# Conception de circuit électronique

J.-M Friedt, G. Cabodevila

7 janvier 2018

## 1 Objectif

L'objectif du projet est de concevoir une carte comprenant un microcontrôleur Atmega32U{2,4} chargé d'asservir la fréquence du quartz qui le cadence sur un signal stable fourni par un récepteur GPS.

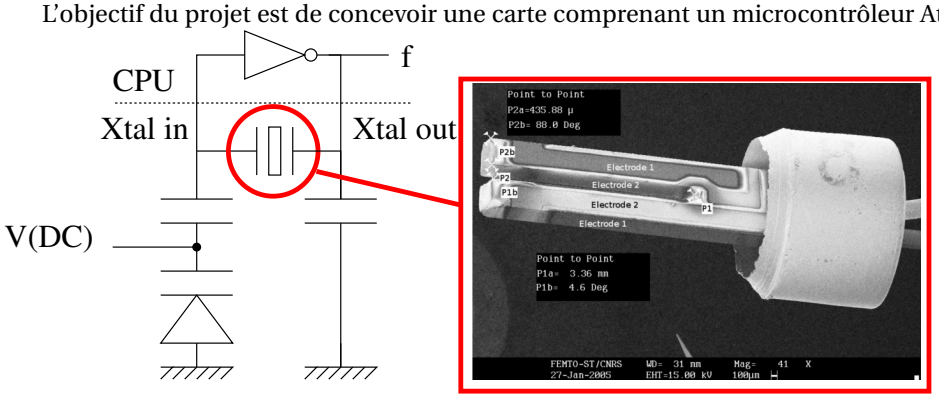



**FIGURE 1:** Principe de tirage en fréquence d'un oscillateur à quartz

Un résonateur à quartz [1] est un élément sélectif en fréquence qui, lorsque couplé à un amplificateur qui compense ses pertes, forme un oscillateur à une fréquence déterminée par la fréquence de résonance du résonateur. Cependant, le quartz présente une dérive de fréquence avec l'environnement (température, contrainte, vieillissement) qu'il est souhaitable de maîtriser, voir de corriger.

Une source stable de temps est issue du GPS, une constellation de satellites munis d'horloges atomiques transmettant un signal permettant de remonter au temps de vol de l'onde électromagnétique émise par chaque satellite et, connaissant leur position, remonter de façon précise à la source de fréquence transmise par chaque satellite. Une forme particulièrement appropriée à notre objectif est l'émission d'une impulsion chaque seconde (1 PPS – 1 pulse par seconde) avec un front précis à quelques dizaines de nanosecondes près. De nombreux récepteurs GPS proposent cette fonctionnalité.

En mesurant le nombre d'oscillations du quartz cadençant le microcontrôleur entre deux impulsions, nous serons capables de mesurer précisément la fréquence du quartz. La condition de rotation de phase pour réaliser un oscillateur (multiple de $2\pi$ dans la boucle d'oscillation) est vérifiée grâce aux condensateurs placés de part et d'autre du dipôle formé par le résonateur à quartz. En ajustant un de ces condensateurs, nous pouvons décaler la fréquence d'oscillation du circuit et ainsi asservir sa fréquence sur le 1 PPS afin de garantir une période indépendante de la température ou des conditions environnementales du microcontrôleur. Alternativement, cette stratégie permet de réaliser d'excellents capteurs grâce aux excellentes références que sont les horloges atomiques dont la stabilité est en partie transposée sur les récepteurs GPS au sol.

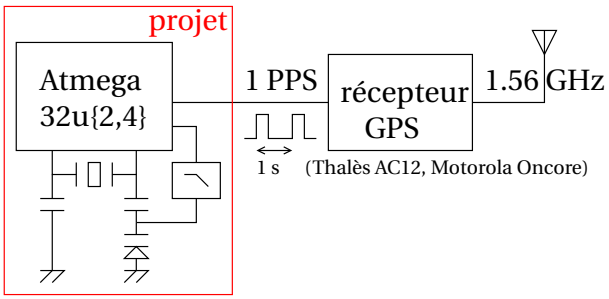



**FIGURE 2:** Mise en œuvre du projet

L'ajustement de la valeur du condensateur de tirage de l'oscillateur se fait au moyen d'un condensateur commandable en tension, ou *varicap*.

## 2 Organisation du projet

5 groupes d'étudiants travaillent en parallèle pour atteindre le même objectif.

### 2.1 Déroulement du projet

1. Première séance de présentation du sujet, identification des composants et architecture du circuit. Formation des groupes d'étudiants, qui se poursuit immédiatement par ...
2. ... deuxième séance : préparation au routage du circuit. Règles de conception d'un circuit imprimé et présentation de Kicad (`kicad-pcb.org`) – un outil de conception de circuits imprimés portable et libre. Le routage du circuit se fera par chaque groupe avant la troisième séance. Choix des composants et des boîtiers, liste des composants et estimation du coût d'approvisionnement.

3. Troisième séance d'analyse des résultats de routages, et proposition d'une solution. Chaque groupe présente le résultat de son travail. Estimation du coût de fabrication.
4. Quatrième séance sur l'utilisation de SPICE pour modéliser le comportement des parties analogiques du circuit,
5. 5 séances "intensives" de 4 h successives :
   (a) assemblage d'une carte avec composants montés en surface (CMS)
   (b) programmation du microcontrôleur pour valider le soudage et le bon fonctionnement des périphériques
   (c) mesure de la stabilité de l'oscillateur cadençant le microcontrôleur par rapport au signal de référence issu du GPS (méthode *input capture*).
   (d) traitement des signaux de stabilité (évolution temporelle et variance d'Allan), commande du quartz par ajustement de la varicap,
   (e) mise en œuvre de la boucle d'asservissement (P ou PI) pour contrôler la fréquence d'oscillation du quartz pilotant le microcontrôleur sous consigne du signal stable du GPS.

### 2.2 Notation du projet

La note est la combinaison d'une fraction (/10)

1. schéma du circuit électronique,
2. routage du circuit électronique,
3. liste des composants et estimation du coût de fabrication (PCB + composants),
4. capacité à programmer des exemples simples pour qualifier le bon fonctionnement, mise en œuvre de l'outil de programmation du microcontrôleur avec un bootloader différent de celui vu en TP,
5. qualité de l'assemblage de la carte, soudure des composants,
6. mesure par Input Capture de la fréquence de l'oscillateur du microcontrôleur,
7. modélisation SPICE,
8. outils de traitement numérique du signal : affichage des signaux (gnuplot, GNU/Octave), modélisation SPICE,
9. conception du filtre passe bas permettant de transformer une sortie PWM en convertisseur numérique analogique,
10. mise en œvre de la boucle d'asservissement,
11. assiduité et participation au projet

liée aux succès de réalisation, une fraction (/5) issue d'une présentation par un des membres du groupe choisi au hasard par tirage au sort. Ces 15 points sont communs au groupe. 5 points sont réservés aux notes individuelles sur la participation pendant les séances et laissé à la discrétion des encadrants.

## 3 Analyse préliminaire

Le dessin du schéma (circuit logique des connexions entre composants) et du circuit (routage des pistes) n'est que le résultat d'une réflexion préliminaire sur les objectifs à atteindre par le circuit et les éléments à placer.

Connaissant le cahier des charges proposé dans la première partie de ce document, nous devons mettre en œuvre un mircocontrôleur ainsi que les composants annexes

**Questions :**

1. **Identifier les broches du microcontrôleur qu'il faut absolument connecter pour le faire fonctionner, en plus des alimentations et du résonateur à quartz.**
2. **Quels composants additionnels, en plus du microcontrôleur, sommes nous susceptibles d'utiliser ?**
3. **Comment gérer la tension d'alimentation du circuit – valeur moyenne et stabilité ?**
4. **Comment générer la tension de commande qui ajustera le condensateur de tirage de l'oscillateur ?**
5. **Quels périphériques du microcontrôleur seront nécessaires aux tâches de gestion du temps qui vont nous intéresser ?**

La conception d'un circuit sous Kicad s'effectue en trois étapes : la sélection des composants et la définition de leurs connexions logiques, sans tenir compte du placement (schéma électrique). Une fois les divers composants sélectionnés et connectés entre eux, il faut choisir le boîtier utilisé dans l'implémentation physique de chaque composant : un même composant peux venir en divers boîtiers, et un choix judicieux par rapport à l'approvisionnement évitera des déboirs lors de l'assemblage. Finalement une fois le boîtier associé à chaque composant sélectionné, le *routage* du circuit place les composants sur un circuit imprimé et les relie entre eux par des pistes. On pourra, si nécessaire, choisir plusieurs couches de cuivre

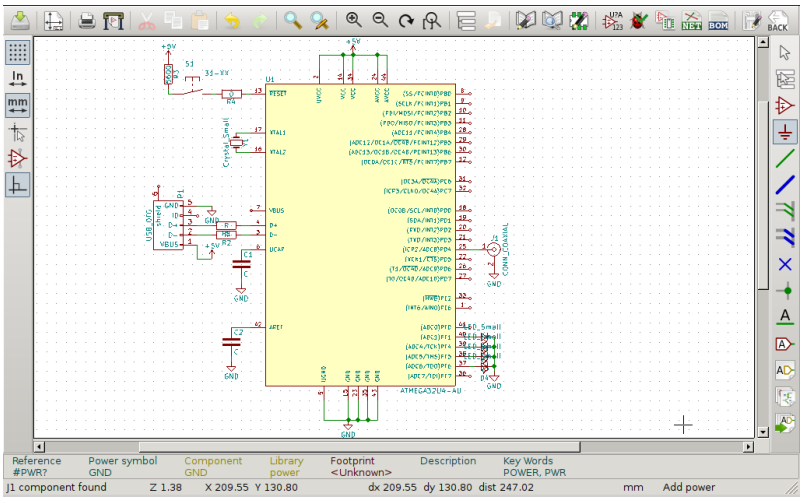
pour router des signaux sur la face comportant les composants ou sur la face opposée. Il est classiquement considéré de bon ton de conserver tous le composants sur une même couche (supérieure) pour faciliter l'assemblage de la carte, la seconde cuisson au reflux se traduisant généralement par la déssoudure de certains composants de la première face et une difficulté à étaler la pâte à braser sur un circuit imprimé dont une face n'est plus plane.

# 4 Schéma

Kicad se lance en ligne de commande par `kicad` qui propose un nouveau projet vierge. Nous allons commencer par tracer le schéma en cliquant sur la première icône `Eeschema`. La principale interaction se fera avec l'icône en forme d'amplificateur opérationnel dans la barre de droite : cette icône sert à ajouter un composant.

Une fois un composant sélectionné (tapper les premières lettres de sa référence dans la barre de recherche), il peut être sélectionné pour être déplacé par `g` (*grab*), tourné par `r`, ou effacé par `DEL`. Toutes les interactions avec les composants se font en appuyant sur une touche du clavier alors que le curseur de la souris se situe au dessus du composant affecté par l'action.

Si un composant n'est pas disponible dans la liste, nous tentons d'ajouter la bibliothèque qui le contient : c'est par exemple le cas du FT230XS, qui nécessite d'ajouter la bibliothèque FTDI. Pour ce faire, `Preferences` → `Component Libraries` et ajouter la bibliothèque adéquate (par exemple `ftdi.lib` pour le composant qui nous intéresse ici). Pour le moment, l'empreinte (boîtier) du composant n'importe pas, nous ne considérons que la relation logique entre les diverses broches des divers composants. À titre d'exemple, le connecteur coaxial qui alimentera le microcontrôleur avec le 1 PPS du GPS n'est pas spécifié à cette étape : coaxial signifie simplement une ame et une tresse, sans se poser la question des dimensions du connecteurs (BNC, SMA, SMB ...) qui seront spécifiées à la prochaine étape (section 5).




Début du schéma en peuplant les composants passifs autour du microcontrôleur.

Les masses et alimentation bénéficient de symboles spécifiques dans la bibiothèque `Power`, aussi accessible par le symbole de masse dans les icônes à droite de la fenêtre.

Penser à définir dès cette étape les valeurs de composants (touche `v` en plaçant le curseur de la souris au dessus du composant à modifier), qui évitera de retrouver le sens de chaque valeur lors de l'assemblage de la carte. Il peut être judicieux de commencer par le composant le plus encombrant – le microcontrôleur Atmega32U4 – puis de peupler les composants passifs qui l'entourent. Continuer en plaçant les alimentations et masse : GND et Vcc. Avec le condensateur de découplage UCAP de 1 $\mu$F (ajouter un composant par le symbole d'amplificateur opérationnel et "C" pour ajouter un condensateur) nous avons le minimum vital pour alimenter le microcontrôleur.

Ajouter ensuite le connecteur USB-mini B : pour ce faire, rechercher dans la bibliothèque `conn` le connecteur `USB_OTG (mini/micro B)`. On notera qu'à ce stage de la conception seule la nature des broches (connecteur USB 5 points) nous intéresse, le boîtier sera défini plus tard.

**Rechercher dans la norme USB quelle broche correspond à quel signal (alimentation, D+ et D-, masse).**

Ajouter le résonateur à quartz qui cadencera le micrcontrôleur, ainsi que le réseau de condensateurs qui permettront de vérifier les conditions de Barkhause pour osciller. Ici encore, le boîtier contenant le résonateur ne sera défini que plus tard, nous avons juste à savoir qu'un résonateur est un dipôle.

Ajouter les divers éléments nécessaires à la finalité du projet : dans notre cas, une varicap sur le quartz, un filtre passe bas sur la sortie timer et une entrée pour mesurer le signal de référence GPS sur une entrée *timer*.

1. **Effectuer la liste des composants en fournissant leurs caractéristiques (valeur, nature du composant ou technologie utilisée), et leur code commande chez un des fournisseurs (Mouser, RS, Farnell, Digikey ...). Il s'agit de la BOM**

**– *Bill of Materials* – un élément clé de la conception pour commander les références nécessaires. Kicad nous aide en générant une partie de la liste à partir du schéma : icône `BOM` et appeler un des greffons, par exemple `bom2cvs` pour fournir une entrée par composant, fournis dans `/usr/lib/kicad/plugins`. Le fichier ainsi généré devra être complété des références fournisseur et coût unitaire.**

2. **Budgéter le coût d'approvisionnement des composants en fonction des volumes.**

⚠ Les **condensateurs de découplage** sont un élément clé du bon fonctionnement d'un circuit numérique induisant des appels de courant significatifs à chaque transition d'état. Nous plaçons généralement deux condensateurs en parallèle, un condensateur de faible valeur mais restant capacitif aux radiofréquences, et un condensateur de forte valeur qui risque de devenir inductif aux hautes fréquences (Fig. 3). Bien que regroupés généralement ensemble sur le schéma, chaque paire de condensateur de découplage doit être placé au plus près de chaque point d'alimentation de chaque composant numérique.

