



RAPPORT TRACKER SOLAIRE

Rémy Salaberry - Adrien Desfachelles - Samuel Périn - David Enrique

Tuteur de projet : M. Philippe LARRAMENDY

REMERCIEMENTS

Tout d'abord, nous tenons à remercier l'ensemble du personnel ainsi que l'équipe pédagogique de Licence Professionnelle Ecologie Industrielle pour leur aide lors de notre projet.

Nous remercions tout particulièrement M. LARRAMENDY pour son temps et son aide en tant que tuteur de notre projet. Son expérience dans le domaine nous a été primordiale pour mener notre projet à bout.

Nous remercions également M. Camille LAVAYSSIERE qui nous a apporté une aide précieuse sur la partie programmation et pour la prise en main de PYSCADA.

Ce projet nous a permis de renforcer notre cohésion au sein de notre groupe mais également au sein de notre promotion grâce à l'entraide qui est née entre certains groupes.

Présentation du groupe projet

Groupe TRACKER SOLAIRE :

pt1.trackersolaire2020@gmail.com

COMPOSITION : Adrien DESFACHELLES : Chef de projet

Rémy SALABERRY : Responsable communication

David ENRIQUE PEDROSA

Samuel PÉRIN

TUTEUR PAR : M. Philippe LARRAMENDY

@ : plarrame@iutbayonne.univ-pau.fr

TUTEUR (Bis) : M. Camille LAVAYSSIERE

@ : camille.lavayssiere@iutbayonne.univ-pau.fr

RÉSUMÉ

Au cours de notre projet tuteuré, notre groupe composé de Adrien DESFACHELLES, David ENRIQUE, Samuel PERIN et Rémy SALABERRY avons continué un travail sur l'élaboration d'un tracker solaire débuté en 2013. Durant ces 6 semaines de projet, nous avons pu mettre en pratique nos connaissances en électronique et découvrir de nouveaux domaines en programmation Python, HTML et JavaScript.

Dans un premier temps, nous avons pris connaissance du travail effectué au préalable et nous nous sommes ensuite fixé les objectifs suivants :

- Réaliser et mettre au point une carte électronique permettant de faire plus de mesures au niveau du panneau solaire (tension, courant, énergie sur le système...)
- Développer une partie supervision sur le logiciel Pyscada installé sur un Raspberry (mini-ordinateur) permettant de communiquer avec le PV et relever les valeurs mesurées.

Ce projet nous a permis d'approfondir notre travail en autonomie et d'appliquer les compétences que nous avons apprises durant cette licence. De plus, il nous a permis de découvrir de nouvelles choses et de s'adapter aux situations et problèmes rencontrés.

ABSTRACT

During our supervised project, our group: Adrien DESFACHELLES, David ENRIQUE, Samuel PERIN et Rémy SALABERRY continued to work on the development of a solar tracker project started in 2013. For 6 weeks, it allows us to put our electronic knowledge into practice and discover new domains in programming Python, HTML et JavaScript.

First, we have read and learned previous work and we fixed these objectives:

- Develop an electronic board in order to make more measurements on the solar panel (voltage, current, energy on the system...)
- Improve the supervision section on Pyscada software installed on a Raspberry (mini-computer) to communicate with the PV and record the measured values.

This project allowed us to deepen our independent work and use the competences we learned during our license. Moreover, we enjoyed discovering new things and adapt ourselves to situations and problems encountered.

Table des matières

REMERCIEMENTS	2
Présentation du groupe projet	2
RÉSUMÉ	3
ABSTRACT	3
I.Présentation générale du projet	5
I.1. Cahier des charges	5
I.2. Analyse de l'existant	6
I.3 Le besoin	6
I.4. Historique	7
II.Diagramme de Gantt	8
III.Programmation (carte de mesure)	9
III.1. Programmation python en utilisant Putty	9
IV.Tuto Pyscada	11
IV.1. Création d'un bouton interactif	11
IV.2. Création d'un popover	12
V.Tuto programmation sur Pyscada	16
VI.Élaboration d'un TP	17
VI.1. Proposition de TP pour les futurs étudiants	17
VII.Carte électronique	18
VII.1. Présentation :	18
VII.2. Création du schéma de la carte électronique :	20
VII.3. Création de la carte avec le logiciel EasyEDA.	25
VIII.Conception 3D d'un boîtier adapté pour les cartes électroniques	28
VIV. Conclusion :	30
TABLE DES RÉFÉRENCES	31
ANNEXE 1	32
Annexe 1.1. Programme python de mesure sur putty	32
Annexe 1.2. Commandes utiles sur putty :	35
Annexe 1.3. Noms des programmes pour le script de pyscada :	35
Annexe 1.4. Programme définition de la variable tableau	35
Annexe 1.5. Code tuto Pyscada :	36
ANNEXE 2	37
Annexe 2.1. Schéma final	37
Annexe 2.2 Nomenclature	38
Annexe 2.3 Schéma final + Ampèremètre	39
Annexe 2.4. Vue 3D de la carte et carte électronique reçue	40

I. Présentation générale du projet

I.1. Cahier des charges

Le département GIM de l'IUT de Bayonne dispose d'un panneau photovoltaïque monté sur un support, celui-ci est orientable sur 2 axes par la commande de deux moteurs (angle d'élévation et angle d'azimut). Ce qui permet de garder un alignement constant avec le soleil afin d'obtenir une production d'énergie optimale. Le projet existant depuis maintenant 7 ans, les groupes précédents ayant bien avancé la partie positionnement du panneau et ainsi que la partie mesurage. Notre groupe devra réaliser les parties suivantes :

- La réalisation et mise au point d'une carte électronique permettant de mesurer des tensions, des courants, des énergies et des températures en différents points du système ainsi que les tests.
- La réalisation d'un boîtier en fonction des dimensions de la carte et des contraintes physiques du système (espace restreint)
- Le développement de la supervision du tracker photovoltaïque sur le logiciel de supervision Pyscada installé sur un Raspberry (mini-ordinateur) ainsi que deux tutoriels (utilisation et programmation).

Schéma simplifié de l'installation

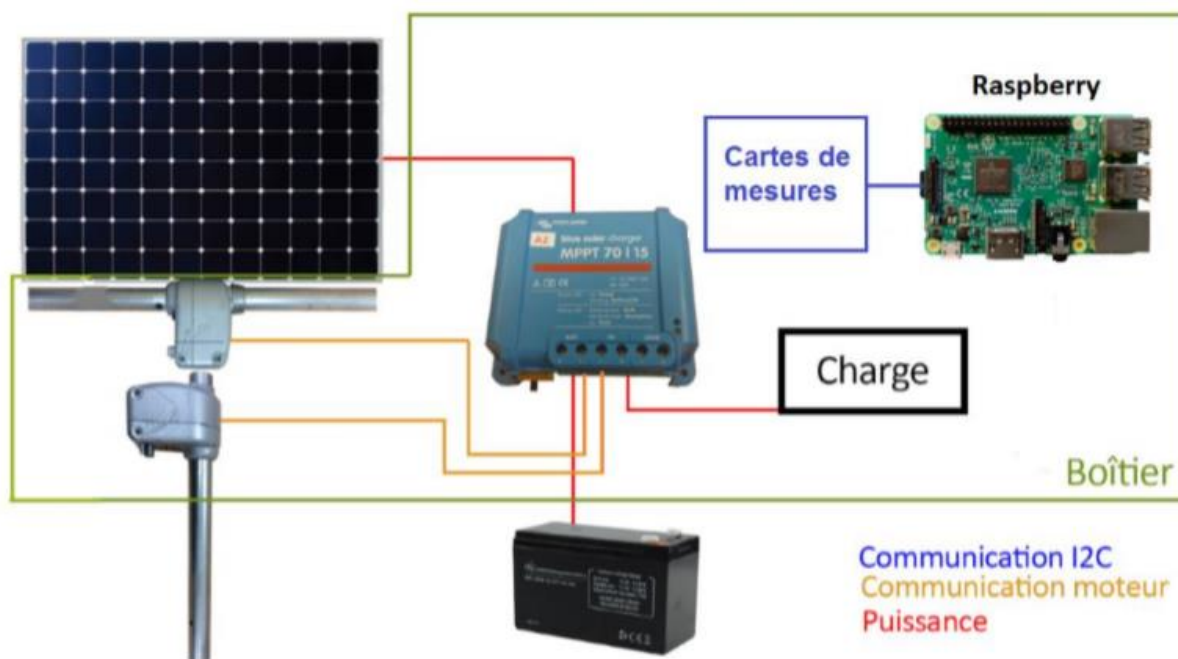


Figure 1 : Schéma simplifié de l'installation

De plus notre groupe se penchera sur la partie éducative potentielle via la création d'un TP pouvant être mis au point.

I.2. Analyse de l'existant

L'équipe de l'année 2018-2019 a rempli la majeure partie de leurs objectifs :

Concevoir un boîtier tout en rendant le système plus étanche, celui-ci devant contenir les deux servomoteurs et la partie électronique qui par la suite sera composé du Raspberry, du régulateur de charge (qui se trouvaient initialement dans l'armoire électrique : MPPT) ainsi que la carte électronique de mesure que nous devons réaliser cette année.

Une modélisation complète du système (panneau solaire + moteurs + boîtier + axes) a été réalisée ce qui nous permettra de pouvoir intégrer le boîtier dans le système et de vérifier si les dimensions sont respectées.

Une interface WEB a été abandonnée car le langage de programmation était laborieux à apprendre, le temps passé pour l'apprendre et le mettre en place était trop élevé pour être satisfaisant. (2015-2016)

Il a donc été choisi d'utiliser putty pour communiquer avec un Raspberry afin d'acquérir les mesures, pour cela nous utilisons des programmes codés en python. Enfin pour traiter les données il a été choisi d'utiliser le logiciel de supervision pycada.

Ils ont commencé la réalisation d'une interface de supervision, affichant les données de mesures nécessaires pour établir des commandes afin de prendre en main en mode manuel le tracker à distance.

Cela pourra potentiellement nous être utile dans la réalisation de TP.

I.3 Le besoin

- La création d'une carte permettant de faire les mesures en différents points du système.
- Le développement d'une interface de supervision interactive permettant le traitement des données mesurées.
- La création d'un boîtier pour les cartes électronique et le système.
- Effectuer les tests de résistance au vent afin de valider l'épaisseur du boîtier.
- Monter le système puis effectuer le câblage dans le but de l'installer sur le toit de l'IUT.
- Effectuer les tests pour valider le projet.

I.4. Historique

Comme nous l'avons évoqué précédemment, notre groupe reprend le travail d'anciens groupes de projet pour poursuivre l'objectif final du projet qui est de créer un tracker solaire ayant le rendement le plus élevé possible puis de le placer sur le toit du bâtiment de l'IUT afin que les élèves du DUT GIM puissent faire des travaux pratiques.

Analyse de l'existant : Le projet a été débuté lors de l'année universitaire 2013-2014. Ils ont réalisé :

- La conception d'un nouveau boîtier permettant la mobilité du panneau verticalement et horizontalement.
- Etude de l'emplacement des points de mesures sur le système afin d'obtenir les données nécessaires pour un suivi du point optimal de fonctionnement du panneau.
- La réalisation de l'interface web sur Pyscada pour traiter les données mesurées et les exploiter.
- L'équation solaire qui permet au tracker de suivre la position du soleil à n'importe quel moment de la journée (angle d'élévation et angle d'azimut en fonction du temps). Cela doit permettre d'augmenter le rendement du panneau. (Début 2013-2014 fin 2017-2018).
- La conception d'un boîtier d'étanchéité pour les servomoteurs ainsi que les cartes électroniques et le module MPPT. (2017-2018).
- La conception de l'armoire électrique indépendante du panneau. (2013-2014).
- La mise en place du Raspberry (micro-ordinateur) qui nous permet de faire l'acquisition des mesures à distance de valeurs à travers des cartes composées d'un capteur. Ainsi que de communiquer à distance grâce à une supervision. (2014-2015).
- La programmation en Python pour communiquer avec le Raspberry (mini-ordinateur) avec l'application Putty. (2014-2015).

