



# Alarme Moto Connectée

07/01/2021

---

MARU Théodros - ROUILLÉ Lucas

IUT Nantes

Module ER3-ER4

Carquefou - 44470



<b>Introduction</b>	<b>3</b>
Qu'est-ce que l'alarme moto connectée ?	4
Bête à corne	4
Synthèse du cahier des charges	5
Le brainstorming	5
Work Package	6
Les Risques	7
Emploi du temps prévisionnel	8
Calcul de budget théorique	10
Test des fonctions	12
Etude théorique du module GSM	18
Les dévolteurs	22
Dévolteur à découpage	28
Création de la carte sur Kicad	40
Fabrication du PCB	44
Code et test	47
Les problèmes rencontrés lors du projet	48
Réunion pendant le projet	52
Tableau de bord	52
Diagramme de Gantt	54
Calcul de budget réel	55
<b>Conclusion du projet ER4</b>	<b>55</b>
<b>Annexe:</b>	<b>56</b>
Code Accéléromètre:	56
Code GPS:	57
Code GSM:	59
Code fonction accéléromètre + Localisation:	61
Code finale:	64
<b>Ressources:</b>	<b>68</b>



## **Introduction**

Pendant notre DUT, nous avons pu étudier les différents points importants liés aux projets en matière CP2 et sur l'importance du recyclage Cycle de vie du produit (CP3). Le but de la matière ER3-ER4 est de mettre en pratique ce qu'on a pu voir lors des projets précédents (ER1 et ER2) . On va devoir travailler en binôme, on devra donc gérer notre temps vis-à-vis du planning, prendre en compte les risques afin de mieux s'y préparer et de les éviter et prendre également en compte les aléas probables. Il faudra aussi prendre en compte le budget même si le financement du projet est couvert par l'IUT, il est très intéressant de se comporter comme en condition réelle (comme en entreprise) et donc d'éviter un maximum de coût inutile. L'objectif sera de mettre en place un projet et de le réaliser en autonomie, le projet sera très peu guidé et donc il faudra faire preuve de réflexion afin d'avancer. En clair, nous allons devoir nous comporter de la même manière que si nous étions en entreprise, nous allons donc pour la première fois créer notre projet qui a pour nom : "Alarme moto connectée".

Ce projet consistera à faire une alarme moto qui informera le propriétaire si cette dernière est en envoyant par SMS la localisation de la moto. Tout cela sera fait à l'aide de quatre modules et d'un dévolteur qui nous permettra de brancher notre alarme directement sur la batterie de la moto. Le module Seeeduino sera le cerveau et contrôlera les trois autres. Le module GSM permet d'envoyer le SMS. Le module GPS permettra de connaître la localisation de la moto et l'accéléromètre de savoir si la moto est en train d'être déplacée. Des explications plus détaillées seront présentées par la suite.

## Qu'est-ce que l'alarme moto connectée ?

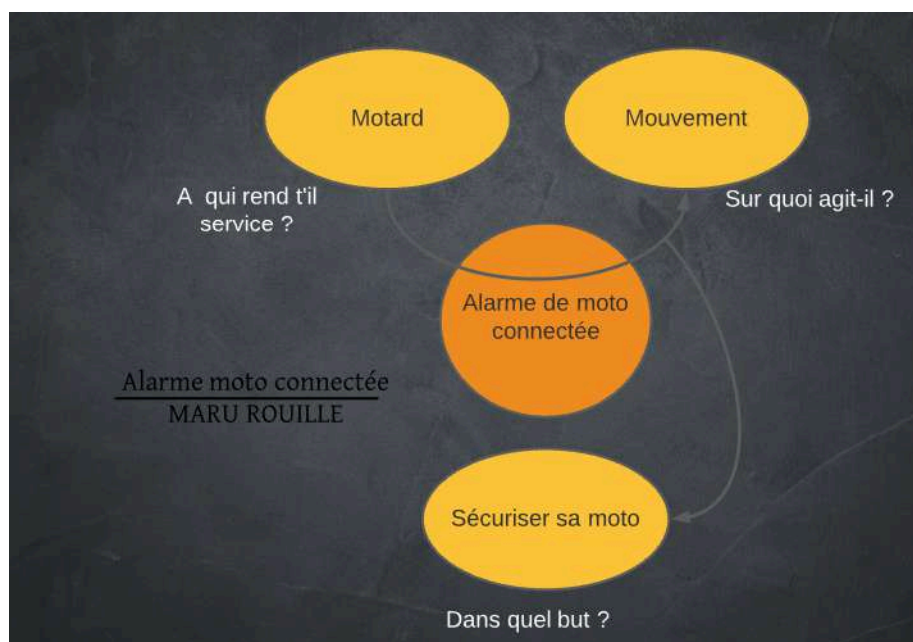
Tout d'abord, il est important de définir qu'est-ce qu'un objet connecté :

Un objet connecté est un objet qui capte, stocke, traite et transmet des données. Il les transmet ensuite via Internet par un protocole de communication pour informer l'utilisateur de son état.

L'alarme moto connectée sera donc une alarme (ou plutôt un traqueur) qui nous permettra de savoir en temps réel où est située la moto. En cas de déplacement non voulu de cette dernière, le propriétaire sera informé via SMS de la localisation exacte de la moto toutes les 5 minutes jusqu'à ce qu'elle soit retrouvée.

## Bête à corne

Afin de mieux comprendre le but de notre projet, nous mettons en place une bête à corne :



Ce schéma est très simpliste, mais c'est la base même de la compréhension du projet.

Pour mieux comprendre ce qui est attendu, nous allons faire une synthèse du cahier des charges.

## Synthèse du cahier des charges

Il est attendu de créer une alarme moto connectée qui a pour fonctionnalité d'envoyer par SMS, les coordonnées GPS de la moto, lors du déplacement de cette dernière.

- Le mouvement de la moto et donc de l'alarme sera détecté par un accéléromètre.
- Les coordonnées GPS seront détectées par le GPS.
- Le SMS pourra être transmis grâce au module GSM.

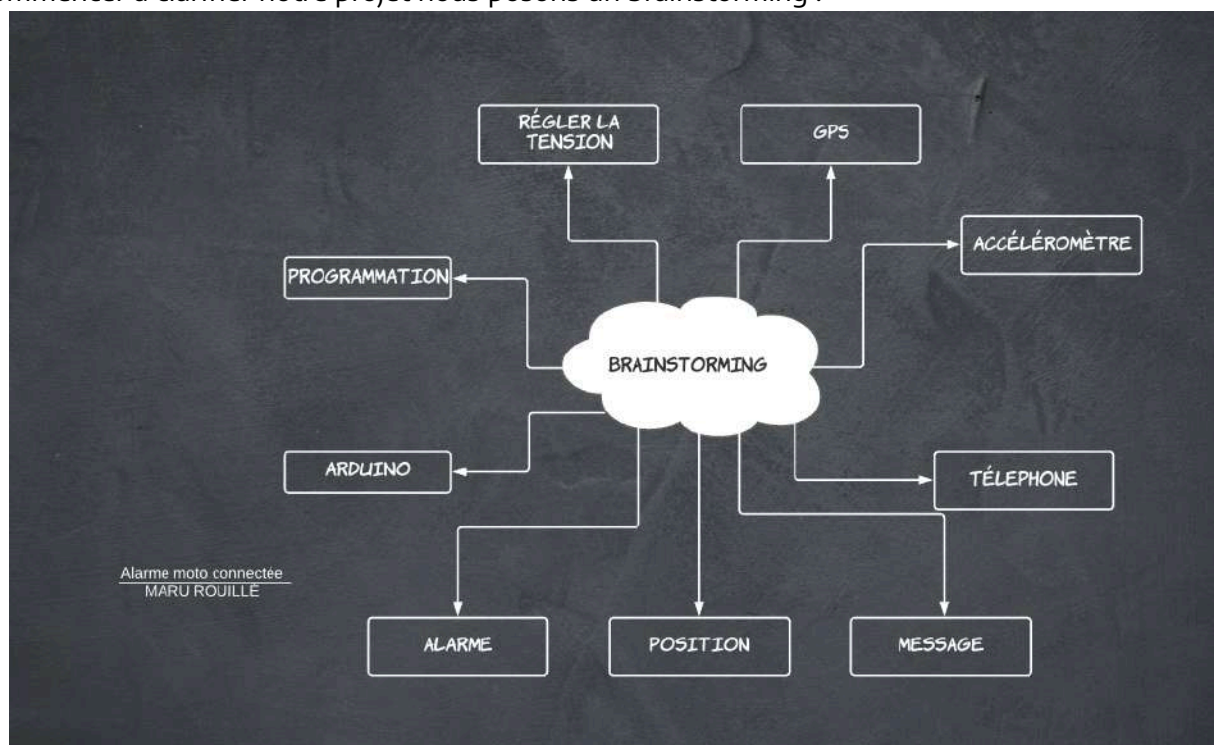
Tout ce processus sera contrôlé par le module Seeeduino.

Le montage doit être alimenté en 5 V, cependant la batterie d'une moto fournie 12 V ainsi il faudra trouver un système qui permette de réguler cette tension.

Maintenant que nous avons mis en place une synthèse du cahier des charges, on met en place un brainstorming.

## Le brainstorming

Pour commencer à clarifier notre projet nous posons un brainstorming :

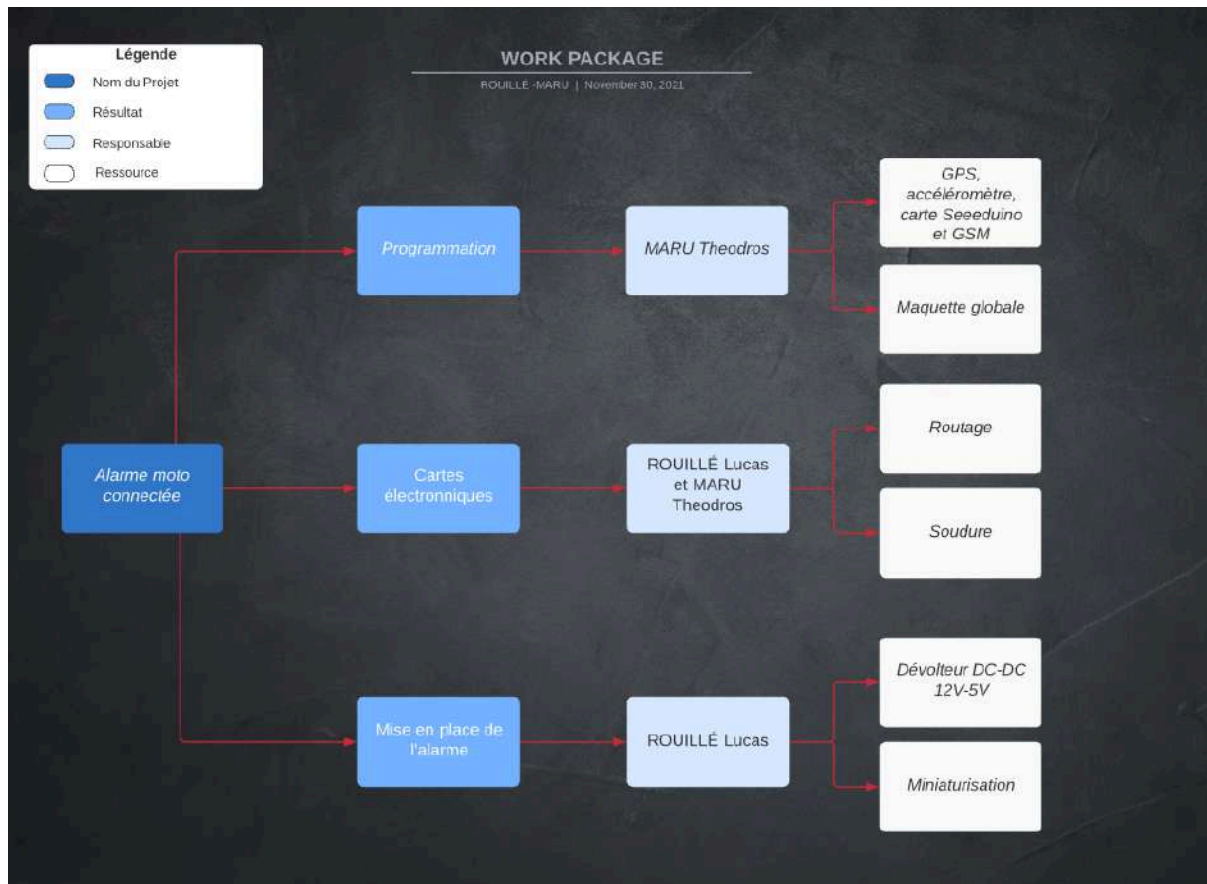


Le but est de comprendre ce qui nous est demandé et ceci commence par des travaux tels que des brainstormings afin d'ordonner nos idées.

Pour pouvoir créer un vrai emploi du temps précis, il faudra avoir recours à un work package dans le but de séparer les différents points liés au projet.

## Work Package

Grâce à notre brainstorming et à notre bête à cornes, on peut modéliser un "work package" regroupant les différentes parties du projet :



On peut remarquer que les 3 règles du "Work Package" ont été respectées :

- Résultat (nom de la fonction).
- Responsable (nom de la personne qui s'occupe de la fonction).
- Ressources (nom des étapes qui conduisent au résultat)

Afin de pouvoir créer un diagramme de Gantt, c'est-à-dire un emploi du temps concret et précis sur le projet, il manque plus qu'à citer les différents risques pour pouvoir par la suite les prendre en compte dans le diagramme de Gantt.

## Les Risques

Dans un projet, il y aura toujours des risques, mais le but va être de leur donner un degré de probabilité et un degré de gravité.

On commence par une liste des risques :

- Dysfonctionnement du matériel fourni. (1)
- Financier (mauvaise estimation du budget). (2)
- Mauvaise soudure. (3)
- Mauvaise polarisation d'un composant. (4)
- Surtension lors d'une mise en tension. (5)
- Erreur lors du routage. (6)
- Conflit au sein du groupe (mise en place de réunion fréquente). (7)
- Dégradation de la carte. (8)

Maintenant, que ceci a été fait, on va les ranger dans un tableau et les trier :

Projet - Alarme moto connectée		Gravité		
		Faible	Modérée	Élevé
Probabilité du risque	Faible		1	7 et 2
	Modérée	3	8 et 4 et 5	
	Élevé		6	

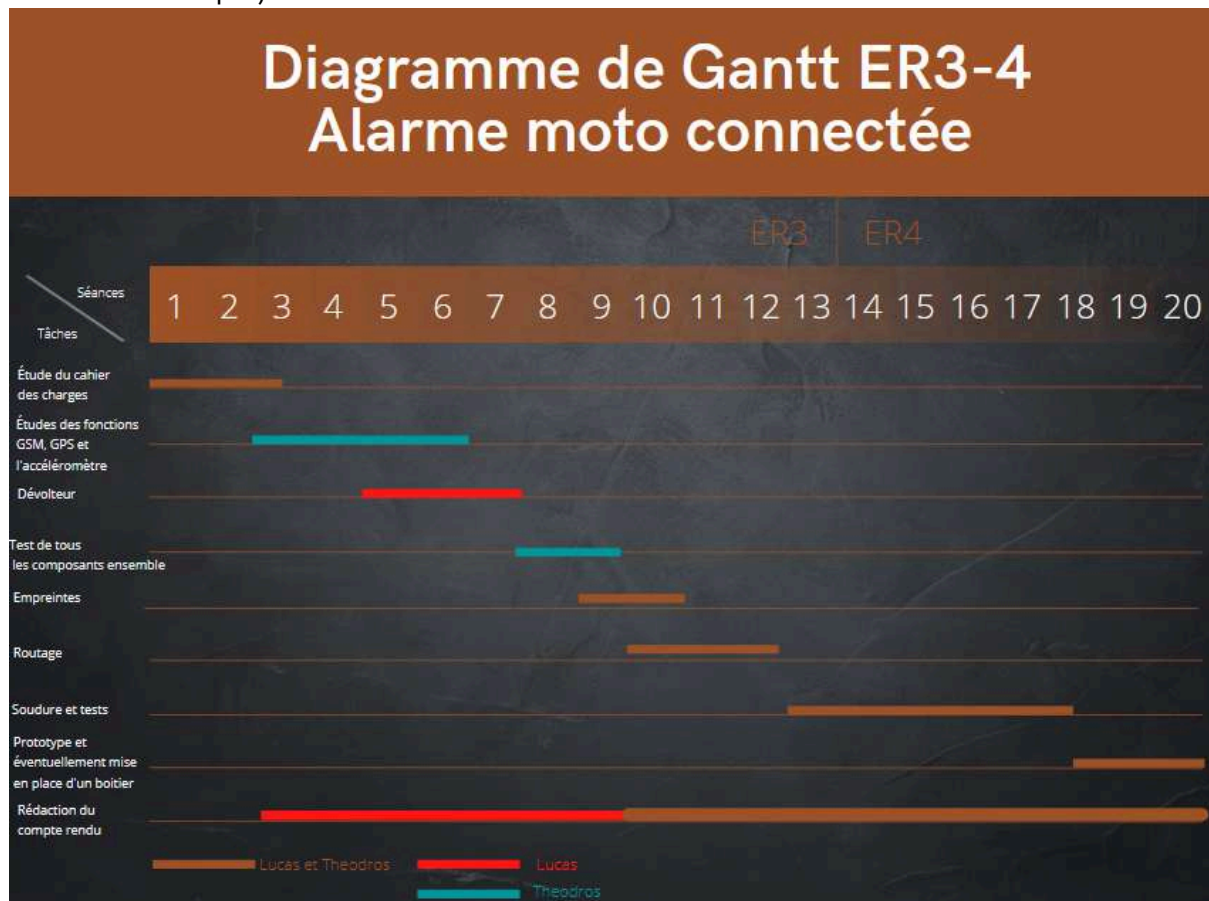
On remarque que le nombre de risques est relativement faible et que leurs gravités sont minimales. Le but est de prendre en compte les risques et de pouvoir les incrémenter dans l'emploi du temps, on appelle ça une marge, cependant, il ne faut pas oublier les aléas (des risques pas prévus, mais qui ont été pris en compte dans l'emploi du temps).

Pour ce qui est de la classification, certains risques sont rangés dans "Élevé" alors que dans un projet traditionnel, il serait d'une gravité modérée, voire faible, car la durée du projet est très courte (85 h).



## Emploi du temps prévisionnel

Après avoir effectué toutes les études précédentes, on a pu déceler le besoin du client et nous avons donc décidé de ramener ces informations afin de faire un diagramme de Gantt pour structurer notre projet au mieux.



Tout d'abord,

Nous commençons par une étude de cahier des charges qui prend environ 2,5 séances de la séance 1 à 2,5.

On a l'étude des différentes fonctions qui prennent en théorie 4 séances de la séance 3 à 6.

Le compte-rendu doit commencer à partir de la séance 5.

La partie dévolteur sera effectuée en 3 séances de la séance 5 à 7.

Par la suite, on testera toutes les fonctions en même temps, ceci devrait prendre 2 séances 7 à 8.

Ensuite,

On pourra donc ainsi commencer la partie sur Kicad, tout d'abord les empreintes pendant environ 2 séances à partir de la séance 8 à 9.