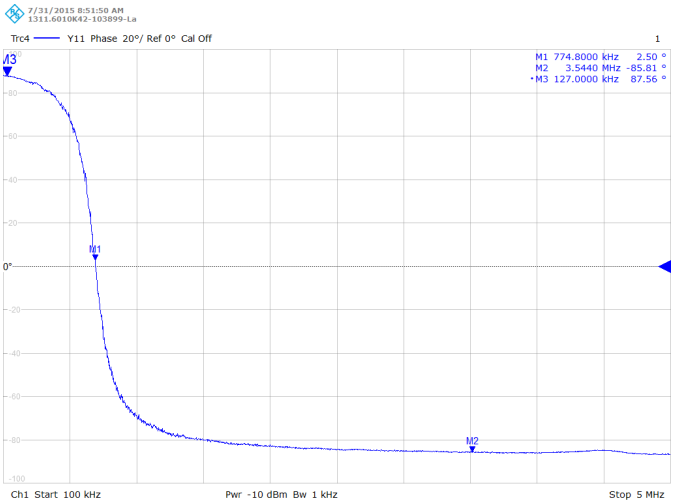
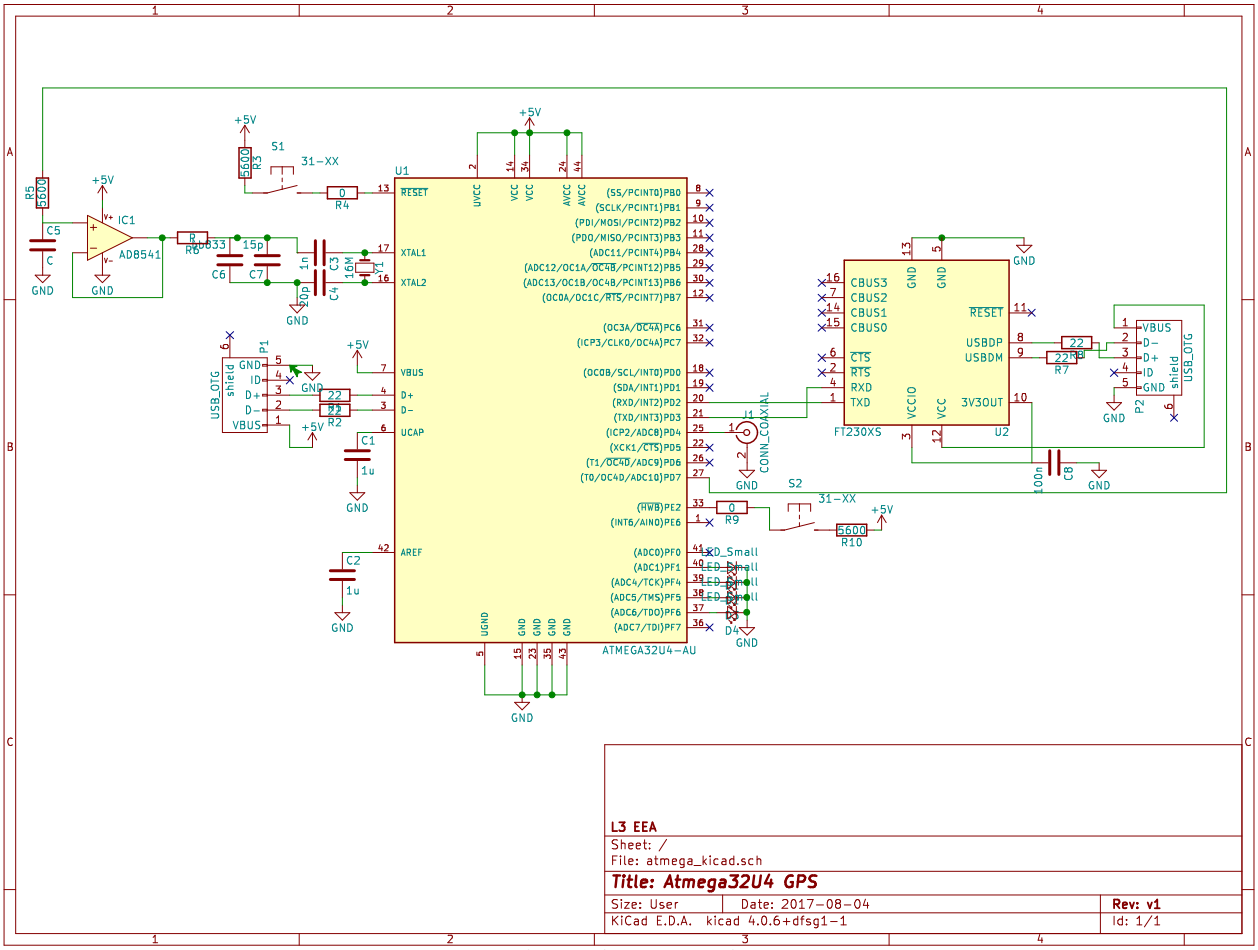



FIGURE 3 – Phase de l'admittance d'un condensateur haute tension, en fonction de la fréquence.

1. **Comment calculer la valeur d'un condensateur de découplage ? Dans quelle gamme de fréquence agit la forte valeur ? la faible valeur ?**



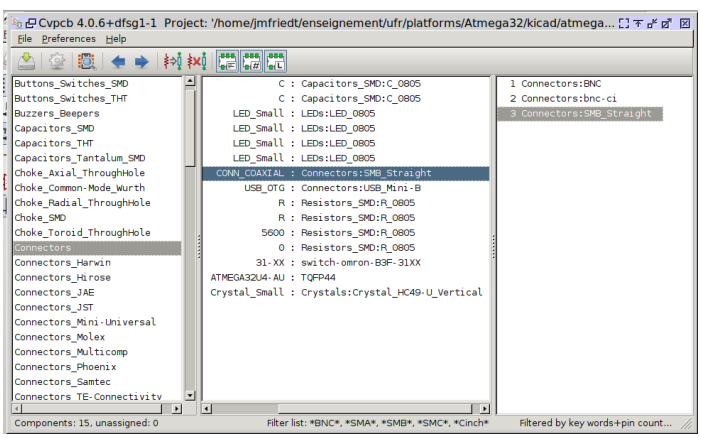

Le schéma du circuit achevé.

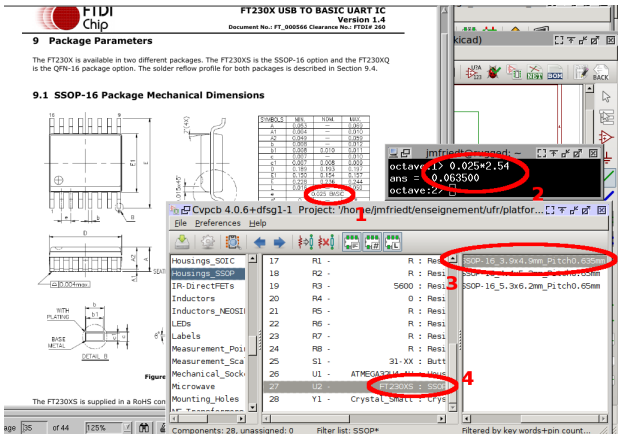
# 5 Sélection du boîtier

Une originalité de Kicad est de ne pas assigner les boîtiers au moment du dessin du schéma mais de proposer cette étape explicitement dans le flux de traitement : il s'agit de l'outil CvPCB, quatrième icône en partant du coin en haut à droite. Nombreux sont les composants qui n'ont qu'un boîtier possible : dans ce cas pas de doute, et Kicad fait le choix automatique. Deux cas plus problématiques se posent :

1. divers boîtiers existent pour une même référence. Dans ce cas nous choisissons, dans la liste des bibliothèques listées dans la colonne de gauche, celle qui contient le bon boîtier, que nous assignons au composant en cours d'analyse. Par exemple les condensateurs et résonateurs seront sélectionnées en composant monté en surface (CMS – *SMD* en anglais) en format 805, bon compomis entre manipulation et soudure à la main.
2. plus problématique, un boîtier n'existe pas dans la liste des bibliothèques chargées par CvPCB. Dans le meilleur des cas nous chargeons la bibliothèque manquante `Preferences → Footprint libraries`. Dans le pire des cas, nous devons dessiner l'empreinte nous mêmes. Ce dernier cas n'a pas été rencontré pour ce projet.

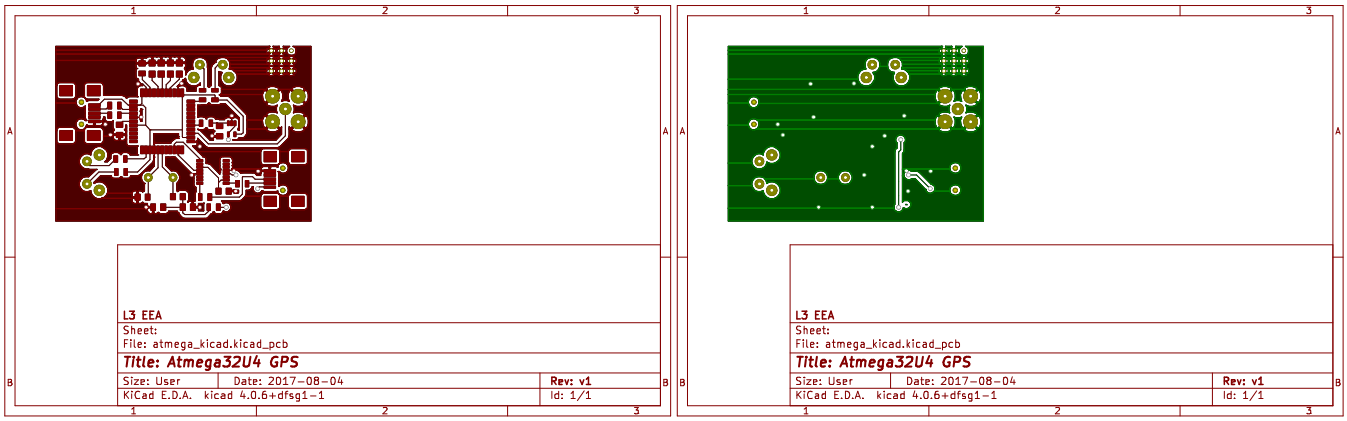
À titre d'exemple du FT230XS ci dessous (droite), plusieurs boîtiers existent même après avoir pris soin de charger la bibliothèque des empreintes FTDI. Une recherche dans la documentation technique permet d'identifier(1) l'espacement entre les broches du composant (après conversion de mm en inch (2)). Un seul boîtier (3) propose un tel espacement, que nous assignons (4) à notre composant.




# 6 Routage du circuit

Une fois les boîtiers assignés à chaque composant, nous sauvons depuis `CvPCB` le résultat de l'association, puis revenons au schéma pour sauver la netlist par `Generate Netlist`. Ce dernier fichier fera le lien avec l'outil de routage. Il est judicieux de valider l'intégrité du schéma à cette étape par l'Electrical Rule Check (ERC – l'icône en forme de coccinelle) qui devrait se conclure, après correction des erreurs, par l'absence de commentaire de cet outil de vérification. En conclusion de toutes ces étapes, nous pouvons enfin engager le routage, en cliquant dans le schéma sur l'icône la plus à droite, `Pcbnew`.




Nous pourrons itérer autant de fois que nécessaire cette séquence schéma-boîtier- routage : les composants déjà routés restent en place, et les ajouts au schéma apparaissent avec leur chevelu dans le coin en haut à gauche de l'espace de routage.

Il est impératif de se remémorer la fonction de chaque composant lors de son placement sur le circuit. Le chevelu ne permet pas de convenablement s'orienter lors de cette étape. Toutes les pistes commencent depuis une broche de composant, et se terminent à une autre broche en appuyant sur la touche END.

Tout circuit contient un plan de masse : **il est inutile de router explicitement les masses**. En l'absence de plan de masse, on évitera absolument les **boucles de masses** qui se comportent comme des antennes magnétiques et induisent des fluctuations radiofréquences de la référence que devrait être la masse.

Noter par ailleurs la présence des **freins thermiques** autour des connecteurs et broches (Fig. 4) lors du routage automatique du plan de masse : ces freins facilitent grandement la soudure de l'embase en évitant que le plan de masse ne se comporte comme un radiateur trop efficace qui dissiperait trop de chaleur et induirait des soudures sèches.

Le plan de masse se forme par `Add filled zones` (sixième icône de la colonne de droite) et en dessinant un rectangle (penser à réduire la résolution de la grille) couvrant la surface finale du circuit imprimé. On se facilitera la vie en plaçant un second plan de masse en couche inférieure. Bien que cette couche ne doive contenir aucun composant, elle permet de proprement relier les masses des composants éventuellement séparés les uns des autres par des pistes. Un via vers le plan de masse inférieur s'obtient par le tracé d'une piste depuis la couche supérieure, puis `v` pour passer en couche inférieure, et END pour achever le dessin en milieu de plan de masse inférieur. Le plan de masse est rempli de cuivre en cliquant sur l'icône en forme de coccinelle ou Design Rule Check (DRC), qui donne aussi la liste des broches qui ne sont pas encore routées. Afin d'éviter une multitude d'erreurs inutiles, on prendra soin d'informer Kicad des broches non-connectées en leur affectant une croix (9ème icône dans la colonne de droite) qui évite les messages d'erreurs concernant les broches flottantes.

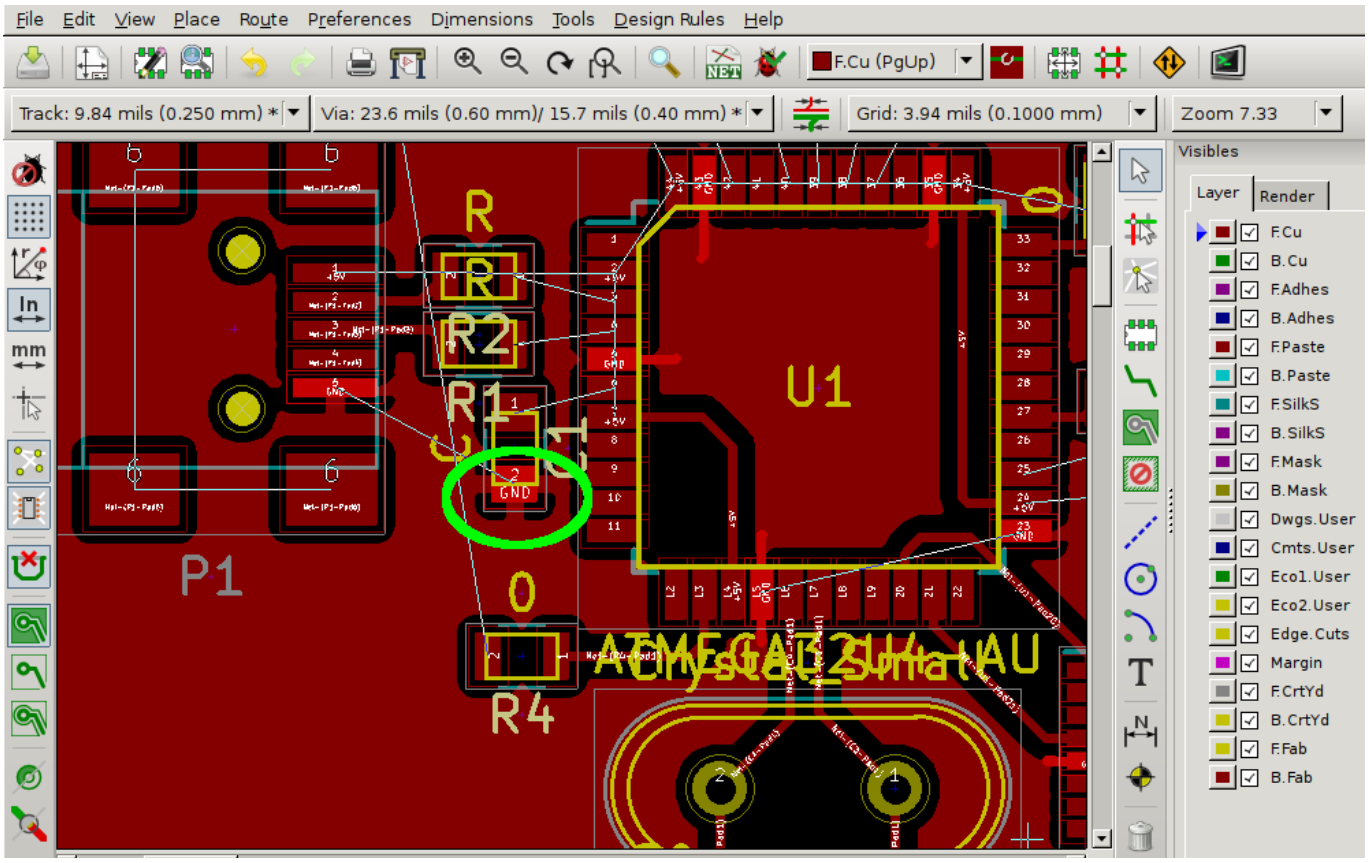



FIGURE 4 – Inutile de router les signaux de masse : cette partie est prise en charge par le plan de masse. Noter par ailleurs les freins thermiques (rond vert) qui aident à étaler l'étain lors de la soudure en évitant une diffusion trop rapide de la chaleur amenée par le fer à souder.

On notera que Kicad ne sait pas créer de vias flottants que nous imposerions à la masse pour cribler une zone qui doit être convenable mise à la masse, comme c'est souvent le cas en radiofréquence. Une solution est proposée à `https://forum.kicad.info/t/protip-nicer-via-stitching/1103/3` que nous avons mise en œuvre, à titre de démonstration, dans le coin en haut à droite du *board*. Les freins thermiques sont bien visibles, et le via du haut à droite est volontairement déconnecté du plan de masse pour illustrer la différence de comportement.