II. Diagramme de Gantt

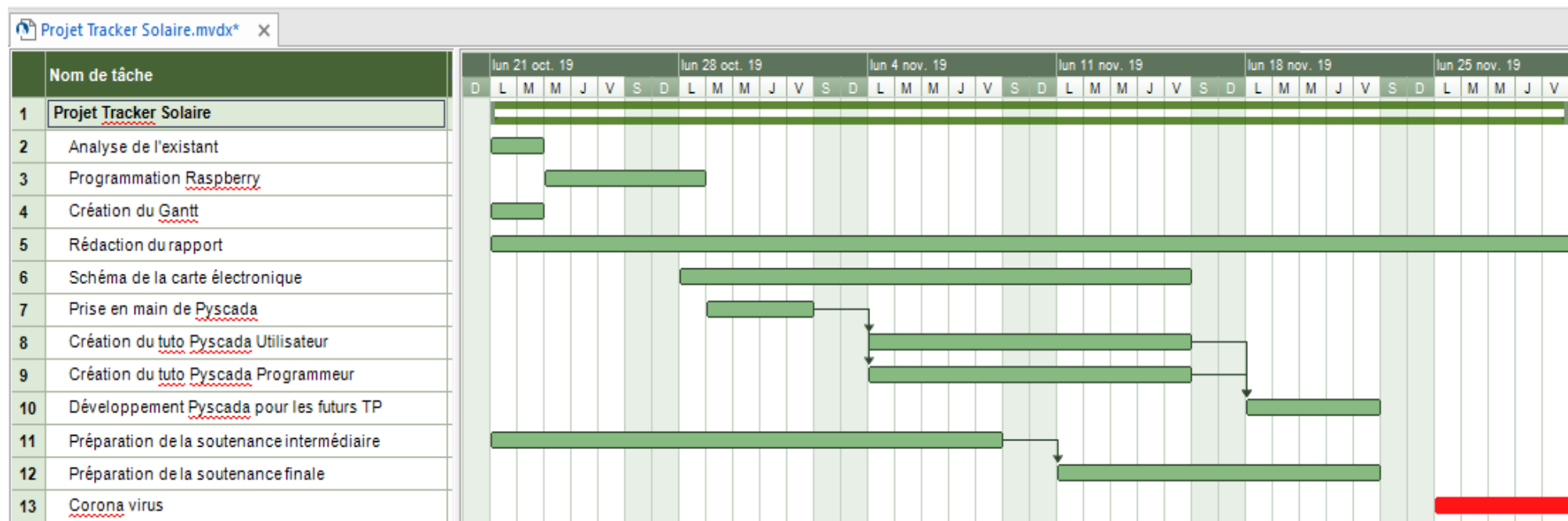


Figure 2 : Diagramme de Gantt

Pour ce qui est de notre organisation dans la réalisation de ce projet, deux parties principales se dégagent : La partie carte électronique assurée par Adrien Desfachelles et la partie programmation et supervision assurée par David Enrique, Rémy Salaberry et Samuel Perin. En effet, nous avons commencé par la rédaction des programmes de mesures en python sur Putty, puis nous nous sommes penchés sur la supervision sur Pyscada, ainsi que les tutoriels utilisateur et développeur. Nous avons ensuite rédigé un exemple de TP pour les futurs étudiants.

Nous avons tout au long du projet contribué à la rédaction du rapport.

III. Programmation (carte de mesure)

III.1. Programmation python en utilisant Putty

Dans cette partie nous traiterons de la programmation sur Putty qui répond à la consigne « Mesure des tensions, des courants et des énergies sur le système » du cahier des charges. Ci-dessous les programmes que nous avons créés et leurs explications.

Putty est une application qui permet de prendre le contrôle d'un PC à distance (dans notre cas un Raspberry pi4) nous allons donc pouvoir obtenir des relevées à distance sur un PC connecté au même réseau que le Raspberry.

TENSION:

```
nano tension.py
```

```
import smbus
bus = smbus.SMBus (1)
address = 0x6f
VinMSB= (bus.read_byte_data(address, 0x1e))*16
VinLSB= (bus.read_byte_data(address, 0x1f))/16
print (VinMSB+VinLSB)*float(0.025)
```

Importation des variables bus et de l'adresse

Lecture de la valeur de tension (*16 pour passer en hexadécimal)

Affichage de la valeur calculée

##La tension est codée sur VinMSB et VinLSB, qui sont codées sur 8 bits chacune. Les 2 valeurs additionnées et multipliées par ce coefficient calculé à partir de la documentation donnent la tension.

COURANT:

```
nano courant.py
```

```
import smbus
bus = smbus.SMBus (1)
address = 0x6f
senseMSB= (bus.read_byte_data(address, 0x14))*16
senseLSB= (bus.read_byte_data(address, 0x15))/16
print (senseMSB+senseLSB)*0.0025
```

##Même chose pour le courant avec les adresses correspondant aux valeurs de courant.

Figure 3 : Programme de mesure de tension et courant

PUISSANCE :

```
nano puissance.py
import smbus
bus = smbus.SMBus (1)
address = 0x6f
MSB2= (bus.read_byte_data(address, 0x05))*16
MSB1= (bus.read_byte_data(address, 0x06))*16
LSB= (bus.read_byte_data(address, 0x07))*16
print (MSB2*65536+MSB1*256+LSB)*float(0.0673157)
```

Même fonctionnement avec un coefficient différent calculé à partir de la documentation de 0.0673157. On multiplie MSB1 par 256 (2^8) car cela correspond au second octet de 8 bits suivant LSB. Idem pour MSB2 à la suite du MSB1 on multiplie donc par 2^{16} .

Figure 5 : Programme de mesure de puissance

ENERGIE :

```
nano energie.py
import smbus
bus = smbus.SMBus (1)
address = 0x6f
LSB=(bus.read_byte_data(address, 0x3F))
MSB3= (bus.read_byte_data(address, 0x3C))*16
MSB2= (bus.read_byte_data(address, 0x3D))*16
MSB1= (bus.read_byte_data(address, 0x3E))*16
print (MSB2*65536+MSB1*256+LSB+MSB3*16777216)*0.0673157
```

Même fonctionnement que pour la puissance avec une plus grande valeur possible, d'où l'ajout d'un MSB3.

Figure 4 : Programme de mesure d'énergie

Avec ces programmes nous sommes maintenant capables de demander au mini-ordinateur (Raspberry) les valeurs mesurées avec la carte électroniques. Afin de comprendre comment nous mesurons ces valeurs, il est nécessaire de lire la partie « VI. Carte électronique » mais avant nous allons vous présenter ce que nous pouvons faire de ces mesures et dans qu'elle cadre nous pourrions les exploiter (exemple création d'un TP). Pour se faire nous allons utiliser l'interface Pyscada qui est un logiciel de supervision sur lequel nous pouvons créer tout type d'interfaces interactifs, diagrammes, courbes ect.. Pour réaliser cela nous avons réalisé les deux parties suivantes (IV ; V) qui vous expliqueront comment créer des interfaces sur ce logiciel.

IV. Tuto Pyscada

Nous avons réalisé un tutoriel pour les nouveaux utilisateurs afin qu'ils puissent comprendre comment naviguer dans l'interface et en avoir une bonne utilisation sans perdre de temps.

Le tutoriel se déroule en plusieurs interactions avec l'utilisateur :

- Un Bouton pour démarrer le tutoriel.
- Des petites fenêtres appelées popovers s'affichent et donnent les informations nécessaires.
- Une fois l'information reçue l'utilisateur a le choix entre arrêter le tutoriel et le poursuivre avec deux boutons : une croix pour fermer la fenêtre et une flèche pour continuer et ceci jusqu'à ce que le tutoriel soit terminé.
- Lorsque l'utilisateur passe la souris sur certains éléments, une petite fenêtre apparaît pour indiquer à quoi sert l'élément, c'est un tooltip.

Le code de ce tutoriel a été rentré sur la console de dialogue locale de l'ordinateur ou nous l'avons conçu, il n'est donc pas encore disponible sur Pyscada car il n'est pas terminé. L'ensemble du code est présent en ANNEXE 1 pour les étudiants des années suivantes.

IV.1. Création d'un bouton interactif



La ligne de code suivante sert à créer un bouton avec les caractéristiques :

Texte/type de bouton/emplacement/affichage du bouton/Fermeture du code/commentaire/appellation du programme

```
$('.navbar-right').prepend('<li class="dropdown"><!-- Tutoriel --><button type="button" id="start_tuto" class="btn btn-info btn-lg" data-container="body" data-placement="left" data-original-title="" title="">Start Tuto</button></li>')
```

Figure 6 : Bouton interactif et programme

IV.2. Création d'un popover

Un popover est une fenêtre qui apparaît après une interaction, en général elles sont utilisées pour venir en aide à l'utilisateur (commentaire ou informations), dans notre cas nous utilisons les popovers afin de faire un tutoriel expliquer l'utilisation de la page. Pour cela nous avons choisi de faire des popovers interactifs, pour cela nous avons dû ajouter des boutons dans les popovers.

Exemple de popover interactif créé et interactions possibles :

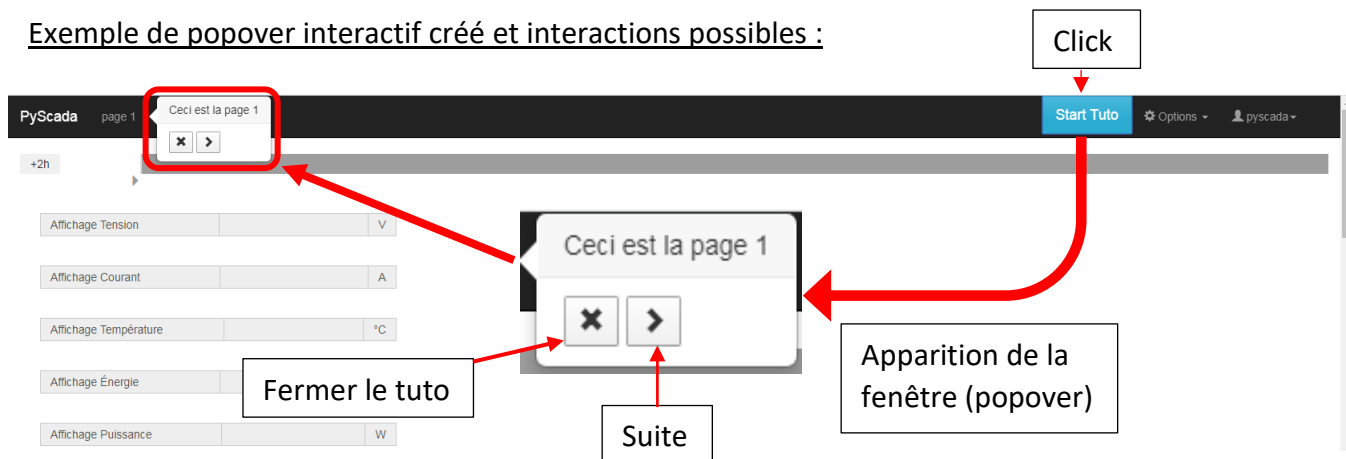


Figure 7 : Utilisation d'un popover

Dans un premier temps afin de créer un popover il nous faut créer une variable appelée « tableau », car par la suite nous allons appeler cette variable dans le programme créant le popover.

Pour créer la variable « tableau » nous utilisons le code suivant :



Figure 8 : Programme de la variable tableau

Lecture du tableau :
[0][0] = [ligne][colonne]
Exemple si on a : [0][3]
On lit « viewport »

Définition des « cases » du tableau.

Répertoire de fonction, variables et caractères utilisés dans le programme suivant.

Code disponible en ANNEXE 1.3

Une fois la variable tableau créée nous allons l'utiliser afin de créer le popover.

Pour cela nous avons besoin d'une boucle if et d'une boucle for ; une boucle if est une boucle qui s'exécute pour une condition remplie, ici nous avons entré la condition supérieur ou égal à 1 afin de commencer après la ligne 0 car elle ne nous est pas utile.

La boucle for nous sert à définir la ligne : [0] [0]

```
if (tableau.length > 1)
for (let i = 1; i < tableau.length; i++) {
$(tableau[i][0]).popover({
  title: tableau[i][1],
  html: true,
  content: contentHtml,
  container: tableau[i][2],
  viewport: tableau[i][3],
  trigger: 'manual'
});
// fonction fermer popover
}).on('shown.bs.popover', function (eventShown) {
  var $popup = $('.navbar-collapse #' + $(eventShown.target).attr('aria-describedby'));
  $popup.find('button.close-popover').click(function (e) {
    $popup.popover('hide');
  });
// fonction montrer popover suivante
$popup.find('button.next-popover').click(function (e) {
  $popup.popover('hide');
  $('span.input-group-addon:contains("V").').popover('show');
});
```

Boucle for dans laquelle on définit i = ligne du tableau

Définition de la colonne : tableau[i][n] (0, 1, 2, et 3) = colonne du tableau. Lecture de la variable tableau popover

Figure 9 : Programme création du popover et affichage

Maintenant que le popover est créé nous devons l'afficher, nous avons pour cela créé un programme utilisant une fonction qui permet d'afficher un élément en cliquant sur un autre élément :

// Cette partie de code permet à la première fenêtre du tutoriel de s'afficher lorsque l'on clique sur le bouton de démarrage du tutoriel.

```
$('#start_tuto').on("click", function() {
  $('a[href$="page1"]').popover('show')
});
```

Click

Start Tuto

Figure 10 : Programme du lancement du popover (tutoriel)

Nous devons tout de même laisser une entière maîtrise à l'utilisateur, pour cela nous avons ajouté une fonctionnalité qui permet de fermer à tout moment le tuto ou de le poursuivre.

