Réalisation d'un VFO
à l'aide d'un DDS piloté par Arduino

Frédéric DI MANNO, F1HKJ

Le VFO d'origine de mon vieux et solide FT-707 avait une fâcheuse tendance à dériver en fréquence. Disposant d'un générateur DDS AD9851 issu d'un ancien kit de F1BBU et piloté par un PIC 16F628 préprogrammé, l'idée m'est venue de le transformer en VFO avec le DDS piloté cette fois par une platine à microcontrôleur "Arduino Uno".

Petit historique

Sur la platine générateur HF à DDS j'ai remplacé l'amplificateur de sortie, qui était à l'origine un monolithique MAV11, par un transfo élévateur 1:4 à tore suivi d'un buffer à Fets. Le signal de sortie est moins élevé qu'avec le MAV11, mais il est plus pur.
Pour la suite, je suis allé à la pêche aux renseignements sur le Web et j'ai trouvé un article sur l'intéressant site d'Alain F5MNA (f5mna.free.fr). Il y décrit un VFO à DDS pour la bande 40m piloté par un Arduino. Je me suis inspiré, avec son accord, du programme d'Alain en ajoutant la gestion d'une commutation USB-LSB, plus celle d'un encodeur rotatif pour la fréquence.
Je vais vous décrire ma réalisation comme VFO du FT-707, mais le système peut être utilisé dans de multiples autres applications. Auparavant, je vais commencer par rappeler les principes de fonctionnement du DDS, de la platine Arduino uno et de leur association en vue de réaliser un générateur HF.

Principe du DDS

Le **DDS** ("**D**irect **D**igital **S**ynthesis oscillator" ou "oscillateur à **S**ynthèse **N**umérique **D**irecte", est composé de trois parties :

- Un **N.C.O.** ("**N**umerical **C**ontrolled **O**scillator") qui génère au rythme d'une horloge les valeurs numériques d'un signal sinusoïdal analogique qui serait échantillonné à la fréquence de l'horloge.
- Un **DAC** ("**D**igital **A**nalogic **C**onverter" ou **CNA** ("**C**onvertisseur **N**umérique **A**nalogique") qui convertit les valeurs numériques en paliers de tension.
- Un **filtre passe-bas** dit "filtre de reconstruction" (domaine temporel) qui intègre le signal avec pour résultat une interpolation entre les paliers. On obtient ainsi un signal analogique pur. Dans le domaine fréquentiel, le filtre est dit aussi "anti-repliement" car il supprime les nombreux signaux résultant des combinaisons du signal utile avec l'horloge et tous leurs harmoniques (les "repliements").

Voir sur la figure 1 le synoptique d'un générateur DDS et le principe du NCO.

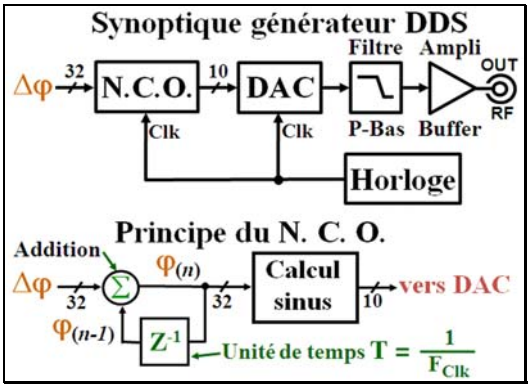


Figure 1 "**Générateur DDS**"

Principe du NCO

C'est une fonction purement numérique.
L'amplitude d'un signal sinusoïdal est fonction du **sinus** d'un angle qui tourne de **ω** radians par seconde (**ω= 2πF**).
Echantillonner un signal sinusoïdal **F1** à la fréquence **Fe** revient à fournir les valeurs successives du sinus d'un angle **φ** qui augmente régulièrement de **Δφ** en fonction du rapport **F1/Fe**.
L'angle de l'horloge (**Fe**) tournant de **2π** radians entre deux échantillons (une période), il est facile de calculer **Δφ** qui est égal à : **2π(F1/Fe)** radians ou **360(F1/Fe)** degrés.
Reste à calculer **sin(φ_n)** avec $\varphi_n = \varphi_{n-1} + \Delta\varphi$.
Le **N.C.O.** calcule et fournit à sa sortie la valeur numérique de **sin(φ_n)** à chaque coup d'horloge. On lui communique à l'aide d'un contrôleur la valeur de **Δφ** correspondant à la fréquence de sortie désirée. Il suffira de changer cette valeur chaque fois que l'on changera de fréquence.

