Création d’un émetteur RTTY à base d’Arduino (2ème partie)

Voici la deuxième partie de la série d'articles consacrés à la création d'un émetteur RTTY. Cette partie est concentrée autour de la réalisation physique de l'émetteur.

Une petite nouvelle avant de commencer.

Au départ je n'avais pas l'intention d'aller plus loin que cette partie, car la création de cet émetteur avait pour but de juste comprendre, avec un protocole simple, le mode digital de A-Z (encodage, modulation, émission, ...). Mais je me suis dit que tant qu'à faire, autant aller jusqu'au bout, c'est-à-dire, jusqu'à un modèle de “production” (PCB, ajout d'une antenne, ...). Donc il y aura encore une 3ème partie après celle-ci :-)

Le matériel :

Petit rappel sur le matériel utilisé :

- Un Arduino nano
- un émetteur NTX2B http://www.radiometrix.com/content/ntx2b
- Des résistances de 100k, 2x22k, 2×4.7k
- Une breadboard
- Des fils

Pour l'écoute, un pc avec fldigi correctement configuré.

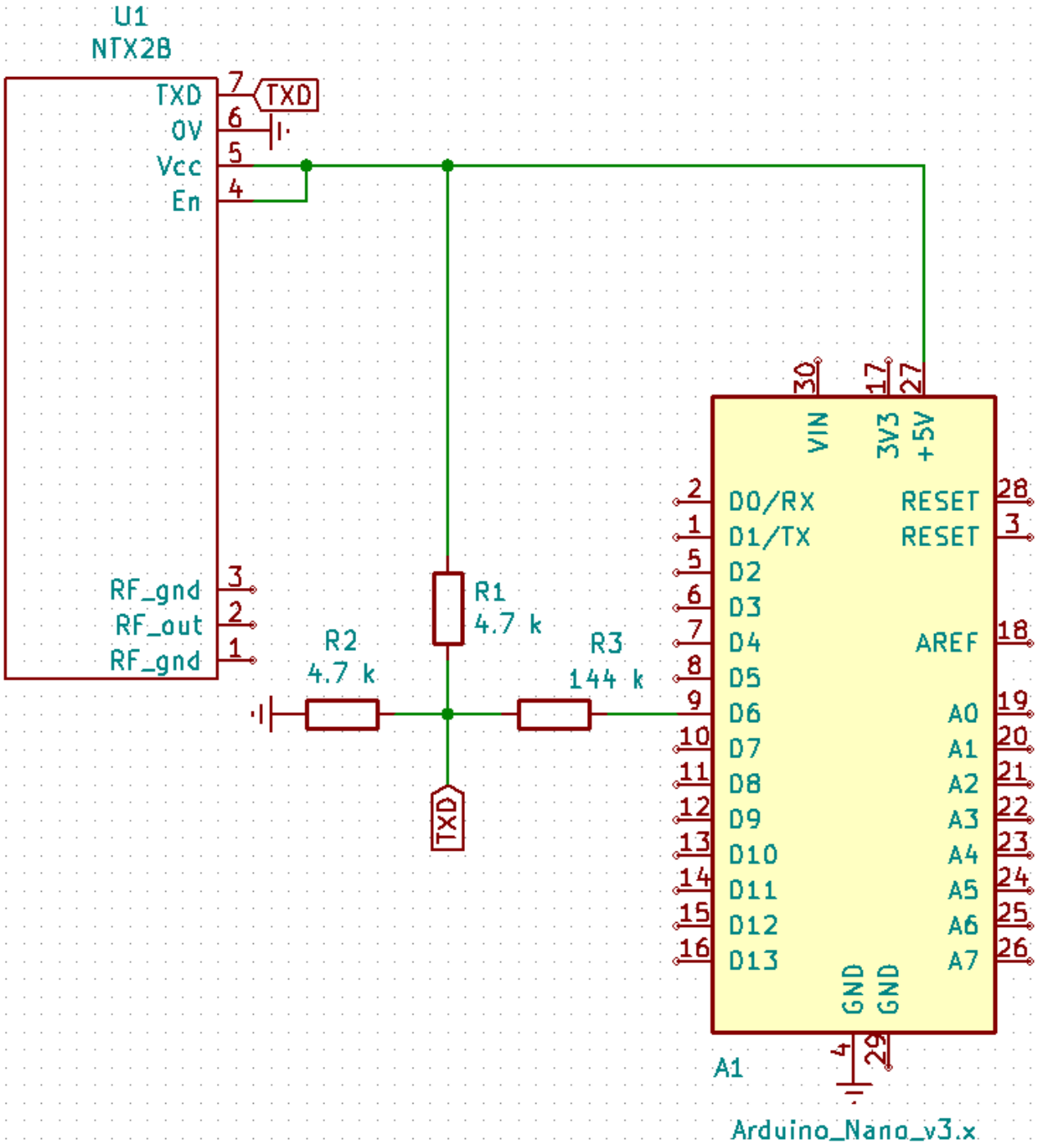
Commençons. Lorsque j'ai attaqué cette partie de la réalisation, deux possibilités s'offraient à moi pour utiliser le NTX2B avec un shift de 170 Hz, on les a déjà un peu vues dans la 1ère partie :