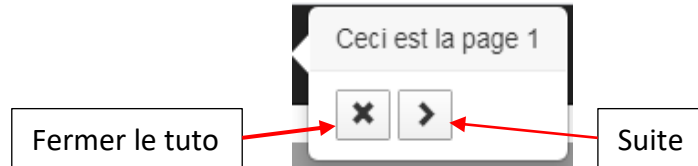


Figure 11 : Fonction du popover

Nous utilisons le code suivant pour créer les boutons et leurs fonctionnements :

```
'<div>',

// Bouton pour arrêter le tutoriel (représenté par la croix)
'<button><span class="glyphicon glyphicon-remove close-popover"></span></button>',

// Bouton pour passer à la fenêtre suivante (représenté par la flèche vers la droite)
'<button><span class="glyphicon glyphicon-chevron-right next-popover"></span></button>',
'</div>'].join("\n");

// Fonction fermer le popover
$popup.find('button .close-popover').click(function (e) {
    $popup.popover('hide');
});

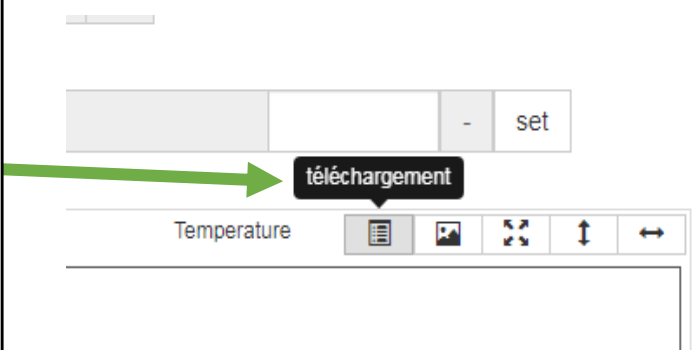
// Fonction popover suivant
$popup.find('button .next-popover').click(function (e) {
    $popup.popover('hide');
    if (tableau[i][4].length > 0){
        $(tableau[i][4]).popover('show');
    }
}
```

Figure 12 : Programme des interactions sur le popover

Les tooltips sont principalement utilisées pour clarifier la signification et la fonction de chaque icône. Elles se montrent si vous placez votre souris sur l'icône.

Les actions des icônes ne sont pas toujours très claires. Avec cet outil, nous visons à fluidifier l'utilisation de notre page, de cette façon l'utilisateur n'aura pas à essayer toutes les icônes pour savoir ce qu'elles font si les icônes ne sont pas assez explicites d'elles-mêmes.

```
If (tableau_tool.length > 1){  
For (let i = 1; i < tableau_tool.length; i++) {  
$(tableau_tool[i][0]).tooltip({  
    title:tableau_tool[i][1]  
    html : true,  
    content : tableau_tool[i][2],  
})  
$(tableau_tool[i][0]).tooltip('show')  
$(tableau_tool[i][0]).tooltip('hide')  
}}  
  
// Idem que pour le popover
```



The diagram illustrates the application of a tooltip to a button in a user interface. On the left, a code block shows a jQuery script that iterates through a table of tool elements and applies a tooltip to each. The tooltip configuration includes a title, HTML content, and actions for showing and hiding the tooltip. A green arrow points from the code to a specific button in a UI mockup. The UI mockup shows a temperature control interface with a 'Temperature' label, a 'téléchargement' button, and a 'set' button. The tooltip is shown hovering over the 'téléchargement' button.

Figure 13 : Programmation de l'affichage d'une bulle informative

À la suite de ces programmes vous devriez maintenant être capables de créer des fenêtres interactives dans l'interface Pyscada qui permettront de guider les prochains utilisateurs de ce logiciel de supervision, cependant il reste important de pouvoir afficher et traiter les valeurs acquises à l'aide de la partie précédente.

Pour cela vous trouverez dans la partie suivante le tuto entier permettant de faire ces différentes tâches (il est important d'accéder au logiciel à l'aide du lien).

Enfin vous pourrez imaginer une interface avec le TP qui suivra cette partie ou encore le potentiel de ce logiciel de supervision.

V. Tuto programmation sur Pyscada

La mise à jour de l'interface utilisateur sur Pyscada devra être faite par les étudiants des années prochaines. Pour faciliter la compréhension du logiciel nous avons fait un tutoriel sur readthedocs: <https://tuto-pyscada.readthedocs.io/fr/latest/index.html>

Identifiant readthedocs: pt1.trackersolaire2020@gmail.com

Mot de passe : Raspberry0

Identifiant Pyscada : Pyscada

Mot de passe : Raspberry

Ce tutoriel a été réalisé à l'aide de sphinx, hébergé sous github puis compilé sur readthedocs. Les pages sources du tutoriel sont aussi disponibles sur github, il suffit de choisir la page voulue et de cliquer sur le bouton "raw" dans github.

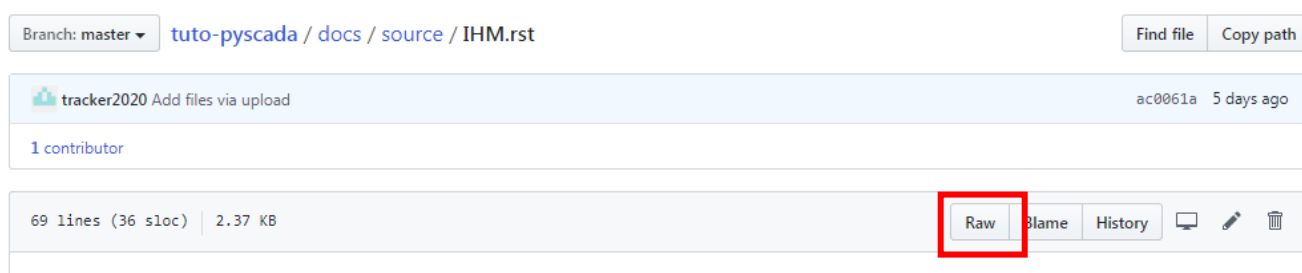


Figure 14 : Page source de github

```
Gras : **texte**
Italique : *texte*

Encadré (pour du code par exemple):
::
    Tant qu'il y aura un espace de tabulation le texte sera encadré.
    Le texte est toujours encadré ici.
Le texte ne sera plus encadré ici.
Environnement pour le sommaire, maxdepth = 2 équivaut à titre et sous-titre maximum
(pas de sous sous titre autorisé). Exemple de notre index :
.. toctree::
    :maxdepth: 2

Titre principal, disposez des « = » en dessous du titre que vous voulez
=====
Sous titre
^^^^^^^^^^
Pour mettre une image il suffit d'écrire ..image :: suivi du chemin de l'image, voici un
exemple :
.. image::pic/vue_pyscada.png
```

Figure 15 : Syntaxe sur sphinx

VI. Élaboration d'un TP

VI.1. Proposition de TP pour les futurs étudiants

TP : Optimisation d'un panneau solaire

L'objectif de ce TP serait de comparer le rendement énergétique d'un panneau solaire dans deux configurations différentes. Une lorsque le panneau est équipé d'un tracker et la seconde lorsque le panneau est fixe dans les mêmes conditions d'ensoleillement. Deux panneaux ont été installés pendant une journée sur le toit de l'IUT, vous trouverez en annexe les valeurs relevées à la suite de ces acquisitions.

1) Acquisition de données

Avec l'aide des tutoriels réalisés sur Pyscada, le but est de créer une interface permettant de relever les mesures de tension, courant, puissance et énergie en temps réel puis de lancer une acquisition sur 10 minutes.

2) Inclinaison optimale du panneau solaire

Le but de cette partie serait de déterminer l'influence de l'angle du panneau par rapport au soleil sur sa productivité.

Pour cela, il faudra faire varier le pitch de 10° en 10° à partir de Pyscada et tracer la puissance générée par le panneau en fonction de son angle d'incidence. Commentez les résultats obtenus et expliquez l'importance du tracker solaire.

3) Calcul d'efficacité d'un tracker solaire

Dans cette partie, il faudra comparer le rendement énergétique d'un panneau solaire équipé d'un tracker solaire et d'un autre équipé d'un bras fixe.

En annexe un relevé aura été fait sur une journée dans les deux configurations.

- 1) En prenant en compte l'énergie nécessaire au fonctionnement des moteurs du tracker solaire, comparer la production de chaque installation et expliquer. Avec l'aide de la seconde annexe (ensoleillement à Anglet sur l'année) estimer la production à l'année.

On supposera que la différence en pourcentage de production des deux installations sera similaire sur toute l'année.

- 2) La différence de prix entre les deux installations étant de X euros, calculer le temps de retour sur investissement d'un tracker solaire.
- 3) En admettant que la durée de vie d'un panneau solaire est de 20 ans, estimer la rentabilité de ce tracker solaire.
- 4) Quelles sont les contraintes d'une telle installation par rapport à un panneau fixe ?

VII. Carte électronique

VII.1. Présentation :

Dans cette partie nous traiterons de la création et réalisation de la carte électronique. Dans un premier temps la partie création sera décomposée par le fonctionnement des composants et du schéma électronique, puis dans une autre partie nous traiterons de la création de la carte sur Easy EDA qui est en lien avec celle-ci.

Tout d'abord, nous sommes partis du travail réalisé l'année précédente. Le groupe a réalisé une carte permettant de mesurer sur un point, différentes grandeurs (énergie, tension, puissance, courant).

Notre objectif cette année est de réaliser un système avec une carte permettant de mesurer en différents points ces grandeurs mais aussi la température sur le panneau, dans le boîtier général du système et sur la carte, tout en étant autonome.

Le fonctionnement d'un panneau solaire normal est celui-ci :

Nous avons un panneau solaire avec un module MPPT (régulateur solaire permettant d'adapter la tension), qui alimente une batterie et une charge.

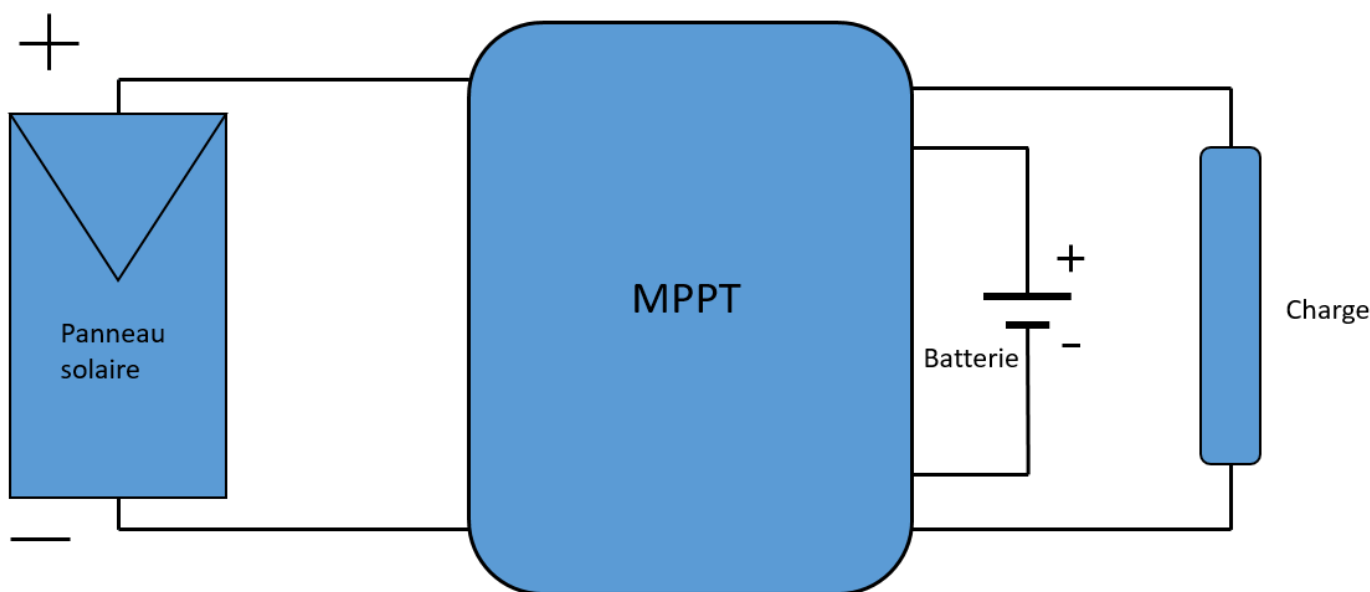


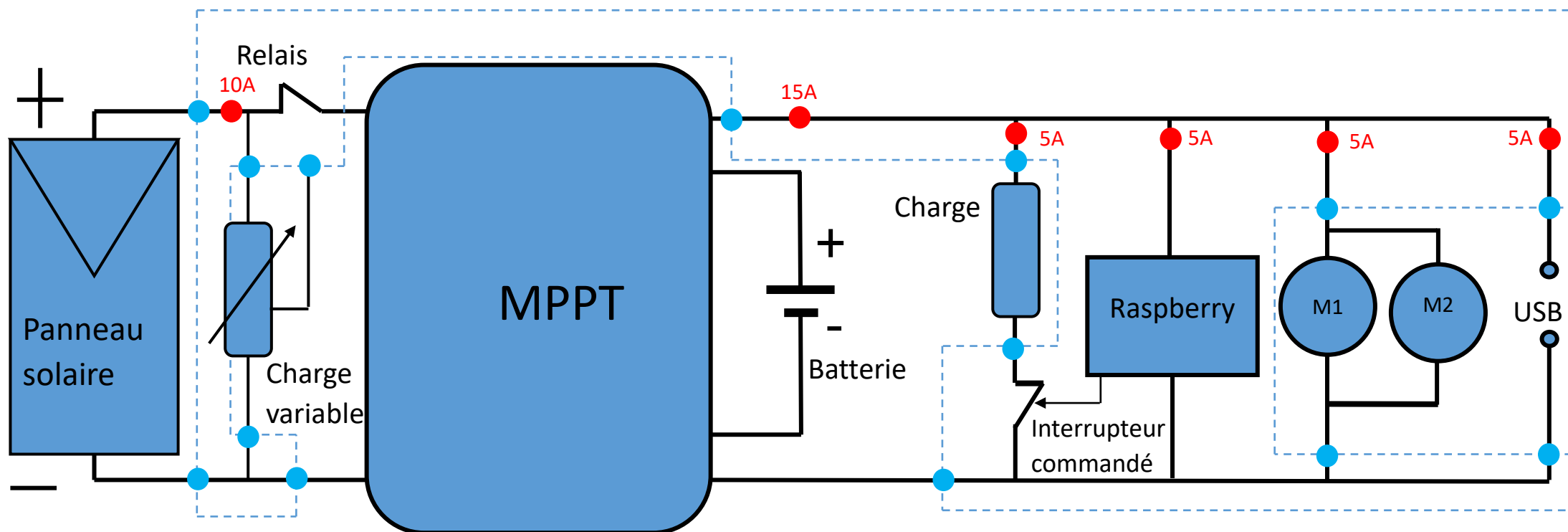
Figure 16 : Schéma d'un système solaire basique

Le schéma suivant est celui que nous avons obtenu après avoir apporté les modifications (ajout de la carte électronique : points de mesure et mini-ordinateur).

