné par le constructeur [4] pour convertir cette valeur en courant.



1. : Formule du courant donnée par le constructeur

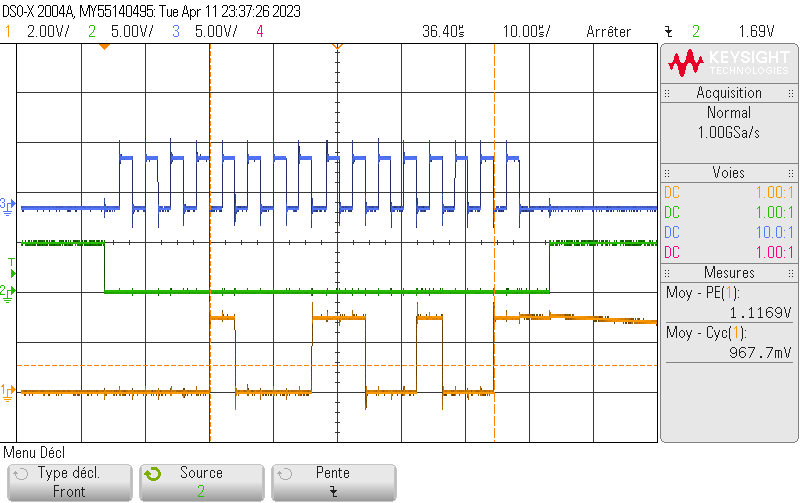
Nous enregistrons le résultat de ce calcul dans une variable que l’on pourra afficher sur l’écran LCD, pouvons voir le calcul au sein du programme en figure 16.

calcul de courant

1. : calcul du courant à partir du MSB et du LSB

## Résultats

Après avoir effectuer et téléverser la programme nous avons mis un oscilloscope en sortie des pine afin de récupérer les différent signaux électrique entant et sortant du capteur de courant. Une fois les broches Chip Select, MISO et CLK branché à l’oscilloscope nous pouvons visualiser la figure 17.

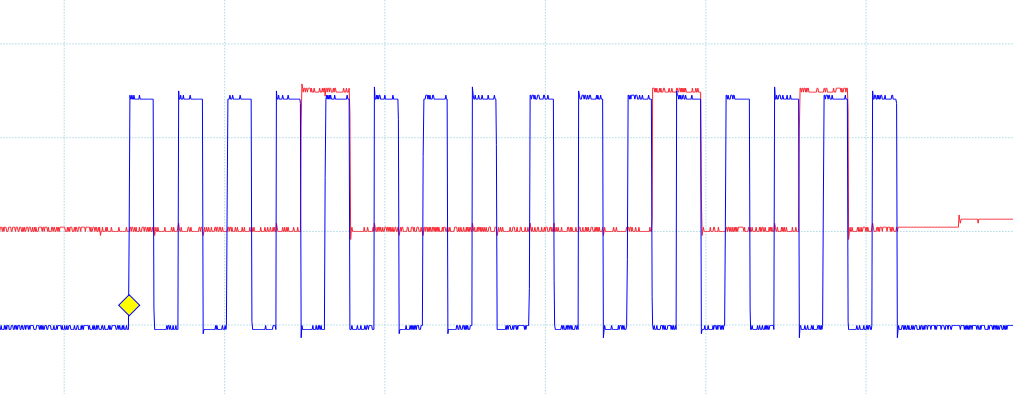


1. : Visualisation des broches CS, MISO et CLK à l’oscilloscope

Nous pouvons voir en bleu le signal de l’horloge ainsi que les 16 fronts d’horloge. Le signal vert est celui du Chip Select, on peut noter qu’il n’y a pas de front d’horloge lorsque qu’il est à l’état haut. Le signal orange est celui du signal du MISO et donc lorsque l’on compte le nombre de bit à l’état un sur le MISO, on peut lire la valeur du courant, de ce cas là d’après le capteur le courant vaut 2.5A.

Nous avons donc décider de faire plusieurs mesure de courant pour vérifier le précision du capteurs. En disposant un ampèremètre comme valeur de référence et le capteur de courant en série nous pourrons voir si le courant calculer par le capteur est bien le même que celui de ampèremètre.