Pour le routage, on prendra 3 séances qui débiteront de la séance 9 à 12.

L'impression devrait être prévue pour la 13e séance (dernière séance de l'ER3).

Les soudures et les tests seront effectués sur une longue période, car il y a de fort risque que l'on rencontre des problèmes comme vu dans la partie sur les risques :

Également, il y a une longue période afin de rattraper un potentiel retard créé par des aléas. Les soudures et tests seront effectués en 5 séances de la séance 13 à 17.



Enfin, si le temps nous le permet, on fera un prototypage de l'alarme de moto connectée avec un boîtier en plastique.

## Calcul de budget théorique

Lors de la réalisation d'un projet il y a un facteur qui sera très important en effet, c'est le budget qui limitera le projet en temps et en matériel.

Dans cette partie, on va essayer d'estimer le coût de fabrication de chaque carte, le but de ces 2 parties budget en début et fin de projet, c'est de voir les estimations du budget par rapport à un vrai projet.

On ne tiendra pas compte du fait qu'il y est différentes valeurs selon la résistance ni pour les condensateurs.

Tout d'abord, on fait un listing des composants nécessaires à la conception (prix unitaire sur RS Components FR) :

- Résistances CMS x8 :

**0,007 €**  
HT

**0,008 €**  
TTC

- condensateurs 100uF :

Prix pour L'unité (en sachet de 200)

**0,24 €**  
HT

**0,29 €**  
TTC

- condensateurs 1000uF

Prix pour L'unité (en sachet de 200)

**0,876 €**  
HT

**1,051 €**  
TTC

- Inductance x1:

Prix pour l'unité (par multiple de 2)

**1,44 €**  
HT

**1,73 €**  
TTC

- LM2575-ADJ x1 :

**3,113 €**  
HT

**3,736 €**  
TTC

- LM7805 x1 :

**0,551 €**  
HT

**0,661 €**  
TTC

Même si les modules sont fournis on les comptera pour ce calcul.

- GSM x1:

**9<sup>99</sup> €**

- GPS :

Prix pour l'unité (en sachet de 10)

**35,815 €**  
HT

**42,978 €**  
TTC

- Seeeduino:

Prix pour la pièce

**5,26 €**  
HT

**6,31 €**  
TTC

- Accéléromètre:

Prix pour la pièce

**7,57 €**

HT

**9,08 €**

TTC

Le coût total des matériaux sera donc de 75.89 €. On prendra une marge de 40% pour le budget théorique du matériel soit  $75.89 + 75.89 \times 0.4 = 106.246$  €

Désormais, on passe au calcul du personnel.

On a 20 séances de 4 h soit 80 h de projet on engage 2 techniciens pour réaliser ce projet, payé 14.29 € brut/heure. Le personnel coûtera donc,  $80 \times 2 \times 14.29 = 2285.7$  €

Le budget total est donc de  $106.246 + 2285.7 = 2391.96$  € que l'on arrondit à 2400€.

On tient à rappeler que les coûts liés à la location des locaux, du matériel, l'électricité [...] ne sont pas comptés également tout ce qui pourrait donner suite à une commercialisation du projet (transport, pub [...]).

## **Test des fonctions**

Dans cette partie, nous allons tester tous les modules un par un, afin de comprendre comment ces derniers fonctionnent pour les utiliser pour notre projet. Après que tous les tests ainsi que le PCB soient réalisés, nous adapterons notre code plus précisément afin que l'alarme soit 100% fonctionnel et utilisable par une personne voulant avoir une protection pour sa moto.

Tous les modules seront reliés au dévolteur dans le but d'avoir une source stable de tension 5V et seront contrôlés par le Seeeduino.

## **Accéléromètre**

L'accéléromètre à 3 axes que nous allons utiliser est le MMA8451 de chez Adafruit. Nous avons choisi d'utiliser un accéléromètre pour détecter le mouvement de la moto, car ce dernier est très sensible aux variations de forces exercées sur ses axes X,Y et Z. Ce qui nous permettra de savoir facilement si la moto est en train d'être bougée.

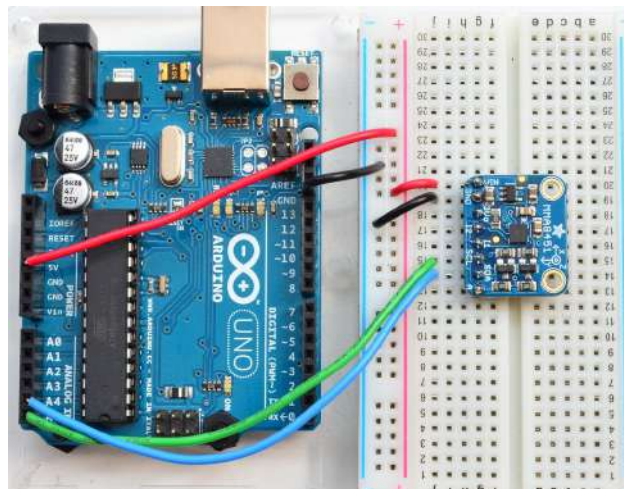
Lorsque le propriétaire a garé sa moto, cette dernière n'est pas censée avoir de force extérieure à celle de la gravité s'exercer sur elle. Ainsi, quand la moto est garée, tous les paramètres de l'accéléromètre sont censés rester les mêmes. Or, si les paramètres de l'accéléromètre changent (X et Y si la moto est relevée et ensuite Z si elle est déplacée par exemple), alors cela veut dire que la moto est mise en mouvement.

Ceci nous permettra d'envoyer un premier message au propriétaire pour l'informer que la moto est mise en mouvement. Si le propriétaire ne désactive pas l'alarme, la carte Seeeduino réveillera le module GPS afin de connaître la localisation de la moto et de savoir où on la déplace.

Comme on peut le voir ci-dessous, selon l'inclinaison donnée à l'accéléromètre, les valeurs présentes en X, Y ou Z changent. Par exemple, si on met la carte à plat, nous avons comme données  $X = Z = 0$  alors que  $Y = 9,84$  (force gravitationnelle) alors que si on met la carte à 180 degrés, on a  $Y = Z = 0$  alors que  $X = 9,84$ .

On pourra donc demander au Seeeduino de vérifier ces valeurs constamment afin de savoir si elles changent. Et si jamais elles changent de notifier le propriétaire.

Pour l'utiliser, il suffit de brancher l'accéléromètre comme suivant :



Source: Adafruit

Le code que nous devons utiliser pour l'accéléromètre nous est fourni par Adafruit. Nous l'avons légèrement modifié afin qu'il réponde à notre cahier des charges (voir [Annexe](#)).

Ce code nous permet de connaître les forces exercées sur l'accéléromètre en  $m/s^2$  et nous affiche pour une certaine orientation:

```
X:      0.05  Y:      -0.65  Z:      0.71  m/s^2
```

Source: Adafruit

Si on modifie l'orientation de l'accéléromètre un changement a bien lieu:

```
X:      -0.75  Y:      0.07  Z:      0.64  m/s^2
```

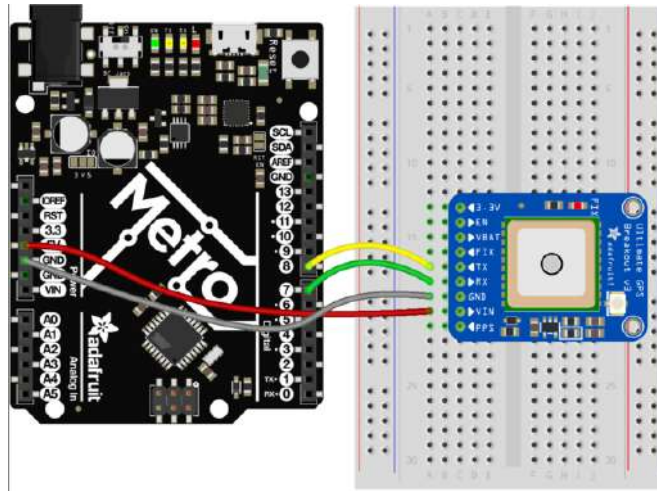
Source: Adafruit

Et c'est justement ce type de changement de force non voulu par le propriétaire sur les axes et qui mettra en "alerte" le Seeeduino et l'amènera à allumer la carte GPS afin de vérifier la localisation.

## GPS

Afin de connaître la localisation de la moto nous allons utiliser l'Adafruit Ultimate GPS. Le code utilisé est celui distribué par Adafruit et n'a été modifié qu'afin de pouvoir avoir les données en français (voir [Annexe](#)).

Le montage à faire afin d'utiliser le GPS est le suivant:



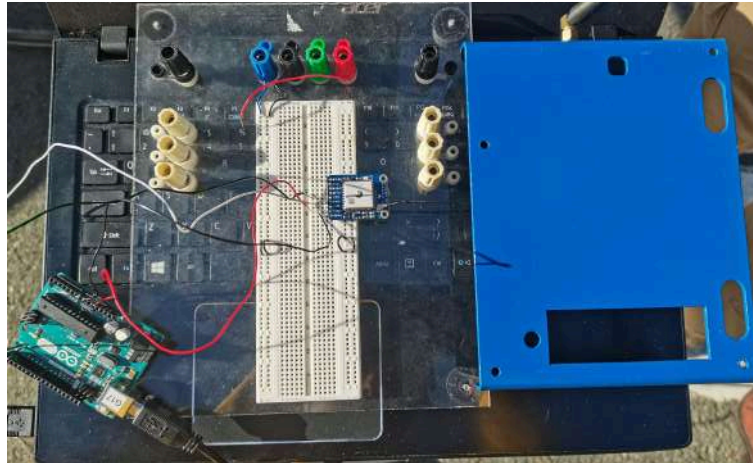
Source: Adafruit

Lorsque nous allumons le GPS à l'intérieur de l'IUT nous pouvons observer (voir ci-dessous) sur la console qu'il n'y a pas les données de localisation (remplacé par des virgules). Ceci est dû à la cage de Faraday créée par le bâtiment qui bloque toutes les ondes. La communication entre le module et le satellite est donc impossible dans le bâtiment.

```
COM19
15:21:57.380 -> Software Serial GPS Test Echo Test
15:21:57.475 -> Get version!
15:21:57.522 -> $GPGGA,202156.084,,,,,0,0,,,M,,,*46
15:21:57.522 -> $GPGLL,,,,,202156.084,V,$PMTK705,AXN_2.51_3339_17112000,0004,1616s,1.0*56
15:21:57.755 -> $GPGGA,202158.084,,,,,0,0,,,M,,,*48
15:21:57.802 -> $GPGLL,,,,,202158.084,V,N*7A
15:21:57.849 -> $GPGSA,A,1,,,,,,,,,,,,,*1E
15:21:57.849 -> $GPGSV,1,1,00*79
15:21:57.896 -> $GPRMC,202158.084,V,,,,,0.00,0.00,070221,,,N*4B
15:21:57.944 -> $GPVTG,0.00,T,,M,0.00,N,0.00,K,N*32
15:21:58.739 -> $GPGGA,202159.084,,,,,0,0,,,M,,,*49
15:21:58.785 -> $GPGLL,,,,,202159.084,V,N*7B
15:21:58.832 -> $GPGSA,A,1,,,,,,,,,,,,,*1E
15:21:58.832 -> $GPGSV,1,1,00*79
```

Source: Adafruit

Pour résoudre ce problème, nous sommes sortis dehors et avons pu avoir les coordonnées GPS d'où nous nous situons.



Nous n'avons plus les captures d'écrans, mais après avoir rentré les coordonnées sur google maps, nous avons observé une erreur de distance d'environ 3m, ce qui est raisonnable.

Nous avons aussi remarqué que quand nous enlevons et remettons l'alimentation du GPS, ce dernier prend approximativement 1 à 1min30 pour retrouver le signal. Ceci est important à prendre en compte, car si le propriétaire ne reçoit plus de coordonnées GPS pendant plus de 2min, cela veut dire que le GPS a été débranché et ne peut donc plus envoyer la localisation de la moto.

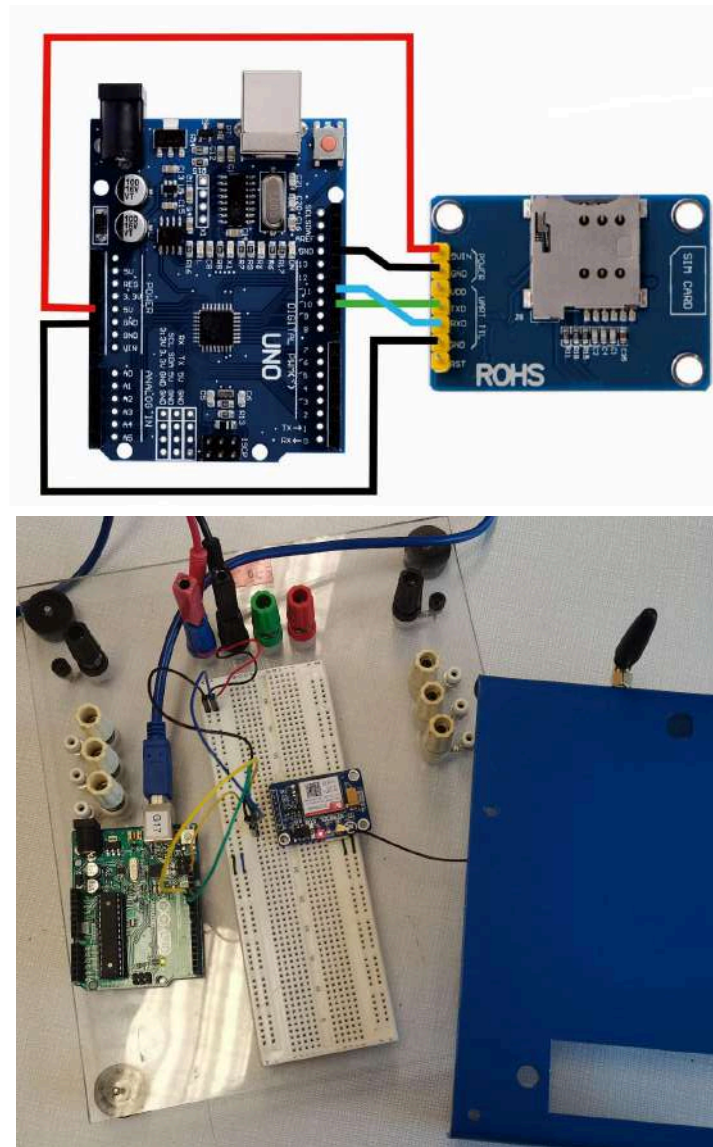
Ainsi, si le temps nous le permet, nous pourrions voir pour insérer une batterie lithium afin qu'elle puisse prendre le relais à la place de la batterie si jamais l'alarme est débranchée de l'alimentation. Ceci permettrait de continuer à envoyer les coordonnées GPS au propriétaire grâce à la carte GSM.



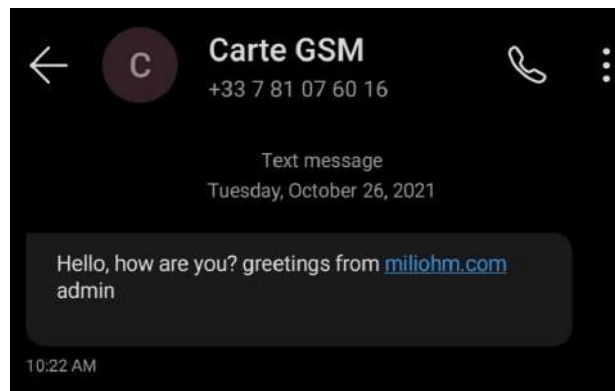
## GSM

Pour pouvoir envoyer par SMS les coordonnées GPS, nous avons besoin d'une carte GSM. Cette dernière est la carte SIM800L et permet de transmettre non seulement les coordonnées GPS, mais aussi si le temps nous le permet de contrôler la carte Seeeduino à distance afin d'éteindre l'alarme si nécessaire. Le code utilisé est celui d'une personne ayant utilisé le SIM800L (voir [Annexe](#)) pour envoyer et de recevoir de simples messages. Nous avons enlevé les parties qui ne nous serviront pas.

Le montage à faire dans le but d'utiliser le GSM est le suivant :



Nous recevons bien un message (qui est celui du code originel) :



Lors de l'étude de la partie théorique, en étudiant scrupuleusement la datasheet on remarque qu'il est conseillé de mettre des résistances de  $1k\Omega$  entre les pins RXD et TXD du module GSM et le seeeduino.

Figure 19: Connection of the serial interfaces

If the voltage of UART is 3.3V, the following reference circuits are recommended. If the voltage is 3.0V, please change the resistors in the following figure from 5.6K to 14K.

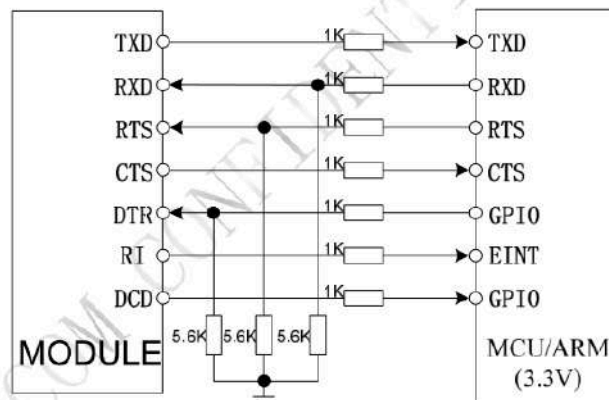


Figure 20: Resistor matching circuit

If the voltage of UART is 3V or 3.3V, the following reference circuits are recommended:

## Étude théorique du module GSM :

Table 53: Operating frequencies

Frequency	Receive	Transmit
GSM850	869 ~ 894MHz	824 ~ 849MHz
EGSM900	925 ~ 960MHz	880 ~ 915MHz
DCS1800	1805 ~ 1880MHz	1710 ~ 1785MHz
PCS1900	1930 ~ 1990MHz	1850 ~ 1910MHz

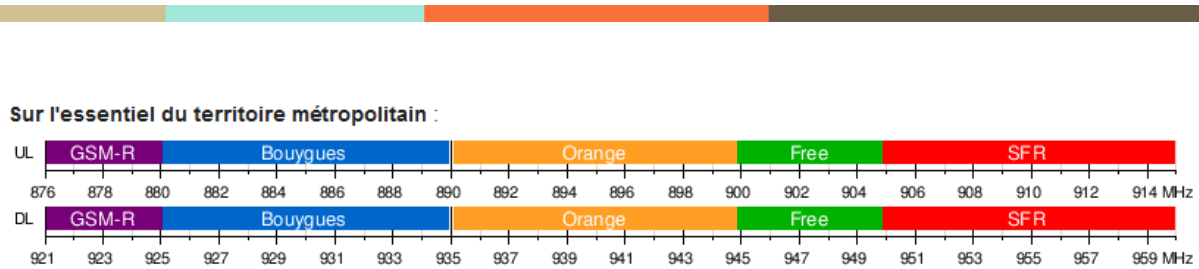
Lors de notre semestre 3, nous avons notamment étudié la PLL ( Phase Locked Loop traduit en français par boucle à verrouillage de phase). Ce système est utilisé par le sim800l pour communiquer avec différents appareils tel qu'un téléphone portable. Dans la datasheet, nous observons plusieurs systèmes de communications avec des fréquences différentes, après quelques recherches en Europe les systèmes de communications, communiquent entre eux avec des GSM 900 et 1800. Les GSM900 sont aussi appelés EGSM900 et les GSM1800 sont aussi appelés DCS1800. Le GSM 850 et le PCS1900 quant à eux sont utilisés au Canada et au États-Unis vous pouvez retrouver toutes ces informations sur le lien suivant [Les différentes fréquences de communications](#) .