1. **Estimer, au moyen du calculateur du site `www.eurocircuits.com`, le coût de fabrication du circuit imprimé.**
2. **Observer l'évolution du coût avec le nombre de couches. Avec la surface. Avec la précision de routage (classe du circuit).**
3. **Budgéter le coût de fabrication du circuit en fonction des volumes.**

| Part | Value | Package | Farnell | Cost |
|---|---|---|---|---|
| C1 | 1u | C0805 | 2470435 | 0.118 |
| C2 | 100n | C0805 | 2470437 | 0.0665 |
| C3 | 20p | C0805 | 1885436 | 0.189 |
| C4 | bb833 | C0805 | 2432690 | 0.23 |
| C5 | 1n | C0805 | 7568665 | 0.375 |
| C6 | | C0805 | | |
| C7 | 100n | C0805 | 2470437 | 0.0665 |
| C8 | 100n | C0805 | 2470437 | 0.0665 |
| C9 | 100n | C0805 | 2470437 | 0.0665 |
| C10 | 100n | C0805 | 2470437 | 0.0665 |
| C11 | 100n | C0805 | 2470437 | 0.0665 |
| C12 | 15p | C0805 | 1885434 | 0.175 |
| IC2 | FT230XS | SSOP16 | 2081321 | 1.94 |
| IC3 | AD8541RJ | SOT23-5 | 1333254 | 0.787 |
| LED1 | | LED_SML0603 | 2112121 | 0.166 |
| LED2 | | LED_SML0603 | 2099221 | 0.102 |
| LED3 | | LED_SML0603 | 2099227 | 0.0996 |
| LED4 | | LED_SML0603 | 2099222 | 0.116 |
| Q1 | 16M | SM49 | 1640900 | 2.79 |
| R1 | 22 | R0805 | 9334122 | 0.0371 |
| R2 | 22 | R0805 | 9334122 | 0.0371 |
| R3 | 5k6 | R0805 | 1750764RL | 0.00624 |
| R4 | 5k6 | R0805 | 1750764RL | 0.00624 |
| R5 | | R0805 | | |
| R6 | 22 | R0805 | 9334122 | 0.0371 |
| R7 | 22 | R0805 | 9334122 | 0.0371 |
| R8 | 0 | R1206 | 9240276 | 0.024 |
| R9 | 0 | R1206 | 9240276 | 0.024 |
| R10 | 5k6 | R0805 | 1750764RL | 0.00624 |
| R11 | 5k6 | R0805 | 1750764RL | 0.00624 |
| R12 | 0 | R1206 | 9240276 | 0.024 |
| S1 | | B3F-31XX | 2468742 | 0.176 ??? |
| S2 | | B3F-31XX | 2468742 | 0.176 |
| U1 | ATMEGA32U4-AU | | 1748525 | 4.75 |
| X1 | MINI-USB-32005-201 | | 2112367 | 0.541 |
| X2 | BU-SMA-V | | 1248990 | 1.61 |
| X3 | MINI-USB-32005-201 | | 2112367 | 0.541 |

# 7 Soudure des composants

Les composants montés en surface (CMS) se manipulent avec une pince sous binoculaire. Les circuits intégrés se soudent normalement par pâte à braser fondue dans un four après positionnement du composant sur les empreintes préalablement enduites au pochoir. En l'absence de cet équipement, nous profiterons de la capillarité qui permet de concentrer l'étain en fusion entre les pattes du composant et le piste et le retirer des ponts formés entre les pattes du composant. Pour ce faire, placer de l'étain en fusion sur toutes les pattes du circuit intégré sans se soucier des ponts. Une fois toutes les pattes couvertes d'étain, éliminer l'excédent à la tresse à déssouder. Vérifier visuellement l'absence de pont induisant un court circuit (Fig. 5, en bas à droite).

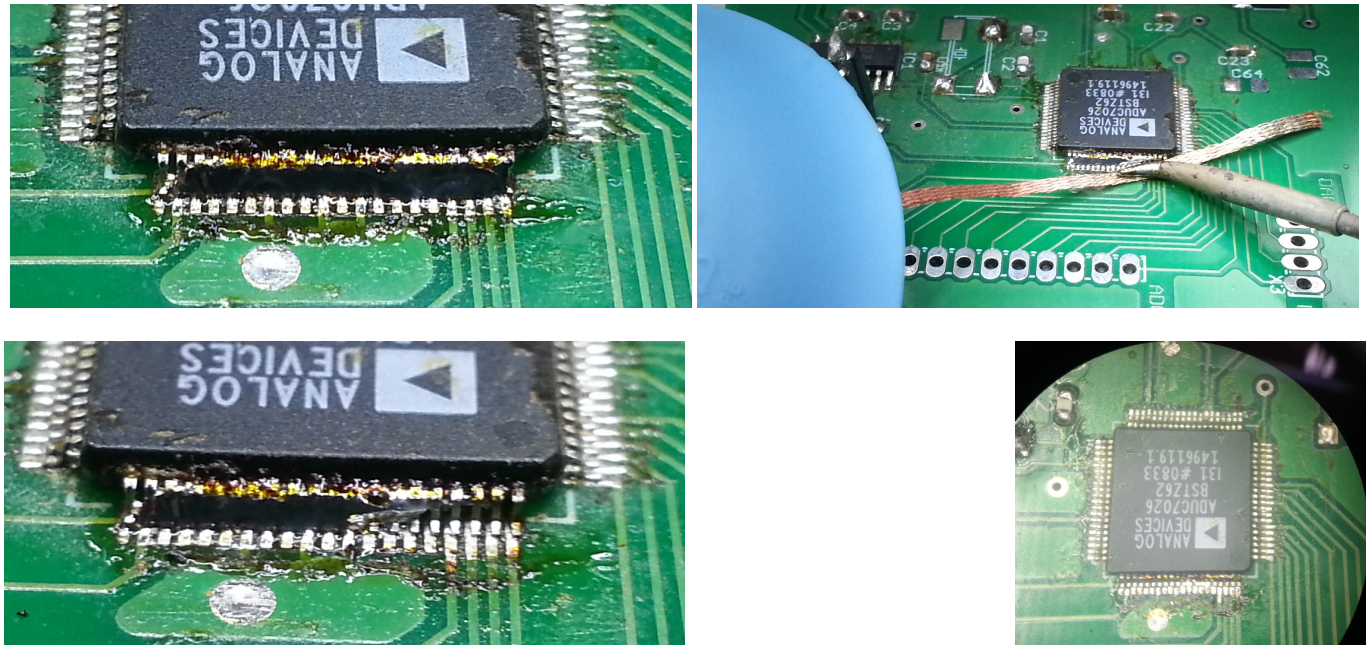


FIGURE 5 – Séquence de soudure d'un circuit CMS : en haut à gauche un volume important d'étain est appliqué sur toutes les pattes – avec vérification visuelle du bon mouillage de la soudure sur les pads de contact du substrat. En haut à droite, l'excès d'étain est retiré par la tresse à dessouder, sans pousser sur les pattes (afin de ne pas les tordre) mais en laissant la soudure imprégner la tresse chauffée au point de fusion de l'étain. En bas à gauche, une partie des pattes a été libéré de l'excès d'étain : la procédure est répétée sur toute la largeur du composant. En bas à droite : vérification visuelle sous binoculaire de l'absence de point d'étain entre les pattes.

Pour les composants discrets, le principal danger tient aux soudures sèches lorsque l'étain ne mouille pas les deux surfaces à connecter. Mouiller convenablement un métal n'est possible que si celui ci n'est pas oxydé et porté à la température de fusion de l'étain : il est inutile de placer un grosse boule d'étain sur le composant si la piste sous jacente n'a pas atteint la température de fusion de la soudure.

Noter les **freins thermiques** sur les connexions de masse du connecteur SMA qui lui évite de se comporter comme radiateur et d'empêcher la soudure de correctement le mouiller. L'objectif du frein thermique est de *faciliter l'échauffement du substrat* : si le substrat n'est pas à la température de fusion de l'étain, un boule de soudure se forme autour du fil (de faible capacité calorifique) sans mouiller le substrat. Il s'agit d'une **soudure sèche**, garantie de contact intermittent et de dysfonctionnement à plus ou moins court terme du circuit, en particulier sur les composants les plus lourds tels que les transformateurs (Fig. 6)

Le comportement d'un résonateur à onde de volume n'est pas simple à intuiter : alors que la branche motionnelle $R_1$, $L_1$, $C_1$ est un circuit RLC classique – caractérisé en particulier par sa fréquence de résonance (maximum d'admittance ou phase nulle avec impédance égale à $R_1$) et son facteur de qualité – la présence de la capacité parallèle $C_0$ induite par les deux électrodes déposées sur les deux faces du quartz (diélectrique) induit un nouveau concept qu'est l'anti-résonance définie comme un maximum d'impédance. Cet aspect est observé expérimentalement sur la Fig. 7.

Afin de mieux appréhender le comportement de ce composant et son interaction avec la varicap dans le circuit oscillant, nous allons nous intéresser à la modélisation numérique de sa fonction de transfert.

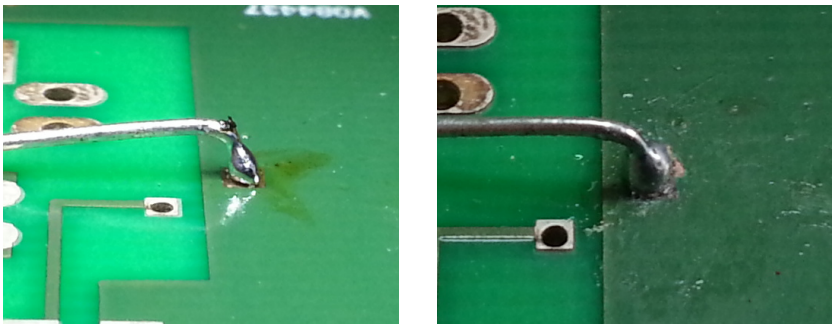


FIGURE 6 – Exemple de soudure sèche, à gauche (la boule d'étain ne touche pas le substrat), et une soudure qui mouille correctement le contact à droite.

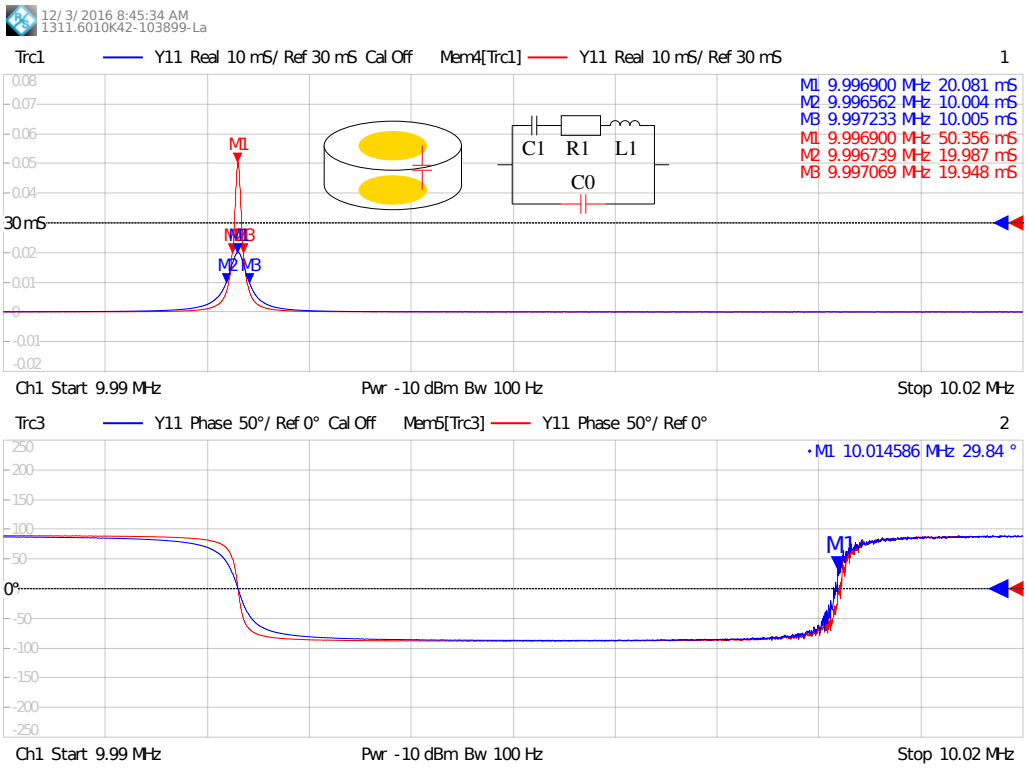



FIGURE 7 – Caractérisation d'un résonateur à onde de volume résonance à 10 MHz sur une bande de fréquences suffisamment large pour faire apparaître l'anti-résonance. L'écart de fréquence entre résonance et anti-résonance est donné par le carré du coefficient de couplage électromécanique $k^2$ [3]. En bleu le résonateur seul, en rouge l'ajout en série d'une résistance de 30 Ω pour amener le minimum d'impédance à 50 Ω réduit le facteur de qualité.

## 8 Modélisation du filtre passe-bas

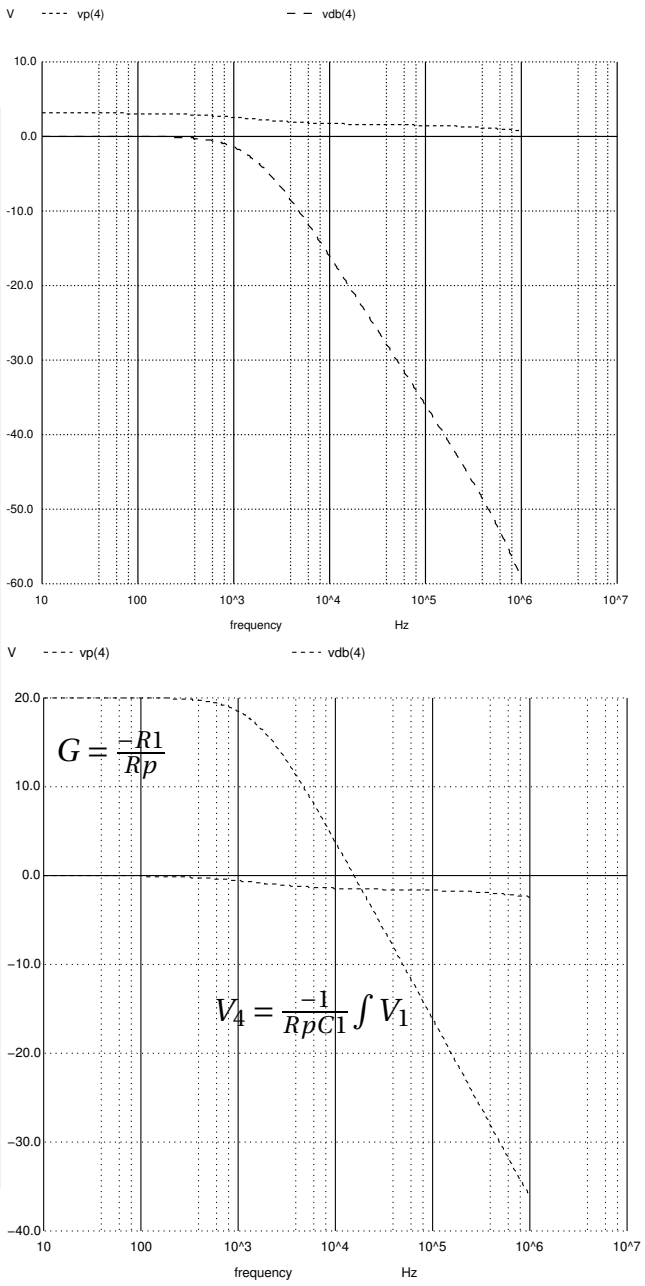
La conversion de la PWM vers un signal continu de moyenne variant avec le rapport cyclique est un point clé de l'asservissement proposé. Le circuit RC passe-bas est bien naïf, mais donne l'opportunité de tester une modélisation du comportement d'un circuit actif sous SPICE. SPICE est un simulateur de circuit électronique numérique ou analogique – et par extension des analogies thermiques et mécaniques avec les circuits électriques, de tout système multiphysique. L'implémentation libre originale de SPICE a significativement évolué pour devenir aujourd'hui `ngspice-ng`, disponible à `git://ngspice.git.sourceforge.net/gitroot/ngspice/ngspice`.

Un exemple relativement trivial de filtre passe-bas est fourni par le script SPICE ci-dessous. Pour exécuter la simulation, lancer la commande `ngspice` et charger le script par `source "script.cir"`. Tout script de description de circuit SPICE commence par un commentaire (symbole `*`) nommant le programme. La liste des composants et leurs points de connexion dans le schéma est ensuite fournie : chaque composant est représenté par la première lettre qui indique sa nature (`R`, `L` ou `C` pour les composants passifs, `V` ou `I` pour les sources de tension et de courant respectivement, `X` pour un composant défini

par l'utilisateur ...) suivi d'un identifiant. Les nœuds de connexion sont décrits par un nombre, et finalement la valeur du composant est indiquée. **Attention** : micro s'indique par un `u`, `m` indique le milli et le méga s'écrit `meg`. Dans l'exemple que nous proposons, un amplificateur opérationnel est modélisé comme une source de tension commandée en tension, sans seuil lorsque les tensions d'alimentation sont atteintes. Une fois le circuit décrit, nous lui ajoutons des sources de signaux, décrits par les propriétés nécessaires pour les divers types de simulation. Pour une simulation de la fonction de transfert spectrale, nous devons connaître l'amplitude du signal alternatif et son offset (`AC` et `DC`). Pour une simulation temporelle (nommée chez SPICE transitoire), nous décrivons la forme du signal – ici une sinusoïde d'amplitude 1 V et d'offset 1,5 V, de fréquence 1 kHz.