DAC + filtre de reconstruction

Ils constituent une fonction analogique commandée numériquement au rythme de l'horloge.
Le théorème de Shannon nous dit que la fréquence ne pourra pas dépasser la moitié de celle de l'horloge et nous aurons un problème de repliements qu'il faudra éliminer par filtrage. Du fait de l'impossibilité d'avoir un filtre à réponse fréquentielle "carrée", on ne pourra pas obtenir une fréquence maxi égale à Fe/2. En pratique on se contentera de Fe/3, au mieux Fe/2,5 avec un filtre elliptique. Voir la problématique des repliements sur la figure 2 pour une fréquence d'horloge de 100 MHz et une fréquence de sortie de 20 MHz

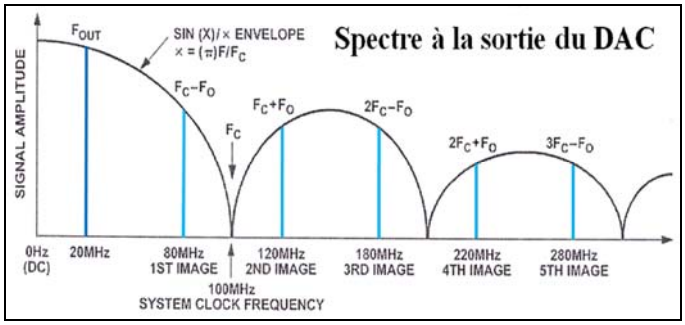


Figure 2 "**Repliements de spectre**"

Générateur d'horloge

L'horloge détermine les instants où la valeur changera à la sortie du DAC. Si l'horloge a du "jitter" (variations temporelles des transitions), cela se traduira par une modulation de phase (bruit de phase) du signal analogique reconstruit, d'où l'importance de la qualité spectrale de l'horloge. C'est aussi elle qui détermine la précision et la stabilité de la fréquence. Attention aux oscillateurs d'horloge bon marché prévus pour le numérique. Cela semble être le cas pour celui du kit F1BBU, mais pas de dégradation apparente de la qualité du FT-707. Pour une application "générateur HF", il ne faudrait pas lésiner sur ce composant. Dans tous les cas, bien soigner le filtrage de son alimentation.

Le DDS AD9851

C'est l'un des premiers composants DDS fabriqués par Analog-Devices. Il comprend un **NCO** et un **DAC 10 bits** avec sortie en courant. La fréquence d'horloge peut monter à **180 MHz** et être fournie, soit directement, soit à $\mathbf{F_{CLK}/6}$ avec un multiplicateur interne de l'horloge par six. La pleine échelle du courant de sortie est proportionnelle à une tension de référence, ce qui permet de faire éventuellement une modulation d'amplitude avec un signal analogique (ou de l'ASK avec un DAC en interface).
L'interfaçage numérique avec le contrôleur s'effectue, soit par un bus série 2 fils, soit par un bus parallèle 8 bits, plus un fil de synchronisation (FQ_UD = prise en compte des données).
La valeur de **Δφ** a une résolution de 32 bits, soit 0,04 Hz pour une horloge à 180 MHz. On peut introduire un offset de phase avec une résolution de 5 bits, ce qui permet de faire une modulation de phase (ou de fréquence). On peut ainsi obtenir un signal modulé en AM, φM, FM, ASK, FSK, MSK, PSK, nQAM, etc.
L'AD9851 a aussi des circuits et des fonctions annexes que l'on n'utilise pas dans notre application de générateur sinusoïdal. Nous avons sur la figure 3 notre schéma de mise en œuvre de celui-ci.