### Bandes de fréquences mobiles en France

Fréquence	Réseau
Plage 694-790 Mhz	4G LTE
Plage 791-862 Mhz	4G LTE
Plage 876-960 Mhz	2G GSM / 3G UMTS
Plage 1710-1880 Mhz	2G GSM / 4G LTE
Plage 1900-1979 Mhz	3G UMTS
Plage 2110-2169 Mhz	3G UMTS / 5G
Plage 2500-2570 Mhz	4G LTE
Plage 2620-2690 Mhz	4G LTE
Plage 3400-3800 Mhz	5G
Plage 24500-27500 Mhz	5G

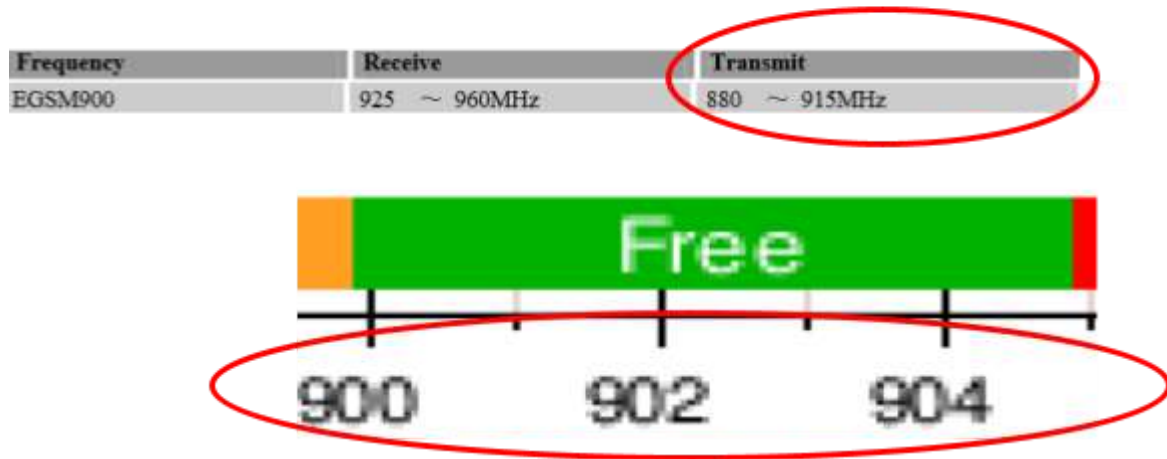
Nous avons un système qui communique seulement en 3G donc, ceci signifie qu'on communique entre 876 et 960 MHZ d'après le tableau des fréquences précédent, on peut en conclure que le système de communication utilisé est donc le EGSM900.

On utilise une carte sim Free mobile, nous avons trouvé les différentes bandes fréquences des différents opérateurs en France :



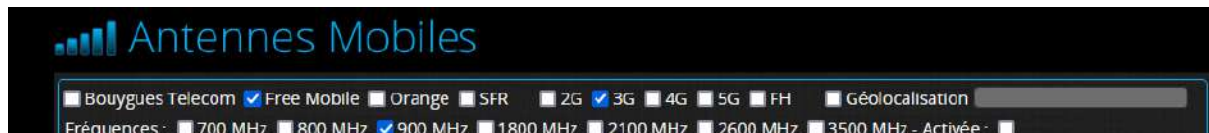
Ces données datent de 2011.

Nous sommes en mode UL (Upload-Download), nous envoyons une information.

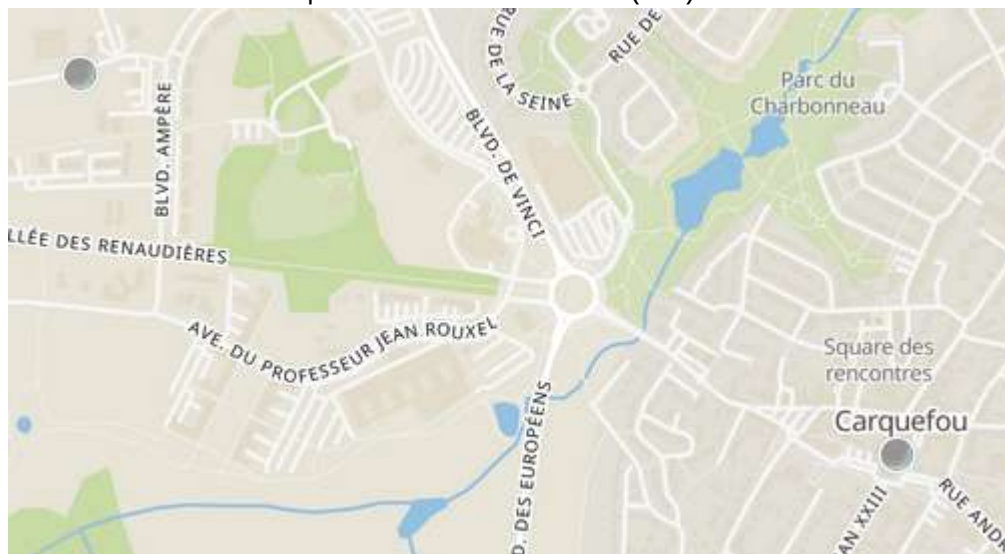


Donc notre bande de communication sera de 900 MHz à 905 MHz.

À l'aide d'un site internet qui répertorie toutes les antennes de France, nous avons filtré la recherche en demandant seulement les antennes Free Mobile 3G en 900 MHz.



On observe 2 antennes proches du lieu des tests (IUT) :



Vous pourrez retrouver ces informations grâce au site suivant : [Les antennes mobiles](#).

En milieu rural, une antenne relais peut couvrir une zone de 10 à 30 kilomètres alors qu'en milieu urbain très dense, cette couverture peut être de seulement 500 mètres.

La puissance d'un signal GSM de téléphonie mobile s'exprime en dBm, ou décibel-milliwatt, qui est une unité de mesure de puissance électrique, et qui permet de connaître la puissance des ondes du signal généré par une antenne de télécommunication. Cette mesure s'étale de -121 dBm (signal faible) à -51 dBm (bon signal). Retrouvez ces informations sur [guides GSM](#).




À l'IUT, on observe un signal de -112 Dbm (mesure réalisée avec mon téléphone personnel [check-phone-network-signal-quality](#) ). Ce qui signifie que le réseau n'est pas de très bonne qualité.

### Calcul du poids de transmission du SMS :

**Table 2: Coding schemes and maximum net data rates over air interface**

Coding scheme	1 timeslot	2 timeslot	4 timeslot
CS-1	9.05kbps	18.1kbps	36.2kbps
CS-2	13.4kbps	26.8kbps	53.6kbps
CS-3	15.6kbps	31.2kbps	62.4kbps
CS-4	21.4kbps	42.8kbps	85.6kbps

Un caractère Standard étant encodé sur 7 bits, un SMS peut contenir au maximum 160 caractères standards ( $160 \times 7 = 1120$ ). Selon le modèle de votre téléphone, les caractères spéciaux peuvent être encodés sur 8 ou 16 bits.



Quelque ce soit le schéma de code et le nombre d'horloges utilisé et le message encodé, le GSM pourra transmettre le message, car la taille maximale d'un SMS est de  $16 * 160 = 3200$  bits. Le message qui sera envoyé à l'utilisateur est :

“Votre véhicule a été déplacé, il se trouve actuellement aux coordonnées suivantes : 41°24'12.2" N 2°10'26.5" E”.

Ce message contient : 83 caractères standards et 10 caractères spéciaux tout possibles à coder sous 8 bits, donc on a un message de  $83*7+10*8 = 661$  bits, le message pourra donc être bien transmis par le GSM.

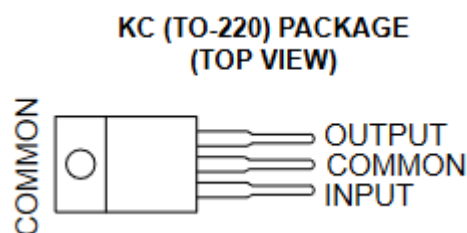


## Les dévolteurs :

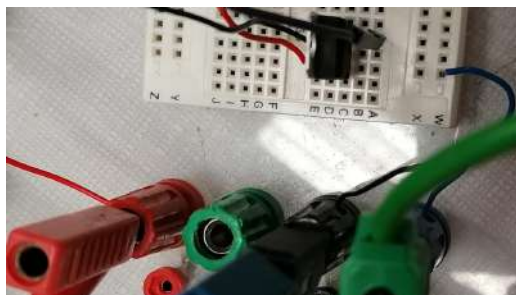
Dans le cahier des charges, il est stipulé qu'il faudra alimenter notre carte depuis la batterie, le problème étant que la batterie fournit du 12V et que le montage doit être alimenté 5V et 4V pour le GSM. Pour cela, nous utiliserons un dévolteur. Suite à notre soutenance ER3, nous avons eu un nouveau GSM, cependant, ce GSM devra être alimenté en 4V donc on a dû refaire notre théorie les résultats pratiques de la partie précédente seront conservés pour laisser une trace du travail effectué.

## Régulateur :

Afin d'alimenter les autres modules à 5 V on utilise un régulateur à 5V en sortie (LM7805)



La mise en place est très rapide :

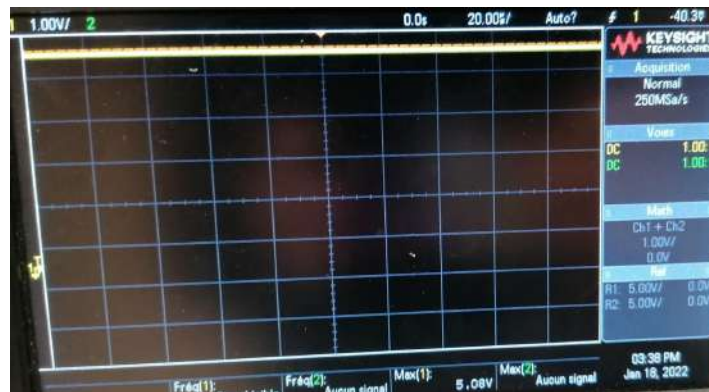


On obtient effectivement 5V :





On a bien une tension à 5 V



On fait un test de charge, afin de déterminer quelle sera la résistance à appliquer, on regarde les différentes consommations de courant des composants :

Consommation Accéléromètre :

- **Current Consumption: 6  $\mu$ A to 165  $\mu$ A**

Consommation GPS :

- **MTK3339 Operating current: 25mA tracking, 20 mA current draw during navigation**

Consommation Seeeduino :

DC Current per I/O Pin	40 mA	DC Current per I/O Pin	40 mA
------------------------	-------	------------------------	-------

On prend le cas le plus défavorable, soit 25 mA + 40 mA + 0.165 mA = 65.165 mA

On fera nos tests pour un courant de 130 mA pour être plus large au niveau de nos mesures.

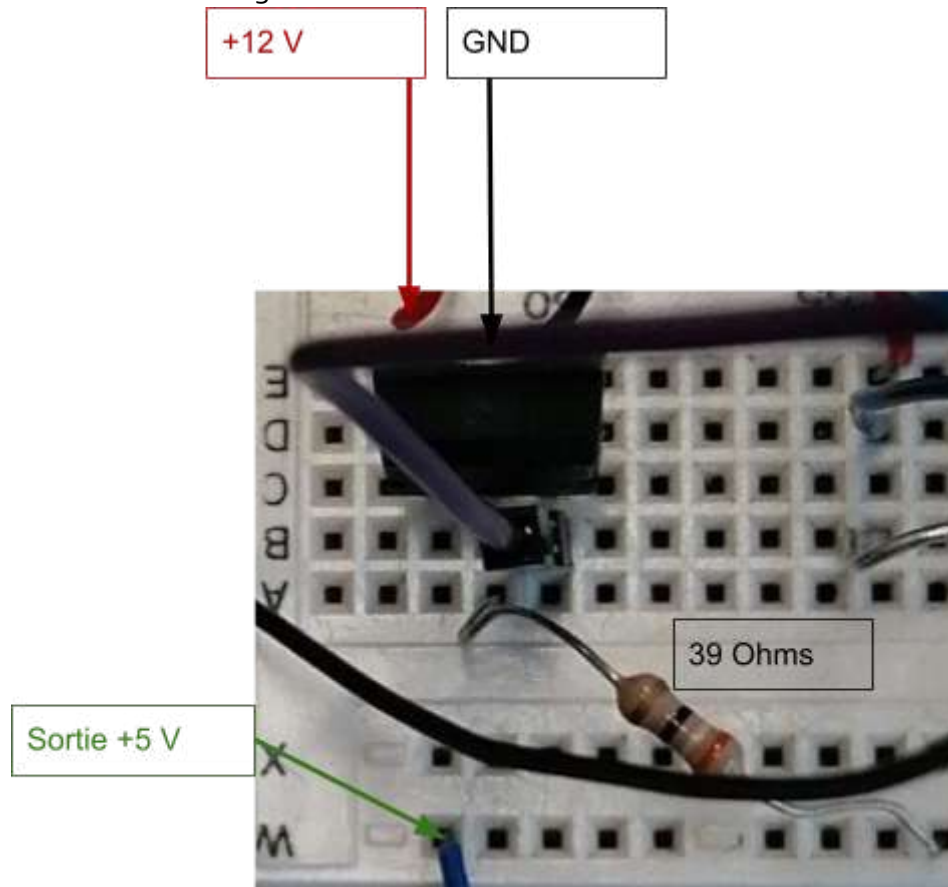
On sait que :

$$R = \frac{U}{I} = \frac{5}{0.130} = 38\Omega$$

On prendra une résistance de 39 $\Omega$ .



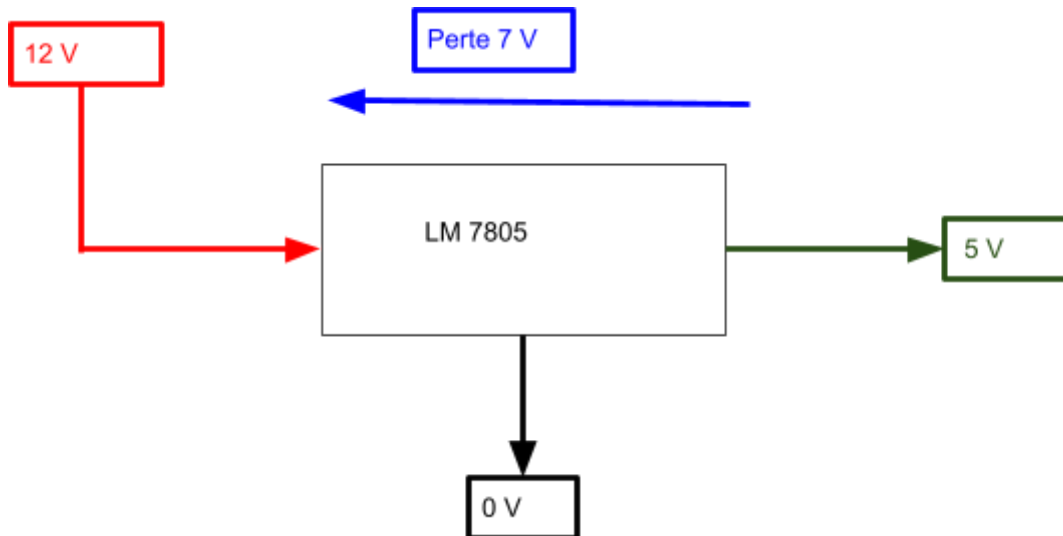
On réalise le montage ci-dessous :



### **Calcul de déperdition calorifique d'un régulateur de tension :**

Contrairement à un dévolteur à découpage, la perte en Volt entre l'entrée et la sortie est la différence des deux tensions.

Ici, ceci signifie qu'on a une perte de 7 V :



On met une résistance de 39  $\Omega$  soit 130 mA donc la puissance perdue sera de

$$P_{\text{Perdue}} = V_{\text{Perdue}} * I = 7 * 0.13 = 0.91 \text{ W}$$

Cette puissance perdue n'est pas très parlante, c'est pour cela que l'on va convertir cela en  $^{\circ}\text{C}$ .

Pour réaliser cette conversion, on se sert d'une formule de déperdition calorifique d'une paroi (convection naturelle) cette formule a été vue lors du module P1.

$$\Phi = h * S * \Delta T$$

$\Phi$  représente la puissance perdue en W

h représente un coefficient de convection en  $\text{W.m}^{-2}.\text{K}^{-1}$

S surface de la paroi en  $\text{m}^2$

$\Delta T$  représente la différence de température ( $T_2 - T_1$ ) étant un écart de température on l'exprimera en  $^{\circ}\text{C}$ .

Tout d'abord on a une puissance perdue :

$$\Phi = 0.91 \text{ W (déterminé précédemment).}$$

Le composant est composé d'incox et de plastique, on prendra ainsi le coefficient de convection de l'aluminium et plastique (donnée trouvée sur Internet).

Polymère, haute densité	0,33 0,52	296	$10^{-16}$ $10^2$	
Inox	16,3	296	1,389 $1,429 \times 10^6$	AISI 302(Fe, Cr 18 %, Ni 8 %)

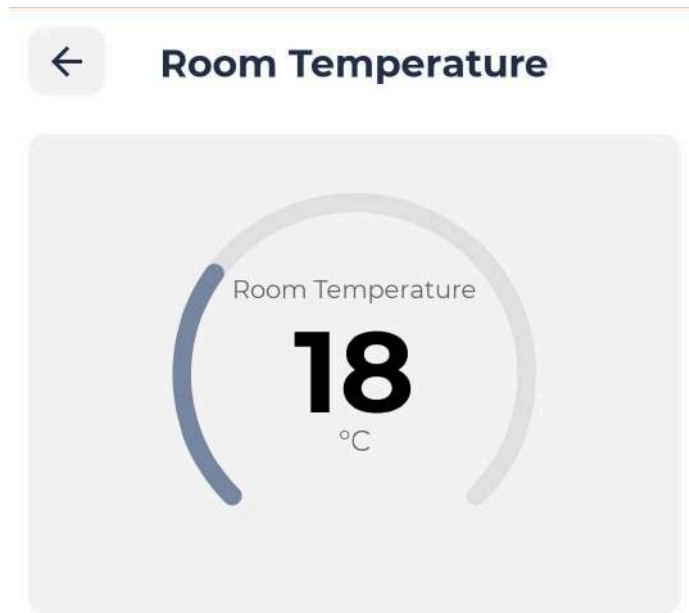
$\lambda_i = 16.3 \text{ W.m}^{-1}.\text{K}^{-1}$  coefficient de convection de l'incox.

$$h_i = \frac{\lambda_i}{e_i} = \frac{16.3}{12.88 \times 10^{-3}} = 1265.53 \text{ W.m}^{-2}.\text{K}^{-1}$$

$\lambda_p = 0.52 \text{ W.m}^{-1}.\text{K}^{-1}$  coefficient de convection du plastique.

$$h_p = \frac{0.53}{9.02 \times 10^{-3}} = 58.76 \text{ W.m}^{-2}.\text{K}^{-1}$$





$T_1 = 18^{\circ}\text{C}$

Donc  $T_2 = T_1 + \Delta T$   $T_2 = 18 + 22.47 = 40.47^{\circ}\text{C}$

Ceci est cohérent avec la chaleur ressentie lors du toucher du composant.