Le circuit ainsi décrit se charge dans `ngpice` par l'instruction `source` prenant comme argument le nom du script de configuration. Les diverses commandes décrivant les simulations peuvent soit être fournies par l'utilisateur, soit être incluses dans le script de configuration après le mot clé `.control`. Dans cet exemple, nous sauvons les graphiques en noir et blanc au format postscript pour une exploitation ultérieure. Les diverses subtilités d'utilisation de SPICE ont été abordées dans [2].

```
1  * RC et suiveur puis integrateur
2
3  .subckt opamp 1 2 3
4  * +in (=1) -in (=2) out (=3)
5  rin 1 2 2meg
6  e 4 0 1 2 100k
7  rbw 4 5 0.5meg
8  cbw 5 0 31.85nf
9  eout 6 0 5 0 1
10 rout 6 3 75
11 .ends opamp
12
13 Ve 1 0 AC 1 DC 0 sin(1.5 1 1k)
14 * RC suivi d'un suiveur
15 Rp 1 2 10k
16 C1 2 0 10n
17 X1 2 3 4 opamp
18 Rr 3  4 0
19
20 * C sur retroaction
21 * Rp 1 2 10k
22 * C1 2 4 1n
23 * R1 2 4 100k
24 * X1 0 2 4 opamp
25 * Rl 4 0 1k
26
27 .control
28 ac dec 101 10 1meg
29 plot vdb(4), vp(4)
30 * tran 100n 5m
31 * plot v(4) v(1)
32 set hcopydevtype=postscript
33 set color0=rgb:f/f/f
34 set color1=rgb:0/0/0
35 * hardcopy lpf.eps vdb(4) vp(4)
36 hardcopy lpf.eps v(4) v(1)
37 .endc
```




Les figures ci-dessus présentent la phase et magnitude issus de la simulation d'un filtre passe bas par `ngspice`. L'amplificateur opérationnel `opamp` est supposé idéal, formé d'une source de tension commandée en tension, de bande passante finie et avec une résistance de sortie finie. En haut un circuit RC suivi de l'amplificateur opérationnel monté en suiveur, en bas l'amplificateur opérationnel monté en intégrateur.

Un script SPICE comprend une description de la netlist sous forme de composants (résistance `R`, inductance `L`, condensateur `C`, circuit spécialisé `X`) reliés entre eux par les nœuds qui décrivent leurs extrémités. Un symbole "`*`" indique un com-

mentaire, les instructions commençant par un point sont pour le simulateur et ne décrivent pas le circuit mais son mode de simulation.

**Pourquoi est-il peu judicieux d'utiliser le montage avec le condensateur sur la rétroaction de l'amplificateur opérationnel?**

Le circuit intégrateur présente l'intérêt de pouvoir fournir un gain (ration de la résistance de rétroaction à la résistance d'entrée en deça de la fréquence de coupure du filtre), mais ce gain est négatif, signifiant qu'une tension positive en entrée se traduit par une tension négative en sortie. Or le microcontrôleur ne peut délivrer qu'une tension positive, et la varicap doit être polarisée par une tension elle aussi positive. Par ailleurs, l'amplificateur opérationnel est polarisé en mode unipolaire, il ne peut donc gérer que des entrées et sortiees positives. Pour ces raisons, nous favoriserons le circuit présentant un filtrage passif (circuit RC) suivi d'un amplificateur opérationnel comme convertisseur d'impédance (montage suiveur).

Les amplificateurs opérationnels se déclinent en trois versions : un amplificateur opérationnel classique ne peut atteindre que les bornes $-V_{cc}+0,7\,\text{V}$ à $+V_{cc}-0,7\,\text{V}$; un amplificateur opérationnel *single supply* peut exploiter des tensions descendant jusqu'à $-V_{cc}$; et un amplificateur opérationnel *rail to rail* qui atteint les deux bornes d'alimentation. Dans notre cas, où l'alimentation est unipolaire sur le port USB, il faut au moins sélectionner un amplificateur *single-supply* pour qu'une tension de commande nulle (à la masse) soit correctement traitée par le circuit de mise en forme.

Ces concepts se démontrent en observant l'évolution temporelle des signaux sous `ngspice`, par une simulation du transitoire (méthode `tran`) tel que illustré Fig. 8.

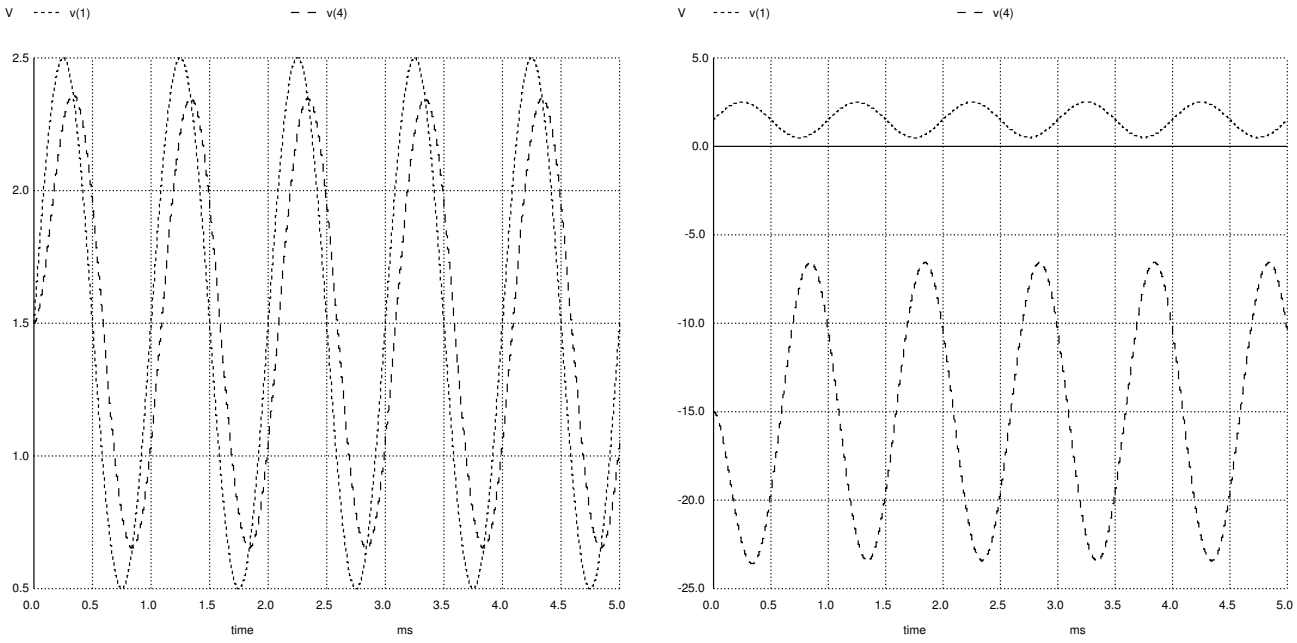



FIGURE 8 – Évolution temporelle (simulation du transitoire sous `ngspice`) des signaux d'entrée V(1) et de sortie V(4) du filtre passe-bas. À gauche le circuit RC suivi d'un amplificateur opérationnel en suiveur : le gain est unitaire et positif. À droite l'amplificateur opérationnel est monté en intégrateur : le gain est négatif et égal, sous la fréquence de coupure, au ratio de la résistance de rétroaction à la résistance d'entrée.

SPICE permet aussi de modéliser le comportement de l'oscillateur à quartz, une fois établies les règles d'oscillations de Barkhausen qui affirment que le gain dans la boucle de rétroaction doit compenser les pertes dans le résonateur, et que la rotation de phase dans l'ensemble de la boucle doit être multiple de $2\pi$. Le résonateur est modélisé comme un dispositif électromécanique dans lequel la résonance mécanique (masse-ressort-piston) est modélisée comme un circuit *RLC* série (branche dite motionnelle), en parallèle de laquelle un condensateur fixe représente le diélectrique entouré d'électrodes. Ce condensateur additionnel a une conséquence importante sur le comportement du dispositif sous forme d'une anti-résonance : en plus d'une condition de résonance qui maximise l'admittance (minimise l'impédance), une seconde condition de confinement de l'énergie mais cette fois dans la branche capacitive maximise l'impédance (ou minimise l'admittance). L'écart de fréquence entre résonance et anti-résonance est donné par le coefficient de couplage électro-mécanique du substrat piézoélectrique dont le résonateur est formé, et définit par ailleurs l'excursion maximale de tirage de la fréquence d'oscillation par variation des condensateurs de pieds [3].

## 8.1 Modélisation de la varicap (varactor)

La modélisation d'une varicap n'est pas triviale. Bien qu'une diode contienne, dans son modèle SPICE, un condensateur, ce dernier n'est pas ajustable par une tension : la varicap doit être commandable par une tension fixe. L'article [4] propose une solution sous la forme d'une mise en équation du comportement d'une varicap et l'inclusion de ce comportement dans un sous-circuit exploitable sous SPICE. Pour résumer rapidement sa démonstration : la tension $V_C$ aux bornes d'un condensateur $C$ est l'intégrale du courant $i$ qui y circule, soit $V_C(t) = \frac{1}{C}\int i(t)\cdot dt \Leftrightarrow C = \frac{1}{V_C}\int i(t)\cdot dt$. Le calcul de l'intégrale est simulé dans SPICE par la tension aux bornes d'un condensateur virtuel de 1 F. Nous allons nous convaincre de la pertinence de cette approche en incluant la varicap dans un circuit $RC$ passe-bas et en observant la fonction de transfert en fonction du potentiel DC appliqué sur la varicap. La Fig. 9 résume les résultats obtenus.

```
* varactor demonstration
* (RC circuit with C: varactor)

* V=1/C \int i(t)dt
* => C=1/V \int i(t)dt
.SUBCKT VARICAP 1 2
R1 1 3 1
VC 3 4
BC 4 2 V=(1/(v(1)*10e-9))*v(int)
BINT 0 INT I=I(VC)
CINT INT 0 1
.ENDS

Ve      1       0       AC      1 DC    2 sin(1.5 1 1k)
R1      1       2       1k
C1      2       0       20n
X1      2       0       VARICAP
Coff    2       4       1u
Ro      4       0       100

.control
```

```
destroy all
set units = degrees
alter @ve[dc]=0
ac lin 10001 2k 100k
alter @ve[dc]=1
ac lin 10001 2k 100k
alter @ve[dc]=2
ac lin 10001 2k 100k
alter @ve[dc]=3
ac lin 10001 2k 100k

set hcopydevtype=postscript
set hcopypscolor=1
set color0=rgb:0/0/0
set color1=rgb:f/f/f
plot db(ac1.v(4)) db(ac2.v(4)) db(ac3.v(4)) db(ac4.v(4)) xlabel f[
plot ph(ac1.v(4)) ph(ac2.v(4)) ph(ac3.v(4)) ph(ac4.v(4)) xlabel f[
hardcopy varicap_abs.ps abs(ac1.v(4)) abs(ac2.v(4)) abs(ac3.v(4))
hardcopy varicap_pha.ps ph(ac2.v(4)) ph(ac3.v(4)) ph(ac4.v(4)) xla
unset units
.endc
```

Les lignes coupées avec les affichages sont sans grand intérêt, et de la forme (complète)
`plot db(ac1.v(4)) db(ac2.v(4)) db(ac3.v(4)) db(ac4.v(4)) xlabel f[Hz] ylabel |mag(V)|[dB]`

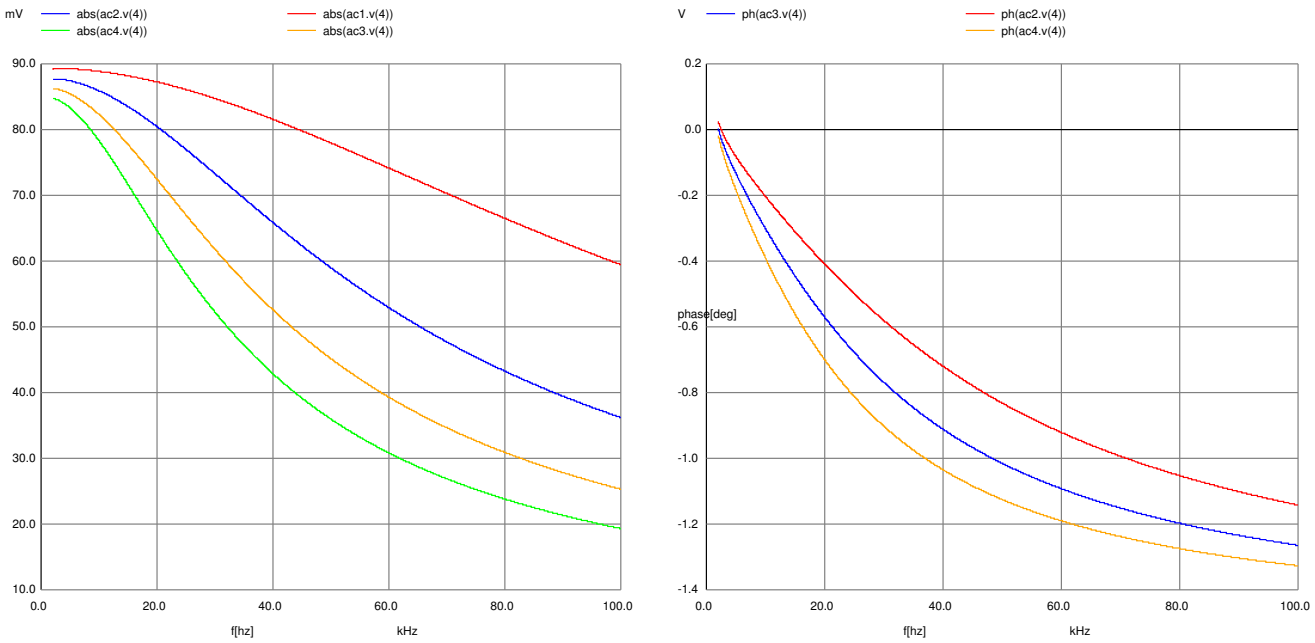



FIGURE 9 – Gauche : magnitude de la fonction de transfert d'un circuit $RC$ dont le condensateur est formé d'une varicap. Droite : phase du même circuit.

Nous constatons bien la variation de la fréquence de coupure avec la valeur de la varicap : la modélisation est fonctionnelle.

## 8.2 Modélisation du tirage de la fréquence d'oscillation

Fig. 10 présente l'évolution de la phase et de la magnitude d'un résonateur à 16 MHz lors de la variation d'un des deux condensateurs de pieds, en accord avec notre expérience.

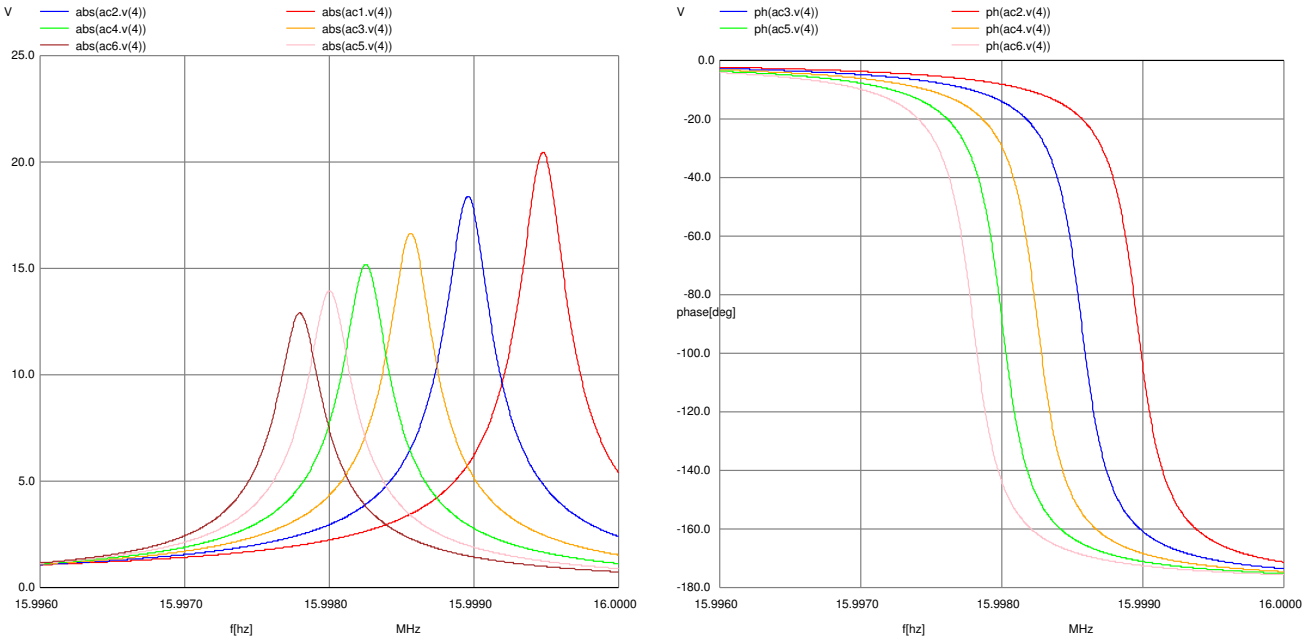



FIGURE 10 – Gauche : magnitude de la fonction de transfert d'un circuit Butterworth-van Dyke avec tirage de fréquence par un condensateur variable. Droite : phase du même circuit.