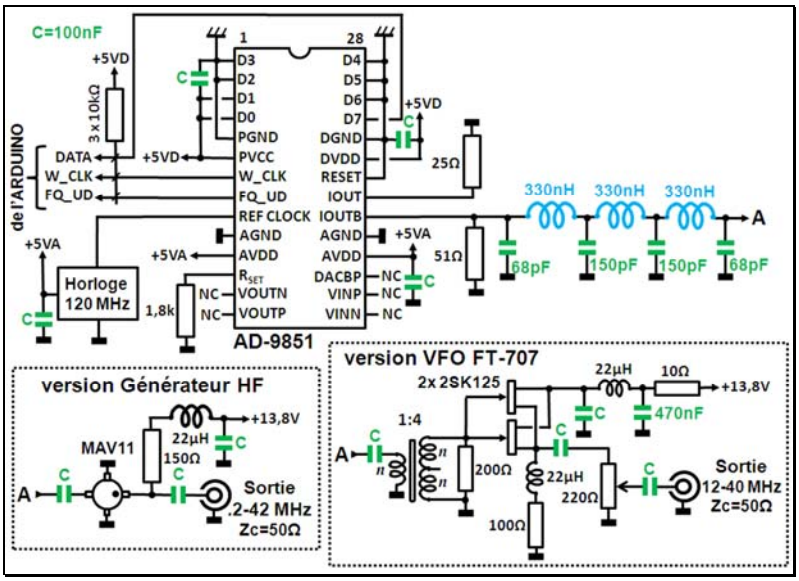


Figure 3 "**Schéma de la platine DDS**"

La charge du filtre anti repliements étant de 50 Ω et le DAC étant une source de courant, il faut disposer une autre résistance de 50 Ω en parallèle sur l'entrée. Ainsi l'impédance de charge dans la bande est de 25 Ω. On équilibre les courants de sortie en mettant une 25 Ω en charge sur la sortie IOUT inutilisée. Pour conserver la tension de sortie, la résistance Rset

passe à 1,8 kΩ (3,9 kΩ nominale). Dans la version VFO, la bande étant restreinte, pas de problème pour le transfo 1:4 à tore. Etant chargé au secondaire par 200 Ω, il ramène 50 Ω pour la charge du filtre. Dans la version "géné HF", le bas de bande est limité par les condensateurs de liaison et le haut de bande par le filtre anti repliements.

Le module Arduino uno

L'Arduino est un système "Open Source" utilisant les contrôleurs ATmega_xx8 d'Atmel qui a été développé pour le prototypage de systèmes électroniques. Il est simple, flexible, et facile à utiliser, tant d'un point de vue matériel que logiciel.
L'Arduino uno comporte un micro contrôleur "ATmega_328" avec ses composants accessoires (horloge, circuit d'alimentation, Reset) et un circuit spécialisé de conversion port série / port USB qui permet de charger un programme résident depuis un P.C. Les entrées / sorties du micro contrôleur sont reliées directement à deux connecteurs de 8 et 10 broches en ligne d'un côté et de l'autre à deux connecteurs de 6 et 8 broches en ligne également. On peut lui superposer une, voire plusieurs cartes "filles" spécialisées dans des applications particulières. Ici, nous utilisons le module Arduino uno seul. Celui-ci ne serait qu'une sorte de carte d'évaluation, comme il en existe pour tous les microcontrôleurs, sans une importante documentation logicielle constituant une bibliothèque de fonctions et d'API [(1)] qui s'enrichit constamment grâce à la contribution des utilisateurs. Il constitue ainsi un standard de fait.
Voir sur la figure 4 le synoptique et la photo du module "Arduino uno".

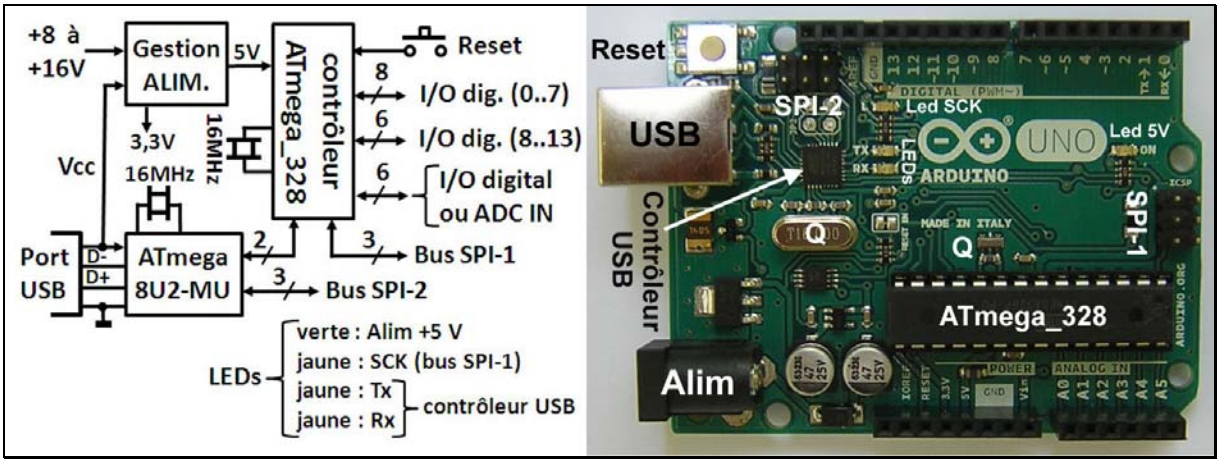


Figure 4 " **Synoptique et photo de l'Arduino uno**"

On peut récupérer le schéma détaillé sur le site "Arduino.org". Il n'a rien d'original. Il permet d'accéder à toutes les fonctionnalités de l'ATmega_328. Ce n'est plus ensuite qu'un problème d'interfaçage et de logiciel.