Nous utilisons un mini-ordinateur « Raspberry » qui permet de commander les moteurs et le système, mais aussi de relever la consommation et certaines grandeurs en temps réel. Celui-ci est directement alimenté par la batterie (notre carte permettra de moduler la tension en sortie de batterie afin d'alimenter le Raspberry en 3,3V). Notre système est donc totalement indépendant du réseau.

Nous retrouvons le même schéma principal mais avec des composants en plus.

Schéma du système final avec points de mesure :



- Carte électronique
- Connexion physique sur la carte
- Point de mesure du courant

Figure 17 : Schéma final du système solaire avec la carte de mesure et le mini-ordinateur

Dans ce schéma nous retrouvons les points de connexion physique sur la carte et les points de mesure du courant. Afin de relever les consommations nous allons relever des intensités de 5A, 10A et 15A avec un composant électronique (partie suivante).

VII.2. Création du schéma de la carte électronique :

L'année précédente le groupe chargé du projet a réalisé et testé une carte électronique composée principalement du capteur **LTC 2946** qui va nous permettre de faire les différentes mesures.

Le capteur LTC 2946 est un capteur qui relève le courant traversant une résistance et la tension à ses bornes tout en comptant le temps avec une horloge externe.

Ce composant communique sur un BUS I²C (bus multi-maitre : SCL et SDA + masse), c'est un mode de communication pouvant permettre le transfert de beaucoup de données en un seul fil. Ce composant comporte une adresse I²C, nous pouvons faire communiquer 256 adresses différentes avec ce bus. Etant donné que nous avons besoin de 6 capteurs, ce mode de transmission est largement suffisant.

Détail des connexions du LTC2946 :

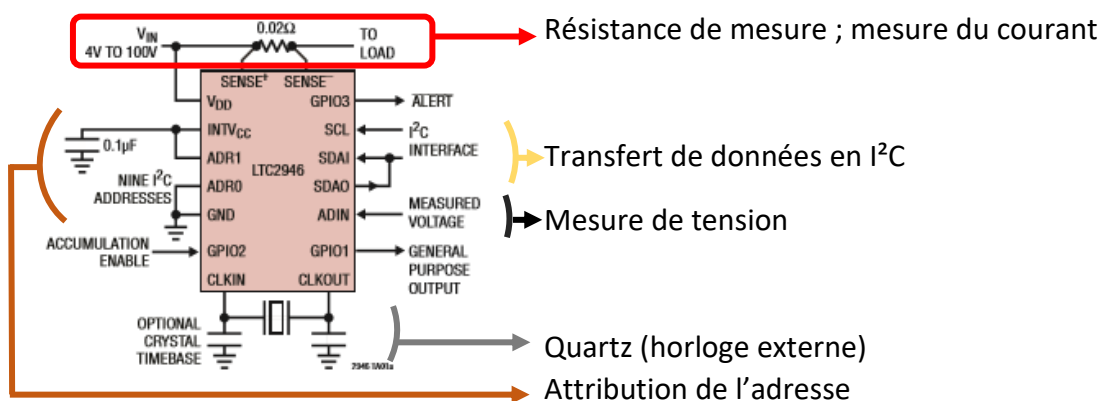


Figure 18 : Schéma des connexions du LTC2946

Le composant mesure en même temps le courant, la tension et les oscillations du quartz (horloge externe permettant de compter le temps) il est alors capable de calculer la puissance et l'énergie avec les formules suivantes : $P=U \times I$ et $E=P \times T$.

Tout de même nous devons adapter ce composant pour notre étude car dans la documentation du constructeur il est précisé que le capteur fonctionne avec une résistance de base de 0.02Ω ce qui permet de mesurer une intensité maximum de 5,12A.

Nous devons donc étudier le fonctionnement interne du composant afin de savoir comment l'adapter à notre projet. Pour cela nous avons dû nous référencer à la documentation fournie par le constructeur.

À la suite de cela nous avons étudié par le calcul la démarche à suivre afin d'obtenir des données pouvant être utilisées.

Schéma interne du LTC2946 :

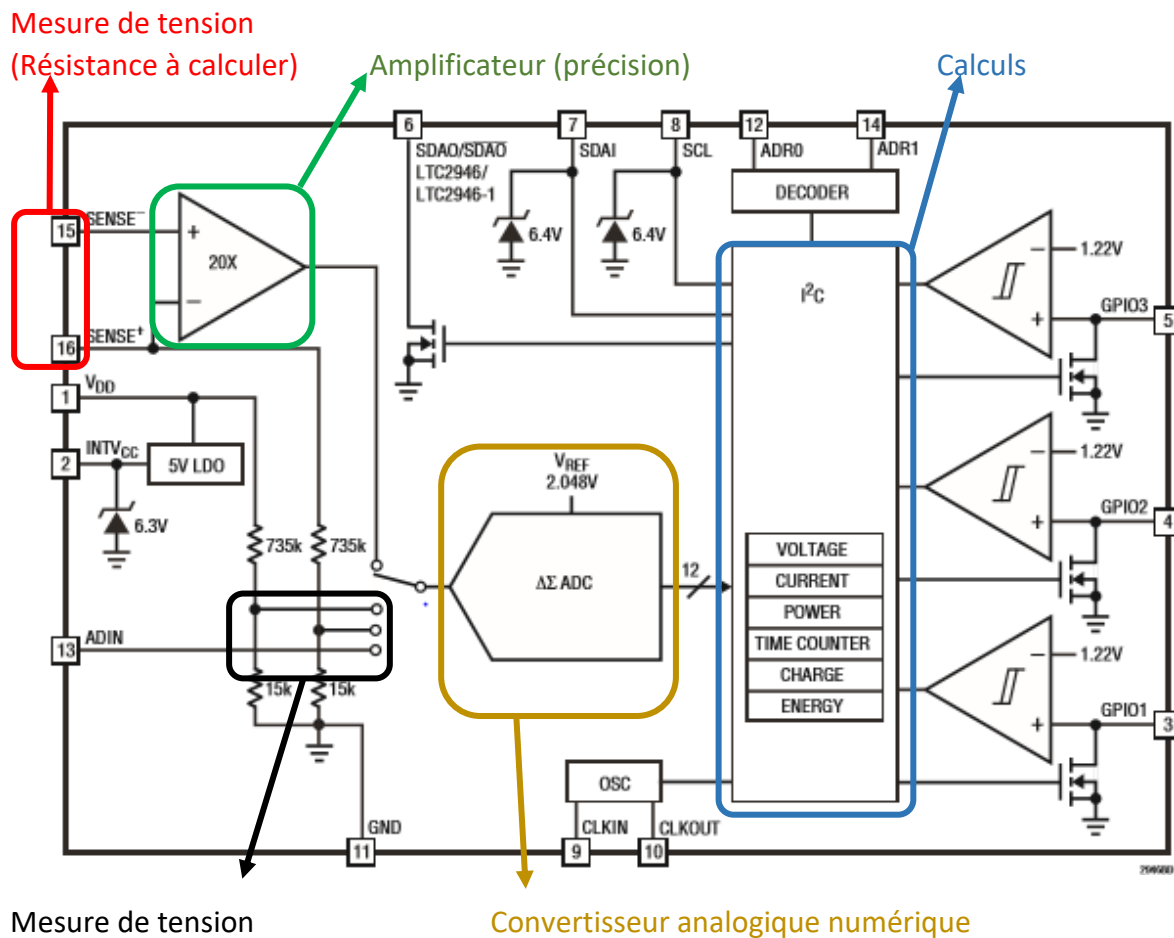


Figure 19 : Fonctionnement interne du LTC2946

Ce schéma nous montre que la valeur du courant (15-16) est mesurée puis amplifiée par 20, en parallèle la mesure de tension est effectuée et divisée à l'aide d'un pont diviseur de tension (valeur de tension max : $V_{max} = 102,4V$). Ensuite ces signaux analogiques sont transformés en signaux numériques avant d'être traités.

Afin de mesurer la grandeur voulue il nous est nécessaire de :

- Dimensionner une résistance par rapport au courant qui la traverse, nous allons donc regarder l'intensité max que peut demander la charge et adapter la résistance nécessaire. Enfin nous choisisons le boîtier adapté en fonction de la puissance dissipée par la résistance ($P=RxI^2$).
- Ensuite nous devons attribuer une adresse au composant afin de pouvoir les différencier les uns des autres et communiquer avec eux.
L'attribution d'une adresse à un composant se fait par le type de connexion sur les PINs d'adressages (ADR0 et ADR1) : « High » (3,3V) ; « LOW » (GND) et « NC » (NO CONNECT)

Calcul des résistances nécessaires afin d'effectuer la mesure de courant :

Dans un premier temps nous savons que la valeur mesurée est multipliée par 20 et la tension du convertisseur numérique analogique est de 2.048V (Vref)

Ainsi en entrée d'amplificateur nous avons : $2,048/20 = 0,1024 \text{ V}$

Ensuite avec la relation $R=U/I$ nous pouvons retrouver les valeurs des résistances.

Le groupe de l'année précédente a réalisé une carte permettant de mesurer une intensité allant jusqu'à 10 A.

Ils ont réalisé la carte suivante :

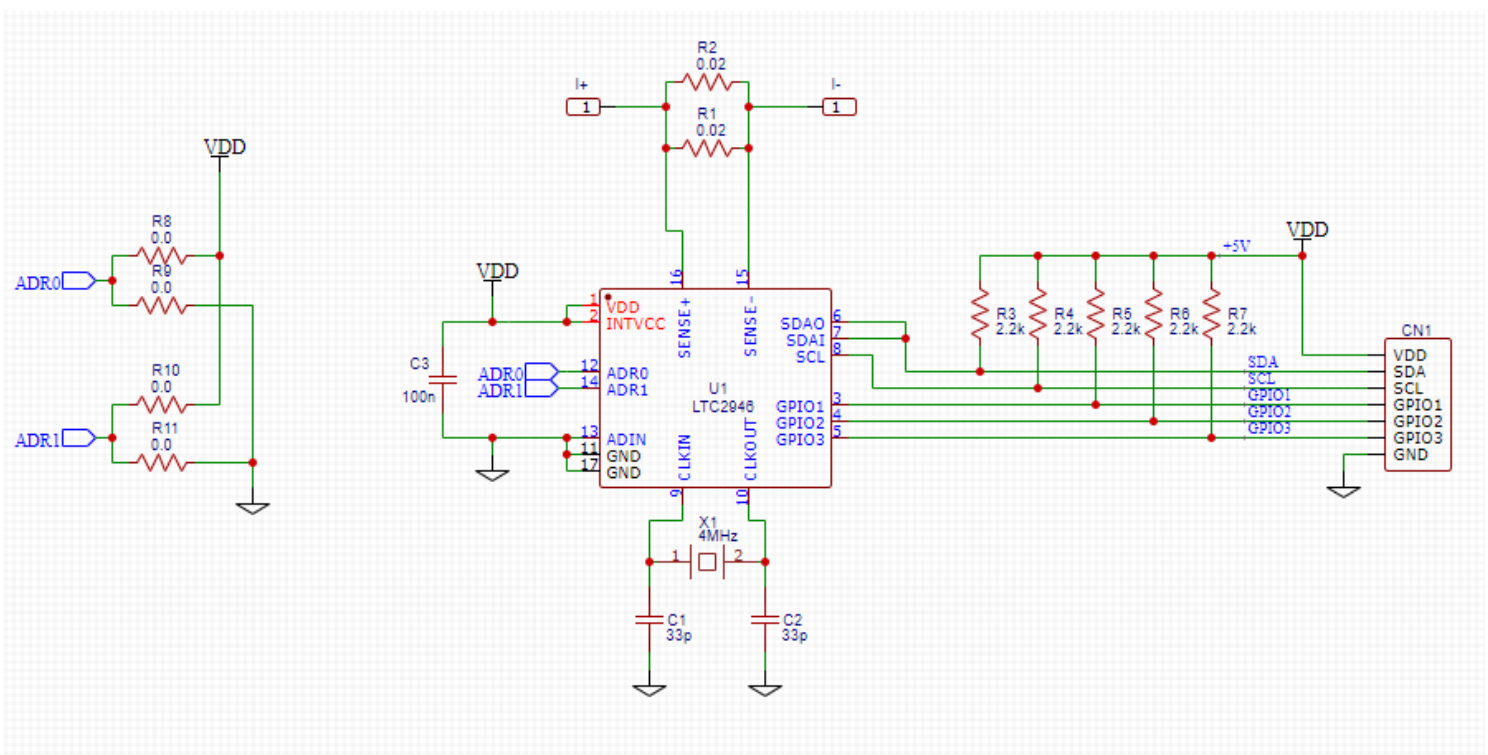


Figure 20 : Schéma électronique du LTC2946

Sur ce schéma électronique nous retrouvons le capteur vu précédemment avec une valeur de résistance équivalente de 0.01Ω qui permet de mesurer 10A

$$R_{eq} = R1 \times R2 / (R1 + R2)$$

Nous devons donc calculer des valeurs de résistance pouvant mesurer 15A (la valeur de la résistance de 5A étant celle donnée dans la documentation du constructeur $0,02\Omega$ et pour 10A $0,01\Omega$)

Donc pour 15A nous avons : $R = U/I = 0,1024/15 = 6\text{m}\Omega$

Une fois les résistances des composants trouvées nous devons calculer la puissance à dissiper par les composants, pour cela nous utilisons la formule $P=R \times I^2$:

Nous trouvons : - Pour 5A : $P=0,002 \times 5^2 = 0,25W$
 - Pour 10A : $P=0,001 \times 10^2 = 2,2W$
 - Pour 15A : $P=0,006 \times 15^2 = 1,35W$

Pour dissiper la puissance nous devons choisir des résistances ayant une puissance maximum émise supérieure à la puissance max pouvant la traverser (se référer à la documentation du constructeur). Si la puissance la traversant est supérieure nous pouvons modifier le boîtier de base (0805) en prenant un boîtier plus grand qui permettra de dissiper plus de chaleur, par exemple pour la résistance de 6mΩ nous avons choisi un boîtier plus grand (format 2512). Nous pouvons encore mettre des résistances en parallèle afin de diviser par 2 le courant les traversant et donc créer 2 points de dissipation (ex : 2x20mΩ).