1. **Reproduire ces simulations compte tenu des informations fournies dans la documentation d'un résonateur, par exemple** `http://www.andhraelec.com/app_quartz_crystals.html`**.**
2. **Si nous devions bobiner une inductance (autour d'un noyau d'air, avec une couche unique de bobinage) de valeur équivalente à celle du modèle Butterworth-van Dyke et tenant dans un volume de 1 cm$^3$, quel serait l'ordre de grandeur de sa résistance interne ?**

Une inductance à cœur d'air de longueur $l$ et de diamètre $2R$ formée de $N$ tours présente une valeur de $L = \mu_0 \frac{N^2 \pi R^2}{l}$. La longueur de fil associée est $2\pi NR$ qui induit une résistance $\rho \frac{l}{2\pi r^2}$ avec $r$ le rayon du fil qui forme la bobine. En supposant le fil formé de cuivre, $\rho_{Cu} = 17$ nΩ.m : la résistance de la bobine est donc $\rho \frac{N \cdot 2\pi R}{\pi r^2} = \rho \frac{4L}{r\pi\mu_0 R}$. Le diamètre du fil étant imposé par la longueur de la bobine divisée par le nombre de tours, nous trouvons un diamètre de fil de 7 $\mu$m pour une longeur de fil de 40 m, soit une résistance de quelques dizaines de kΩ, à comparer aux quelques dizaines d'ohms d'un résonateur à quartz. Le facteur de qualité est inversement proportionnel à cette résistance, et impacte directement sur la stabilité de l'oscillateur résultant.

3. **Quelle serait la conséquence sur le facteur de qualité du résonateur résultant ? Commenter.**
4. **Étendre la plage de fréquences de simulation : commenter sur l'évolution de la phase. Comment se compare-t-elle à la phase d'un circuit** *RLC* **série ?**

# 9 Programmation du microcontrôleur

## 9.1 Le *bootloader*

Le mécanisme de programmation et de configuration – classiquement représenté par l'acronyme ISP (*In System Programming*, donc pas d'outil matériel dédié pour flasher le microcontrôleur) est nommé par Atmel DFU (*Device Firmware Upgrade* décrit à `www.atmel.com/Images/doc7618.pdf`), implémenté dans une série d'outils disponibles à `https://dfu-programmer.github.io/`. sous le nom de
`dfu-programmer`. Ce logiciel embarqué, fourni par défaut par Atmel, ne suivent pas le même protocole que le logiciel fourni avec les circuits utilisés en travaux pratique, commercialisés par Olimex, qui sont équipés d'un *bootloader* compatible AVR109 [1]. Ainsi, `avrdude` n'est pas utilisable mais nous utiliserons à la place `dfu-programmer`.

1. décrit à `www.atmel.com/images/doc1644.pdf`

Comme d'habitude pour un microcontrôleur stockant son programme en mémoire flash, nous devons d'abord en effacer le contenu en plaçant tous les bits à 1, avant de configurer les bits qui passent à 0 lors du transfert du programme. En fin de programmation, le microcontrôleur est réinitialisé pour lancer l'exécution depuis l'instruction située à l'adresse 0x0000. Ces opérations se résument dans le `Makefile` par

```
MCU=atmega32u2
DFU=dfu-programmer

install: $(EXEC).hex
        $(DFU) $(MCU) erase
        $(DFU) $(MCU) flash $(EXEC).hex
        $(DFU) $(MCU) reset
```

Les sources du *bootloader* exécuté au démarrage de la plateforme Olimexino 32U4 est disponible à `https://github.com/arduino/Arduino/blob/master/hardware/arduino/avr/bootloaders/caterina/` et en particulier `Caterina.c`. L'utilisation d'un *bootloader* pose un problème d'initialisation des périphériques : nous constatons par exemple que le watchdog est désactivé et l'USB initialisé. Le problème survient lorsque nous désirons exécuter notre propre code sans passer par le *bootloader* : l'oubli d'initialisation d'un périphérique pris en charge par le *bootloader* se traduira par un dysfonctionnement de notre programme particulièrement pénible à déverminer. Il est donc fondamental de bien avoir conscience des implications du *bootloader* sur le fonctionnement du microcontrôleur – un argument supplémentaire en faveur des outils *opensource* puisqu'un *bootloader* propriétaire ne peut être consulté par ses utilisateurs.

## 9.2 LED

L'allumage de diodes électroluminescentes (LED) pour indiquer l'état du microcontrôleur n'est qu'une distraction sans intérêt technique *a priori*. Le choix de placer les LEDs sur quelques broches libres – PF{1, 4, 5, 6} – est cependant l'opportunité de découvrir une nouvelle subtilité du microcontrôleur pour accéder à ces broches, à savoir leur configuration par défaut comme interface JTAG. En effet, l'Atmega32U4 propose une fonctionnalité de déverminage matérielle par l'interface standard qu'est le JTAG dévolue à ces broches. Afin d'accéder aux fonctions GPIO ou de conversion analogique-numérique, il faut désactiver la fonction JTAG du microprocesseur. Étant donné qu'il s'agit d'une fonction matérielle fondamentale si l'utilisateur en requiert l'utilisation, la désactivation n'est pas triviale : il faut confirmer deux fois de suite, avec un intervalle de moins de 4 cycles d'horloge entre les ordres, la désactivation du JTAG en mettant à 1 le bit `JTD` du registe MCUCR : "a timed sequence must be followed when changing this bit : the application software must write this bit to the desired value twice within four cycles to change its value" (datasheet p.323, section 26.5.1). On configurera donc

```
MCUCR|=(1<<JTD);
MCUCR|=(1<<JTD);
```

avant d'utiliser le port F.

## 9.3 PWM

On s'intéresse dans un premier temps à la PWM. Pour les *timers* 1 et 3, la résolution de la PWM est 10 bits, ainsi que pour le *timer* sur 10 bits numéro 4. Nous laisserons donc les deux premiers *timers* libres pour des activités requérant toute leur résolution, pour se focaliser sur le *timer* 4.

Ce *timer* se configure en PWM après avoir défini le facteur de division (*pre-scaler*) de l'horloge qui le cadence (Fig. 11). Le décompte du *timer* s'effectue de 0 à la valeur stockée dans OCR4C (voir extrait de datasheet de la Fig. 12) tandis que le déclenchement de changement d'état est déterminé par le décompte atteignant la valeur stockée dans OCR4D.

⚠La broche par laquelle la PWM est émise doit être explicitement placée en sortie (`DDRnx=1`), faute de quoi aucun signal ne sera visible.

1. **Quelle est la fréquence la plus lente accessible par le *timer4* lorsqu'il est cadencé par la sortie de la PLL interne à 48 MHz?**
2. **Démontrer le fonctionnement du *timer* en faisant clignoter une LED sur la broche PD7 (OC4D) de la carte Olimexino32U4. Faire varier la fréquence en restant dans la gamme du clignotement visible.**
3. **Démontrer le fonctionnement du *timer* en faisant varier le rapport cyclique de la PWM, en sélectionnant une fréquence qui ne permette *pas* à l'œil de visualiser le clignotement. Observer la variation d'intensité lumineuse. Démontrer que le *timer* est configuré sur 10 bits, c'est à dire que le compteur évolue de 0 à 1023.**
4. **Observer la tension en sortie du filtre passe bas et le comportement comme source de tension ajustable lorsque le rapport cyclique de la PWM varie.**

• **Bits 3 .. 0 - CS43, CS42, CS41, CS40: Clock Select Bits 3, 2, 1, and 0**

The Clock Select bits 3, 2, 1, and 0 define the prescaling source of Timer/Counter4.

**Table 15-15.** Timer/Counter4 Prescaler Select

| CS43 | CS42 | CS41 | CS40 | Asynchronous Clocking Mode | Synchronous Clocking Mode |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | T/C4 stopped | T/C4 stopped |
| 0 | 0 | 0 | 1 | PCK | CK |
| 0 | 0 | 1 | 0 | PCK/2 | CK/2 |
| 0 | 0 | 1 | 1 | PCK/4 | CK/4 |
| 0 | 1 | 0 | 0 | PCK/8 | CK/8 |
| 0 | 1 | 0 | 1 | PCK/16 | CK/16 |
| 0 | 1 | 1 | 0 | PCK/32 | CK/32 |
| 0 | 1 | 1 | 1 | PCK/64 | CK/64 |
| 1 | 0 | 0 | 0 | PCK/128 | CK/128 |
| 1 | 0 | 0 | 1 | PCK/256 | CK/256 |
| 1 | 0 | 1 | 0 | PCK/512 | CK/512 |
| 1 | 0 | 1 | 1 | PCK/1024 | CK/1024 |
| 1 | 1 | 0 | 0 | PCK/2048 | CK/2048 |
| 1 | 1 | 0 | 1 | PCK/4096 | CK/4096 |
| 1 | 1 | 1 | 0 | PCK/8192 | CK/8192 |
| 1 | 1 | 1 | 1 | PCK/16384 | CK/16384 |

FIGURE 11 – Registre de configuration de la division d'horloge alimentant le *timer* 4.

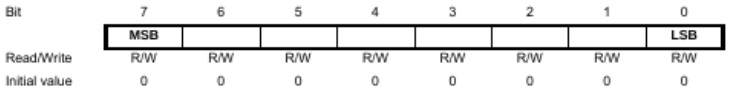
• **Bits 1:0 - WGM41, WGM40: Waveform Generation Mode Bits**

This bit associated with the PWM4x bits control the counting sequence of the counter, the source for type of waveform generation to be used, see Table 15-19. Modes of operation supported by the Timer/Counter4 are: Normal mode (counter), Fast PWM Mode, Phase and Frequency Correct PWM and PWM6 Modes.

**Table 15-19.** Waveform Generation Mode Bit Description

| PWM4x | WGM41..40 | Timer/Counter Mode of Operation | TOP | Update of OCR4x at | TOV4 Flag Set on |
|---|---|---|---|---|---|
| 0 | xx | Normal | OCR4C | Immediate | TOP |
| 1 | 00 | Fast PWM | OCR4C | TOP | TOP |
| 1 | 01 | Phase and Frequency Correct PWM | OCR4C | BOTTOM | BOTTOM |
| 1 | 10 | PWM6 / Single-slope | OCR4C | TOP | TOP |
| 1 | 11 | PWM6 / Dual-slope | OCR4C | BOTTOM | BOTTOM |

**OCR4D – Timer/Counter4 Output Compare Register D**



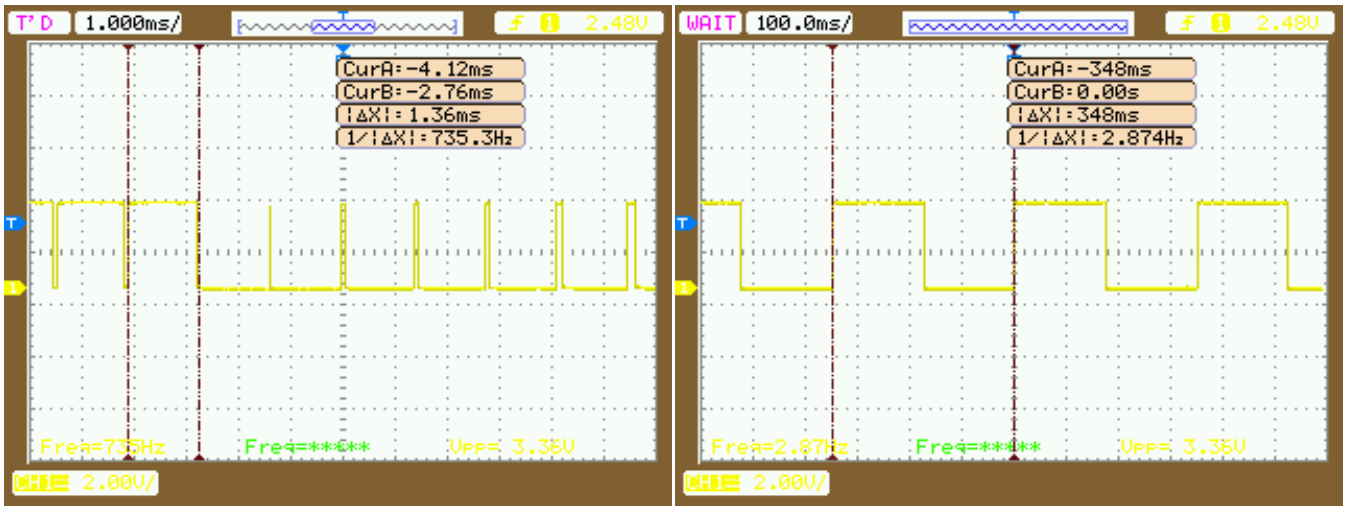

The output compare register D is an 8-bit read/write register.

The Timer/Counter Output Compare Register D contains data to be continuously compared with Timer/Counter4. Actions on compare matches are specified in TCCR4A. A compare match does only occur if Timer/Counter4 counts to the OCR4D value. A software write that sets TCNT4 and OCR4D to the same value does not generate a compare match.

A compare match will set the compare interrupt flag OCF4D after a synchronization delay following the compare event.

Note that, if 10-bit accuracy is used special procedures must be followed when accessing the internal 10-bit Output Compare Registers via the 8-bit AVR data bus. These procedures are described in section "Accessing 10-Bit Registers" on page 159.

FIGURE 12 – Bornes du décompte et déclenchement de la PWM.




## 9.4 Input Capture

Étant capable de générer un signal périodique, il nous faut désormais observer le 1 PPS issu du GPS pour obtenir un décompte du nombre d'oscillations du quartz dans un intervalle de temps connu avec précision. Une fonction matérielle permettant une telle opération est l'*input capture*. Sachant qu'un oscillateur à quartz est susceptible de varier de ±50 ppm dans la gamme de température, un oscillateur autour de 16 MHz est susceptible de varier de ±800 Hz. Cette valeur ne se code pas sur 8 bits : nous choisissons donc un des *timers* comptant sur 16 bits, soit le *timer* 1 ou 3.

Le choix du front sur lequel la valeur du compteur est capturée est important : la norme du 1 PPS définit avec précision le front montant mais pas la durée de l'impulsion. Il est donc peu judicieux d'asservir sur le front descendant : le déclenchement sur front montant se sélectionne dans Input Capture Edge Select de TCCR1B.

1. **Connecter la sortie de la PWM sur l'entrée adéquate permettant de compter un intervalle de temps au moyen du *timer* 1 (Table 1).**
2. **Constater que la valeur du décompte est en accord avec les prévisions lorsque la PWM est réglée sur la fréquence minimale.**

TABLE 1 – Exemple de mesures de sortie de PWM par Input Capture, pour divers modes et paramètres de mesures de l'Input Capture.

⚠Afin d'afficher des valeurs hexadécimales au moyen de `gnuplot`, on pensera à préfixer les valeurs des caractères `0x` afin de préciser la base de la représentation.

## 9.5 Validation de la commande

Dans un premier temps, on constatera la dérive de l'oscillateur avec la température, en se référant au 1 PPS du GPS comme référence supposée parfaite, et en comptant le nombre d'oscillations de l'oscillateur local en fonction des conditions environnementales, par exemple en chauffant le boîtier du quartz avec un fer à souder.

Noter que l'excellente stabilité de la référence (1 PPS) permet d'utiliser l'oscillateur à quartz comme excellent capteur de température. Hewlett Packard a exploité ce principe depuis longtemps [2].