$T_J$   

---

 $0^{\circ}\text{C to } 125^{\circ}\text{C}$

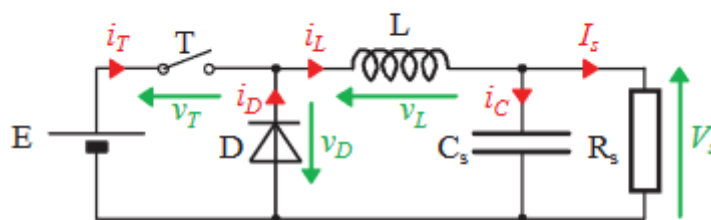
On a une limite thermique du composant à  $125^{\circ}\text{C}$  (donnée trouvée sur la datasheet), on peut en conclure que la mise en place d'un radiateur est inutile pour ce montage.

## Dévolteur à découpage :

### Rappel théorique :

Tout d'abord un montage dévolteur également appelé convertisseur Buck est représenté comme ceci :

Schéma classique représentant une structure Buck :



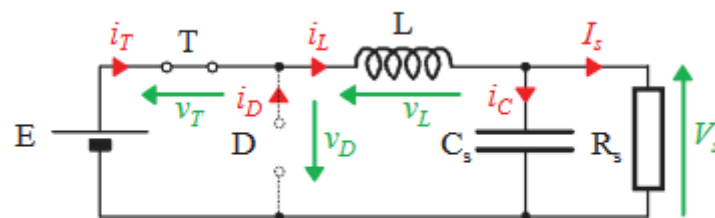
T : Interrupteur (IGBT, MOSFET...)

D : Diode

Il fonctionne en deux temps, lorsque le transistor est passant la diode est bloquée et inversement quand la diode est passante le transistor est bloqué. Cette commutation sera effectuée pendant une période. Cette transition sera dépendante d'un élément appelé rapport cyclique représenté par la lettre alpha ( $\alpha$ ). La commutation sera effectuée au temps  $\alpha T$ .

Pour étudier le système, il faut donc séparer ces deux temps, tout d'abord, on étudie le système si T est passant et D est bloquée.

$t \in [0 ; \alpha.T_d]$ , on commande T à l'amorçage.



T est passant, D est donc bloquée afin de ne pas court-circuiter la source E (règle d'interconnexion des sources).

Avec l'application de la loi des mailles et la loi des nœuds, on peut donc faire des conclusions sur le système :

#### Les tensions :

$$\begin{aligned} V_T &= 0 \text{ V;} \\ V_D &= -E; \\ V_s &= -V_L - V_D, \quad V_s = -V_L + E; \\ V_L &= E - V_s. \end{aligned}$$

#### Les courants :

$$\begin{aligned} i_D &= 0 \text{ A;} \\ i_T &= i_L; \end{aligned}$$

Pour déterminer la forme d' $i_L$ .

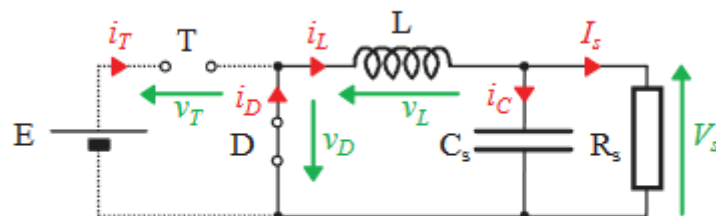
Dans une inductance,

$$V_L = L \frac{di_L}{dt}, \quad i_L = \int \frac{V_L}{L} dt \text{ on sort les constantes, soit } i_L = \frac{E - V_s}{L} * \int 1 * dt$$

$$i_L = \frac{E - V_s}{L} * t + K_1 \quad K_1 \text{ sera égale à } i_{\min}.$$

On étudie maintenant le système pour T bloqué et I passant.

$t \in [\alpha.T_d; T_d]$ , on commande T au blocage.



T est bloqué, D est donc passante afin d'assurer la circulation du courant  $i_L$  (règle d'interconnexion des sources).

Avec l'application de la loi des mailles et la loi des nœuds, on peut donc faire des conclusions sur le système :



### Les tensions :

$$\begin{aligned}V_T &= E; \\V_D &= 0V; \\V_s &= -V_L; \\V_L &= -V_s;\end{aligned}$$

### Les courants :

$$\begin{aligned}i_D &= i_L; \\i_T &= 0 \text{ A};\end{aligned}$$

Pour déterminer la forme d' $i_L$ .

Dans une inductance,

$$V_L = L \frac{di_L}{dt}, i_L = \int \frac{V_L}{L} dt \text{ on sort les constantes, soit } i_L = \frac{1}{T} * \frac{-V_s}{L} * \int 1 * dt$$

$$i_L = \frac{-V_s}{L} * t + K_2, K_2 \text{ sera égale à } i_{\max}.$$

On peut exprimer  $V_s$  en fonction de  $E$  et  $\alpha$ .

La valeur moyenne d'une inductance par définition est nulle soit :

$$\langle V_L \rangle = \frac{1}{T} \left( \int_0^{\alpha T} V_L(t) dt + \int_{\alpha T}^T V_L(t) dt \right) = \frac{1}{T} * ((E - V_s) * \alpha T + (-V_s) * (T - \alpha T))$$

$$\langle V_L \rangle = (E - V_s) * \alpha - V_s + V_s * \alpha$$

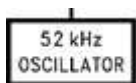
$$\langle V_L \rangle = E\alpha - V_s\alpha - V_s + V_s\alpha = E\alpha - V_s = 0 \text{ V}$$

$$\text{donc, } V_s = E\alpha$$

Nous avons une tension d'entrée de 12 V et une tension de sortie de 4 V

$$\text{Donc } \alpha = \frac{V_s}{E}, \alpha = \frac{4}{12} = 0.33.$$

$$f = 52 \text{ kHz soit } T = 19.2 \mu s$$



On mesure  $\Delta T$  de 0 à  $\alpha T$

$$E = L \frac{\Delta I}{\Delta T}, \Delta I = \frac{E * \Delta T}{L}, \Delta T = \alpha T - 0 \text{ soit } \Delta I = \frac{E * \alpha}{L * f} \Delta I = \frac{4}{100 * 10^{-6} * 52 * 10^3} = 0.77 \text{ A.}$$

● Class 4 (2W) at GSM 850 and EGSM 900

$$R_s = \frac{U^2}{P} = \frac{4^2}{2} = 8\Omega$$

$$I_s = \frac{V_s}{R_s} = \frac{4}{8} = 0.5 \text{ A}$$
 Sachant que  $\langle I_c \rangle = 0 \text{ A}$ ,

$$\langle I_L \rangle = \langle I_s \rangle, \text{ } I_s \text{ étant un signal continu, } \langle I_L \rangle = 0.5 \text{ A}$$

On peut décomposer  $\langle I_L \rangle$  par  $\langle I_L \rangle = I_{Lmin} + \frac{\Delta I}{2}$  et  $\langle I_L \rangle = \langle I_L \rangle = I_{Lmax} - \frac{\Delta I}{2}$ . Ainsi, on peut déterminer  $I_{Lmin}$  et  $I_{Lmax}$ .

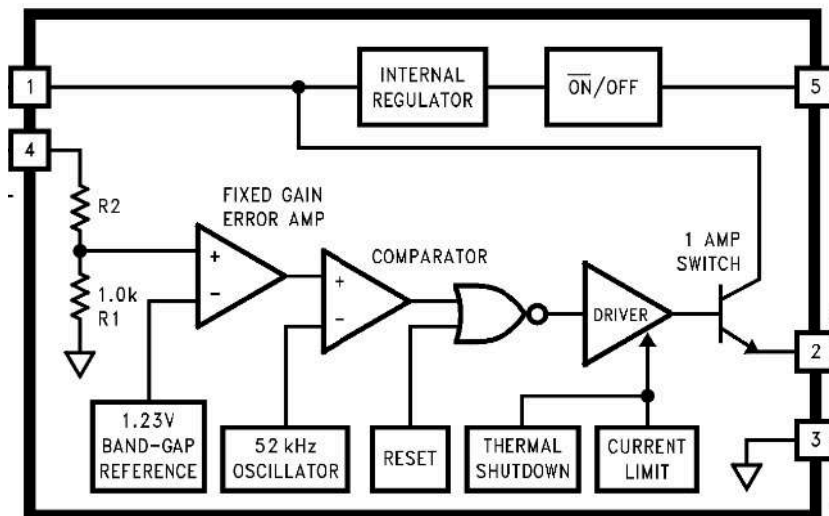
$$I_{Lmin} = \langle I_L \rangle - \frac{\Delta I}{2}, I_{Lmin} = 0.115 \text{ A et } I_{Lmax} = \langle I_L \rangle + \frac{\Delta I}{2} = 0.885 \text{ A.}$$

Toutes ces études montrent les limites de courants dans notre inductance.

## Etude théorique :

Un composant est directement fourni par l'IUT : LM2576T-ADJ P+.

En allant sur la datasheet on observe ce qu'il y a à l'intérieur.



On observe beaucoup de composants, les résistances ne sont pas comprises dans le composant, il faudra donc les rajouter.

On nous donne  $R_1 = 1k\Omega$ , on veut  $V_s = 4V$ ,  $V_{ref} = 1.23V$ .

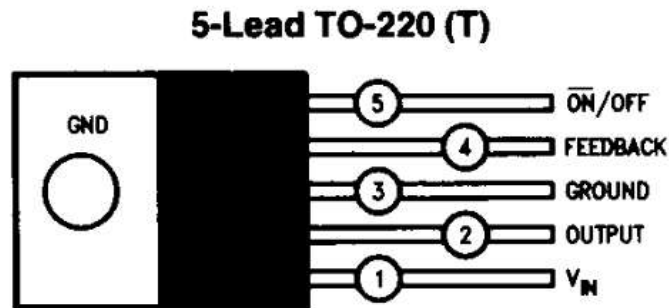
$$R_2 = \left( \frac{V_s}{V_{ref}} - 1 \right) * R_1 \text{ soit } R_2 = \left( \frac{4}{1.23} - 1 \right) * 1000 = 2252\Omega.$$

Malheureusement, en pratique, nous ne disposons pas de résistance de cette valeur, on prendra donc une résistance  $R_2 = 2200\Omega$ .

$$\text{Soit } V_s = 1.23 * \left( 1 + \frac{2.2}{1} \right) = 3.936 \text{ V (d'après la datasheet).}$$

## Etude pratique :

Désormais, nous allons passer à la pratique, voici comment sont disposés les différents pins du dévolteur :

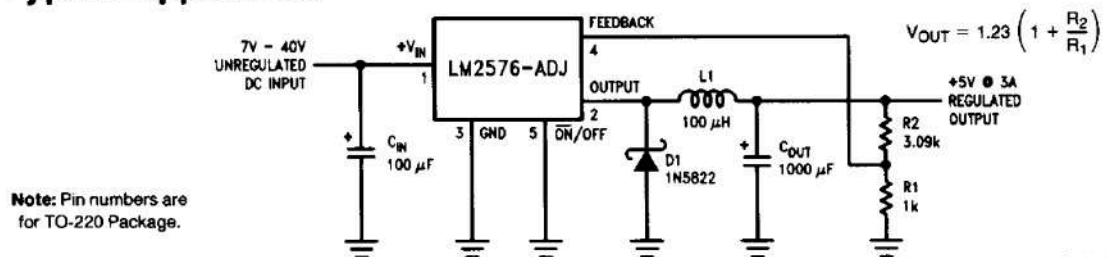


- La pin 1,  $V_{IN}$  reçoit la tension +12V de l'alimentation.
- La pin 2, OUTPUT nous donnera la tension de sortie (+4V).
- La pin 3, GROUND sera la masse du montage (+0V).
- La pin 4, FEEDBACK sert à transmettre la tension de référence  $V_{ref}$  (+1.23V).
- La pin 5, ON/OFF sert à faire une mise en veille.

## Manipulation pour obtenir une sortie 4 V :

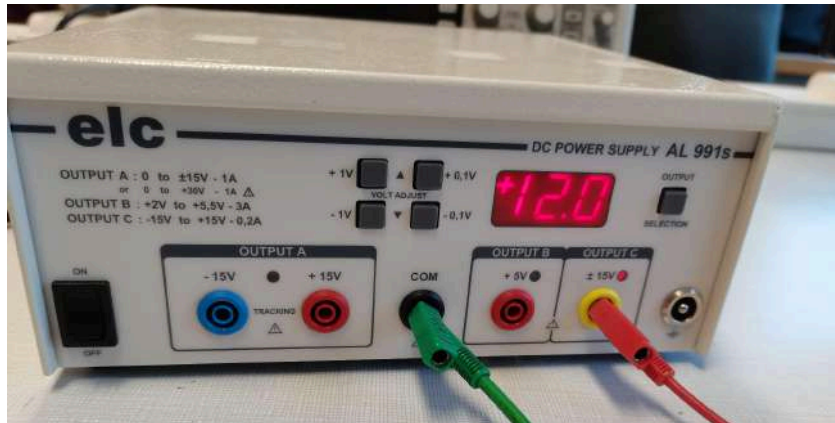
Nous mettons en place le montage à l'aide du montage ci-dessous :

### Typical Application

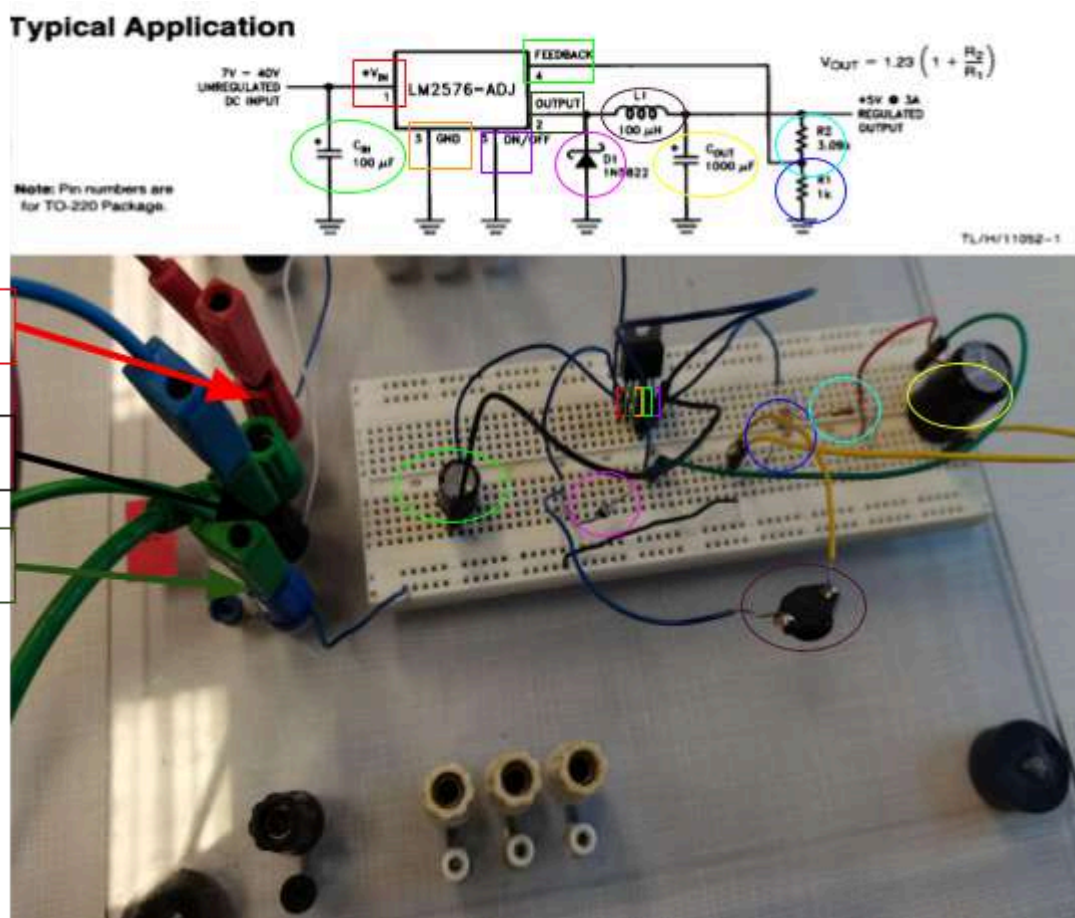


Ce schéma est tiré de la datasheet du composant LM2576-ADJ.

On alimente en 12V continu notre montage comme le ferait la batterie de la moto.



On met en place le montage :



Les carrés de couleurs font référence aux différents pins et les cercles font référence aux différents composants.

Le montage pour l'alimentation 4V est la même cependant il faudra modifier la résistance ça ne sera plus une  $R2 = 3.3k\Omega$  mais une résistance de  $R2 = 2.2k\Omega$ .