- Utiliser la fonctionnalité PWM de l'arduino.
- Utiliser un diviseur de tension.

La meilleure solution :

J'ai donc testé les deux. Celle que j'ai retenue et qui fonctionne très bien est la solution avec l'utilisation du diviseur de tension. Concernant la solution PWM, il y aura dans cet article une parenthèse à ce sujet.

Voici donc le schéma de la solution retenue :



La pin Vcc du NTX2B est connectée à la pin 5V de l'arduino afin d'alimenter l'émetteur. Ensuite la pin En est, elle aussi directement connectée à l'alimentation, car comme décrit dans la documentation (http://www.radiometrix.com/files/additional/NTX2B.pdf), elle permet d'activer notre NTX2B. La pin 0V est quant à elle toujours suivant la documentation reliée à la terre tout comme la pin 4 de l'arduino.

Maintenant regardons du côté de la pin 7 de l'émetteur, celle-ci nous permettant d'envoyer notre signal.
Rappelons nous que le NTX2B a + ou - un Shift de 6000 Hertz entre les deux fréquences maximales sur lesquelles il peut transmettre et qu'il attend sur sa pin TXD un voltage entre 0 et 3 V . Donc nous en avons déduit qu'un Shift de 1 Hertz équivaut à 3/6000, donc un pas de 0.0005 V. Et