Dans un second temps, une fois les mesures pouvant être réalisées nous devons paramétrer l'adressage de chaque composant, pour cela nous utilisons la documentation du constructeur :

Tableau des adressages de la documentation du constructeur :

Table 1. LTC2946 Device Addressing

DESCRIPTION	HEX DEVICE ADDRESS	BINARY DEVICE ADDRESS								LTC2946 ADDRESS PINS	
	h	a6	a5	a4	a3	a2	a1	a0	R/W	ADR1	ADR0
Mass Write	CC	1	1	0	0	1	1	0	0	X	X
Alert Response	19	0	0	0	1	1	0	0	1	X	X
0	CE	1	1	0	0	1	1	1	X	H	L
1	D0	1	1	0	1	0	0	0	X	NC	H
2	D2	1	1	0	1	0	0	1	X	H	H
3	D4	1	1	0	1	0	1	0	X	NC	NC
4	D6	1	1	0	1	0	1	1	X	NC	L
5	D8	1	1	0	1	1	0	0	X	L	H
6	DA	1	1	0	1	1	0	1	X	H	NC
7	DC	1	1	0	1	1	1	0	X	L	NC
8	DE	1	1	0	1	1	1	1	X	L	L

Adresse en hexadécimale

Connexion réelle à appliquer sur le composant

Figure 21 : Tableau des adressages du LTC2946

Les adresses en hexadécimale vont nous servir à les distinguer avec le Raspberry puis à récupérer les données fournies par les composants via le Bus I²C.

Les connexions ADR1 et ADR0 sont les PINS attribuées pour paramétrer l'adresse du composant, nous pouvons donc paramétrer au maximum 8 adresses différentes sur les composants.

Une fois que les différentes caractéristiques précédemment expliquées sont paramétrées nous avons mis au point un tableau récapitulatif dans lequel nous retrouvons toutes ces données regroupées, ces données seront utiles pour la création du schéma électronique final (Voir [ANNEXE 2.1](#)).

Tableau récapitulatif :

	LTC2946 n°1	LTC2946 n°2	LTC2946 n°3	LTC2946 n°4	LTC2946 n°5	LTC2946 n°6
Charge	PV	MPPT	Charge	Raspberry	2 Moteurs	USB
Adresse hexadécimale	CE	D0	D2	D4	D6	D8
Connection ADR1/ADR0	H/L	NC/H	H/H	NC/NC	NC/L	L/H
Intensité mesurée	10A	15A	5A	5A	5A	5A
Résistance	2x20mΩ en //	6m Ω	20m Ω	20m Ω	20m Ω	20m Ω
Puissance à dissiper	2,2W	1,35W	0,25W	0,25W	0,25W	0,25W
Boîtier du composant	1206	2512	1206	1206	1206	1206
Puissance dissipée par 1 résistance	1,5W	2W	1,5W	1,5W	1,5W	1,5W

Ensuite notre système étant alimenté par une batterie nous devons intégrer un convertisseur interne permettant de convertir la tension de 12V en tension de 3,3V, pour cela nous utilisons le composant ADP2303 :

Schéma électronique de l'alimentation :

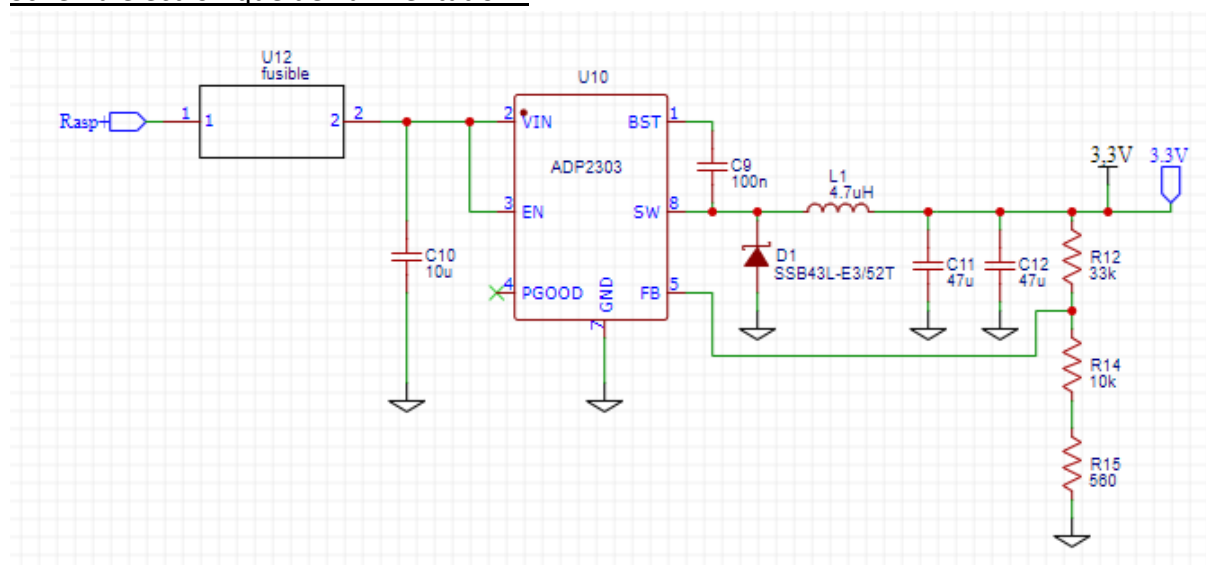


Figure 22 : Schéma électronique du ADP2303

Nous avons vu précédemment comment relever les grandeurs demandées (capteur LTC2946) et comment rendre le système autonome (alimentation de la carte), il nous reste à mesurer les températures.

Pour cela nous utilisons des capteurs de températures fonctionnant de 3V à 5,5V, des DS1820.

VDD est la PIN d'alimentation et GND la masse, DQ est le signal de sortie. Dans notre schéma nous allons lire sur ce composant la température. Or nous ne pouvons pas connecter la pâte du composant directement sur notre Raspberry (mini-ordinateur), il faut que nous ajoutons une résistance en de 4,7k Ω comme il l'est recommandé sur la documentation constructeur :

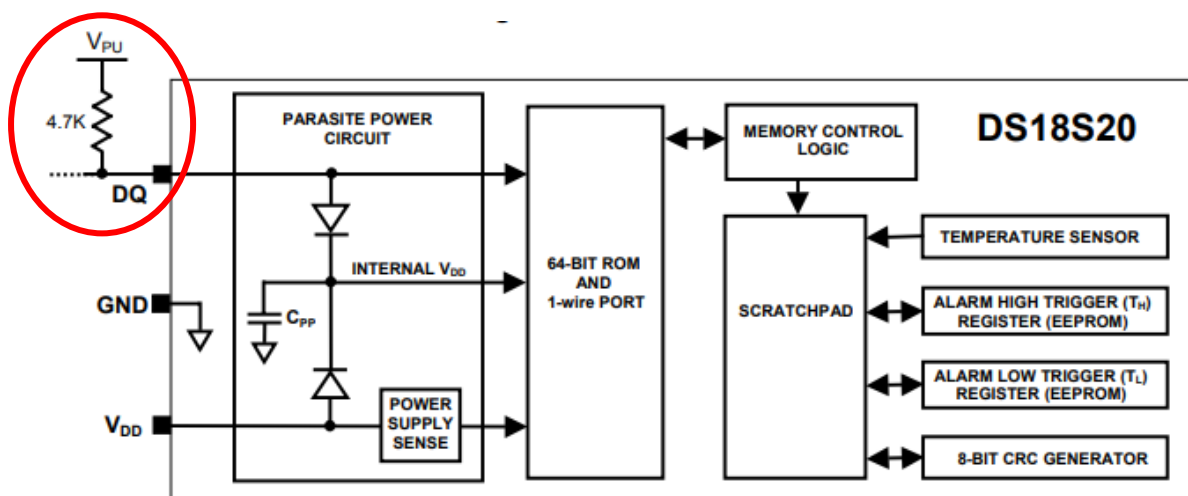
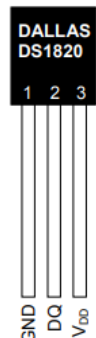


Figure 23 : Fonctionnement interne du DS1820

Avec toutes ces données nous avons mis au point le schéma de la carte électronique situé en annexe ([ANNEXE 2.1](#)).

VII.3. Création de la carte avec le logiciel EasyEDA.

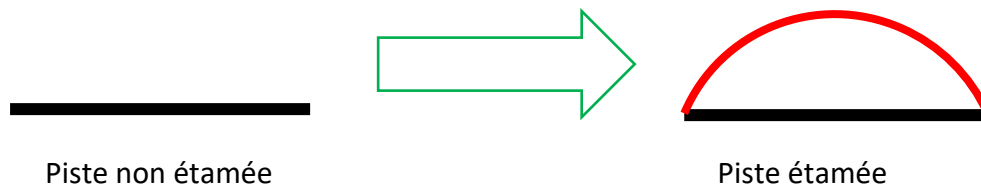
EasyEDA est un logiciel qui permet de modéliser une carte électronique à partir d'un schéma. La partie précédente nous a permis de réaliser cette carte, maintenant il faut réaliser la partie routage (PCB).

Le routage est l'étape réalisée avant la réalisation de la carte qui est faite par une entreprise.

Afin de réaliser ce routage il faut créer ou alors trouver l'empreinte réelle des composants dans la banque mise à disposition des utilisateurs.

L'empreinte d'un composant est très souvent détaillée dans la documentation du constructeur.

De plus ayant une carte qui reçoit de la puissance il nous faut dimensionner la largeur des pistes. Le principe pour passer de la puissance sur une piste est d'ajouter un gros apport en étain sur la piste afin de la faire grossir et obtenir une plus grande surface de passage pour le courant :



Calcul de la largeur des pistes :

On utilise un produit en croix : on sait que pour 1 ampère il nous faut une surface de 1mm^2 .

Donc par exemple pour 10 Ampères il nous faut 10mm^2 donc une piste de 5mm de large et 2mm de haut ou plus car on ne peut obtenir un rectangle parfait.

Une fois la largeur des pistes calculée il faut vérifier si nous avons des contraintes propres aux composants (données trouvables dans la documentation du constructeur) par exemple pour le capteur de courant (LTC2946) il est fortement conseillé de tenir le plus proche possible le quartz des pattes.