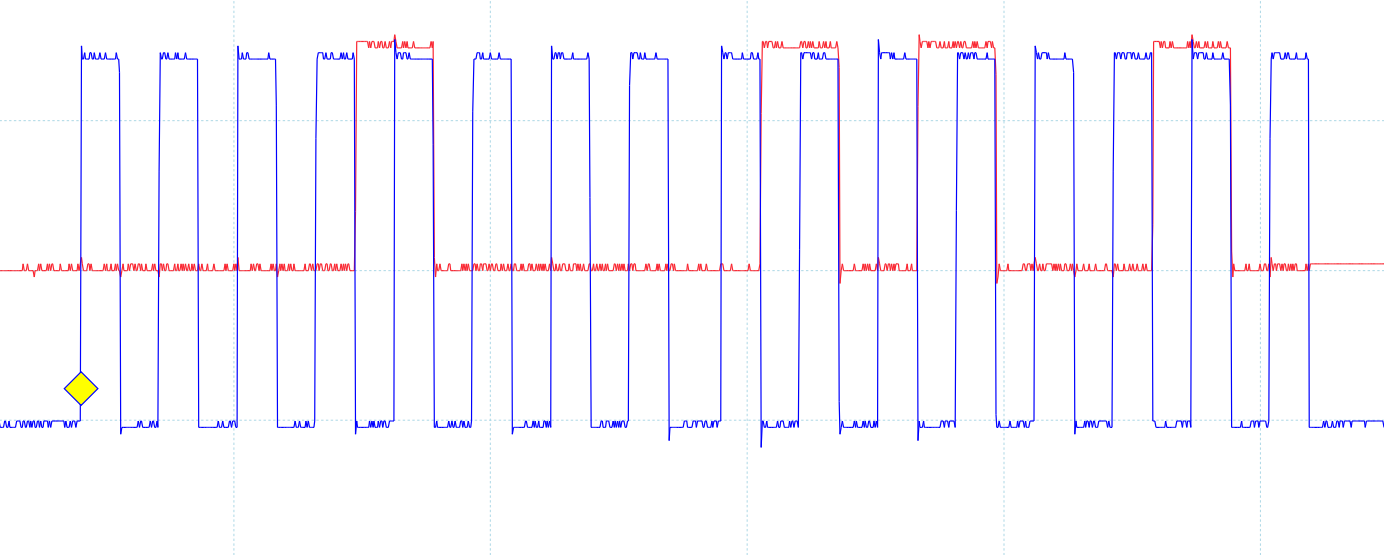
La figure 18 prise grâce à un oscilloscope pico représente le signal MISO du capteur de courant alignés au signal de l’horloge.



1. : Signal d’horloge et signal MISO du capteur de courant

Pour la lecture des bits nous regardons si le signal du MISO est à l’état 1 en même temps que le front montant de l’horloge. Nous pouvons voir que le bit 12 est à l’état 1 ce qui signifie que le courant est positif. Parmi les bits qui permettent de calculer la valeur du courant nous pouvons voir que les bit 2 et 5 sont également à l’état haut. Lorsque l’on fait la somme des bits nous trouvons une valeur de 2066. En appliquant la formule du constructeur nous avons une valeur de courant de 0.20A alors que la valeur réel est de 0.28A ce qui nous fait une erreur de 28.6%.

Nous avons refait des mesures avec un courant plus élevé pour voir sir la marge d’erreur varie ou non. En figure 19 nous avons les signaux dans le cas où le courant vaux 1.05A.



1. : Signal d’horloge et du MISO à 1.05A

Avec ce deuxième exemple, après calcul nous avons un valeur de 0.911A au lieux de trouver 1.05A ce qui nous faite une erreur de 13%.

