Le module transceiver VHF DRA818V

Cet article est dans la continuité d'aider à débuter dans la « bidouille » avec ce module très séduisant. On le trouve tout monté par exemple sur le site de SV1AFN pour une prix très modique de la trentaine d'euros.

Mais reste à trouver avec lui un moyen de dialoguer avec son port série pour le configurer.

Mon souhait, toujours dans des configurations minimalistes, est d'utiliser de raspberry pi zero, à la place ou en complément des arduinos. Utiliser un Pi4 me semble bien du gaspillage ici. Mais que ce soit avec un pi zero ou un autre modèle de Pi, je me suis tout de suite planté sur l'utilisation des outils les plus évidents. Ainsi minicom ne fonctionne pas avec ce module. La raison est certainement qu'il ne gère pas correctement des octets série en temps réels du module DRA818V qui gère certainement sa liason série d'une manière très rudimentaire. Avec le pi zero, on reste en mode console avec la version légère de raspbian, c'est à dire sans mode graphique.

Sur Internet, je me suis aperçu bien sur que je n'étais pas le premier à m'être planté avec minicom. La solution est un script en python qui fonctionne.

Le but ici est de configurer ce module, rien d'autre.

Ce module se branche sans autre composant sur le port GPIO du raspberry. On prend soin de brancher l'alimentation du module sur la broche 1 du GPIO qui est en 3,3 V, et on branche de même en direct en croisé les fils Tx et Rx du module et du pizero. Une LED s'allume sur le module quand on met sous tension le raspberry.

1- Configurer une carte SD avec Raspbian:

A ce stade, si ce pi zero doit servir pour travailler avec ce module comme avec APRX ou tout autre utilisation en AX25, mieux vaut partir d'une image « neuve ». Je ne décris pas ici la manip trop bien connue.

Mais veiller à faire des mises à jours de raspbian, c'est quand même la chose minimale à faire.

2 – Installer python et ses librairies:

Ici, on installe python en 3.7, mais la version 2.7 s'installe aussi pour garantir une compatibilité des scripts plus anciens.

Surtout on a besoin de la librairie « python-serial », puisqu'il s'agit d'entrées-sorties séries. Elle est en version python 2.7 seulement.

Utiliser la commande : apt-get install python-serial

ou

aptitude install python-serial.

3- Le script en python:

```
#!/usr/bin/env python2.7import time
import serial
ser= serial.Serial(port='/dev/ttyAMA0´,
baudrate = 9600,
parity=serial.PARITY_NONE,
bytesize=serial.EIGHTBITS,
stopbits=serial.STOPBITS_ONE,
)
commande=raw_input(« Entrez votre commande: »)
print commande
ser.write(commande+ »\r\n »)
x=ser.readline()
print x
```

Faire un chmod +x de ce script pour le rendre exécutable.

Pour lancer le script, faire sous root donc avec sudo si l'utilisateur root n'est pas validé:

Allezr dans le répertoire où réside le script, et faire:

```
$ sudo ./nom_du_script
```

A l’exécution, on vous demande d’entrer la commande à envoyer au module. Par exemple si on tape:

```
AT+DMOCONNECT
```

le module répond:

```
+DMOCONNECT:0
```

S’il y a un problème, on obtient:

```
+DMOERROR
```

3 – Configurer le module;

Maintenant, on va configurer à minima le module. Pour cela, il faudra saisir dans la commande demandée:

```
AT+DMOSETGROUP=GBW,TFV, RFV,Tx_CTCSS,SQ,Rx_CTCSS
```

Ci-dessus, après le signe « = », ce sont des valeurs numériques qu’il faut entrer.

Par exemple on veut utiliser ce module en APRS donc la fréquence 144,800 Mhz

- GBW est la largeur de bande et des canaux. « 0 » configure en 12,5 khz, et 1 en 25 khz
- TFV est la fréquence en émission, attention, il faut 4 chiffres après le point;
- RFV de même pour la fréquence en réception;
- Tx_CTCSS est la féquence en hertz du signal CTCSS éventuel. la valeur 00000 enlève le CTCSS en émission;
- SQ est la valeur du squelch, la valeur « 0 » enlève le squelch, c'est à dire ouvre entièrement le récepteur;
- Rx_CTCSS est la veleur à recevoir du CTCSS, la valeur 00000 enlève la réception du CTCSS.

Donc la commande exacte à envoyer est:

AT+DMOSETUPGROUP=0,144.8000,144.8000,0000,4,0000

- « 0 » indique une bande passante de 12,5 khz;
- 144.8000 la valeur de la fréquence en émission et en réception;
- « 0000 » enlève l'émission du CTCSS, il faut 4 chiffres;
- « 4 » est la valeur du squelch;
- « 0000 » enlève le CTCSS en réception, il faut 4 chiffres.

On a ainsi configuré le module, la notice technique donne la manière du PTT etc.

Cette notice donne aussi la façon de faire varier le volume du

signal BF de sortie en réception.

C’est tout pour cet article qui n’a d’autre prétention de vous permettre de bricoler avec ce module pour l’usage que vous voudrez.

Remarque:

Ce script est très facilement adaptable aux modules TX et RX de chez Radiometrix, la parité chez ces bidules est toujours 8 bits, 1 ou 2 stop bits (une émission avec 2 stop bits est lue par une réception configurée avec 1 stop bits). Le changement de vitesse est évident à la lecture du script.

Il est vrai que les raspberry peuvent poser problème quand on veut dialoguer avec beaucoup de modules, python permet d’y remédier.