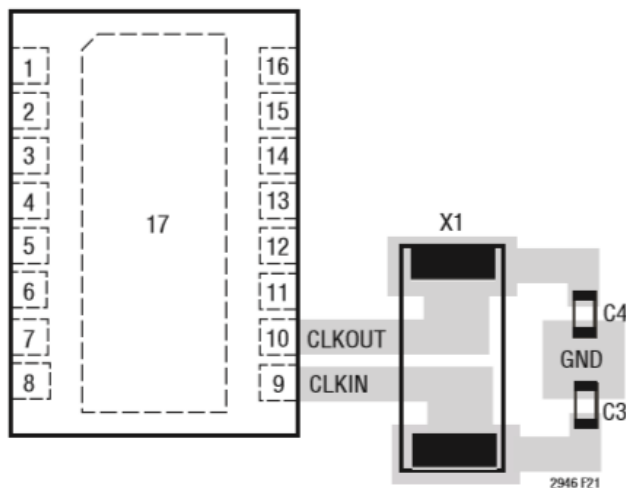


Figure 21. Recommended Layout for Crystal Oscillator

Figure 24 : Schéma de routage conseillé

Après avoir vérifié et entré les empreintes des composants sur le logiciel ainsi que la taille des pistes à respecter et les contraintes, nous avons réalisé le PCB suivant :

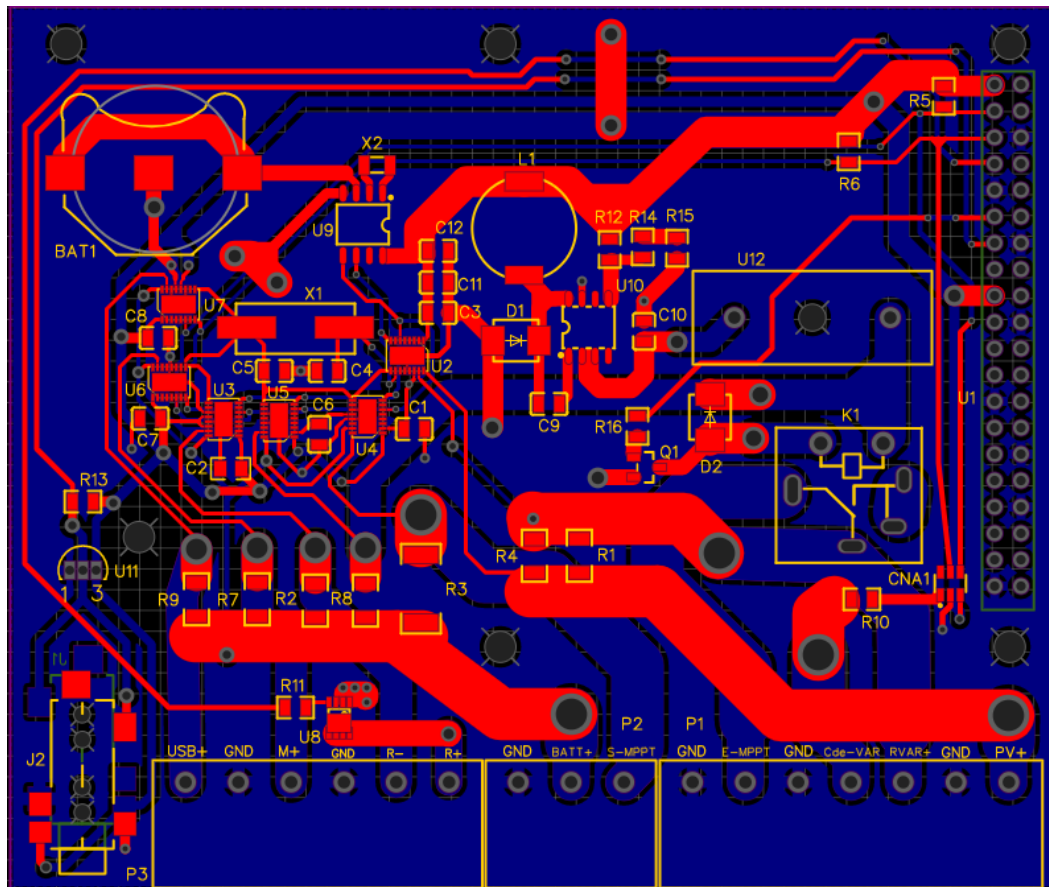


Figure 25 : Vue de la carte en mode PCB

Sur ce PCB nous retrouvons :

- les 6 capteurs (U2, U3, U4, U5, U6, U7) placés autour du quartz (X1).
- l'alimentation avec le ADP2303 (U10), la bobine L1 et le fusible U12.
- Le relais K1
- Les capteurs de température déportés dans le panneau et le boîtier (prise jack J1 et J2), ainsi que celui placé sur la carte U11.
- Les ports permettant de relier le panneau et la carte P1, P2 et P3.

Ainsi que d'autres composants permettant au système de fonctionner, pour plus de détails voir nomenclature [ANNEXE 2.2](#)

Après la conception du PCB nous pouvons avoir une vue 3D de la carte ([ANNEXE 2.4](#)) puis une fois validée nous avons envoyé la carte à une entreprise qui l'a réalisée puis qui nous l'a envoyée.

Nous avons reçu la carte dans la période précédant l'épidémie du COVID-19, nous n'avons donc pas pu souder les composants sur la carte et faire les tests (carte en [ANNEXE 2.5](#)). Cette partie sera probablement traitée l'année suivante ou alors pendant la période suivant la remise des comptes rendus de projet.

VIII. Conception 3D d'un boîtier adapté pour les cartes électroniques

Dans cette partie, nous présenterons la conception du boîtier accueillant les cartes électroniques sur le système comprenant le panneau solaire, les moteurs et le MPPT.

L'année précédente, le groupe chargé du projet a eu pour objectif de réaliser sur SolidWorks (logiciel de conception 3D) l'ensemble du système permettant au panneau solaire de bouger avec le panneau (voir compte rendu année 2018-2019).

Une fois la carte réalisée nous avons dû chercher un boîtier pouvant contenir les deux cartes. Mais un boîtier sur-mesure étant trop onéreux nous avons opté pour un boîtier imprimé en 3D. Nous avons donc réalisé le boîtier suivant :

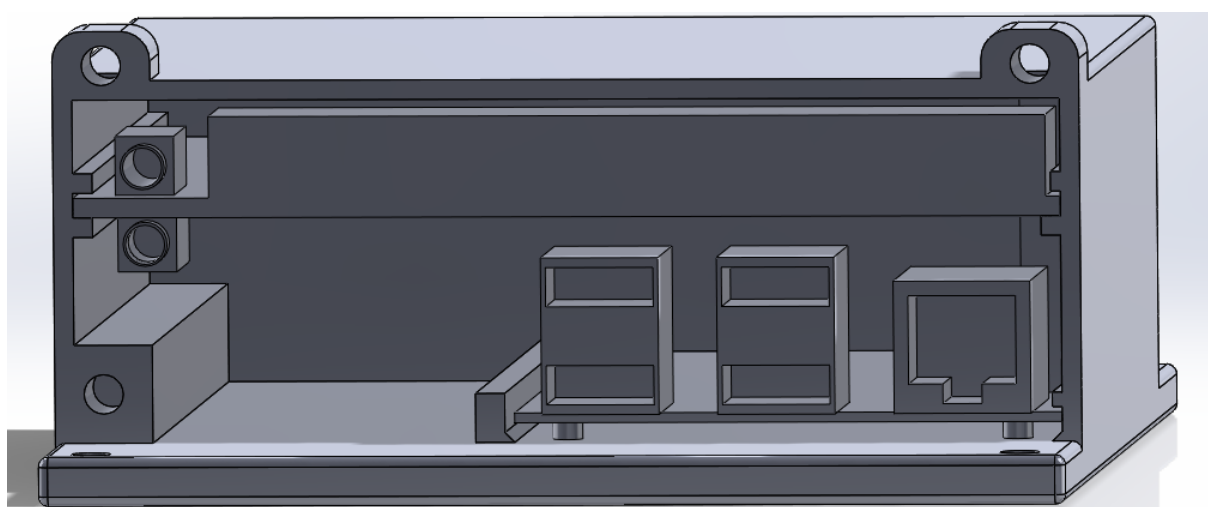


Figure 26 : Assemblage de la modélisation du boîtier avec les cartes

Le boîtier est muni de 2 glissières permettant un maintien des cartes dans le boîtier ainsi que 3 trous pour y visser le couvercle et 4 trous sur les extrémités du socle afin de le visser au boîtier du panneau solaire.

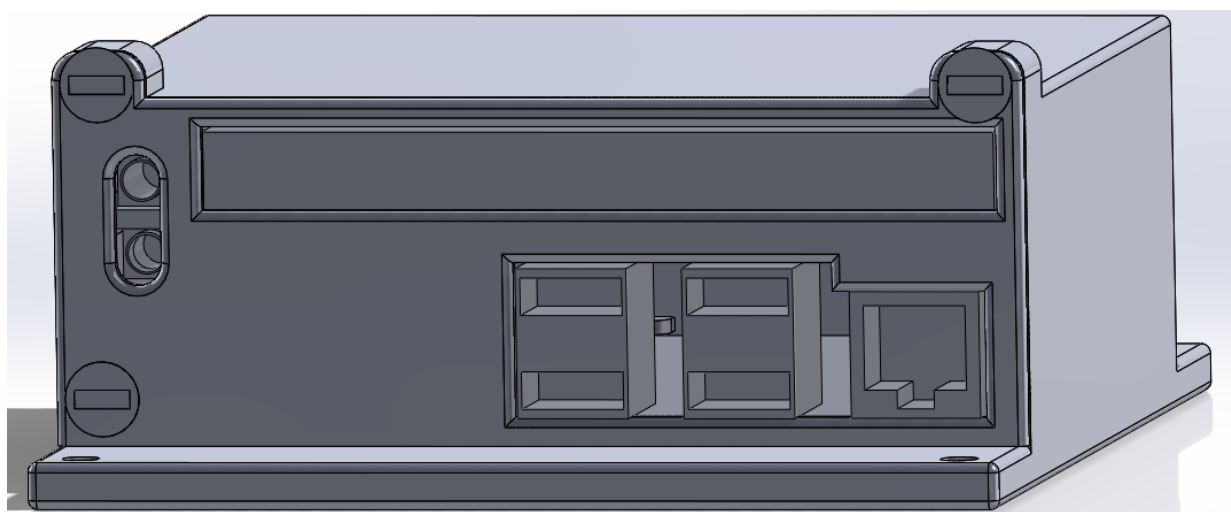


Figure 27 : Assemblage du boîtier

Une fois le boîtier réalisé nous l'avons intégré dans le boîtier du panneau solaire afin de voir s'il rentre sans gêner le passage des câbles et le pivotement du panneau solaire en fin de course.

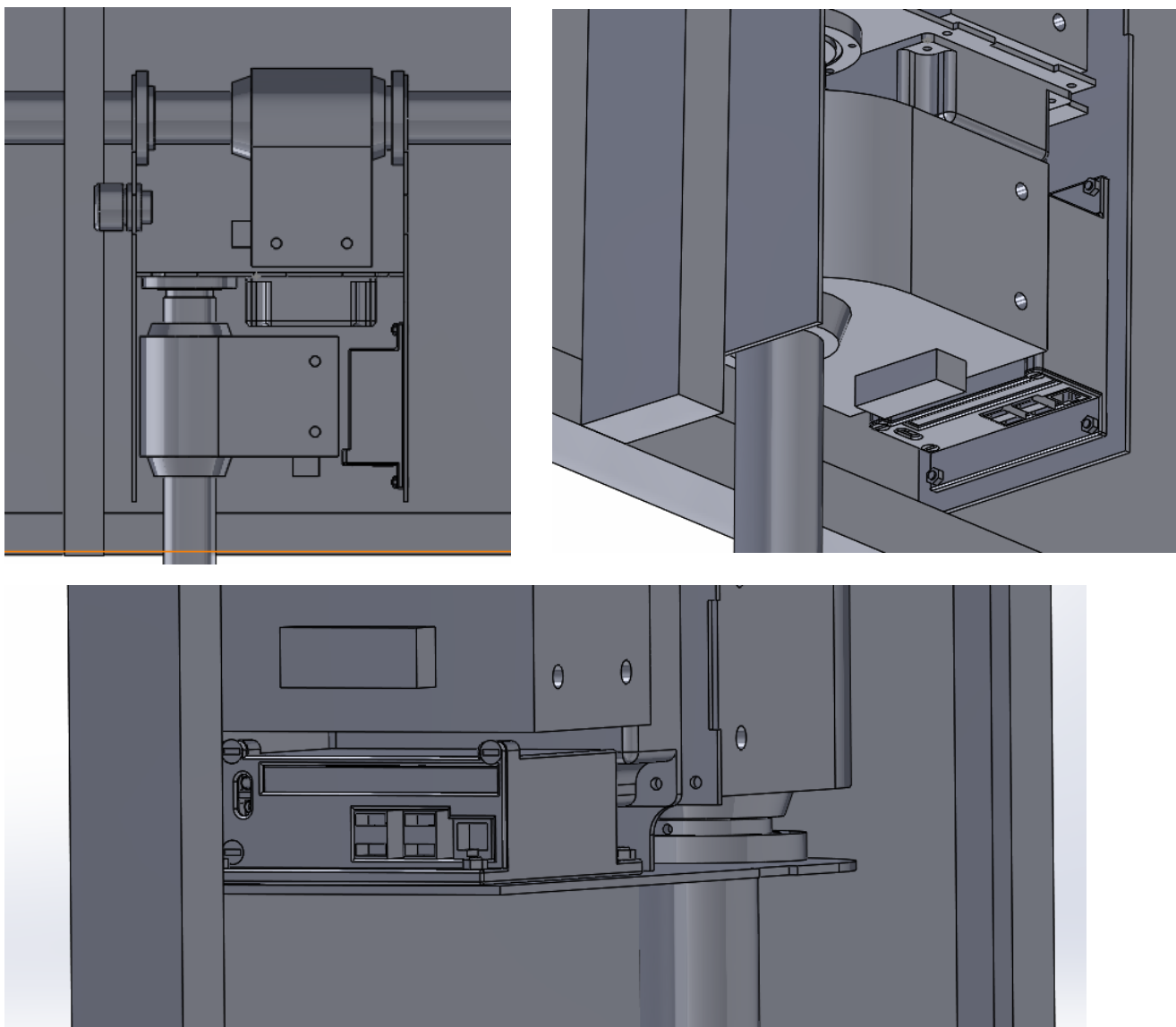


Figure 28 : Vues de l'assemblage final du système

Ces différentes vues nous permettent de constater que notre boîtier s'adapte parfaitement dans le système.

Tout de même ce boîtier est seulement un prototype, il faudrait l'imprimer et faire des essais et si nécessaire l'adapter.