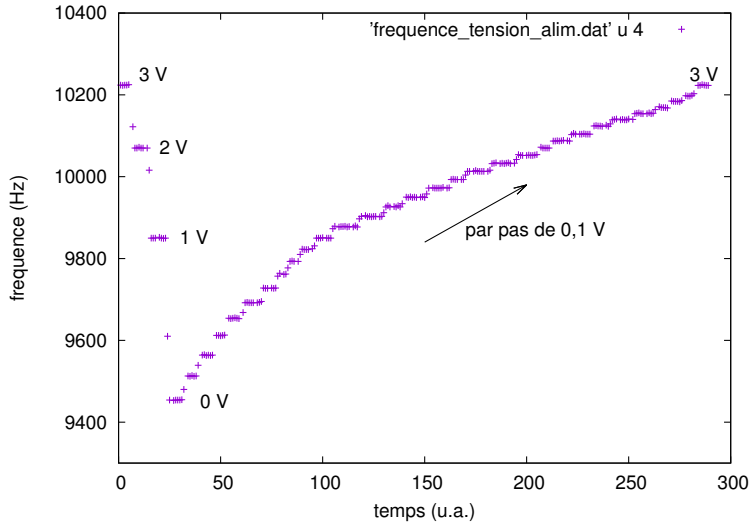
En connectant cette fois l'entrée d'Input Capture sur le 1 PPS du GPS, observer l'évolution de la fréquence du quartz lorsque la sortie de PWM varie dans sa plage de fonctionnement (Fig. 13).

1. **Quelle est la capacité de correction de la fréquence du montage ?**
2. **Que peut-on dire sur l'hypothèse sous-jacente à toute la théorie du contrôle appliquée pour réaliser la boucle d'asservissement ?**

Sous Python, avec numpy, la lecture du fichier se fait par

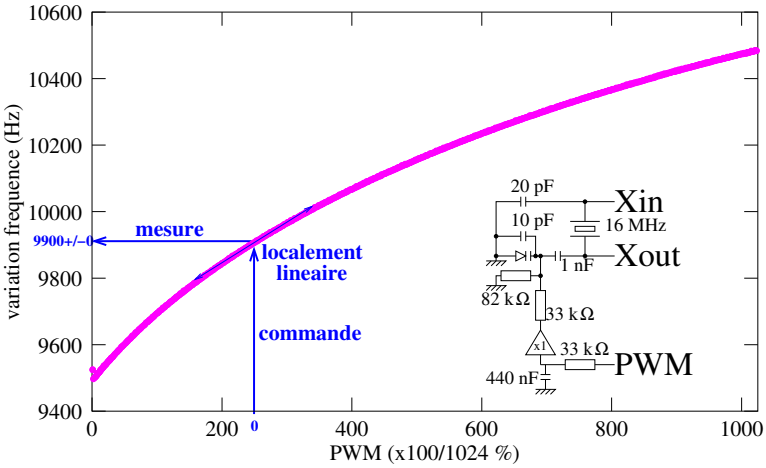
```
1 from numpy import *
2 import matplotlib.pyplot as plt
3
```

2. The Linear Quartz Thermometer – a New Tool for Measuring Absolute and Difference Temperatures, Hewlett-Packard Journal **16** (7), 1965, disponible à `http://www.hpl.hp.com/hpjournal/pdfs/IssuePDFs/1965-03.pdf`




```
pl 'frequence_tension_alim.dat' u 4
set yrange [9300:10400]
set xlabel 'temps (u.a.)'
set ylabel 'frequence (Hz)'
set label "3 V" at 10,10250
set label "2 V" at 20,10070
set label "1 V" at 30,9850
set label "0 V" at 40,9450
set label "3 V" at 280,10250
set arrow from 150,9840 to 200,9980
set label "par pas de 0,1 V" at 170,9870
repl
```

FIGURE 13 – Gauche : mesure de fréquence du quartz en fonction de la tension de polarisation de la varicap – ici par une alimentation de laboratoire. Droite : programme `gnuplot` pour afficher la courbe.




Sous GNU Octave ou Matlab, la lecture du fichier s'obtient (en supposant la présence de 5 colonnes) par

```
f=fopen('automatique');
d=fscanf(f,'%x');
plot(d(1:5:end),d(5:5:end),'.')
xlabel('PWM (*100/1024 %)')
ylabel('variation frequence (Hz)')
axis([0 1025 9400 10600])
print -dfig frequence_tension_pwm_c.fig
```

FIGURE 14 – Gauche : mesure de fréquence du quartz en fonction de la tension de polarisation de la varicap – ici par une PWM filtrée pour une tension de polarisation entre 0 et 5 V. Droite : programme GNU/Octave pour afficher la courbe.

```
mesure=loadtxt('fichier.dat', dtype='int', \
    converters={0:lambda s: (int(s,16)),1:lambda s: int(s,16), 2:lambda s: int(s,16)}) # dtype= OUTPUT type
plt.plot(mesure[:,0]-mean(mesure[:,0]),'-',label='commande')
plt.plot(mesure[:,2]-mean(mesure[:,2]),'-',label='frequence')
plt.ylabel('frequence (Hz)')
plt.xlabel('temps (s)')
plt.legend(loc=1) # ,prop={'size':6})
plt.show()
```

en supposant cette fois que les 3 colonnes (commande, overflow et input capture) de chiffres ne comportent *pas* le préfixe 0x devant les nombres en hexadécimal.

Cette courbe (Fig. 14) nous informe de plusieurs informations en plus de la plage de tirage du quartz. D'une part, nous constatons qu'une hypothèse de l'asservissement de systèmes **LTI** (*Linear Time Invariant*) n'est **pas vérifiée** puisque la relation entre entrée et sortie n'est clairement pas linéaire. Elle a cependant le bon goût d'être bijective (donc le signe de la rétroaction sera constant), et de ne présenter qu'une courbure modeste. Nous n'aurons donc qu'à éventuellement tenir compte d'une petite correction du gain donnée par la tangente à la courbe entrée sortie, ou sélectionner des paramètres d'asservissement dans le pire cas, qui fonctionnement toujours, éventuellement avec des performances un peu dégradées.

Le deuxième point que nous avons mis en évidence sur la graphique de la Fig. 14 est qu'une loi de commande est déterminée autour d'un point de fonctionnement supposé atteint pour une commande nulle. Ainsi dans cet exemple, une commande de l'ordre de 250 se traduit par une fréquence de 9900 [65536]. Lors de l'identification, nous allons commander le système **autour de ce point de fonctionnement, et effectuerons la soustraction de la valeur moyenne** de l'entrée et de

la sortie pour nous placer autour du point de fonctionnement. Ces deux biais seront ensuite ajoutés manuellement grâce à cette étape d'identification du système.

## 9.6 Boucle de régulation

### 9.6.1 Principes généraux

Ayant une mesure de la grandeur d'intérêt (fréquence de l'oscillateur à quartz) et une commande (tension de polarisation de la varicap), nous sommes en mesure d'établir une boucle de régulation[3] qui contrôle la fréquence du quartz afin de la maintenir proche d'une valeur de consigne.

Un ordinateur travaille en temps discret avec, entre deux pas d'asservissement, un intervalle de temps supposé fixe $dt$ ou, toujours en temps discret, $T_e$. Dans ce contexte, les notions de dérivée et d'intégrale se réexpriment respectivement par

$$\frac{dx}{dt} \xrightarrow[temps\ discret]{} \frac{x_n - x_{n-1}}{T_e}$$

$$X = \int x \cdot dt \xrightarrow[temps\ discret]{} X += x_n \cdot T_e$$

En temps continu, le contrôleur $c(t)$ proportionnel, intégral et dérivée s'écrit

$$c(t) = K_p \cdot e(t) + K_i \int e(t) + K_d \frac{de(t)}{dt}$$

Cette expression sera implémentée, grâce aux relations vues ci-dessus, en temps discret. Les constantes $K_d$, $K_i$ et $K_p$ sont les paramètres cruciaux qui déterminent les performances de l'asservissement. Une méthode de réglage, utilisée en temps-continu en particulier, consisite dans un premier temps à choisir $K_d$ et $K_i$ nuls pour maximiser $K_p$ jusqu'à atteindre la condition d'oscillation, pour ensuite faire croître $K_i$ pour réduire la sensibilité au bruit et finalement $K_d$ pour réduire les dépassements de la consigne. Nous ne nous servirons pas de cette méthode empirique mais exploiterons la méthode de Takahashi qui déduit $K_d$, $K_i$ et $K_p$ de la réponse indicielle (ou impulsionnelle) du système.

### 9.6.2 Implémentation

Le passage de la théorie à l'implémentation nécessite de prendre garde à un point : le codage des données sous forme de valeurs entières ne doit pas se heurter à la limite de stockage. En particulier, le terme intégral peut diverger le temps que la boucle se stabilise : nous prendrons donc soin de saturer l'intégrale si elle s'approche de la limite de stockage ($2^{N-1}-1$ sur $N$ bits), une procédure nommée *anti-windup*.

Un tel algorithme s'écrit donc

```
erreur=0;
integrale=0;
erreur_avant=0;
while(1) {
  mesure(&freq_mesure);                          // prend un temps Te
  erreur=consigne-freq_mesure;
  integrale+=erreur;                             // *Te
  sature_integrale(&integrale);                  // anti-windup
  derivee=(erreur-erreur_avant);                 // /Te
  commande_pwm=Kp*erreur+Ki*integrale+Kd*derivee; // PID
  sature_commande(commande_pwm);                 // rester dans la gamme
  erreur_avant=erreur;
}
```

Une approche récursive est établie en observant l'évolution de $c(t)$ et en particulier en considérant, en temps discret, la différence $c_{n+1} - c_n$. La somme infinie de l'intégrale se réduit dans ce cas à $K_i \cdot \varepsilon_{n+1}$ tandis que les termes proportionnel et intégral sont $K_p \cdot (\varepsilon_{n+1} - \varepsilon_n) \cdot T_e$ et $K_d \cdot \frac{(\varepsilon_{n+1}-\varepsilon_n)-(\varepsilon_n-\varepsilon_{n-1})}{T_e} = K_d \cdot \frac{(\varepsilon_{n+1}-2\varepsilon_n+\varepsilon_{n-1})}{T_e}$. En conséquent, l'algorithme peut aussi s'écrire selon une équation récursive qui exprime la transformée en Z du PID :

$$C(z) \cdot z^{-1} = E(z)\left(K_p \cdot z^{-1} + K_i \cdot T_e + K_d \frac{z - 2 + z^{-1}}{T_e}\right)$$

que l'on peut réécrire sous la forme

$$\Leftrightarrow \frac{C(z)}{E(z)} = \frac{\alpha z + \beta + \gamma z^{-1}}{z-1} = \frac{\alpha + \beta z^{-1} + \gamma z^{-2}}{1 - z^{-1}} \Leftrightarrow C_n = C_{n-1} + K_p(\varepsilon_n - \varepsilon_{n-1}) + K_i \varepsilon_n + K_d(\varepsilon_n + \varepsilon_{n-2} - 2 \cdot \varepsilon_{n-1})$$

3. **Asservissement** : la sortie doit reproduire fidèlement la consigne, avec un temps de réponse aussi court que possible.
**Régulation** : la consigne doit être maintenue, quelquesoient les perturbations et le bruit sur la mesure. Nous nous plaçons dans ce second cas.

```
erreur=0;
erreur_1=0;
erreur_2=0;
commande_pwm=0;
while(1) {
  mesure(&freq_mesure);                          // prend un temps Te
  erreur_2=erreur_1;
  erreur_1=erreur;
  erreur=consigne-freq_mesure;
  commande_pwm+=Kp*(erreur-erreur_1)+Ki*erreur+Kd*(erreur+erreur_2-2*erreur1);
// ou commande_pwm+=(Kp+Ki+Kd)*erreur-(Kp+2*Kd)*erreur_1+Kd*erreur_2;
  sature_commande_et_integrale(&commande_pwm);
}
```

### 9.6.3 Identification des coefficients

L'identification des paramètres $K_p$, $K_d$ et $K_i$ [6] passe par une **analyse de la réponse indicielle** du système. Dans notre système qui se comporte de façon quasi-idéale, nous observons la sortie (fréquence du quartz) soumis à des échelons de commande (tension polarisant la varicap). Afin d'améliorer le rapport signal à bruit, nous répétons la mesure que nous **moyennons** : le signal s'accumule tandis que le bruit se réduit comme la racine du nombre de mesures (Fig. 16). Dans le cas général d'un système bruité, il faut passer par l'**identification d'un modèle dynamique** de comportement du système formé du filtre passe-bas en sortie de la PWM, de la varicap, et du résonateur dans sa boucle d'oscillation. Le modèle le plus simple est le modèle ARX (Fig. 15) dans lequel la dymamique du bruit additif est supposée être la même que la dynamique du système. Nous minimisons ainsi le nombre de paramètres à identifier. Puisqu'un modèle *linéaire* (vérifiant donc le théorème de superposition) relie une combinaison linéaire des valeurs passées, pondérées, des sorties $s$ et les valeurs passées, pondérées, des entrées $e$, alors $s_n + \sum_k a_k \cdot s_{n-k} = \sum b_k \cdot e_{n-k} + \varepsilon_n$ avec $\varepsilon_n$ le bruit. Cela s'exprime sous forme polynomiale, dans un formalisme de transformée en $z$ ($z^{-n}$ représentant donc le retard de $n$ pas de temps) par $A(z^{-1}) \cdot S = B(z^{-1}) \cdot E + \varepsilon$ et donc $S = \frac{B}{A} \cdot E + \frac{1}{A} \cdot \varepsilon$. Les polynômes $A$ et $B$ sont ceux fournis par la fonction `arx` de GNU/Octave ou Matlab.

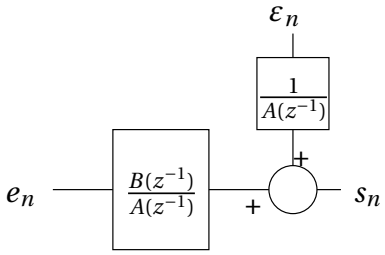


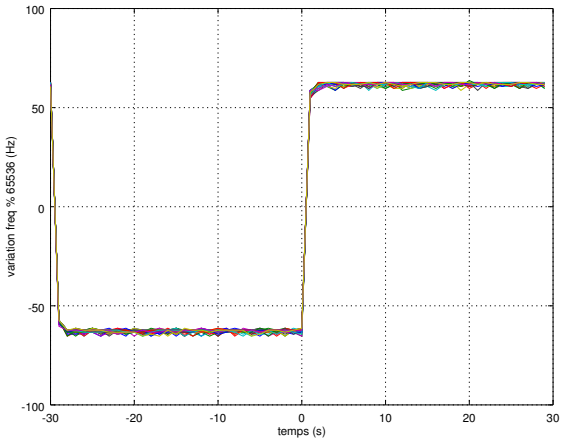

**FIGURE 15:** Modèle ARX

Si nous revenons dans le domaine temporel, l'équation de récurrence s'écrit de façon générale $s_n + a_1 \cdot s_{n-1} + ... + + a_{na} \cdot s_{n-na} = b_1 \cdot e_{n-nk} + ... + b_{n-nb} \cdot e_{n-nk-nb+1} + \varepsilon_n$ qui est complètement déterminée par la définition de $na$, $nb$ et $nk$. Ce sont ces paramètres que nous fournissons lorsque nous écrivons `mydata=iddata(s,e,1);arx(mydata,'na',4,'nb',4,'nk',0);` [4].