Association de l'Arduino avec le DDS

Ici l'Arduino et ses périphériques constituent ce qu'on appelle une "IHM" (**I**nterface **H**omme-**M**achine). En entrée nous avons les boutons poussoirs et la roue codeuse et en sortie nous avons l'afficheur LCD et le DDS. Voir le schéma sur la figure 5.

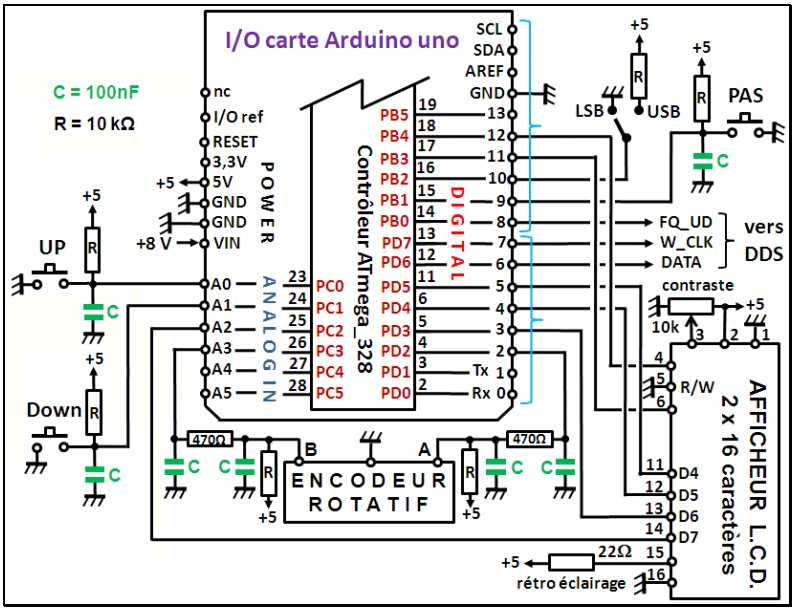


Figure 5 : "**Schéma de l'interfaçage Aduino - DDS**"

Celui-ci n'a rien de compliqué, vue la simplicité des fonctions. On notera les cellules intégratrices "anti-rebond" sur les sorties de la roue codeuse. Etant plus électronicien que programmeur, j'ai préféré régler le problème d'anti-rebond de cette manière.

L'encodeur rotatif

Les encodeurs rotatifs fonctionnent tous sur le même principe : On utilise un disque crénelé de ***n*** créneaux symétriques. Par exemple **30**. Ainsi chaque créneau occupera **12°** (360/30) partagé en deux parties "ON" et "OFF" (ou "plein" et "trou") de **6°**. On dispose deux capteurs (peuvent être de simples contacts ou des photo-coupleurs) décalés de **(*m*×6°) + 3°**. Alors quand on tournera la roue codeuse, leurs réponses seront celles **A** et **B** de la figure 6 [(2)].

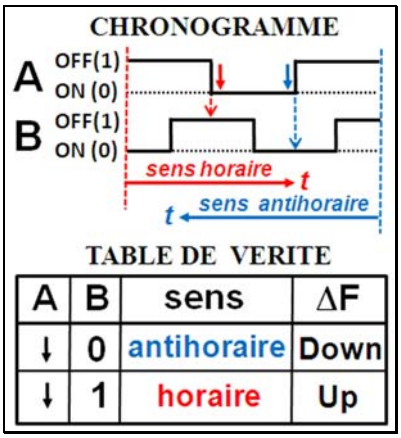


Figure 6 : "**Chronogramme et table de vérité de l'encodeur rotatif**".

On utilise une sortie (ici la "A") pour échantillonner sur une transition la deuxième sortie (ici la "B"). Alors la lecture de cette dernière donne le sens, horaire ou antihoraire, comme dans l'exemple de la fig. 6 qui représente notre cas. Dans celui-ci, on se sert d'une transition négative de A pour déclencher une interruption. La routine d'interruption lit la sortie B et la mémorise dans une variable après avoir positionné un "flag" [(3)] pour signaler la rotation. Voir plus loin l'organigramme logiciel.