## Conclusion

Grâce au test nous pouvons voir qu’il y a une marge d’erreur qui est de 28%, nous avons donc choisit d’appliquer une correction afin de réduire ce pourcentage d’erreur à 2% en ajoutant une amplification de 0.132%. Après correction nous avons pus mettre en place le tableau de la figure 20 qui représente les différente valeurs du courant avec et sans la correction.

|  |  |  |
| --- | --- | --- |
| Courant réel | Courant du capteur sans correction | Courant du capteur avec correction |
| 0 A | 0 A | 0 A |
| 0,5 A | 0,35 A | 0,4 A |
| 1 A | 0,8 A | 0,93 A |
| 1,5 A | 1,27 A | 1,43 A |
| 2 A | 1,71 A | 1,94 A |
| 2,5 A | 2,17 A | 2,44 A |

1. : Tableau des différentes prises de mesures du courant

# Commande du hacheur

## Introduction

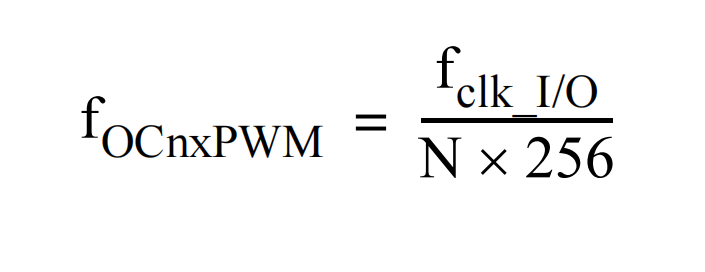
Pour la commande du hacheur nous allons le commander grace à la gâchette du transistor qui se trouve dans le circuit du hacheur. On le commandera à une fréquence supérieur à 20kHz pour ne pas entendre de bruit, de ce fait nous avons décider de l’utiliser à 50kHz. Pour effectuer la commande du hacheur nous allons utiliser la fonction PWM de la carte Arduino, seulement, la sortie de la carte fournit une tension de 5V or la gâchette du transistor s’active à une tension minimale de 10V. Nous allons donc mettre en sortie du PWM un Amplificateur Opérationnel.

## Partie 1 : PWM

### Méthode

La fonction PWM (Pulse Width Modulation) de la carte Arduino signifie Modulation de largeur d’impulsion, elle permet de créer une signal périodique d’une fréquence de 500Hz, la largeur de l’impulsion peut être modifier ainsi que le choix de port de sortie du signal. Dans notre cas nous allons modifier la largeur de l’impulsion mais aussi travailler sur le fréquence plus élevée, pour cela nous devons faire des modifications dans les registres.

Pour cela il faut augmenter la fréquence de l’horloge des sorties PWM et cette modification ne peut s’applique que pour certaine sorties appelés Timers. Selon Locoduino [2] qui explique sur son site le fonctionnement des Timers de la carte Arduino, nous avons décider d’utiliser le Timer 2 car nous allons utiliser la pin 3 comme sortie. Au niveau du programme, il faut la définir comme sortie et paramétrer les différents registres afin d’avoir les résultats souhaités. Selon la documentation de l’ATMega328P [6] il nous faut régler la sortie en «*Fast PWM Mode»*, ce mode permet d’avoir une fréquence de sortie la plus élevée possible. Nous devons choisir dans un premier temps la fréquence voulu grace à l’équation de la figure X.

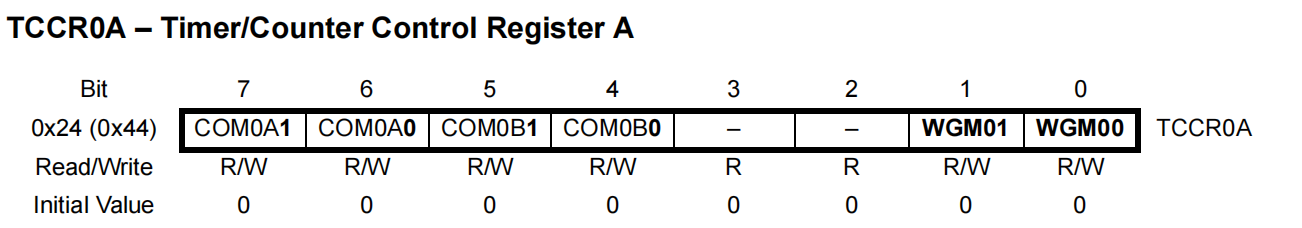


1. : Equation permettant de trouver la fréquence de sortie

Dans cette équation «N» est la variable de «prescaler», cette variable est une horloge interne qui permet de modifier la fréquence de l’horloge interne. Elles peut prendre les valeurs 1, 8, 64, 256 ou 1024. Avec les différentes valeurs possible nous ne pouvons pas avoir le fréquence de 50kHz souhaité, nous devons donc trouver une alternative.