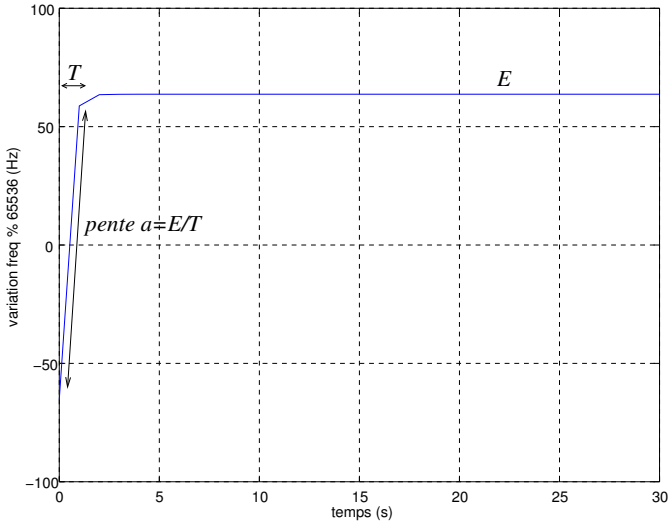
```
close;clear
f=fopen('PIDnone_openloop_cmd200-400.dat');
d=fscanf(f,'%x');
Te=1.; % periode d'echantillonnage = 1s

entree=d(1:3:end);entree=entree-mean(entree);
sortie=d(3:3:end);sortie=sortie-mean(sortie);
sortie=sortie(1:end-1);
entree=entree(2:end);
mydata=iddata(sortie,entree,Te); % vv pour octave, PAS matlab
[arx111,x0]=arx(mydata,'na',1,'nb',1,'nk',0); % Att. nk
sysd=tf(arx111)
[Y,T]=step(sysd,30);

tau=0; % la plus grande pente est en 0
a=0.6/1; % pente de la reponse indicielle

% Takahashi
ki=0.27/(a*(tau+0.5*Te)^2)
kp=0.9/(a*(tau+Te))-0.5*ki*Te

corr=tf([kp+ki*Te -kp],[1 -1],Te) % PID
% autre solution : moyenne des reponses indicielles
nombre=floor(length(entree)/60)-1;
e=reshape(entree(29:nombre*60+28),60,nombre);
s=reshape(sortie(30:nombre*60+29),60,nombre);
hold on
plot([-30:29],s); %plot([-30:29],e)
xlabel('temps (s)');ylabel('variation freq % 65536 (Hz)');grid

figure;plot([-30:29],mean(s')); %plot([-30:29],e)
xlabel('temps (s)');ylabel('variation freq % 65536 (Hz)');grid

figure;plot(T,(Y/max(Y))*(max(max(s))-min(min(s)))+min(min(s)))
xlim([0 30])
xlabel('temps (s)');ylabel('variation freq % 65536 (Hz)');grid

% modele par intercorrelation
% Imp = impulse(mydatad,[-1 40],'PW',[]) % pour Matlab
y=xcorr(sortie, entree);
Cuu=xcorr(entree, entree);
N=20; % longueur de la reponse impulsionelle desiree
M=2000; % nb de lignes de la matrice

Y=y(length(sortie):length(sortie)+M);
X=Cuu(length(sortie):length(sortie)+N)';
for i=1:1:M
    X=[X;Cuu(length(sortie)-i:length(sortie)+N-i)'];
end
hopt=inv(X'*X)*X'*Y;
% plot(hopt);xlabel('temps (s)');ylabel('reponse indicielle (u.a.)');
plot(cumsum(hopt));xlabel('temps (s)');ylabel('reponse indicielle (u.a.)');
```

Les coefficients du PID (proportionnel et intégral suffiront dans une boucle d'asservissement qui ne nécessite pas de D) s'établissent à partir de la réponse indicielle du système mesurée près du point de fonctionnement (Fig. 16). Ces courbes expérimentales sont acquises pour une commande passant, toutes les 30 secondes, de 200 à 400, soit une excursion de

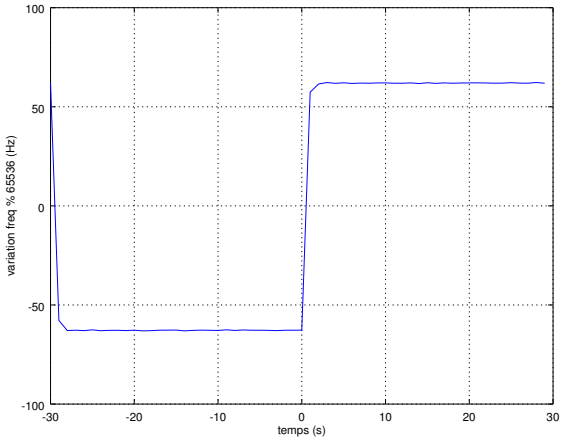
4. on notera étrangement que le même résultat est obtenu par la commande GNU/Octave `arx(mydata,'na',4,'nb',4,'nk',0);` et Matlab `arx(mydata,[4 4 1]);`. Il semble donc y avoir désaccord sur la définition de `nk`!



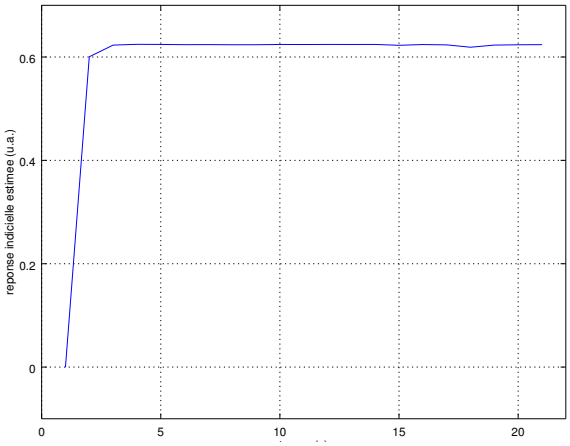

Mesures expérimentales de la sortie lorsque le système est soumis à une entrée en créneau, répétée 42 fois




Réponse à un échelon (non-unitaire) du modèle ARX




Moyenne des mesures expérimentales de la sortie lorsque le système est soumis à une entrée en crénau




Réponse indicielle (système soumis à un échelon unité) par intercorrélation.

FIGURE 16 – Haut-gauche : 42 mesures expérimentales de la réponse indicielle du système, acquises pendant 30 secondes chacune. Bas-gauche : moyenne des mesures expérimentales. Haut-droite : réponse indicielle du modèle ARX(1,1,0) ajustant au mieux les données expérimentales. Bas-droite : réponse indicielle déduite de l'intercorrélation entre sorties et entrées.

200 unités. Nous constatons que la sortie varie autour de valeurs de l'ordre de ±60, soit un gain de 0,6. Par ailleurs, le système ne présente pas de retard, et le passage de 0 à 0,6 se fait en 1 pas de temps donc 1 s, soit une pente de la courbe à l'origine de 0,6/1=0,6. De ces constantes, nous déduisons les coefficients dans une loi de type PI par les règles de Takahashi (Table 2).

| Modèle | Coefficients issus de l'essai indiciel |
|---|---|
| proportionnel $P$ | $K_p = \frac{1}{a \cdot (L + T_e)}$ |
| proportionnel-intégral $PI$ | $K_i = \frac{0{,}27}{a \cdot (L + 0{,}5 \times T_e)^2}$, $K_p = \frac{0{,}9}{a \cdot (L + 0{,}5 \times T_e)} - 0{,}5 \times K_i \times T_e$ |
| $PID$ | $K_i = \frac{0{,}6}{a \cdot (L + 0{,}5 \times T_e)^2}$, $K_p = \frac{1{,}2}{a \cdot (L + 0{,}5 \times T_e)} - 0{,}5 \times K_i \times T_e$, $K_d = \frac{0{,}5}{a}$ |

TABLE 2 – Règles d'identification des coefficients de $K_p$, $K_i$ et $K_d$ pour un système échantillonné en temps discrets. $L$ est le retard (chez nous nul), $a$ la pente et $T_e$ la période d'échantillonnage [7, p.243].

**Concept de matrice pseudo-inverse :** la résponse impulsionnelle du système est issue de l'analyse par corrélation de la sortie à l'entrée, qui doit être normalisée par la stastique du signal d'entrée, c'est à dire sa fonction d'autocorrélation. Si le signal d'entrée est un bruit, le problème est simple puisque sa fonction d'autocorrélation est un Dirac et la matrice d'autocorrélation se réduit à un scalaire (la puissance du bruit) multiplié par l'identité (tous les termes sont nuls sauf pour le retard 0). Dans le cas général, la sortie $Y$ est une combinaison linéaire $A$ des valeurs passées de la sortie et de l'entrée $X$ : $Y = X \cdot A + \varepsilon$ avec $Y$ une matrice de 1 colonne et $N$ lignes si la mesure est répétée $N$ fois, $A$ un vecteur colonne contenant les $M$ coefficients de pondération $a_k$ et $b_k$ des valeurs passées de la sortie et de l'entrée, et $X$ une matrice contenant une représentation, décallée dans chaque lignes, des valeurs passées de la sortie et de l'entrée

$$\underbrace{\begin{pmatrix} y_n \\ y_{n-1} \\ y_{n-2} \\ y_{n-2} \\ \dots \end{pmatrix}}_{Y(1\times N)} = \underbrace{\left(\begin{array}{ccccc|ccc} y_{n-1} & y_{n-2} & y_{n-3} & \dots & x_{n-1} & x_{n-2} & \dots \\ y_{n-2} & y_{n-3} & y_{n-4} & \dots & x_{n-2} & x_{n-3} & \dots \\ y_{n-3} & y_{n-4} & y_{n-5} & \dots & x_{n-3} & x_{n-4} & \dots \\ y_{n-4} & y_{n-5} & y_{n-6} & \dots & x_{n-4} & x_{n-5} & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{array}\right)}_{X(M\times N)} \cdot \underbrace{\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \dots \\ b_1 \\ b_2 \\ \dots \end{pmatrix}}_{A(1\times M)}$$

Ainsi, une version optimale de $A$ est issue, en considérant que le bruit s'abaisse statistiquement avec un nombre croissant de représentations, comme $A = X^{-1} \cdot Y$. Cependant, si $X$ n'est pas carrée (i.e. $M \neq N$), alors $X^{-1}$ n'existe pas mais est remplacée par la pseudo-inverse $(X^t \cdot X)^{-1} \cdot X^t$ avec $X^t$ la transposée de $X$. Si $X$ est carrée, alors la pseudo-inverse de $X$ vaut $X^{-1}$. On se convainc par une analyse dimensionnelle que $X$ est formée de $N$ lignes et $M$ colonnes, alors $X^t$ est formée de $M$ lignes et $N$ colonnes, et $X^t \cdot X$ ainsi que $(X^t \cdot X)^{-1}$ sont formées de $M \times M$ éléments (la matrice carrée est inversible). Ensuite, $(X^t \cdot X)^{-1} \cdot X^t$ est formée de $N$ colonnes et $M$ lignes, qui multipliée par $Y$ vecteur de $N$ lignes donne un vecteur de $M \times 1$ éléments : nous avons bien un résultat contenant une estimation optimals des $(a_k, b_k)$ coefficients liant les valeurs passées de la sortie et de l'entrée à la valeur courante de la sortie.

Dans nos exemples où la période d'échantillonnage est $T = 1$ s, $a = 0,6$ et $L = 0$, nous trouvons à l'issue de l'essai indiciel $K_i = \frac{0,27}{0,6\times(0+0,5)^2} = 1,8$ et $K_p = \frac{0,9}{0,6\times(0+0,5)} - 0,5 \times 1,8 \times 1 = 2,1$. ⚠*Noter l'erreur dans [8] où le retard L est, de façon erronée, au numérateur de $K_p$ et $K_i$.*

⚠Lors de l'utilisation de calculs sur les entiers, il est interdit de représenter des nombres à virgule tels que 2,1 ou 1,8. Une façon de résoudre le problème, sans passer par un calcul à virgule fixe, consiste à exprimer le nombre à virgule comme une fraction rationnelle. La fonction `rats(v,l)` de GNU/Octave propose la fraction rationnelle représentant le nombre à virgule `v` avec `l` décimales.
**Exprimer 2,13 comme fraction rationnelle. Quel critère doit vérifier le numérateur? Quelle limite rencontrerons nous pour exprimer un nombre avec un grand nombre de décimales tel que $\frac{400\cdot 10^6}{2^{32}}$ ?**

# 10 Simulation du système identifié

Un oscillateur contrôlé en tension a une réponse quasi-instantanée compte tenue de la période d'échantillonnage qui nous intéresse. Le modèle est donc presque un retard pur, un modèle du premier ordre de type `arx('na',1,'nb',1,'nk',0);` devrait suffire. Le résultat de l'identification à partir de mesures expérimentales pour une consigne oscillant entre 200 et 400 est $f = D(V)$

```
        0.6
 y1:  ----------
      z - 0.0382
```

qui fournit le comportement du dynamique près de son point de fonctionnement, qui est quant à lui identifié par une ajustement polynomial de la relation statique commande-fréquence $f = F(V)$

```
x=[0 400 1023]
y=[9500 10000 10500]
a=polyfit(x,y,2)
  -4.374e-04   1.4249e+00   9.5000e+03
```

Nous traduisons ces résultats issus de GNU/Octave en C pour une modélisation du système en boucle fermée. Pour ce faire, nous devons établir le gain du système dynamique sur toute sa plage de fonctionnement. En effet, notre système ne présente pas une relation linéaire entre entrée et sortie sur toute la plage des commandes : l'identification du système ne s'est faite que localement autour d'un point de fonctionnement ($(200+400)/2 = 300$). Nous avons vu qu'autour de ce point de fonctionnement, le gain est de l'ordre de 0,6. Nous devons donc éliminer ce gain du modèle ARX localement établi et le

remplacer par le gain $Gain_{local}$ issu de la relation statique entrée sortie : $0,6 = G \times \left.\frac{dF}{dV}\right|_{V=300} \Leftrightarrow 0,6 = G \times 1,2945 \Leftrightarrow G = 0,515$.
Le modèle ARX global est donc

$$\frac{0,515}{z-0,382} \times Gain_{local} = \frac{0,515 \cdot z^{-1}}{1-0,382 \cdot z^{-1}} \times Gain_{local}$$

```
void input_capture() // appele' par interruption : d et res sont globales volatile
{float gain;
 gain=-0.0004374*2*d+1.4249;
 res=0.0382*sy1+0.515*gain*x1; // 0.6=0.515*gain @ d=300
 x1=d;
 sy1=res;
 res+=(-0.0004374*d*d+1.4249*d+9500.);
}
```

qui donne les résultats de la Fig. 17

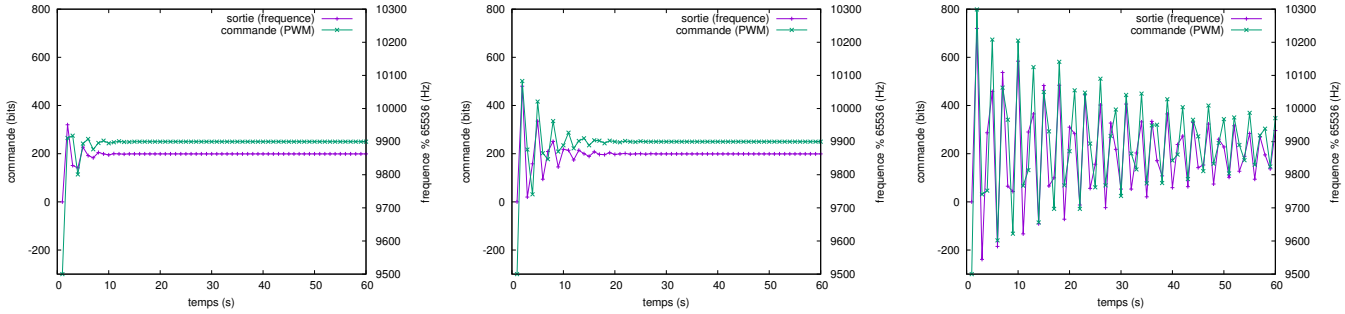



FIGURE 17 – Simulation du système modélisé comme une fonction de transfert du premier ordre, avec pour coefficients $K_p$ et $K_i$ respectivement 0,4 et 0,4 à gauche ; 0,6 et 0,6 au milieu ; et 0,6 et 1,2 à droite. Dans tous les cas $K_d = 0$.

## 11 Validation du projet

Dans un premier temps, nous constatons que la fréquence de l'oscillateur est dépendante de l'environnement du résonateur, et en premier lieu de sa température (Fig. 18). Notre objectif est donc de commander la fréquence de l'oscillateur pour compenser ces fluctuations.

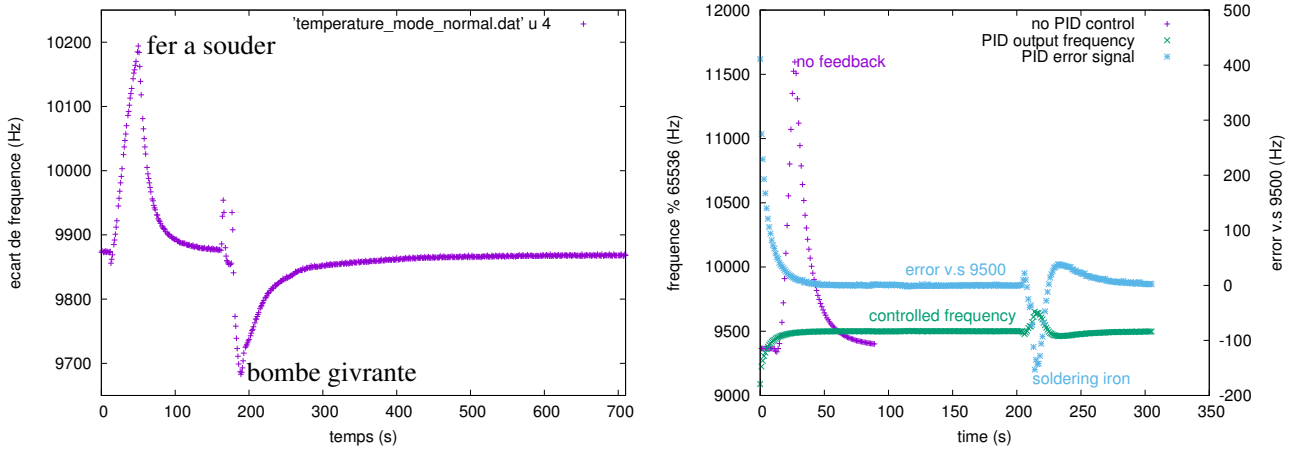



FIGURE 18 – Gauche : oscillateur en boucle ouverte, soumis à un échauffement (fer à souder) et un refroidissement (bombe givrante) . Droite : oscillateur en boucle ouverte (violet) et en boucle fermée (vert pour la fréquence, bleu pour l'erreur) soumis à un échauffement par fer à souder. Noter la correction amenée par la boucle fermée visant à compenser la dérive de fréquence liée à la dérive de température. Noter que pour un échauffement presque identique, la variation de fréquence de l'oscillateur a été divisée par un facteur supérieur à 10 (de plus de 2000 Hz à moins de 200 Hz).