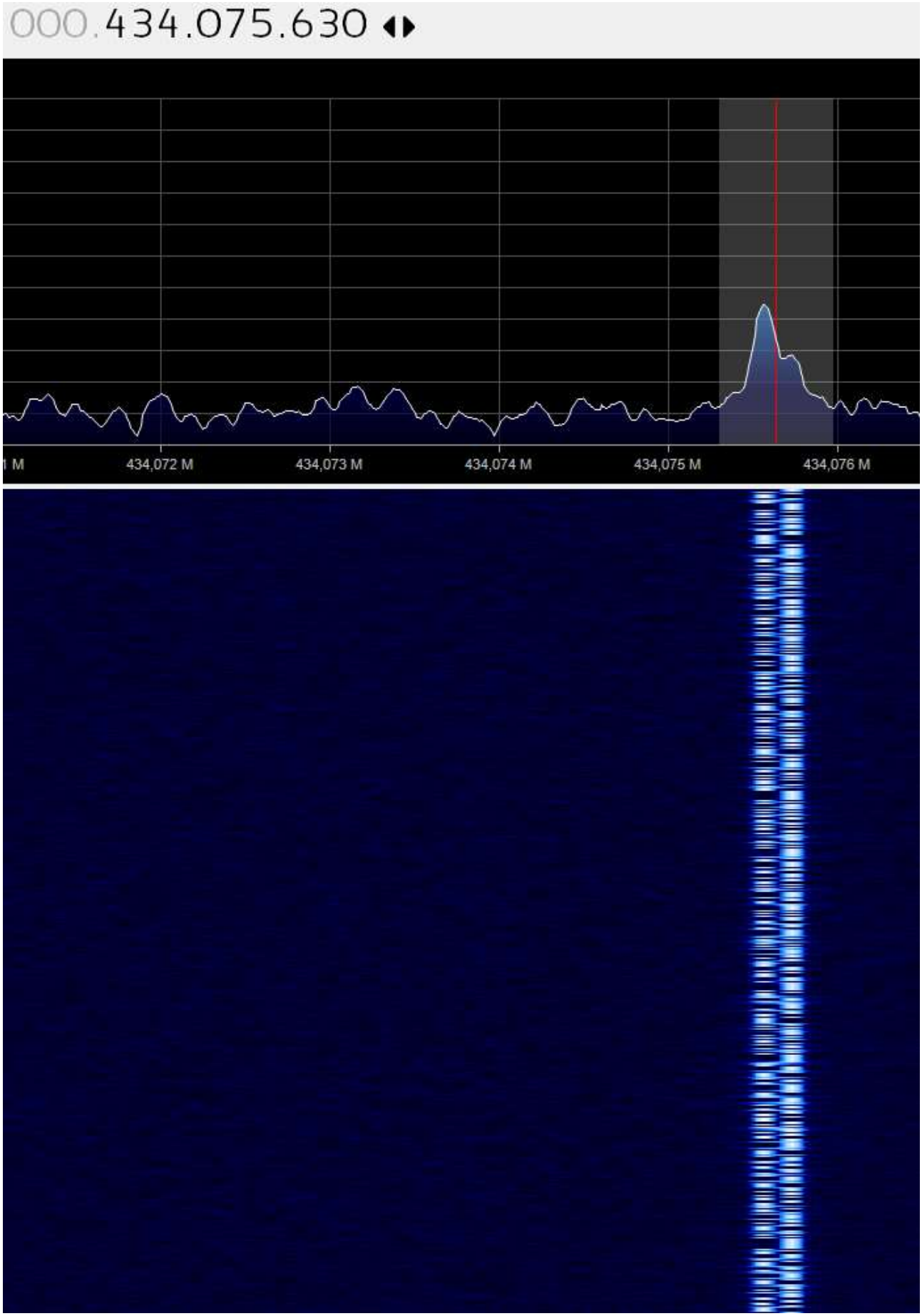
nous avions trouvé que pour avoir notre shift de 170 Hz nous devions avoir un pas de 0.085 V. Maintenant calculons les valeurs des résistances pour notre pont diviseur. Les deux résistances de 4.7 k Ohms (R2 et R1) on été choisies de manière complètement arbitraire et le fait qu'elles aient la même valeur n'est pas une coincidence, cela nous permet de simplifier notre calcul :-)

Comment trouver R3 maintenant ? Et bien il nous faut une valeur qui nous rapproche le plus possible de 0.085 V. Avec les résistances à ma disposition, le meilleur résultat était avec une résistance équivalente de 144 k Ohms. Voici le calcul :