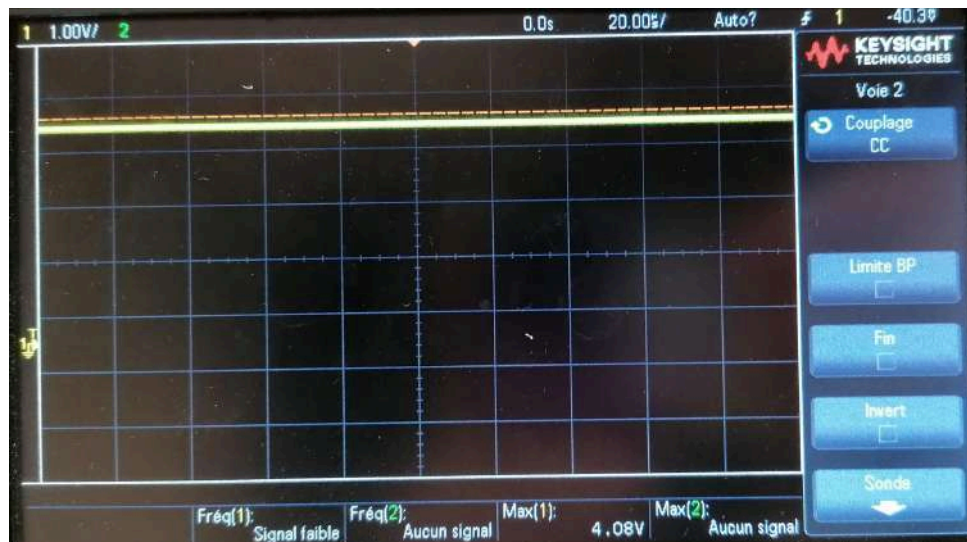


On obtient les résultats suivants :

- On a effectivement 4 V



On observe un signal bien continu à 4 V



On sait que le GSM délivre une puissance de 2W :

- **Class 4 (2W) at GSM 850 and EGSM 900**

Selon que le calcul d'une puissance est :

$$P = U \cdot I \quad i = \frac{P}{U} = \frac{2}{4} = 0.5A$$

Donc, on a un courant  $I$  de 0.5 A

Nous implémenterons un plan de charge, pour cela on fait un calcul simple,

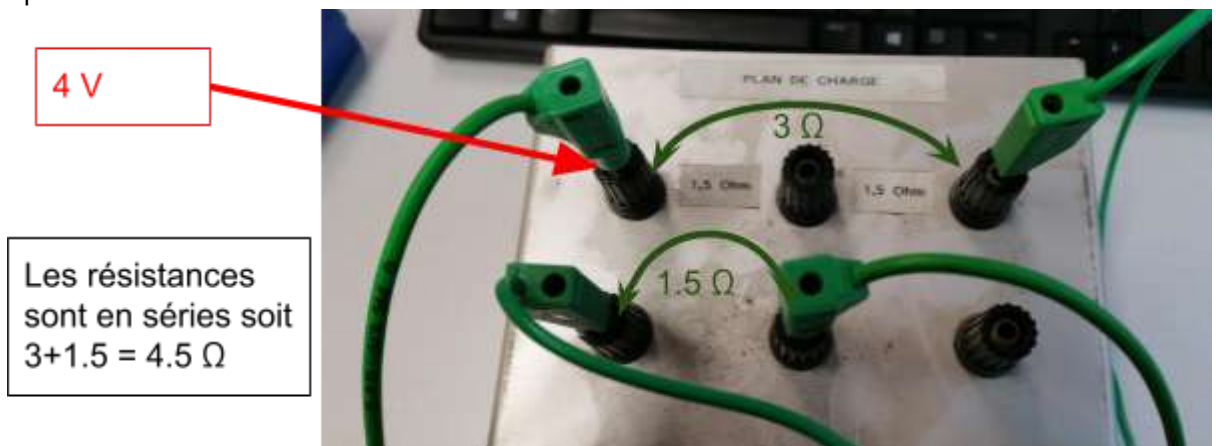
$$P = r \cdot i^2 \quad r = \frac{P}{i^2}, \quad r = \frac{2}{0.5^2} = 8 \Omega$$

Donc, pour correspondre au montage réel, on met une charge de  $4.5 \Omega$  pour avoir environ  $1 \text{ A}$  afin de s'assurer de la bonne dissipation thermique du composant :

En théorie on devrait avoir en sortie :

$$I = \frac{V}{R} = \frac{4}{4.5} = 0.89 \text{ A}$$

On applique donc nos  $4.5 \Omega$



On a  $4.8 \Omega$  ceci doit être dû à un défaut du plan de charge



Et donc en pratique on obtient  $0.843 \text{ A}$



Pour déterminer la mise en place d'un radiateur,

On étudie la datasheet afin de déterminer une puissance dissipée (p19),



Total power dissipated by the LM2575 can be estimated as follows:

$$P_D = (V_{IN}) (I_Q) + (V_O/V_{IN}) (I_{LOAD}) (V_{SAT})$$

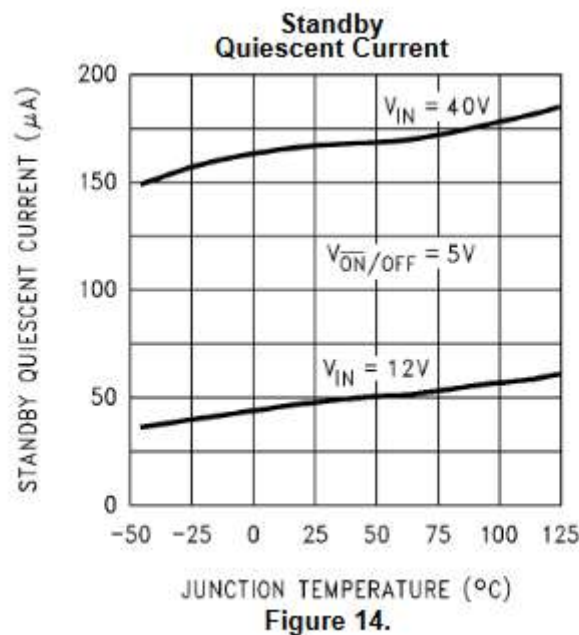
where

- $I_Q$  (quiescent current) and  $V_{SAT}$  can be found in the Characteristic Curves shown previously,
- $V_{IN}$  is the applied minimum input voltage,
- $V_O$  is the regulated output voltage
- and  $I_{LOAD}$  is the load current.

Avec,

- $V_{IN} = 12\text{ V}$
- $V_O = 4\text{ V}$
- $I_{LOAD} = 0.843\text{ A}$

Pour trouver  $I_Q$  et  $V_{SAT}$  il faudra faire une étude de courbe de la datasheet (p9) :



On observe ici le courant perdu,  $V_{IN} = 12\text{V}$ , la température ambiante est à  $25^{\circ}\text{C}$  on estime que le courant  $I_Q = 50\mu\text{A}$ .



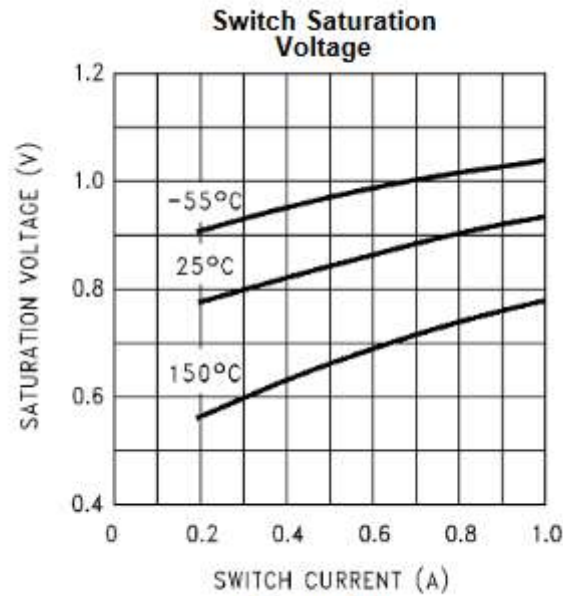


Figure 16.

On observe ici le courant perdu,  $I_{LOAD} = 0.843 \text{ A}$  la température ambiante est à  $25^\circ\text{C}$  on estime que la tension  $V_{SAT} = 0.9 \text{ V}$ .

Ainsi,

$$P_D = V_{IN} * I_Q + \frac{V_O}{V_{IN}} * I_{LOAD} * V_{SAT}$$

$$P_D = 12 * 50 * 10^{-6} + \frac{4}{12} * 0.843 * 0.9 = 0.25 \text{ W}$$

Il est possible de déterminer  $P_D$  d'une autre manière,

L'intégrale se fera au point  $\alpha T_d$  moment de commutation,

$$P_D = \frac{1}{T_d} \int_{\alpha T_d} V_{sat} * I_{Load} dt$$

$$P_D = \frac{V_{sat} * I_{Load}}{T_d} [\alpha T_d] = V_{SAT} * I_{LOAD} * \alpha$$

$V_S = E\alpha$  (formule d'un dévolteur classique).

$$\alpha = \frac{4}{12} = 0.33$$

$$P_{D\text{-conduction}} = 0.9 * 0.843 * 0.33 = 0.25 \text{ W}$$

Ceci correspond bien à la valeur trouvée grâce à la formule donnée par le constructeur.

On détermine le rendement,

$$P_s = V_{out} \cdot I_{LOAD} = 4 \cdot 0.843 = 3.372 \text{ W}$$

Le rendement de notre système est de  $\eta = \frac{P_s}{P_s + P_d} = 93\%$  Nous pouvons en conclure que nous avons très peu de perte, ce qui explique une puissance dissipée si faible et donc une légère chauffe du composant.

On reprend la même formule que celle utilisée pour le régulateur :

$$T_1 = 18 \text{ }^\circ\text{C}$$

$$\Phi = 0.25 \text{ W}$$

$$\Phi = h \cdot S \cdot \Delta T$$

$$\Delta T = \frac{\Phi}{h \cdot S} = \frac{\Phi}{(h_p \cdot S_p) + (h_i \cdot S_i)} = \frac{0.25}{(58.76 \cdot 8.07 \cdot 10^{-5}) + (1265.53 \cdot 2.794 \cdot 10^{-5})} = 6.24 \text{ }^\circ\text{C}$$

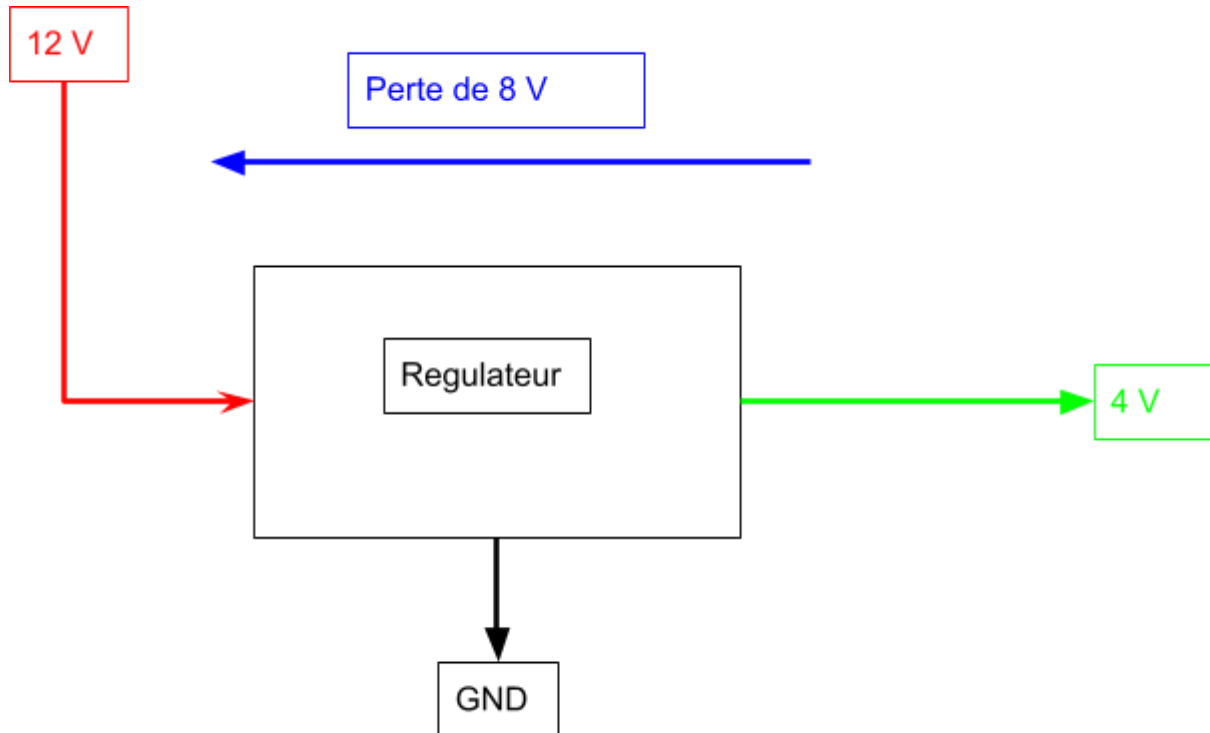
$$\text{Soit } \Delta T = T_2 - T_1, T_2 = \Delta T + T_1, \text{ soit } T_2 = 6.24 + 18 = 24.24 \text{ }^\circ\text{C}.$$

Temperature Range	LM1575	$-55^\circ\text{C} \leq T_J \leq +150^\circ\text{C}$
	LM2575/LM2575HV	$-40^\circ\text{C} \leq T_J \leq +125^\circ\text{C}$

La limite de température du composant est de  $125 \text{ }^\circ\text{C}$ . Il n'est donc pas nécessaire d'implémenter un radiateur.

## Pourquoi un dévolteur à découpage et pas un simple régulateur ?

Le régulateur dissipe la tension d'écart entre la sortie et l'entrée,



On a un courant de 0.843 A qui traverse la sortie donc on a une puissance perdue de  $P_{\text{perdue}} = V_{\text{perdue}} * I$  soit  $P_{\text{perdue}} = 0.843 * 8 = 6.744 \text{ W}$

En reprenant les mêmes dimensions que le régulateur et la même composition,

$$\Phi = 6.744 \text{ W}$$

$$\Delta T = \frac{\Phi}{h \cdot S} = \frac{\Phi}{(h_p \cdot S_p) + (h_i \cdot S_i)} = \frac{6.744}{(58.76 * 8.07 * 10^{-5}) + (1265.53 * 2.794 * 10^{-5})} = 168.38 \text{ } ^\circ\text{C}$$

$$\text{Donc } T_2 = T_1 + \Delta T \quad T_2 = 18 + 168.38 = 186.18 \text{ } ^\circ\text{C}$$

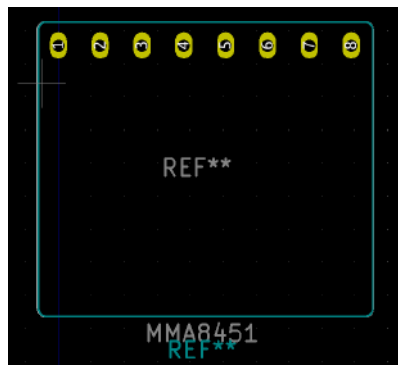
Notre composant ne résisterait pas à une telle chaleur car on rappelle que dans sa datasheet le composant résiste jusqu'à 125 °C. C'est donc pour cette raison que nous avons choisi un dévolteur à découpage.

## Création de la carte sur Kicad

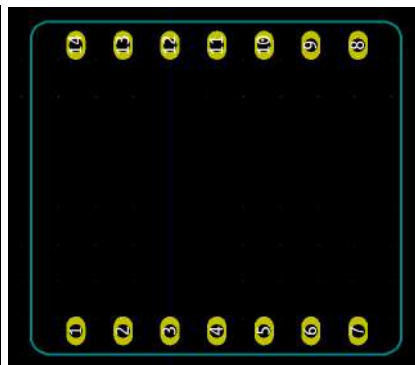
Afin que l'alarme prenne le moins de place possible, nous aurons deux PCBs. Le premier étant le dévolteur et le deuxième étant la partie fonctionnelle. Ces deux PCBs s'empilent grâce à quatre barres droites et sont reliés pour l'alimentation avec deux fils assez épais pour réduire la résistance.

### Création d'empreintes

Vu que nous n'avons pas les empreintes pour les composants, nous avons fait une version basique, mais fonctionnel de ces derniers :



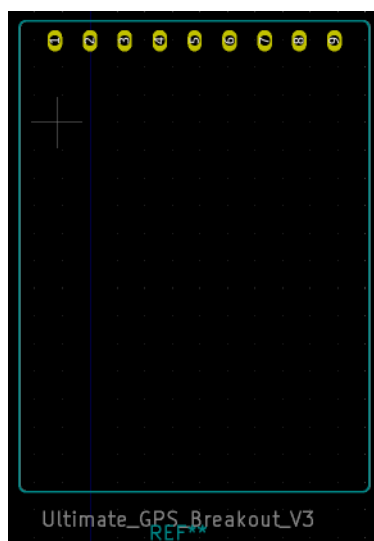
Accéléromètre



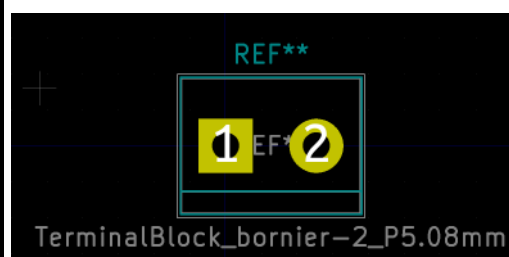
Seeeduino



GSM



GPS



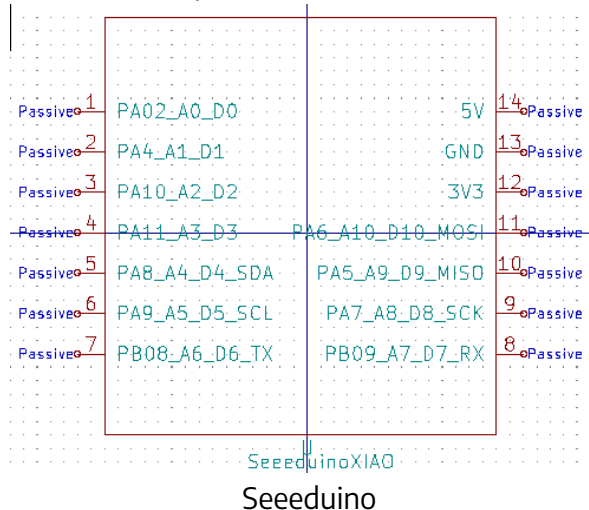
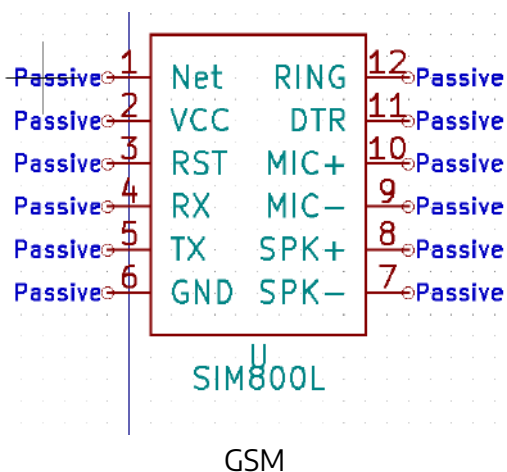
Bornier

Nous n'avons pas inséré l'entrée d'alimentation pour le Seeeduino, car nous devons la relier manuellement via une soudure et des fils directement à la sortie du dévolteur. Le problème est que cette partie sera donc fragile.