L’autre moyen d’avoir la fréquence du PWM souhaitée est de créer un signal périodique et d’activer la sortie à chaque fois que la valeur souhaité est dépasser. Selon le signal périodique choisit nous pouvons avoir une fréquence très élevée ou avoir une fréquence peu élevée mais plus précise. Nous allons donc choisir les réglages de manière à avoir la fréquence la plus élevée ce qui correspond à un réglage en *«Fast PWM»*.

Pour cela il faut régler plusieurs registres, le premier registre, TCCR0A qui signifie «Timer/Counter Control Register A», il nous permet de régler l’horloge de fonctionnement de la sortie PWM. Le paramétrage du registre se fait en activant ou en déactivant les différents bit visible en figure X.



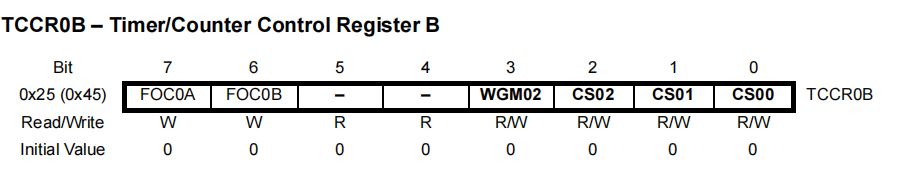
1. : Bits du registre TCCR0A

Le registre se compose de 6 bits pouvant être mise à l’état 1 (activé) ou 0 (désactivé), pour activer les différents bit lors de la programmation, on peut écrire manuellement les bits que nous voulons mettre à 1 ou alors on peut donner à la variable TCCR0A une suite de 0 et de 1 qui signifie si le bit doit être à l’état un ou 0. Nous choisissons de mettre à 1 ou à 0 les bits que l’on veux car pour certain paramètres il nous est nécessaire de déclarer explicitement l’état de la variable.

Les bits COM0A et COM0B vont permettre de choisir les différents mode de fonctionnement de la sortie PWM. Dans notre cas nous allons désactiver les bits COM0A1 et COM0A0 ce qui signifie que nous allons pas utiliser le port OC0A

Les bits WGM01, WGM00 et WGM02 (dans les registre TCCR0B) nous permettent de choisir le mode de fonctionnement du PWM. Dans notre cas nous voulons une fréquence spécifique qui est très élevée pour le fonctionnement de la carte Arduino, nous allons donc nous mettre en mode *Fast PWM* et pour avoir le fréquence exact nous allons comparer deux valeurs et créer une impulsion dés que la valeur de OCRB sera supérieur à la valeur de OCRA.

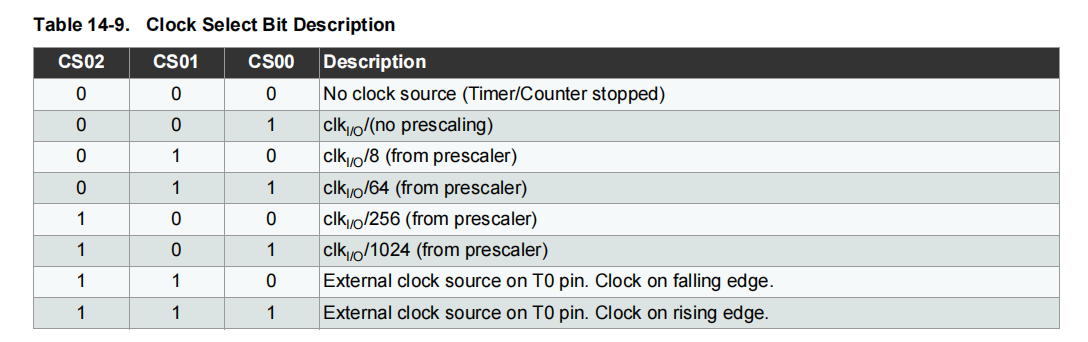
Le second registre à paramétrer est le registre TCCR0B qui signifie «Timer/Counter Control Register B» , il contient 6 bits pouvant être régler, il sont visible en figure X.