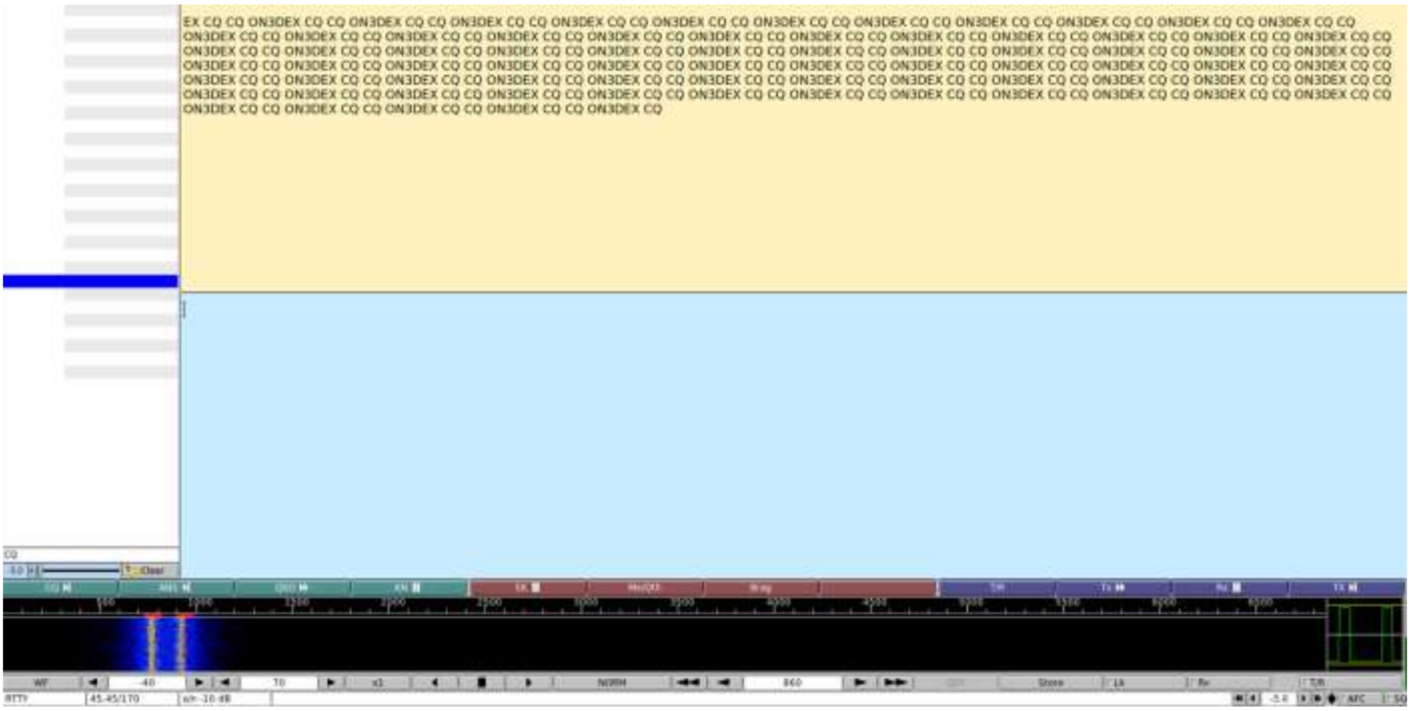
1) (R1 || R2) = 1 / (1/R1) + (1/R2) = 1 / (1/4700) + (1/4700) = 2350 Ohms
2) (R1 || R3) = (R2 || R3) = 1 / (1/R1) + (1/R3) = 1 / (1/144000) + (1/4700) = 4551 Ohms
3) Dans le cas d'une tension de 5V sur la pin D6 : R2 / (R2 + (R3 || R1)) x 5 = 2.54 V
4) Dans le cas d'une tension de 0V sur la pin D6 : (R3 || R2) / ((R3 || R2) + R1) x 5 = 2.45 V
5) Le pas est donc de 0.09 V (2.54 V – 2.45 V = 0.09 V) Ce n'est pas exactement 0.085 V, mais cela fera l'affaire.

Vous avez pu constater dans le schéma que la partie RF n'est pas cablée, c'est normal, les tests étant faits en “laboratoire” l'utilisation d'une antenne n'est pas nécessaire pour des tests à faible portée. Cette partie RF sera traitée dans le troisième article.

Une fois que tout est cablé comme il se doit, notre code uploadé sur l'arduino, ce dernier alimenté, nous avons notre émetteur RTTY :-)