Réalisation (coffret)

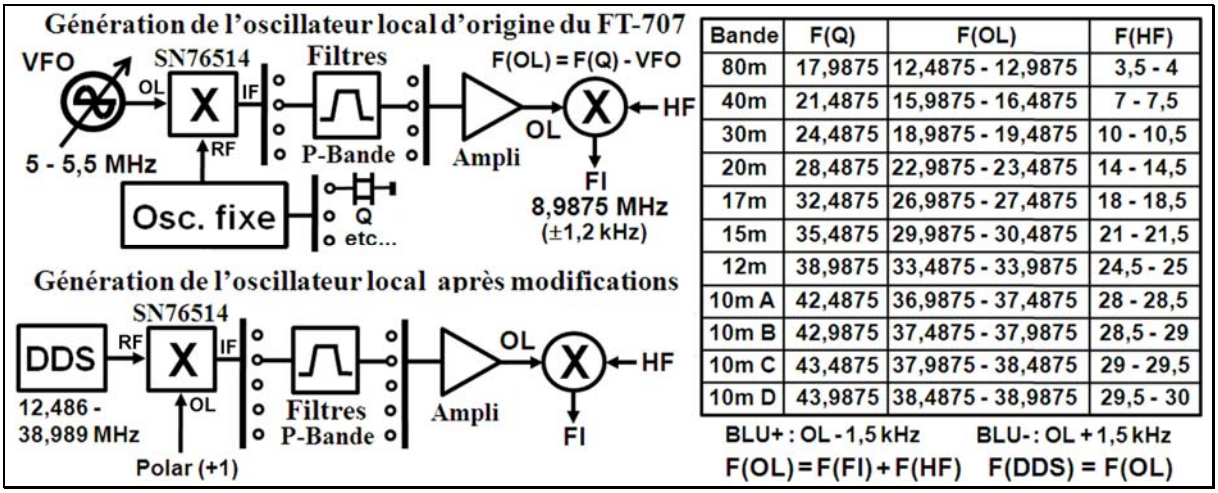
J'ai récupéré un coffret sensiblement de la même taille que ma boîte d'accord antenne qui me permet de loger confortablement tout le matériel. N'ayant pas les connecteurs adéquats pour l'Arduino, j'ai utilisé des fils de câblage rigides qui font office de broches. Ceci entraîne un câblage un peu "brouillon", mais cela fonctionne. Le matériel n'est pas prévu pour être testé à la table de vibration (Hi). Voir sur la figure 7 la face avant du coffret et la disposition des éléments à l'intérieur.

Figure 7. "**Le coffret VFO à DDS (sur ma boîte d'accord antenne) et l'intérieur**"

La carte DDS est celle du kit de F1BBU. Si vous la réalisez entièrement, penser que nous avons affaire à de la HF. Prendre ses précautions pour séparer les parties digitales et analogiques. Pour cela, s'inspirer de la carte d'évaluation de l'AD9851 (data-sheet d'Analog-Devices).

Modifications du VFO du FT-707

A l'origine le FT-707 utilise un VFO unique allant de 5 à 5,5 MHz pour toutes les bandes. On obtient la bonne fréquence du signal OL en soustrayant la sortie du VFO d'un oscillateur à quartz (un par bande) par mélange dans un circuit intégré SN76514. Sa sortie à $F_{(OL)}$ est d'abord filtrée (un filtre par bande) puis amplifiée et appliquée à l'entrée d'un mélangeur équilibré à diodes recevant par ailleurs le signal HF après filtrage.
La fréquence affichée par le FT-707 est la sortie d'un fréquencemètre mesurant la fréquence de l'OL avec un offset négatif égal à la fréquence FI ±1,5 KHz selon que l'on se trouve en mode lsb ou usb (la fréquence du VFO ne change pas, c'est le quartz "porteuse FI" qui est commuté). J'ai recopié le même principe de décalage dans mon programme et les affichages se retrouvent bien synchronisés (pour le même mode).
La figure 8 montre le synoptique de la partie "génération OL" ainsi que le plan des fréquences.