1. : Bits du registre TCCR0B

Les bits FOC0A et FOC0B ne sont pas utiliser car ils permettent de comparer les sorties entre elles alors que ce n’est pas ce que l’ont souhaite et dans le documentation technique il est stipulé qu’il faut mettre le bit à 0 si on ne veux pas l’activer.

Les bits CS02, CS01 et CS02 fonctionnent ensemble, ils permettent de donner une valeur de prescaler, le tableau de paramétrage est visible en figure X.



1. : Tableau de paramétrage des bits CS02, CS01 et CS02

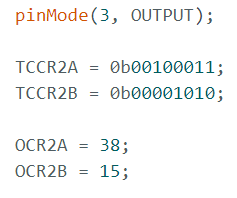
### Résultats

Dans le programme le paramétrage de la fréquence se fera qu’une seule fois et c’est le temps de cycle que nous allons modifier afin de pouvoir bénéficier de la puissance maximale du panneaux solaire.

Dans un premier temps nous définissons la sortie que l’on souhaite utiliser car ce mode de fonctionnement ne peut pas être appliqué à toutes les sortie de la carte Arduino. Par soucis de disponibilité des broches nous choisissons d’utiliser la broche 3 comme sortie de fréquence.

### Conclusion

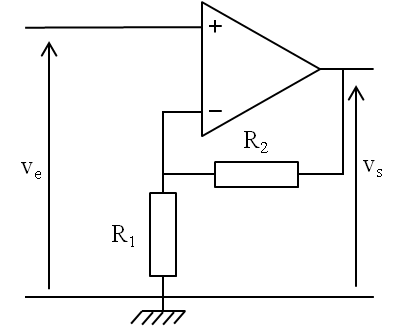
Le paramétrage de de la fréquence de 50kHz nécessite un paramétrage spéciale dans



1. : Paramétrage de la fréquence de 50kHz

## Partie 2 : Amplificateur

Pour faire passer une tension de 5V à 10V il nous faut la multiplier par deux, nous allons donc utiliser un Amplificateur Opérationnel Parfait (AOP) en amplificateur non inverseur, le câblage à mettre en place est visible en figure X, sa fonction de transfert est la suivante :

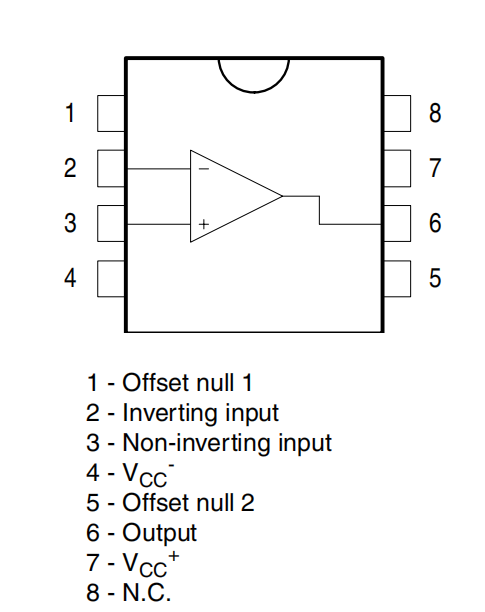


1. : Schéma électrique d’un Amplificateur non-inverseur

Avec Ve la tension d’entrée et Vs est la tension de sortie, de plus pour avoir un gain d’amplification de deux il nous faut donner au résistances les même valeurs.

### Méthode

Pour réaliser l’amplification de tension nous allons utiliser le TL071, ce composent comporte un seul Amplificateur Opérationnel ainsi qu’ une bande passante de 3MHz et peut être alimenté par des tensions de +15V et -15V, ses broches de raccordement sont 7 et 4, visible en figure X.