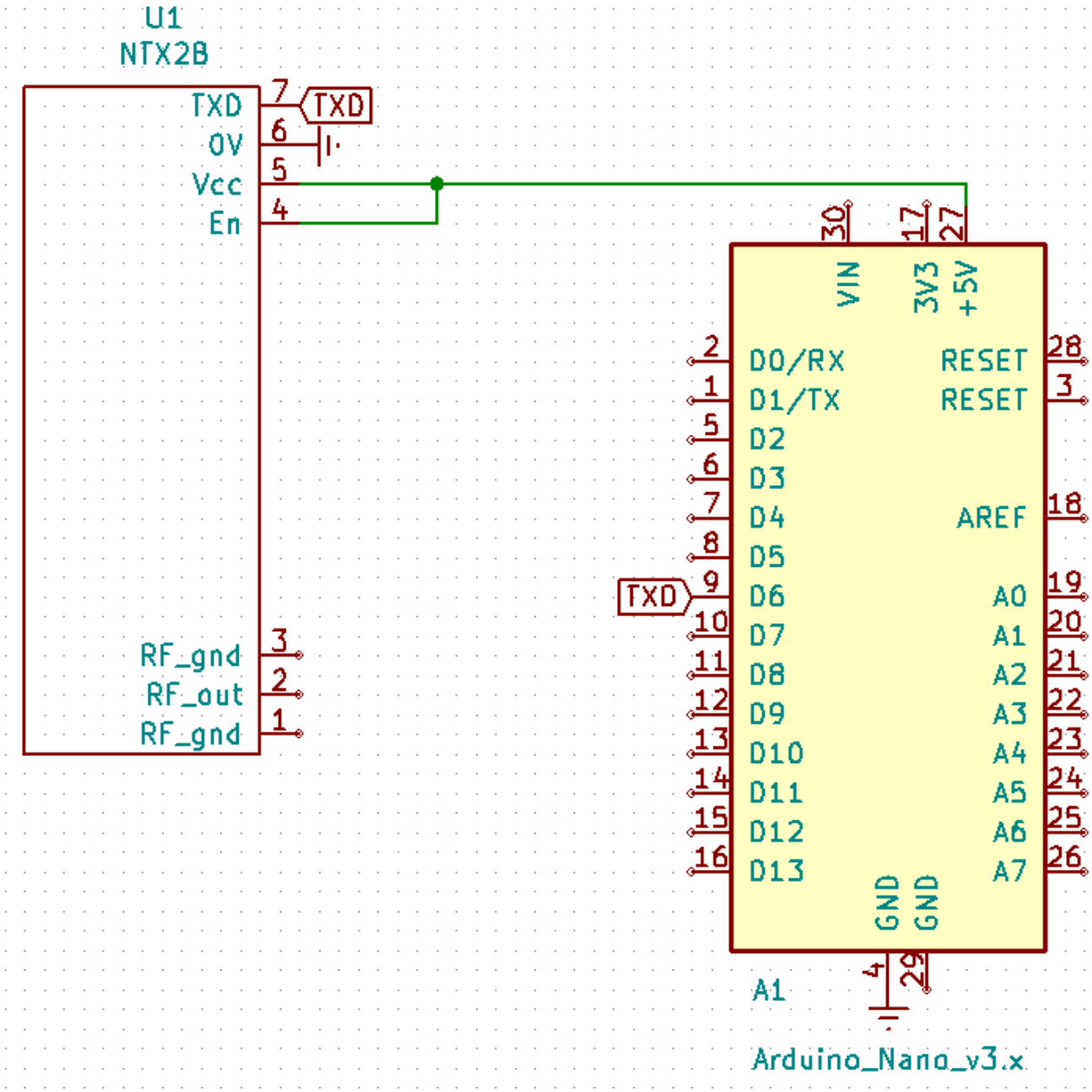
Notre signal réceptionné via un pc pas loin muni d'un lime SDR mini et de SDR console V3