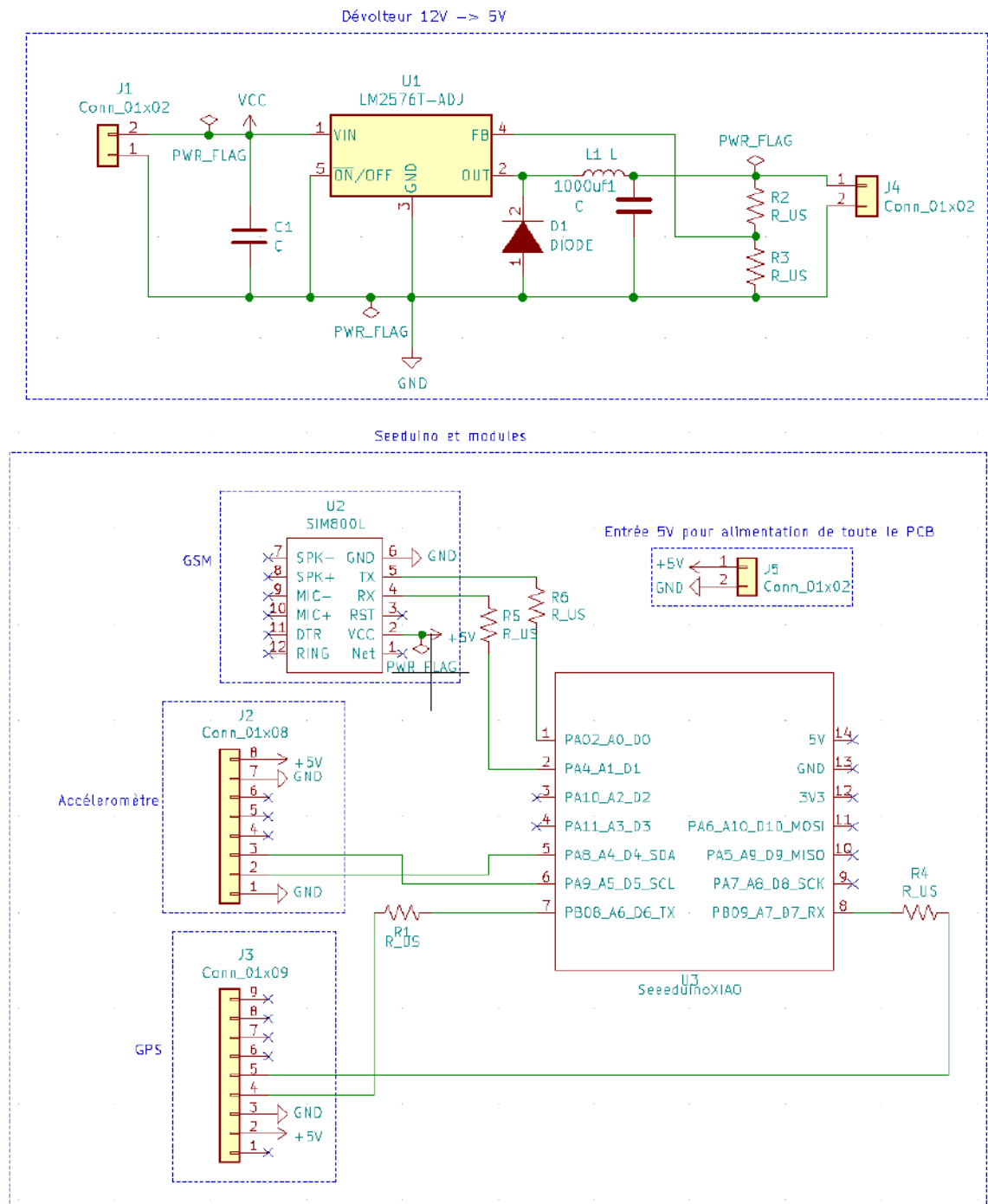
## Création de symboles

Nous n'avons pas les symboles pour le SIM800L et pour le Seeeduino (nous utilisons de simples connecteurs en ligne pour les autres modules) ainsi, nous avons donc dû les créer.



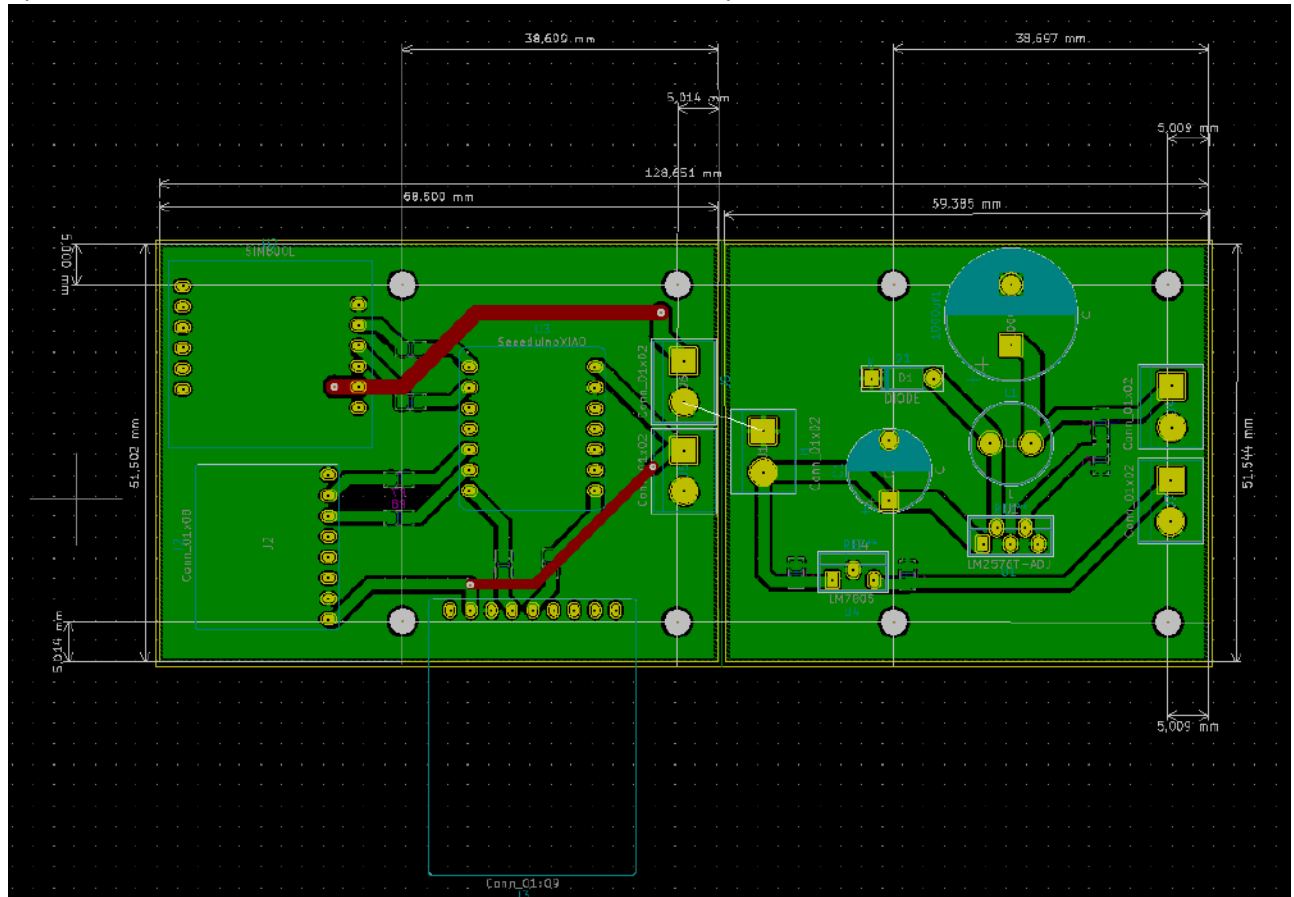
## Schémas

Le schéma du dévolteur ainsi que celui du Seeduino avec ses modules est effectué sur un seul et même schéma afin de pouvoir fabriquer les PCBs sur une seule et même plaque de cuivre.



Afin de pouvoir découper les deux PCBs nous avons regardé une [vidéo](#) de Mr Eric Peronnin afin de savoir comment mettre en panneau plusieurs PCBs.

Après avoir fini les PCBs nous avons le résultats suivant pour les PCBs:

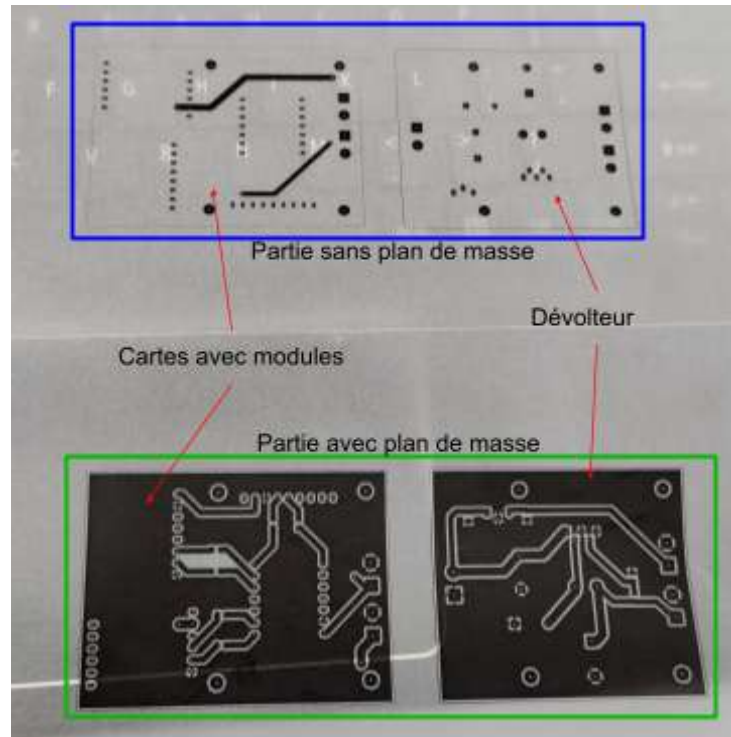




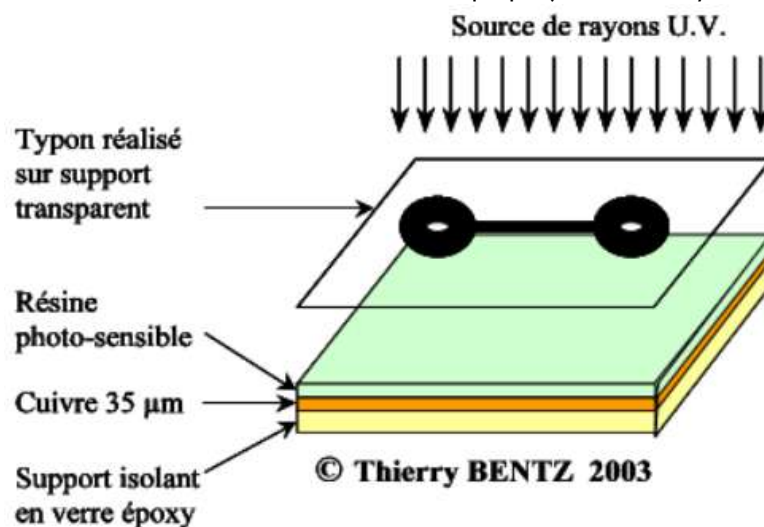
## Fabrication du PCB

### Première étape:

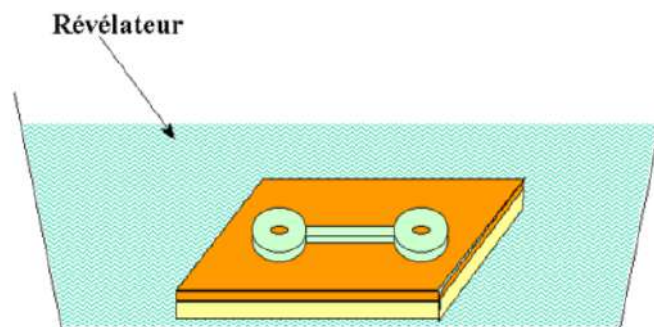
Pour pouvoir faire notre pcb chimiquement, nous devons tout d'abord imprimer sur un papier calque(typon) les deux deux faces des deux PCB's:



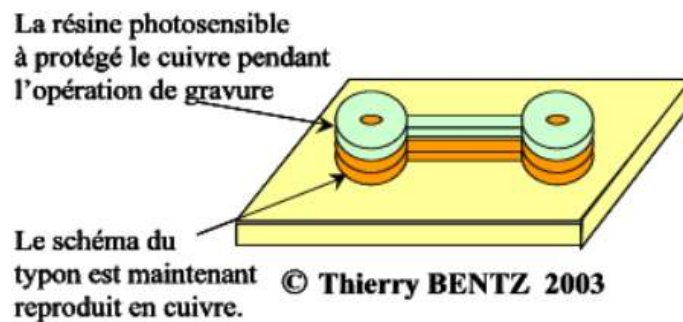
Après avoir fait le typon, nous mettons au milieu une plaque de cuivre ayant une résine photosensible et nous la mettons dans une machine qui projette des rayons UV's:



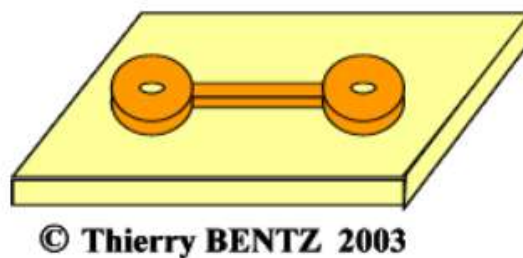
Ensuite, nous mettons cette carte dans un bain révélateur afin de dissoudre la résine insolée par les U.V et donc mettre le cuivre à nu à cet endroit :



Ensuite on met la carte dans une solution de perchlorure de fer ainsi le cuivre qui n'est plus protégé par la résine se trouve dissout et disparaît de la carte:



Finalement on enlève la résine photosensible en le trempant la carte dans un liquide:

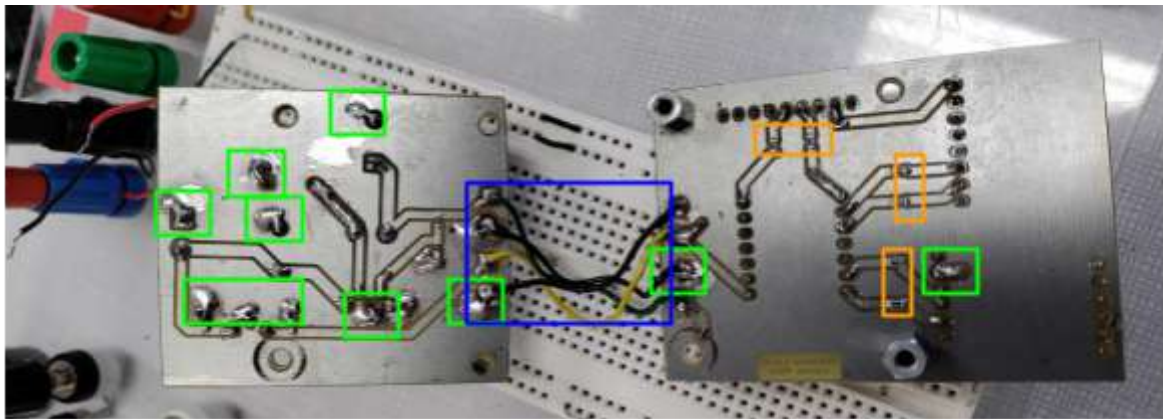


## Deuxième étape:

Lors de la deuxième étape nous devons souder les composants aux deux pcbs et connecter ces deux derniers entre eux. Lorsque nous avons reçu nos PCBs nous avons remarqué qu'il y avait un problème. Aucune masse n'était reliée au plan de masse. Pour remédier à cela nous avons soudé à la main les masses aux plans de masse à l'aide d'un fer à souder.

Nous avons relié les deux PCBs à l'aide de fils multibrins afin d'éviter que les fils se cassent à force que les PCBs soient manipulés.

Vu du dessous:

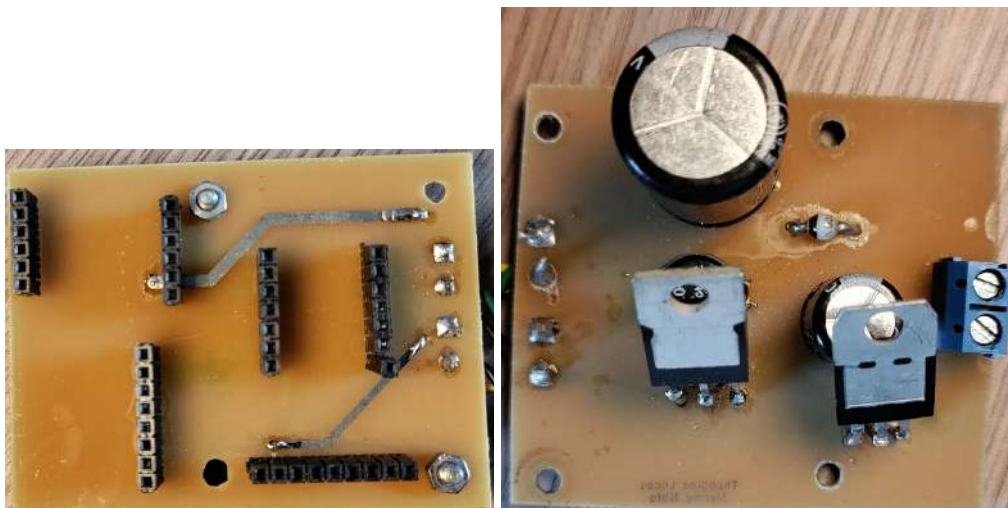


Soudure afin de connecter les masses

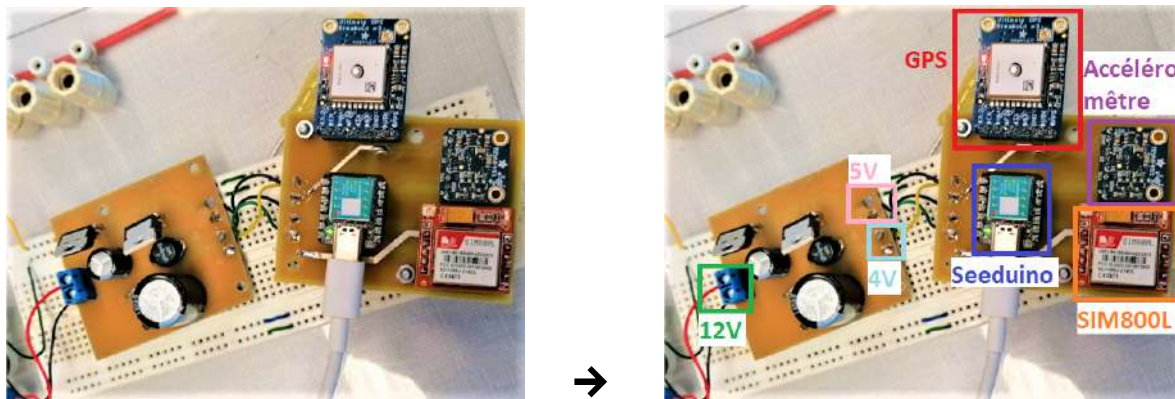
Connection du dévolteur aux modules avec des fils multibrins

CMS de 1000 ohm pour protéger les entrées et sorties Rx et Tx

Vu du dessus sans les modules:



vu du dessus avec les modules:



## Code et test:

Afin de réaliser le code final, j'ai commencé par étudier les différents codes et voir comment chacun fonctionne.

▷ Pour l'accéléromètre, j'ai regardé quels étaient les intervalles de données des différentes accélérations et étudié leurs sensibilités aux différentes variations auxquelles l'accéléromètre peut être soumis (vibrations, mouvement horizontal, vertical, variation d'angle).

J'en ai déduit les intervalles suivants afin que l'accéléromètre ne soit pas perturbé par de petites variations :

	Variation min	Variation max
x	-7	7
y	-7	7
z	7	17

▷ Pour le GPS j'ai regardé comment les données étaient transmises sur le moniteur série et lesquelles étaient utiles.

Lorsqu'une accélération inhabituelle est détectée, nous mettons une variable *état* à 1 afin que les coordonnées GPS soient transmis en continu et non pas seulement quand il y a une accélération. Cette variable pourra ensuite être remise à 0 et l'envoi des coordonnées GPS arrêtées via l'envoi d'un message par le propriétaire indiquant que la moto n'est pas volée, par exemple en envoyant "MOTO OK" (ceci est une application future afin d'améliorer l'alarme).