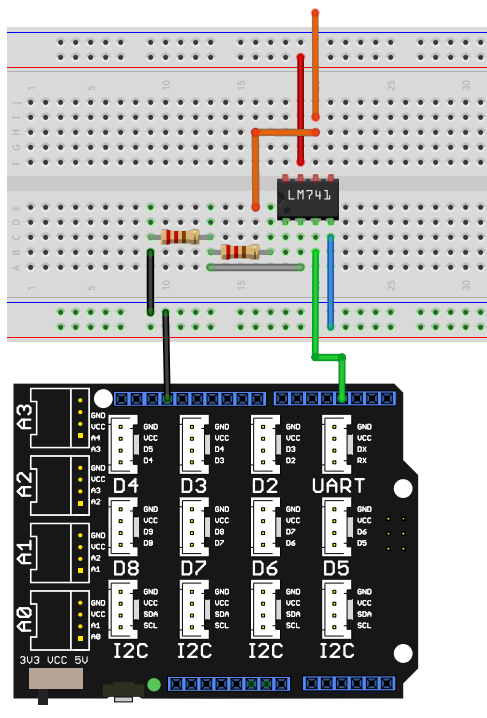


1. : Schéma de répartitions des broches de l’AOP.

Nous raccordons la pin 3 à notre sortie de PWM et nous récupérons le signal amplifié Vs sur la broche numéro 6.

### Résultats

Lors de la réalisation du câblage nous savons utilisé un breadboard afin de disposer les composants. Sur la broche d’entrée 3 qui correspond à l’entrée non inverseur nous allons raccorder la sortie PWM à 50kHz de la carte Arduino Cela nous permet de finalement obtenir le figure X suivante.

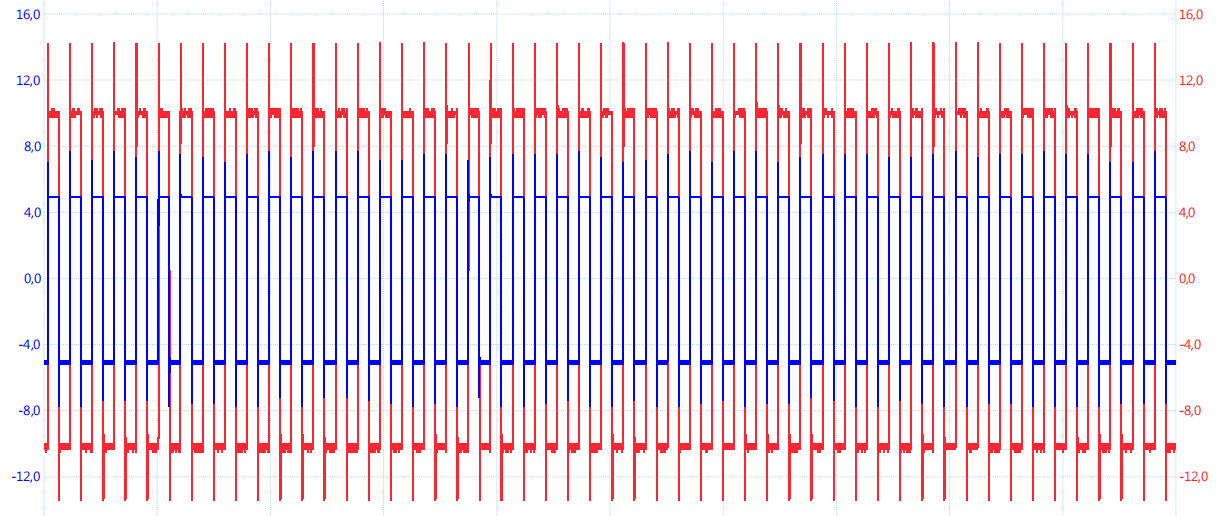


1. : Schéma de câblage de l’amplificateur non inverseur

Les fils rouge et bleu sont les fils +15V et -15V d’alimentation, la tension de sortie Vs est visualisé sur le fils orange.

### Conclusion

Une fois le câblage réalisé et la carte Arduino raccordé au port d’entrée nous visualisons les signaux de la figure X sur laquelle on peut visualiser en rouge la tension d’entrée en bleu et la tension de sortie en rouge. Nous pouvons voir qu’il y a bien un amplification de la tension sans changement au niveau de la fréquence.