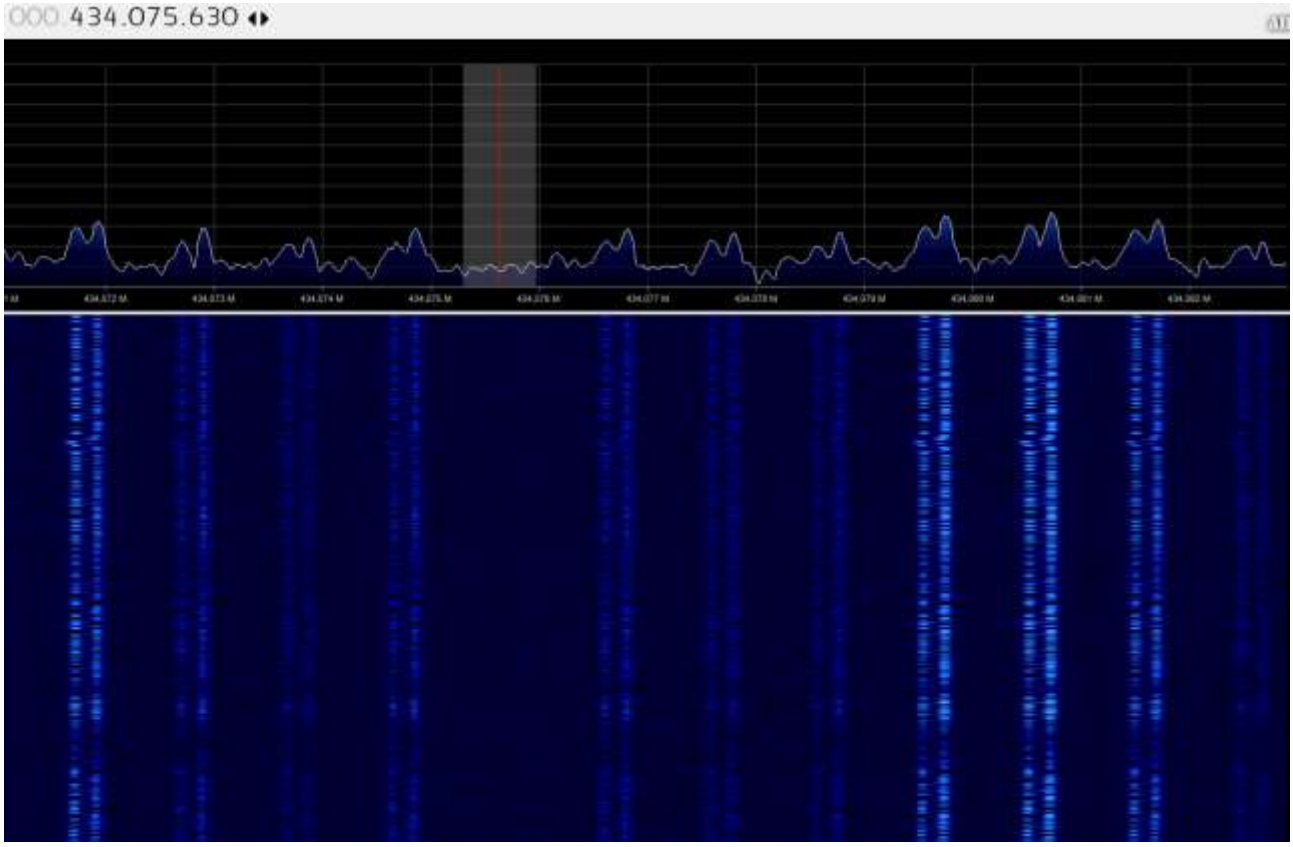
Quelques captures dans fldigi de la réception de notre signal (nous avons bien quasi un shift de 170 Hz).

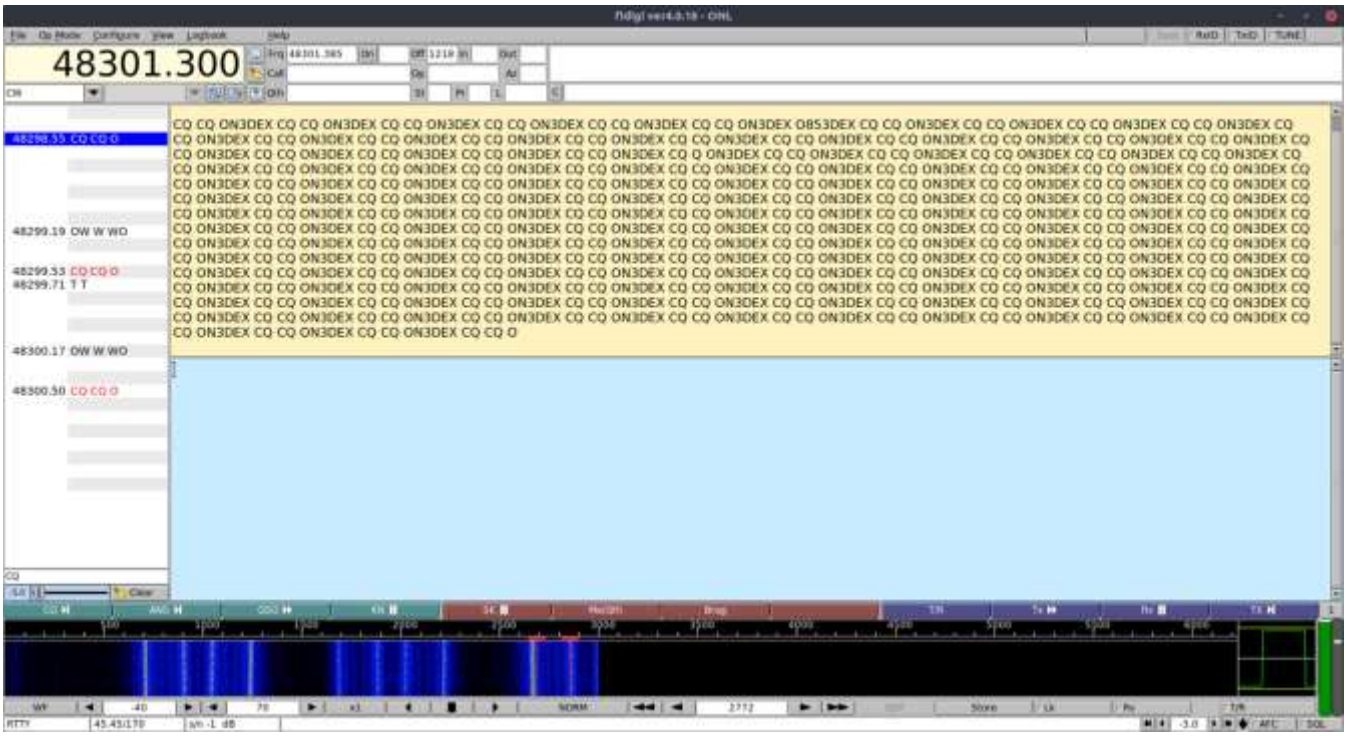
Le problème rencontré avec l'utilisation du PWM :