1. **Démontrer la capacité de correction de l'oscillateur pour compenser les perturbations externes, et ce en vue de stabiliser la sortie de fréquence lorsque le résonateur est, par exemple, chauffé.**

# 12 Conclusion

Nous avons été capables d'implémenter, avec des modifications matérielles mineures, l'asservissement d'un oscillateur à quartz sur le signal 1 PPS du GPS, afin de compenser les dérives de fréquences avec les conditions environnementales du quartz.

1. **Les variations observées sont cependant bien moindres que celles mentionnées dans [5]. Commenter les causes de cette différence.**

Le 1 PPS est un signal lent qui ne permet que d'asservir périodiquement le quartz sur un signal stable à long terme. Ce principe est exactement celui des horloges atomiques, dans lesquelles un signal sonde lent (transition de niveau énergétique des atomes de césium excités par un signal micro-onde cadencé par le quartz qui doit être asservi) permet d'observer et donc de corriger les dérives lentes du quartz. Nous sommes dont dans une situation où deux constantes de temps sont mises en jeu : à court terme, le quartz avec son excellent facteur de qualité, et donc sa capacité à stocker de l'énergie pour s'affranchir des perturbations environnantes, détermine la stabilité court-terme de l'horloge. Cependant le quartz subit les dérives lentes de son environnement (température, contrainte, oxydation des électrodes et autres vieillissements) qui sont observées par comparaison avec un signal de référence insensible à ces effets, par exemple une transition atomique.

1. Supposons que nous voulions maintenant asservir deux oscillateurs rapides, de deux microcontrôleurs distants : comment nous y prendrions nous? quel résultat pourrions nous attendre en terme de qualité de la synchronisation?

Nous avons imposé un contrôleur de type PID par soucis de robustesse et de simplicité.

1. Comparer avec les autres contrôleurs vus en cours, par exemple (D)LQR.

# A Trames du GPS Motola Oncore

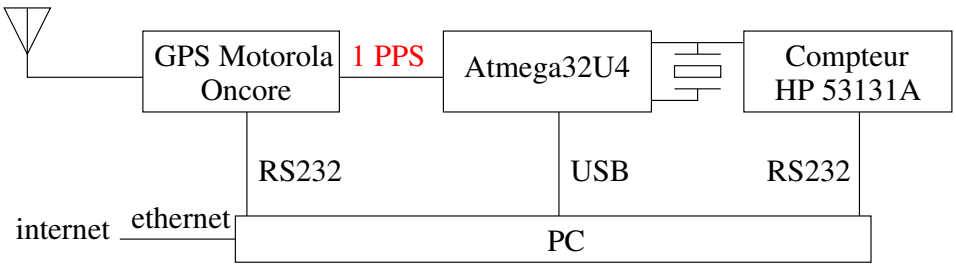



FIGURE 19 – Schéma du dispositif de mesure et validation (compteur de fréquence externe de référence).

Le 1 PPS n'est exact que si le récepteur GPS est en vue des satellites et reçoit les informations permettant de transmettre la stabilité des horloges atomiques embarquées vers le sol. Il est donc prudent d'observer les messages émis par le récepteur GPS pour s'assurer qu'il a bien acquis les messages des satellites et que son oscillateur local est asservi sur ces signaux. Motorola a choisi de communiquer dans un format binaire sur port RS232 (Fig. 19) au débit de 9600 bauds, 8N1, selon un protocole propriétaire décrit dans `https://www.febo.com/pages/hardware/VPCommands.pdf`. Nous y apprenons en particulier que les messages commençant par les lettres `@@Ea` contiennent la date, l'heure et la position (dans un référentiel quelque peu inhabituel). Un petit programme en C fourni ci-dessous permet de capturer ces informations et les afficher : les arguments de `@@Ea` sont `b 15 7 df 9 11 1a 3b 9a c4 1` qui indiquent que les mesures sont effectuées le 21 Novembre 2015 (0x0b=11 pour le mois, 0x15=21 pour le jour, 0x07DF=2015 pour l'année, 0x09 pour l'heure en temps universel, 0x11=17 pour les minutes et 0x1A pour les secondes ...). Nous constatons que le 7ème argument s'incrémente bien toutes les secondes, validant le format de la trame. Si ces données sont erronées, le récepteur n'est pas asservi sur la constellation de satellites et le 1 PPS n'est pas stabilisé par les horloges atomiques spatiales.

Listing 1 – Initialisation du port série : `rs232.c`

```
1 /* All examples have been derived from miniterm.c */
2 #include "rs232.h"
3
4 extern struct termios oldtio,newtio;
5
6 int init_rs232()
7 {int fd;
8  fd=open(RS232DEVICE, O_RDWR | O_NOCTTY );
9  if (fd <0) {perror(RS232DEVICE); exit(-1); }
```

```
 tcgetattr(fd,&oldtio); /* save current serial port settings */
 bzero(&newtio, sizeof(newtio)); /* clear struct for new port settings */
 newtio.c_cflag = BAUDRATE | CS8 | CLOCAL | CREAD; /* _no_ CRTSCTS */
 newtio.c_iflag = IGNPAR; // | ICRNL |IXON;
 newtio.c_oflag = IGNPAR; //ONOCR|ONLRET|OLCUC;
 newtio.c_cc[VTIME] = 0; /* inter-character timer unused */
 newtio.c_cc[VMIN] = 1; /* blocking read until 1 character arrives */
 tcflush(fd, TCIFLUSH);tcsetattr(fd,TCSANOW,&newtio);
 return(fd);
}

void free_rs232(int fd)
{//tcsetattr(fd,TCSANOW,&oldtio);
 close(fd);} /* restore the old port settings */
```

Listing 2 – Initialisation du port série : `rs232.h`

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#include <sys/types.h>
#include <sys/stat.h>
#include <string.h> /* declaration of bzero() */
#include <fcntl.h>
#include <termios.h>
int init_rs232();
void free_rs232();
void sendcmd(int,char*);
struct termios oldtio,newtio;

#define BAUDRATE B9600
#define RS232DEVICE "/dev/ttyUSB0"
```

Listing 3 – Lecture des trames : `read_oncore.c`

```
// lecture des trames binaires issues d'un Motorola Oncore
// et affichage de la trame @@Ea qui contient la date de mesure
// => utile pour verifier que le GPS est locke' sur les satellites
#include "rs232.h"
#include <sys/io.h>
#include <sys/ioctl.h>
#include <arpa/inet.h>

int main(int argc,char **argv)
{int fd,k;
 unsigned char cmdc;
 unsigned short year=0;
 unsigned short month=0;
 unsigned short day=0;
 unsigned short hour=0;
 unsigned short min=0;
 unsigned long sec=0,fracs=0,latitude=0,longitude=0,height;

 fd=init_rs232();
 if (fd<0) {printf("oups\n");return -1;}
 while(1) {
   do {read(fd,&cmdc,1);} while (cmdc!='@'); // resync
   read(fd,&cmdc,1); // @
   if (cmdc=='@')
     {read(fd,&cmdc,1); // E
      if (cmdc=='E')
        {read(fd,&cmdc,1); // a
         if (cmdc=='a')
            {for (k=0;k<72;k++)
                 {read(fd,&cmdc,1);printf("%hhx␣",cmdc);
```

```
                if (k==2) year=cmdc;
                if (k==3) year=year*256+cmdc;
                if (k==0) month=cmdc;
                if (k==1) day=cmdc;
                if (k==4) hour=cmdc;
                if (k==5) min=cmdc;
                if (k==6) sec=cmdc;
                if (k==7) fracs=cmdc;
                if (k==8) fracs=fracs*256+cmdc;
                if (k==9) fracs=fracs*256+cmdc;
                if (k==10) fracs=fracs*256+cmdc;
                if (k==11) latitude=cmdc;
                if (k==12) latitude=latitude*256+cmdc;
                if (k==13) latitude=latitude*256+cmdc;
                if (k==14) latitude=latitude*256+cmdc;
                if (k==15) longitude=cmdc;
                if (k==16) longitude=longitude*256+cmdc;
                if (k==17) longitude=longitude*256+cmdc;
                if (k==18) longitude=longitude*256+cmdc;
                if (k==23) height=cmdc;
                if (k==24) height=height*256+cmdc;
                if (k==25) height=height*256+cmdc;
                if (k==26) height=height*256+cmdc;
               }
            printf("\n => date=%d/%d/%d time=%d:%d:%d-%d pos=%f %f %f\n",year,month,day,hour,min,sec,fracs,\
                  (float)latitude/324000000.*90.,(float)longitude/648000000.*180.,(float)height/100.);
          }
       }
    }
  }
  return(0);
}
```

La sortie de ce programme qui analyse les trames issues du récepteur GPS Oncore, et garantit ainsi la réception des signaux reçus des satellites, est de la forme

```
1 2 7 d3 7 1 c 0 7 eb ca a 23 6d de 1 49 18 df 0 0 90 c6 0 0 7d f6 0 0 0 0 0 0 0 0 1 9 0 b5 0 6 0 aa 0 18 0 99 0
e 8 4f a0 1 0 9a 0 19 0 95 0 14 0 9d 0 1c 0 a7 0 8 43 d a
 => date=2003/1/2 time=7:1:12-519114 pos=47.248347 5.991031 322.460000
1 2 7 d3 7 1 d 0 8 1d 8f a 23 6d de 1 49 18 df 0 0 90 c6 0 0 7d f6 0 0 0 0 0 0 0 0 1 9 0 a2 0 6 0 9d 0 18 0 ca 0
e 8 4e a0 1 0 aa 0 19 0 ad 0 14 0 a9 0 1c 0 ae 0 8 b9 d a
 => date=2003/1/2 time=7:1:13-531855 pos=47.248347 5.991031 322.460000
1 2 7 d3 7 1 e 0 8 4f 55 a 23 6d de 1 49 18 df 0 0 90 c6 0 0 7d f6 0 0 0 0 0 0 0 0 1 9 0 a5 0 6 0 b4 0 18 0 ab 0
e 8 4b a0 1 0 ad 0 19 0 a5 0 14 0 9d 0 1c 0 b2 0 8 5f d a
 => date=2003/1/2 time=7:1:14-544597 pos=47.248347 5.991031 322.460000
1 2 7 d3 7 1 f 0 8 81 1c a 23 6d de 1 49 18 df 0 0 90 c6 0 0 7d f6 0 0 0 0 0 0 0 0 1 9 0 91 0 6 0 bb 0 18 0 9e 0
e 8 49 a0 1 0 c9 0 19 0 b3 0 14 0 a8 0 1c 0 ba 0 8 9a d a
 => date=2003/1/2 time=7:1:15-557340 pos=47.248347 5.991031 322.460000
c 11 7 df 7 1 10 0 8 b2 e4 a 23 6d de 1 49 18 df 0 0 90 c6 0 0 7d f6 0 0 0 0 0 0 0 0 1 9 0 95 0 6 0 c7 0 18 0 9f 0
e 8 44 a0 1 0 e2 0 19 0 9e 0 14 0 a1 0 1c 0 b1 0 8 2c d a
 => date=2015/12/17 time=7:1:16-570084 pos=47.248347 5.991031 322.460000
c 11 7 df 7 1 11 0 8 e4 ac a 23 6d de 1 49 18 df 0 0 90 c6 0 0 7d f6 0 0 0 0 0 0 0 0 1 9 0 99 0 6 0 b5 0 18 0 97 0
e 8 47 a0 1 0 e7 0 19 0 9f 0 14 0 9c 0 1c 0 c5 0 8 b d a
 => date=2015/12/17 time=7:1:17-582828 pos=47.248347 5.991031 322.460000
c 11 7 df 7 1 12 0 9 16 76 a 23 6d de 1 49 18 df 0 0 90 c6 0 0 7d f6 0 0 0 0 0 0 0 0 a 1 9 0 8a 0 6 0 c0 0 18 0 9e 0
e 8 45 a0 1 0 ee 0 19 0 ae 0 14 0 9a 0 1c 0 c7 0 8 7a d a
 => date=2015/12/17 time=7:1:18-595574 pos=47.248347 5.991031 322.460000
c 11 7 df 7 1 13 0 9 48 7d a 23 6d de 1 49 18 df 0 0 90 c6 0 0 7d f6 0 0 0 0 0 0 0 0 a 1 9 0 96 0 6 0 be 0 18 0 9f 0
e 8 43 a0 1 0 db 0 19 0 9f 0 14 0 97 0 1c 0 bd 0 8 38 d a
 => date=2015/12/17 time=7:1:19-608381 pos=47.248347 5.991031 322.460000
c 11 7 df 7 1 14 0 9 7a 47 a 23 6d de 1 49 18 df 0 0 90 c6 0 0 7d f6 0 0 0 0 0 0 0 0 a 1 8 0 96 0 a 0 ca 0 b 0 96 0
e 8 46 a0 1 0 e0 0 10 0 a8 0 12 0 a2 0 1b 0 c0 0 8 1d d a
 => date=2015/12/17 time=7:1:20-621127 pos=47.248347 5.991031 322.460000
c 11 7 df 7 1 15 0 9 ac 10 a 23 6d de 1 49 18 df 0 0 90 c6 0 0 7d f6 0 0 0 0 0 0 0 0 a 1 8 0 ff 0 a 0 ff 0 b 0 ff 0
e 8 45 a0 1 0 e9 0 10 0 ff 0 12 0 ff 0 1b 0 ff 0 8 97 d a
 => date=2015/12/17 time=7:1:21-633872 pos=47.248347 5.991031 322.460000
```

Nous constatons qu'initialement le récepteur n'était pas locké (date 2003), et le signal est acquis en milieu de fichier (passage de la date à 2015 et de l'heure à 7 h en temps universel). La localisation du récepteur est correcte, à 47°N et près de 6°E.

# Références


[1] T. Hunkin & R. Garrod, *Secret Life Of Machines : The Quartz Watch* (1991), disponible à `https://www.youtube.com/watch?v=nQ9_bOIj49s`

[2] J.-M Friedt, *Introduction à SPICE3 : simulation de circuits électroniques, et au-delà*, OpenSilicum 1 (Janvier-Mars 2011) – noter le changement de convention sur `set hcopypscolor=1` (au lieu de `true`) et l'inversion des argument `color0` et `color1` comme couleur de fond et de police de caractères.

[3] J.R. Vig, *Quartz Crystal Resonators and Oscillators*, tutorial UFFC, New Orleans (2000), `http://www.am1.us/Local_Papers/U11625%20VIG-TUTORIAL.pdf`, diapositive 73

[4] C. Basso, *SPICE Analog Behavioral Modeling of Variable Passives*, Power Electronics Technology (April 2005)

[5] J.-M. Friedt, A. Masse, F. Bassignot, *Les microcontroleurs MSP430 pour les applications faibles consommations – asservissement d'un oscillateur sur le GPS.*, GNU/Linux Magazine France **98**, Octobre 2007, disponible à `http://jmfriedt.free.fr/lm_msp430.pdf`

[6] K. Åström & T. Hägglund, *PID controllers – 2nd Ed.*, Instrument Society of America (1995)

[7] Y. Takahashi, C.S. Chan & D.M. Auslander, *Parametereinstellung bei linearen DDC-Algorithmen*, Automatisierungstechnik (1971), 237-244

[8] A. Besançon-Voda & S. Gentil, *Régulateurs PID analogiques et numériques*, Tech. de l'ingénieur R7416 (1999), ou sans les fautes, le cours de Gonzalo Cabodevila disponible à `http://jmfriedt.free.fr/Gonzalo_cours1A.pdf`