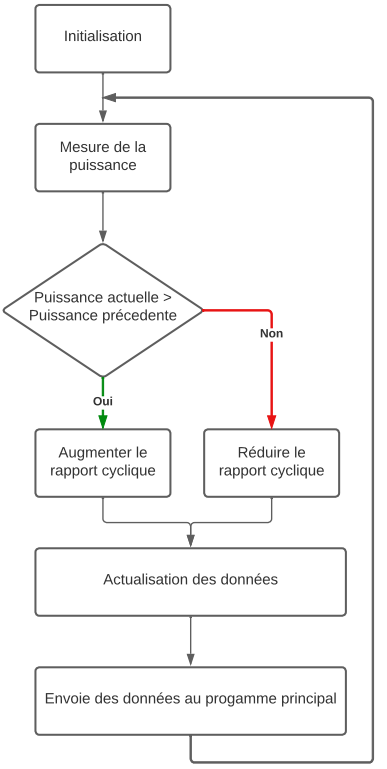
1. : Signal du PWM amplifié avec un gain de 2

# MPPT

Le MPPT signifie Maximum Power Point Tracker, en d’autre terme la recherche du point de puissance maximale, c’est ce qui va nous intéresser dans cette partie.

## Methode

Pour réaliser le programme du MPPT nous allons devoir suivre son algorithme de fonctionnement visible en figure X.



1. : Algorithme permettant de calculer le MPPT

En regardant l’algorigramme nous pouvons voire que lorsque la puissance augmente le rapport cyclique augmente aussi ce qui augmentera la puissance de sortie, tandis que lorsque le panneau solaire va générer moins de puissance; le rapport cyclique va se réduire et dont tot de même réussir à obtenir le maximum de puissance possible. La gestion de la recherche du point de fonctionnement ce fera dans le programme grâce à une fonction à part.

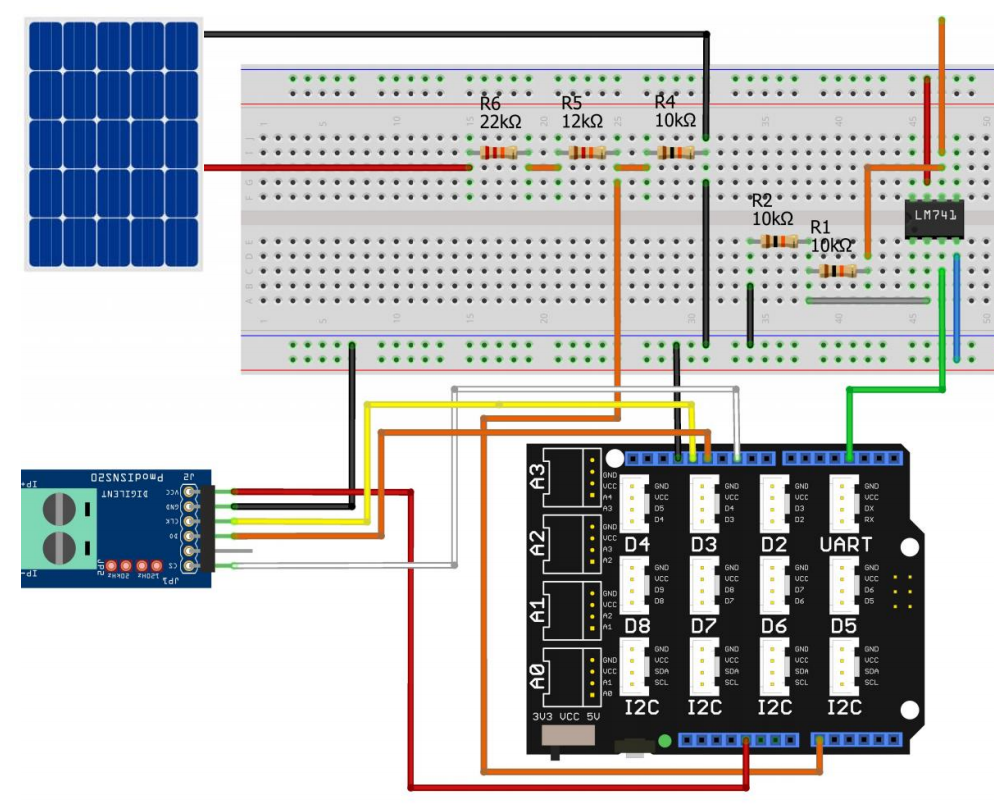
## Programmme

Pour alléger le programme principale nous allons créer une fonction qui va calculer et mettre à jour la puissance fournit par le panneaux solaire. Pour appeler cette fonction , il faut lui envoyer la valeur de la tension et du courant afin qu’elle puisse recalculer la puissance instantanée, de plus cette puissance va être comparée à la puissance précédente pour savoir quelle est la modification qu’il faut apporter au PWM afin d’avoir la puissance de sortie maximale. Cette fonction a aussi comme paramètre la valeur du PWM, car c’est elle qui va le faire augmenter ou diminuer. De ce fait pur utiliser la fonction il faut la paramétrer comme en figure X. Dans cette fonction nous envoyons trois adresses de variables globale afin de modifier dans la fonction leurs valeurs.