Avec l'utilisation du PWM de l'arduino, je me suis heurté à un problème dont je comprends la cause, mais pour lequel je n'ai pas encore trouvé la bonne solution pour avoir un bon résultat. Voici le schéma de câblage :



Ici l'alimentation et la terre ne changent pas. Simplement la pin D6 est maintenant directement cablée à la pin TXD de l'arduino (Le shift étant géré par la fonction PWM à l'aide de la méthode analogWrite). Voici le résultat :





Cela fonctionne, nous avons bien notre shift de 170 Hz, mais aussi de multiples signaux fantômes.

Pour “simuler” de l'analogique (j'entends par là, pouvoir délivrer une tension autre que 0 ou 5 V) la fonction PWM travaille par cycle. X % de front haut durant un cycle est égale à une tension de X V, par exemple 90 % du temps d'un cycle à 5V serait égal à une tension de 4.5 V en sortie (ceci est un exemple, les valeurs n'ont pas été calculées). Visiblement cela entraîne une instabilité au niveau de la tension de sortie sur la pin D6, les tensions ne sont pas stables, d'où les signaux fantomes.

J'ai testé plusieurs condensateurs de capacités différentes pour tenter de stabiliser la tension et en effet, cela réduit l'apparition de signaux fantomes, mais pas totalement et avec un effet très négatif que je vais décrire ci-après.

J'ai tout d'abord testé avec des condensateurs de faibles capacités (22 µF), cela réduisait la multiplication de signaux, mais vraiment pas de beaucoup. Plus j'augmentais la capacité du condensateur utilisé, plus les signaux parasites disparaisaient ou s'affaiblisaient (vers les 470 µF)), mais avec la mauvaise surprise de réduire le shift jusqu'à pratiquement ne plus avoir de distinction entre la fréquence des MARKS et celle des SPACES. Ce qui est logique étant donné que le condensateur est là pour “lisser”, stabiliser la tension, avec une grande capacité celui-ci par contre lisse beaucoup trop et notre pas de tension servant a définir le shift diminue.

Au moment d'écrire ces lignes, je n'ai pas trouvé la bonne solution pour complètement corriger le problème. J'ai aussi utilisé une méthode dans le code qui permet de définir la vitesse (fréquence) du PWM, sans grand succès jusqu'a présent car cela aussi ne corrige que partiellement l'apparition des signaux parasites. Plus d'informations ici :
https://www.arduino.cc/en/Tutorial/SecretsOfArduinoPWM

D'une manière générale, je trouve la méthode utilisant le diviseur de tension beaucoup plus élégante, robuste, simple et diablement éfficace, néanmoins si d'autres OM ont déjà eu le soucis ou ont d'autres pistes je suis bien sûr preneur.

Ceci est donc la fin de ce deuxième opus consacré à la création d'un émetteur RTTY. J'espère que cela vous aura intéressé.

Par Julien ON3DEX