Bande	F(Q)	F(OL)	F(HF)
80m	17,9875	12,4875 - 12,9875	3,5 - 4
40m	21,4875	15,9875 - 16,4875	7 - 7,5
30m	24,4875	18,9875 - 19,4875	10 - 10,5
20m	28,4875	22,9875 - 23,4875	14 - 14,5
17m	32,4875	26,9875 - 27,4875	18 - 18,5
15m	35,4875	29,9875 - 30,4875	21 - 21,5
12m	38,9875	33,4875 - 33,9875	24,5 - 25
10m A	42,4875	36,9875 - 37,4875	28 - 28,5
10m B	42,9875	37,4875 - 37,9875	28,5 - 29
10m C	43,4875	37,9875 - 38,4875	29 - 29,5
10m D	43,9875	38,4875 - 38,9875	29,5 - 30

Figure 8 : "**Plan de fréquences du FT-707 et modification de l'OL**"

La modification consiste à remplacer l'oscillateur à quartz par le DDS et le VFO 5 - 5,5 MHz par une constante égale à 1 (polar continue). Ainsi le mélangeur SN76514 se transforme en amplificateur et la fréquence du DDS doit être celle de la fréquence OL du FT-707. Les modifications de la carte RF sont indiquées en rouge sur la figure 9.

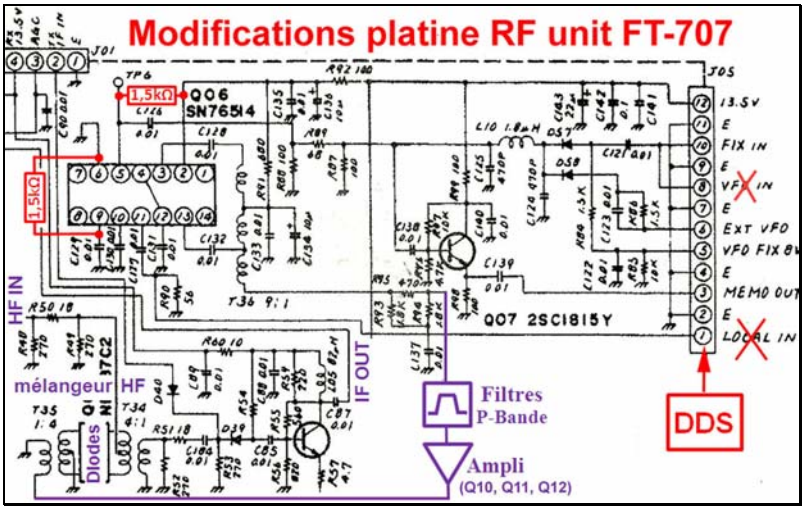


Figure 9 : "**Modification de la carte RF du FT-707**"

On commence par supprimer les entrées "VFO IN" (VFO 5 - 5,5 MHz) en broche 8 de J05 et "LOCAL IN" (oscillateur à quartz) en broche 1 de J05. On injectera le DDS sur cette broche.
Pour transformer le mélangeur Q06 SN76514 en amplificateur, il suffit de connecter une résistance de 1,5 kΩ entre sa broche 5 (LOC OSC INPUT) et le +VCC (2) et une autre entre la broche 9 (DECOUPLE 1) et la masse (6). Le signal VFO DDS ainsi amplifié passe par le transfo T36 et arrive aux filtres de bandes OL pour être traité comme l'OL d'origine.
Il faut faire attention au niveau du VFO DDS appliqué à la broche 1 de J05. J'ai réglé ce niveau à 40 mV càc, ce qui donne 280 mV càc sur le connecteur J06 du fréquencemètre et 1V càc sur le collecteur de Q12 (2SC2407). Au dessus, il y a écrêtage du signal sur Q12.
La modification ne nécessite pas obligatoirement de retourner la platine RF Unit, j'ai pu souder les deux résistances de 1,5 kΩ côté composants. Noter que l'appareil peut facilement être remis dans son état d'origine.
Le niveau plancher du bruit en réception ne semble pas avoir été modifié et pas d'apparition de signaux « fantômes ».

Pour l'émission, je ne dispose pas d'analyseur de spectre. J'ai juste pu contrôler le spectre d'émission au récepteur SDR et je n'ai constaté aucune différence avec le VFO d'origine. Quant au bruit de phase, je ne suis pas non plus équipé pour pouvoir le mesurer. Je n'ai pas constaté de différence avec le VFO d'origine. Il dépend essentiellement de la qualité de l'horloge. Celle-ci étant un oscillateur à quartz à 120 MHz (ce qui évite d'utiliser le multiplicateur par 6 du DDS), si son alimentation est bien filtrée, on doit obtenir les caractéristiques de l'AD 9851 qui sont largement suffisantes pour le FT-707 et très acceptables pour un générateur HF.
J'ai effectué une mesure de stabilité en fréquence à $F_{(DDS)}$ = 20 MHz. Du départ à froid et pendant les vingt premières minutes, le VFO DDS dérive de - 40 Hz. Ensuite aucune dérive constatée **(4)**.