VIV. Conclusion :

Nous n'avons pas pu durant ces semaines de projet réaliser la carte électronique et les tests, cela est dû principalement à l'épidémie ayant frappé le monde durant cette année 2020.

Il serait donc possible pour le groupe de l'année suivante de finir la carte, le boîtier et essayer de faire une étude du système en fonctionnement si cela n'a pas été fait durant la période de stage suivant les projets.

De plus il serait intéressant de calibrer les grandeurs mesurées. Pour cela il faudrait placer un ampèremètre en sortie du module MPPT (ANNEXE 2.3) et calculer un coefficient entre la valeur de l'ampèremètre et le capteur situé après le module MPPT. Ensuite il faudrait multiplier toutes les autres valeurs de courant par ce coefficient afin d'obtenir les valeurs réelles.

À la suite du travail réalisé nous avons mis en place un TP qui pourrait être réalisable en DUT, sur le thème de la programmation ou encore de l'étude énergétique.

Ce projet nous aura permis de nous améliorer en plusieurs points, que ce soit au niveau de l'organisation et du travail en équipe, de nos compétences en électronique, en programmation et en supervision. Mais aussi au niveau de notre capacité d'adaptation malgré les événements survenus cette année.

Nous avons apprécié travailler sur le projet du tracker solaire tout au long de l'année car non seulement nous trouvons ce projet intéressant, mais nous avons aussi bénéficié d'une aide précieuse venant de nos tuteurs lorsque nous étions bloqués.

TABLE DES RÉFÉRENCES

Figure 1 : Schéma simplifié de l'installation	5
Figure 2 : Diagramme de Gantt	8
Figure 3 : Programme de mesure de tension et courant	9
Figure 4 : Programme de mesure d'énergie	10
Figure 5 : Programme de mesure de puissance	10
Figure 6 : Bouton interactif et programme	11
Figure 7 : Utilisation d'un popover	12
Figure 8 : Programme de la variable tableau	12
Figure 9 : Programme création du popover et affichage	13
Figure 10 : Programme du lancement du popover (tutoriel)	13
Figure 11 : Fonction du popover	14
Figure 12 : Programme des interactions sur le popover	14
Figure 13 : Programmation de l'affichage d'une bulle informative	15
Figure 14 : Page source de github	16
Figure 15 : Syntaxe sur sphinx	16
Figure 16 : Schéma d'un système solaire basique	18
Figure 17 : Schéma final du système solaire avec la carte de mesure et le mini-ordinateur ..	19
Figure 18 : Schéma des connexions du LTC2946	20
Figure 19 : Fonctionnement interne du LTC2946	21
Figure 20 : Schéma électronique du LTC2946	22
Figure 21 : Tableau des adressages du LTC2946	23
Figure 22 : Schéma électronique du ADP2303	24
Figure 23 : Fonctionnement interne du DS1820	25
Figure 24 : Schéma de routage conseillé	26
Figure 25 : Vue de la carte en mode PCB	27
Figure 26 : Assemblage de la modélisation du boîtier avec les cartes	28
Figure 27 : Assemblage du boîtier	28
Figure 28 : Vues de l'assemblage final du système	29

ANNEXE 1

Annexe 1.1. Programme python de mesure sur putty

PROGRAMME MESURE DE TEMPÉRATURE :

```
nano temperature.py
from os import system
from time import sleep
##module GPIO 1-wire et capteur de température##
system('modprobe wl-gpio')system('modprobe wl-therm')

## chemin vers les sondes ##
base_dir = '/sys/bus/w1/devices/'

## Remplacez les répertoires 28-xxxxxxxxxxx ##
## par vos propres répertoires. ##
## Et si vous avez un nombre de sonde différent ##
## supprimer (ou ajouter) les lignes ci-dessous ##

sonde1 = "/sys/bus/w1/devices/w1_bus_master1/28-000003f8fb09/w1_slave"
## et ajuster aussi les 2 lignes ci-dessous ##

Sondes = [sonde1]
sonde_value = [0]
## fonction ouverture et lecture d'un fichier ##

def lire_fichier(fichier):
    f = open(fichier, 'r')
    lignes = f.readlines()
    f.close()
    return lignes
## code principal ##

for (i, sonde) in enumerate(sondes):
    lignes = lire_fichier(sonde)
    while lignes[0].strip()[-3:] != 'YES': # lit les 3 derniers char de
la ligne 0 et recommence si$
        sleep(0.2)
    lignes = lire_fichier(sonde)
    temp_raw = lignes[1].split("=")[1] # quand on a eu YES, on lit la
temp apres le signe = sur la $
## le 2 arrondi a 2 chiffres après la virgule ##

    sonde_value[i] = round(int(temp_raw) / 1000.0, 2)

## affichage a l'écran ##
    print "sonde",i,"=",sonde_value[i]
```

TEST MESURE DE TENSION ET COURANT DANS UN INTERVALLE DE TEMPS :

`nano boucle.py`

```
import time
import smbus

n = 0
while n <= 20:
    n = n+1
    (start_time) = (time.time())
    bus = smbus.SMBus (1)
    address = 0x6f
    VinMSB= (bus.read_byte_data(address, 0x1e))*16
    VinLSB= (bus.read_byte_data(address, 0x1f))/16
    print (VinMSB+VinLSB)*float(0.025)
    time.sleep(0.02)
    print ("Temps d execution :%s s" %(time.time() - start_time))
    (start_time) = (time.time())
    senseMSB= (bus.read_byte_data(address, 0x14))*16
    senseLSB= (bus.read_byte_data(address, 0x15))/16
    print (senseMSB+senseLSB)*0.0025
    #print ("Temps d execution :%s s" %(time.time() - start_time))
```

##Le but de cette boucle est d'effectuer 20 mesures de tension - courant avec un intervalle de temps de 0.02sec dans ce cas. Cela permet de connaître le temps de rafraichissement des mesures.##

COMMANDE DE LA CHARGE VARIABLE :

```
nano cna2.py
```

```
import smbus  
bus = smbus.SMBus (1)  
address = 0x60  
data = 2500  
bus.write_word_data(address, data//256, data)  
bus.close()
```

```
##Cette partie permet d'incrémenter une valeur au transistor  
pour changer la tension et le courant. La valeur maximale est de  
4095 (environ 4.7V).##
```

CHANGEMENT D'HEURE EXTERNE :

```
##Commande pour avoir un accès administrateur##  
pi@raspberrypi:~ $ sudo bash
```

```
##Callage de l'heure sur l'horloge système définie auparavant avec  
la commande sudo date « date »##
```

```
root@raspberrypi:/home/pi# sudo hwclock -w
```

```
##Pour lire la valeur de l'horloge externe##  
root@raspberrypi:/home/pi# sudo hwclock -r
```

REMISE A ZÉRO DE L'ÉNERGIE :

```
nano raz.py
```

```
import smbus  
import time  
bus = smbus.SMBus (1)  
address = 0x6f
```

```
##Effacement de l'ancienne valeur##  
(bus.write_byte_data(address, 0x01, 2))
```

```
##Remplacement par 0##  
(bus.write_byte_data(address, 0x01, 0))
```

```
##Lecture de l'énergie pour vérification du reset##  
result= (bus.read_byte_data(address, 0x01))  
print result
```

Annexe 1.2. Commandes utiles sur putty :

History : *pour voir les commandes effectuées précédemment*

Sudo i2cdetect -y 1 : *scan du raspberry*

ls -l : *tous les programmes existants*

tail -fn100 /var/www/pyscada/PyScadaServer/pyscada_debug.log : *logs de pyscada (ctrl+c pour stopper).*

Annexe 1.3. Noms des programmes pour le script de pyscada :

nano scripttension.py

nano scriptcourant.py

nano scriptpuissance.py

nano scriptenergie.py

nano scripttemperature.py

Annexe 1.4. Programme définition de la variable tableau

```
Var tableau = [  
  ['element', 'title', 'container', 'viewport'],  
  ['a[href$="page1"]', 'Ceci est la page 1', '.navbar-collapse', '.navbar-collapse'],  
  ['span.input-group-addon:contains("V")', 'barre de temps', '.panel-body', 'span.input-group-addon:contains("V")']  
]
```

//Dans ce tableau nous allons définir les valeurs utiles pour la création des popovers.

//Chaque ligne correspond à un popover.

//Tableaux à compléter

```
var tableau_tool = [  
  ['element', 'title', 'content'],  
  ['.chart-save-csv', 'Téléchargement', 'content'],  
]
```

// Tableau à compléter

Annexe 1.5. Code tuto Pyscada :

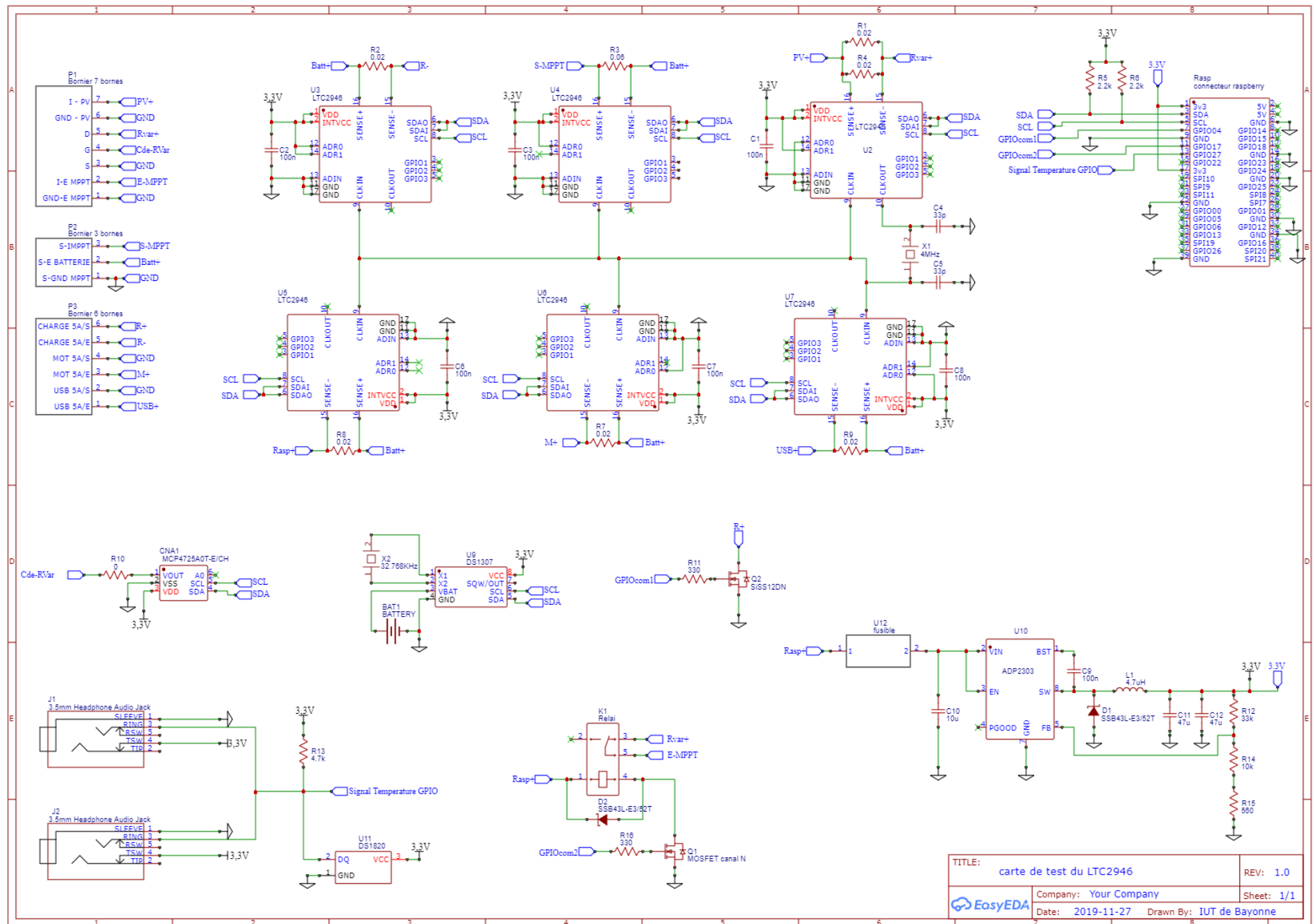
```
$('.navbar-right').prepend('<li class="dropdown"><!-- Tutoriel --><button type="button" id="start_tuto" class="btn btn-info btn-lg" data-container="body" data-placement="left" data-original-title="" title="">Start Tuto</button></li>')
var tableau = [
  ['element', 'title', 'container', 'viewport'],
  ['a[href$="page1"]', 'Ceci est la page 1', '.navbar-collapse', '.navbar-collapse'],
  ['span.input-group-addon:contains("V")', 'barre de temps', '.panel-body', 'span.input-group-addon:contains("V")']
]
if tableau.length > 1

for (let i = 1; i < tableau.length; i++) {

$(tableau[i][0]).popover({
  title: tableau[i][1],
  html: true,
  content: contentHtml,
  container: tableau[i][2],
  viewport: tableau[i][3],
  trigger: 'manual'

}).on('shown.bs.popover', function (eventShown) {
  var $popup = $('.navbar-collapse #' + $(eventShown.target).attr('aria-describedby'));
  $popup.find('button .close-popover').click(function (e) {
    $popup.popover('hide');
  });
  $popup.find('button .next-popover').click(function (e) {
    $popup.popover('hide');
    $('span.input-group-addon:contains("V")').popover('show');
  });
  $('#start_tuto').on("click", function() {
    $('a[href$="page1"]').popover('show')
  })
  '<div>',
    '<button><span class="glyphicon glyphicon-remove close-popover"></span></button>',
    '<button><span class="glyphicon glyphicon-chevron-right next-popover"></span></button>',
    '</div>'].join('\n');
  $popup.find('button .close-popover').click(function (e) {
    $popup.popover('hide');
  });
  $popup.find('button .next-popover').click(function (e) {
    $popup.popover('hide');
    if (tableau[i][4].length > 0){
      $(tableau[i][4]).popover('show');
    }
  })
var tableau_tool = [
  ['element', 'title', 'content'],
  ['.chart-save-csv', 'Téléchargement', 'content'],
  If (tableau_tool.length > 1){
  For (let i = 1; i < tableau_tool.length; i++) {
  $(tableau_tool[i][0]).tooltip({
    title:tableau_tool[i][1]
    html : true,
    content : tableau_tool[i][2],
  })
  $(tableau_tool[i][0]).tooltip('show')
  $(tableau_tool[i][0]).tooltip('hide')
  }}
}
```