Ainsi quand on rassemble l'accéléromètre et le GPS on a le résultat suivant lorsqu'un mouvement est exercé:

<pre>LA MOTO EST ENTRAIN D'ÊTRE BOUGÉ Date: 3/4/2022 Fixation: 1 qualité: 2 Localisation: 4714.9336N, 133.5461W Vitesse (km/h): 0 Angle: 29.34 Altitude: 56.90 Satellites: 6</pre>	<p>données qui seront envoyées par le GSM au propriétaire</p> <p>Données non nécessaires et qui ne seront pas envoyées mais qui sont utiles pour vérifier les coordonnées</p>
--	---

```
$GPGGA,152741.000,4714.9336,N,00133.5461,W,2,06,1.43,56.9,M,48.6,M,0000,0000*47
$GPRMC,152741.000,A,4714.9336,N,00133.5461,W,0.30,18.54,030422,,D*4F
$PGTOP,11,2*6E
$GPGGA,152742.000,4714.9337,N,00133.5460,W,2,06,1.43,56.9,M,48.6,M,0000,0000*44
$GPRMC,152742.000,A,4714.9337,N,00133.5460,W,0.44,16.20,030422,,D*42
$PGTOP,11,2*6E
$GPGGA,152743.000,4714.9338,N,00133.5460,W,2,06,1.43,56.9,M,48.6,M,0000,0000*4A
$GPRMC,152743.000,A,4714.9338,N,00133.5460,W,0.38,3.99,030422,,D*71
$PGTOP,11,2*6E
$GPGGA,152744.000,4714.9339,N,00133.5460,W,2,06,1.64,56.9,M,48.6,M,0000,0000*49
$GPRMC,152744.000,A,4714.9339,N,00133.5460,W,0.34,2.25,030422,,D*7D
$PGTOP,11,2*6E
$GPGGA,152745.000,4714.9340,N,00133.5460,W,2,06,1.64,56.9,M,48.6,M,0000,0000*46
$GPRMC,152745.000,A,4714.9340,N,00133.5460,W,0.47,3.05,030422,,D*75
$PGTOP,11,2*6E
$GPGGA,152746.000,4714.9341,N,00133.5459,W,2,06,1.64,56.9,M,48.6,M,0000,0000*4E
$GPRMC,152746.000,A,4714.9341,N,00133.5459,W,0.52,5.75,030422,,D*78
$PGTOP,11,2*6E
```

On vérifie sur un site de [localisation GPS](#) que les coordonnées sont bonnes :



Les coordonnées sont bonnes ainsi on peut conclure que nos modules fonctionnent. Cependant, petites corrections, ils fonctionnent ensemble sur un breadboard et non pas sur le pcb([voir le problème](#)).

▷ Pour l'affectation avec le GSM, nous n'avons pas réussi à le faire fonctionner, mais nous avons fait un code qui pourrait fonctionner en étudiant son code et regardant comment d'autre personne l'ont utilisé.



## Les problèmes rencontrés lors du projet :

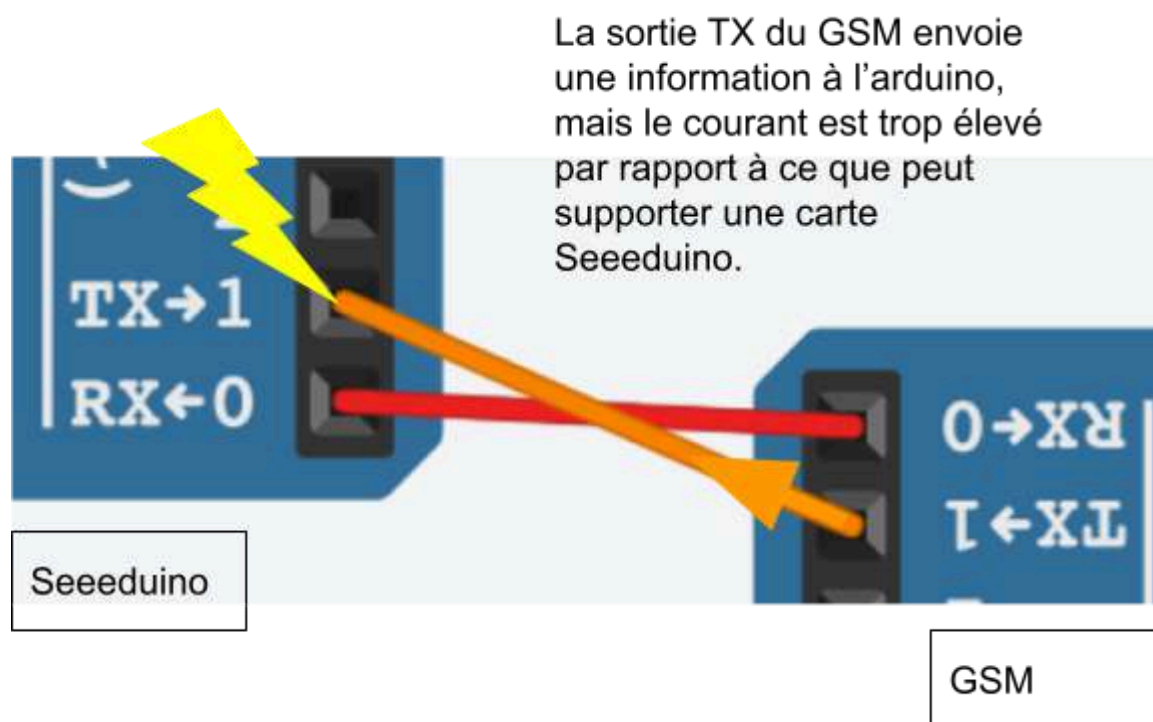
### Le GSM :

#### Rx et TX:

La première fois que nous avons effectué le montage et utilisé le code, nous n'avons pas pu envoyer de message, car la carte SIM était bloquée. Nous avons donc dû la mettre sur un portable et la débloquer afin que le module GSM puisse l'utiliser.

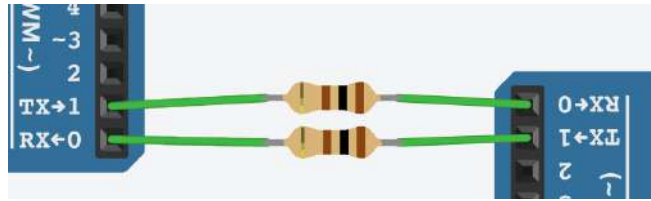
Le deuxième problème est arrivé lors de l'envoi du message. Le module demandant un courant plus important que ce que la carte Arduino peut fournir, elle ne pouvait envoyer aucun message. Pour résoudre ce problème, nous avons branché l'alimentation du SIM800L directement sur l'alimentation continue sur la paillasse. Ensuite, pour éviter d'avoir un problème avec la cage de faraday du bâtiment, nous avons mis l'antenne près de la fenêtre.

Le troisième problème est survenu lorsque nous avons refait le montage avec le Seeeduino. Nous avons accidentellement inversé le TXD et le RXD (de manière informatique) sur la carte SIM800L et sur le Seeeduino. Ceci a provoqué un court-circuit sur la carte GSM, car les deux sorties se sont envoyées une information, ainsi, un court-circuit c'est créé.



Pour réduire le risque de reproduire cette erreur ou d'éviter de griller un module si nous commettons une erreur, nous avons décidé avec le professeur de placer une résistance d'un minimum de  $100\Omega$  sur la sortie TXD et l'entrée RXD de chaque module afin de réduire le courant entrant ou sortant au maximum.

Nous représentons cette solution sur un schéma simple créé sur tinkercad :



Ainsi, même s'il y a une quelconque modification informatique et que TXS devient RXD le courant sera trop insuffisant pour détériorer le matériel.

### **Courant :**

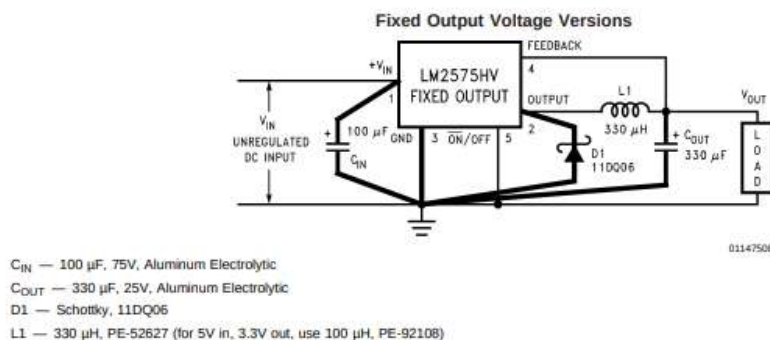
Lorsque nous avons réalisé les tests pour pouvoir faire fonctionner le GSM nous avons remarqué qu'il consommait une grande quantité de courant quand il essayait de se connecter ou d'envoyer un message. Cette demande de courant étant spontanée, le dévolteur ne peut pas forcément subvenir à cette demande. Ainsi pour régler ce problème, nous avons fixé aux bornes de l'arrivée de tension du sim800L un condensateur de 1500uF - 16V qui permet de fournir le courant nécessaire lors d'éventuel pic.



### **Le dévolteur :**

Pour le montage du dévolteur, nous avons rencontré un problème, car on avait un mauvais montage sans les résistances en sortie :

#### **Test Circuit and Layout Guidelines (Continued)**



On a dû parcourir la datasheet de nouveau afin de trouver un montage qui cette fois-ci a fonctionné. Voir le bon montage sur la partie :

[Etude pratique :](#)

Du dévolteur.



## **Le PCB :**

Nous avons rencontré un souci lors de l'implémentation du dévolteur sur notre carte électronique, en effet, nous n'avions pas la tension attendue en sortie après de longue réflexion sur :

- Est-ce le composant ?
- Est-ce une mauvaise polarisation?
- Est-ce un problème avec le couple de résistance ?
- Est-ce un souci au niveau de l'alimentation ?

En réalité, le problème venait d'un composant, la diode, en effet, la diode était une diode traditionnelle et pas une diode schottky. La différence est le temps de réponse entre ces deux diodes, la commutation du transistor étant très rapide, la diode schottky permet de bien transmettre les 4 Volts.

La carte qui réunit les différents modules a été imprimée dans le mauvais sens, conséquence il a fallu réimprimer la carte comprenant les modules.

## **Le GPS :**

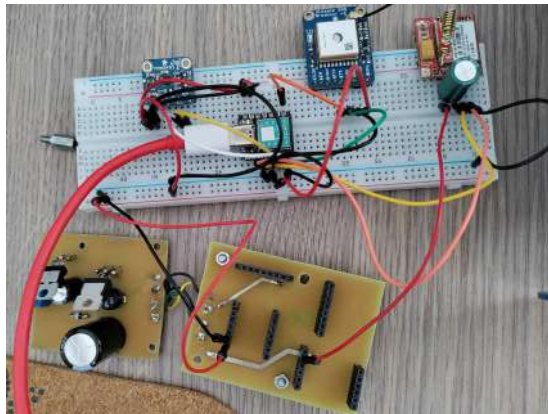
Nous avons remarqué que le GPS pouvait avoir des difficultés à se connecter à des satellites, ainsi, il est nécessaire de lui assigner une antenne, ce qui résout ce problème et permet au GPS de se connecter en quelques secondes.

## **L'accéléromètre:**

Lorsque nous avons implanté l'accéléromètre et que nous avons essayé de le faire fonctionner avec le seeduino, nous n'avons pas réussi à recevoir les données sur les différentes accélérations sur le moniteur série. Cela est dû au plan de masse des deux PCBS qui est commun.

Pour éviter ce problème, nous devons relier les masses des différents modules directement au seeduino et après relier la masse de ce dernier au plan de masse du dévolteur. Ceci nous permet de ne pas avoir deux plans de masse qui créent des conflits de transmission de données.

Solution présentée sur un breadboard:





## **Réunion pendant le projet :**

Même si nous sommes seulement en binôme, il est important de faire de manière récurrente des réunions afin de faire le point sur comment on avance sur le projet et ainsi parler des différentes contraintes rencontrées.

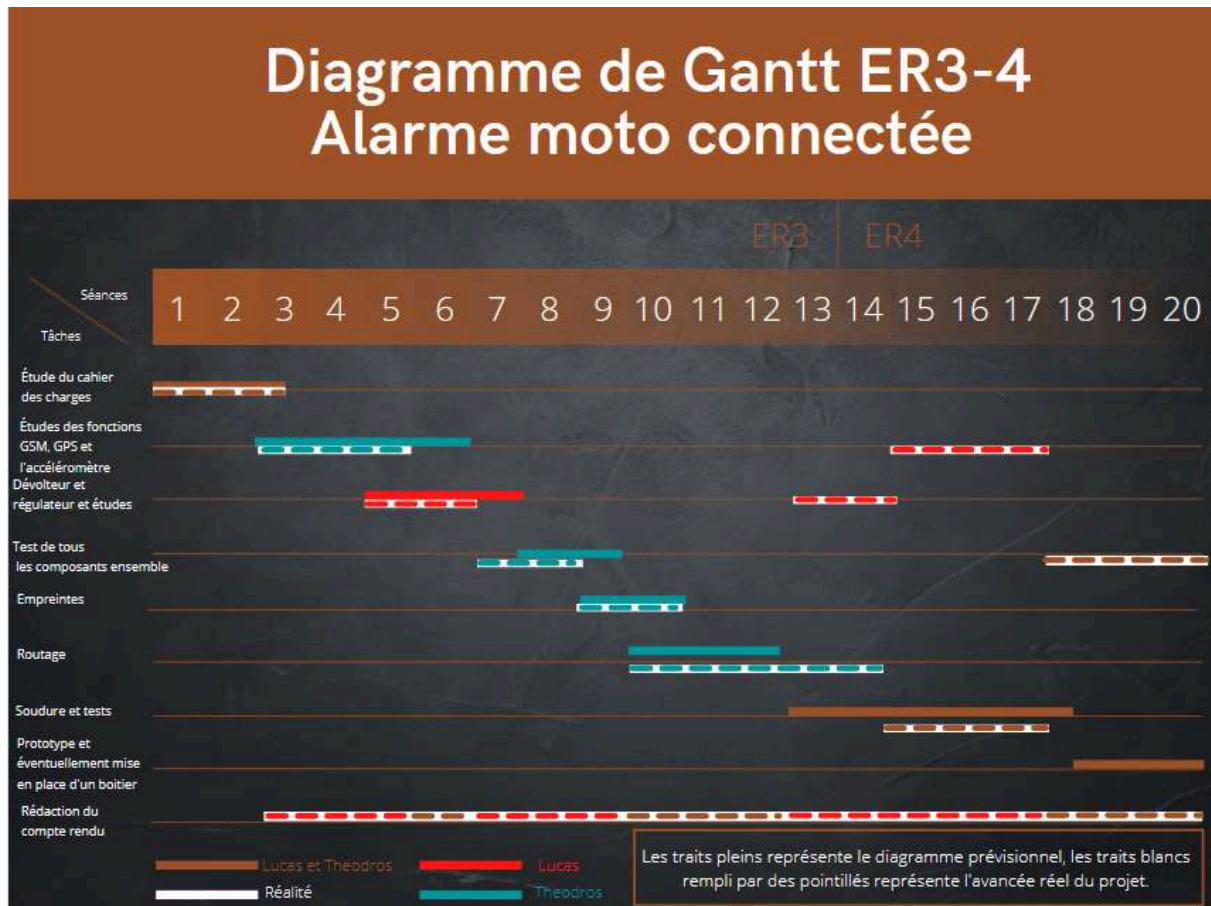
Chaque début de séance on a échangé pour mettre au point qui allait faire quoi telle séance au milieu pour savoir comment la tâche avance comme ça on peut s'aider si un membre du groupe a un problème, en fin de séance et on remplit le tableau de bord et notre Gantt afin de pouvoir nous situer sur comment le projet avance.

## **Tableau de bord**

Contrairement au projet d'ER2, nous ne sommes pas divisés par fonction, mais travaillons ensemble. Ce tableau de bord nous permettra de suivre en temps réel notre projet et nous permettra de remplir notre diagramme de Gantt.

Alarme de Moto	Theodros	Lucas
ER3		
Séance 1	Début du projet	
Séance 2	Découverte du projet, documentation sur comment dérouler le projet	
Séance 3	Accéléromètre (fonctionnel)	
Séance 4	GSM (fonctionnel)	
Séance 5	Test GPS (fonctionnel)	Mise en place d'un dévolteur (non-fonctionnel)
Séance 6	Test et montage du dévolteur (fonctionnel)	
Séance 7	Rassemblement globale des composants	Démarrage du compte rendu
Séance 8	Carte GSM cassé vérification, du module GPS et fonctionnement d'un nouveau GSM	Continuation du compte rendu
Séance 9	Début empreintes	
Séance 10	Continue empreinte et la schématique	
Séance 11	Routage	Compte rendu
Séance 12	SOUTENANCE ER3	
	Routage	Rectification dévolteur et compte-rendu
Séance 13	ABS	Mise en place du dévolteur 4 V plus régulateur 5 V
ER4		
Séance 14	Routage du PCB et correction des erreurs	Calcul théorique de déperdition thermique des dévolteurs
Séance 15	perçage et soudure CMS	
	Impression carte	Calcul théorique de déperdition thermique des dévolteurs
Séance 16	Soudure de la carte	Étude GSM
Séance 17	Dépannage dévolteur (voir <a href="#">Le PCB</a> )	
Séance 18	Dépannage carte qui comprend les modules (voir <a href="#">Le PCB</a> )	
Séance 19	Essais GSM	
Séance 20	SOUTENANCE ER4	

## Diagramme de Gantt :



Lors de l'ER4, nous avons rencontré beaucoup d'aléas, nous avons prévu des délais supplémentaires pour pallier ces problèmes.

Nous avons réalisé des études théoriques, telles que celle du GSM (calcul de sa bande de fréquence, calcul du nombre d'octets transmissible par le GSM), on a également réalisé une étude théorique sur l'alimentation à découpage et sur les déperditions thermiques de ces alimentations.

On a fini le routage, nous avons pu imprimer la carte, la souder, la tester et implémenter nos modules.

Nous avons aussi fait fonctionner différents modules ensemble.

Malheureusement, le projet n'a pas pu aboutir dans les temps. En effet, il manquait seulement que le GSM fonctionne pour que le projet soit terminé.

## **Calcul de budget réel**

En théorie, nous avons prévu 106.25 € nous avons dépensé 93.46 € en effet, on a malheureusement cassé un GSM et deux dévolteurs et 6 résistances CMS soit  $75.89 + 9.99 + 2 \times 3.76 + 0.008 \times 8 = 93.46$  €

Pour le budget du personnel, on estime que le nombre d'heure prévu est suffisant donc le budget théorique du personnel est respecté soit 2285.7 €

Le budget réel du projet est de  $2285.7 + 93.46 = 2379.16$  € on avait prévu 2400 € on est légèrement inférieur à cette estimation.

On tient à rappeler que les coûts liés à la location des locaux, du matériel, l'électricité [...] ne sont pas comptés également tout ce qui pourrait donner suite à une commercialisation du projet (transport, pub [...]).