Le logiciel

Le langage de programmation de l'Arduino dérive du langage C avec quelques syntaxes et fonctions spécifiques. Ce langage permet une approche "intuitive" de la programmation et la rend facilement accessible à un débutant. Il n'est pas question ici de reproduire le programme dans ses détails. Celui-ci peut être chargé depuis le site du radio-club F6KRK **(5)**. Nous nous contenterons de disserter sur les organigrammes de la figure 10.

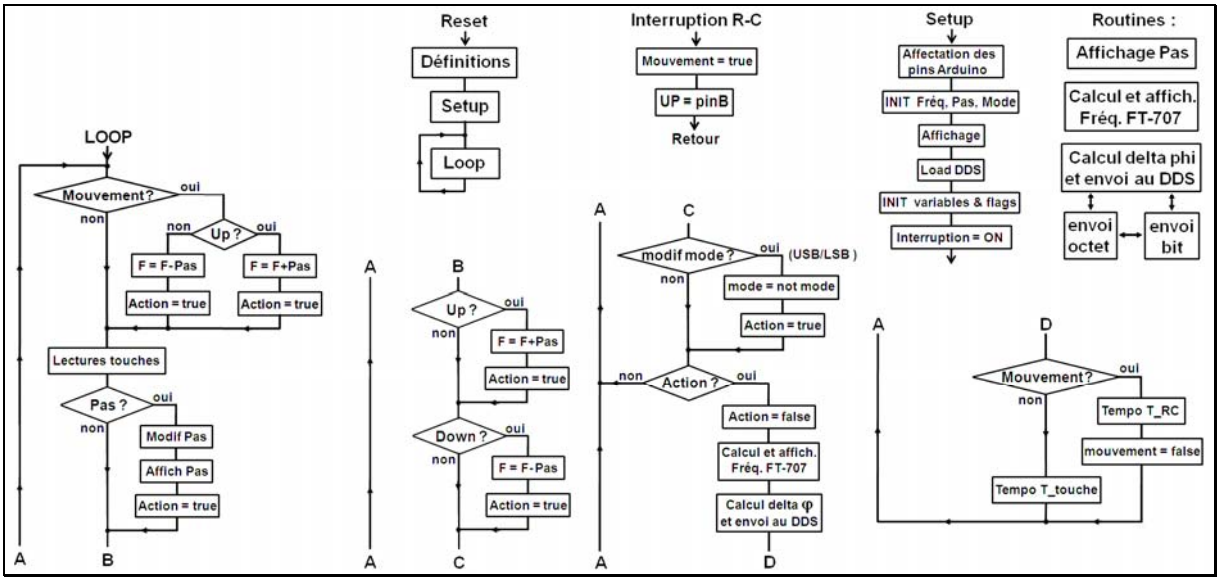


Figure 10 "**Organigrammes du logiciel**"

Le programme s'articule en quatre parties :

- Les **définitions** qui concernent l'interfaçage entre l'Arduino et ses périphériques d'une part et la bibliothèque d'API d'autre part.
- Le **setup** qui initialise le système.
- La **loop** qui constitue le programme tournant en boucle. Il fonctionne en mode "scrutation" (des touches), "analyse" et "action" (affichage et chargement DDS).
- Les **routines** qui sont appelées par le "**setup**" et la "**loop**", plus la routine d'**interruption** qui est appelée lors d'un mouvement de la roue codeuse.

Les **définitions** comportent les constantes modifiables utilisées pour les tempos, les butées de fréquence, etc. Elles comportent également les constantes déclarées se rapportant aux E/S de l'Arduino, ainsi que les variables globales (de divers types) qui seront utilisées par la suite.

Enfin, si besoin, on déclare ("include") les incorporations d'**API** (bibliothèque), par exemple celle concernant l'affichage LCD, en renseignant l'interfaçage (qui de l'Arduino correspond à quoi de l'API).

Le **setup** constitue le début du logiciel "actif". C'est ici que le programme démarre après un "**Reset**" (à la mise en route ou après appui sur la touche "Reset" de l'Arduino). On initialise les **I/O** de l'Arduino (entrée, sortie, ou fonction alternative). On effectue les opérations qui n'ont lieu qu'au démarrage (par ex. affichage informatif temporaire). On met le système en état d'initialisation en faisant appel aux routines adéquates avec les constantes prédéfinies (fréquence, pas, mode, etc.). On initialise les variables pour se mettre dans le même état qu'à la fin de la **loop** et on autorise les interruptions.