Annexe 2.1. Schéma final

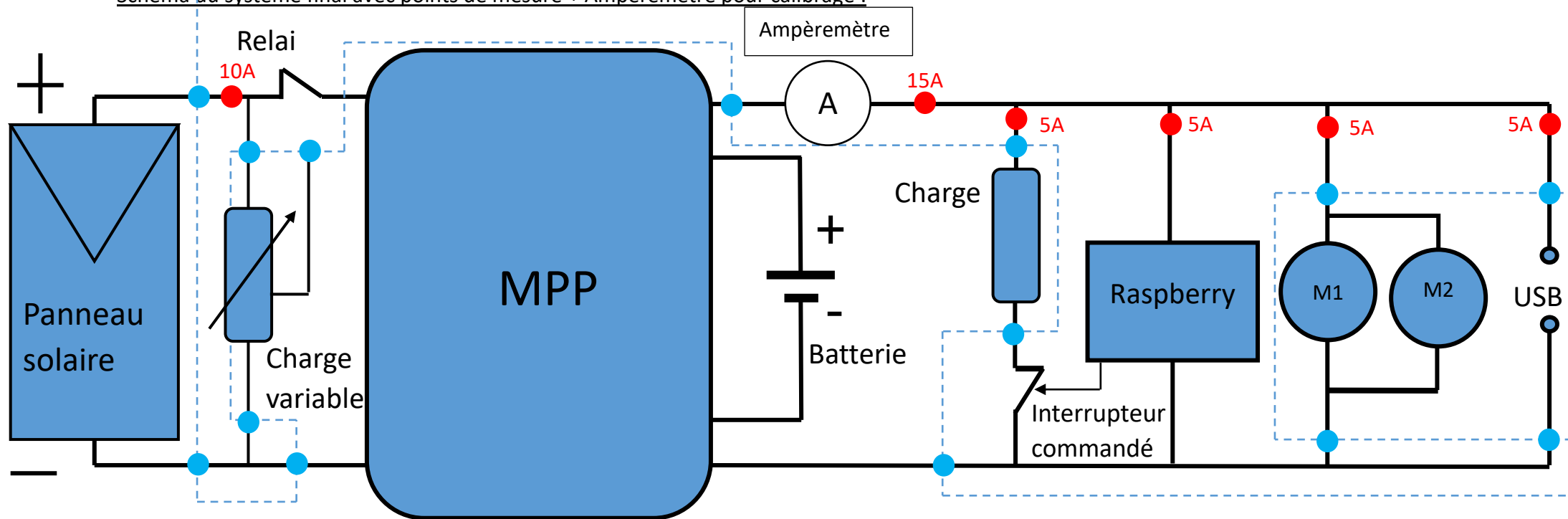


Annexe 2.2 Nomenclature

ID	Name	Designator	Footprint	Quantity	Manufacturer Part	Manufacturer	Supplier	Supplier Part
1	Bornier 7 bornes	P1	BORNIER 7 BORNES	1	790-1017	RS PRO	RS-Online	790-1017
2	4.7uH	L1	INDUCTANCE 4700	1	VLU1012-4R7MB	FH	Farnell	2144063
3	Relai	K1	PCB RELAY	1	CP1H12	PANASONIC	RS-Online	699-6008
4	Bornier 6 bornes	P3	BORNIER 6 BORNES	1	CTB9359/6	RS PRO	RS-Online	897-1272
5	MCP4725A0T-E/CH	CNA1	TSOT-23-6_L2.9-W1.6-P0.95-LS2.8-BR	1	MCP4725A0T-E/CH	MICROCHIP	RS-Online	669-6290
6	BATTERY	BAT1	PORTE PILE	1	866-0653	RS	RS-Online	185-4669
7	100n	C1,C3,C6,C7,C9,C8	C0805	6	08052R104K8BB	Yageo	RS-Online	378-665
8	33p	C4,C5	C0805	2	C0805C330J5GACTU	KEMET	RS-Online	264-4264
9	100n	C2	0805'	1	08052R104K8BB	Yageo	RS-Online	378-665
10	10u	C10	C0805	1	C2012X5R1E106K125AB	TDK	RS-Online	788-3035
11	47u	C11,C12	C0805	2	TDK	C2012X5R1A476M125AC	RS-Online	788-3029
12	DS1307	U9	SOP-8_L4.9-W3.9-P1.27-LS6.0-BL	1	DS1338Z-33+	MAXIM	RS-Online	732-7368
13	DS1820	U11	PR35	1	DS18S20+	Maxim	RS-Online	540-2849
14	SSB43L-E3/52T	D1,D2	DO-214AA_L5.6-W4.0-RD	2	SSB43L-E3/52T	Vishay Intertech	RS-Online	700-0990
15	2.2k	R6,R5	R0805	2	ERJP06F2201V	Panasonic	RS-Online	721-7775
16	0.02	R1,R4,R2,R8,R7,R9	R1206	6	VMK-R020-1.0-U	Isabellenhutte	RS-Online	192-9965
17	0.06	R3	R2512	1	TLR3A20DR006FTDG	TE connectivity	RS-Online	771-7828
18	330	R11,R16	R0805	2	ERJP06F3300V	Panasonic	RS-Online	721-7707
19	33k	R12	R0805	1	ERJP06F3302V	Panasonic	RS-Online	721-7864
20	10k	R14	R0805	1	RK73H2ATTD1002F	KOA	RS-Online	631-4939
21	560	R15	R0805	1	CRCW0805560RFKEA	Vishay	RS-Online	679-1547
22	0	R10	R0805	1	RK73Z2ATTD	KOA	RS-Online	631-3964
23	4.7k	R13	R0805	1	CR0805-FX-4701GLF	Bourns	RS-Online	740-9031
24	ADP2303	U10	SOIC-8_L4.9-W3.9-P1.27-LS6.0-BL	1	ADP2303ARDZ	Analog devices	RS-Online	759-2916
25	MOSFET SiSS12DN	U8	MOSFET SISS12DN	1	SiSS12DN-T1-GE3	Vishay	RS-Online	178-3920
26	LTC2946	U2,U4,U3,U5,U6,U7	DFN-16-1EP_3X4MM_PITCH0.45MM	6	LTC2946IMS#PBF	Analog devices	Mouser	584-LTC2946IMS#PBF
27	4MHz	X1	SMD QUARTZ MODIFIÉ	1	ABLS-4.000MHZ-B2-T	ABRACON	Farnell	1652553
28	32.768KHz	X2	PCB QUARTZ 32.768KHz	1	X1A000141000312	EPSON	RS-Online	904-7503
29	fusible	U12	PCB FUSIBLE	1	64900001039	Littelfuse	RS-Online	787-4164
30	MOSFET canal N	Q1	SOT-23-3_L2.9-W1.6-P1.90-LS2.8-BR	1	FDN337N	ON Semiconductor	RS-Online	671-0429
31	Bornier 3 bornes	P2	BORNIER 3 BORNES	1	CTB9352/3	RS PRO	RS-Online	897-1187
32	3.5mm Headphone Audio Jack	J2,J1	3.5MM HEADPHONE JACK 5 PIN RIGHT ANGLE	2	35RASMT4BHNTRX	Switchcraft Inc.	RS-Online	705-1490
33	Raspberry ports	U1	PCB RASPBERRY	1	ESQ-120-34-G-D	Samtec	RS-Online	180-3899

Annexe 2.3 Schéma final + Ampèremètre

Schéma du système final avec points de mesure + Ampèremètre pour calibrage :



----- Carte électronique



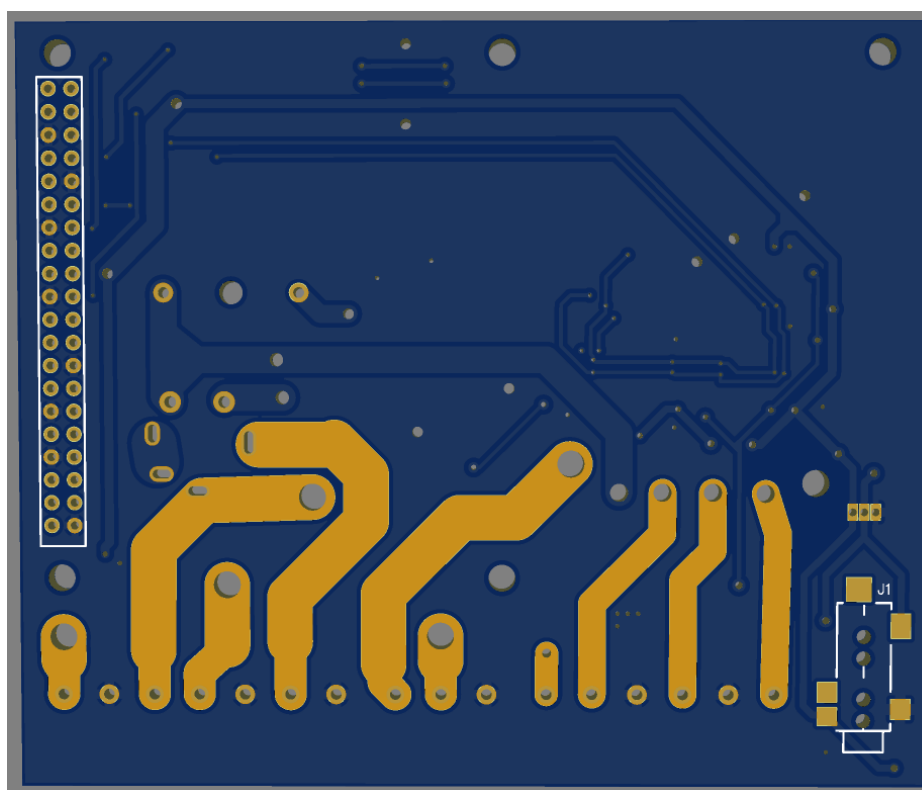
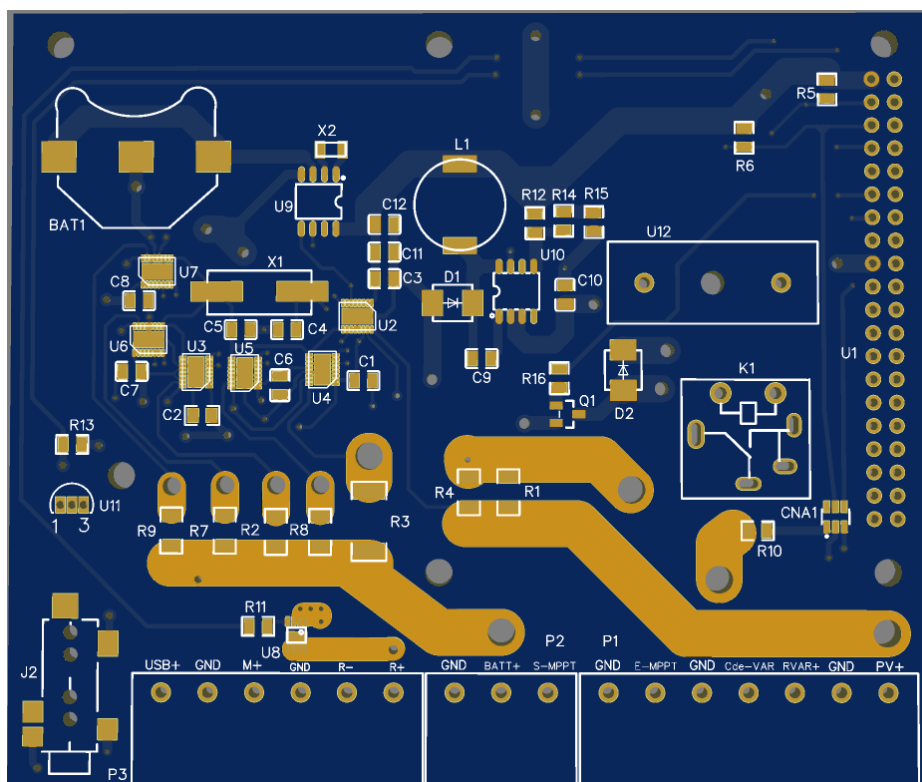
Connexion physique sur la



Point de mesure du courant

Annexe 2.4. Vue 3D de la carte et carte électronique reçue

Vue 3D de la carte électronique :



Carte électronique reçue :