## **Conclusion du projet ER4**

Nous pouvons donc conclure, que nous nous sommes réellement investi dans ce projet. Nous avons donné le meilleur de nous-mêmes à réaliser ce projet, et même s'il n'a malheureusement pas pu aboutir, nous sommes fiers du travail fourni. En effet, ce projet a été une grande découverte pour nous, nous ne connaissions absolument pas le fonctionnement des différents modules et malgré cela, nous avons appris à les connaître, on termine donc le projet par une carte fonctionnel, en effet, les deux dévolteurs sont fonctionnels, la carte Seeeduino communique correctement avec le GPS et l'accéléromètre, lorsque l'on bouge l'accéléromètre, on a un scan des données GPS, ils manquaient seulement que le GSM envoie ces données par SMS. Ce projet nous tient à cœur et nous espérons que vous avez apprécié de le partager avec nous à travers ce rapport écrit.

Nous remercions M. CABRIOLÉ qui nous a accompagnés durant toute la durée de ce projet et qui nous a permis de nous aider à notamment comprendre nos erreurs.

Nous remercions également l'IUT qui nous a fourni tout le matériel nécessaire à la réalisation du projet.

## Annexe:

Code Accéléromètre:

```
#include <Wire.h>
#include <Adafruit_MMA8451.h>
#include <Adafruit_Sensor.h>

Adafruit_MMA8451 mma = Adafruit_MMA8451();

void setup(void) {
  Serial.begin(9600);

  /* Affiche un message d'erreur si l'accéléromètre ne démarre pas */
  if (!mma.begin()) {
    Serial.println("Ne peux pas démarrer");
    while (1);
  }

  /* Affiche un message quand l'accéléromètre démarre */
  Serial.println("MMA8451 Trouvé!");

  mma.setRange(MMA8451_RANGE_2_G);

  Serial.print("Intervalle = "); Serial.print(2 << mma.getRange());
  Serial.println("G");
}

void loop() {
  /* Créer un nouvel événement pour le capteur*/
  sensors_event_t event;
  mma.getEvent(&event);

  /* Affiche les résultats en m/s^2 */
  Serial.print("X: \t"); Serial.print(event.acceleration.x);
  Serial.print("\t");
  Serial.print("Y: \t"); Serial.print(event.acceleration.y);
  Serial.print("\t");
  Serial.print("Z: \t"); Serial.print(event.acceleration.z);
  Serial.print("\t");
  Serial.println("m/s^2 ");
}
```

```
    delay(500);  
}
```

Code GPS:

```
#include <Adafruit_GPS.h>  
  
// what's the name of the hardware serial port?  
#define GPSSerial Serial1  
  
// Connect to the GPS on the hardware port  
Adafruit_GPS GPS(&GPSSerial);  
  
// Set GPSECHO to 'false' to turn off echoing the GPS data to the Serial  
// console  
// Set to 'true' if you want to debug and listen to the raw GPS  
// sentences  
#define GPSECHO false  
uint32_t timer = millis();  
  
void setup()  
{  
    //while (!Serial); // uncomment to have the sketch wait until Serial  
    // is ready  
  
    // connect at 115200 so we can read the GPS fast enough and echo  
    // without dropping chars  
    // also spit it out  
    Serial.begin(115200);  
    Serial.println("Adafruit GPS library basic parsing test!");  
  
    // 9600 NMEA is the default baud rate for Adafruit MTK GPS's- some use  
    // 4800  
    GPS.begin(9600);  
    // uncomment this line to turn on RMC (recommended minimum) and GGA  
    // (fix data) including altitude  
    GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);  
    // uncomment this line to turn on only the "minimum recommended" data  
    //GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMONLY);  
    // For parsing data, we don't suggest using anything but either RMC  
    // only or RMC+GGA since  
    // the parser doesn't care about other sentences at this time  
    // Set the update rate  
    GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ); // 1 Hz update rate
```



```

    // For the parsing code to work nicely and have time to sort thru the
    data, and
    // print it out we don't suggest using anything higher than 1 Hz

    // Request updates on antenna status, comment out to keep quiet
    GPS.sendCommand(PGCMD_ANTENNA);

    delay(1000);

    // Ask for firmware version
    GPSSerial.println(PMTK_Q_RELEASE);
}

void loop() // run over and over again
{
    // read data from the GPS in the 'main loop'
    char c = GPS.read();
    // if you want to debug, this is a good time to do it!
    if (GPSECHO)
        if (c) Serial.print(c);
    // if a sentence is received, we can check the checksum, parse it...
    if (GPS.newNMEAreceived()) {
        // a tricky thing here is if we print the NMEA sentence, or data
        // we end up not listening and catching other sentences!
        // so be very wary if using OUTPUT_ALLDATA and trying to print out
data
        Serial.print(GPS.lastNMEA()); // this also sets the
newNMEAreceived() flag to false
        if (!GPS.parse(GPS.lastNMEA())) // this also sets the
newNMEAreceived() flag to false
            return; // we can fail to parse a sentence in which case we should
just wait for another
    }

    // approximately every 2 seconds or so, print out the current stats
    if (millis() - timer > 2000) {
        timer = millis(); // reset the timer
        Serial.print("\nTime: ");
        if (GPS.hour < 10) { Serial.print('0'); }
        Serial.print(GPS.hour, DEC); Serial.print(':');
        if (GPS.minute < 10) { Serial.print('0'); }
        Serial.print(GPS.minute, DEC); Serial.print(':');
        if (GPS.seconds < 10) { Serial.print('0'); }
        Serial.print(GPS.seconds, DEC); Serial.print('.');
        if (GPS.milliseconds < 10) {

```

```

    Serial.print("00");
  } else if (GPS.milliseconds > 9 && GPS.milliseconds < 100) {
    Serial.print("0");
  }
  Serial.println(GPS.milliseconds);
  Serial.print("Date: ");
  Serial.print(GPS.day, DEC); Serial.print('/');
  Serial.print(GPS.month, DEC); Serial.print("/20");
  Serial.println(GPS.year, DEC);
  Serial.print("Fix: "); Serial.print((int)GPS.fix);
  Serial.print(" quality: "); Serial.println((int)GPS.fixquality);
  if (GPS.fix) {
    Serial.print("Localisation: ");
    Serial.print(GPS.latitude, 4); Serial.print(GPS.lat);
    Serial.print(", ");
    Serial.print(GPS.longitude, 4); Serial.println(GPS.lon);
    Serial.print("Vitesse (nœuds): "); Serial.println(GPS.speed);
    Serial.print("Angle: "); Serial.println(GPS.angle);
    Serial.print("Altitude: "); Serial.println(GPS.altitude);
    Serial.print("Satellites: "); Serial.println((int)GPS.satellites);
  }
}
}

```

Code GSM:

```

#include <SoftwareSerial.h>
SoftwareSerial sim(10, 11);
int _timeout;
String _buffer;
String number = "+33627692805"; //-> change en fonction du propriétaire
void setup() {
  delay(7000); //Attends 7 secondes afin d'être sur que le module reçoit
  bien un signal
  Serial.begin(9600);
  _buffer.reserve(50);
  Serial.println("Système allumé...");
  sim.begin(9600);
  delay(1000);
  Serial.println("s pour envoyer un sms, r pour recevoir un SMS");
}
void loop() {
  if (Serial.available() > 0)
    switch (Serial.read())
    {

```

```

        case 's':
            SendMessage();
            break;
        case 'r':
            RecieveMessage();
            break;
    }
    if (sim.available() > 0)
        Serial.write(sim.read());
}
void SendMessage()
{
    Serial.println ("Sending Message");
    sim.println("AT+CMGF=1"); //Met le module GSM en mode texte
    delay(200);
    //Serial.println ("Set SMS Number");
    sim.println("AT+CMGS=\"\" + number + "\"\r"); //Numéro de téléphone à
    qui est envoyée le message
    delay(200);
    String SMS = "Connection faite avec le propriétaire"; //informe que la
    connection est bien faite
    sim.println(SMS);
    delay(100);
    sim.println((char)26); // ASCII code de CTRL+Z
    delay(200);
    _buffer = _readSerial();
}
void RecieveMessage()
{
    Serial.println ("SIM800L Read an SMS");
    sim.println("AT+CMGF=1");
    delay (200);
    sim.println("AT+CNMI=1,2,0,0,0"); // Commande pour recevoir un SMS
    delay(200);
    Serial.write ("Message reçu"); //envoie un message au propriétaire
    pour l'informer que le message est bien reçu
}
String _readSerial() {
    _timeout = 0;
    while (!sim.available() && _timeout < 12000 )
    {
        delay(13);
        _timeout++;
    }
    if (sim.available()) {

```

```
    return sim.readString();  
}  
}
```

Code fonction accéléromètre + Localisation:

```
//Librairies  
#include <Wire.h>  
#include <Adafruit_MMA8451.h>  
#include <Adafruit_Sensor.h>  
#include <Adafruit_GPS.h>  
  
Adafruit_MMA8451 mma = Adafruit_MMA8451();  
// what's the name of the hardware serial port?  
#define GPSSerial Serial1  
  
// Connect to the GPS on the hardware port  
Adafruit_GPS GPS(&GPSSerial);  
  
// Nous mettons false pour ne pas recevoir sur le moniteur série les  
données non nécessaire  
#define GPSECHO false  
  
//Défini un timer pour calibrer le GPS  
uint32_t timer = millis();  
  
unsigned long previousMillis = 0;           // Permet de garder le dernier  
temps enregistré  
  
// constants won't change:  
const long interval = 5000;                // Interval pour envoyer les messages  
5s  
  
//etat de la moto (tombé ou pas par exemple)  
int etat = 0;  
  
void setup()  
{  
    etat = 0;  
    while (!Serial); // On attends que le GPS soit prêt  
  
    // connecter à 115200 afin que nous puissions lire le GPS assez  
    rapidement et l'écho sans laisser tomber les caractères
```

```

Serial.begin(115200);
Serial.println("Test de la librairie du GPS");

//9600 NMEA est le débit par défaut pour les GPS Adafruit MTK
GPS.begin(9600);
Serial.begin(9600);

Serial.println("Les composants sont prêts à fonctionner");

//Si l'accéléromètre ne fonctionne pas on affiche ce message
if (!mma.begin()) {
    Serial.println("L'accéléromètre ne fonctionne pas. Veuillez
vérifier le composant");
    while (1);
}
Serial.println("L'accéléromètre fonctionne ");

mma.setRange(MMA8451_RANGE_2_G);

Serial.print("Range = "); Serial.print(2 << mma.getRange());
Serial.println("G");

//commenter cette ligne pour ne pas allumer le RMC (minimum
recommandé) et le GGA (données de repère), comprenant l'altitude
GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
// décommenter cette ligne pour n'activer que les données "minimum
recommandé"
//GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMONLY);

//Définir le taux de mise à jour (1Hz est le minimum recommandé
GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ); // 1 Hz update rate

// Demander des mises à jour sur l'état de l'antenne, commenter pour
ne pas recevoir
GPS.sendCommand(PGCMD_ANTENNA);

delay(1000);

//Demande la version du micrologiciel
GPSSerial.println(PMTK_Q_RELEASE);
}

void loop()

```

```

{
    //Vérifie que c'est le moment d'envoyer le message
    unsigned long currentMillis = millis();

    // On regarde les données de l'accéléromètre
    mma.read();
    sensors_event_t event;
    mma.getEvent(&event);

    //On regarde si les accélération sur les différents axes sont
    inchangés
    //Ainsi, si la moto reste sur le même angle mais se fait par exemple
    frapper par qqc
    //un message sera envoyé car les accélérations auront changés
    if ((event.acceleration.x < -7 || event.acceleration.x > 7) ||
        (event.acceleration.y < -7 || event.acceleration.x > 7) ||
        (event.acceleration.z < 7 || event.acceleration.z > 17))
    {
        //La moto étant en mouvement nous mettons l'etat à 1 afin que nous
        continuons de d'envoyer la localisation
        // La variable etat pourra être remise à 0 avec un message du
        propriétaire lorsque nous rajouterons le sim800l
        etat = 1;
    }

    while(etat == 1){
        //lis les données du GPS dans la boucle principale
        char c = GPS.read();

        //Vérifie que tout soit bon et debug
        if (GPSECHO)
        if (c) Serial.print(c);
        //Regarde si qqc a été reçu
        if (GPS.newNMEAreceived()) {
            Serial.print(GPS.lastNMEA());
            if (!GPS.parse(GPS.lastNMEA()))
                return;
        }
        //Si c'est le bon moment d'envoyer le message on l'envoie
        if (currentMillis - previousMillis >= interval) {
            //Sauvegarde le dernier envoie de données
            previousMillis = currentMillis;
            Serial.println("");
            Serial.println("LA MOTO EST ENTRAIN D'ÊTRE BOUGÉ");

            //On envoie toutes les données voulu(localisation, vitesse de

```





```

#define GPSSerial Serial1

// Connect to the GPS on the hardware port
Adafruit_GPS GPS(&GPSSerial);

// Nous mettons false pour ne pas recevoir sur le moniteur série les
données non nécessaire
#define GPSECHO false

uint32_t timer = millis(); //Défini un timer pour calibrer le GPS
unsigned long previousMillis = 0; // Permet de garder le dernier
temps enregistré
const long interval = 5000; // Interval pour envoyer les messages
5s
int etat = 0; //etat de la moto (tombé ou pas par exemple)
int _timeout;
String _buffer;
String number = "0637618385"; //-> change en fonction du propriétaire

void setup() {
  while (!Serial); // On attends que les modules soit prêts
  Serial.begin(9600);
  _buffer.reserve(50);
  Serial.println("Système allumé...");
  sim.begin(9600);
  // connecter à 115200 afin que nous puissions lire le GPS assez
rapidement et l'écho sans laisser tomber les caractères
  Serial.begin(115200);
  Serial.println("Test de la librairie du GPS");
  //9600 NMEA est le débit par défaut pour les GPS Adafruit MTK
  GPS.begin(9600);
  Serial.begin(9600);
  Serial.println("Les composants sont prêts à fonctionner");
  //Si l'accéléromètre ne fonctionne pas on affiche ce message
  if (!mma.begin()) {
    Serial.println("L'accéléromètre ne fonctionne pas. Veuillez
vérifier le composant");
    while (1);
  }
  Serial.println("L'accéléromètre fonctionne ");

  mma.setRange(MMA8451_RANGE_2_G);

  Serial.print("Range = "); Serial.print(2 << mma.getRange());
  Serial.println("G");

```

```

//commenter cette ligne pour ne pas allumer le RMC (minimum
recommandé) et le GGA (données de repère), comprenant l'altitude
GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
// décommenter cette ligne pour n'activer que les données "minimum
recommandé"
//GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMONLY);

//Définir le taux de mise à jour (1Hz est le minimum recommandé
GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ); // 1 Hz update rate

// Demander des mises à jour sur l'état de l'antenne, commenter pour
ne pas recevoir
GPS.sendCommand(PGCMD_ANTENNA);

delay(1000);

//Demande la version du micrologiciel
GPSSerial.println(PMTK_Q_RELEASE);
}
void loop() {
// On regarde les données de l'accéléromètre
mma.read();
sensors_event_t event;
mma.getEvent(&event);

//On regarde si les accélération sur les différents axes sont
inchangés
//Ainsi, si la moto reste sur le même angle mais se fait par exemple
frapper par qqc
//un message sera envoyé car les accélérations auront changés
if ((event.acceleration.x < -7 || event.acceleration.x > 7) ||
(event.acceleration.y < -7 || event.acceleration.x > 7) ||
(event.acceleration.z < 7 || event.acceleration.z > 17))
{
//La moto étant en mouvement nous mettons l'etat à 1 afin que nous
continuons de d'envoyer la localisation
// La variable etat pourra être remise à 0 avec un message du
propriétaire lorsque nous rajouterons le sim800l
etat = 1;
}
while (etat == 1) {
SendMessage();
}
}

```

```

    }
    if (sim.available() > 0)
        Serial.write(sim.read());
}
void SendMessage()
{
    //Vérifie que c'est le moment d'envoyer le message
    unsigned long currentMillis = millis();
    //lis les données du GPS dans la boucle principale
    char c = GPS.read();

    //Vérifie que tout soit bon et debug
    if (GPSECHO)
        if (c) Serial.print(c);
    //Regarde si qqc a été reçu
    if (GPS.newNMEAreceived()) {
        Serial.print(GPS.lastNMEA());
        if (!GPS.parse(GPS.lastNMEA()))
            return;
    }

    //Si c'est le bon moment d'envoyer le message on l'envoie
    if (currentMillis - previousMillis >= interval) {
        sim.print("AT+CMGF=1\r"); //Set the module to SMS
mode
        delay(100);
        sim.println("AT+CMGS=\"" + number + "\"\r"); //Numéro de téléphone
à qui est envoyé le message
        delay(500);
        //Sauvegarde le dernier envoie de données
        previousMillis = currentMillis;
        sim.println("");
        sim.println("LA MOTO EST ENTRAIN D'ÊTRE BOUGÉ");

    //On envoie toutes les données voulu(Localisation, vitesse de
déplacement,
        //qualité du signal, angle(permets de vérifier que la moto à bien
bougé).
        if (millis() - timer > 4000) {
            int v = GPS.speed * 0.539957; //passage de la vitesse en noeuds à
des km\h

            sim.print("Date: ");
            sim.print(GPS.day, DEC); sim.print('/');

```

```

sim.print(GPS.month, DEC); sim.print("/20");
sim.println(GPS.year, DEC);
sim.print("Fixation: "); sim.print((int)GPS.fix);
sim.print(" qualité: "); sim.println((int)GPS.fixquality);
if (GPS.fix) {
sim.print("Localisation: ");
sim.print(GPS.latitude, 4); sim.print(GPS.lat);
sim.print(", ");
sim.print(GPS.longitude, 4); sim.println(GPS.lon);
sim.print("Vitesse (km/h): "); sim.println(v);
sim.print("Angle: "); sim.println(GPS.angle);
sim.print("Altitude: "); sim.println(GPS.altitude);
sim.print("Satellites: "); sim.println((int)GPS.satellites);
delay(500);
sim.print((char)26); // (required according to the datasheet)
delay(500);
sim.println();
Serial.println("Text Sent.");
delay(500);
_buffer = _readSerial();
}
}

String _readSerial() {
_timeout = 0;
while (!sim.available() && _timeout < 12000 )
{
delay(13);
_timeout++;
}
if (sim.available()) {
return sim.readString();
}
}
}

```

## ● **Ressources:**

- [Définition objet connecté](#)
- [Datasheet LM2575-ADJ](#)
- [Adafruit MMA8451](#)
- [Adafruit ultimate GPS](#)
- [SIM800L](#)
- [Tinkercad](#)
- [Image carte Seeeduino](#)
- [Mise en panneau de plusieurs PCBs](#)
- [Datasheet Accéléromètre](#)
- [Datasheet GPS](#)
- [Datasheet seeeduino](#)
- [Datasheet régulateur](#)
- [coefficient de transfert thermique](#)
- [calcul thermique](#)
- [application thermomètre](#)
- [Datasheet GSM](#)
- [Les différentes fréquences de communications](#)
- [Gravure chimique](#)