Capture d'écran 2023-05-16 091714

1. : Appel de la fonction MPPT

Dans cette fonction nous allons calculer la valeur de la Puissance Instantanée (pInst) grâce à la valeur du courant et de la tension, une fois cette valeur obtenue, nous allons la comparer avec la valeur de la Puissance Précédente (pPrec). Il y a deux cas possible, si la puissance instantanée est plus grande que la précédente, cela veux dire que l’on a une augmentation de la puissance et donc nous augmentons la valeur du PWM de 1. Au contraire, si la puissance instantanée est plus faible que la puissance précédente nous avons une diminution de la puissance et donc nous allons réduire la valeur du PWM de 1. Nous prenons tout de même le soins de mettre des limite au valeurs que peut prendre le pwm, de manière à ce qu’il ne soit jamais nul ou jamais à cent pour cent.



**Amplificateur de tension**

**Shield**

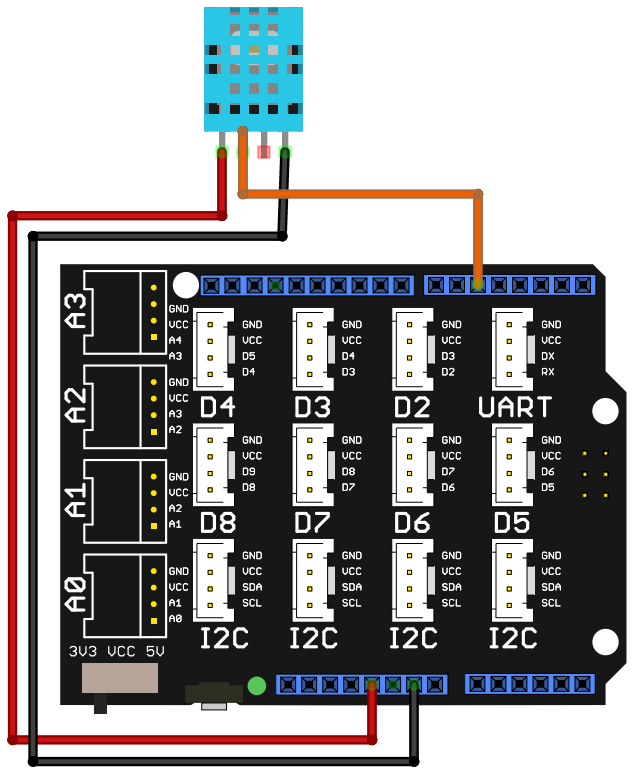
**Panneaux Solaire**

**Capteur de courant**

**Pont diviseur de tension**

1. : Schéma de câblage pour la mesure du MPPT

# Capteur de température



1. : Schéma de raccordement du capteur de température

# Capteur de luminosité

# Annexe

**Récupérer les différentes sources des composants et les ajouter :**

**- DHT11(**

**Listes des différentes librairies**

**Renuméroter les figures et intégrer les sources au texte**

# Sources

1. Arduino, «Arduino Uno R3», Arduino, 2023
2. SeeedStudio, «Base Shield V2», Arduino, 2014
3. Gotronic, «Manuel d’utilisation de l’afficheur I2C LCD 16x2», Joy-IC,2017
4. DIGILENT , «PmodISNS20 Reference Manual», Texas Instrument, 2016
5. Locoduino, «Les Timers (IV)», Internet :https://www.locoduino.org/spip.php?article119, [25/05/2023]
6. Atmel Corporation, «ATmega328P Datasheet», Atmel, 2015
7. Texas Instruments Incorporated, «TL07xx Low-Noise FET-Input Operational Amplifiers», Texas Instruments,2014