La **loop** constitue le cœur du programme. Elle remplit cinq fonctions principales :

a) **lecture** des entrées touches
b) **tests** des entrées touches plus la roue codeuse
c) si un ou plusieurs tests sont positifs, on effectue les **actions** correspondantes (mise à jour des variables et des flags, affichage et chargement DDS)
d) **temporisation** selon l'entrée, roue codeuse ou touche, jouant un rôle d'anti-rebond et de vitesse maxi de répétition (touche enfoncée en permanence ou roue codeuse lancée à grande vitesse).
e) **réinitialisation** des flags, et retour au début.

Les **tempos** sont à optimiser selon la roue codeuse et ses préférences. Dans mon cas, une tempo "**T_RC**" de 10 ms convient parfaitement pour ma roue codeuse, quelle que soit sa vitesse manuelle de rotation. Une tempo "**T_touche**" de 200 à 400 ms convient pour la répétition des touches enfoncées en permanence (pour ma part je préfère 200 ms).
Noter que s'il n'y a aucun évènement, la boucle ne comporte que les lectures de touches et cinq tests négatifs. Il n'y a aucune action ni aucune tempo et le tour se fait en une centaine de microsecondes.
Les **routines** n'appellent pas de commentaires particuliers. Elles utilisent, soit les fonctions de l'API "afficheur LCD", soit les fonctions avancées du langage Arduino. Quant à la **routine d'interruption**, elle se contente de positionner le flag "mouvement" et de noter le sens dans le flag "Up" (recopie de la pinB).
Le logiciel détaillé peut être téléchargé depuis le site du radio-club F6KRK [(5)]. Il utilise une programmation simple et facile à comprendre d'autant qu'il est abondamment commenté en français. On facilitera la lecture en s'aidant des synoptiques de la fig. 10.

Conclusions

Considérant la réalisation, je pense avoir rempli mon contrat. J'ai essayé de mettre en faute le logiciel en appuyant sur plusieurs touches, en tournant la roue codeuse en même temps, et il ne se plante pas. Pourtant c'est ma première expérience en programmation, mais j'avoue m'être fait aider. J'étais parti du programme de F5MNA en lui ajoutant une sorte de verrue logicielle concoctée à partir de l'épluchage des bibliothèques Arduino. Mais je n'étais pas totalement satisfait car ma roue codeuse semblait tourner trois à quatre fois moins vite que la mécanique. Ayant consulté un OM habitué à la programmation des microcontrôleurs, il m'a dit que j'avais pris le problème par le mauvais bout. Je devais d'abord établir un organigramme fonctionnel et seulement après, consulter les bibliothèques et ce qui avait déjà été fait dans le genre. Donc j'ai revu ma copie et ma roue codeuse est devenue un TGV.
Considérant l'article, j'avais réalisé une simple description de ma réalisation qui me satisfaisait. Puis, l'ayant relue à tête reposée, je me suis demandé qui cela aurait intéressé, à

part l'OM qui voudrait résoudre exactement le même problème. Or le sujet me semble moderne, intéressant et facilement transposable pour d'autres applications. Il ne restait plus qu'à rendre l'article didactique pour qu'il puisse servir à un maximum d'OM [(6)]. Le lecteur dira si j'ai réussi mon pari.

73 de F1HKJ.

Notes.

(1) Une API est un logiciel d'application avec une interface standardisée que l'on intègre dans son propre logiciel (directive "include").
(2) On remarquera que la méthode pour déterminer le sens est mathématiquement la même que celle utilisée pour séparer les bandes latérales dans un circuit "phasing" (utilisation de deux signaux en quadrature).
(3) un "flag" est une variable booléenne qui ne peut prendre que deux valeurs : "0" = "false" et "1" = "true". Par exemple, la routine d'interruption de la roue codeuse positionne le flag "mouvement". Si lors d'un test (if (mouvement)), il est "vrai" alors c'est qu'il y a eu rotation.
(4) La dérive est entièrement liée à l'oscillateur simple à quartz de l'horloge à 120 MHz. Si celui-ci était un TCXO ou mieux un OCXO, la dérive pourrait être de l'ordre de quelques Hertz.
(5) "www.blog.f6krk.org", catégories "Techniques / Constructions" et "Divers".
(6) Le meilleur prétexte pour approfondir ses connaissances sur un sujet est de vouloir les transmettre aux autres.